Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Григоряна Сергія Валерійовича*
(ПІБ)

академічної групи *122-20-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми Комп'ютерні науки
(назва освітньої програми)

на тему: *Розробка веб-платформи для онлайн консультацій з використанням PHP*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-3* (група) Григоряна Сергія Валерійовича (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-платформи для онлайн консультацій з використанням PHP*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***20.05.2024.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***03.06.2024 р.***

Завдання видав _____ *доц. Кабак Л.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Григорян С.В*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: N с., N рис., N дод., N джерел.

Об'єкт розробки: веб–орієнтований додаток платформи для онлайн консультацій.

Мета кваліфікаційної роботи: розробка веб–орієнтованої платформи, що забезпечить доступ до послуг, та обміну знаннями в різноманітних сферах діяльності.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку системи віртуального середовища, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленого застосунку, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні функціональної веб–платформи, яка може бути використана для надання консультаційних послуг в різноманітних предметних областях, та полегшить процес пошуку та взаємодії користувачів платформи.

Актуальність розробки програмного продукту полягає в зростаючому попиті на онлайн–консультації, які стають затребувані через зручність та доступність.

Список ключових слів: КОНСУЛЬАНТ, КОНСУЛЬТАЦІЯ, ПЛАТФОРМА, LARAVEL, БАЗА ДАНИХ, ФРЕЙМВОРК.

ABSTRACT

Explanatory note: N pages, N figures, N appendices, N sources.

The object of development is a web-oriented application platform for online consultations.

The purpose of the diploma project is to develop a web-oriented platform that provides access to services and facilitates knowledge exchange in various fields of activity.

The introduction covers the analysis and current state of the problem, specifies the purpose of the qualification work and its application area, substantiates the relevance of the topic, and clarifies the task set.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the problem statement is formulated, and the requirements for software implementation, technologies, and software are indicated.

In the second section, the available solutions are analyzed, a platform for development is selected, the design and development of a gaming application for interactive user skills development are carried out, the operation of the system, the algorithm and structure of its functioning, as well as the call and loading of the application, are described, the input and output data are determined, and the composition of the parameters of the technical means is characterized.

In the economic section, the labor intensity of the developed application is determined, the cost of application development work is calculated, and the time required for its creation is estimated.

The practical value lies in creating a functional web platform that can provide consultation services in various subject areas, facilitating the process of searching and interacting with platform users.

The relevance of the development is due to the growing demand for online consultations, which are becoming popular due to their convenience and accessibility.

List of keywords: CONSULTANT, CONSULTATION, PLATFORM, LARAVEL, DATABASE, FRAMEWORK.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ Програмне забезпечення;

SSL - Secure Sockets Layer;

TLS - Transport Layer Security;

MVC - Model-view-controller;

ORM - Object-relational mapping;

RMDBS - Relational Database Management System;

СКБД - Система керування базами даних;

HTTP - HyperText Transport Protocol;

IDS - Intrusion Detection System;

IPS - Intrusion Prevention System;

ЗМІСТ

РЕФЕРАТ 3
ABSTRACT 4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ 10
1.1. Загальні відомості з предметної галузі 10
1.1.1. Консультації як спосіб освіти в предметних областях в Україні 10
1.1.2. Аналіз існуючих програмних рішень 12
1.1.3. Опис і значущість онлайн- консультацій 15
1.2. Призначення розробки та галузь застосування 17
1.4. Постановка завдання 19
1.5. Вимоги до програми або програмного виробу 21
1.5.1. Вимоги до функціональних характеристик 21
1.5.2. Вимоги до інформаційної безпеки 23
1.5.3. Вимоги до складу та параметрів технічних засобів 25
1.5.4. Вимоги до інформаційної та програмної сумісності 26
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 28
2.1. Функціональне призначення інформаційної системи 28
2.2. Опис застосованих математичних методів 29
2.3. Опис використаних технологій та мов програмування 29
2.3.1. Фреймворк Laravel та мова PHP 29
2.3.2. Система об’єктно-реляційного відображення Eloquent 31
2.3.3. База даних MariaDB 32
2.3.4. Клієнт–серверна архітектура 33
2.3.5. Архітектурний шаблон MVC 35
2.3.6. Механізм шаблонів Blade 37
2.3.7. Технологія Bootstrap 38

2.4. Опис структури системи та алгоритмів її функціонування 39
2.4.1. Опис структури програмного додатку 39
2.4.2. Структура бази даних..... 41
2.4.3. Опис файлової структури 48
2.5. Обґрунтування та організація вхідних та вихідних даних програми 51
2.6. Опис розробленої системи 52
2.6.1. Використані технічні засоби 52
2.6.2. Використані програмні засоби..... 53
2.6.3. Виклик та завантаження програми..... 55
2.6.4. Опис інтерфейсу користувача 56
РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ..... 66
3.1. Розрахунок трудомісткості та вартості розробки програмного продукту 66
3.2. Розрахунок витрат на створення програми..... 70
ВИСНОВКИ..... 72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 73
ДОДАТОК А. Лістинг програми..... 77
ДОДАТОК Б. Відгук керівника економічного розділу 98
ДОДАТОК В. Перелік файлів на диску..... 99

ВСТУП

Онлайн-консультування наразі користується попитом. Людина може легко знайти професійного консультанта в мережі та пройти бесіду, не виходячи з дому. Вже існують численні веб-сайти, присвячені консорціумам консультантів, а також сотні сайтів, створених для надання різних консультаційних послуг. Онлайн-консультації охоплюють широкий спектр категорій, включаючи здоров’я, фінансові поради, освітні консультації, кар'єрний розвиток, юридичні поради та багато інших сфер.

В даній кваліфікаційній роботі розглядається платформа для надання консультативних послуг, а саме створення веб–сайту, яка націлена на надання можливості запису на різноманітні за тематикою консультації. Це стає доступним тім людям, що мають обмежену можливість для пересування, або для тих, хто живе в віддалених регіонах, а також поліпшують процес швидкості проведення необхідних бесід. Такі платформи дозволяють здійснювати ефективний обмін знаннями та підтримку в різних життєвих ситуаціях, що сприяє підвищенню якості життя людей.

Об’єктом дослідження є веб–орієнтований додаток платформи для онлайн-консультацій.

Метою кваліфікаційної роботи є розробка веб–орієнтованої платформи, що зможе надавати доступ до послуг у різних сферах діяльності, а також обмін знаннями. При цьому, користувач легко зможе знаходити та взаємодіяти з професійними консультантами.

Для досягнення поставленої мети необхідно:

– провести аналіз існуючих рішень, визначити їхні переваги та недоліки;

– визначити основні вимоги до функціональних можливостей, зокрема для серверної частини;

– розробити архітектуру веб–платформи, включаючи структуру бази даних, модулі користувача та консультанта;

– реалізувати інтерактивний інтерфейс користувача;

– забезпечити захист даних користувачів, використовуючи заходи безпеки, такі як SSL та шифрування даних;

– впровадити механізми для онлайн–бронювання консультацій, та управління графіками консультантів.

Актуальність полягає в зростаючому попиті на онлайн–консультації, які стають затребувані через зручність та доступність. Це особливо важливо для людей, які проживають у віддалених регіонах від місця проведення, або мають обмежену мобільність. Розробка такої платформи сприятиме забезпечення доступу до необхідних консультацій для широкого кола користувачів.

Практична значущість роботи – це створення функціональної веб–платформи, яка може бути використана для надання консультаційних послуг у різноманітних предметних областях. Дане рішення полегшить процес пошуку та взаємодії з консультантами, і забезпечить рівень безпеки конфіденційності інформації.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Консультації як спосіб освіти в предметних областях в Україні

Освіта стає не тільки засобом обміну знаннями та технологіями на міжнародному рівні, але й важливим елементом капіталу, інструментом конкуренції на ринку та засобом вирішення геополітичних завдань. Сучасне суспільство вимагає масової та високоякісної освіти, яка може задовільнити потреби як споживачів, так і виробників. Забезпечити виконання соціального замовлення суспільства шляхом збільшення фінансування освіти та кількості навчальних закладів, а також іншими традиційними методами, стає важким завданням навіть для багатих країн. Виникнення дистанційної освіти є логічним етапом розвитку освітньої системи, яка пристосовується до сучасних умов і вимог.

Консультації можна порівняти з дистанційною освітою, оскільки вона також представляє собою форму навчання в певних предметних областях. Як і дистанційна освіта, онлайн консультації надають можливість отримувати знання та навички без фізичної присутності в навчальному закладі. Вони забезпечують доступ до експертів та професійних консультантів, що дозволяє студентам та фахівцям гнучко планувати свій навчальний процес. Виходячи з цього, аналіз якості дистанційної освіти за останні роки може відобразити оцінку ефективності консультацій.

В Україні дистанційна освіта надзвичайно актуальна. Причина такої успішності криється в масовій перепідготовці і підготовці максимальної кількості фахівців по всій території України з використанням мінімальних коштів[1].

Покращена форма дистанційного навчання базується на використанні глобальних та локальних комп'ютерних мереж. Цей метод зберігає переваги традиційних форм навчання, водночас усуваючи їхні недоліки. Навчання через Інтернет забезпечує постійний зв'язок та активний обмін інформацією між студентом і викладачем, незважаючи на те, що їх можуть розділяти великі відстані. Однією з головних переваг такого навчання є його гнучкість, яка дозволяє студенту самостійно планувати своє навчання, не перериваючи роботу або не покидаючи свого місця проживання.

Розглядаючи консультації як спосіб дистанційної освіти, можна зробити перелік переваг, що відрізняють її від звичайних методів навчання:

По-перше, вона створює новий освітній простір, який може бути глобальним, національним, регіональним, міським або локальним. Це дозволяє студентам займатися у зручний для них час, в комфортному місці та у власному темпі, без жорстких обмежень у часі на засвоєння дисципліни.

По-друге, дистанційна освіта дозволяє поєднувати навчання з професійною діяльністю, не перериваючи робочий процес. Студенти мають можливість використовувати численні джерела навчальної інформації, такі як електронні бібліотеки, бази даних та знань. Спілкування між студентами та викладачами відбувається через Інтернет та електронну пошту, що сприяє ефективному обміну інформацією.

Подання навчального матеріалу в сконцентрованій формі та доступність його в будь-який момент підвищує ефективність засвоєння знань.

Використання сучасних інформаційних та телекомунікаційних технологій не тільки покращує процес навчання, але й навчає студентів працювати з цими технологіями. Дистанційна освіта надає рівні можливості для всіх, незалежно від місця проживання, стану здоров’я, соціального статусу чи фінансового становища студента.

Крім того, дистанційне навчання допомагає подолати психологічні бар’єри, такі як сором’язливість або страх публічних виступів, що може бути перешкодою в традиційному навчанні. Також, дистанційна освіта сприяє

обміну та впровадженню світових досягнень у сфері освітніх послуг, дозволяючи студентам і викладачам користуватися найкращими практиками та знаннями з усього світу.

Але існують і недоліки такої форми навчання, які включають відсутність безпосереднього контакту між фахівцем. Дивлячись на таку форму освіти в контексті надання консультацій та послуг, це буде зручним та ефективним способом отримання інформації, виходячи з актуальності використання даної форми в Україні.

1.1.2. Аналіз існуючих програмних рішень

В якості існуючих веб–орієнтованих рішень було розглянуто платформи Malt Strategy [2] та Catalant [3]. В цілому, мета даних додатків полягає в наданні професійних послуг. Головні сторінки цих сайтів зображено на рис. 1.1. – 1.2.

Маркетинговий консультант – це фахівець, який надає експертні поради та рекомендації компаніям щодо створення та впровадження дієвих маркетингових стратегій. Вони можуть працювати самостійно або бути частиною консалтингової фірми, зазвичай маючи досвід у таких сферах, як ринкові дослідження, брендинг, реклама, цифровий маркетинг і продажі.

В випадку продуктів, що розглянуто, на стартових сторінках наявні пропозиції про пошук консультанта, а також створення особистого акаунту. Це привертає увагу користувачів, що хочуть замовити послугу на детальне вивчення сайту.

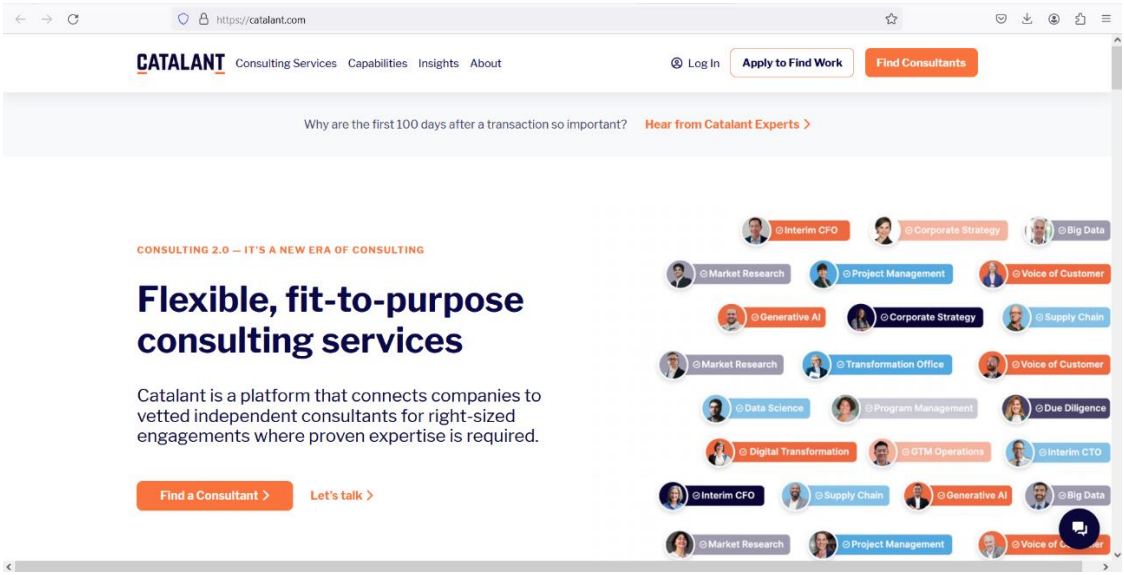

Рис. 1.1 – Стартова сторінка «Catalant»

Рис. 1.2 – Стартова сторінка «Malt»

Як можна побачити, продукти мають спільні риси в дизайні. Вони включають в себе:

– обидві платформи мають простий і зрозумілий інтерфейс, що дозволяє легко орієнтуватися як замовникам, так і консультантам;

– навігація по сайту логічна, з чіткими категоріями та фільтрами для пошуку потрібних фахівців;

– платформи мають систему відгуків та рейтингів, яка допомагає оцінювати якість роботи консультантів на основі досвіду попередніх клієнтів (рис 1.3);

– обидві платформи інтегрують засоби для комунікації між клієнтами та консультантами, такі як внутрішні повідомлення, що полегшує обговорення проектів та вимог.

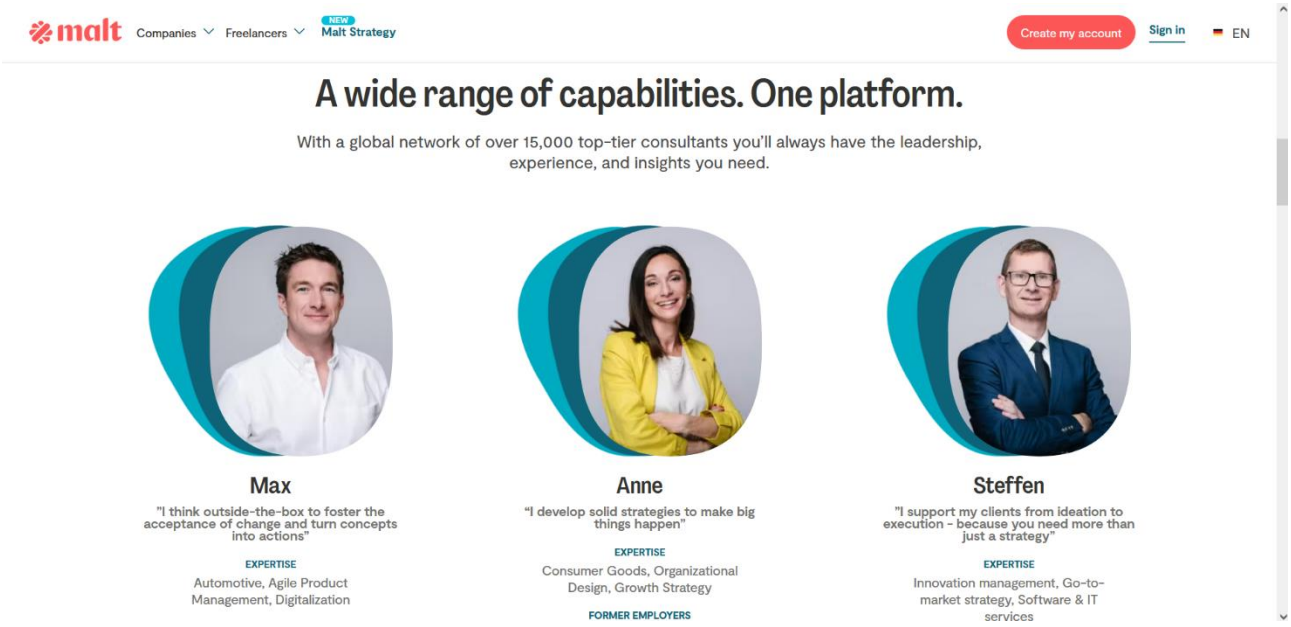

Рис. 1.3 – Відгуки «Malt»

Виходячи з аналізу обох платформ, слід зазначити, що можливо зробити покращення, а саме:

– додати список консультацій – як список товарів та відповідного консультанта;

– створити можливість перемикання режиму консультанта та звичайного користувача;

– реалізувати можливість створення нових консультацій в профілі для кожного;

– створити календар записів консультацій та вільних місць.

Таким чином, буде досягнута концепція інтернет–магазину у вигляді платформи для консультацій, де товар – це консультація, а продавець – експерт.

Дана реалізація сприятиме спрощенню інтерфейсу користувача, а також більшій конкретизації списку запропонованих послуг.

1.1.3. Опис і значущість онлайн-консультацій

Онлайн-консалтинг – це динамічна галузь, яка активно розвивається та використовує цифрові платформи для надання експертних консультацій, порад і рішень як для окремих осіб, так і для компаній. Це віртуальний аналог традиційного консалтингу, який завдяки Інтернету надає гнучкість і доступність.

Онлайн-консультування, також відоме як електронне або віртуальне консультування, включає надання професійних послуг і порад через цифрові канали. Ця сфера охоплює широкий спектр експертизи, зокрема бізнес-консультування, маркетингові стратегії, фінансове планування, кар'єрне консультування тощо.

Консультант – це фахівець, який надає поради в певній галузі. Вони пропонують свої послуги компаніям і людям, надаючи цінність завдяки своєму глибокому досвіду. Єдина різниця між онлайн-консалтингом і традиційним полягає в тому, що онлайн-консультант виконує ті ж функції, що і звичайний консультант, але використовує для цього цифрові технології.

Часто консультування плутають з тренінгом, але ці два види діяльності є досить різними. Тренинг передбачає задавання запитань, щоб допомогти людині самостійно знайти відповіді на свої проблеми, тоді як консультування полягає в наданні готової інформації та рекомендацій.

Онлайн-консультування є широким терміном, який охоплює багато різних напрямків роботи. Попит на такі послуги постійно зростає, оскільки вони необхідні ринку. Сьогодні стати консультантом у цій галузі може будь-хто, хто має достатні знання в певній темі.

Онлайн-консультант відіграє ключову роль у наданні експертних порад, рішень та рекомендацій клієнтам через цифрові канали. Обсяг їхньої роботи

може суттєво відрізнятися залежно від спеціалізації, але основні аспекти діяльності залишаються схожими.

Консультанти аналізують потреби, проблеми та цілі клієнтів, щоб зрозуміти контекст їхньої ситуації. Вони надають обґрунтовані висновки та поради, базуючись на своєму досвіді в певній галузі, будь то бізнес, маркетинг, технології, фінанси чи інша сфера. Консультанти допомагають клієнтам розробляти стратегічні плани для досягнення їхніх цілей, будь то покращення бізнес-процесів, підвищення видимості в Інтернеті чи покращення загальної ефективності. Вони також підтримують впровадження рекомендованих стратегій, надаючи покрокові інструкції для забезпечення ефективного виконання плану. У процесі реалізації вони вирішують виникаючі проблеми та пропонують рішення для підтримання процесу на правильному шляху.

Спеціалісти використовують цифрові інструменти комунікації, такі як відеодзвінки, електронна пошта та миттєві повідомлення для спілкування з клієнтами, сприяючи продуктивній співпраці. Вони, також, аналізують дані для надання рекомендацій, заснованих на доказах, та вимірюванню успіху впроваджених стратегій. Консультанти створюють звіти для оновлення клієнтів про прогрес, ключові показники та необхідні корективи для оптимізації результатів. Однією з цілей консультантів є надання клієнтам знань та навичок, що дозволяють їм самостійно приймати обґрунтовані рішення.

