МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Web-додаток підбору тренувань

Здобувача групи ІТз-91с Ігнатенка Олександра Олександровича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр ІГНАТЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доц., Юлія ПАРФЕНЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ІТ

_____ Світлана ВАЩЕНКО
«__» _____ 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Ігнатенко Олександр Олександрович

1 Тема роботи *Web-додаток підбору тренувань*

керівник роботи *Парфененко Ю.В., к.т.н., доцент*,

затверджені наказом по університету від « 26 » 05 2023 р.
№0578-VI

2 Строк подання студентом роботи «12» червня 2023 р.

3 Вхідні дані до роботи програми тренувань, відеозаписи тренувань

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування web-додатку, розробка web-додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____16 січня 2023 року _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	16.01.2023-25.01.2023	
2	Оформлення технічного завдання	26.01.2023-10.02.2023	
3	Проведення аналізу предметної області	11.02.2023-26.02.2023	
4	Проведення проектування webдодатку	26.02.2023-10.03.2023	
5	Розробка web-додатку	11.03.2023-06.04.2023	
6	Тестування web-додатку	07.04.2023-25.04.2023	
7	Завантаження web-додатку на хостинг	26.04.2023-15.05.2023	
8	Оформлення пояснювальної записки	16.05.2023-12.06.2023	

Студент _____ Олександр ІГНАТЕНКО
(підпис)

Керівник роботи _____ к.т.н., доц., Юлія ПАРФЕНЕНКО
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підбору тренувань».

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел із 16 найменувань, трьох додатків. Загальний обсяг пояснювальної записки складає 93 сторінки, у тому числі 45 сторінок основного тексту, 2 сторінки списку використаних джерел, 46 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку по підбору тренувань.

У першому розділі проведено огляд останніх досліджень за тематикою роботи та проаналізовано аналоги розроблюваного web-додатку, визначено їх переваги та недоліки. Також було поставлено мету й задачі проекту, визначено засоби реалізації.

У другому розділі проведено структурно-функціональне моделювання, визначено варіанти використання web-додатку. У результаті було зроблено такі діаграми як: модель web-додатку, прогноз розвитку фітнес-додатків, контекстна діаграма IDEF0, діаграма варіантів використання.

У третьому розділі описано процес розробки web-додатку, яка є результатом проектування. Також наведено архітектуру web-додатку. Проведено тестування web-додатку.

Ключові слова: web-додаток, reactjs, javaspring, postresql, тренування, розробка.

ЗМІСТ

ВСТУП

Фітнес-додатки – це програми, розроблені для підтримки форми та здоров'я. Ціль цих додатків – зробити спосіб життя здоровішим, відстежуючи споживання їжі, споживання води та режим тренувань. Деякі програми навіть відстежують частоту серцевих скорочень та кров'яний тиск. Окрім цього в них є тренер, який допомагає своїм клієнтам ефективно досягати своїх цілей.

Варто зазначити, що в сучасному світі здоровий спосіб життя та фізична активність відіграють все більш важливу роль. З кожним днем все більше людей усвідомлює необхідність ведення здорового способу життя, включаючи регулярні тренування, для досягнення фізичної форми, зміцнення здоров'я та покращення загального самопочуття. У цьому контексті виникає потреба в ефективних інструментах для планування тренувань, які можуть допомогти людям досягти своїх фітнес-цілей та зробити процес тренувань більш структурованим та результативним [1].

Об’єктом дослідження даної роботи є проектування та розробка веб-додатку, заснованого на сучасних технологіях веб-розробки, що надаватиме користувачеві можливість ефективно планувати свої тренування та досягати бажаних результатів. Особлива увага буде приділена розробці зручного інтерфейсу, адаптованого під різні типи пристроїв, щоб програма була доступна та зручна використовувати як на персональних комп'ютерах, так і на мобільних пристроях.

Предметом дослідження є методи та підходи, які використовуються при розробці web-додатків для планування тренувань. У ході роботи буде проаналізовано існуючі підходи до розробки подібних додатків, вивчено сучасні технології веб-розробки та вибрано найбільш підходящі інструменти та методи для реалізації поставленої мети.

Метою даної роботи є розробка та реалізація web-додатку, що надає користувачеві можливість планувати та відстежувати свої тренування. Додаток повинен надавати зручний та інтуїтивно зрозумілий інтерфейс, де користувач зможе створювати персоналізовані тренувальні програми та купувати їх.

Для досягнення мети потрібно вирішити такі задачі:

-провести аналіз предметної області розроблення web-додатків для фітнес-тренувань;

-виконати постановку задачі дослідження та планування робіт;

-виконати проектування web-додатку;

-розробити web-додаток.

Такий web-додаток має актуальність у сучасному суспільстві, де багато людей прагнуть здорового способу життя та фізичної активності, але стикаються з труднощами у плануванні та систематизації своїх тренувань.

Переваги такого web-додатку полягають у тому, що користувачі зможуть створювати свої індивідуальні тренувальні програми, враховуючи свої цілі, фізичну підготовку та переваги. Вони зможуть вибирати різні типи вправ, а також отримувати рекомендації та поради для оптимізації своїх тренувань. Це дозволить користувачам більш ефективно використовувати свій час та ресурси, а також підвищити мотивацію та задоволення від занять спортом [2].

У результаті розробка та реалізація web-додатку для планування тренувань має велику значущість. Це допоможе покращити фізичну форму та здоров'я користувачів, забезпечуючи їм індивідуалізацію тренувального процесу та мотивацію для досягнення своїх фітнес-цілей. Ця робота включатиме аналіз існуючих підходів та методів, вибір оптимальних технологій та розробку зручного та функціонального веб-додатку, який зможе задовольнити потреби користувачів у плануванні тренувань.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Фітнес-додаток – це технічне рішення, яке допомагає користувачам покращити загальний стан здоров'я та самопочуття. Тому в ньому розглядаються аспекти, пов'язані з харчовими звичками, активністю, загальним самопочуттям та психічним здоров'ям. У міру того, як користувачі згодом ставали свідомішими, фітнес-додатки стали керувати великою кількістю функцій [1].

В останні роки було проведено багато досліджень, пов'язаних із розробкою та покращенням веб-додатків для планування тренувань. Ось кілька основних напрямів досліджень:

Інтерактивний та інтуїтивний інтерфейс. При вивченні предметної області увага була зосереджена на створені зручного та інтуїтивного інтерфейсу для веб-застосунків, який дозволяв би користувачам легко створювати та налаштовувати свої тренувальні програми. Використання інтерактивних елементів, візуальних підказок та персоналізації інтерфейсу допомагає підвищити зручність використання та мотивацію користувачів [3].

