

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Веб-сайт тренажерної зали»

Виконав студент групи ІПЗ-19-1
САУХ Ярослав Васильович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: САУХ Ярослав Васильович

Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**

Тема роботи: **«Веб-сайт тренажерної зали»**,

затверджена Наказом університету від «14» лютого 2023 р. № 70/с

Вихідні дані роботи: **методи та засоби, необхідні для підвищення кількості клієнтів тренажерної зали.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Толстой.І.А, викладач кафедри ІПЗ	22.02.2023р.	05.03.2023р.
2	Толстой.І.А, викладач кафедри ІПЗ	06.03.2023р.	17.05.2023р.
3	Толстой.І.А, викладач кафедри ІПЗ	18.05.2023р.	28.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	5 лютого 2023 – 14 лютого 2023	Виконано
2	Аналіз сайтів конкурентів	15 лютого 2023 – 20 лютого 2023	Виконано
3	Формулювання технічного завдання	20 лютого 2023 – 28 лютого 2023	Виконано
4	Опрацювання теоретичних матеріалів	1 березня 2023 – 15 березня 2023	Виконано
5	Проектування структури	16 березня 2023 – 31 березня 2023	Виконано
6	Розробка бази даних та написання програмного коду	1 квітня 2023 – 15 травня 2023	Виконано
7	Тестування	16 травня 2023 – 20 травня 2023	Виконано
8	Написання пояснювальної записки	21 травня 2023 – 28 травня 2023	Виконано
9	Попередній захист	31 травня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу веб-сайту тренажерної зали та пояснювальної записки. Текстова частина викладена на 50 сторінках друкованого тексту. Пояснювальна записка має 22 сторінки додатків. Список використаних джерел містить 31 пункт та займає 3 сторінки. В роботі наведено 37 рисунків. Загальний обсяг роботи 72 сторінки.

Метою роботи є розробка веб-сайту тренажерної зали для подальшого його використання.

В розділах пояснювальної записки було проаналізовано аналогічні продукти, розглянути методи та засоби для реалізації проекту та вибрано найкращі технології для реалізації веб-сайту. Також було побудовано схему варіантів використання веб-додатку. Крім того, було проведено тестування веб-сайту та продемонстрованого його функціонал.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, ТРЕНАЖЕРНИЙ ЗАЛ, БАЗА ДАНИХ, PYTHON, DJANGO.

					ІПЗ.КР.Б-121-23-ІПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Саух.Я.В			Сайт тренажерної зали	Літ.	Арк.	Аркушів	
Керівник		Толстой.І.А					4	69	
					Пояснювальна записка	Житомирська політехніка, група ІПЗ-19-1			
Н. контр.		О.П. Тичина							
Зав. каф.		А.В. Морозов							

ABSTRACT

The bachelor's thesis consists of a software complex of gym website and an explanatory note. The text part is laid out on 50 pages of printed text. The explanatory note has 22 pages of annexes. The list of used sources contains 31 items and occupies 3 pages. There are 37 drawings in the work. The total volume of work is 72 pages.

The purpose of the work is the development of an online store of mobile phones and accessories for its further development.

In the sections of the explanatory note, similar products were analyzed, methods and means for the implementation of the project were considered, and the best technologies were selected for the implementation of the website. A web application use case diagram was also built. In addition, the website was tested and its functionality was demonstrated.

KEYWORDS: WEBSITE, WEB APPLICATION, GYM WEBSITE, DATABASE, PYTHON, DJANGO.

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						5

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ АНАЛОГІВ ТА ВАРІАНТИ ПОБУДОВИ ВЕБ-САЙТУ ...	10
1.1 Постановка задачі	10
1.2 Аналіз аналогічних веб-ресурсів	13
1.3 Вибір архітектури веб-ресурсу	16
1.4 Обґрунтування вибору засобів та вимог до програмного забезпечення	17
Висновки до розділу 1	18
РОЗДІЛ 2. АРХІТЕКТУРА І ПОБУДОВА ВЕБ-САЙТУ ТРЕНАЖЕРНОЇ ЗАЛИ.....	19
2.1 Визначення варіантів використання та структури системи	19
2.2 Розробка бази даних веб-сайту	24
2.3 Проєктування та реалізація алгоритмів роботи веб-сайту.....	26
2.4 Реалізація функціонала веб-сайту тренажерної зали.....	29
Висновки до розділу 2	33
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ ТРЕНАЖЕРНОЇ ЗАЛИ.....	34
3.1 Порядок встановлення та налаштування веб-додатку.....	34
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-сайтом.....	34
Висновки до розділу 3	46
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ	48
ДОДАТКИ.....	51

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						6

БД – База даних;

ОС – Операційна система;

Backend – Серверна частина;

Frontend – Клієнтська частина;

ПЗ – Програмне забезпечення;

ORM – Об'єктно-реляційна проєкція;

CMS – Система управління контентом;

Гб – Гігабайт;

Мб – Мегабайт;

ГГц – Гігагерц;

ВСТУП

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						7

Проблема автоматизації виробничого процесу гостро стоїть у всіх системах масового електронного користування, оскільки здійснює суттєвий вплив на економічні показники будь-якої організації незалежно від форми власності та виду діяльності. Наслідками неефективної політики в галузі автоматизації виробничої діяльності можуть бути як затримки в розвитку фірми, так і цілковита втрата конкурентних переваг та рентабельності бізнесу. Альтернативою розв'язання проблеми ефективної продуктивності компанії може стати впровадження інтернет-технологій у налагоджену систему продажів товарів чи послуг, адже створення веб-сайту є неодмінним атрибутом успішного функціонування електронної комерції. Купівля товарів або замовлення послуг через інтернет наразі витісняє шопінг у офлайн магазинах, оскільки сайт інтернет-магазину дозволяє кожному охочому оперативно ознайомитися з асортиментом продукції/послуги та зробити замовлення, не виходячи з дому.

Здоровий образ життя завжди був і буде у тренді, адже більшість людей прагнуть бути сильними та здоровими, а також привабливо виглядати. Тренажерні зали допомагають вирішувати ці питання. Зокрема веб-сайт тренажерної зали потрібен для того, щоб підвищити продаж клубних карт та кількість індивідуальних занять з тренером, а також для зручності запису членів клубу на заняття без участі адміністраторів та необхідності наймання спеціального персоналу для виконання цієї рутинної роботи. Крім того, веб-додаток тренажерного залу надасть змогу збирати контактні дані та іншу корисну інформацію про відвідувачів.

Виходячи з вищенаведеного, наше дослідження особливостей розробки та практичного застосування веб-сайту для тренажерної зали є актуальним.

Основне завдання кваліфікаційної роботи полягає в наступному:

1. проаналізувати теоретичні засади застосування веб-сайтів;
2. визначити основні етапи розробки веб-сайту;
3. дослідити алгоритм створення веб-сайту для тренажерної зали;
4. спроектувати модель та обґрунтувати вибір системи управління вмістом;

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						8

5. розробити методичні рекомендації щодо використання засобів системного програмування для реалізації клієнт-серверних систем під час створення веб-сайту для тренажерної зали ;

Предметом дослідження кваліфікаційної роботи є теоретико-методичні аспекти та емпіричні засади застосування програмних засобів для реалізації клієнт-серверних систем.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						9

РОЗДІЛ 1. ДОСЛІДЖЕННЯ АНАЛОГІВ ТА ВАРІАНТИ ПОБУДОВИ ВЕБ-САЙТУ

1.1 Постановка задачі

Основна мета створення веб-сайту тренажерної зали полягає в подальшому використанні цього сайту. Це допоможе підняти рівень лояльності відвідувачів даного спортзалу та збільшити впізнаваність сайту. Ці фактори допоможуть залучати більше нових клієнтів, адже вони будуть обирати нашу тренажерну залу через зручність, що в свою чергу допоможе збільшити прибуток.

Інтерфейс програмного продукту повинен бути максимально простим у використанні для того щоб будь-який користувач зміг одразу знайти всю необхідну йому інформацію^[3]. Функціонал програмного продукту повинен мати усі основні функції, які дозволять користувачам легко знаходити всю потрібну інформацію, а саме:

- Перегляд та редагування інформації в особистому кабінеті;
- Перегляд списку тренерів;
- Перегляд детальної інформації про тренера;
- Перегляд списку абонементів;
- Перегляд списку програм тренування;
- Перегляд детальної інформації про програму тренувань;

В процесі проєктування і розробки веб-сайту є безліч етапів. Від збору вихідної інформації до створення веб-сайту і, нарешті, до обслуговування для підтримки веб-сайту в актуальному стані^[9].

Перерахуємо стандартні етапи розробки веб-сайту^[9]:

- Збір інформації;
- Планування;
- Дизайн;
- Розвиток;
- Тестування і обслуговування;

Перший крок у створенні веб-сайту – це збір інформації. При створенні зовнішнього вигляду сайту необхідно враховувати безліч чинників. Цей перший крок насправді найважливіший, оскільки він вимагає твердого розуміння компанії, для якої він створений. Це вимагає хорошого розуміння бізнес-цілей, а також того, як можна використовувати Інтернет для досягнення цих цілей.

- Слід враховувати наступні моменти:
- Призначення (мета сайту). Сайт буде надавати інформацію, просувати послугу чи продавати продукт^[5].
- Цільова аудиторія. Чи є конкретна група людей, яка допоможе досягти поставлених цілей? Корисно уявити ідеального відвідувача веб-сайту. Врахувати його вік, стать або інтереси – пізніше це допоможе визначити кращий стиль дизайну для сайту^[5].
- Контент. Яку інформацію цільова аудиторія буде шукати на сайті^[5].

Наступним етапом є розробка плану для майбутнього веб-сайту. Це місце, де розробляється карта сайту. Карта сайту являє собою список всіх основних тематичних областей сайту, а також підтем, якщо є. Це служить керівництвом щодо того, який контент буде на сайті, і необхідний для розробки послідовної, простий для розуміння системи навігації^[5].

При розробці сайту слід мати на увазі кінцевого користувача веб-сайту, також відомого як клієнт. Зрештою, це люди, які будуть дізнаватися про послуги або купувати ваш продукт. Хороший користувацький інтерфейс створює зручний для навігації веб-сайт і є основою для цього^[21].

На етапі планування веб-дизайнер також допоможе вам вирішити, які технології слід впровадити. При плануванні веб-сайту обговорюються такі елементи, як якась CMS, наприклад WordPress, будуть потрібні які-небудь контактні форми. Спираючись на інформацію, зібрану до цього моменту, слід визначити зовнішній вигляд сайту^[9].

Цільова аудиторія – один з ключових чинників, що беруться до уваги. Наприклад, сайт, призначений для підлітків, буде відрізнятися від сайту призначеного для фінансової установи. На етапі проектування також важливо

включити такі елементи, як логотип компанії або кольору, щоб посилити індивідуальність компанії на веб-сайті^[9].

Варто створити один або кілька прототипів для веб-сайту. Зазвичай це зображення у форматі .jpg того, як буде виглядати остаточний дизайн^[9].

Стадія розробки – це момент, коли створюється сам веб-сайт. На цьому етапі розробник візьме всі окремі графічні елементи з прототипу і використає їх для створення реального функціонального сайту. Зазвичай для цього спочатку створюється домашня сторінка, а потім «оболонка» для внутрішніх сторінок. Оболонка служить шаблоном для сторінок вмісту сайту, оскільки вона містить основну структуру навігації для веб-сайту.

Як тільки оболонка буде створена, розробник візьме наявний контент і поширить його по всьому сайту у відповідних областях.

Такі елементи, як CMS, такі як WordPress, інтерактивні контактні форми або візки для покупок електронної комерції, також впроваджуються і стають функціональними на цьому етапі. Весь цей час розробник повинен продовжувати робити незавершений веб-сайт доступним для перегляду, щоб користувач міг запропонувати будь-які додаткові зміни або виправлення^[7].

З технічної точки зору успішний веб-сайт вимагає розуміння інтерфейсної веб-розробки. Це включає в себе написання дійсного коду HTML/CSS, який відповідає поточним веб-стандартам, максимізує функціональність, а також робить його доступним для максимально широкої аудиторії^[7].

На етапі тестування займаються остаточними деталями та тестують майбутній веб-сайт. Тестуванню підлягають такі речі, як повна функціональність форм або інших скриптів, а також останнє тестування на предмет проблем сумісності в останню хвилину (перегляд відмінностей між різними веб-браузерами), гарантуючи, що веб-сайт оптимізований для правильного перегляду в самих останніх версіях браузерів^[9].

1.2 Аналіз аналогічних веб-ресурсів

Наразі в мережі інтернет досить багато веб-сайтів тренажерних залів, хоча багато залів не мають свого веб-сайту взагалі. В наш час це є доволі поганою практикою, адже веб-сайт дає можливість будь-кому знайти відповіді на більшість його питань та зацікавити майбутнього клієнта.

Щоб розуміти як має виглядати наш веб-сайт потрібно проаналізувати веб-сайти конкурентів для того щоб знайти всі їх помилки, та не допустити їх під час створення власного веб-сайту.

Першим став фітнес-зал «TRX Pantera». Це фітнес-зал в Житомирі який спеціалізується на заняттях за методикою TRX.