Враховуючи динамічну природу багатьох галузей, онлайн-консультанти інвестують у безперервне навчання, щоб бути в курсі тенденцій індустрії, найкращих практик та нових технологій. Вони можуть допомагати клієнтам у покращенні їхньої онлайн-присутності, оптимізації веб-сайтів та розробці ефективних цифрових маркетингових стратегій. Консультанти часто займаються налагодженням професійних зв'язків, співпрацюють з іншими експертами та розширюють свою клієнтську базу, беручи участь у мережевих заходах.

1.2. Призначення розробки та галузь застосування

Як об'єкт впровадження розглядається веб-платформа для онлайн консультацій, яка являє собою маркетплейс, де користувачі можуть пропонувати свої консультаційні послуги та записуватися на консультації з будь-якої сфери життя. Ця платформа створена для того, щоб забезпечити зручний інтерфейс для взаємодії між консультантами та клієнтами, пропонуючи широкий спектр консультаційних послуг у різних галузях.

Призначення розробки полягає в створенні універсальної платформи, яка об'єднає консультантів і клієнтів, надаючи їм можливість швидко і легко знаходити один одного. Платформа повинна забезпечити ефективну систему для бронювання, управління графіком консультацій, обміну інформацією та проведення онлайн-зустрічей. Основна мета полягає у створенні зручного та надійного середовища для надання та отримання консультацій, яке відповідатиме сучасним вимогам користувачів.

Терміни, що були використані в цій роботі, та потрібні для розуміння структури системи:

– маркетплейс–платформа, де пропозиції консультаційних послуг зустрічаються з попитом на них, створюючи ринок для консультантів і клієнтів;

– онлайн-консультації–надання експертних порад або послуг через Інтернет за допомогою відеозв'язку, чату або електронної пошти;

– консультант–професіонал, який пропонує свої консультаційні послуги на платформі;

– клієнт–користувач, який шукає консультаційні послуги та записується на них через платформу;

– бронювання–процес запису клієнта на консультацію в зручний для нього час;

– онлайн-зустріч–віртуальна сесія між консультантом і клієнтом, що відбувається через відеозв'язок або інші засоби комунікації;

– профіль користувача–інформаційна сторінка, яка містить дані про консультанта або клієнта, включаючи їхню контактну інформацію, біографію та відгуки;

– відгуки–оцінки та коментарі клієнтів про якість отриманих консультаційних послуг, які допомагають іншим користувачам приймати рішення;

Причини виникнення необхідності розробки полягають в наступному:

– Обмеження традиційних методів консультування. Традиційні методи, що вимагають особистої присутності, часто є незручними та часозатратними як для консультантів, так і для клієнтів.

– Гнучкість для клієнтів та консультантів. Можливість планувати консультації у зручний час, не порушуючи робочий графік або особисті плани.

– Доступність для людей з обмеженими можливостями. Онлайн-консультації забезпечують доступ до послуг для людей з обмеженою мобільністю або тих, хто живе у віддалених регіонах.

– Потреба в інноваційних методах навчання та підтримки. Онлайн-платформи дозволяють ефективно поширювати знання та досвід, надаючи можливість безперервного навчання та професійного розвитку.

– Скорочення психологічних бар’єрів. Дистанційна форма консультування допомагає людям, які відчувають дискомфорт або сором'язливість під час особистих зустрічей, отримувати необхідну допомогу.

Області, в яких може застосовуватися система, включають широкий спектр галузей та видів діяльності. Це може бути медицина, психологія, фінансові консультації, юридичні послуги, освітні консультації, кар'єрне тренування, IT-консультації, маркетинг та різноманітні сфери діяльності. Платформа дозволяє об'єднати експертів з різних галузей та їхніх потенційних клієнтів, створюючи ефективну систему для надання професійних порад та допомоги.

1.4. Постановка завдання

Метою розробки веб-платформи для онлайн консультацій є створення ефективного, зручного та безпечного середовища для надання та отримання консультаційних послуг у різних сферах життя. Платформа повинна забезпечити можливість взаємодії між консультантами та клієнтами, пропонувати інструменти для планування та проведення консультацій, а також підтримувати функції обміну інформацією та забезпечення конфіденційності даних.

Призначення розробки полягає у створенні універсальної платформи, яка дозволяє консультантам пропонувати свої послуги, а клієнтам – зручно їх знаходити та записуватися на консультації. Платформа має охоплювати широкий спектр консультаційних послуг, включаючи медицину, психологію, фінанси, право, освіту та інші галузі. Основне завдання платформи – забезпечити високий рівень якості послуг та задоволення потреб користувачів.

Технічно-економічна сутність завдання полягає у створенні веб-платформи, яка поєднує сучасні технології для забезпечення високої продуктивності, доступності та безпеки. Платформа має використовувати мову програмування PHP для реалізації серверної частини, а також сучасні фреймворки та бібліотеки для забезпечення адаптивного дизайну та інтерактивного користувацького інтерфейсу. Економічна доцільність проекту полягає у зниженні витрат на організацію консультаційних послуг та підвищенні їх доступності для широкого кола користувачів.

Вихідна інформація включає результати консультацій, рекомендації від консультантів, графіки проведення консультацій, відгуки клієнтів та аналітичні дані про використання платформи. Ця інформація надається користувачам у вигляді звітів, повідомлень та інших документів, що можуть бути збережені або передані для подальшої обробки.

Вхідна інформація складається з даних, які вводять користувачі під час реєстрації, бронювання консультацій, заповнення анкет та форм зворотного зв’язку. Це включає особисті дані клієнтів і консультантів, деталі запитів на

консультації, інформацію про платежі та інші релевантні дані. Організація збору та передачі вхідної інформації повинна забезпечувати високу швидкість обробки та надійність даних, з використанням шифрування для захисту конфіденційності.

Розподіл функцій між персоналом і технічними засобами передбачає автоматизацію більшості процесів, таких як реєстрація користувачів, бронювання консультацій, управління графіками та обробка платежів. Персонал платформи відповідає за модерацію контенту, підтримку користувачів та вирішення технічних питань. Консультанти використовують платформу для управління своїми послугами та комунікації з клієнтами, тоді як технічні засоби забезпечують автоматичне зберігання, обробку та захист даних.

Для користувача необхідні бути наступні функції:

– можливість створювати обліковий запис, вводячи необхідну інформацію, таку як ім'я, електронну пошту, номер телефону та інші контактні дані. Після реєстрації користувач може створити та налаштувати свій профіль, додавши особисту інформацію, фотографію та біографічні дані;

– вхід до системи, використовуючи свої облікові дані (логін та пароль). Також є можливість відновлення паролю у випадку його втрати через електронну пошту;

– перегляд списку доступних консультаційних послуг, відфільтрований за категоріями, рейтингом, ціною та іншими параметрами. Вони можуть детально переглядати інформацію про кожного консультанта, включаючи його спеціалізацію, кваліфікацію, відгуки та рейтинг;

– вибір консультанта та запис на консультацію, вибравши зручний час та дату з доступного розкладу, редагування або скасування бронювання у разі потреби;

– Для проведення консультацій доступна можливість організації онлайн-зустрічей через відеозв'язок, чат або електронну пошту. Користувачі можуть брати участь у відеоконференціях, обмінюватися текстовими повідомленнями та прикріплювати документи;

– Відгуки та оцінки про отримані консультації, що допомагає іншим користувачам у виборі консультанта. Перегляд відгуків та рейтингів інших користувачів;

– Перегляд історії консультацій, включаючи деталі проведених зустрічей;

– Користувачі можуть редагувати інформацію у своєму профілі, змінювати налаштування облікового запису, оновлювати контактні дані та змінювати пароль;

– Для вирішення питань або отримання допомоги, користувачі можуть звертатися до служби підтримки через вбудовану систему обробки запитів або електронну пошту;

– Оплата консультацій через інтегровані платіжні системи, використовуючи різні методи оплати, такі як кредитні картки;

– Отримання доступу до навчальних матеріалів, вебінарів та інших ресурсів, які надаються платформою або окремими консультантами;

– Платформа також надає можливість користувачам створювати особисті консультації, і надавати свої послуги.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Виходячи з цілі платформи для консультацій, слід зазначити наступні функціональні характеристики, з якими веб-платформа сприятиме взаємодії між консультантами та клієнтами:

– Функції реєстрації та авторизації повинні дозволяти користувачам створювати облікові записи, вводячи особисту інформацію, таку як ім'я, електронна пошта та номер телефону. Авторизація має здійснюватися за

допомогою логіна та пароля, з можливістю відновлення пароля через електронну пошту;

– Система повинна надавати можливість перегляду доступних консультаційних послуг, відфільтрованих за категоріями. Користувачі повинні мати доступ до детальної інформації про кожного консультанта, включаючи його спеціалізацію, кваліфікацію, відгуки та рейтинг;

– Платформа повинна забезпечувати функції бронювання консультацій. Користувачі мають мати можливість обирати консультанта та записуватися на консультацію, вибираючи зручний час та дату з доступного розкладу. Система повинна дозволяти змінювати або скасовувати бронювання у разі потреби;

– Історія консультацій повинна зберігатися у профілі користувача. Користувачі мають мати можливість переглядати деталі проведених консультацій, отримані рекомендації та записи розмов.

– Платформа повинна забезпечувати функції редагування профілю. Користувачі мають мати можливість змінювати налаштування облікового запису, оновлювати контактні дані та змінювати пароль.

– Система підтримки користувачів повинна забезпечувати можливість звернення до служби підтримки через вбудовану систему обробки запитів або електронну пошту.

– Платіжні функції повинні дозволяти користувачам оплачувати консультації через інтегровані платіжні системи, використовуючи різні методи оплати.

– Вхідні дані повинні вводитися користувачами у діалоговому режимі через форми на веб-сайті. Вони включають особисту інформацію, деталі запитів на консультації та інформацію про платежі.

– Система має надавати можливості створення нових консультацій кожному користувачу.

1.5.2. Вимоги до інформаційної безпеки

Виходячи з аналізу джерел [4], інформаційна безпека охоплює інструменти та процеси, які організації використовують для захисту інформації. Це включає налаштування політик, які запобігають доступу неавторизованих осіб до бізнесової або особистої інформації. Безпека є зростаючою та динамічною галуззю, яка охоплює широкий спектр напрямів, від захисту мереж і інфраструктури до тестування та аудиту.

Інформаційна безпека захищає конфіденційну інформацію від несанкціонованих дій, включаючи перегляд, зміну, запис, а також будь-яке порушення або знищення. Її метою є забезпечення безпеки та конфіденційності критичних даних, таких як дані клієнтських акаунтів, фінансова інформація або інтелектуальна власність.

Наслідки інцидентів з безпекою включають крадіжку приватної інформації, підробку даних і їхнє видалення. Атаки можуть порушувати робочі процеси, завдавати шкоди репутації компанії та мати відчутну вартість.

Захист інформації платформи включає наступні пункти:

– Необхідно використовувати надійні механізми авторизації для обмеження доступу до функцій платформи тільки авторизованим користувачам.

– Шифрування даних має забезпечувати захист всіх даних, що передаються між користувачами та сервером, за допомогою SSL/TLS. Дані, що зберігаються на сервері, включаючи конфіденційну інформацію користувачів, також повинні бути зашифровані.

– Контроль доступу має передбачати чітке визначення ролей і прав доступу для користувачів платформи, зокрема для адміністраторів, консультантів та клієнтів. Потрібно регулярно проводити аудит прав доступу для виявлення та усунення зайвих або несанкціонованих доступів.

– Захист від атак включає впровадження механізмів для запобігання DDoS-атакам для забезпечення безперебійної роботи платформи. Платформа

повинна використовувати системи виявлення та запобігання вторгненням (IDS/IPS) для моніторингу та запобігання підозрілим активностям.

– Резервне копіювання повинно бути регулярним та автоматичним для забезпечення можливості відновлення інформації у разі збоїв або втрат. Резервні копії мають зберігатися у захищених місцях із використанням шифрування.

– Моніторинг та логування мають забезпечити постійне відстеження активності на платформі для виявлення підозрілих дій. Журнали подій повинні зберігатися для аналізу та розслідування інцидентів безпеки.

– Захист від шкідливого ПЗ має включати використання антивірусного програмного забезпечення для захисту серверів та клієнтських пристроїв. Регулярне оновлення антивірусних баз даних є необхідним для забезпечення захисту від нових загроз.

– Конфіденційність даних повинна бути забезпечена обмеженням доступу до конфіденційної інформації лише для тих осіб, яким це необхідно для виконання їхніх обов’язків. Використання технологій для захисту особистих даних користувачів має відповідати вимогам законодавства.

– Навчання та підвищення обізнаності повинні включати регулярні тренінги для персоналу з питань інформаційної безпеки. Користувачі мають бути інформовані про важливість дотримання правил безпеки, включаючи використання надійних паролів та обережність при обробці конфіденційної інформації.

– Відповідність нормативним вимогам передбачає забезпечення відповідності платформи законодавчим вимогам та стандартам у галузі інформаційної безпеки. Необхідно проводити регулярні аудити та перевірки для підтвердження відповідності нормативним вимогам.

1.5.3. Вимоги до складу та параметрів технічних засобів

Платформа для онлайн-консультацій вимагає технічних засобів, які забезпечать її надійне функціонування. Ці засоби охоплюють серверне обладнання, мережеву інфраструктуру, клієнтські пристрої та інші компоненти.

Серверна частина складатиметься з двох серверів: сервер самого застосунку і сервер бази даних. Для серверу застосунку виставляються наступні умови:

- ОП не менше 16 GB RAM;
- SSD з об’ємом пам’яті не менше 512 GB для операційної системи та додатковий SSD або HDD з об’ємом від 1 TB для резервних копій;
- ЦП класу Intel Core i5 або еквівалентний (тактова частота не менше 2.5 ГГц, кількість ядер не менше 4);
- стабільне підключення до Інтернету зі швидкістю не менше 1 Gbps;
- використання RAID масивів для підвищення надійності даних;
- операційна система на основі Linux, рекомендовано використовувати Ubuntu Server LTS або CentOS з підтримкою регулярних оновлень безпеки.

Сервер бази даних також потребуватиме наступні параметри:

- ОП не менше 32 GB RAM;
- SSD накопичувачі з об'ємом від 1 TB;
- ЦП класу Intel Xeon або еквівалентний (тактова частота не менше 2.5 ГГц, кількість ядер не менше 4);
- стабільне підключення до Інтернету зі швидкістю не менше 1 Gbps;
- використання RAID масивів для підвищення надійності та швидкості доступу до даних;
- операційна система на основі Linux, рекомендовано використовувати Ubuntu Server LTS або CentOS;
- система управління базами даних на основі MySQL.

Комп'ютери користувачів повинні відповідати наступним вимогам:

– Мінімум 2-ядерний процесор (Intel Core i3 або еквівалентний AMD);
– 4 GB RAM;
– 256 GB SSD;
– Операційна система не нижче Windows 10, macOS 10.13
– Браузери повинні підтримувати сучасні веб-стандарти.

Мобільні пристрої повинні відповідати наступним вимогам:

– Мінімум 4-ядерний процесор;
– 2 GB RAM;
– Операційна система iOS 12 або новіше, Android 8.0 або новіше;
– Браузери повинні підтримувати сучасні веб-стандарти.

Безпека даних повинна бути забезпечена багаторівневим захистом, включаючи шифрування, аутентифікацію та авторизацію користувачі.

1.5.4. Вимоги до інформаційної та програмної сумісності

Платформа повинна підтримувати основні операційні системи для серверів, такі як Linux (рекомендовано Ubuntu Server LTS або CentOS) та Windows Server. Це забезпечить гнучкість у виборі серверного програмного забезпечення.

– На стороні клієнтів платформа повинна коректно працювати на основних операційних системах: Windows (версія 10 і вище), macOS (версія 10.13 і вище) та сучасні дистрибутиви Linux.
– Платформа повинна підтримувати сучасні веб-браузери, такі як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Це забезпечить доступ до платформи для широкого кола користувачів.
– Необхідно забезпечити коректне відображення та функціонування всіх елементів інтерфейсу на різних браузерах.
– Платформа повинна підтримувати роботу на мобільних пристроях з операційними системами iOS (версія 12 і вище) та Android (версія 8.0 і вище).

– Інтерфейс користувача повинен бути адаптивним, забезпечуючи зручне використання на екранах різних розмірів.

– Система повинна забезпечувати сумісність з базами даних, такими як MySQL. Це забезпечить ефективне зберігання та обробку даних.

– Забезпечення безперебійної роботи платформи вимагає підтримки резервного копіювання та відновлення даних, що дозволить зберегти інформацію у разі збоїв.

– Система повинна підтримувати багатомовність, що дозволить використовувати платформу користувачам з різних країн, забезпечуючи локалізацію інтерфейсу та контенту.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення інформаційної системи

Функціональне призначення веб–орієнтованого застосунку включає наступні пункти:

- реалізувати перегляд наявних консультацій за категоріями, зокрема інформації про доступні записи;
- надати можливість вибору дати запису на доступні пропозиції;
- створити панель керування та створення консультацій;
- забезпечити гнучкість архітектури та додавання нових компонентів до системи;
- виконати сумісність програмного забезпечення з актуальними операційними системами;
- надати систему контролю записами, зокрема редагування існуючих пропозицій;
- використання модульної архітектури для легкого інтегрування нових функцій та компонентів.

Перелік реалізованих функцій в програмі містить наступні пункти:

- організація журналу записів за датами та часом, відображення інформації про позиції за датою та можливість відміни;
- Перевірка доступності консультантів у реальному часі та відображення можливих варіантів для запису
- перегляд інформації про користувачів – консультантів;
- реєстрація користувачів;
- аутентифікація та авторизація в систему;
- створення на перегляд особистих консультацій та запис на існуючі;

Експлуатаційне призначення застосунку полягає автоматизації ручної праці, що зменшує кількість помилок, покращуючи якість обслуговування клієнтів через забезпечення доступу до інформації про консультації та їх записи в режимі реального часу.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв’язування задачі не передбачають застосування математичних методів, при розробці веб–орієнтованого застосунку платформи онлайн консультацій математичні методи не використовуються.

2.3. Опис використаних технологій та мов програмування

2.3.1. Фреймворк Laravel та мова PHP

Розроблена система використовує фреймворк Laravel [5], що є інструментом на мові програмування PHP. Laravel підтримує модульну структуру, яка дозволяє розширювати функціональність за допомогою пакетів і бібліотек, керованих через Composer [6]. Це забезпечує гнучкість та масштабованість системи, дозволюючи інтегрувати нові компоненти.

Веб-фреймворк— це набір бібліотек і інструментів, які покращують дизайн веб-додатків, додаючи додаткові функції та чіткість у розробці. Фреймворк також дозволяє автоматизувати завдання, оскільки він інтегрує низку підпрограм, реалізованих нативно, і, таким чином, його використання створює узгоджену архітектуру.

Аналізуючи електронні джерела [7], можна стверджувати, що мова програмування PHP є однією з найпоширеніших у використанні серед конкурентних мов (рис. 2.1). PHP широко застосовується для серверного програмування і відома своєю здатністю спрощувати розробку веб-додатків.

Фреймворки на різних мовах, зокрема на PHP, базуються на архітектурному шаблоні Model-View-Controller (MVC) [8] із доповненнями, такими як помічники форм та контролери баз даних. Під час розробки з використанням простого PHP бізнес-логіка змішується із запитами до бази даних, що ускладнює обслуговування та масштабованість програми. Для вирішення цієї проблеми PHP пропонує інструменти, зокрема Laravel. Фреймворки надають базову модель, а також повний набір API, бібліотек і розширень. Це підвищує продуктивність і зменшує кількість повторюваного коду, а також знижує складність архітектури, підвищує гнучкість коду та полегшує його повторне використання.

Одним з основних переваг використання PHP є його висока сумісність з веб-серверами та операційними системами, а також велика кількість вбудованих функцій для роботи з базами даних, обробки запитів та генерування динамічного контенту.