Адаптивність та мобільність. З огляду на дедалі більшу популярність мобільних пристроїв дослідження також звертали увагу на адаптивність веб-додатків для планування тренувань під різні типи пристроїв. Розробка мобільних версій та використання адаптивного дизайну стають все більш важливими для забезпечення доступності та зручності використання [4].

Інтелектуальні рекомендації. Деякі дослідження фокусувалися на розробці інтелектуальних систем рекомендацій для планування тренувань. Ці системи використовують алгоритми машинного навчання та аналізу даних для надання персоналізованих рекомендацій користувачеві. Вони враховують індивідуальні характеристики користувача, його фізичну підготовку, цілі та переваги, щоб допомогти йому ефективно планувати тренування та досягати бажаних результатів.

Важливо відзначити, що область web-додатків для планування тренувань продовжує активно розвиватися, і нові дослідження та публікації вносять нові інновації та покращення в цю область. Деякі з останніх досліджень, можливо, присвячені більш точному моніторингу фізичної активності через використання датчиків і пристроїв, що носяться, інтеграції додатків з соціальними мережами для обміну досвідом і мотивації, а також розробці алгоритмів машинного навчання для більш точних та індивідуалізованих рекомендацій.

Web-додаток для проведення інтервальних тренувань призначений для автоматизації процесу проведення інтервальних тренувань, а саме забезпечення користувача комфортного стеження за часом. У досліджуваної предметної області можна назвати такі основні бізнес-процеси:

— розпочати тренування;

— завершити чи перервати тренування;

— одержати повідомлення про зміну інтенсивності;

— встановити, завантажити та зберегти налаштування;

— ведення статистики тренувань [5].

Результатом проведення аналізу предметної області є побудова її моделі, яка представлена на рисунку 1.1. Атрибутами предметної області веб-додатка для проведення інтервальних тренувань є: налаштування, тренування, тарифи та кошик.

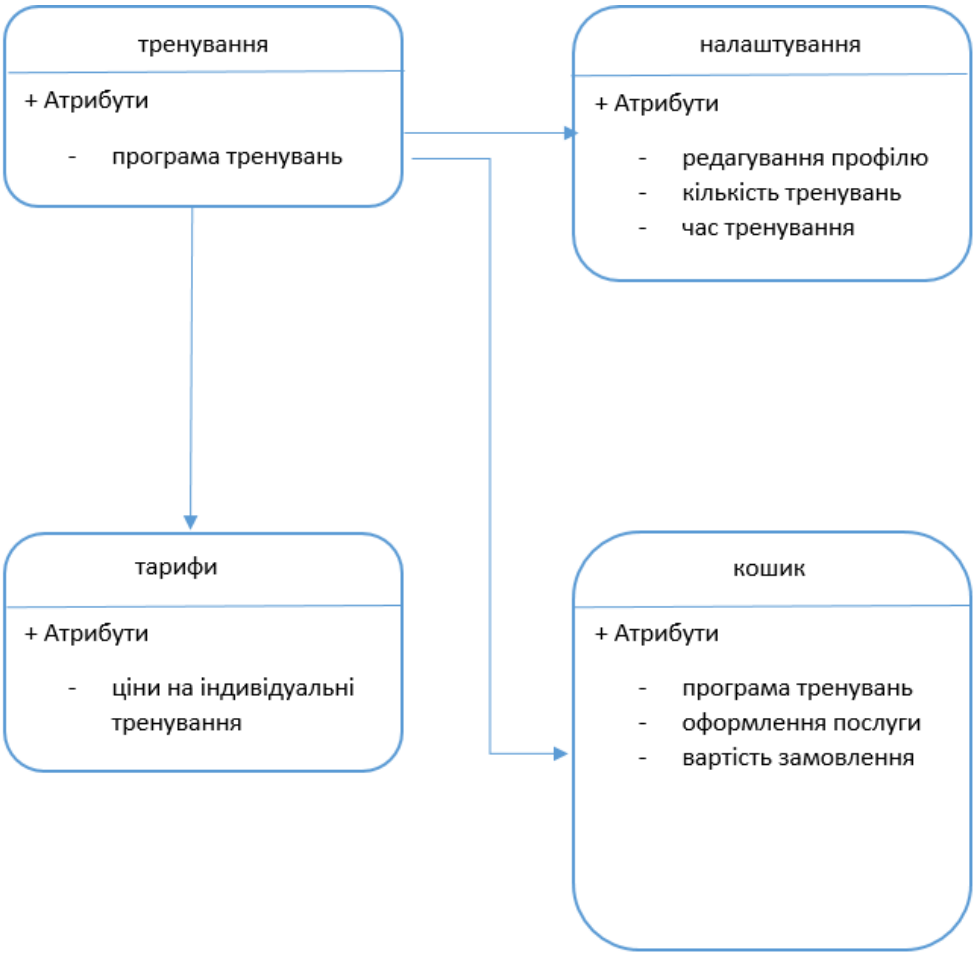

Рисунок 1.1 – Модель web-додатку

Насамкінець, дослідження та розробки в галузі web-додатків для планування тренувань мають велику актуальність та значущість, допомагаючи людям досягати своїх фітнес-цілей та підтримувати здоровий спосіб життя. У цій роботі основна увага зосереджена на розробці web-додатку, враховуючи існуючі дослідження та інновації, щоб запропонувати ефективне та зручне рішення для планування тренувань та досягнення бажаних результатів [6].