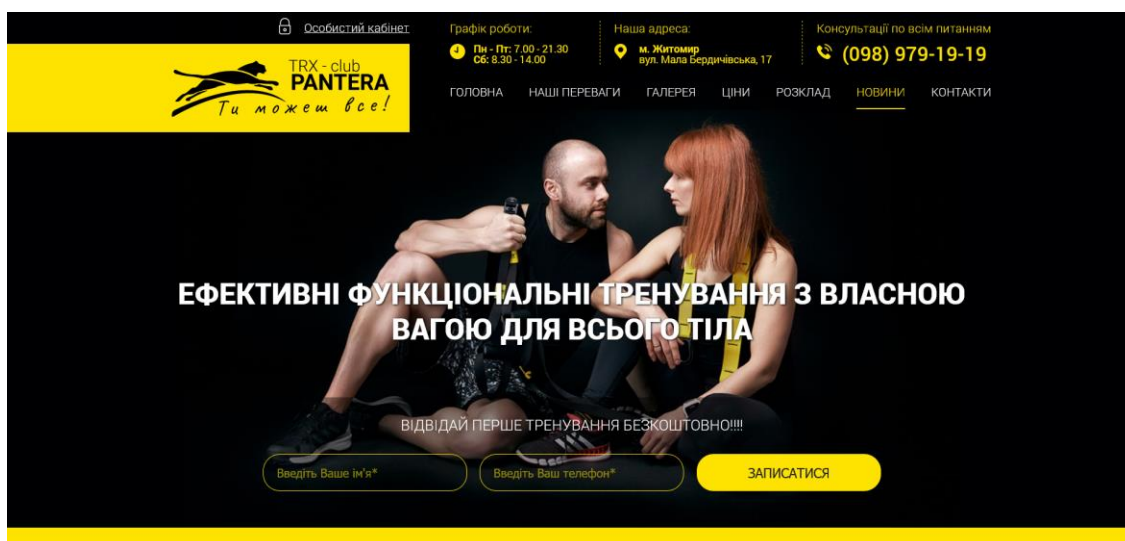


Рис. 1.1 – Головна сторінка сайту TRX Pantera

Даний веб-сайт представлений у вигляді лендінгу на якому розміщена вся найбільш важлива інформація про зал. Сайт виконаний в приємній кольоровій гамі, та не нагромаджений великою щільністю інформації. Також на сайті передбачена можливість входу в особистий кабінет, в якому можна переглянути свої записи, та записатись на нове тренування.

Наступним став веб-сайт фітнес-клубу «Arena Gym». Це спортзал в центрі міста Житомир, який надає широкий спектр послуг, наприклад таких як фітнес клуби, заняття з боксу та кікбоксингу.

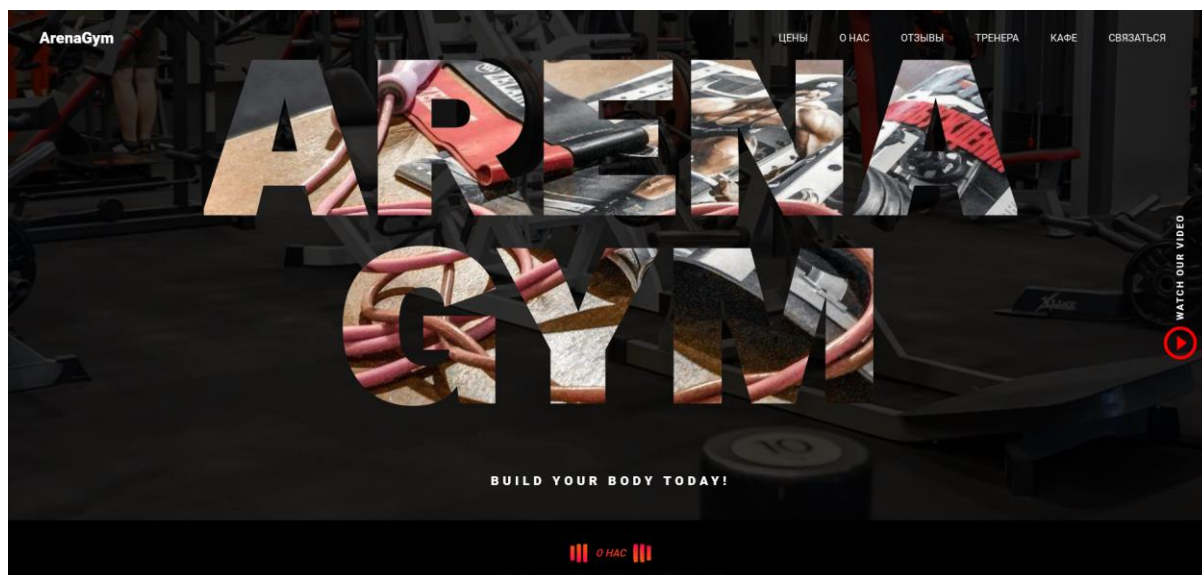


Рис. 1.2 – Головна сторінка сайту Arena Gym

Щодо дизайну даного веб-сайту, то він досить простий, та не викликає великого зацікавлення. В цілому це досить типовий представник веб-сайтів спортзалів. Цей веб-сайт також є лендінгом, але тут немає можливості зайти у власний кабінет. Також на даний момент частина функцій даного веб-сайту не працює.

Третім сайтом для аналізу став веб-сайт мережі фітнес центрів «Sport Life». Це найбільша мережа фітнес-центрів в Україні. У закладах мережі окрім тренажерного залу ви можете відвідати басейн, лазню та спа-комплекс.

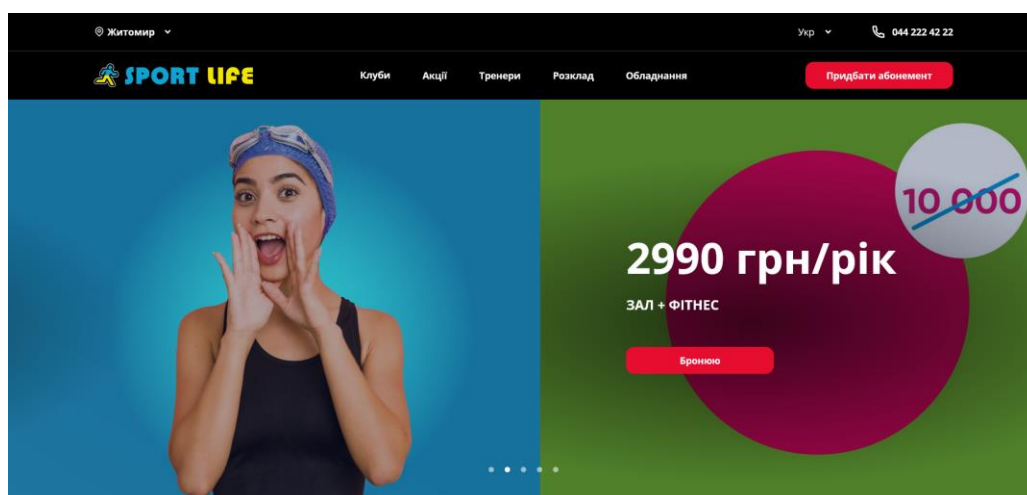


Рис. 1.3 – Головна сторінка сайту Sport Life

Переглядаючи даний веб-сайт можна відзначити його продуманість та зручність у використанні, вся необхідна інформація розподілена по вкладках, що дає змогу досить просто знайти всю необхідну інформацію. На сайті

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						14

представлено багато корисної інформації щодо роботи клубу, тренерів, розкладу групових занять та партнерів.

Останнім сайтом для порівняння став сайт фітнес-клубу «Lime Fitness».

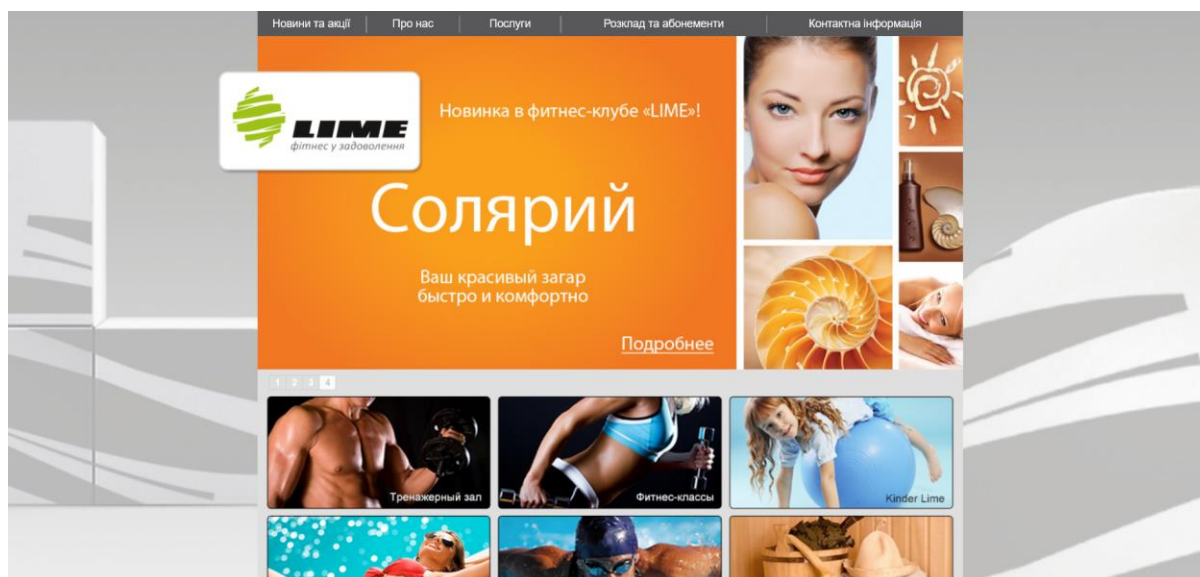


Рис. 1.4 – Головна сторінка сайту Lime Fitness

Цей фітнес-клуб за концепцією дуже схожий зі Sport Life. Переглядаючи сайт даного клубу можна помітити що його дизайн є застарілим, та не дуже приємним для сприйняття. Незважаючи на це, сайт залишається досить інформативним та його навігація є інтуїтивно зрозумілою.

Таблиця 1.1

Характеристика	TRX Pantera	Arena Gym	Sport Life	Lime Fitness
Особистий кабінет	Є	Немає	Немає	Немає
Перегляд абонементів	Немає	Немає	Є	Є
Перегляд тренерів	Є	Немає	Немає	Немає
Функціональність	Розширена	Базова	Розширена	Базова
Навігація	Інтуїтивна	Інтуїтивна	Інтуїтивна	Не інтуїтивна

В таблиці 1.1 порівняно певні важливі характеристики кожного із проаналізованих конкурентів. У висновку можна сказати що кожен з них має як плюси, так і мінуси, і всі вони є типовими представниками даної сфери. Але в

більшість з них являє собою прості лендінги на яких ми можемо просто переглядати базову інформацію про тренажерну залу таку як локація, але користувачу було б приємно бачити і більш корисну інформацію, таку як рекомендовані програми тренувань, та мати власний кабінет для перегляду історії занять та редагування інформації про себе. Саме тому я вирішив розробити такий веб додаток, який дозволить користувачам активно взаємодіяти з веб-сайтом.

1.3 Вибір архітектури веб-ресурсу

Надважливими якостями програмного продукту мають бути надійність, ефективність та масштабованість. Для дотримання цих якостей я вирішив використовувати клієнт-серверну архітектуру, тому що вона чи не найкраще підходить для таких проектів. Найважливішою перевагою цієї архітектури є розділення відповідальності, тобто клієнтська частина відповідає за відображення інформації, в той час як серверна частина відповідає за обробку запитів клієнтів, роботу з БД та забезпечення безпеки.

Також оптимізація роботи сайту досягається завдяки клієнт-серверній моделі, де клієнтська частина виконується на пристроях користувачів, а серверна – на фізичних серверах. Такий підхід дозволяє ефективно використовувати ресурси.

Дана модель повністю задовольняє потреби даного продукту, тому що забезпечує ефективну та надійну роботу, а також відповідає усім сучасним вимогам до веб-сайтів.

1.4 Обґрунтування вибору засобів та вимог до програмного забезпечення

Вивчаючи популярні технології для розробки веб-сайтів було обрано фреймворк Django^[11]. Завдяки цій технології був в короткі строки розроблений веб-сайт тренажерної зали.

Django – це фреймворк Python який використовується для створення невеликих та середніх за розміром веб-сайтів для електронної комерції. Завдяки великій кількості вбудованих функцій та класів Django дозволяє розробляти веб-сайти швидко та надійно. Django має багато вбудованих модулів, таких як `django.auth` або `django.admin` які значно спрощують деякі процеси розробки, і дозволяють сконцентруватися на розробці саме бізнес логіки, а не на розв’язанні тривіальних задач. Також Django має дуже зручну ORM для роботи з базою даних, що дозволяє не писати SQL код. Це дуже спрощує написання складних запитів в базу даних, та дозволяє оптимізувати деякі складні запити.

Висновки до розділу 1

В першому розділі були визначені та розділені на пункти задачі на виконання кваліфікаційної роботи. Слідуючи цим пунктам ми можемо успішно реалізувати веб-сайт тренажерної зали.

У другій частині першого розділу було розглянуто веб-сайти конкурентів, та порівняно їх за певними критеріями. Аналіз конкурентів дав змогу зрозуміти що обов'язково потрібно реалізувати, а чого краще уникати.

Крім того, була обрана найбільш підходяща архітектура для розробки веб-сайту, а саме, клієнт-серверна. Вона забезпечує розподіл задач між клієнтською та серверною частинами.

В кінці розділу було розглянуто засоби програмного забезпечення для реалізації веб-сайту. Було обрано Django через його зручність та можливість зосередити зусилля над написанням бізнес-логіки.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						18

РОЗДІЛ 2. АРХІТЕКТУРА І ПОБУДОВА ВЕБ-САЙТУ ТРЕНАЖЕРНОЇ ЗАЛИ

2.1 Визначення варіантів використання та структури системи

Веб-сайт створюється для того, щоб в подальшому його можна було використовувати для реального тренажерного залу, що дасть можливість залучати нових клієнтів та збільшити впізнаваність залу.

Для початку визначимо вимоги користувачів.