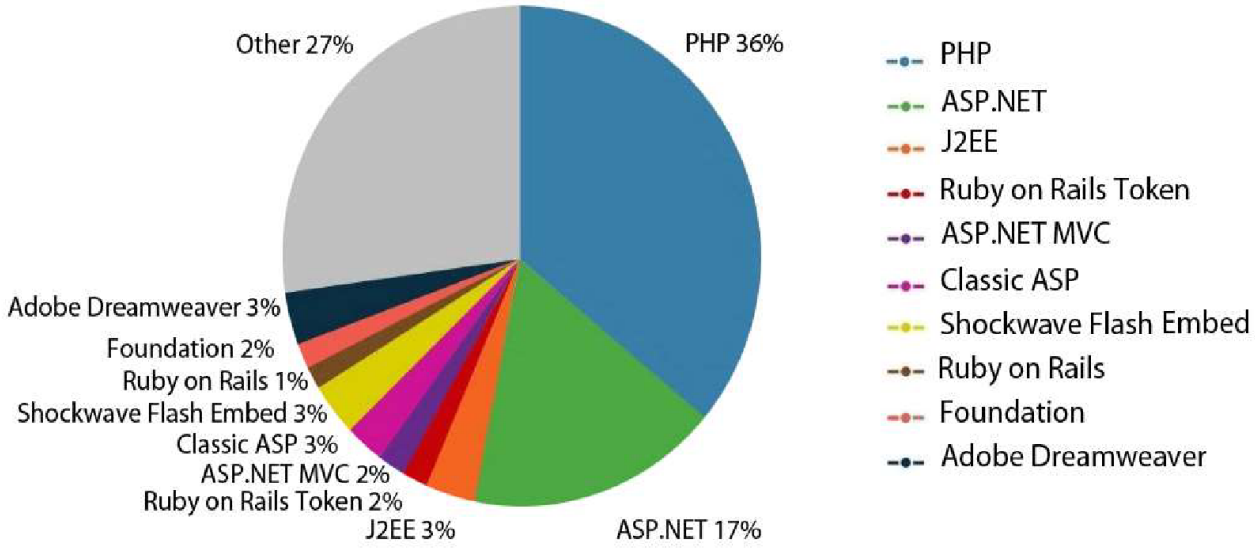

Рис. 2.1. Статистика використання фреймворків з вибіркою 1 млн. сайтів

Фреймворк Laravel використовує понад 20 різних бібліотек та розділений на окремі модулі. Він тісно інтегрований з Composer Dependency Manager, та створений для полегшення тестування та поставляється з кількома помічниками, які дозволяють відвідувати маршрути з тестів, сканувати HTML, гарантувати

виклик методів у певних класах і навіть імітувати автентифікованих користувачів. Laravel надає гнучкість у визначенні маршрутів, дозволяючи прив’язувати прості анонімні функції до маршрутів за допомогою HTTP-дієслів (GET, POST, PUT, DELETE) [9] та приєднувати функції фільтра, що виконуються на певних маршрутах. Крім того, фреймворк дозволяє керувати конфігурацією в залежності від середовища, в якому працює додаток, автоматично обираючи параметри для кожного середовища. Він також постачається з плавним конструктором запитів, який дозволяє виконувати запити до бази даних із синтаксисом PHP, і надає реалізацію ORM [10] і ActiveRecord [11] через Eloquent [12], спрощуючи визначення взаємопов'язаних моделей. Крім того, Laravel дозволяє визначати схему бази даних, відстежувати зміни за допомогою міграцій [13] бази даних і заповнювати таблиці.

2.3.2. Система об’єктно-реляційного відображення Eloquent

Eloquent ORM постачається з Laravel, та використовується в даній кваліфікаційній роботі для керування базою даних, і взаємодії з бізнес логікою. Eloquent використовує техніку програмування об’єктно-реляційного відображення, яка полегшує взаємодію з базою даних шляхом визначення власних моделей і зв’язків без написання запитів SQL, при цьому застосовуючи реалізацію з Active Record.

Шаблон Active Record є підходом для доступу до даних у базі даних. Це дозволяє реляційним базам даних бути представленими в об’єктно-орієнтованому коді. Кожен клас моделі, створений у структурі MVC, відповідає таблиці в базі даних, екземпляр об’єкта моделі прив’язаний до одного рядка в таблиці, а атрибути моделі прив’язані до полів таблиці. Після створення та збереження об’єкта, до таблиці додається новий рядок. Коли об’єкт завантажується, він отримує інформацію з бази даних, а при оновленні відповідний рядок у таблиці також оновлюється [14].

Об’єктно-реляційне відображення (ORM) — це метод відображення об’єктно-орієнтованої моделі домену в реляційну базу даних. На рис. 2.2 описано робочий процес Eloquent ORM, коли код програми містить будь-яку операцію з базою даних, код належним чином перетворюється на оператори SQL Laravel за лаштунками, які потім виконуються на механізмі бази даних. Результат виконаного оператора SQL надсилається повністю назад до програми.

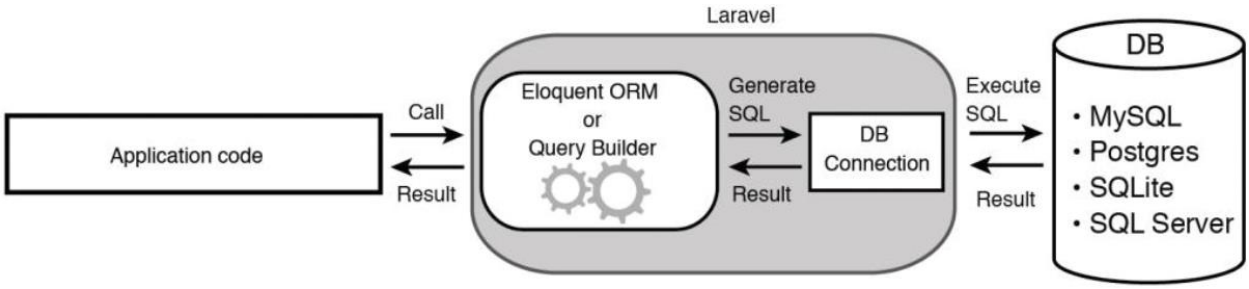

Рис. 2.2. Робочий процес ORM

Метою використання Eloquent ORM є усунення невідповідності між об’єктною моделлю та реляційною базою даних. Невідповідність виникає через те, як дані зберігаються та представлені в СКБД (система керування реляційною базою даних) та об’єктно-орієнтованими мовами. СКБД, наприклад MySQL [15], зберігає дані у вигляді табличного формату. Об'єктно-орієнтовані мови програмування, такі як Java [16] і PHP, представляють дані у вигляді об'єктів. Під час зберігання та завантаження об’єктів за допомогою реляційної бази даних виникають такі проблеми невідповідності.

2.3.3. База даних MariaDB

MariaDB [17] є системою управління реляційними базами даних (RDBMS), і є відкритим вихідним кодом, випущений з GNU [18]. Ця база даних є розгалуженням MySQL. Як і інші системи керування Користувач може спілкуватися з MariaDB, створюючи оператори на SQL. Вирази можуть бути

складені програмою, яка може надсилати їх до бази даних. Це дозволяє програмам будь-якого типу, включаючи веб-додатки, взаємодіяти з MariaDB для керування своїми даними.

Аналіз електронних джерел показує, що MariaDB базується на відкритому коді MySQL. Це розгалуження або розгалуження вихідного коду. MaraiDB має нові функції, покращення продуктивності, краще тестування та виправлення помилок, яких немає в MySQL. Деякі з них були розроблені основними розробниками MariaDB, а інші надходять від сторонніх окремих учасників і компаній, таких як Facebook, Twitter, Google [19].

MariaDB має низку покращень продуктивності, зокрема вдосконалений оптимізатор, який працює швидше за MySQL у складних навантаженнях. Оптимізатор перетворює SQL-команди на інструкції для бази даних. Реплікація також покращена, зокрема через групову фіксацію для двійкового журналу, що значно прискорює операції з великою кількістю оновлень. Функція Table Elimination [20] дозволяє обробляти запити без звернення до деяких таблиць, що знижує час виконання. У MariaDB є багато інших вдосконалень продуктивності.

Таким чином, дана реляційна база даних була обрана для зберігання та використання даних, безпосередньо, використовуючи Laravel.

2.3.4. Клієнт–серверна архітектура

Сервер — це незалежний процес, що приймає запити, виконує їх і повертає результати. Клієнт є програмою, що запитує послуги у сервера. У системах більшість серверів також виступають клієнтами, виконуючи запити та надсилаючи їх іншим серверам. Архітектурна модель клієнт-сервер має численні переваги над монолітними рішеннями [21]. Розподіл завдань між клієнтами і серверами дозволяє кожному компоненту зосередитися на вузькій спеціалізації, що спрощує підтримку та оновлення окремих частин системи. Вузькоспеціалізовані сервери можуть бути більш оптимізовані для виконання конкретних завдань, що підвищує загальну продуктивність. Клієнт-серверна

модель також забезпечує масштабованість системи. Легкість додавання нових серверів, або збільшення потужностей існуючих дозволяє адаптувати систему до зростання навантаження без необхідності суттєвих змін у її структурі, їх обслуговування. Клієнтські та серверні процеси працюють окремо, що забезпечує апаратний захист пам'яті, запобігаючи відказ в роботі інших процесів навіть при виникненні помилок.

Елементи системного програмного забезпечення використовують модель обміну повідомленнями. Сервісні процеси, не зайняті обробкою запитів, блокуються на отримання нових запитів і не використовують CPU. Клієнти блокуються до отримання відповіді. Після обробки запиту сервер повертає результат і знову блокується. На рис. 2.3 зображений механізм взаємодії клієнта з сервером, і процес отримання та надсилання відповідей. Ця система використовується в проекті, де клієнтами є користувачі та їх браузери, а програмний додаток Laravel опрацьовує дані та надає представлення в зручному для користувача вигляді.

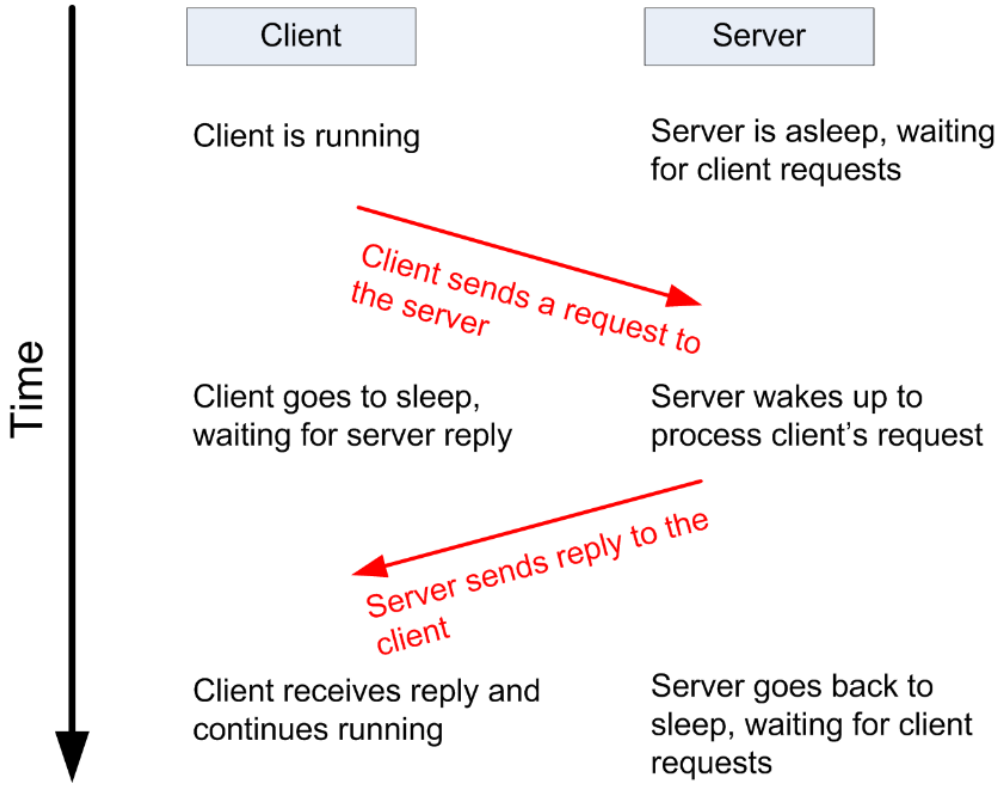

Рис. 2.3. Механізм взаємодії клієнт–серверної архітектурної

Дана архітектура забезпечує чітке розділення обов'язків між клієнтами і серверами. Клієнти зосереджуються на інтерфейсі користувача і запитах, тоді як сервери виконують обробку даних та інші складні операції.

2.3.5. Архітектурний шаблон MVC

Як показано на рис 2.4, архітектура MVC ділиться на три компоненти, а саме модель, яка розкриває логічну область, представлення надає інтерфейс користувача та елемент керування, який керує змінами в перегляді.

Модель представляє дані та правила, які керують доступом до цих даних та їх оновленням. У корпоративному програмному забезпеченні модель часто є програмним наближенням процесу реального світу. Подання відображає вміст моделі. Він визначає, як саме мають бути представлені дані моделі. Коли дані моделі змінюються, представлення має оновлювати своє представлення за потреби. Цього можна досягти за допомогою моделі push [22], у якій представлення реєструє себе в моделі для сповіщень про зміни, або моделі pull [23], у якій представлення відповідає за виклик моделі, коли йому потрібно отримати найновіші дані. Контролер перетворює взаємодію користувача з представленням у дії, які виконуватиме модель. У автономному клієнті з графічним інтерфейсом користувача взаємодії можуть бути натисканнями кнопок або вибором меню, тоді як у корпоративній веб-програмі вони відображаються як HTTP-запити GET і POST. Залежно від контексту, контролер може також вибрати нове представлення, наприклад, веб-сторінку з результатами, щоб показати назад користувачеві.

Laravel використовує архітектурний шаблон MVC для побудови програмних додатків. Реалізація даної архітектури зображена на рис 2.n.

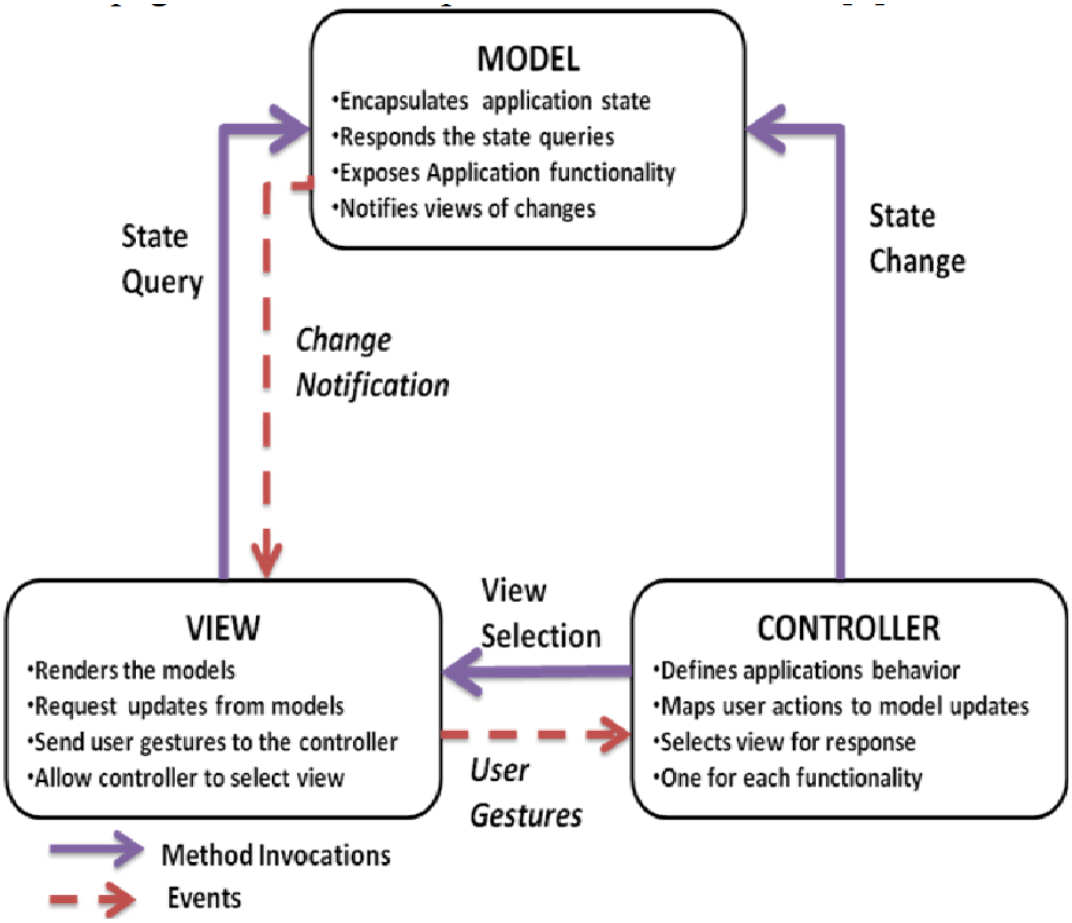

Рис. 2.4. Реалізація шаблону MVC

Метод MVC пропонує розділити одну проблему на три частини: модель, представлення та контролер. Усі види бізнес-логіки містяться в моделі. Представлення відповідає лише за рендеринг виведення. Контролер працює як клей між моделлю та видом. Модель, подання та контролер є самостійними. Контролер отримує запит користувача, потім контролер спілкується з моделлю, щоб надіслати реалізовані дані бізнес-логіки. Згодом ці дані надсилаються до перегляду для рендерингу. Усі види взаємодії користувача відбуваються лише з контролером. Таким чином, ця архітектурна парадигма дозволяє розділити відповідальності та забезпечити модифікацію та розширення програмних систем в процесі тривалої розробки.

2.3.6. Механізм шаблонів Blade

В поточному програмному рішенні, з ціллю уникнення коду в представленні в вигляді запитів до бази даних, та іншої логіки, дотримання концепцій контролера, моделі та поділу завдань на частини, виконується форматування спроектованих частин.

Представлення призначені для того, щоб бути здебільшого позбавленими синтаксису мови програмування, з ціллю їх підтримки та розширення.

Фреймворки, зокрема Laravel, створюють робоче середовище шляхом використання механізмів шаблонів, які надають спрощений синтаксис для вбудовування логіки безпосередньо у представлення.

Механізм шаблонів Laravel називається Blade [24], і реалізує функції, включаючи успадкування, фільтрацію виводу, умовні оператори та цикли. Для розпізнавання фреймворком представлення, доповнене Blade, використовується розширення .blade.php. У цьому розділі ми розглянемо низку різних прикладів із використанням синтаксису Blade і вигляду welcome.blade.php.

Типова веб-програма складається з елементів дизайну, таких як верхня і нижня частини, і ці елементи зазвичай знаходяться на кожній сторінці. Оскільки усунення надмірності є одним із головних принципів Laravel, очевидно, що вам не захочеться повторно вставляти такі елементи, як логотип сайту та панель навігації, у кожне подання. Натомість ви використовуватимете синтаксис Blade для створення головного макета, який потім можна буде успадкувати різними переглядами сторінок. Щоб створити макет, спочатку створіть каталог у ресурсах/переглядах під назвою layouts, а всередині нього створіть файл під назвою master.blade.php.

У фреймворку Laravel, для усунення надмірності, використовується механізм Blade для організації головного макета сторінки, який можна буде використовувати для різних видів сторінок без повторення коду. Цей механізм дозволяє вбудовувати фрагменти коду, такі як логотип сайту та панель навігації у головний макет, тим самим уникнувши повторення цих елементів у кожному

окремому перегляді. Таким чином, Blade дозволяє створювати компоненти, та використовувати їх повторно, що спрощую процес проектування сторінок і розробки в додатку розроблюваної платформи.

2.3.7. Технологія Bootstrap

Bootstrap [25] є фреймворком для створення веб-додатків. Він надає можливість використовувати один і той самий шаблон макета у веб-програмі з готовими класами та темами, що дозволяє зберігати узгодженість використовуваних елементів при розробці.

Фреймворк підтримує версії найбільш використовуваних браузерів [26]. Залежно від браузера, рендеринг елементів може відрізнятися від інших, наприклад, у версіях Chrome і Firefox для Linux. На рис. 2.5 зображена сумісність браузерів з Bootstrap.

Основною метою інтеграції Bootstrap у веб-проект є застосування вибраних кольорових схем, розмірів, шрифтів та макетів, передбачених даною платформою. Платформа надає базові стилістичні визначення для всіх HTML-елементів. Це призводить до уніфікованого вигляду тексту, таблиць та елементів форм у веб-переглядачах.

Bootstrap також включає компоненти JavaScript, які не потребують зовнішніх бібліотек, таких як jQuery [27]. Ці компоненти забезпечують додаткові елементи інтерфейсу користувача, включаючи діалогові вікна, спливаючі підказки, індикатори виконання, навігаційні розкривні меню та каруселі. Кожен компонент побудований на основі HTML-структури, декларацій CSS і, у деяких випадках, супровідного коду JavaScript. Вони розширюють функціональність деяких існуючих елементів інтерфейсу.

В межах веб–орієнтованого додатку, що розроблюється, Bootstrap використовується для побудови сторінки, зокрема блоків та елементів, та з ціллю узгодження структури програмного продукту.

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	Yes	Yes	N/A	No	N/A
iOS	Yes	N/A	N/A	No	Yes (iOS 7 + for v4)
OS X	Yes	Yes	N/A	Yes	Yes
Windows	Yes	Yes	Yes (IE9 + for v4)	Yes	No

Рис. 2.5. Сумісність браузерів з Bootstrap

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис структури програмного додатку

Для опису загальної структури проекту було застосовано метод декомпозиції. Цей підхід системи розкладає системи на цілісні, чітко визначені підсистеми, а підсистеми розкладаються на цілісні компоненти. Представлення структури зображено в табл. 2.1.

Таблиця 2.1.