На рисунку 1.2 можна подивитись прогноз розвитку сфери розробки web-додатків для тренувань:

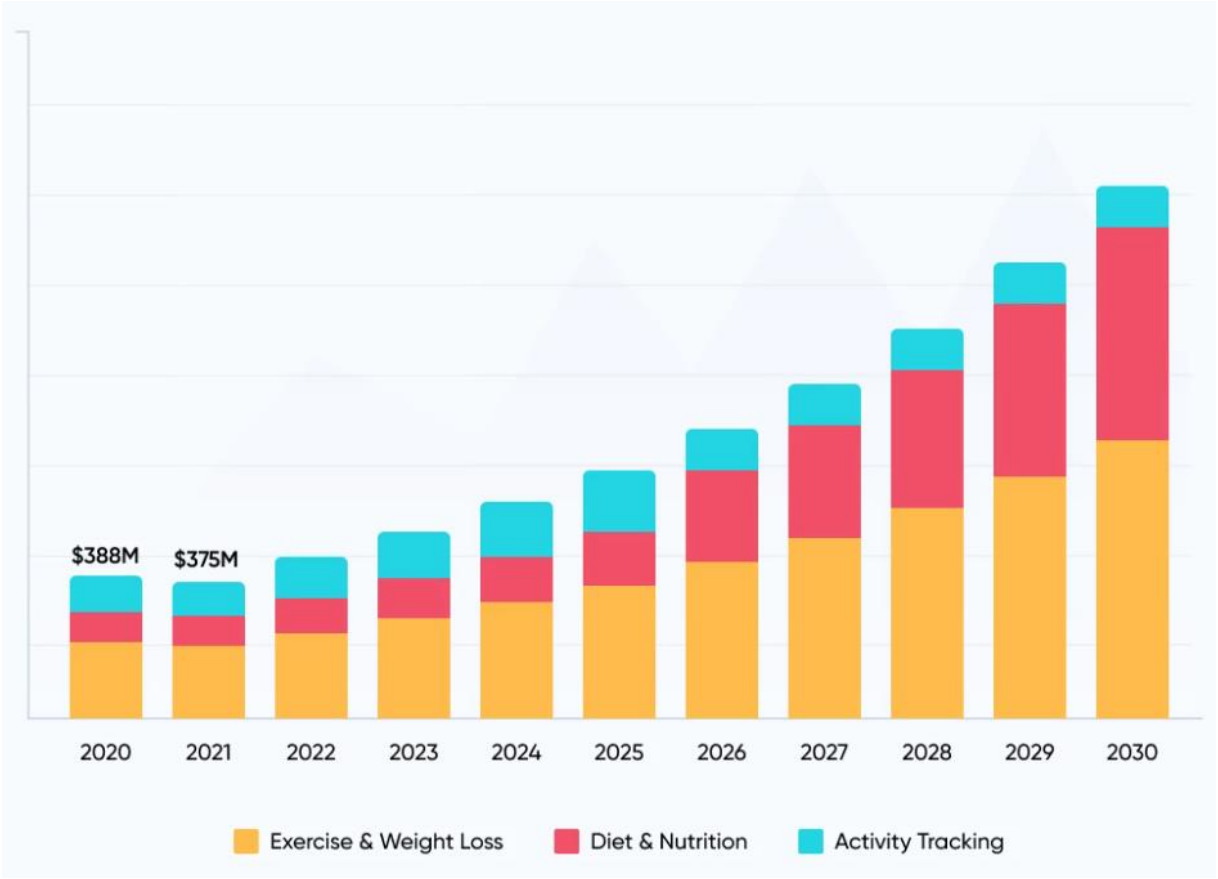

Рисунок 1.2 – Прогноз розвитку фітнес-додатків

За даними звіту Grand View Research, Inc. світовий ринок веб-додатків для тренувань зростає із середньорічним темпом 44,2% і, як очікується, сягне 111,8 млрд доларів США до 2025 року [7].

1.2 Аналіз програмних продуктів - аналогів

Оскільки web-додатки категорії «Здоров'я та фітнес» представлені великим набором, варто розглянути лише кілька найбільш популярних аналогів. Але більшість з них представлена у вигляді мобільного додатку.

8fit. Ця програма поєднує тренування за запитом і планування харчування і відмінно підходить для людей, які люблять багато порад та інструкцій. Програма створює індивідуальну програму для дієти та вправ на основі поставленої мети. Як тільки користувач поділиться з 8fit, чого він хоче досягти, програма складає плани харчування та тренувань, використовуючи рецепти та списки покупок. Безкоштовна версія відкриває доступ лише до деяких тренувань, які вимагають, щоб користувачі часто дивилися та натискали на екран.

Крім того, у додатку можна відстежувати свою вагу та активність. Версія Pro краща та відкриває персоналізовані страви, списки покупок, реєстрацію калорій та додаткові тренування. Ціни на план Pro становлять 79,99 доларів на рік, 59,99 доларів за шість місяців або 24,99 доларів на місяць [8].

Aaptiv. Ця програма спеціалізується на аудіо-тренуваннях під керівництвом тренерів. Тренер каже, що робити. Додаток пропонує тренування, які підходять конкретному користувачеві, на основі наданої їм інформації, наприклад, які вправи подобаються конкретному користувачеві (розтяжка, силові тренування, йога, їзда на велосипеді, біг, підйом сходами та інші) та які стилі музики вони віддають перевагу. При реєстрації доступна 7-денна безкоштовна демо-версія [9].

FitOn. Ця програма пропонує відеотренування на запит під керівництвом тренерів, і значна частина контенту безкоштовна. У додатку можна підібрати різні тренування, залежно від рівня інтенсивності, типу тренування, включаючи йогу, сідниці та стегна, прес, розтяжку тощо. Якщо користувач носить пульсометр під час тренування, він може бачити свій пульс на екрані під час руху. Також є таблиця лідерів, де можна змагатись з іншими учасниками чи групою друзів [10].

Jefit. Додаток має великий список вправ з гіф-анімаціями. Можна шукати за назвою (додаток англійською мовою), застосовувати фільтри по м'язах, що навантажуються, та обладнання, додавати в обране. Якщо не знайдено потрібної вправи, можна додати свою. Дуже зручно складати свою програму. Достатньо вказати день, додати вправи, редагувати кількість підходів та повторень, задавати час відпочинку.

Вже в процесі тренування можна швидко змінити вагу та кількість повторень, закінчити підхід та включити таймер відпочинку зі зворотним відліком [11].