Адміністратор:

- Редагування та додавання вмісту сайту;
- Додавання та редагування користувачів;
- Перегляд усієї інформації на сайті;

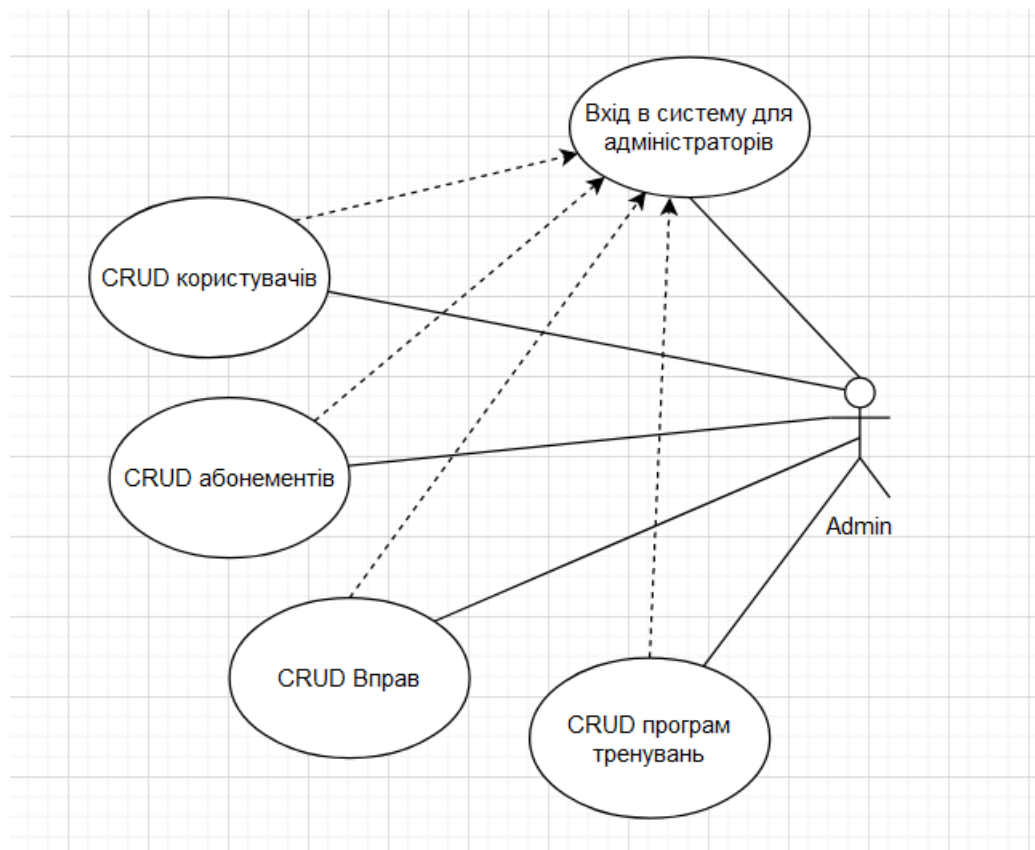


Рис. 2.1 – Діаграма варіантів використання для адміністратора

Зареєстрований користувач:

- Перегляд вмісту сайту;
- Перегляд списку тренерів;
- Перегляд списку програм;

- Перегляд детальної інформації про тренера;
- Перегляд детальної інформації про програму;
- Перегляд детальної інформації про вправу;
- Редагування інформації про себе;

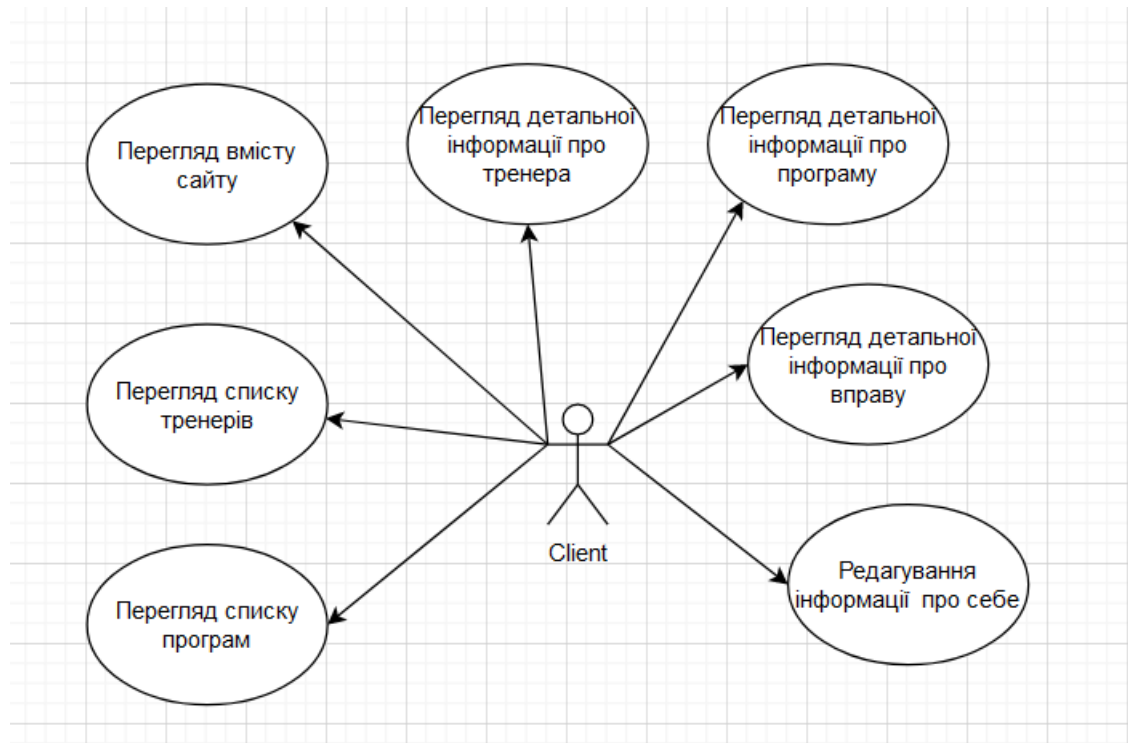


Рис. 2.2 – Діаграма варіантів використання для зареєстрованого користувача

Гість:

- Перегляд вмісту сайту;
- Перегляд списку тренерів;
- Перегляд детальної інформації про тренера;

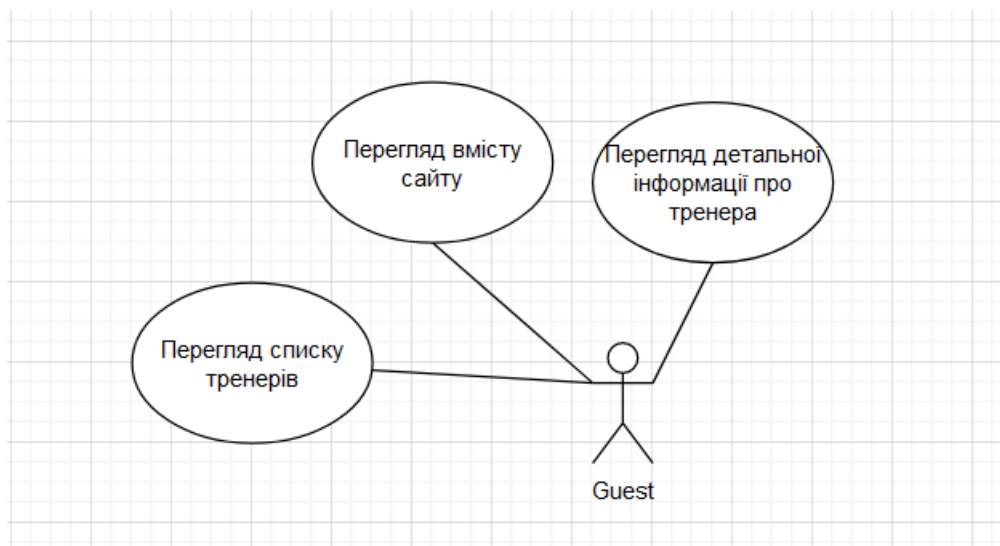


Рис. 2.3 – Діаграма варіантів використання для гостя

Тренер:

- Перегляд вмісту сайту;
- Перегляд списку тренерів;
- Перегляд списку програм;
- Перегляд детальної інформації про тренера;
- Перегляд детальної інформації про програму;
- Перегляд детальної інформації про вправу;
- Додавання вправ;
- Додавання програм;
- Редагування інформації про себе;

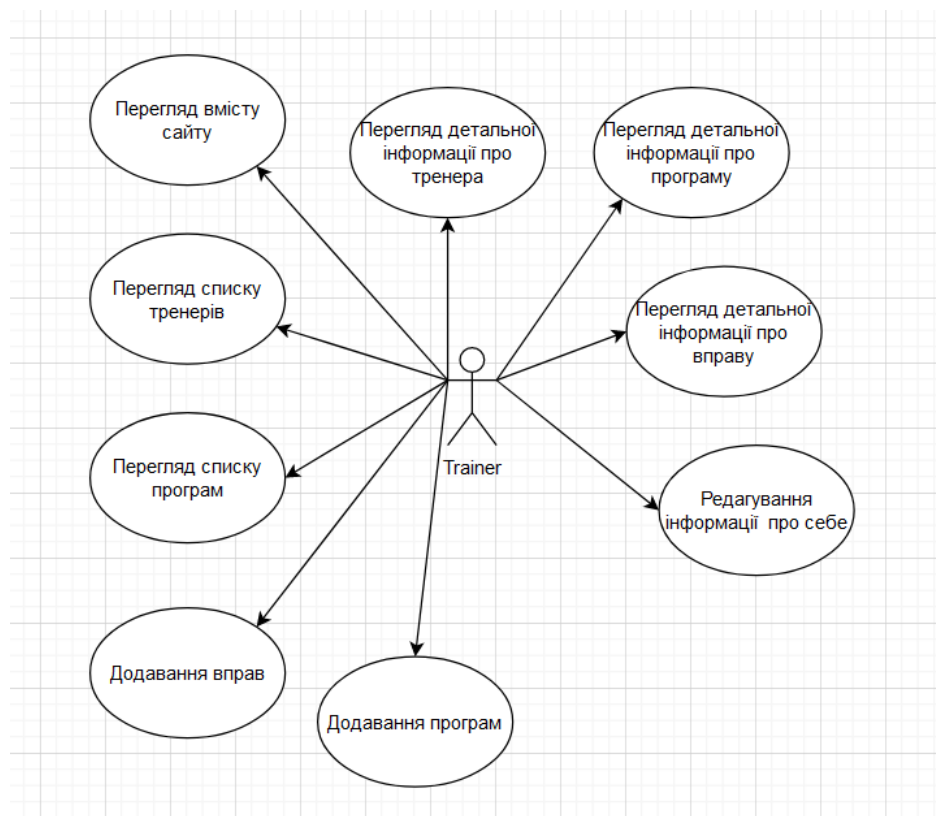


Рис. 2.4 – Діаграма варіантів використання для тренера

Далі розглянемо вимоги до веб-сайту

Функціональні вимоги:

- **Авторизація:** клієнти та тренери повинні мати змогу увійти в уже наявний акаунт;
- **Валідація:** дані, які відправляються на сервер повинні проходити валідацію;
- **Структуризація:** всі дані повинні бути структуровані для зручності перегляду;
- **Повноцінність:** користувачі повинні мати змогу бачити детальну інформацію про тренера, програму та вправу;

Системні вимоги:

- Пристрій на якому буде використовуватись сайт має підтримувати браузери останніх версій та мати вихід в інтернет;
- Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1 Гб;

- Дисковий простір, необхідний для збереження програми та файлів даних не більше 300 Мб для робочої станції та 20 Гб;
- Двоядерний процесор з тактовою частотою не менше 1.5 ГГц;

Не функціональні вимоги:

- Інтерфейс має бути простим та інтуїтивно зрозумілим;
- Час відповіді на запити не має перевищувати 4 секунди;
- Час для засвоєння має бути не більше ніж 1 година;

Розглянемо контролери backend частини:

- MainPage – контролер який відповідає за відображення головної сторінки;
- AboutPage – контролер який відповідає за відображення сторінки з контактною інформацією;
- ProgramsPage – контролер який відповідає за відображення сторінки з програмами;
- ProgramPage – контролер який відповідає за відображення сторінки програми;
- PricesPage – контролер який відповідає за відображення сторінки абонементів;
- TrainersPage – контролер який відповідає за відображення сторінки тренерів;
- SingleTrainer – контролер який відповідає за відображення сторінки тренера;
- AddExercise – контролер який відповідає за відображення сторінки додавання вправи;
- AddProgram – контролер який відповідає за відображення сторінки додавання програми;
- Profile – контролер який відповідає за відображення сторінки профілю користувача;
- user_login – контролер який відповідає за обробку форми авторизації;

- `user_logout` – контролер який відповідає за обробку форми виходу з акаунту;
- `willing_person` – контролер який відповідає за обробку форми запису на пробне заняття;
- `add_exercise` – контролер який відповідає за обробку форми додавання вправи;
- `add_p` – контролер який відповідає за обробку форми додавання програми;

Також є моделі даних, вони потрібні для здійснення різних взаємодій з даними.

І ще одним важливим елементом є маршрути. В них прописуються шляхи до потрібних контролерів для подальшої взаємодії з ними.

2.2 Розробка бази даних веб-сайту

База даних є важливим елементом для кожного веб-сайту. В ній зберігаються усі необхідні дані, які надалі можуть змінюватися в залежності від взаємодії користувачів з сайтом.

У проєкті було вирішено використовувати базу даних SQLite, так як вона не потребує окремого сервера для розгортання.

Фрагмент з файлу `settings.py`, який відповідає за підключення до бази даних:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Після того, як база даних була підключена потрібно створити моделі виконуючи команди «`py manage.py makemigrations`» для створення міграції та «`py manage.py migrate`» для застосування цієї міграції в терміналі.