<table>
<tr><td rowspan="11">Платформа онлайн консультацій</td><td rowspan="11">Консультації</td><td colspan="4">Розклад записів</td></tr>
<tr><td>Пошук</td><td colspan="3">Категорії</td></tr>
<tr><td rowspan="4">Замовники</td><td rowspan="2">Пропозиції</td><td rowspan="2">Сторінка консультації</td><td>Система запису</td></tr>
<tr><td>Розклад пропозиції</td></tr>
<tr><td rowspan="3">Розклад записів</td><td colspan="2">Назва пропозиції</td></tr>
<tr><td colspan="2">Замовник</td></tr>
<tr><td colspan="2">День тижня</td></tr>
<tr><td rowspan="4">Редактор консультацій</td><td rowspan="3">Наявні консультації</td><td rowspan="3">Розклад записів</td><td>Назва пропозиції</td></tr>
<tr><td>Замовник</td></tr>
<tr><td>День тижня</td></tr>
<tr><td>Створення пропозиції</td><td colspan="2">Назва</td></tr>
</table>

Продовж. табл. 2.1

				Категорія
				Графік роботи
				Години роботи
				Опис
				Тривалість сеансу
		Консультанти	Облікова інформація	Електронна пошта
				Телефон
				Ім'я та прізвище
				Дата реєстрації
	Система реєстрації	Ім'я та прізвище		
		Електронна пошта		
		Номер телефону		
		Пароль		
	Профіль	Фото користувача		
		Ім'я та прізвище		
		Електронна пошта		
		Номер телефону		
		Видалення профілю		
	Авторизація	Перевірка прав користувача	Категорія «Експерт» або «Замовник»	
	Аутентифікація	Логін		
		Пароль		

Представлена таблиця описує основні елементи системи в програмному додатку. Процес аутентифікації здійснюється при кожній спробі входу в систему користувачем, при цьому данні надсилаються в відповідний контролер. Процес авторизації, що перевіряє роль користувача в системі. Таким чином, стає можливим обрати роль експерта або консультанта.

2.4.2. Структура бази даних

Таблиці, створені для платформи, виконують функції для забезпечення управління даними. Основні сутності включають:

– Користувачі. Зберігає інформацію про користувачів платформи, включаючи їх імена, прізвища, номери телефонів, електронні адреси, паролі та ролі. Ролі користувачів, визначені в таблиці «Ролі», дозволяють організувати права доступу та обов'язки кожного користувача.

– Консультації. Містить дані про консультації, включаючи їх назви, описи, ідентифікатори користувачів, які створили консультації, та категорії консультацій, що зберігаються в таблиці «Категорії». Категорії допомагають класифікувати консультації за тематикою.

– Сесії консультацій містять дані про конкретні сесії, такі як ідентифікатори консультацій, розкладу, користувачів, дати створення сесій та їх статуси.

– Таблиця «Розклад» зберігає розклади консультацій, включаючи час початку та завершення сесій, їх тривалість та робочі дні.

Зв'язки між таблицями встановлені за допомогою зовнішніх ключів, що забезпечує цілісність даних та підтримку взаємозв'язків між користувачами, ролями, консультаціями, сесіями та розкладом. Логічна структура таблиць бази даних представлена на рис 2.6.

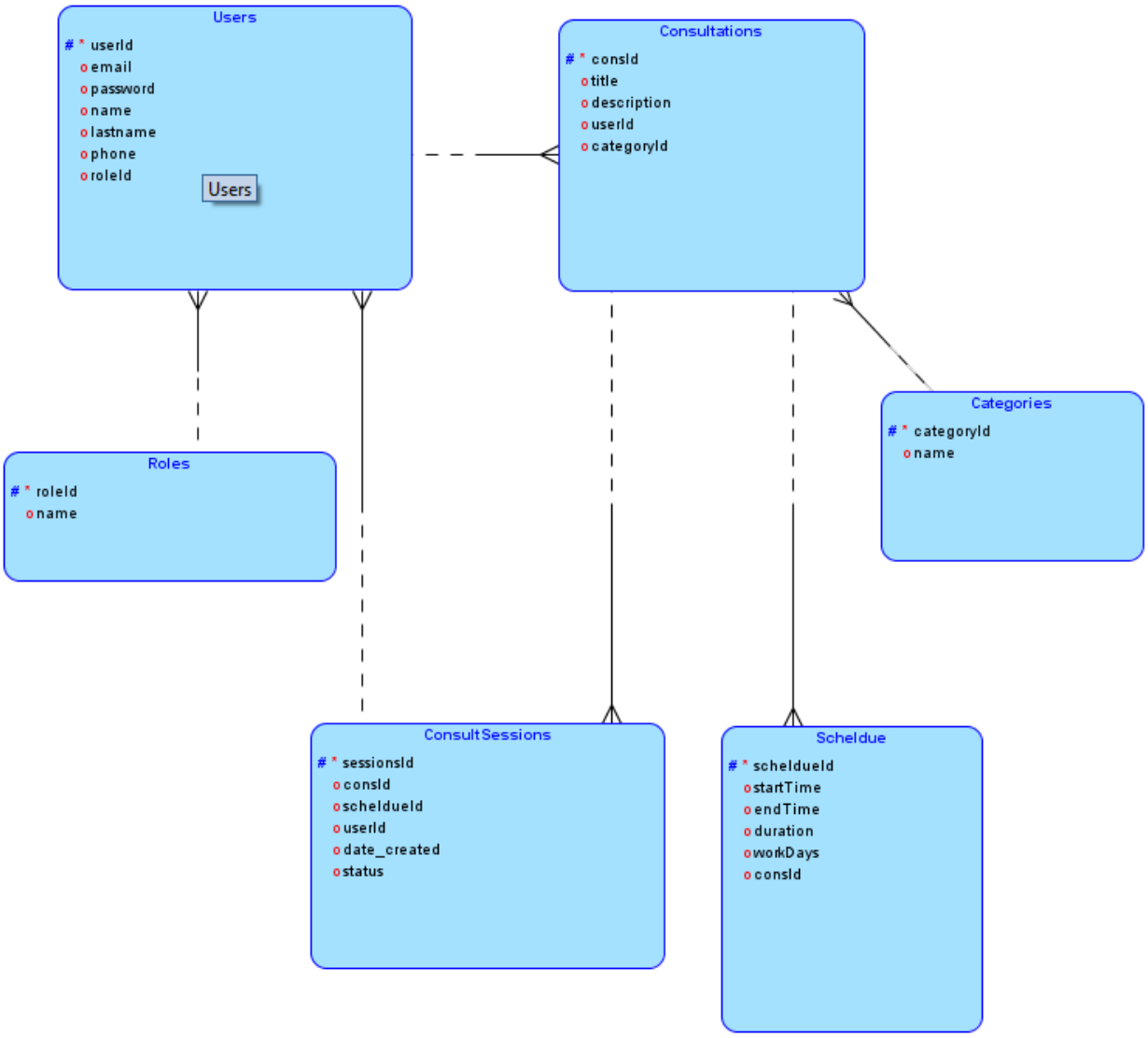

Рис. 2.6. Логічна структура бази даних

Призначення сутностей та їх атрибутів описано в таблиці 2.2.

Таблиця 2.2

СУТНІСТЬ	ПЕРВИННИЙ КЛЮЧ	АТРИБУТИ
Users	Унікальний ключ номер користувача	Унікальний ключ номер користувача Електронна адреса Пароль користувача Ім’я

Продовж. табл. 2.2

		Прізвище Номер телефону Унікальний ключ номер ролі
Roles	Унікальний ключ номер ролі	Унікальний ключ номер ролі Назва ролі
Consultations	Унікальний ключ номер консультації	Унікальний ключ номер консультації Назва консультації Опис Унікальний ключ номер користувача Унікальний ключ номер категорії
Categories	Унікальний ключ номер категорії	Унікальний ключ номер категорії Назва категорії
ConsultSessions	Унікальний ключ номер сесії	Унікальний ключ номер сесії Унікальний ключ номер консультації Унікальний ключ номер розкладу Унікальний ключ номер користувача Дата створення Статус

Продовж. табл. 2.2

Scheldue	Унікальний ключ номер розкладу	Унікальний ключ номер розкладу Початок сеансу Кінець сеансу Тривалість сеансу Робочі дні Унікальний ключ номер консультації

Описана логічна структура бази даних має наступний перелік зв’язків:

– Зв'язок «один до багатьох» існує між таблицями «Користувачі» та «Ролі», де кожен користувач має одну роль, але одна роль може бути пов'язана з багатьма користувачами.

– Між таблицями «Консультації» і «Користувачі» є зв'язок «один до багатьох», де кожна консультація належить одному користувачу, але один користувач може мати багато консультацій.

– «Консультації» та «Категорії» мають зв'язок «один до багатьох», де кожна консультація належить до однієї категорії, але одна категорія може містити багато консультацій.

– Між сутностями «Сесії консультацій» і «Консультації» створено зв'язок «один до багатьох», де кожна сесія консультації прив'язана до однієї консультації. Таким чином, одна консультація може мати багато сесій.

– «Сесії консультацій» та «Користувачі» мають зв'язок «один до багатьох», де кожна сесія консультації відноситься до одного користувача.

– Зв'язок «один до багатьох» між таблицями «Сесії консультацій» та «Розклад» означає, що кожна сесія консультації має один розклад. Один розклад може містити багато сесій.

– Між таблицями «Розклад» і «Консультації» існує зв'язок «один до багатьох», кожен розклад пов'язаний з однією консультацією, та консультація може мати багато розкладів.

Фізична схема таблиць бази даних зображено на рис 2.7. Діаграма зображує відносини між сутностями, їх типи даних та розмірність.

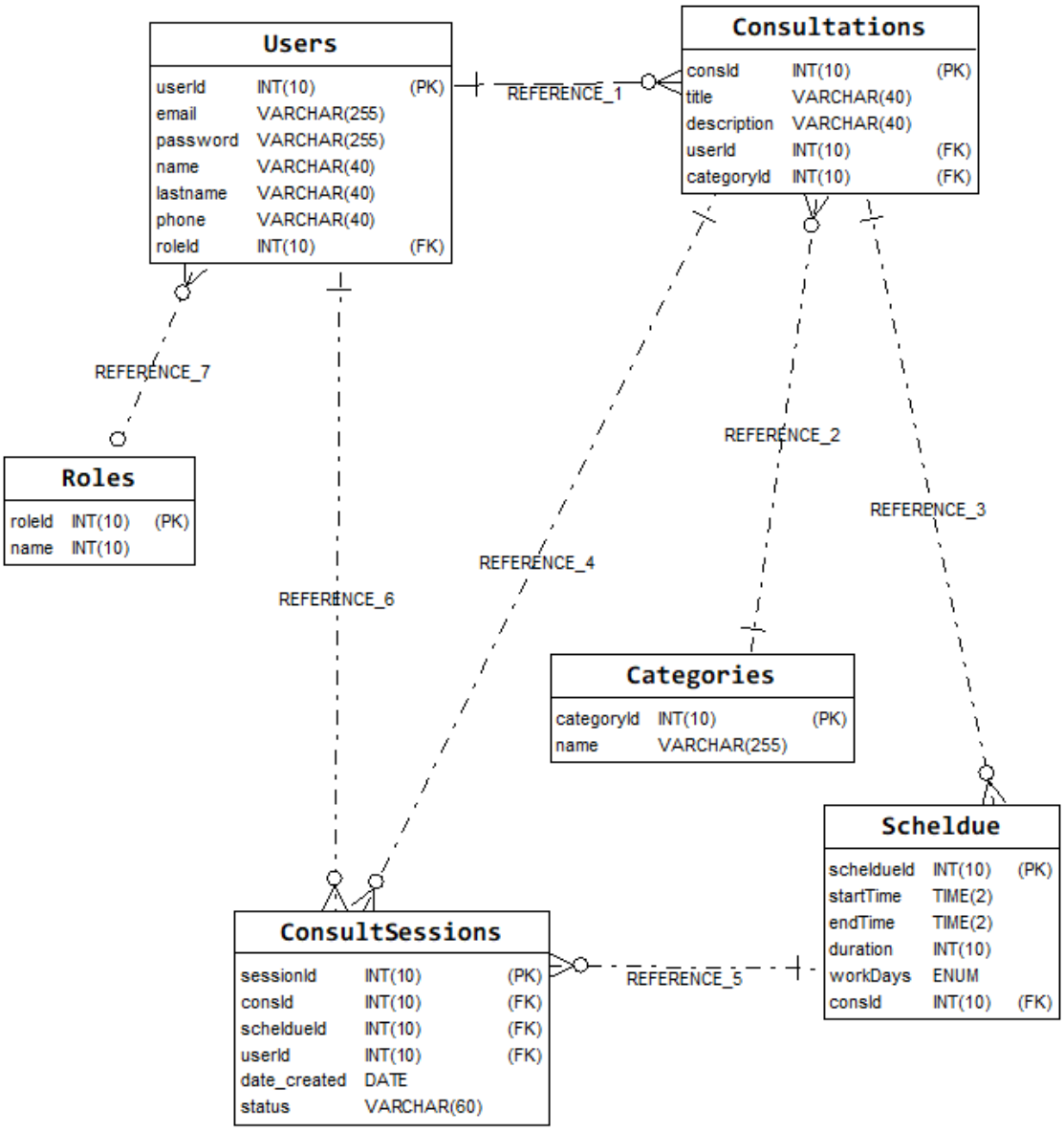

Рис 2.7. Фізична схема бази даних

Детальне представлення фізичного опису сутностей та їх атрибутів зображено в табл. 2.3 – 2.8.

Таблиця 2.3

Таблиця «Користувачі»

Users (Користувачі)				1
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	userId	Унікальний ключ номер користувача	Integer	
2.	email	Електронна адреса	varchar	255
3.	password	Пароль	password	255
4.	name	Ім’я користувача	varchar	40
5.	lastname	Прізвище користувача	varchar	40
6.	phone	Номер телефону	varchar	40
7.	roleid	Унікальний ключ номер ролі	integer	

Таблиця 2.4

Таблиця «Ролі»

Roles (Ролі)				2
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	roleId	Унікальний ключ номер ролі	Integer	
2.	name	Ім’я користувача	varchar	40

Таблиця 2.5

Таблиця «Категорії»

Categories (Категорії)				3
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	categoryId	Унікальний ключ номер категорії	Integer	
2.	name	Назва категорії	varchar	255

Таблиця 2.6

Таблиця «Консультації»

Consultations (Консультації)				4
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	consId	Унікальний ключ номер консультації	Integer	
2.	title	Назва консультації	varchar	40
3.	description	Опис консультації	varchar	40
4.	userId	Унікальний ключ номер користувача	integer	
5.	categoryId	Унікальний ключ номер категорії		

Таблиця 2.7

Таблиця «Сесії консультацій»

ConsultSessions (Сесії консультацій)				5
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	sessionId	Унікальний ключ номер сесії	integer	
2.	consId	Унікальний ключ номер консультації	integer	
3.	scheldueId	Унікальний ключ номер розкладу	integer	
4.	userId	Унікальний ключ номер користувача	integer	
5.	date_created	Дата створення консультації сесії	date	
6.	status	Статус сесії	varchar	60

Таблиця 2.8

Таблиця «Розклад»

Scheldue (Розклад)				2
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1.	scheldueId	Унікальний ключ номер розкладу	integer	

Продовж. табл. 2.8

2.	startTime	Початковий час	time	2
3.	endTime	Кінцевий час	time	2
4.	duration	Тривалість	integer	
5.	workDays	Робочі дні	enum	

2.4.3. Опис файлової структури

Структура проекту Laravel містить наступні елементи на компоненти:

– app/: Вміщує основний код програми. Тут знаходяться контролери, моделі, міделвари, провайдери послуг та інші класи.

– Console/: Містить команди Artisan [28].

– Exceptions/: Обробка винятків.

– Http/: Містить контролери, забезпечення проміжного слою та інші класи, пов'язані з HTTP-запитами.

– Controllers/: Контролери програми.

– Middleware/: Програмне забезпечення для обробки запитів.

– Models/: Містить моделі Eloquent.

– Providers/: Сервіс-провайдери, які завантажують сервіси програми.

– Sevices/: Основні сервіси системи, які реєструються в програмі, та відповідають за бізнес – логіку..

– database/: Містить міграції, фабрики та сідери для бази даних.

– factories/: Фабрики моделей для створення фейкових даних.

– migrations/: Файли міграцій для створення таблиць бази даних.

– seeders/: Сідери для заповнення бази даних початковими даними.

– public/: Публічна директорія, також статичні файли CSS, JavaScript, зображення

– routes/: Містить файли маршрутизації.

– api.php: Маршрути для API.
– channels.php: Маршрути для каналів мовлення.
– console.php: Консольні команди Artisan.
– web.php: Веб-маршрути.

Вище описані елементи структури є основними в Laravel. На рис. 2.8 – 2.9 зображено детальний вигляд файлів в середовищі розробки.

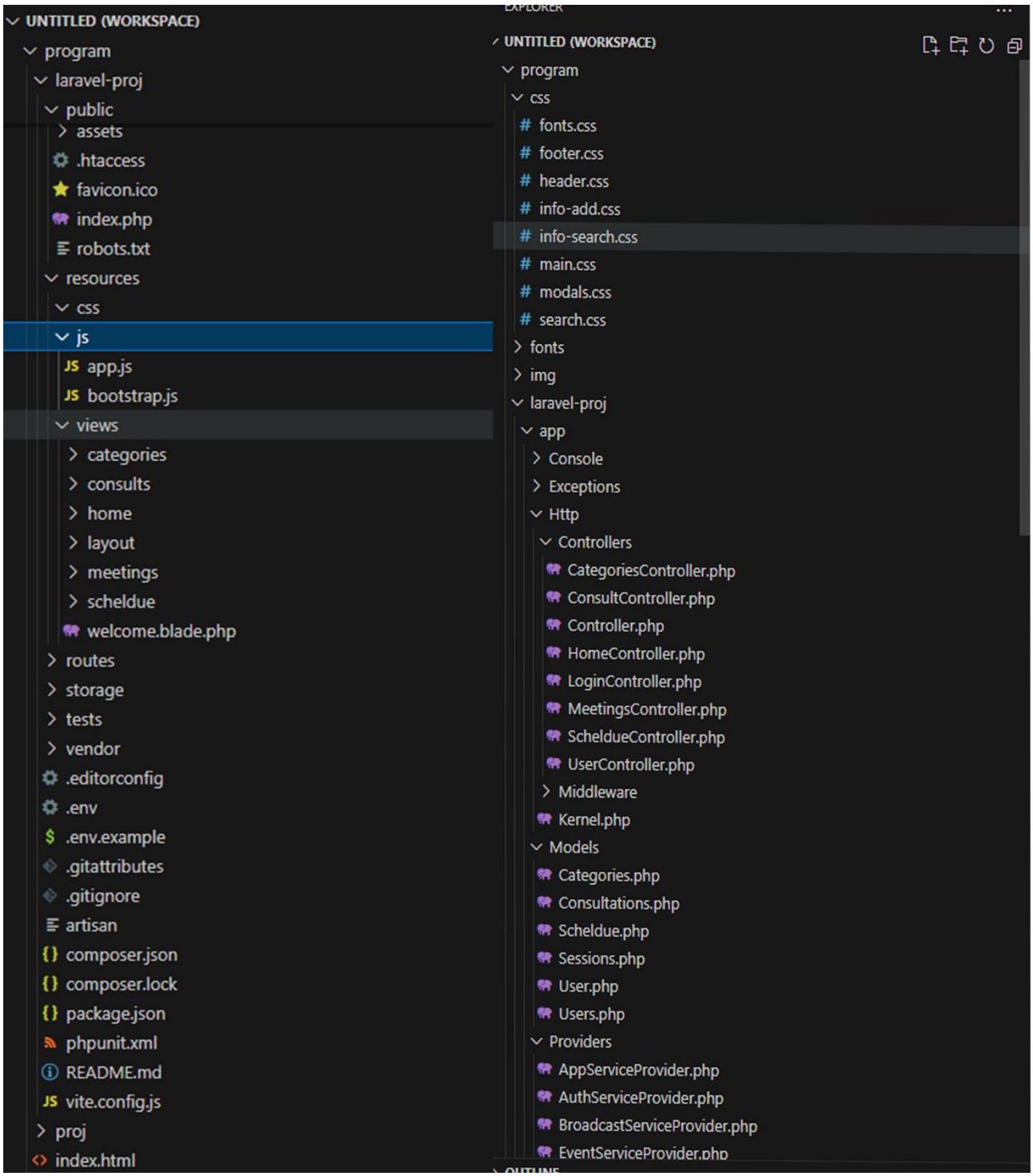

Рис. 2.8. Файлова структура проекту

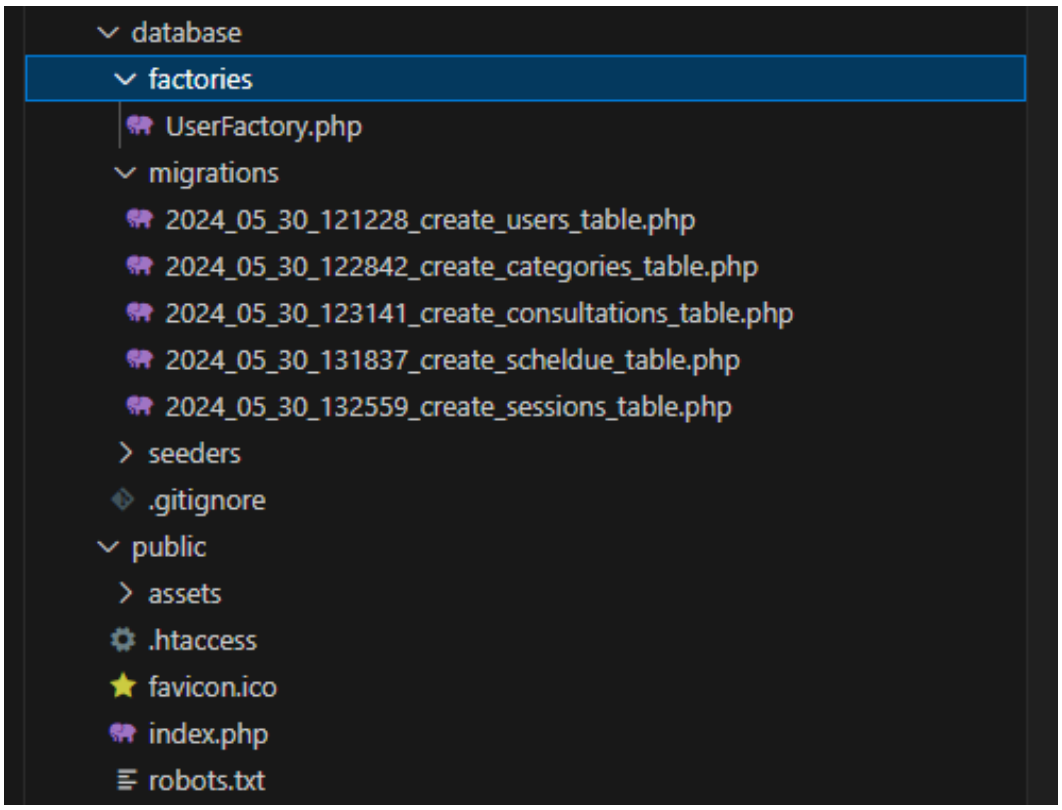

Рис. 2.9. Розгорнута файлова структура

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані включають інформацію про консультації, яка вводиться та створюється консультантом через вікно створення, а також розклад записів, що автоматично генерується на основі вибору замовниками наявних консультацій та пропозицій. Дані про обрані позиції надсилаються серверу, де виконується обробка. Окрім цього вхідні данні організуються наступними способами:

– введення запитів через пошукову форму для знаходження консультацій за категоріями;

– діалогові форми, з попередньою організацією формату даних у вигляді JSON [29].