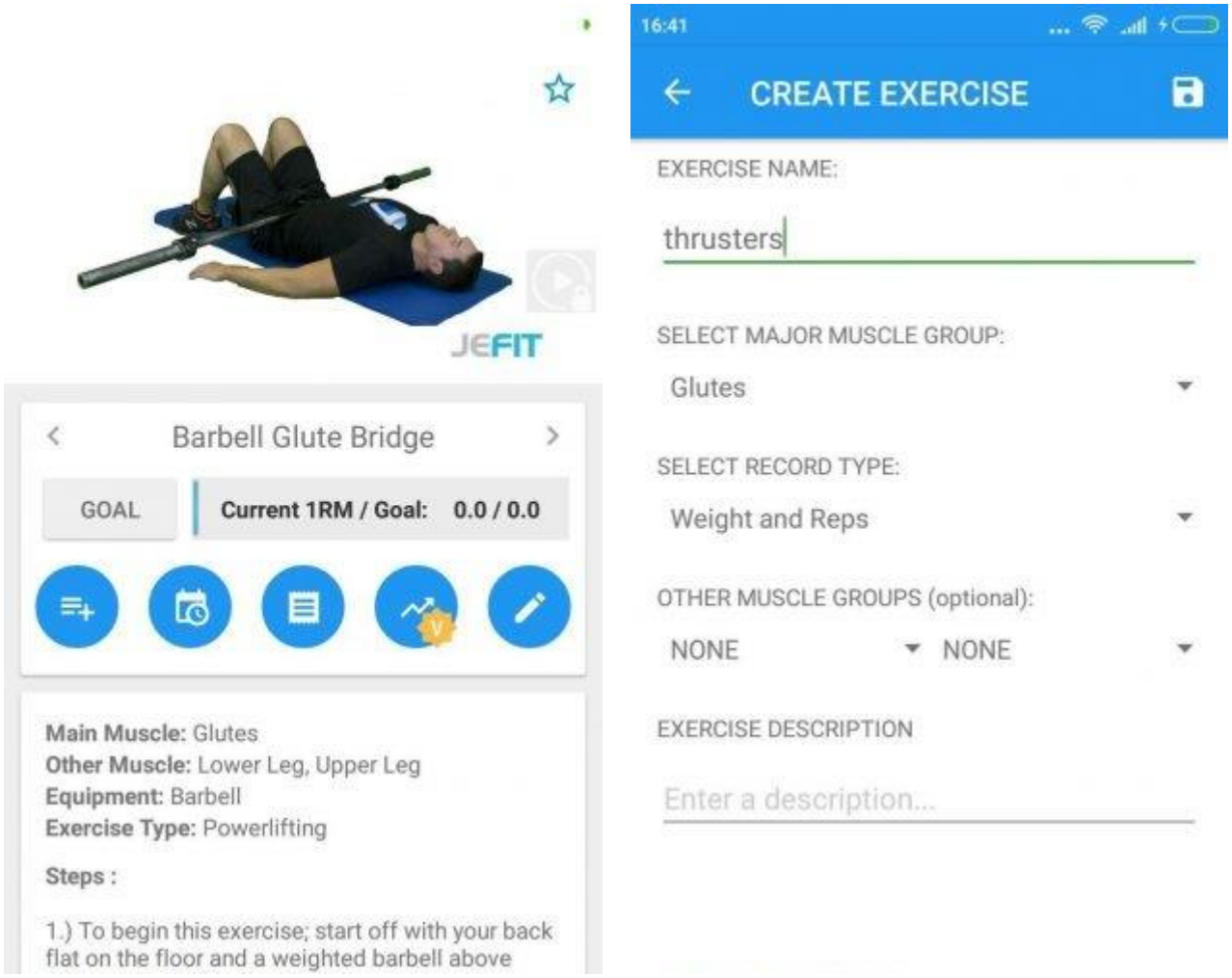

Рисунок 1.3 – Скрін додатку Jefit

WODster. У додатку можна вибрати комплекс із великої бази готових тренувань або записати свій WOD та прикріпити фотографію дошки з кросфіт-боксу. Можна додавати WOD на час, кількість раундів, за методом EMOM, на максимальну вагу і за протоколом табату. У будь-якому випадку користувачі мають відповідні таймери зі звуковим сигналом і можливістю відзначати закриті раунди. А щоб веселіше було їх закривати, можна додати свою музику зі скачаної на пристрій.

Після закінчення додаток записується результат, можна зробити фото та залишити коментар. Крім запису WOD, можна заносити в додаток персональні рекорди, наприклад свій максимум у силових та важкоатлетичних вправах [12].

1.3 Аналіз підходів до розробки WEB-додатків

Визначення методології недооцінюється у циклі розробки програми. Вибір неправильного методу може поставити під загрозу весь проект. Методологія, що використовується, визначає загальну структуру процесу розробки програми, наприклад:

1. Як управляється проект
2. Як збираються вимоги
3. Як проводиться тестування
4. Як впроваджуються зміни

У розробці програм використовується кілька методологій, кожна з них має переваги та недоліки. Деякі з цих методологій забезпечують більшу гнучкість і швидший вихід ринку, тоді як інші зосереджені на тестуванні перед випуском товару, документації тощо.

На тип обраної методології впливають такі чинники, як виділений бюджет, швидкість випуску, доступні завершення проекту і тд. Більшість методологій розробки додатків можна зарахувати до однієї з трьох категорій.

- Водоспад
- RAD
- Гнучка методологія.

Методологія водоспаду. Така назва зумовлюється тим, що якщо ви впадете, ви вже не зможете повернутися нагору. Команда працює довгий час. Вся програма створюється, а потім тестується. Потім демонструється замовнику та готова до реалізації. Метод водоспаду передбачає, що вимоги до проекту точні, а замовник та керівник проекту очікують один результат.

Методологія швидкої розробки програм (RAD).

Мета полягає в тому, щоб швидко створити робочу версію програми та після цього виконувати ітерації. Команда та замовник тісно співпрацюють один з одним. Команди RAD включають лише досвідчених розробників.

Гнучка методологія розробки додатків. Ця методологія схожа на RAD, але включає деякі зміни, щоб зробити її більш придатною для більших проектів. Кожна функція розробляється командою, замовник бере участь у перегляді та затвердженні функцій перед розробкою.

Agile використовує спринти або коли необхідно створити, протестувати та подати певну функцію. Включає весь SDLC для функції в кожний спринт. Це допомагає дотримуватись запланованого графіка та дозволяє проводити часті перевірки. Замовник інформується частіше, ніж Waterfall, але бачить лише готову роботу, на відміну RAD.[13]

1.4 Сучасні технології веб-розробки

Веб-розробка змінюється швидко з розвитком технологій. Розробники повинні бути в курсі всіх останніх веб-технологій, які можуть бути реалізовані. Щоб підтримувати та покращувати веб-розробку, необхідно вибрати правильний інструмент, і це може бути важким рішенням, оскільки є безліч варіантів.

Нижче можна ознайомитись із сучасними технологіями веб-розробки, які використовуються розробниками у 2023 році.