Після цього було створено усі необхідні таблиці, та поля в них за допомогою ORM. Нижче представлений код для генерації однієї з таблиць бази даних.

```
class Program(models.Model):
    name = models.CharField(max_length=100)
    date = models.DateField(auto_now_add=True)
    duration = models.IntegerField(default=60)
    author = models.ForeignKey('CustomUser', on_delete=models.CASCADE)
    exercise = models.ManyToManyField('Exercise', related_name='exercises')

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('program', kwargs={'pk': self.pk})
```

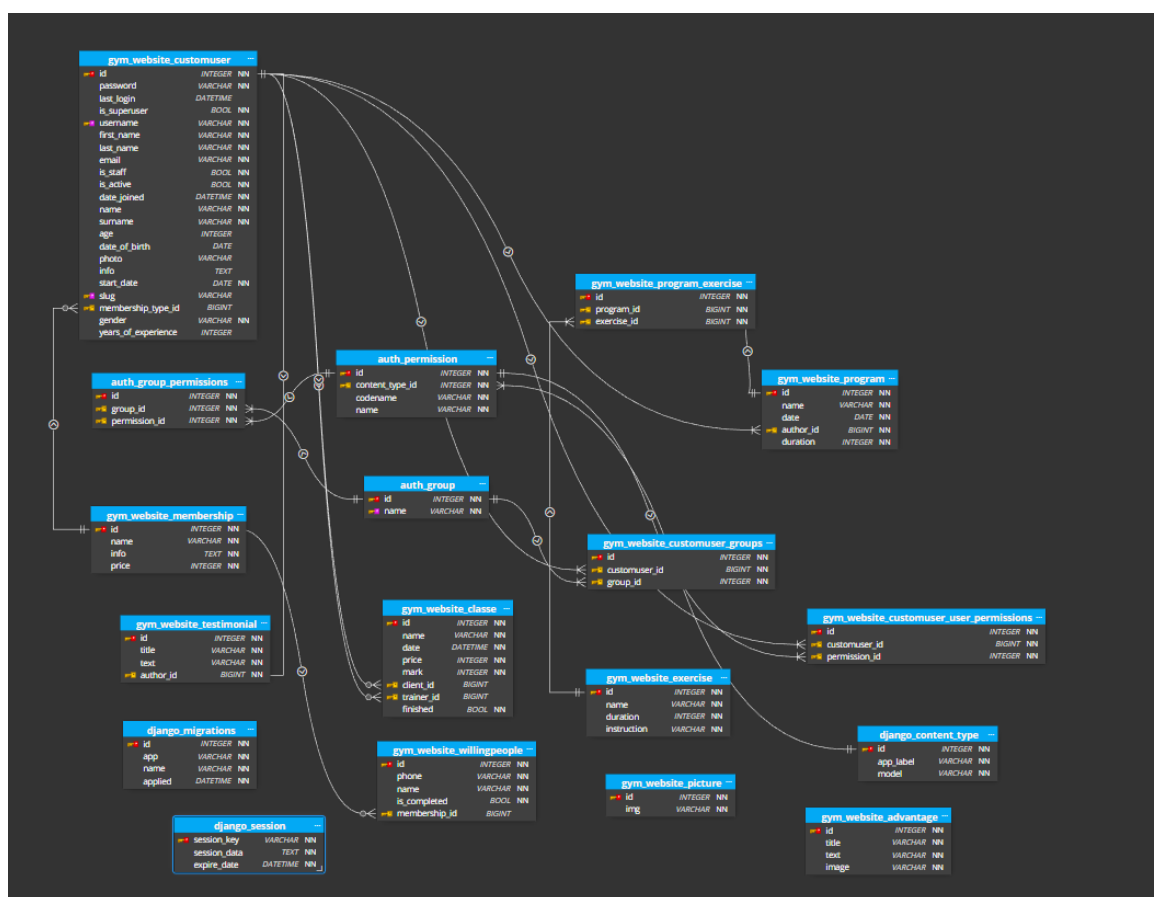


Рис. 2.5 – Загальний вигляд діаграми бази даних

2.3 Проектування та реалізація алгоритмів роботи веб-сайту

Основним компонентом, що описує використання ресурсу, є діаграма діяльності, яка містить опис різних процесів, виконуваних користувачами.

На наведеному нижче зображенні 2.6 показана загальна діаграма діяльності всіх користувачів. Ця діаграма ілюструє спосіб взаємодії користувачів з нашим вебресурсом, залежно від їх ролі та рівня доступу до веб-сайту тренажерної зали.

Згідно з діаграмою, користувачі з роллю «Адміністратор» мають доступ до адмінпанелі, тоді як інші користувачі з ролями «Гість», «Користувач» або «Тренер» зможуть користуватися лише frontend частиною. Це реалізовано за допомогою клієнт-серверної архітектури, яка дозволяє розділити наш веб-додаток на дві окремі частини, пов'язані між собою.

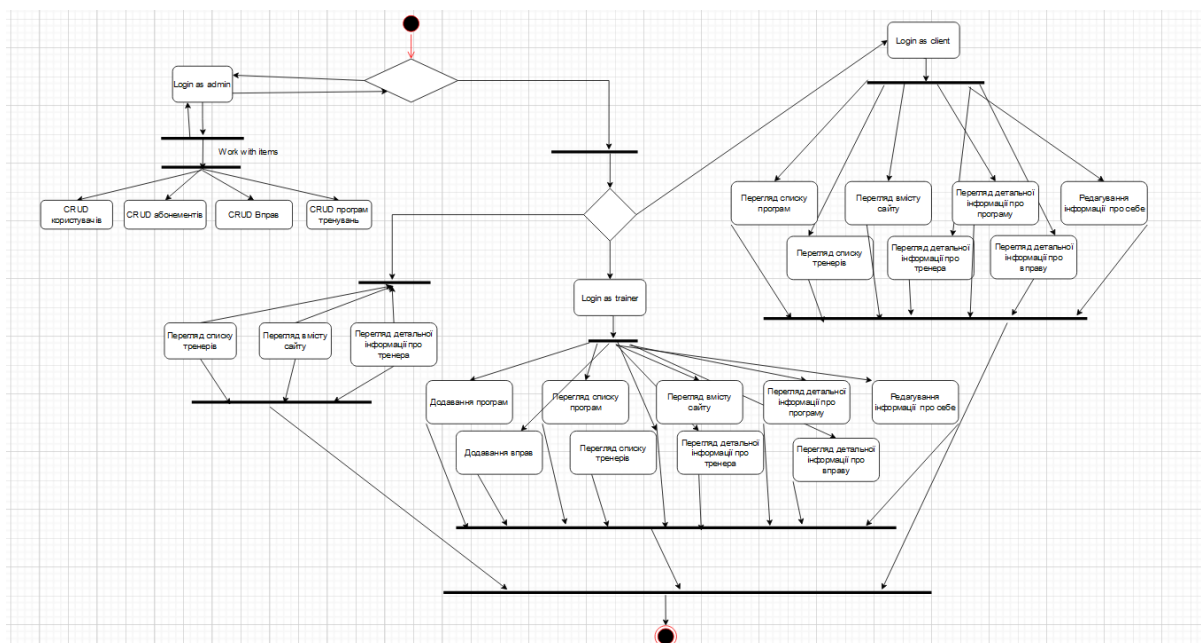


Рис. 2.6 – Загальна діаграма діяльності дій користувачів.

На діаграмі 2.6 представлена загальна схема діяльності всіх користувачів, яка демонструє різні ролі на сайті. Основна роль належить адміністратору, який відповідає за наступні завдання: наповнення сайту, коректне відображення всіх елементів та багато іншого.

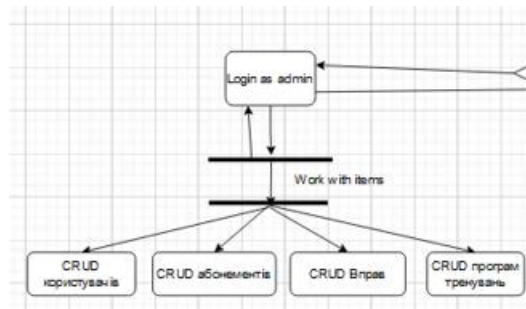


Рис. 2.7 – Діаграма діяльності користувача з роллю адміністратора

Також на діаграмі є роль «Guest». Ця роль автоматично присвоюється усім користувачам які зайшли на сайт. В такого користувача дуже обмежений доступ до інформації. Він може переглядати вміст сайту, список тренерів та детальну інформацію про конкретного тренера.

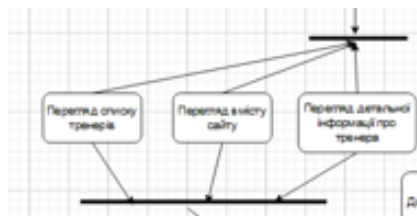


Рис. 2.8 – Діаграма діяльності користувача з роллю гостя

Роль «Client» належить користувачу який є клієнтом даного тренажерного залу. Він може переглядати вміст сайту, список тренерів та програм і переглядати детальну інформацію про тренера, програму, та вправу. Також він може редагувати особисту інформацію у кабінеті.

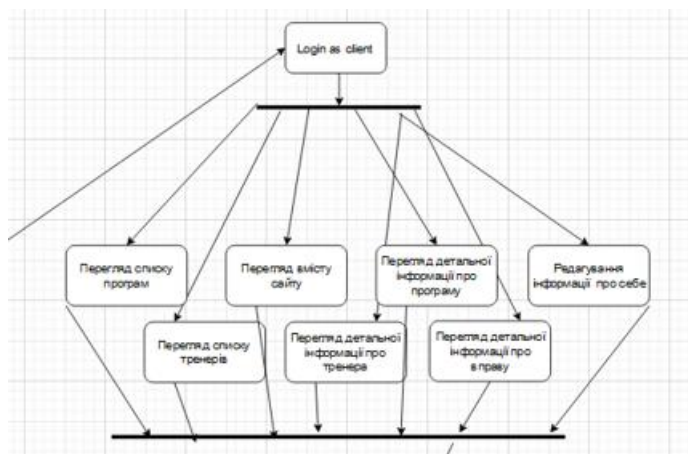


Рис. 2.9 – Діаграма діяльності користувача з роллю клієнта

І ще однією роллю, є «Trainer». Він має всі можливості що і клієнт, але може також додавати вправи та програми.

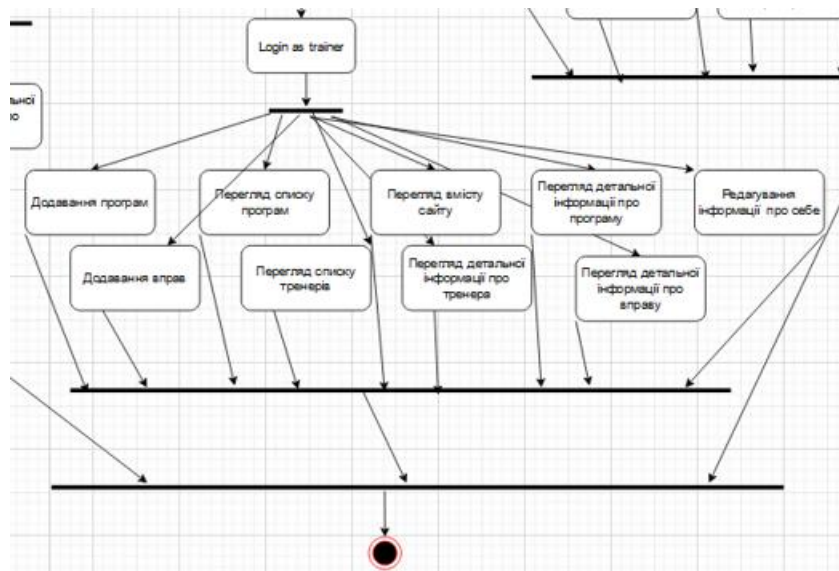


Рис. 2.10 – Діаграма діяльності користувача з роллю тренера

При розробці веб-сайту була створена діаграма класів, яка наведена на рисунку 2.11.

На діаграмі класів представлено 11 основних класів таких як:

- Permission;
- Group;
- CustomUser;
- Testimonial;
- Membership;
- Picture;
- Advantage;
- Ecercise;
- Program;
- WillingPeople;
- Classe;

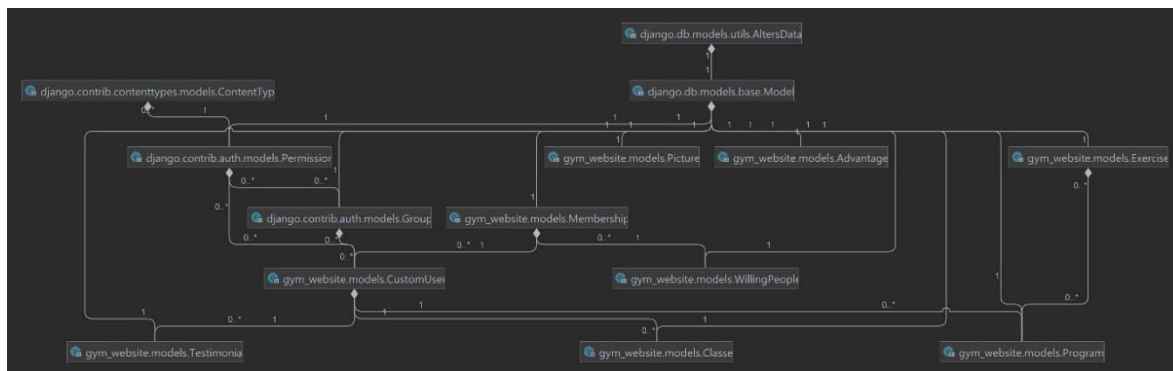


Рис. 2.11 – Діаграма класів

На діаграмі 2.11 можна ознайомитися з усіма класами, які реалізовані в проєкті. Це дасть змогу використати ці класи при реалізації певних функцій та компонентів.

2.4 Реалізація функціонала веб-сайту тренажерної зали

Для належного опису функціоналу нашого програмного забезпечення важливо усвідомити, що наша система складається з двох частин: фронтенду (frontend) та бекенду (backend). Ці компоненти взаємопов'язані, тому якщо ми реалізуємо певний метод на бекенді, але не забезпечимо його зв'язок з фронтендом, користувач не зможе скористатися цим функціоналом.

Для реалізації програмного продукту був обраний патерн MVC, а саме його варіація яку використовує Django (MTV). MVC - це скорочення від Model-View-Controller (Модель-Представлення-Контролер). Це популярний архітектурний шаблон, який використовується для розробки програмного забезпечення, особливо у веб-додатках^[27]. У шаблоні MVC функціональність програми поділяється на три головні компоненти:

- **Модель (Model):** Відповідає за управління даними та бізнес-логікою програми. Модель представляє собою структуру даних або клас, який зберігає та обробляє інформацію, а також забезпечує методи для доступу до цих даних^[27].
- **Представлення (View):** Відповідає за відображення даних користувачеві та

обробку користувацького інтерфейсу. Представлення отримує дані з моделі та відображає їх у відповідному форматі, наприклад, у вигляді веб-сторінки чи графічного інтерфейсу. Представлення також може надсилати дії користувача контролеру^[27].

- Контролер (Controller): Відповідає за обробку вхідних подій від користувача та взаємодію з моделлю та представленням. Контролер приймає вхідні дані від користувача через представлення, обробляє їх і виконує відповідні дії, такі як оновлення моделі чи зміна представлення. Контролер також може відправляти команди моделі та представленню для оновлення стану додатку^[27].

Це рішення дозволить структурувати весь код для зручної роботи з ним, та легкої подальшої підтримки нашого веб-додатку.

Для того, щоб все працювало належним чином нам необхідно налаштувати відповідні маршрутизатори (роути) для наших методів^[9]. В проєкті є кілька місць де вони прописуються. Перше це файл `urls.py`, який відповідає за роути, які використовуються для всього проєкту, другий файл це `gym_website/urls.py`, який використовується для нашого додатку. Також в проєкті присутні роути для навігації по адмін-панелі, але вони створюються автоматично, тому не будемо тут їх розглядати.