Вихідні дані подаються в наступному вигляді:

– інформація про консультації виводиться у вигляді списку доступних консультацій за категоріями з детальною інформацією про кожну;

– графік записів на консультації з описом та часом;

– результати пошуку відображаються у вигляді списку консультацій, що відповідають критеріям пошуку;

– інформація про замовників та час запису;

Вихідні дані мають попередню обробку та організацію контролером, яка в подальшому надається користувачам в вигляді сторінок. Дані відображаються у вигляді таблиць, списків, карток на веб-сторінках з використанням HTML для структурованого відображення

Формати вхідних даних включають текстові поля, дати, електронні адреси та номери телефонів, які вводяться через форми на веб-сторінках. Всі дані зберігаються в базі даних у відповідних таблицях з використанням кодування UTF-8.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для розробки і тестування застосунку було використано наступну технічну конфігурацію:

– Процесор: Intel Core i7-10700K 3.80GHz.

– Оперативна пам'ять: 32 ГБ.

– SSD диск: Intel SSD 660p Series.

– Відеокарта: Intel Iris Xe Graphics 6 ГБ.

– Операційна система: 64-розрядна Windows 10.

– Дисплей:

  – Частота оновлення: 120 Гц.

  – Глибина кольору: 8 біт.

  – Формат кольору: RGB.

  – Роздільна здатність: 2560x1440.

- Співвідношення сторін: 16:9.
- Клавіатура.
- Миша.

Вище зазначені вимоги були застосовані для розробки програмного забезпечення, і можуть коректно працювати в межах процесу створення та тестування застосунку, з обмеженою вибіркою даних.

2.6.2. Використані програмні засоби

В процесі розробки веб – орієнтованого застосунку було використано наступні програмні засоби:

- Figma;
- Visual Studio Code;
- Система керування версіями GIT;
- Клієнт MariaDB CLI;
- Oracle Data Modeler;

Figma [30] – інструмент для UX/UI дизайну з режимом реального часу для спільного редагування. Підтримується на Windows, Mac, Linux та ChromeOS. Figma дозволяє створювати прототипи, інтерфейси та графічні елементи. Перевагою є можливість одночасно працювати над проектом кільком користувачам. Цей веб–орієнтований інструмент зберігає зміни в хмарі, але також має локальний кеш для автономної роботи з файлами. Під час розробки використовувався для створення дизайну сайту.

Visual Studio Code [31] – редактор коду, створений Microsoft на основі веб-технологій і доступний для використання під ліцензією Microsoft. Розроблений разом з відкритою спільнотою як редактор з відкритим вихідним кодом. Цей редактор має задокументоване розширення API для інтеграції додаткових функцій, і підтримки багатьох мов і інструментів розробника. В поточній

кваліфікаційній роботі, цей редактор є основним та єдиним інструментом для розробки та тестування програмного додатку.

GIT – система керування версіями для розробки програмного забезпечення. Забезпечує можливість відстежувати та контролювати зміни у файлах і папках проекту. Головною метою є збереження історії змін у коді. Цей засіб використовує концепцію коміту [32] для фіксації змін у файлів, які мають унікальний ідентифікатор, що дозволяє точно відслідковувати розвиток коду. Таким чином, цей інструмент використовується для відстеження змін і можливості відкочування в проекті. На рис. 2.10 зображено Git Bash [33] для використання Git на операційній системі Windows.

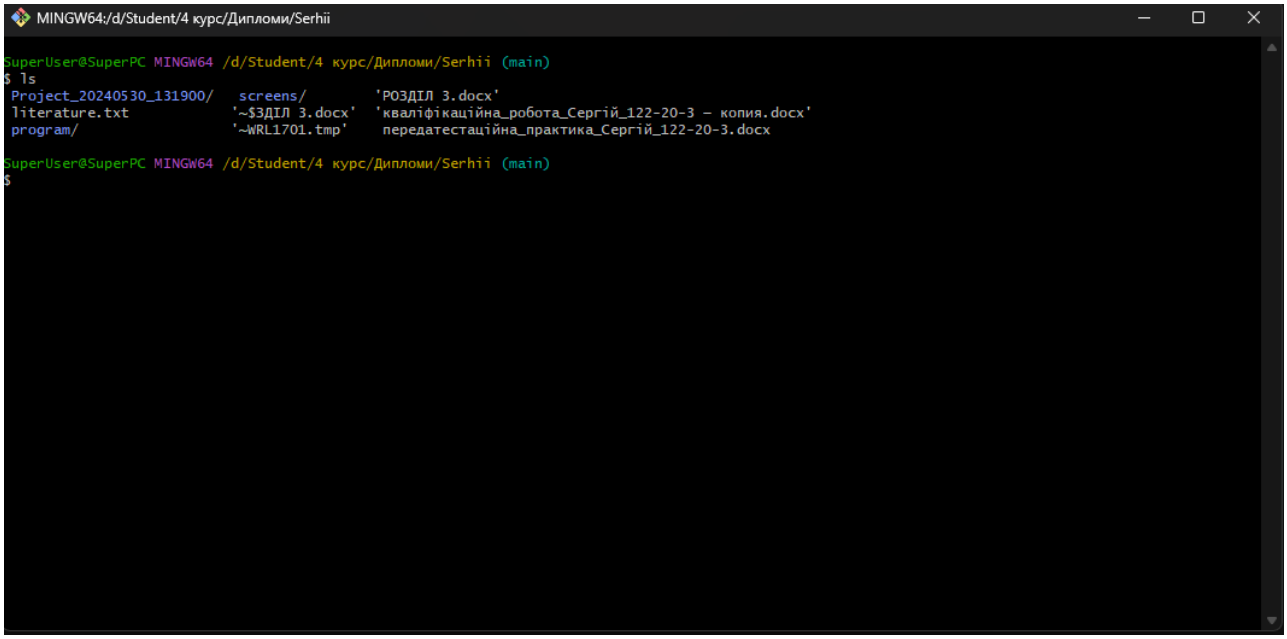

Рис. 2.10. Оболонка Git Bash

MariaDB client [34] – є програмним інструментом, призначеним для взаємодії з базою даних через командний рядок. Цей клієнт дозволяє виконувати операції з базою даних, такі як створення, зміна та видалення таблиць, внесення даних, виконання запитів SQL, а також управління користувачами і дозволами доступу. Клієнт використовується з ціллю

контролю та перевірки коректності створення даних в процесі міграцій (рис 2.11).

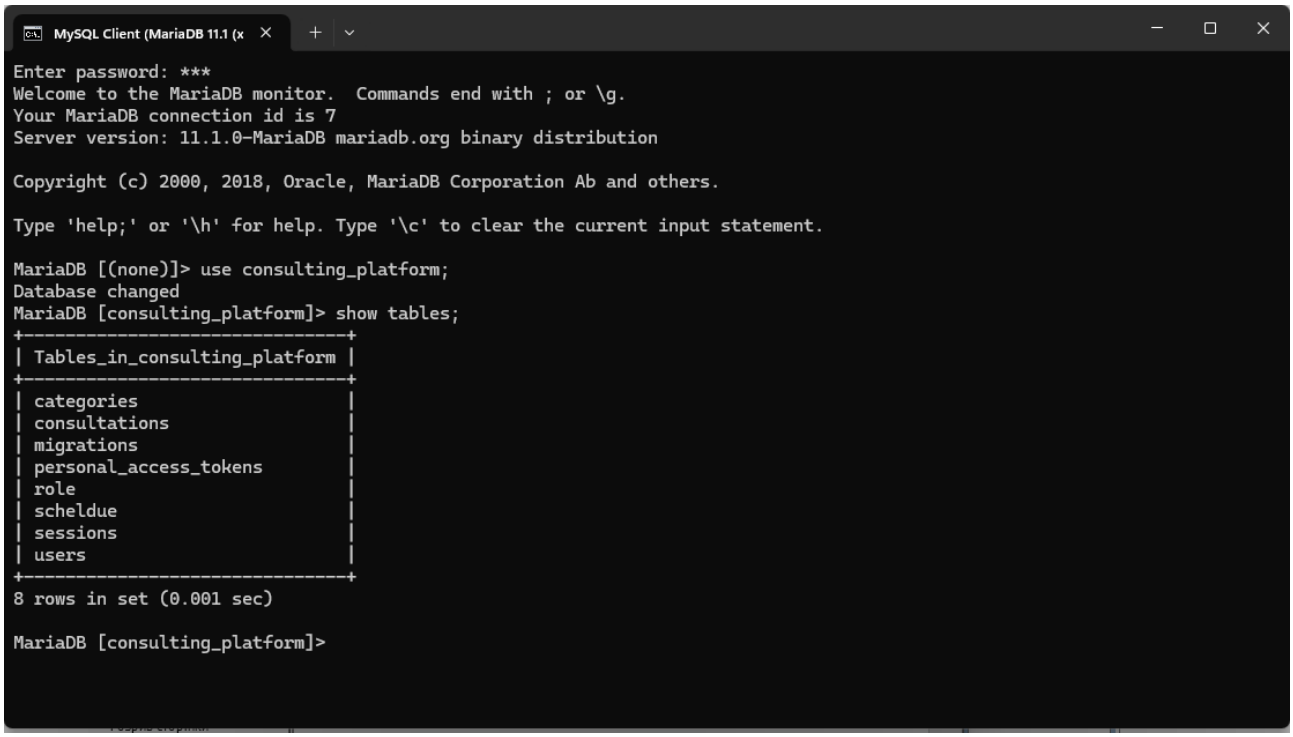

Рис. 2.11. Клієнт MariaDB

DataModeler [35] – інструмент моделювання даних, що дозволяє створювати, переглядати та редагувати логічні, реляційні, фізичні, багатовимірні моделі та моделі типів даних. Цей інструмент надає спектр інструментів і утиліт для моделювання даних і баз даних, включаючи моделювання для діаграм зв’язків сутностей, дизайн бази даних і типу даних із прямим і зворотним проектуванням і DDL [36]. В контексті розробки платформи для консультацій, DataModeler використовувався для проектування бази даних та сутностей.

2.6.3. Виклик та завантаження програми

Для запуску та розгортання програмного застосунку, слід виконати наступні етапи:

– перевірити доступність та працездатність технічного обладнання;

– встановити інтерпретатор PHP, включаючи Composer для управління залежностями;

– встановити сервер MariaDB, налаштувати параметри підключення;

– виконати міграції бази даних за допомогою Laravel;

– завантажити код додатку на сервер або локальну машину;

Для роботи програмного забезпечення обов'язковим етапом є створення таблиць бази даних, виконавши міграції інструментом командного рядка управління і розробки проектів на Laravel – artisan, та перевірити коректність створених сутностей у системі.

2.6.4. Опис інтерфейсу користувача

Після звернення до сайту, відображується головна сторінка, яка зображена на рис 2.12. Користувачу надається меню навігації (рис. 2.13), яке розташовано в шапці, та додаткові елементи управління, зокрема профіль та перемикач сторінки.

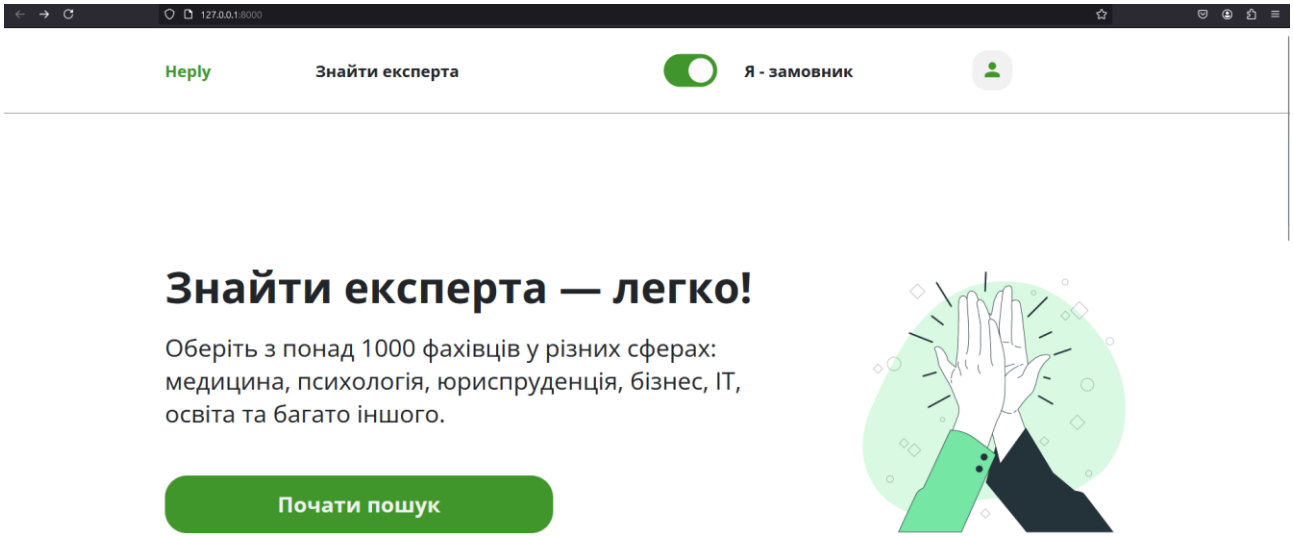

Рис. 2.12. Головна сторінка додатку

Меню навігації дозволяє обирати існуючи категорії консультацій. Цю операцію, також, можливо виконати, натиснувши кнопку «Почати пошук».

Кнопка–перемикач змінює вигляд сторінки для зміни ролі користувача на консультанта, або замовника (рис. 2.14).

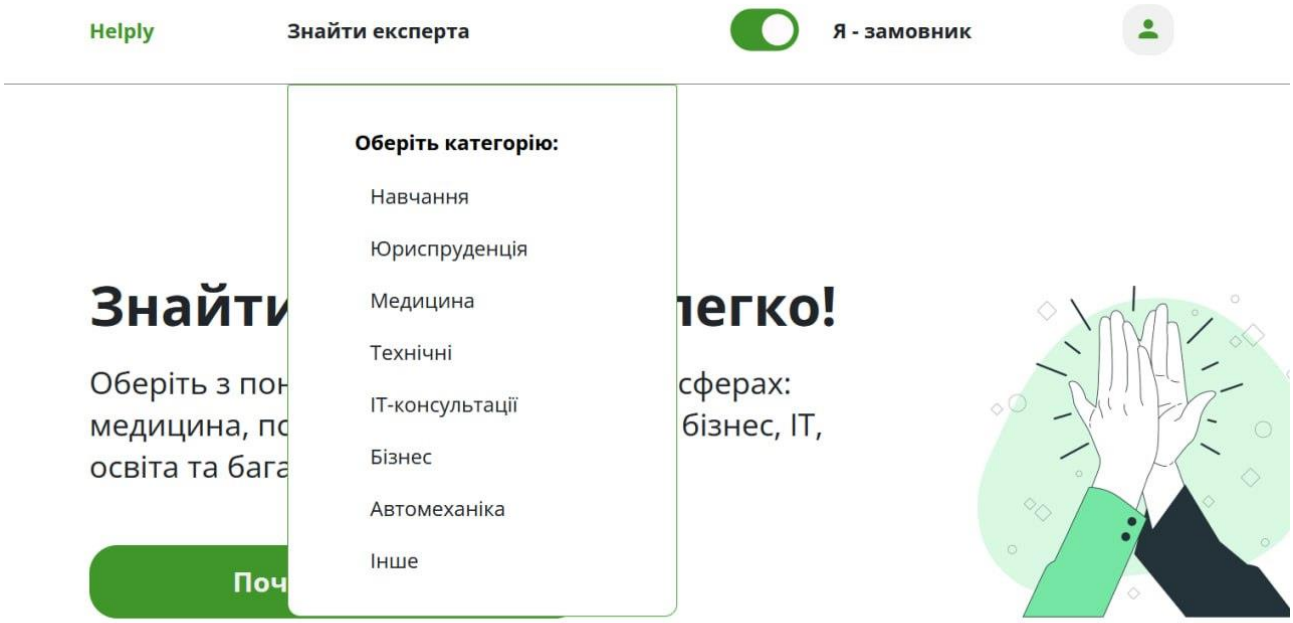

Рис. 2.13. Меню навігації

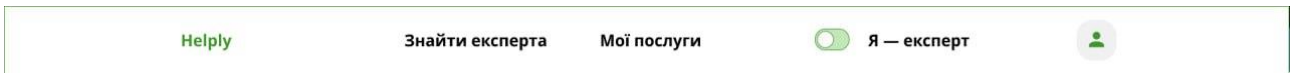

Рис. 2.14. Перемикання сторінки

Нижче, на головній сторінці, розташовані інформативні блоки (рис 2.15 – 2.16), а також футер сайту (рис. 2.16). Шапка та футер присутні на всіх сторінках додатку як спільний шар коду.

Рис. 2.15. Перший інформативний блок

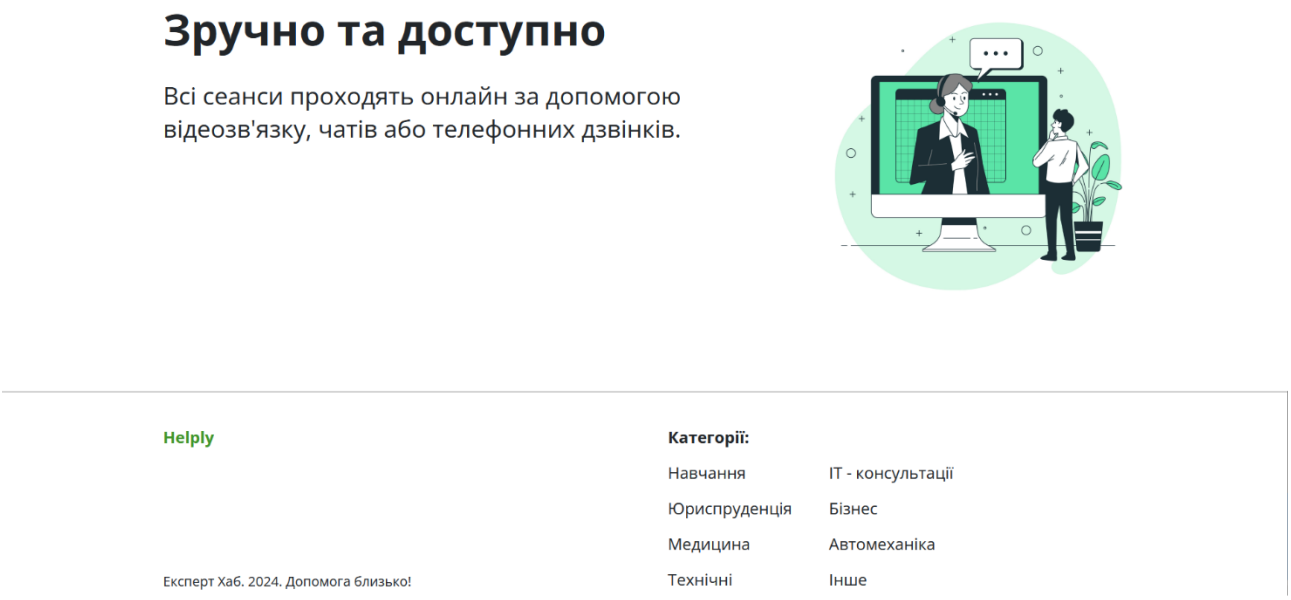

Рис. 2.16. Другий інформативний блок та футер

Сторінка категорій представляє інформацію та фільтри існуючих пропозицій, як відображено на рис. 2.17 – 2.18, інформацію про автора, зокрема функціонал перегляду кожної окремої пропозиції (рис. 2.19), та можливість запису (рис. 2.20). Таким чином, кожен користувач може бути як консультантом, так і замовником.