1. Прогресивний веб-додаток

Прогресивна веб-програма має характеристики як веб-сайту, так і локально встановленої програми. Він має нові можливості для сучасних додатків. Він пропонує переваги традиційної локальної програми в налаштуванні, яка використовується для різних типів пристроїв. Безболісно керувати та підтримувати.

PWA включають сучасні функції шаблонів, API та інші. Ці програми працюють в автономному режимі, їх можна встановити, їх легко синхронізувати, вони можуть надсилати push-сповіщення тощо. За даними Smashing Ideas, компанії, які переходять на PWA, отримують від 20 до 250 відсотків збільшення трафіку.

2. Yii

Yii – це середовище розробки веб-додатків із відкритим вихідним кодом. Він використовує PHP5 і має гарні інструменти для налагодження та тестування програм. Це дуже просто та зручно для розробників.

3. Метеор JS

Meteor JS використовує Node.js і допомагає розробникам створювати веб-програми, що використовуються на різних платформах. Це веб-фреймворк JavaScript з відкритим кодом, який має низький час завантаження веб-сторінки.

4. Джанго

Django - це популярний фреймворк, заснований на Python і використовує архітектуру MVC. Це спрощує процес розробки додатків та надає кілька інструментів, таких як ORM, моделі, адміністрування Django, шаблони тощо.

5. Motion UI

Клієнту потрібний зручний для мобільних пристроїв досвід з інтерактивним використанням. Motion UI використовується для готових ефектів як класів CSS. Ці рухи використовуються для ефектів переходу, щоб покращити анімацію ковзання, повороту, згасання, масштабування та обертання.

Інтерфейс користувача Motion використовується для забавних елементів, таких як Zoom Out, Zoom In, Slide та анімація паралакса. Це додає жвавості сайту та додатку.

Що стосується розробки веб-додатку для цієї дипломної, то для цього використовується гібридний підхід. У якості технологій розробки використовується Rest API з окремим backend та frontend частинами за допомогою засобів react+js, а також Spring+Java.

1.5 Постановка задачі

Розроблений web-додаток для планування тренувань призначений для автоматизації процесу проведення тренувань. Цей додаток повинен забезпечувати зберігання та обробку інформації про налаштування та статистику тренувань.

Система орієнтована на кінцевого користувача, що не має високої кваліфікації в галузі обчислювальної техніки. Тому додаток повинен мати простий, зручний і легко освоюваний інтерфейс.

Постановка задачі для розробки web-додатку для планування тренувань включає наступні аспекти:

- Розробка функціональних вимог. Визначення основних функцій та можливостей, які має надавати програма. Це включає створення та налаштування персоналізованих тренувальних програм, відстеження виконання вправ та прогресу, ведення журналу тренувань, надання

рекомендацій та порад для оптимізації тренувального процесу, а також можливість взаємодії з іншими користувачами та тренерами [14].

- Визначення дисфункційних вимог. Встановлення обмежень та критеріїв продуктивності, безпеки, доступності та зручності використання програми. Це включає вибір відповідних технологій і архітектури, забезпечення захисту даних користувачів, адаптивність інтерфейсу для різних пристроїв і браузерів, а також забезпечення високої продуктивності і чуйності програми.
- Дослідження технологій. Оцінка різних технологій, інструментів та фреймворків, які можуть бути використані для розробки програми. Це включає вибір мови програмування, фронтенд та бекенд технологій, бази даних та інших компонентів, необхідних для реалізації функціональності та вимог програми.
- Проектування інтерфейсу. Розробка дизайну та користувальницького інтерфейсу програми з урахуванням зручності використання, естетики та узгодженості сучасних веб-стандартів. Це включає створення прототипів, макетів екранів, визначення навігації та взаємодії користувача з програмою [15].
- Реалізація та тестування. Фактична розробка програми на основі розроблених вимог та дизайну. Це включає програмування функцій, налаштування бази даних, тестування та налагодження програми, а також впровадження механізмів для забезпечення якості та надійності.
- Розвиток та підтримка. Планування подальшого розвитку програми, включаючи додавання нових функцій, покращення продуктивності, виправлення помилок та оновлення технічної бази програми. Також передбачається підтримка програми після її розгортання, включаючи моніторинг роботи, регулярне оновлення та забезпечення безпеки даних користувачів.

- Оцінка ефективності та задоволеності користувачів: Після розробки та впровадження програми необхідно провести оцінку його ефективності та задоволеності користувачів. Це може включати збір зворотного зв'язку від користувачів, проведення тестової експлуатації, аналіз використання та моніторинг показників продуктивності для оцінки успішності досягнення поставлених цілей та потреб користувачів.

В цілому, постановка задачі для розробки web-додатку для планування тренувань включає визначення функціональних та нефункціональних вимог, вибір технологій, проектування інтерфейсу, розробку, тестування, розвиток та підтримку програми. Етапи виконання завдання можуть змінюватись в залежності від конкретних потреб та особливостей проекту

Користувачів web-додатку можна розділити на дві категорії: персональний тренер тренажерного залу та спортсмен, який займається без допомоги тренера.

Таблиця 1.1. Користувачі web-додатку

Назва	**Опис**
Тренер	Користувач, який складає програму тренувань для кожного користувача, індивідуальні параметри.
Спортсмен	Користувач, який використовує програму особисто, самостійно встановлює налаштування, зацікавлений у ефективних тренуваннях

З таблиці видно, що користувачі додатка відрізняються тільки характером використання функцій зберігання списку налаштувань і ведення статистики.

Web-додаток складається з двох частин. Перша частина – серверна з базою даних, призначена для зберігання інформації про налаштування та статистику тренувань. Друга частина – клієнтська програма, що дозволяє автоматизувати багато функцій, пов'язаних із забезпеченням автоматизації процесу складання програм тренувань.

На основі даних, отриманих при аналізі предметної області, виділено варіанти використання для різних користувачів. Аналіз варіантів використання для різних користувачів показав, що персональному тренеру мають бути доступні такі самі функції, як і спортсмену, що отримує програму тренувань від тренера. Таким чином, всі категорії користувачів мають однаковий набір варіантів використання.

2. Проектування інформаційної системи

Для розробки додатку було створено IDEF0 та Use Case діаграми, окрім цього розроблено схему бази даних.