Фрагмент з файлу `urls.py`:

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('gym_website.urls'))
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Фрагмент з файлу gym_website/urls.py:

```
from django.urls import path
from .views import *

urlpatterns = [
    path("", MainPage.as_view(), name='home'),
    path('about/', AboutPage.as_view(), name='about'),
    path('trainers/', TrainersPage.as_view(), name='trainers'),
    path('programs/', ProgramsPage.as_view(), name='programs'),
    path('trainer/<int:pk>', SingleTrainer.as_view(), name='trainer'),
    path('program/<int:pk>', ProgramPage.as_view(), name='program'),
    path('pricing/', PricesPage.as_view(), name='prices'),
    path('login/', user_login, name='login'),
    path('logout/', user_logout, name='logout'),
    path('willing/', willing_person, name='willing'),
    path('add-exercise/', AddExercise.as_view(), name='add_exercise'),
    path('add-program/', AddProgram.as_view(), name='add_program'),
    path('add-p/', add_p, name='add_p'),
    path('profile/<int:pk>', Profile.as_view(), name='profile')
]
```

В кожному роуті прописано маршрут за яким має викликатися певний метод, та ім'я роуту для зручності використання.

Для кращого розуміння розглянемо роут «about/». Цей роут використовується для відображення інформаційної сторінки про наш спортзал. Тобто підчас переходу по адресі 'http://localhost/about' буде спрацьовувати клас «AboutPage» для генерації цієї сторінки.

Методи для генерації сторінок та обробки форм знаходяться у файлі views.py^[5]. Для прикладу розглянемо один метод який слугує для відображення головної сторінки нашого веб-сайту.

Фрагмент з файлу views.py:

```
class MainPage(TemplateView):
    template_name = 'gym_website/main_page.html'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['advantages'] = Advantage.objects.all()
        context['trainers'] =
CustomUser.objects.filter(groups__name='Trainer').order_by("?")[:4]
        context['gallery'] = Picture.objects.all()
        context['form'] = UserAuthenticationForm()
```

```
context['willing_form'] = WillingForm()
return context
```

Вище ми бачимо клас для генерації головної сторінки сайту. Тут ми можемо бачити змінну `template_name` для призначення шаблону куди буде передаватись інформація. Також ми можемо побачити метод `get_context_data` який потрібен для передачі додаткових змінних у шаблон.

Для виводу інформації використовуються html файли. Загальну структуру такого файлу розберемо на прикладі файлу `programs.html`.

Фрагмент з файлу `programs.html`

```
{% extends 'base.html' %}
{% block title %}Programs{% endblock %}
{% block content %}
    <div style="min-height: 79vh" class="container">
        <div class="mt-4">
            <h2>Програми тренувань</h2>
        </div>
        <div class="mt-4">
            <ul class="list-group list-group-flush">
                {% for program in programs %}
                    <li class="list-group-item">
                        <div class="d-flex justify-content-between">
                            <h5><a href="{{ program.get_absolute_url }}">{{ program.name }} - {{
program.author.surname }}</a></h5>
                            <h5><a href="{{ program.author.get_absolute_url }}">{{ program.author
}}</a></h5>
                        </div>
                    </li>
                {% endfor %}
            </ul>
        </div>
    </div>
{% endblock %}
```

Ми можемо побачити html файл з деякими змінами. Даний файл наслідується від базового шаблону. Також можна помітити що основний код знаходиться в блоку контент. Всередині цього блоку ми бачимо цикл який проходиться по масиву значень які ми передали у класі, а далі з кожного елементу масиву беруться певні дані та вставляються в потрібні місця на сторінці.

Висновки до розділу 2

У другому розділі були розглянуті всі етапи розробки веб-сайту тренажерної зали. Ми розпочали з аналізу діаграм варіантів використання, були розглянуті ролі адміністратора, зареєстрованого користувача, гостя та тренера.

Далі був описаний процес розробки бази даних, а саме, створення самої бази даних та її підключення до серверної частини. Було розглянуто створення таблиць, додавання полів та призначення їм відповідних типів та назв.

Далі був представлений приклад створених моделей та контролерів. Контролери слугують для опису методів, які використовуються для реалізації функціоналу як на серверній, так і на клієнтській частинах.

Також було пояснено як взаємодіють між собою клієнтська та серверна частини.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ ТРЕНАЖЕРНОЇ ЗАЛИ

3.1 Порядок встановлення та налаштування веб-додатку.

Фреймворк Django надає вбудований сервер, також ми використовуємо базу даних SQLite, для якої не потрібен сервер, тому ми можемо використовувати готове рішення. Для запуску нам потрібно перейти в теку з проектом та в консолі прописати команду «`python manage.py runserver`» яка запустить наш сайт і він буде доступний за адресою `http://127.0.0.1:8000/`.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-сайтом

Для того, щоб користувач міг використовувати сайт йому потрібно перейти за посиланням на цей сайт, або відкрити збережену вкладку, якщо користувач раніше відвідував даний сайт, після чого він потрапить на головну сторінку сайту.

На головній сторінці ми можемо почитати про переваги нашого спортзалу, переглянути профілі тренерів, подивитись галерею та подивитися локацію на мапі. Також користувач може заповнити форму для запису на пробне заняття.

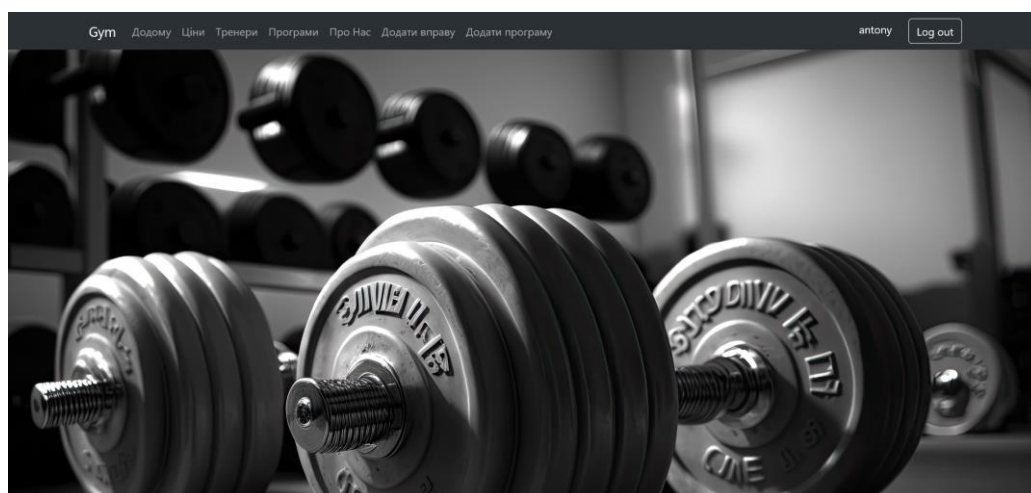


Рис. 3.1 – Головна сторінка сайту

Наші переваги

Приєднуйтеся до нас сьогодні і розпочніть свою пригоду до міцного тіла та духовного благополуччя.

Ласкаво просимо до нашого спортзалу, де ви зможете зануритися у світ активності та здорового способу життя. Наша команда професіоналів стежить за вашим прогресом і забезпечить вам високоякісні тренування та розумне планування тренувань.

[Записатись](#)

Гнучкий графік
Можливість тренуватися в зручний час.

Розташування
Зручне розташування для доступності.

Паркінг
Просторе паркування для клієнтів.

Система знижок
Привабливі знижки для постійних клієнтів.

Наші тренери



Іван Григоренко



Олександр Петренко



Олена Петренко

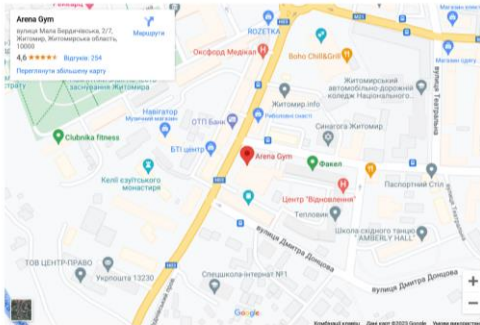


Володимир Ковальчук

Галерея



Розташування



[Гум](#) [Додому](#) [Ціни](#) [Тренери](#) [Програми](#) [Про нас](#)

Рис. 3.2 – Головна сторінка сайту

Для входу в особистий акаунт потрібно натиснути на кнопку «Login» та заповнити форму у модальному вікні.

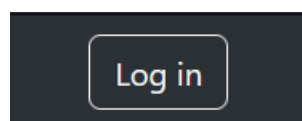


Рис. 3.3 – Кнопка Входу

Після цього користувачу треба ввести дані для входу.

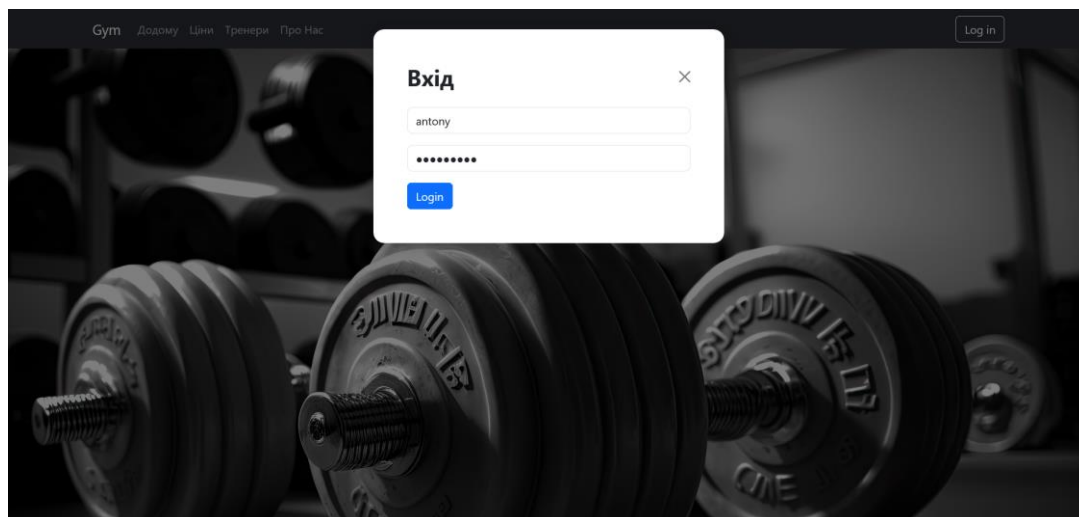


Рис. 3.4 – Вікно входу

Після натискання на кнопку «Login» введені дані перевіряються на відповідність, і при знаходженні відповідного користувача в базі даних користувач буде залогінений і перенаправлений на головну сторінку.

Також на головній сторінці користувач може заповнити форму для того щоб записатись на пробне заняття. Для цього потрібно натиснути на кнопку Записатись на головній сторінці.

Наші переваги

Приєднуйтесь до нас сьогодні і розпочніть свою пригоду до міцного тіла та духовного благополуччя.

Ласкаво просимо до нашого спортзалу, де ви зможете зануритися у світ активності та здорового способу життя. Наша команда професіоналів стежить за вашим прогресом і забезпечить вам високоякісні тренування та розумне планування тренувань.

[Записатись](#)



Гнучкий графік

Можливість тренуватися в зручний час.



Паркінг

Просторе паркування для клієнтів.



Розташування

Зручне розташування для доступності.



Система знижок

Привабливі знижки для постійних клієнтів.

Рис. 3.5 – Кнопка для запису

Гym

Додому

Ціни

Тренери

Програми

Про Нас

Додати вправу

Додати програму

Привіт, Антон



Антон

Жуковский

30

Man

07.06.1993

Досвідчений тренер з багаторічним стажем роботи у тренажерному залі. Він спеціалізується на функціональному тренуванні та силовому тренінгу. Максим завжди надає особисту увагу своїм клієнтам, допомагаючи їм досягати своїх фітнес-цілей. Він є експертом у розробці індивідуальних програм тренувань, які враховують потреби та можливості кожного клієнта.

Оновити

Ваші заняття

Назва	Дата	Тренер	Вартість	Статус
Петренко Сидоренко	June 23, 2023, 5 p.m.	Володимир Ковальчук	100	✓

На даній сторінці користувач може переглянути та змінити інформацію про себе, а також подивитися статус своїх тренувань. Для того, щоб оновити інформацію, треба ввести її в потрібне поле та натиснути кнопку оновити. Після натискання кнопки введена інформація валідується та за умови що вся інформація коректна, вона перезаписується, і ми потрапляємо на сторінку тренера.

Gym

Додому

Ціни

Тренери

Програми

Про Нас

Додати вправу

Додати програму

antony

Log out

Антон Жук

Вік:	30
Дата народження:	June 7, 1993
Досвід:	2

Досвідчений тренер з багаторічним стажем роботи у тренажерному залі. Він спеціалізується на функціональному тренуванні та силовому тренінгу. Максим завжди надає особисту увагу своїм клієнтам, допомагаючи їм досягати своїх фітнес-цілей. Він є експертом у розробці індивідуальних програм тренувань, які враховують потреби та можливості кожного клієнта.

Програми тренера

#	Назва	Дата створення	Час виконання .xh
8	Програма 1	June 21, 2023	60
9	Програма 2	June 21, 2023	40

Gym

Додому

Ціни

Тренери

Програми

Про Нас

Особистий кабінет – це сторінка на яку може зайти будь-який користувач для перегляду детальної інформації про тренера та його програми, але

користувач з роллю гостя не зможе бачити програми тренера, а лише інформацію про нього.

Тепер розглянемо сторінку з цінами на абонементи.