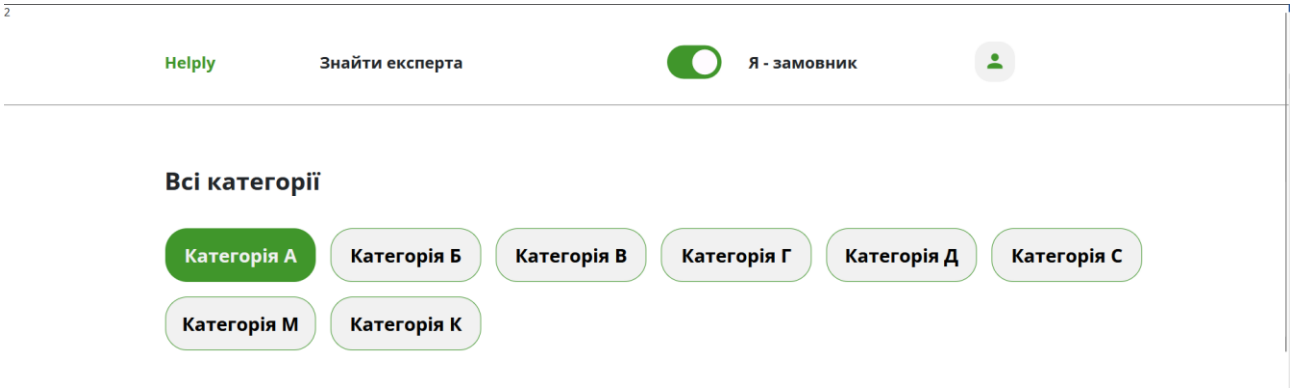

Рис. 2.17. Перелік категорій

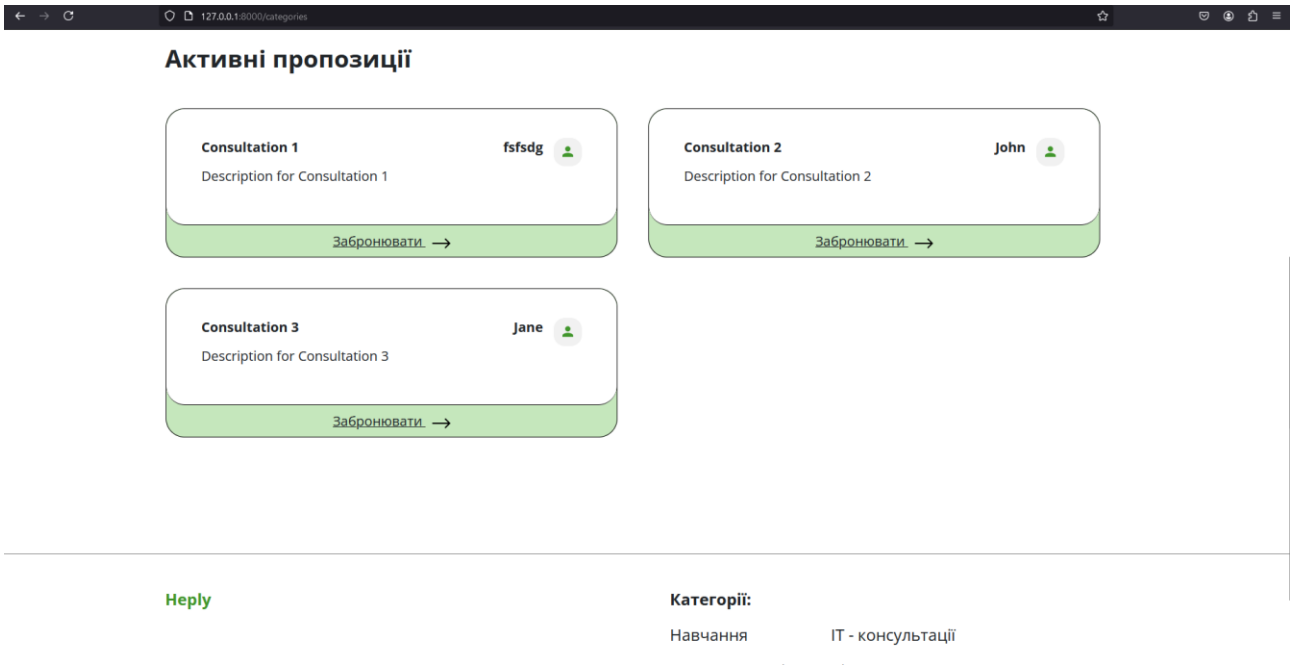

Рис. 2.19. Активні пропозиції

При переході на окрему сторінку, в адресному рядку передається параметр унікальному ідентифікатору запису, який буде відображено (рис 2.21).

На сторінці доступний перегляд дат та днів тижня для запису, кожен з яких містить вільні та заброньовані години. При цьому, існує функція перегляду інформації про консультанта – ім’я та способи зв’язку (рис 2.22).

Рис. 2.20. Інформація про консультацію

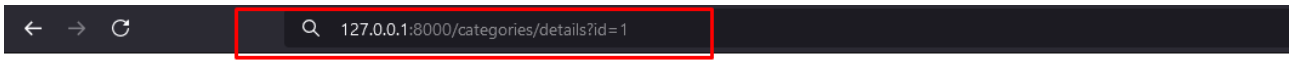

Рис. 2.21. Адресний рядок

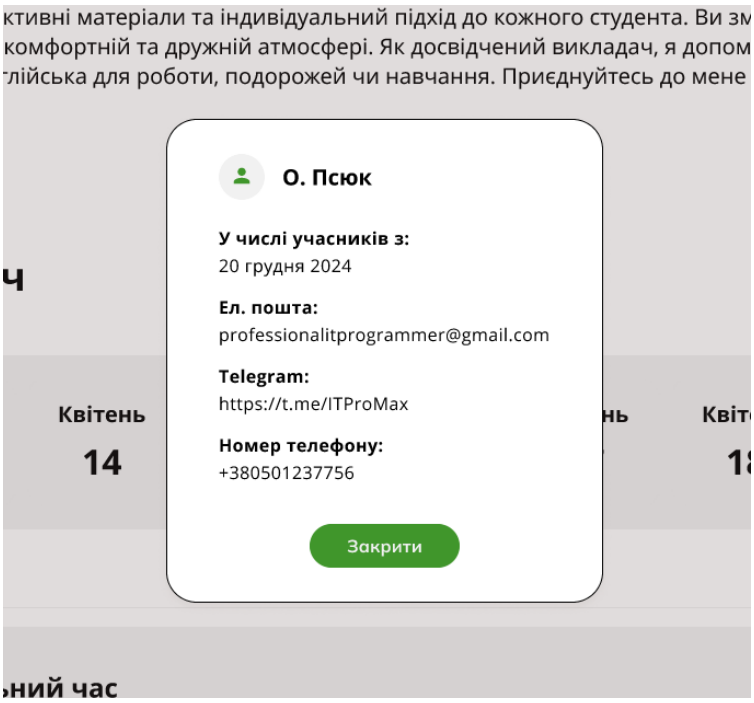

Рис. 2.22. Дані про консультанта

Вікна запису, що були заброньовані, виокремлюються порівняно з доступними. Таким чином, можна переглянути усі зайняті години, та обрати необхідні (рис. 2.23).

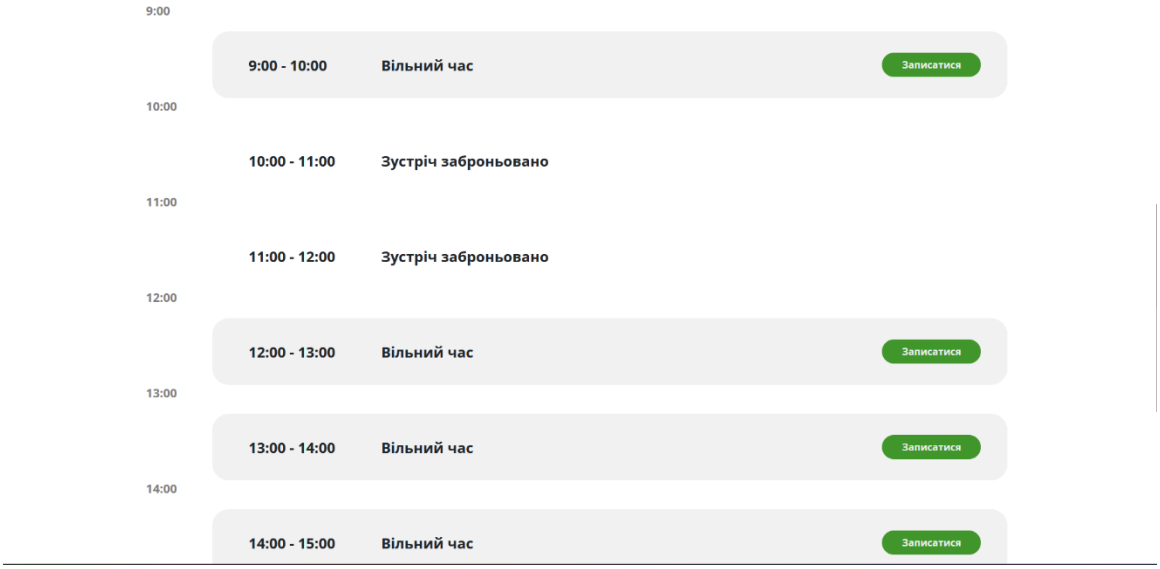

Рис. 2.23. Перелік годин для запису

Модальне вікно вибору надає функцію вибору тривалості сеансу та додаткову інформацію (рис 2.24).

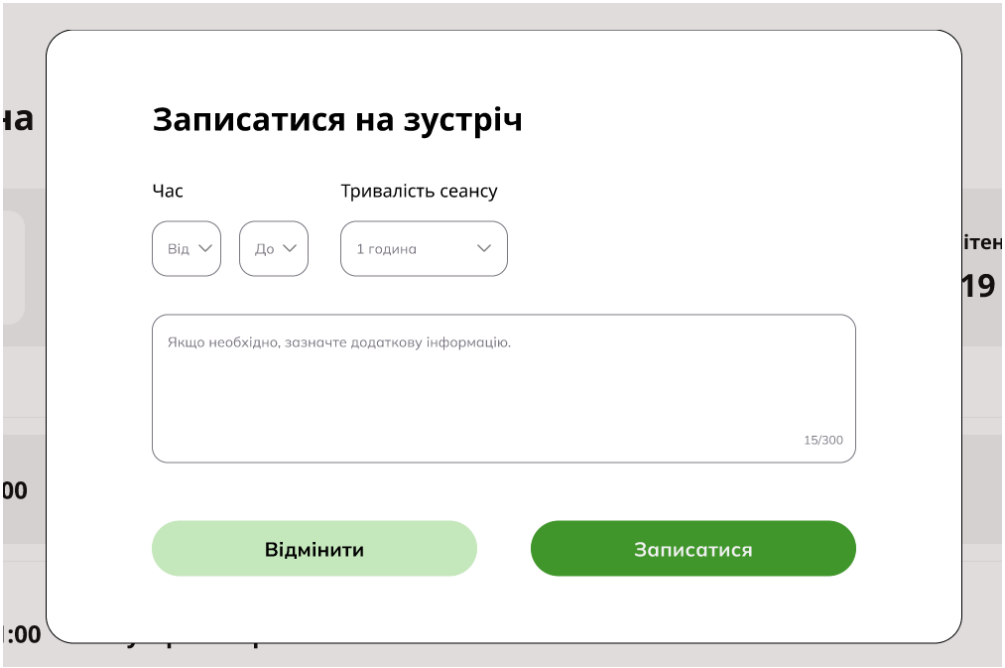

Рис. 2.24. Модальне вікно запису на зустріч

Для отримання можливості запису, користувач повинен бути авторизований в системі. Іконка профілю, яка зображена в шапці сайту, є кнопкою, що активує модальне вікно для входу в систему (рис 2.25).

В модальному вікні присутня кнопка «Зареєструйся зараз», що є тригером визову вікна реєстрації користувача. Ця форма перевіряє введенні дані, реєструє та автоматично авторизує відвідувача в системі (рис 2.26), даючи доступ до функціоналу. В разі невалідності даних, з’являється відповідне повідомлення (рис 2.27).

Після реєстрації, стає доступним інформація про профіль та його редагування (рис 2.n – 2.n).

Рис. 2.25 Форма входу в систему

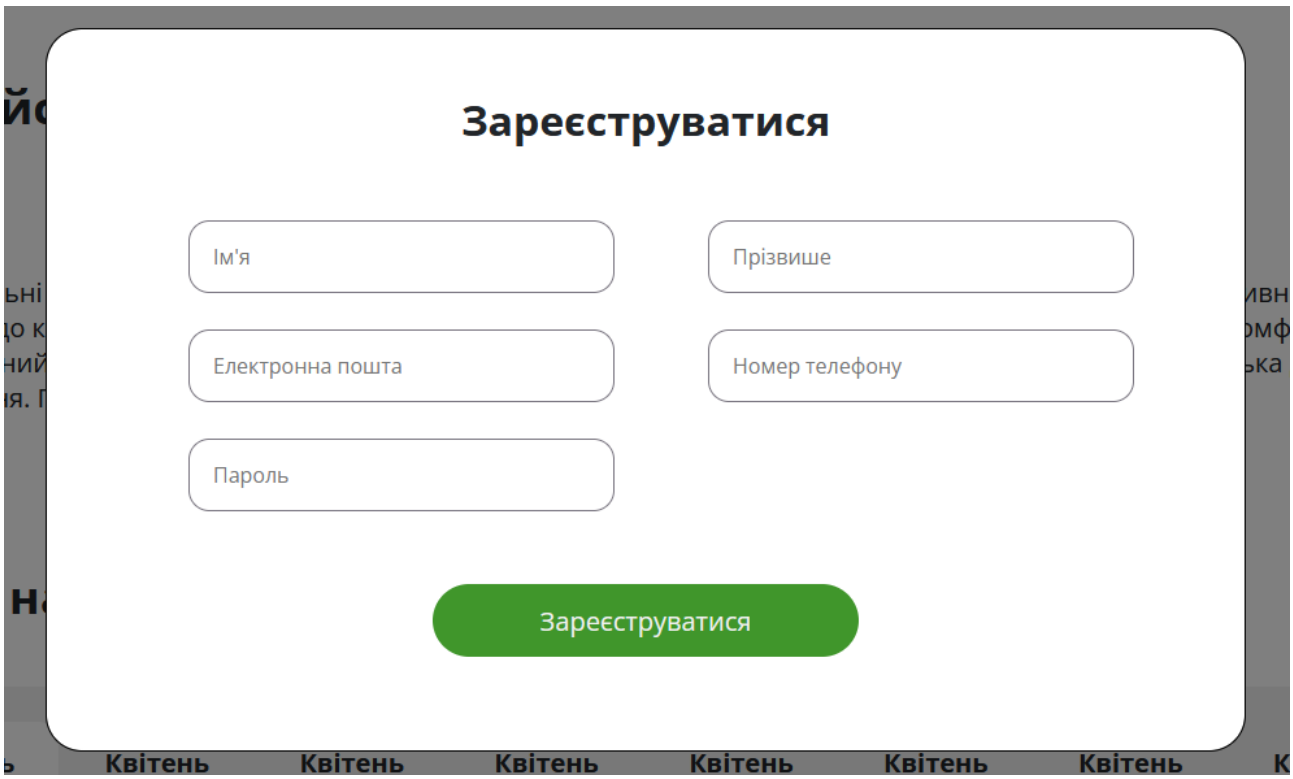

Рис. 2.26. Форма реєстрації

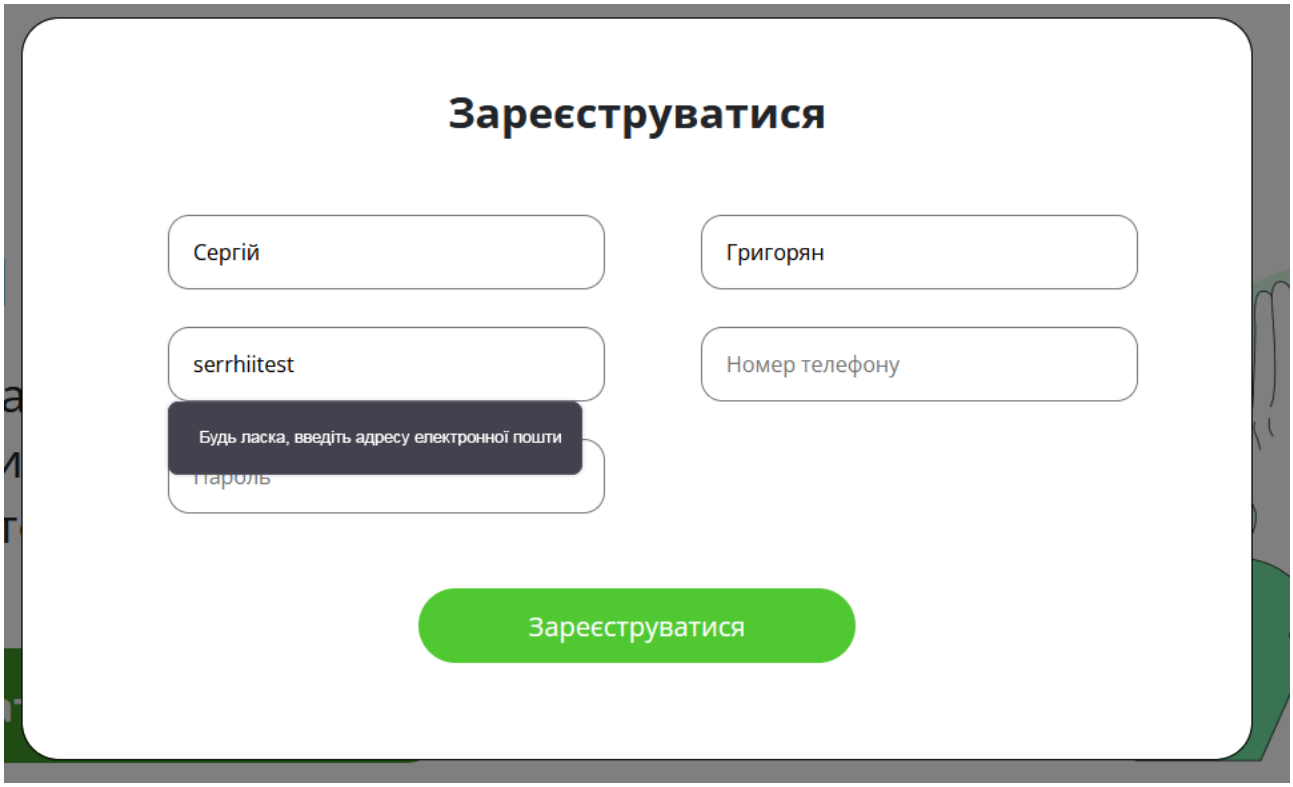

Рис. 2.27. Невалідні дані форми

В випадаючому списку профіля користувача існують функції виходу з системи, редагування поточного профіля та списки консультацій замовника (рис 2.28 – 2.30).