IDEF0 – це методологія моделювання яка використовується задля опису логічних відносин в додатку не зважаючи на її реалізацію. Ця діаграма відображає вхідні та вихідні данні, механізми(людина, веб-додаток, тощо), та вимоги до додатку. За допомогою цієї діаграми можливо визначитися з бізнес процесами необхідними для монетизації та спростити розроблення програмної реалізації [16].

Спочатку я розробив контекстну IDEF0 діаграму(рисунок 2.1), яка дозволяє побачити загальну стрктуру веб – додатку, та бізнес процеси. За допомогою цієї діаграми з'являється можливість аналізу ризиків та чітке розуміння потреб клієнтів та необхідних реалізацій.

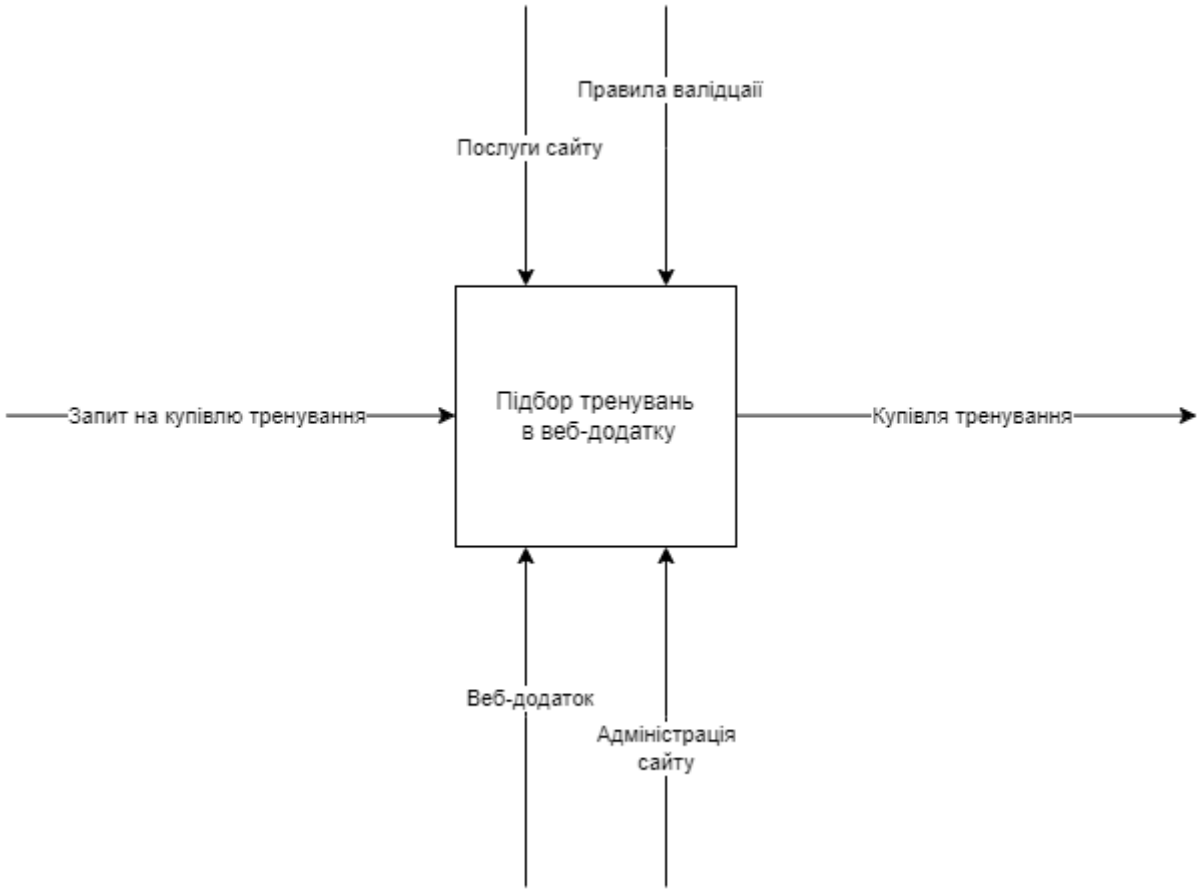

Рисунок 2.1 – Контекстна IDEF0 діаграма

Після створення контекстної діаграми необхідно зробити її декомпозицію, яка дозволить отримати загальне розуміння потреб веб-додатку для звичайного користувача.

Діаграма декомпозиції більш детально показує необхідні для юзера процеси, які необхідно додати до проекту задля того, щоб задовільнити його. Ця діаграма буде використана при розробці Use-case діаграм – кожен процес який є на цій діаграмі має бути описаний в Use-case діаграмі. Результуюча діаграма зображена на рисунку 2.2