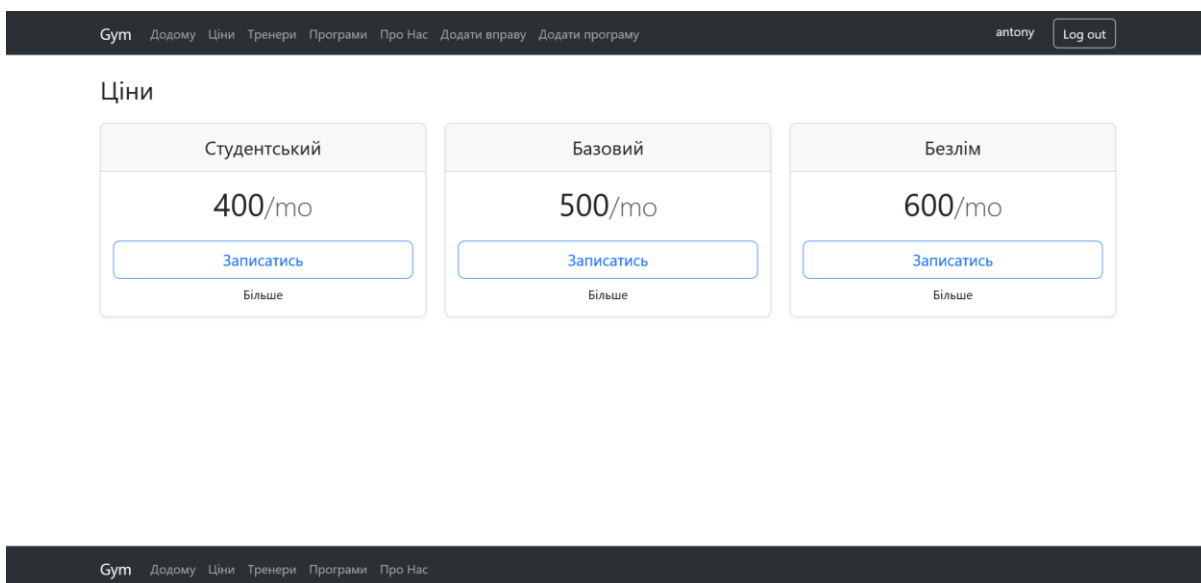


Рис. 3.10 – Сторінка з абонементом

На цій сторінці користувач може ознайомитись з цінами на абонементи та почитати детальнішу інформацію про них. Для того, щоб переглянути детальну інформацію про абонемент користувачу потрібно натиснути на надпис «Більше».

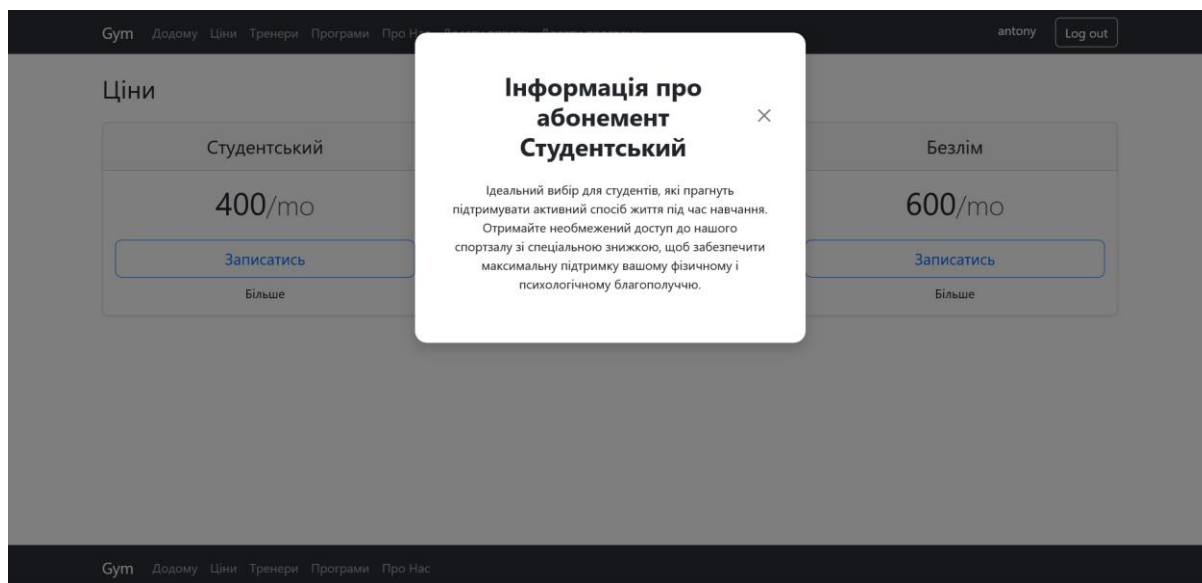


Рис. 3.11 – Детальна інформація про абонемент

Користувач також може натиснути на кнопку «Записатись» для того, щоб записатись на заняття.

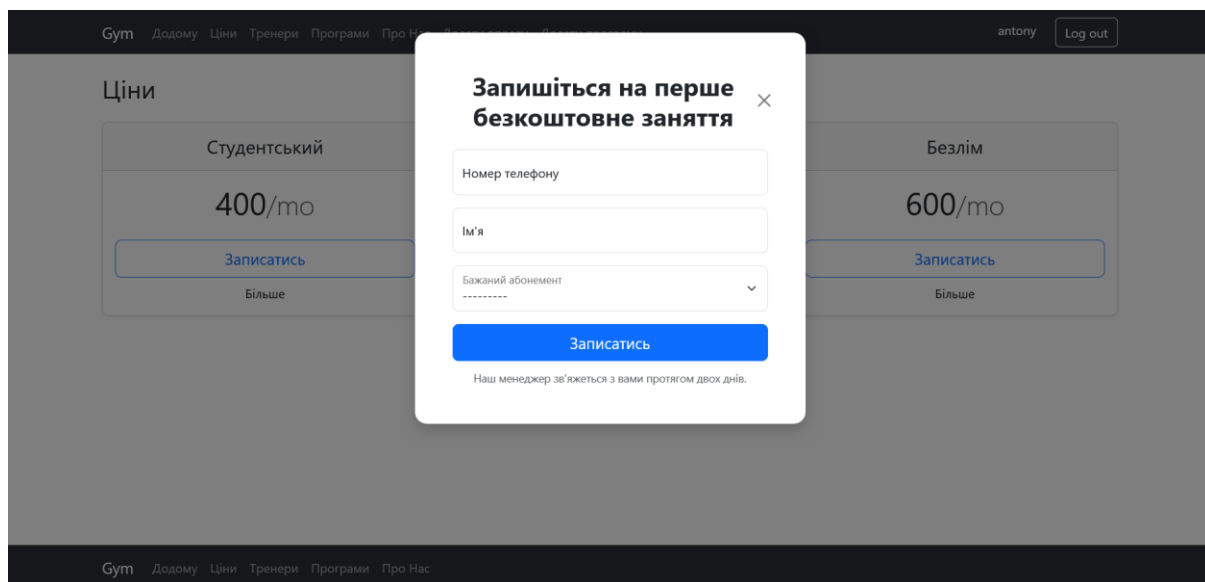


Рис. 3.12 – Вікно запису на заняття

Наступною розглянемо сторінку зі списком усіх тренерів. На ній користувач може переглянути усіх тренерів. Для того, щоб перейти на сторінку треба користувачу потрібно натиснути на ім'я тренера.

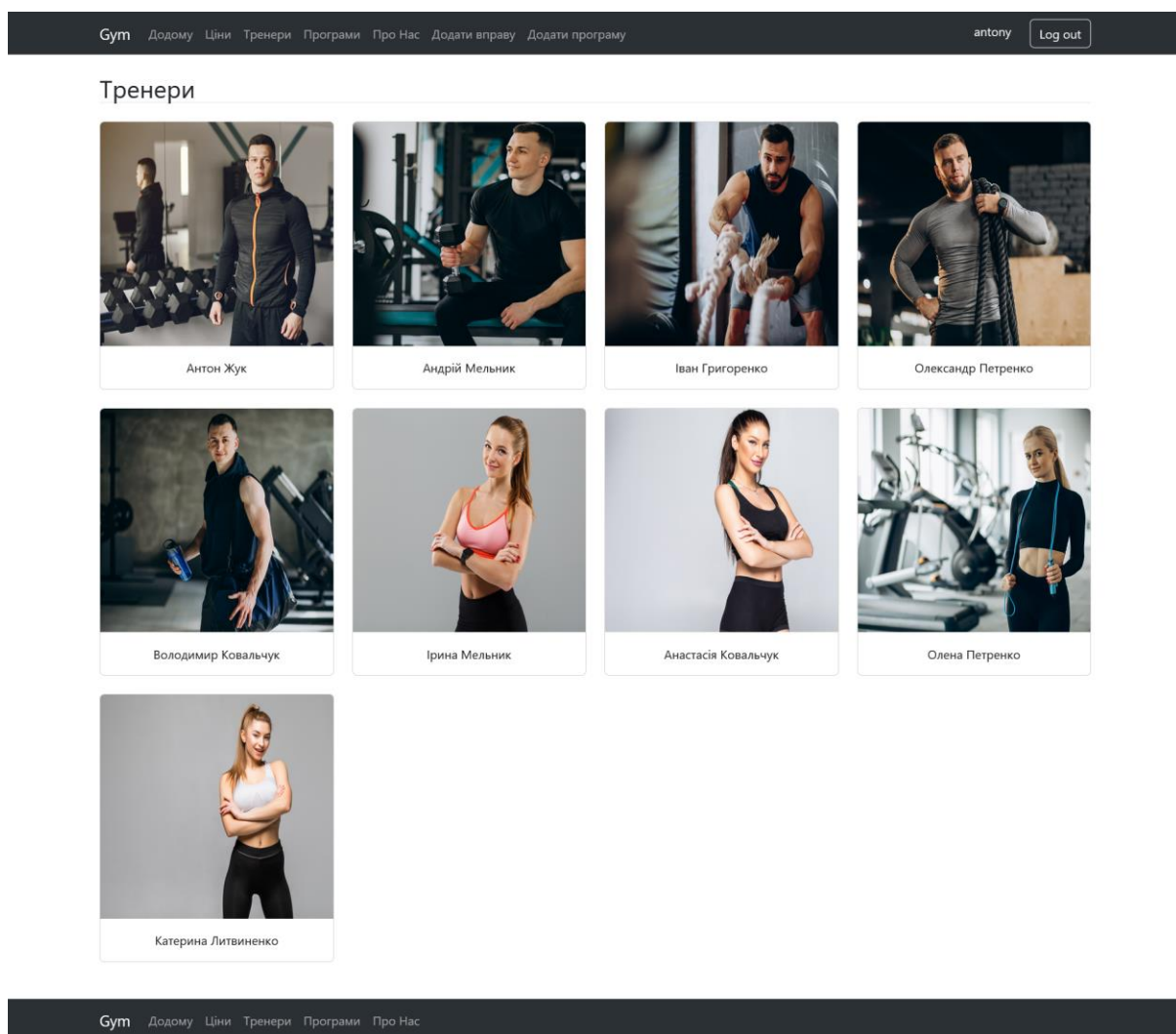


Рис. 3.13 – сторінка перегляду списку тренерів

Далі ми розглянемо сторінку зі списком програм тренувань.

Gym Додому Ціни Тренери Програми Про Нас Додати вправу Додати програму		antony	Log out
Програми тренувань			
Програма 1 - Жук	Антон Жук		
Програма 2 - Жук	Антон Жук		
Програма 1 - Литвиненко	Катерина Литвиненко		
Програма 2 - Литвиненко	Катерина Литвиненко		

Gym Додому Ціни Тренери Програми Про Нас
--

Рис. 3.14 – сторінка перегляду списку програм тренувань

На даній сторінці ми можемо побачити список усіх програм тренувань та їх авторів. З даної сторінки ми можемо перейти на сторінку програми або автора, для цього потрібно натиснути на назву або автора відповідно. Перейдемо на сторінку програми.

Gym Додому Ціни Тренери Програми Про Нас Додати вправу Додати програму		antony	Log out
Програма 1			
#	Назва	Відеоінструкція	Час виконання .хв
2	Віджмання	Посилання	10
3	Підтягування	Посилання	10
4	Присідання	Посилання	15
5	Стрибки на скакалці	Посилання	10

Gym Додому Ціни Тренери Програми Про Нас
--

Рис. 3.15 – сторінка перегляду програми тренувань

На даній сторінці ми можемо бачити список вправ в даній програмі тренувань. Для кожної вправи ми можемо бачити назву, посилання на відеоінструкцію та рекомендований час виконання.

Перейдемо на сторінку з контактною інформацією.

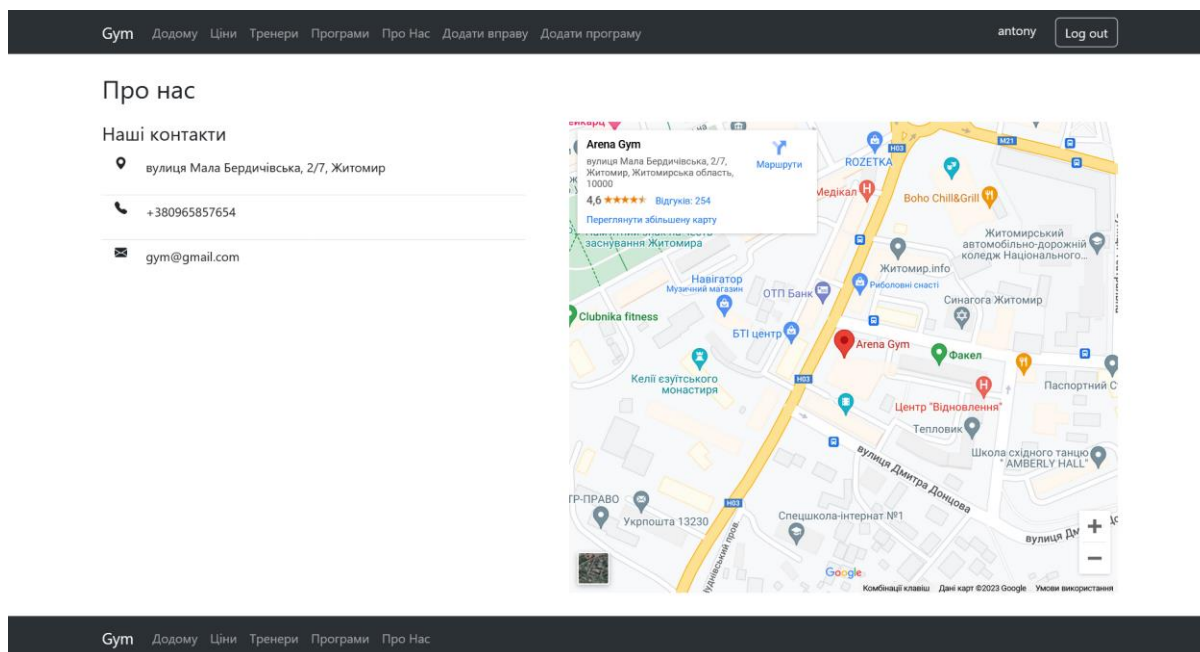


Рис. 3.16 – сторінка перегляду програми тренувань

Тепер перейдемо до сторінок доступних лише для тренерів. Першою з них буде сторінка додавання вправи.