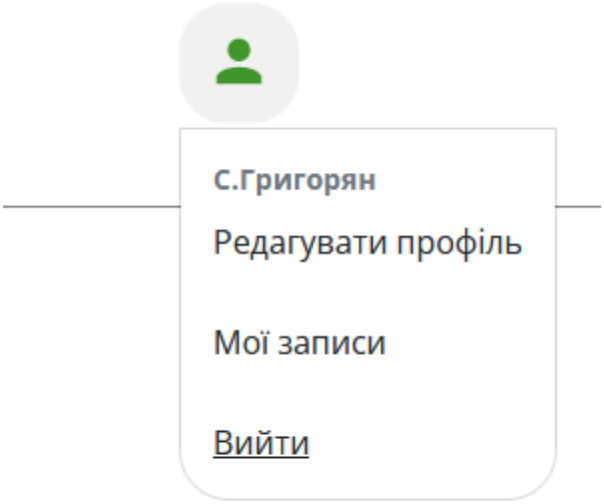

Рис. 2.28. Випадаючий список профілю користувача

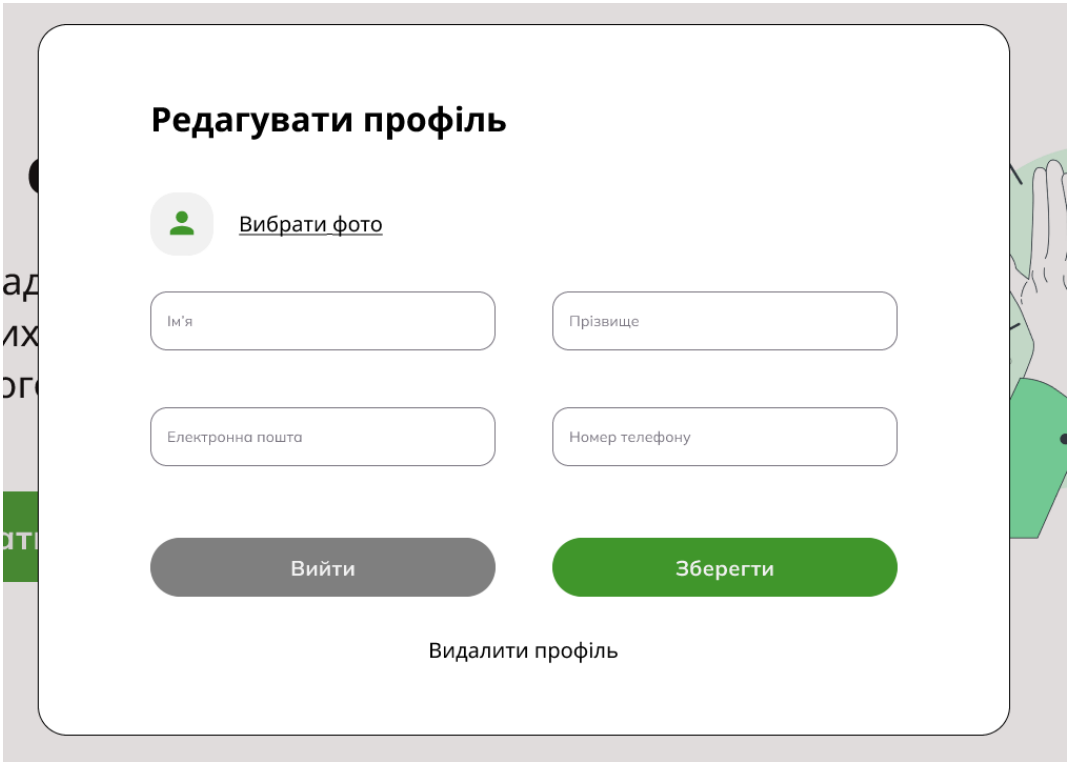

Рис. 2.29. Редагування профілю

В модальному вікні редагування, доступна зміна основних даних користувача та вибору фото, зокрема можливість видалення профілю.

Рис. 2.30. Активні записи користувача

Окрім зазначеного функціоналу, кожен консультант може створювати нові консультації (рис 2.31), що стають доступними в переліку, вносячи основні дані. Також, сторінка дозволяє переглядати вже створені консультації.

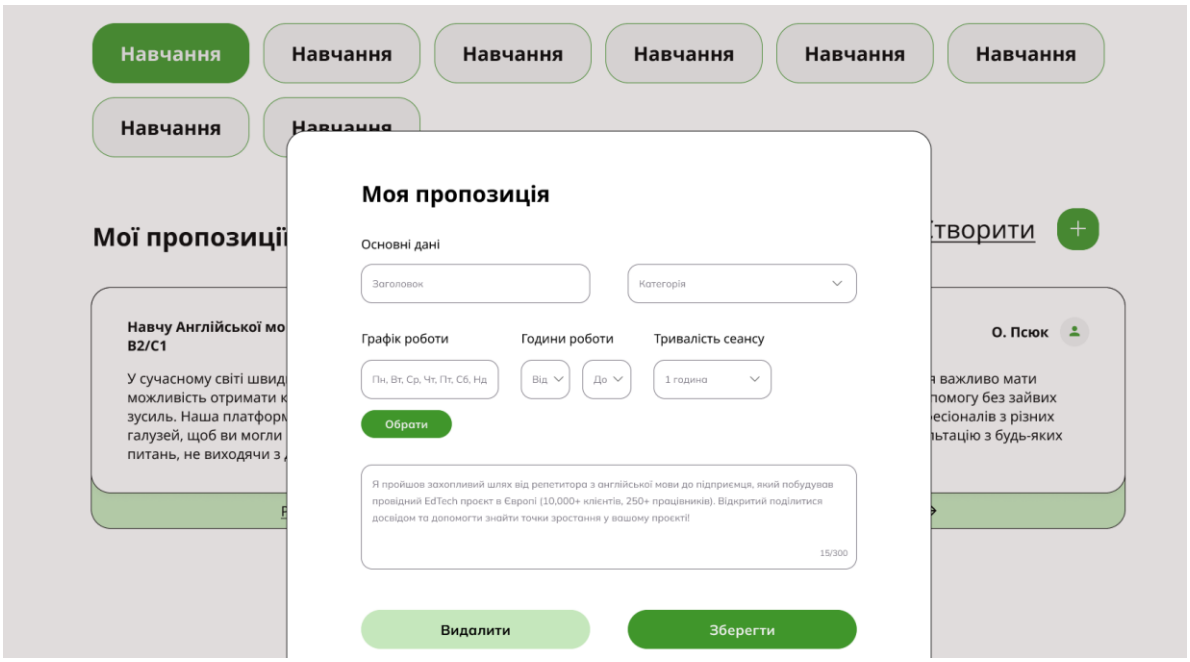

Рис. 2.31. Створення пропозиції

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

– передбачуване число операторів – 2000;

– коефіцієнт корекції програми в ході її розробки – 0,05;

– коефіцієнт складності програми – 1,5;

– годинна заробітна плата програміста– 346,99 грн/год;

Середня щомісячна зарплата PHP розробника в Україні, виходячи з даних представлених на сайті Jooble [37], станом на 8 червня 2024 року складає 55519 грн, або 1371,52$ при курсі валют НБУ 40,48 грн до одного долара [38]. Отже, середня погодинна заробітна плата складатиме 346,99 грн.

– коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,3;

– коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;

– вартість машино-години ЕОМ – 9,1 грн/год;

Розрахунок вартості витрат проведений з урахуванням ціни електроенергії за кВт/год, який становить 2,64 грн враховуючи ПДВ. Спираючись на кількість електроенергії, що споживається пристроєм, на якому ведеться розробка, що становить 70Вт на годину, погодинні витрати складають 0,070*2,64 = 0,18 грн. Для розробки та тестування ефективним рішенням буде використання ноутбука, вартість оренди якого складає 6000 грн в місяць. Тобто, ціна оренди становить 8,33 грн/год. Окрім цього, інтернет – підключення до інтернет – провайдеру «Фрегат» [39] проводиться за умовами тарифного плану «Gigabit» [40], оплата якого складає 395 грн в місяць, або 0,55 грн/год. Отже, значення ЕОМ дорівнює 9,1 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \tag{3.1}$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів (2000);

C - коефіцієнт складності програми (1,5);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси за формулою (3.2) умовне число операторів в програмі:

$$Q = 2000 * 1,5 * (1+0,05) = 3150 \text{ людино-годин}, \tag{3.3}$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \tag{3.4}$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (В = 1,3). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання за формулою (3.4):

$$t_u = (3150 * 1{,}3) / (85 * 1{,}2) = 40{,}15 \text{ людино-годин.} \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \quad (3.6)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.6), отримаємо:

$$t_a = 3150 / (20 \cdot 1{,}2) = 131{,}25 \text{ людино-годин,} \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25) \cdot k} \quad (3.8)$$

За формулою (3.8) та значенням параметру Q, обчислюємо витрати праці на програмування по готовій блок-схемі:

$$t_n = 3150 / (25 \cdot 1{,}2) = 105 \text{ людино-годин,} \quad (3.9)$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k} \tag{3.10}$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання за формулою (3.10):

$$t_{отл} = 3150 / (5 \cdot 1{,}2) = 525 \text{ людино-годин}, \tag{3.11}$$

– за умови комплексного налагодження завдання:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл} \tag{3.12}$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.12):

$$t^{k}_{отл} = 1{,}5 \cdot 525 = 787{,}5 \text{ людино-годин}, \tag{3.13}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \tag{3.14}$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \tag{3.15}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др} \tag{3.16}$$

За формулами (3.15), (3.16) та (3.14) обчислюємо витрати праці на документацію:

$$t_{др} = 3150 / (20 \cdot 1{,}2) = 131{,}25 \text{ людино-годин}, \tag{3.17}$$

$$t_{до} = 0{,}75 \cdot 131{,}25 = 98{,}44 \text{ людино-годин}, \quad (3.18)$$

$$t_{д} = 131{,}25 + 98{,}44 = 229{,}69 \text{ людино-годин}, \quad (3.19)$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 40{,}15 + 131{,}25 + 105 + 525 + 229{,}69 = 1081{,}1 \text{ людино-годин}. \quad (3.20)$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ} \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР} \quad (3.22)$$

де: t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 145,4 грн / год, за формулою (3.22) отримуємо:

$$З_{ЗП} = 1081{,}1 \cdot 346{,}99 = 375127{,}42 \text{ грн}, \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч} \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (22,7 грн/год).

Підставивши в формулу (3.24) відповідні значення, обчислюємо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 525 \cdot 9{,}1 = 4777{,}5 \text{ грн}, \quad (3.25)$$

Звідси, за формулою (3.21), витрати на створення програмного продукту:

$$К_{ПО} = 375127{,}42 + 4777{,}5 = 379\,904{,}92 \text{ грн}, \quad (3.26)$$

Очікуваний період створення програмного застосунку:

$$T = \frac{t}{B_k \cdot F_p}, \quad (3.27)$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси, за формулою (2.27), витрати на створення програмного продукту:

$$T = 1081{,}1 / (1 \cdot 176) \approx 6{,}14 \text{ місяців} \quad (3.28)$$

Висновки: розроблене програмне забезпечення платформи для онлайн консультацій, загальна трудомісткість якого становить 1081,1 людино – годин. Очікуваний період створення веб – орієнтованого додатку становитиме, приблизно, 6,14 місяців, з сумарними витратами на розробку - 379 904,92 грн.

ВИСНОВКИ

В даній кваліфікаційній роботі розроблено веб-орієнтований додаток для платформи онлайн-консультацій.

Проведено аналіз існуючих програмних рішень, який показав, що платформи «Malt» та «Catalant» інтегрують засоби комунікації між клієнтами та консультантами, зокрема внутрішні повідомлення. Системи спрощують інтерфейс та надають можливість орієнтуватися як замовникам, так і консультантам.

Розглянуто формат дистанційної освіти в Україні на поточний момент, що має спільні риси з онлайн-співбесідами. Формат дистанційної освіти дозволяє поєднувати навчання з професійною діяльністю, зокрема сприяє обміну інформації через електронні джерела та електронну пошту між студентами і викладачами.

Практична значущість роботи полягає в створенні веб–платформи, яка може бути використана для надання консультаційних послуг у різноманітних предметних областях. Таким чином, платформа пропонує наступний функціонал:

– організація журналу записів за датами та часом, відображення інформації про позиції за датою та можливість відміни;

– перевірка доступності консультантів у реальному часі та відображення можливих варіантів для запису;

– перегляд інформації про користувачів – консультантів;

– реєстрація користувачів;

– аутентифікація та авторизація в систему;

– створення на перегляд особистих консультацій та запис на існуючі.

Розрахунки економічного розділу демонструють, що загальна трудомісткість проекту становить 1081,1 людино – годин. Очікуваний період створення - 6,14 місяців, з сумарними витратами на розробку - 379 904,92 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лавриненко Лариса Миколаївна. "Освіта в реальності сьогодення – дистанційне навчання." 10 квітня 2020 рік, Луцьк, Україна: МЦНД. DOI: 10.36074/10.04.2020.v1.01. ORCID ID: 0000-0002-7104-6079. Канд. екон. наук, старший науковий співробітник, Інститут економіки та прогнозування НАН України, Україна.

2. Malt Strategy / URL: https://nordics.malt.com/c/maltstrategy. Дата звернення: 12.06.2024.

3. Catalant / URL: https://nordics.malt.com/c/maltstrategy. Дата звернення: 25.05.2024.

4. Soomoro, Z; Shah, M. and Ahmed, J. 2020 Information security management needs more holistic approach: A literature review. International Journal of Information Management 36, 215-225. DOI: 10.1016/j.ijinfomgt.2015.11.009.

5. Laravel / URL: https://laravel.com/. Дата звернення: 12.05.2024.

6. Ahmad, Haafiz Waheed-ud-din. Building RESTful Web Services with PHP 7: Lumen, Composer, API testing, Microservices, and more. Packt Publishing, September 2019. ISBN: 978-1-78712-774-6. Published: 11 September 2019.

7. Laaziria, Majida, Khaoula Benmoussa, Samira Khoulji, and Mohamed Larbi Kerke. "A Comparative study of PHP frameworks performance." The 12th International Conference Interdisciplinarity in Engineering. Procedia Manufacturing, vol. 32, 2019, pp. 864–871. Information System Engineering Research Group, National School of Applied Sciences, Abdelmalek Essaâdi University, Tetouan, Morocco; Faculty of Sciences, Abdelmalek Essaâdi University, Tetouan, Morocco.

8. Hindawi Publishing Corporation. "The Emerging MVC Standard for 3D Video Services." EURASIP Journal on Advances in Signal Processing, vol. 2009, article ID 786015, 13 pages, doi:10.1155/2009/786015.

9. Touch, Joe, John Heidemann, and Katia Obraczka. "Analysis of HTTP Performance." USC / Information Sciences Institute, Aug. 16, 2019. *USC/ISI Research Report* 98-463, Dec. 2019.

10. K. Yagoub, P. Belknap, B. Dageville, K. Dias, S. Joshi, and H. Yu. Oracle's SQL Performance Analyzer. IEEE Data Engineering Bulletin, 2021.

11. Stauffer, M. 2023. Laravel: Up & Running (3rd ed.). O'Reilly Media, Inc. ISBN: 1098153227, 9781098153229.

12. I. J. Education and Management Engineering. 2021. Performance with Eloquent and Query Builder in Crowdfunding System with Laravel Framework. MECS, 3, 31-39. Published online June 2021. DOI: 10.5815/ijeme.2021.03.04.

13. Getting Started with Laravel 4." January 2014. Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK. ISBN 978-1-78328-703-1. Production Reference: 1130114.

14. A. Alghamdi, M. Owda, and K. Crockett, "Natural Language Interface to Relational Database (NLI-RDB) Through Object Relational Mapping (ORM)," In Advances in Computational Intelligence Systems, Springer International Publishing, 2017, pp. 449-464.

15. MySQL / URL: https://www.mysql.com/. Дата звернення: 19.05.2024.

16. Java / URL: https://www.java.com/en/. Дата звернення: 19.05.2024.

17. MariaDB / URL: https://mariadb.org/. Дата звернення: 19.05.2024.

18. Mateos-Garcia, J., & Steinmueller, W. 2019. The institutions of open source software: Examining the Debian community. Information, Economics and Policy, 20,4, 333-344.

19. Dyer, R. J. T. (2015). Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB. "O'Reilly Media, Inc.". ISBN: 1449362877, 9781449362874.

20. Kenler, E., & Razzoli, F. 2021. MariaDB Essentials. Packt Publishing Ltd. ISBN: 178398287X, 9781783982875.

21. Pariag, D., Brecht, T., Harji, A., Buhr, P., & Shukla, A. 2020. Comparing the Performance of Web Server Architectures. Presented at EuroSys'07, March 21–23, 2020, Lisboa, Portugal.

22. Assistant Professor, Department of MCAPES Institute of Technology. 2021. A Comparative Analysis on Modeling and Implementing with MVC

Architecture. In Proceedings of the International Conference on Web Services Computing (ICWSC) 2021. Published by International Journal of Computer Applications® (IJCA). 100 Feet Ring Road, BSK- III Stage, Bangalore- 560085.

23. García, R.F. (2023). MVC: Model–View–Controller. In iOS Architecture Patterns. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-9069-9_2.

24. Blade templates / URL: https://laravel.com/docs/11.x/blade. Дата звернення: 23.05.2024.

25. Bootstrap / URL: https://getbootstrap.com/. Дата звернення: https://getbootstrap.com/.

26. SANGEORZAN, L., ENACHE-DAVID, N., & CARSTEADO, C. 2019. ASPECTS ABOUT INTEGRATE RESPONSIVE WEBSITES FOR CYBER DEFENSE STRATEGIES - BOOTSTRAP VERSUS W3.CSS. Scientific Research and Education in the Air Force – AFASES 2019. DOI: 10.19062/2247-3173.2019.21.10.

27. JQuery / URL: https://jquery.com/. Дата звернення: 23.05.2024.

28. Artisan console / URL: https://laravel.com/docs/11.x/artisan. Дата звернення: 23.05.2024.

29. JSON Schema. 2019. In Proceedings of the 25th International Conference on World Wide Web (WWW 2019) (pp. 25-30). Montréal, Québec, Canada. DOI: 10.1145/2872427.2883029.

30. Figma / URL: https://laravel.com/docs/11.x/artisan. Дата звернення: 23.05.2024.

31. Visual Studio Code / URL: https://code.visualstudio.com/. Дата звернення: 23.05.2024.

32. Blischak, J. D., Davenport, E. R., & Wilson, G. 2019. A Quick Introduction to Version Control with Git and GitHub. Retrieved January 19, 2019.

33. Git documentation / URL: https://www.git-scm.com/book/uk/v2. Дата звернення: 18.05.2024.

34. MariaDB client / URL: https://mariadb.com/docs/server/connect/clients/mariadb-client/. Дата звернення: 23.05.2024.

35. DataModeler / URL: https://www.google.com/search?client=firefox-b-d&q=dataModeler. Дата звернення: 23.05.2024.

36. Atchariyachanvanich, K., Nalintippayawong, S., & Permpool, T. (2017). Development of a MySQL Sandbox for Processing SQL Statements: Case of DML and DDL Statements. Faculty of Information Technology. IEEE. DOI: 978-1-5090-4834-2/17.

37. Joobje / URL: https://ua.jooble.org/salary/middle-php-developer#hourly. Дата звернення: 23.05.2024.

38. НБУ України / URL: https://bank.gov.ua/. Дата звернення: 23.05.2024.

39. Інтернет-провайдер «Фрегат» / URL: https://fregat.com/. Дата звернення: 23.05.2024.

40. Фрегат–тарифні плани / URL: https://fregat.com/rates-and-promotions/price/. Дата звернення: 23.05.2024.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
//shared layout
<!DOCTYPE html>
<html lang = "en">
   <head>
      <!-- Bootstrap CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
         <meta name="csrf-token" content="{{ csrf_token() }}">
      @section('styles')
         <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/main.css') }}">
         <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/fonts.css') }}">
         <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/header.css') }}">
         <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/footer.css') }}">
      @show
   </head>
   <body>
      <header>
         <div class = "header__container d-flex">
            <div class="d-flex logo">
               <div class ="h75 d-flex align-items-center">
                  <img src = "{{ asset('assets/img/customer.png') }}">
               </div>
               <div class="d-flex">
                  <span class="text-green f-24 bold align-self-center">Експерт Хаб</span>
               </div>
            </div>

            <div class = "header__tab-btn">
              <span aria-expanded="false" type="button" data-bs-toggle="dropdown" class = "f-24 bold align-
self-center">Знайти експерта</span>
               <ul class = "dropdown-menu">
                  <h4 class="dropdown-header">Оберіть категорію:</h4>
                  <li><a class="dropdown-item" type="button">Навчання</a></li>
                  <li><a class="dropdown-item" type="button">Юриспруденція</a></li>
                  <li><a class="dropdown-item" type="button">Медицина</a></li>
                  <li><a class="dropdown-item" type="button">Технічні</a></li>
                  <li><a class="dropdown-item" type="button">IT-консультації</a></li>
                  <li><a class="dropdown-item" type="button">Бізнес</a></li>
                  <li><a class="dropdown-item" type="button">Автомеханіка</a></li>
                  <li><a class="dropdown-item" type="button">Інше</a></li>
               </ul>
            </div>

            <div class = "header__switch-btn d-flex">
               <img src="{{ asset('assets/img/switcher-on.svg') }}">
               <span class = "f-24 bold">Я - замовник</span>
            </div>

            @if(!session('username'))
               <div class="header__profile-btn">
```

```
            <div type = "button" class="btn" data-bs-toggle="modal" data-bs-target="#auth">
                <img src="{{ asset('assets/img/profile.svg') }}">
            </div>
        </div>
    @else
        <div class="header__profile-btn">
            <div type = "button" data-bs-toggle="dropdown" class = "f-24 bold align-self-center">
                <img src="{{ asset('assets/img/profile.svg') }}">
            </div>
            <ul class = "dropdown-menu">
                <h4 class="dropdown-header black">{{ $viewModel }}</h4>
                <li><a class="dropdown-item" type="button">Редагувати профіль</a></li>
                <li><a class="dropdown-item" type="button">Мої записи</a></li>
                <li><a class="dropdown-item" type="button"><u>Вийти</u></a></li>
            </ul>
        </div>
    @endif
  </div>
</header>

@yield('main')

<footer>
  <div class = "frame-container">
    <div class = "col-6">
      <div>
        <div class="d-flex logo">
          <img src = "{{ asset('assets/img/customer.png') }}">
          <div class="d-flex">
            <span class="text-green f-24 bold align-self-center">Експерт Хаб</span>
          </div>
        </div>
      </div>