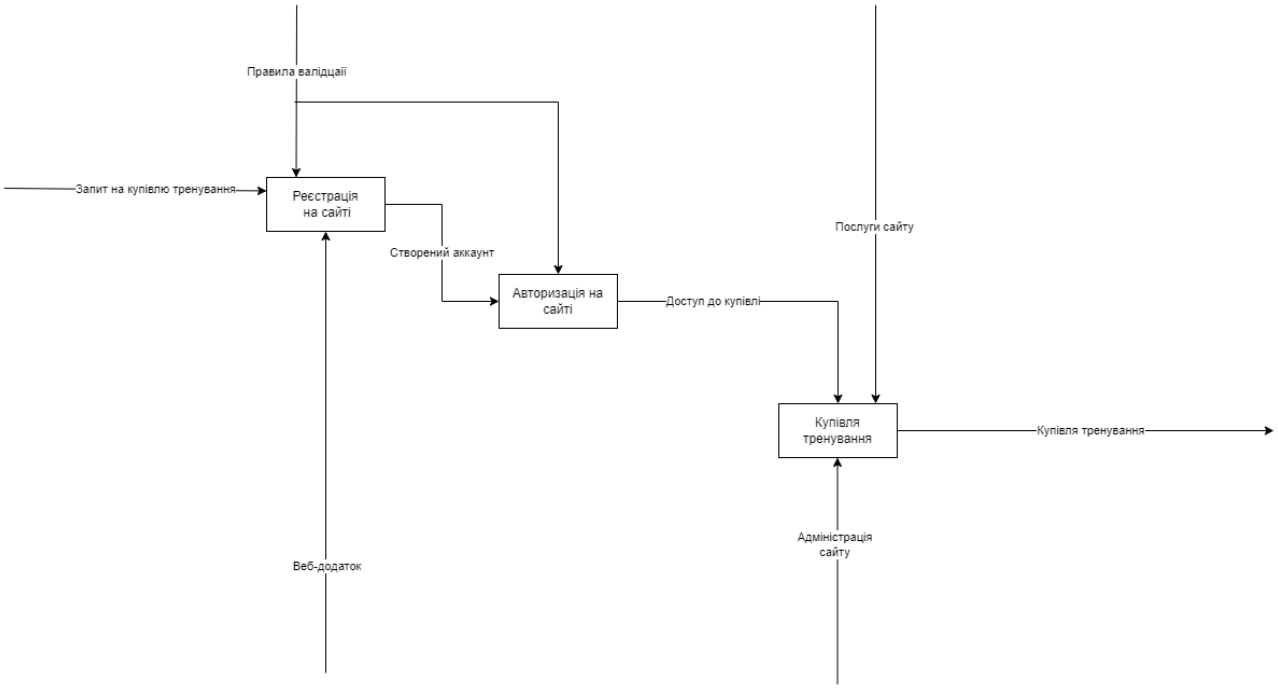

Рисунок 2.2 – Декомпозиція контекстної діаграми

Як видно з діаграми – основними процесами на які необхідно звернути увагу та створити реалізацію є :

- Реєстрація на сайті – звичайна можливість реєстрація, яка дозволяє користувачу авторизуватися і отримати доступ до купівлі тренування
- Авторизація на сайті – процес після якого клієнт отримує можливість купувати тренування
- Купівля тренування – процес який дозволяє клієнту купити необхідне тренування, оскільки для цього необхідно отримати ці тренування необхідно додати функціонал створення тренування.

Оскільки IDEF0 зазвичай не містять деталей реалізації, для спрощення програмної реалізації було створено Use Case діаграми.

Use Case діаграма – це графічне подання взаємодій між акторами разом із системою. Для опису відносин використовуються спеціальні відносини які позначають як елементи та кроки взаємодіють між собою.

У випадку додатку для підбору тренувань необхідно спочатку визначити акторів та можливості використання додатку. Для цього необхідно продумати хто і як буде взаємодіяти з додатком. Акторами які взаємодіють можуть бути:

- Звичайний користувач – це звичайна людина яка має потребу знайти та купити спеціально підібране тренування. Цей юзер хоче як найшвидше та найпростіше підібрати та купити тренування. Окрім цього він може захотіти спробувати безкоштовні тренування перед тим як купити спеціальні тренування.
- Тренер – це спеціальна людина яка має досвід тренування та хоче заробити гроші на цьому. Тренеру необхідна можливість створити та продавати тренування. Окрім цього він має мати можливість також купувати тренування як і юзер.
- Система – це програмна реалізація логіки проекту яка містить можливість опрацювати вхідні данні та зберігати їх в базі даних.
- База даних – це база даних яка зберігає та повертає необхідні дані.

Можливостями які необхідно описати є використані в Рисунку 4 дані.

Use Case діаграма реєстрації (Рисунок 2.3)

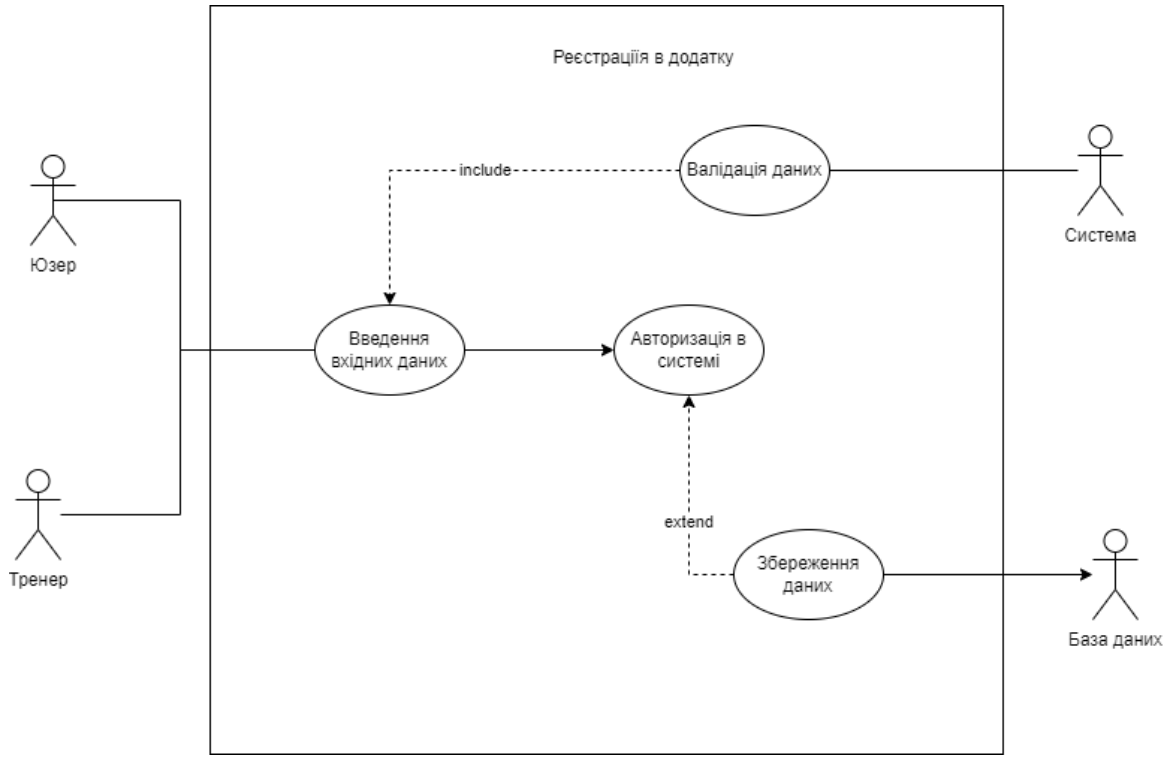

Рисунок 2.3 – Діаграма варіантів використання

Use Case діаграма Авторизації (Рисунок 2.4)