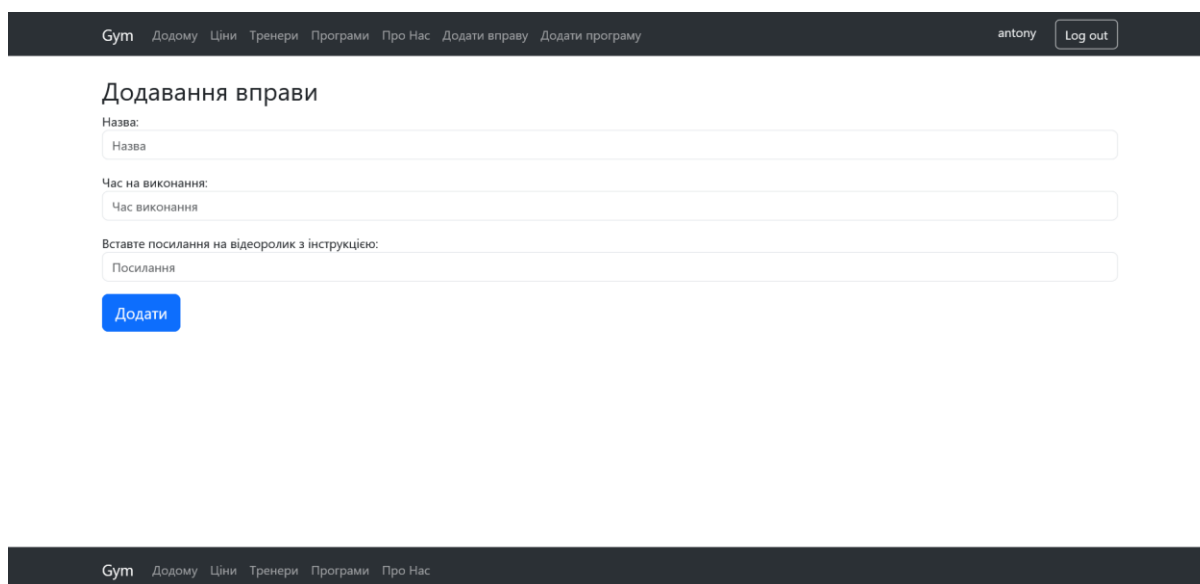


Рис. 3.17 – сторінка додавання вправи

Для того щоб додати вправу потрібно заповнити потрібні поля та натиснути кнопку «Додати». Спробуємо ввести не валідне значення в поле часу і протестуємо поведінку нашого додатку.

Назва:

тест

- Введіть ціле число.

Час на виконання:

абвг

Вставте посилання на відеоролик з інструкцією:

<https://www.youtube.com/watch?v=dFVxGRekRSg>

Додати

Рис. 3.18 – Вивід помилки про невалідний тип даних

Після успішного додання вправи відбудеться переадресація на головну сторінку.

Тепер перейдемо до сторінки додавання програми тренувань.

Gym Додому Ціни Тренери Програми Про Нас Додати вправу Додати програму antony Log out

Додавання програми

Назва програми:

Назва

Оберіть одну або більше вправ:

Віджмання
Підтягування
Присідання
Стрибки на скакалці

Приблизна тривалість:

60

Додати

Gym Додому Ціни Тренери Програми Про Нас

Рис. 3.19 – сторінка додавання програми тренувань

Для того, щоб додати програму тренувань потрібно заповнити всі поля.
Для того, щоб додати більше однієї вправи до програми потрібно натиснути Ctrl

та обрати потрібні вправи. Тепер спробуємо не заповнити жес із полів, і подивимось що з цього вийде.

Рис. 3.20 – Помилка про пустий рядок

Тепер перейдемо до огляду засобів адміністратора. Для того щоб увійти в адмін-панель потрібно перейти по посиланню <http://127.0.0.1:8000/admin/login/?next=/admin/> та ввести логін та пароль.

Рис. 3.21 – сторінка входу в адмін-панель.

Користувач який не має ролі адміністратора не зможе увійти в адмін-панель.

Після авторизації адміністратор може виконувати операції з записами у базі даних з адмін-панелі.

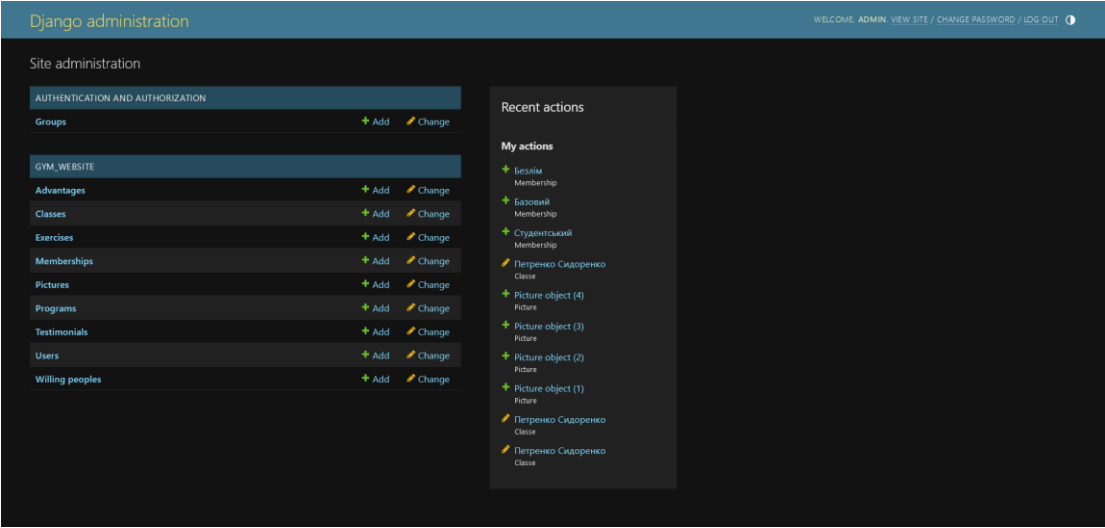


Рис. 3.22 – головна сторінка адмін-панелі

Висновки до розділу 3

В третьому розділі я розглянув інтерфейс реалізованого веб-сайту тренажерного залу. Більше того, я розглянув сторінки, які є на сайті, та які може використовувати користувач. Також, я розглянув інтерфейс, з яким буде взаємодіяти користувач, перевінив його зручність та простоту у використанні. Також було протестовано реакцію на введення не валідних даних та не введення даних взагалі.

В цілому веб ресурс має бути доопрацьований. Ось список із того що потрібно доопрацювати в першу чергу:

- Впровадити можливість оцінки занять;
- Додати можливість писати відгуки на тренерів та програми тренувань;
- Переробити клієнтську частину додатка з використанням таких технологій як React та TypeScript;
- Додати можливість користувачу записуватись до тренера на заняття на певний час;
- Додати категорії для програм тренувань;
- Додати можливість сортувати тренерів за рейтингом;
- Додати автоматизовану систему смс-розсилки;
- Додати можливість додавати програми тренувань в обране;
- Додати можливість додавання запису на тренування у Google Календар;

ВИСНОВКИ

Під час виконання моєї кваліфікаційної роботи, було проведено аналіз методів та засобів розробки веб-сайту тренажерної зали. Для цього було проаналізовано схожі ресурси, які є популярними в Україні. Цей аналіз дозволив визначити вимоги до веб-додатку: функціональність, зручність використання та привабливий дизайн.

Для розробки сайту була обрана клієнт-серверна архітектура, що відповідає сучасним вимогам е-commerce проєктів. Були проаналізовані функціональні, системні та нефункціональні вимоги, а також була створена діаграма варіантів використання для зрозуміння взаємодії користувачів з сайтом.

Детально розглянутий процес створення веб-додатку, його загальний принцип роботи та робота компонентів ПЗ. У результаті кваліфікаційної роботи був створений повноцінний веб-сайт тренажерної зали, який можна надалі вдосконалити та налаштувати під специфічні потреби, що сприятиме розвитку тренажерного залу, збільшенню клієнтської бази та підвищенню впізнаваності бренду.

Виконання кваліфікаційної роботи дало можливість застосувати теоретичні знання, набуті протягом 4 років навчання в університеті, на практиці. Це сприятиме подальшому розвитку в ІТ-сфері.

Реалізація е-commerce проєкту є важливим кроком у професійному й освітньому зростанні. Цей досвід дозволив краще зрозуміти процес розробки веб-сайтів і здобути цінний практичний досвід, який буде корисним у майбутній професійній діяльності.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Web Development with Django – Second Edition [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/web-development-with-django-second-edition/9781803230603>
2. Django 4 By Example – Fourth Edition [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/django-4-by-example-fourth-edition/9781801813051>
3. Django 4 for the Impatient [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/django-4-for-the-impatient/9781803245836>
4. Web Development with Django [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/web-development-with-django/9781839212505>
5. Becoming an Enterprise Django Developer [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/becoming-an-enterprise-django-developer/9781801073639>
6. Building SPAs with Django and HTML Over the Wire [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/building-spas-with-django-and-html-over-the-wire/9781803240190>
7. Django 3 By Example – Third Edition [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/django-3-by-example-third-edition/9781838981952>
8. Django RESTful Web Services [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/django-restful-web-services/9781788833929>
9. Django Design Patterns and Best Practices – Second Edition [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/django-design-patterns-and-best-practices-second-edition/9781788831345>
10. Mastering Django: Core [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/mastering-django-core/9781787281141>
11. Django documentation [Електронний ресурс] – режим доступу до ресурсу: <https://docs.djangoproject.com/en/4.2/>
12. Python 3.11.4 documentation [Електронний ресурс] – режим доступу до ресурсу: <https://docs.python.org/3/>
13. JavaScript from Beginner to Professional [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/javascript-from-beginner-to-professional/9781800562523>
14. Responsive Web Design with HTML5 and CSS - Fourth Edition [Електронний ресурс] – режим доступу до ресурсу: <https://www.packtpub.com/product/responsive-web-design-with-html5-and-css-fourth-edition/9781803242712>

					ІПЗ.КР.Б - 121 - 23 - ІЗ	Арк.
						48

15. Bootstrap documentation [Электронный ресурс] – режим доступа до ресурсу:
<https://getbootstrap.com/docs/5.3/getting-started/introduction/>
16. Learn Python Programming - Third Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/learn-python-programming-third-edition/9781801815093>
17. Python Object-Oriented Programming - Fourth Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/python-object-oriented-programming-fourth-edition/9781801077262>
18. Mastering Python - Second Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/mastering-python-second-edition/9781800207721>
19. Clean Code in Python - Second Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/clean-code-in-python-second-edition/9781800560215>
20. The Python Workshop - Second Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/the-python-workshop-second-edition/9781804610619>
21. Expert Python Programming - Fourth Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/expert-python-programming-fourth-edition/9781801071109>
22. Advanced Python Programming - Second Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/advanced-python-programming-second-edition/9781801814010>
23. Mastering Python Design Patterns - Second Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/mastering-python-design-patterns-second-edition/9781788837484>
24. Python Architecture Patterns [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/python-architecture-patterns/9781801819992>
25. Python for Geeks [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/python-for-geeks/9781801070119>
26. Python Fundamentals [Электронный ресурс] – режим доступа до ресурсу:
<https://www.packtpub.com/product/python-fundamentals/9781789807325>
27. Algorithms, 4th Edition [Электронный ресурс] – режим доступа до ресурсу:
<https://algs4.cs.princeton.edu/home/>
28. Data Structures and Algorithms Made Easy [Электронный ресурс] – режим доступа до ресурсу:
<https://www.goodreads.com/book/show/11289197-data-structures-and-algorithms-made-easy>

					ИПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						49

- | | | | | | |
|--|--|--|--|--|-------------------------|
| | | | | | ПЗ.КР.Б - 121 - 23 - ПЗ |
| | | | | | |
| | | | | | |

ДОДАТКИ

					ІІЗ.КР.Б - 121 - 23 - ІЗ	Арк.
						51

admin.py

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .forms import *
```

```
from .models import *
```

```
# Register your models here.
```

```
admin.site.register(Program)
admin.site.register(Membership)
admin.site.register(Advantage)
admin.site.register(Testimonial)
admin.site.register(Classe)
admin.site.register(Picture)
admin.site.register(WillingPeople)
admin.site.register(Exercise)
```

```
class AdminCustomUser(UserAdmin):
    model = CustomUser
    save_on_top = True
    add_fieldsets = (
        *UserAdmin.add_fieldsets, (
            'Custom fields', {
                'fields': (
                    'name',
                    'surname',
                    'age',
                    'gender',
                    'date_of_birth',
                    'photo',
                    'years_of_experience',
                    'info',
                    'membership_type',
                    'slug',
                    'groups',
                )
            }
        )
    )
    fieldsets = (
        *UserAdmin.fieldsets, (
            'Custom fields', {
                'fields': (
                    'name',
                    'surname',
```

					ІПЗ.КР.Б - 121 - 23 - ІЗ	Арк.
						52

```

        'age',
        'gender',
        'date_of_birth',
        'photo',
        'years_of_experience',
        'info',
        'membership_type',
        # 'groups',
        'slug',
    )
}
)
)

```