      <div class = "m_t">
        <span class = "f-20">Експерт Хаб. 2024. Допомога близько!</span>
      </div>
    </div>
    <div class = "col-6 d-flex">
      <div class = "col-4">
        <ul>
          <li class = "bold">Категорії:</li>
          <li>Навчання</li>
          <li>Юриспруденція</li>
          <li>Медицина</li>
          <li>Технічні</li>
        </ul>
      </div>

      <div class = "col-4">
        <ul class="m_f">
          <li>IT - консультації</li>
          <li>Бізнес</li>
          <li>Автомеханіка</li>
          <li>Інше</li>
        </ul>
```

```
            </div>
          </div>
        </div>
      </footer>

      @section('scripts')
        <!-- jQuery -->
        <script src="https://code.jquery.com/jquery-3.7.1.min.js" integrity="sha256-
/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo=" crossorigin="anonymous"></script>
        <!-- Bootstrap JS -->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
      @show

      <!-- Modals -->
      <div class="modal fade" id="auth" tabindex="-1" aria-labelledby="authLabel" aria-hidden="true">
        <div class="modal-dialog modal-dialog-centered">
          <div class="modal-content">
            <div class="modal-header justify-content-center pt-5 mb-4">
              <span class="f-32 bold">Вхід</span>
            </div>
            <div class="modal-body">
              <div class="d-flex justify-content-center">
                <form class="fit pb-5">
                  <input placeholder="Логін" class="form-control mb-3">
                  <input placeholder="Пароль" class="form-control mb-3">
                  <button type="button" class="btn btn-primary main-btn auth-btn">Увійти</button>
                  <div class="d-flex justify-content-center btn mt-3" data-bs-toggle="modal" data-bs-
target="#reg" data-bs-dismiss="modal">
                    <u>Зареєструйся зараз</u>
                  </div>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>

      <div class="modal fade" id="reg" tabindex="-1" aria-labelledby="regLabel" aria-hidden="true">
        <div class="modal-dialog modal-dialog-centered">
          <div class="modal-content">
            <div class="modal-header justify-content-center pt-5 mb-4">
              <span class="f-32 bold">Зареєструватися</span>
            </div>
            <div class="modal-body">
              <div class="d-flex justify-content-center">
                <form action="{{ route('user.create') }}" method="POST" class="fit pb-5">
                  @csrf
                  <div class="d-flex">
                    <div class = "col-6 me-5">
                      <input name="name" placeholder= "Ім'я" pattern="[A-Za-zA-Яа-яІіЇїЄє\- ]+"
required title="Приклад: 'Сергій'">
                      <input name="email" type="email" placeholder= "Електронна пошта" required
title="Приклад: user@example.com">
```

```
                        <input name="password" type="password" placeholder= "Пароль" required>
                    </div>
                    <div class="col-6">
                        <input name="lastname" placeholder="Прізвище" required>
                        <input name="phone"  type="tel" placeholder="Номер телефону" required>
                    </div>
                </div>
                <div class = "d-flex justify-content-center pt-2 pb-2">
                    <button type="submit" class="btn btn-primary main-btn auth-
btn">Зареєструватися</button>
                </div>
            </form>
        </div>
    </div>
</div>
</div>

<script>
    $(document).ready(function() {
        $('form').on('submit', function(event) {
            event.preventDefault();

            let jsonData = {
                name: $('input[name="name"]').val(),
                lastname: $('input[name="lastname"]').val(),
                email: $('input[name="email"]').val(),
                password: $('input[name="password"]').val(),
                phone: $('input[name="phone"]').val()
            };

            $.ajax({
                url: '{{ route("user.create") }}',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(jsonData),
                headers: {
                    'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
                },
                success: function(response) {
                    if (response.success) {
                        alert('Форма відправлена');
                    } else {
                        let errorMessage = response.message || 'Помилка при відправці';
                        alert(errorMessage);
                    }
                },
                error: function(xhr, status, error) {
                    let errorMessage = xhr.responseJSON ? xhr.responseJSON.message : 'Невідома помилка';
                    alert('Помилка: ' + errorMessage);
                }
            });
        });
    });
</script>
</body>
```

```
</html>
```

//GridBuilder.php

```
<?php
namespace App\Services;
use Illuminate\Database\Eloquent\Collection;
use App\Models\Users;
class GridBuilder
{
   private $_consultations;
   private int $_count;
   private const GRID_COL = 2;
   private int $_rowCount;
   private $_grid = [[]];
     public function __construct(Collection $consultations){
      $this->_consultations = $consultations;
      $this->_count = $consultations->count();
      $this->_rowCount = ceil($this->_count / self::GRID_COL);
   }
   public function Build(){

      $grid = [[]];

      echo $this->_rowCount;
      for($i = 0, $n = 0; $i < $this->_rowCount; $i++){
         for($j = 0; $j < self::GRID_COL && $n < $this->_count; $j++, $n++){

            $key = Users::find($this->_consultations[$n]->userId)->name;
            $value = $this->_consultations[$n];

            $grid[$i][$j] = [$key => $value];
         }
      }
      $this->_grid = $grid;
   }
   public function GetGrid(){
      return $this->_grid;
   }
}
```

IAuthService.php

```
<?php
namespace App\Services;
interface IAuthService{
   public function login($username, $password) : bool;
   public function logout() : bool;
   public function isLoggedIn() : bool;
}
```

AuthService.php

```
<?php
namespace App\Services;
use App\Models\Users;
class AuthService implements IAuthService{
   public function login($username, $password) : bool{

      $isDataValid = Users::where('username', $username)
                     ->where('password', $password)
                     ->exists();

      if($isDataValid){
         session()->put('username', $username);
      }
      return $isDataValid;
   }
   public function logout() : bool{

      if(session()->has('username')){
         session()->forget('username');

         return true;
      }
      return false;
   }
   public function isLoggedIn() : bool{
      return session()->has('username');
   }
}
```

userTable.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
   /**
    * Run the migrations.
    */
   public function up(): void
   {
      Schema::create('users', function (Blueprint $table) {
         $table->increments('id');
         $table->string('name', 40);
         $table->string('lastname', 40);
         $table->string('phone', 40);
         $table->string('email', 255);
         $table->string('password', 255);
         $table->timestamps();
      });
   }

   /**
    * Reverse the migrations.
```

```
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->increments('id');
            $table->timestamps();
            $table->string('name', 255);

        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('categories');
    }
};
```

//homeController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\View\View;
use App\Models\User;
use App\Models\Post;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Mail;
use App\Mail\ExampleMail;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\Log;

class HomeController extends Controller
{
```

```
public function index(): View
{
    return view('home.index');
}

public function showUsers(): View
{
    $users = User::all();
    return view('home.users', compact('users'));
}

public function createUserForm(): View
{
    return view('home.create_user');
}

public function storeUser(Request $request)
{
    $validated = $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:8',
    ]);

    User::create([
        'name' => $validated['name'],
        'email' => $validated['email'],
        'password' => bcrypt($validated['password']),
    ]);

    return redirect()->route('users.index')->with('success', 'User created successfully');
}

public function showPosts(): View
{
    $posts = Post::paginate(10);
    return view('home.posts', compact('posts'));
}

public function createPostForm(): View
{
    return view('home.create_post');
}

public function storePost(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|string|max:255',
        'content' => 'required|string',
    ]);

    Post::create($validated);

    return redirect()->route('posts.index')->with('success', 'Post created successfully');
}
```

```
public function uploadFileForm(): View
{
    return view('home.upload_file');
}

public function uploadFile(Request $request)
{
    $validated = $request->validate([
        'file' => 'required|file|mimes:jpg,png,pdf,docx|max:2048',
    ]);

    $path = $request->file('file')->store('uploads');

    return redirect()->route('upload.file')->with('success', 'File uploaded successfully: ' . $path);
}

public function sendEmail(Request $request)
{
    $validated = $request->validate([
        'email' => 'required|email',
        'message' => 'required|string',
    ]);

    Mail::to($validated['email'])->send(new ExampleMail($validated['message']));

    return redirect()->route('email.form')->with('success', 'Email sent successfully');
}

public function cacheExample()
{
    if (Cache::has('cached_data')) {
        $data = Cache::get('cached_data');
    } else {
        $data = ['some', 'example', 'data'];
        Cache::put('cached_data', $data, 600); // Cache for 10 minutes
    }

    return view('home.cache', compact('data'));
}

public function logExample()
{
    Log::info('This is an informational message.');
    Log::warning('This is a warning message.');
    Log::error('This is an error message.');

    return view('home.log');
}

public function downloadFile($filename)
{
    $path = storage_path('app/uploads/' . $filename);

    if (!file_exists($path)) {
        abort(404);
    }
```

```
        return response()->download($path);
    }

    public function apiExample(Request $request)
    {
        $response = Http::get('https://api.example.com/data');

        if ($response->successful()) {
            $data = $response->json();
        } else {
            $data = [];
        }

        return view('home.api', compact('data'));
    }

    public function jsonResponseExample()
    {
        return response()->json([
            'status' => 'success',
            'message' => 'This is a JSON response',
        ]);
    }
}
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('consultations', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title', 255);
            $table->string('description', 255);
            $table->unsignedInteger('userId');
            $table->unsignedInteger('categoryId');
            $table->timestamps();

            $table->foreign('userId')->references('id')->on('users');
            $table->foreign('categoryId')->references('id')->on('categories');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
```

```
        Schema::dropIfExists('consultations');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('scheldue', function (Blueprint $table) {
            $table->increments('id');
            $table->time('startTime', 0);
            $table->time('endTime', 0);
            $table->integer('duration');
            $table->enum('workDays', ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']);
            $table->unsignedInteger('consId')->unique();
            $table->timestamps();

            $table->foreign('consId')->references('id')->on('consultations');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('scheldue');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('sessions', function (Blueprint $table) {
            $table->id();
            $table->unsignedInteger('consId');
            $table->unsignedInteger('scheldueId');
```

```
            $table->unsignedInteger('userId');
            $table->string('status', 100);
            $table->timestamps();

            $table->foreign('consId')->references('id')->on('consultations');
            $table->foreign('scheldueId')->references('id')->on('scheldue');
            $table->foreign('userId')->references('id')->on('users');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('sessions');
    }
};


<?php



namespace App\Http\Controllers;

use App\Services\IAuthService;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;

use Illuminate\Validation\Rule;

class LoginController extends Controller
{
    private IAuthService $_authService;

    public function __construct(IAuthService $authService) {
        $this->_authService = $authService;
    }
    public function Login(string $username, string $password){
        $this->_authService->login($username, $password);

        return view("home.index", ['viewModel', $username]);
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\Consultations;
use App\Services\GridBuilder;
use Illuminate\Http\Request;
use Illuminate\View\View;
```

```
class CategoriesController extends Controller
{
   public function Index(): View{

      $consultations = Consultations::all();

      $builder = new GridBuilder($consultations);
      $builder->Build();

      return view('categories.index', ['viewModel' => $builder->GetGrid()]);
   }

   public function Details(): View{
      return view('categories.detail');
   }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Users;

use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;

use Illuminate\Validation\Rule;
use Illuminate\View\View;

class UserController extends Controller
{
   public function Create(Request $request){
      $validatedData = $request->validate([
         'name' => 'required|string|max:40',
         'lastname' => 'required|string|max:40',
         'phone' => 'required|string|max:40',
         'email' => 'required|string|email|max:255|unique:users',
         'password' => 'required|string|min:8|max:255'
      ]);

      Users::create($validatedData);

      return view('home.index');
   }
}
```

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
```

```
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];
}
```

//providers

```
<?php

namespace App\Providers;

use Illuminate\Cache\RateLimiting\Limit;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\RateLimiter;
use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider
{
    /**
     * The path to your application's "home" route.
     *
     * Typically, users are redirected here after authentication.
```

```
     *
     * @var string
     */
    public const HOME = '/home';

    /**
     * Define your route model bindings, pattern filters, and other route configuration.
     */
    public function boot(): void
    {
        RateLimiter::for('api', function (Request $request) {
            return Limit::perMinute(60)->by($request->user()?->id ?: $request->ip());
        });

        $this->routes(function () {
            Route::middleware('api')
                ->prefix('api')
                ->group(base_path('routes/api.php'));

            Route::middleware('web')
                ->group(base_path('routes/web.php'));
        });
    }
}
```

//consultdetails

```
@extends('layout.shared')

@section('styles')

    @parent
    <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/categories/details.css') }}">
    <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/modals.css') }}">

    @endsection

@section('main')

    <section id = "details">
        <div class ="details__container">
            <h3> {{ $view[‘name’] }} </h3>

            <div class = "details__subtitle">
                <div>
                    <img src="{{ asset('assets/img/profile.svg') }}"> <span class = "f-20 bold">О.Псюк</span>

                    <p class="f-20">
                        Я пропоную індивідуальні та групові заняття для всіх рівнів володіння мовою. Використовую сучасні методики, інтерактивні матеріали та індивідуальний підхід до кожного студента.
                        Ви зможете покращити граматику, лексику, вимову та розмовні навички в комфортній та дружній атмосфері. Як досвідчений викладач, я допоможу вам досягти ваших цілей, незалежно від того,
                        чи потрібна вам англійська для роботи, подорожей чи навчання. Приєднуйтесь до мене і відчуйте, як англійська стає частиною вашого життя!
                    </p>
```

```
    </div>
</div>

<div class="detail_mt_xl">
    <h3>Записатися на зустріч</h3>

    <div class = "details__timeline">
        <table>
            <tr>
                <td> <div class="ms-5 pe-4"> <img src="{{ asset('assets/img/left.svg') }}"> <div> </td>

                <td>
                    <div class = "details__cell details__cell_active">
                        <div class="center">
                            <div>
                                <span class="f-20 bold">Квітень</span>
                                <span class="f-32 bold">12</span>
                            </div>
                        </div>
                    <div>
                </td>

                <td>
                    <div class = "details__cell">
                        <div class="center">
                            <div>
                                <span class="f-20 bold">Квітень</span>
                                <span class="f-32 bold">13</span>
                            </div>
                        </div>
                    <div>
                </td>


                <td>
                    <div class = "details__cell">
                        <div class="center">
                            <div>
                                <span class="f-20 bold">Квітень</span>
                                <span class="f-32 bold">14</span>
                            </div>
                        </div>
                    <div>
                </td>

                <td>
                    <div class = "details__cell">
                        <div class="center">
                            <div>
                                <span class="f-20 bold">Квітень</span>
                                <span class="f-32 bold">15</span>
                            </div>
                        </div>
                    <div>
                </td>
```

```
                    <td>
                        <div class = "details__cell">
                            <div class="center">
                                <div>
                                    <span class="f-20 bold">Квітень</span>
                                    <span class="f-32 bold">16</span>
                                </div>
                            </div>
                        <div>
                    </td>

                    <td>
                        <div class = "details__cell">
                            <div class="center">
                                <div>
                                    <span class="f-20 bold">Квітень</span>
                                    <span class="f-32 bold">17</span>
                                </div>
                            </div>
                        <div>
                    </td>

                    <td>
                        <div class = "details__cell">
                            <div class="center">
                                <div>
                                    <span class="f-20 bold">Квітень</span>
                                    <span class="f-32 bold">18</span>
                                </div>
                            </div>
                        <div>
                    </td>


                    <td>
                        <div class = "details__cell">
                            <div class="center">
                                <div>
                                    <span class="f-20 bold">Квітень</span>
                                    <span class="f-32 bold">19</span>
                                </div>
                            </div>
                        <div>
                    </td>

                    <td class="ps-4"> <div class="d-flex justify-content-end me-5"> <img src="{{
asset('assets/img/right.svg') }}"> </div> </td>
                </tr>
            </table>

            <div class="details__scheldue">
                <div class="row">
                    <div class="col-1 left-up"><span class="f-20 bold grey"> 9:00 </span></div>
                    <div class="col-11">
                        <div class="row line mt-5">
```

```
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">9:00 -
10:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Вільний час
</span></div>
                        <div class="col-2 align-self-center">
                           <button class="main-btn btn-size">Записатися</button>
                        </div>
                     </div>
                  </div>
               </div>
               <div class="row">
                  <div class="col-1 left-up"><span class="f-20 bold grey"> 10:00 </span></div>
                  <div class="col-11">
                     <div class="row line mt-5 non-active">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">10:00 -
11:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Зустріч
заброньовано </span></div>
                        <div class="col-2 align-self-center">
                        </div>
                     </div>
                  </div>
               </div>
               <div class="row">
                  <div class="col-1 left-up"><span class="f-20 bold grey"> 11:00 </span></div>
                  <div class="col-11">
                     <div class="row line mt-5 non-active">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">11:00 -
12:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Зустріч
заброньовано </span></div>
                        <div class="col-2 align-self-center">

                        </div>
                     </div>
                  </div>
               </div>

               <div class="row">
                  <div class="col-1 left-up"><span class="f-20 bold grey"> 12:00 </span></div>
                  <div class="col-11">
                     <div class="row line mt-5">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">12:00 -
13:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Вільний час
</span></div>
                        <div class="col-2 align-self-center">
                           <button class="main-btn btn-size">Записатися</button>
                        </div>
                     </div>
                  </div>
               </div>

               <div class="row">
                  <div class="col-1 left-up"><span class="f-20 bold grey"> 13:00 </span></div>
```

```
                <div class="col-11">
                    <div class="row line mt-5">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">13:00 -
14:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Вільний час
</span></div>
                        <div class="col-2 align-self-center">
                            <button class="main-btn btn-size">Записатися</button>
                        </div>
                    </div>
                </div>
            </div>

            <div class="row">
                <div class="col-1 left-up"><span class="f-20 bold grey"> 14:00 </span></div>
                <div class="col-11">
                    <div class="row line mt-5">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">14:00 -
15:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Вільний час
</span></div>
                        <div class="col-2 align-self-center">
                            <button class="main-btn btn-size">Записатися</button>
                        </div>
                    </div>
                </div>
            </div>

            <div class="row">
                <div class="col-1 left-up"><span class="f-20 bold grey"> 15:00 </span></div>
                <div class="col-11">
                    <div class="row line mt-5">
                        <div class="col-2 align-self-center">  <span class="f-24 bold ms-5">15:00 -
16:00</span> </div>
                        <div class="col-8 align-self-center"> <span class="f-24 bold ms-5"> Вільний час
</span></div>
                        <div class="col-2 align-self-center">
                            <button class="main-btn btn-size">Записатися</button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

@endsection

@section('scripts')
    @parent

@endsection
```

```
@extends('layout.shared')

@section('styles')

   @parent
   <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/categories/category.css') }}">
   <link rel = "stylesheet" type="text/css" href = "{{ asset('assets/css/modals.css') }}">

   @endsection
//details
@section('main')

   <section id = "category">
      <div class ="category__container">
         <div class = "category__inner">
            <h3>Всі категорії</h3>

            <div class="category__btns">
               <table>
                  <tr>
                     <td> <button class="main-btn category-btn">Категорія А</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія Б</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія В</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія Г</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія Д</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія С</button> </td>
                  </tr>
                  <tr>
                     <td> <button class="main-btn category-btn non-active">Категорія М</button> </td>
                     <td> <button class="main-btn category-btn non-active">Категорія К</button> </td>
                  </tr>
               </table>
            </div>

            <div class = "category__offers">
               <h3>Активні пропозиції</h3>
               <table>
                  @foreach($viewModel as $row)
                     <tr>
                        @foreach($row as $cell)
                           @foreach($cell as $key => $value)
                              <td>
                                 <div class = "offer-container">
                                    <div class = "offer-info">
                                       <div class="d-flex">
                                          <p class="col-8 f-20 bold">
                                             <span class = "offer_t-size">{{ $value->title }}</span>
                                          </p>
                                          <div class = "col-4 f-20 bold d-flex justify-content-end">
                                             <div class="d-flex">
                                                <span class="offer_name">{{ $key }}</span>
                                                <img src = "{{ asset('assets/img/profile.svg') }}">
                                             </div>
                                          </div>
                                       </div>
                                       <p class = "f-20">
```

```
                                    {{ $value->description }}
                                </p>
                            </div>

                            <div class="offer-book">
                                <span class="f-20"><u> Забронювати </u></span>
                                <img src = "{{ asset('assets/img/arrow.svg') }}">
                            </div>
                        </div>
                    </td>
                @endforeach
            @endforeach
        </tr>
    @endforeach
</table>
</div>
</div>
</section>

@endsection

@section('scripts')
    @parent

@endsection
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Consultations;
use App\Services\GridBuilder;
use Illuminate\Http\Request;
use Illuminate\View\View;

class CategoriesController extends Controller
{
    public function Index(): View{

        $consultations = Consultations::all();

        $builder = new GridBuilder($consultations);
        $builder->Build();

        return view('categories.index', ['viewModel' => $builder->GetGrid()]);
    }

    public function Details(): View{
        return view('categories.detail');
    }
}
```

ДОДАТОК Б

ВІДГУК

Керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

«Розробка веб-платформи для онлайн консультацій з використанням PHP»

Студента групи 122-20-3 Григоряна Сергія Валерійовича

Керівник економічного розділу

Л.В. Касьяненко

доц. каф. ПЕП та ПУ, к.е.н

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	Опис
Пояснювальні документи	
Кваліфікаційна робота_Григорян.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_Григорян.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Diplom.zip	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Григорян.pptx	Презентація кваліфікаційної роботи