```
admin.site.register(CustomUser, AdminCustomUser)
```

forms.py

```

class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = (
            'name', 'surname', 'age', 'gender', 'date_of_birth', 'years_of_experience', 'photo', 'info',
            'membership_type',
            'slug', 'groups')

class CustomUserChangeForm(UserChangeForm):
    class Meta:
        model = CustomUser
        fields = (
            'name', 'surname', 'age', 'gender', 'date_of_birth', 'years_of_experience', 'photo', 'info',
            'membership_type',
            'slug', 'groups')

```

views.py

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.models import User
from django.shortcuts import render, redirect
from django.urls import reverse_lazy
from django.views.generic import TemplateView, DetailView, CreateView, View, UpdateView
from .models import *
from .forms import *
from django.contrib.auth import login, logout
from django.db.models import Q
```

```
# Create your views here.
```

```
class MainPage(TemplateView):
```

```
    template_name = 'gym_website/main_page.html'
```

```
    def get_context_data(self, *, object_list=None, **kwargs):
```

```
        context = super().get_context_data(**kwargs)
```

```
        context['advantages'] = Advantage.objects.all()
```

```
        context['trainers'] = CustomUser.objects.filter(groups__name='Trainer').order_by(«?»)[:4]
```

```
        context['gallery'] = Picture.objects.all()
```

```
        context['form'] = UserAuthenticationForm()
```

```
        context['willing_form'] = WillingForm()
```

```
        return context
```

```
class AboutPage(TemplateView):
```

```
    template_name = 'gym_website/about.html'
```

```

class ProgramsPage(TemplateView):
    template_name = 'gym_website/programs.html'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['programs'] = Program.objects.all()
        return context


class ProgramPage(DetailView):
    template_name = 'gym_website/program.html'
    model = Program


class PricesPage(TemplateView):
    template_name = 'gym_website/prices.html'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['membership'] = Membership.objects.all()
        context['willing_form'] = WillingForm()
        return context


class TrainersPage(TemplateView):
    template_name = 'gym_website/trainers.html'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        # !!!!!
        context['trainers'] = CustomUser.objects.filter(groups__name='Trainer')
        return context


class SingleTrainer(DetailView):

```

					ИПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						55

```

# !!!!!!!
model = CustomUser
template_name = 'gym_website/trainer.html'

def get_context_data(self, *, object_list=None, **kwargs):
    context = super().get_context_data(**kwargs)
    # !!!!!!!
    context['programs'] = Program.objects.filter(author_id=self.kwargs['pk'])
    return context

```

```

class AddExercise(CreateView):
    model = Exercise
    form_class = AddExercise
    template_name = 'gym_website/add_excercise.html'
    success_url = reverse_lazy('home')

```

```

# class AddProgram(CreateView):
#     model = Program
#     form_class = AddProgram
#     template_name = 'gym_website/add_excercise.html'
#     success_url = reverse_lazy('home')

```

```

class AddProgram(LoginRequiredMixin, TemplateView):
    template_name = 'gym_website/add_program.html'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['add_program_form'] = AddP()
        return context

```

```

class Profile(UpdateView):
    template_name = 'gym_website/profile.html'

```

					ПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						56

```
model = CustomUser
```

```
form_class = ProfileForm
```

```
def get_context_data(self, **kwargs):
```

```
    context = super().get_context_data(**kwargs)
```

```
    user = CustomUser.objects.get(pk=self.kwargs['pk'])
```

```
    context['classes'] = Classe.objects.filter(Q(trainer_id=self.kwargs['pk']) |
```

```
Q(client__id=self.kwargs['pk']))
```

```
    print(user)
```

```
    return context
```

```
def user_logout(request):
```

```
    logout(request)
```

```
    return redirect('home')
```

```
def user_login(request):
```

```
    if request.method == 'POST':
```

```
        auth_form = UserAuthenticationForm(data=request.POST)
```

```
        if auth_form.is_valid():
```

```
            user = auth_form.get_user()
```

```
            login(request, user)
```

```
            return redirect('home')
```

```
    else:
```

```
        auth_form = UserAuthenticationForm()
```

```
    return render(request, 'base.html', {'form': auth_form})
```

```
def willing_person(request):
```

```
    if request.method == 'POST':
```

```
        willing_form = WillingForm(data=request.POST)
```

```
        if willing_form.is_valid():
```

```
            willing_form.save()
```

```
            return redirect('home')
```

					ППЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						57

```

else:
    willing_form = WillingForm()
return render(request, 'base.html', {'willing_form': willing_form})

```

```

def add_exercise(request):
    if request.method == 'POST':
        add_exercise_form = AddExercise(data=request.POST)
        if add_exercise_form.is_valid():
            add_exercise_form.save()
            return redirect('home')
    else:
        add_exercise_form = AddExercise()
    return render(request, 'base.html', {"": add_exercise_form})

```

```

def add_p(request):
    if request.method == 'POST':
        add_program_form = AddP(data=request.POST)
        if add_program_form.is_valid():
            obj = add_program_form.save(commit=False)
            obj.author = CustomUser.objects.get(pk=request.user.id)
            obj.save()
            obj.exercise.set(add_program_form.cleaned_data['exercise'])
            return redirect('home')
    else:
        add_program_form = AddP()

```

models.py

```

from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator
from django.urls import reverse
from django.contrib.auth.models import AbstractUser

```

Create your models here.

```
class Program(models.Model):
    name = models.CharField(max_length=100)
    date = models.DateField(auto_now_add=True)
    duration = models.IntegerField(default=60)
    author = models.ForeignKey('CustomUser', on_delete=models.CASCADE)
    exercise = models.ManyToManyField('Exercise', related_name='exercises')

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('program', kwargs={'pk': self.pk})

class Meta:
    verbose_name = 'Program'
    verbose_name_plural = 'Programs'
    ordering = ['date']
```

```
Gender = (
    ('man', 'Man'),
    ('women', 'Women')
)
```

```
class CustomUser(AbstractUser):
    name = models.CharField(max_length=20)
    surname = models.CharField(max_length=20)
    age = models.IntegerField(blank=True, null=True)
    gender = models.CharField(default='man', choices=Gender, max_length=20)
    date_of_birth = models.DateField(blank=True, null=True)
    photo = models.ImageField(upload_to='photos/trainers/', blank=True, null=True)
    info = models.TextField(blank=True, null=True)
```

					ППЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						59

```

start_date = models.DateField(auto_now_add=True)

membership_type = models.ForeignKey('Membership', null=True, on_delete=models.SET_NULL,
blank=True)

slug = models.SlugField(max_length=50, verbose_name='Url', unique=True, blank=True, null=True)
years_of_experience = models.IntegerField(blank=True, null=True)


def __str__(self):
    return f'{self.name} {self.surname}'


def get_absolute_url(self):
    return reverse('trainer', kwargs={'pk': self.pk})


class Meta:
    verbose_name = 'User'
    verbose_name_plural = 'Users'
    ordering = ['start_date']


class Membership(models.Model):
    name = models.CharField(max_length=100)
    info = models.TextField()
    price = models.IntegerField()
    client = models

    def __str__(self):
        return self.name


class Meta:
    verbose_name = 'Membership'
    verbose_name_plural = 'Memberships'
    ordering = ['price']


class Advantage(models.Model):
    title = models.CharField(max_length=20)

```

					ППЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						60

```

text = models.CharField(max_length=60)
image = models.ImageField(upload_to='photos/advantages')

```

```

def __str__(self):
    return self.title

```

```

class Meta:
    verbose_name = 'Advantage'
    verbose_name_plural = 'Advantages'
    ordering = ['title']

```

```

class Testimonial(models.Model):
    title = models.CharField(max_length=20)
    text = models.CharField(max_length=150)
    author = models.ForeignKey('CustomUser', on_delete=models.CASCADE)

```

```

def __str__(self):
    return self.title

```

```

class Meta:
    verbose_name = 'Testimonial'
    verbose_name_plural = 'Testimonials'
    ordering = ['title']

```

```

class Classe(models.Model):
    name = models.CharField(max_length=200)
    date = models.DateTimeField(blank=False)
    client = models.ForeignKey('CustomUser', related_name='client', null=True,
on_delete=models.SET_NULL)
    trainer = models.ForeignKey('CustomUser', related_name='trainer', null=True,
on_delete=models.SET_NULL)
    price = models.IntegerField()
    mark = models.IntegerField(default=4, validators=[MinValueValidator(0), MaxValueValidator(5)])

```

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						61

```
finished = models.BooleanField(default=False)
```

```
def __str__(self):  
    return self.name
```

```
class Meta:  
    verbose_name = 'Classe'  
    verbose_name_plural = 'Classes'  
    ordering = ['date']
```

```
class Picture(models.Model):  
    img = models.ImageField(upload_to='photos/gallery')
```

```
class WillingPeople(models.Model):  
    phone = models.CharField(max_length=13)  
    name = models.CharField(max_length=20)  
    membership = models.ForeignKey('Membership', on_delete=models.SET_NULL, null=True)  
    is_completed = models.BooleanField(default=False)
```

```
def __str__(self):  
    return f'Client {self.pk} – {self.name}'
```

```
class Exercise(models.Model):  
    name = models.CharField(max_length=100)  
    duration = models.IntegerField()  
    instruction = models.URLField()
```

```
def __str__(self):  
    return self.name
```

urls.py

```
from django.conf import settings
```

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						62

```

from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('gym_website.urls'))
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

gym_website/urls.py

```

from django.urls import path
from .views import *

urlpatterns = [
    path("", MainPage.as_view(), name='home'),
    path('about/', AboutPage.as_view(), name='about'),
    path('trainers/', TrainersPage.as_view(), name='trainers'),
    path('programs/', ProgramsPage.as_view(), name='programs'),
    path('trainer/<int:pk>', SingleTrainer.as_view(), name='trainer'),
    path('program/<int:pk>', ProgramPage.as_view(), name='program'),
    path('pricing/', PricesPage.as_view(), name='prices'),
    path('login/', user_login, name='login'),
    path('logout/', user_logout, name='logout'),
    path('willing/', willing_person, name='willing'),
    path('add-exercise/', AddExercise.as_view(), name='add_exercise'),
    path('add-program/', AddProgram.as_view(), name='add_program'),
    path('add-p/', add_p, name='add_p'),
    path('profile/<int:pk>', Profile.as_view(), name='profile')
]

```

settings.py

```

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings – unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-@30jf^%+uy%b1ybqx#kfd0pb9_vlfcesq_l8^()thdb^!l+@07'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'gym_website.apps.GymWebsiteConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'gym.urls'
IMAGE_FIELD_MAX_FILE_SIZE = 1000000

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
    },
    {
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [

```

```

        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
],

WSGI_APPLICATION = 'gym.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

```

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						65

```

#user
AUTH_USER_MODEL = 'gym_website.CustomUser'

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'gym/static'),
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'

```

					ІПЗ.КР.Б - 121 - 23 - ІЗ	Арк.
						66

base.html

```

<!DOCTYPE html>
<html lang=«en»>
<head>
  <meta charset=«UTF-8»>
  <title>{% block title %}{% endblock %}</title>
  <link href=«https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css» rel=«stylesheet»
    integrity=«sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM»
crossorigin=«anonymous»>
</head>
<body>
{% include '_inc/_header.html' %}
{% block content %}{% endblock %}
{% include '_inc/_footer.html' %}

<script src=«https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js»
  integrity=«sha384-
geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz»
crossorigin=«anonymous»></script>
</body>
</html>

```

_header.html

```

<nav class=«navbar navbar-expand-lg bg-body-tertiary» data-bs-theme=«dark» style=«z-index: 100»>
  <div class=«container-lg»>
    <a class=«navbar-brand» href=«{% url 'home' %}»>Gym</a>
    <button class=«navbar-toggler» type=«button» data-bs-toggle=«collapse» data-bs-
target=«#navbarSupportedContent»
      aria-controls=«navbarSupportedContent» aria-expanded=«false» aria-label=«Toggle
navigation»>
      <span class=«navbar-toggler-icon»></span>
    </button>
    <div class=«collapse navbar-collapse» id=«navbarSupportedContent»>
      <ul class=«navbar-nav me-auto mb-2 mb-lg-0»>
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'home' %}»>Додому</a>
        </li>
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'prices' %}»>Ціни</a>
        </li>
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'trainers' %}»>Тренери</a>
        </li>
        {% if request.user.is_authenticated %}
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'programs' %}»>Програми</a>

```

						Арк.
					ІІЗ.КР.Б - 121 - 23 - ІІЗ	67

```

    </li>
    {% endif %}
    <li class=«nav-item»>
      <a class=«nav-link» href=«{% url 'about' %}»>Про Нас</a>
    </li>
    {% if request.user.is_authenticated %}
      {% if user.groups.all.0.name != 'Client' %}
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'add_exercise' %}»>Додати вправу</a>
        </li>
        <li class=«nav-item»>
          <a class=«nav-link» href=«{% url 'add_program' %}»>Додати програму</a>
        </li>
      {% endif %}
    {% endif %}
  </ul>

  <div class=«d-flex <»
    {% if request.user.is_authenticated %}
      <a href=«{% url 'profile' pk=user.pk %}»
        class=«nav-link text-light me-4 mt-1»>{{ user.username }}</a>
      {#      <span class=«text-light me-2 mt-1»>{{ user.groups.all.0.pk }}</span>#}
      {#      <span class=«text-light me-2 mt-1»>{{ user.pk }}</span>#}
      <a href=«{% url 'logout' %}» class=«btn btn btn-outline-light» type=«submit»>Log
out</a>
    {% else %}
      {#      <a class=«btn btn btn-outline-success»>Sign in</a>#}
      <button class=«btn btn-outline-light» data-bs-toggle=«modal» data-bs-target=«#modal-
login»>Log in
      </button>
    {% endif %}
  </div>

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

```

_footer.html

```

<nav class=«navbar navbar-expand-lg bg-body-tertiary mt-4» data-bs-theme=«dark»
  style=«z-index: 100»>
  <div class=«container-lg»>
    <a class=«navbar-brand» href=«#»>Gym</a>
    <button class=«navbar-toggler» type=«button» data-bs-toggle=«collapse» data-bs-
target=«#navbarSupportedContent»
      aria-controls=«navbarSupportedContent» aria-expanded=«false» aria-label=«Toggle
navigation»>
      <span class=«navbar-toggler-icon»></span>
    </button>
    <div class=«collapse navbar-collapse» id=«navbarSupportedContent»>
      <ul class=«navbar-nav me-auto mb-2 mb-lg-0»>
        <li class=«nav-item»>

```

```

        <a class=«nav-link» href=«{% url 'home' %}»>Додому</a>
    </li>
    <li class=«nav-item»>
        <a class=«nav-link» href=«{% url 'prices' %}»>Ціни</a>
    </li>
    <li class=«nav-item»>
        <a class=«nav-link» href=«{% url 'trainers' %}»>Тренери</a>
    </li>
    {% if request.user.is_authenticated %}
        <li class=«nav-item»>
            <a class=«nav-link» href=«{% url 'programs' %}»>Програми</a>
        </li>
    {% endif %}
    <li class=«nav-item»>
        <a class=«nav-link» href=«{% url 'about' %}»>Про Нас</a>
    </li>
</ul>

</div>

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

```