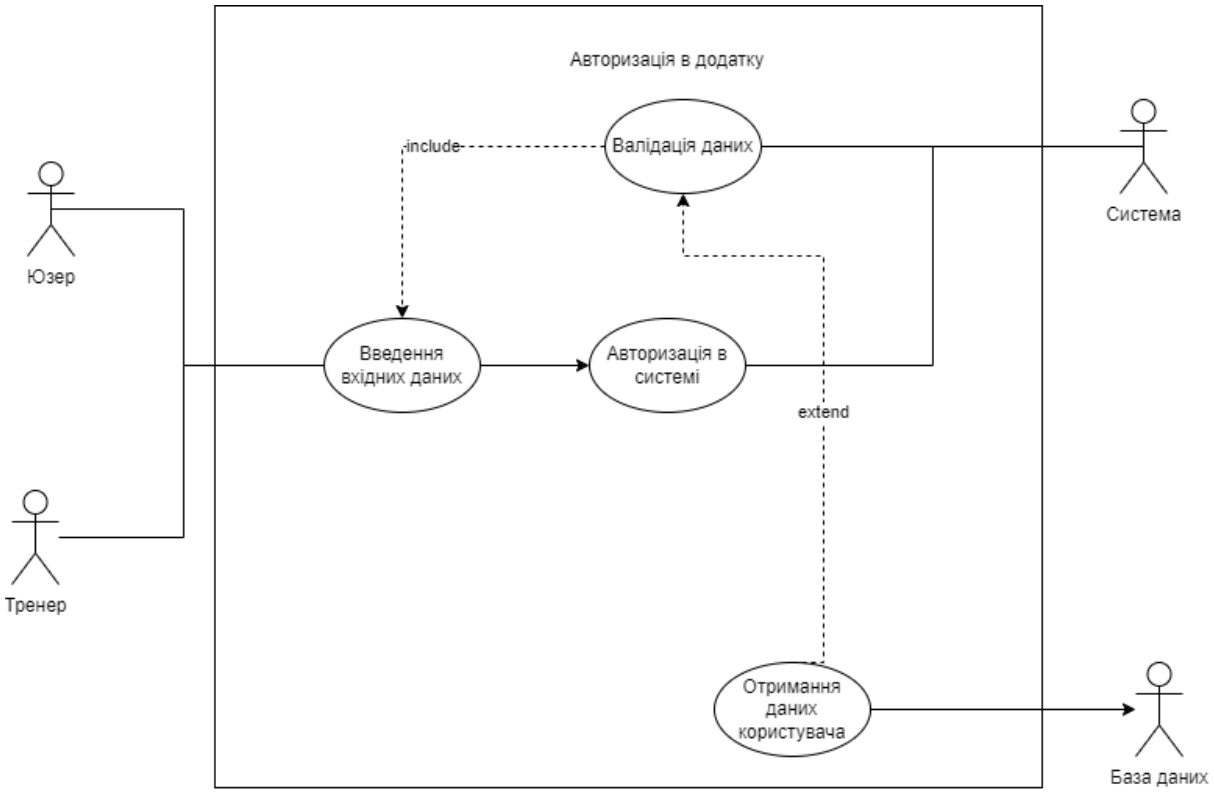

Рисунок 2.4 – Діаграма авторизації

Use Case діаграма Створення тренування (Рисунок 2.5)

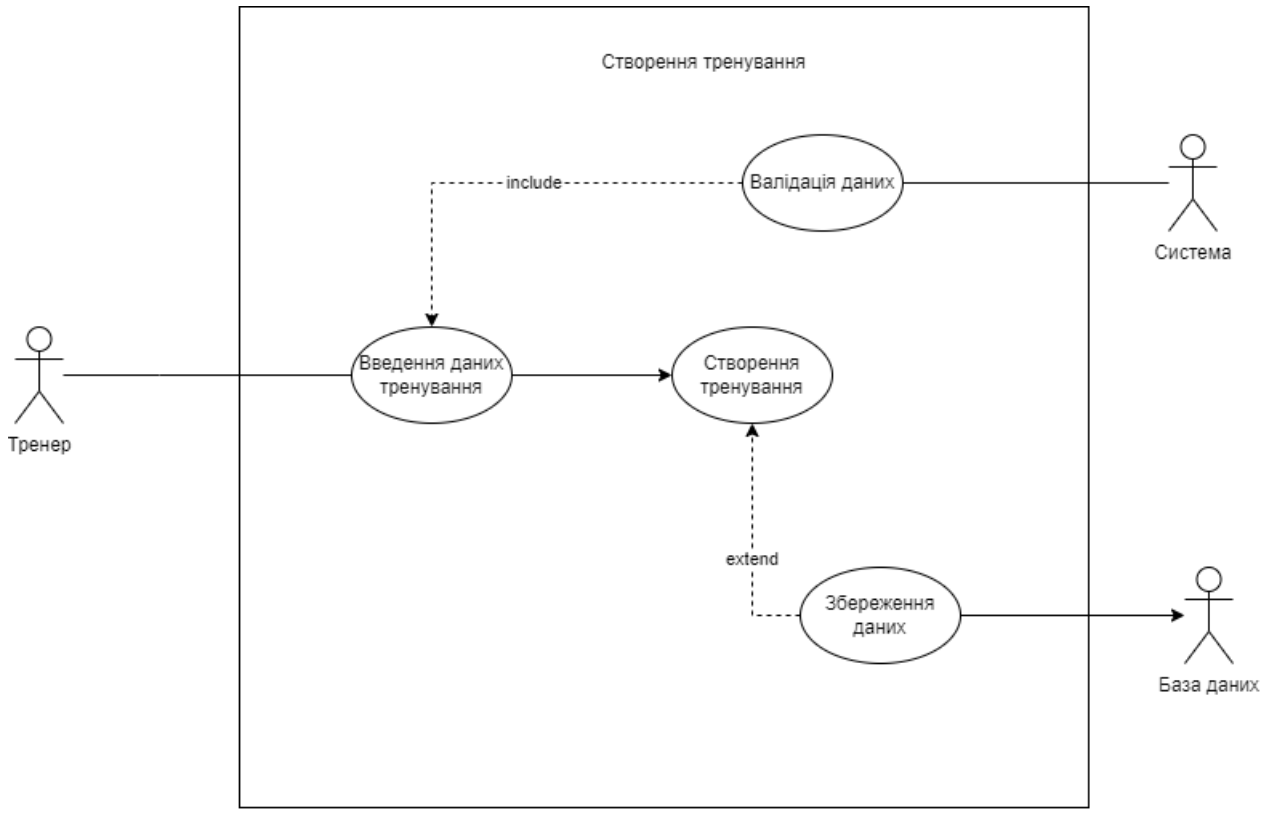

Рисунок 2.5 – діаграма створення тренування

Use Case діаграма Купівля тренування (Рисунок 2.6)

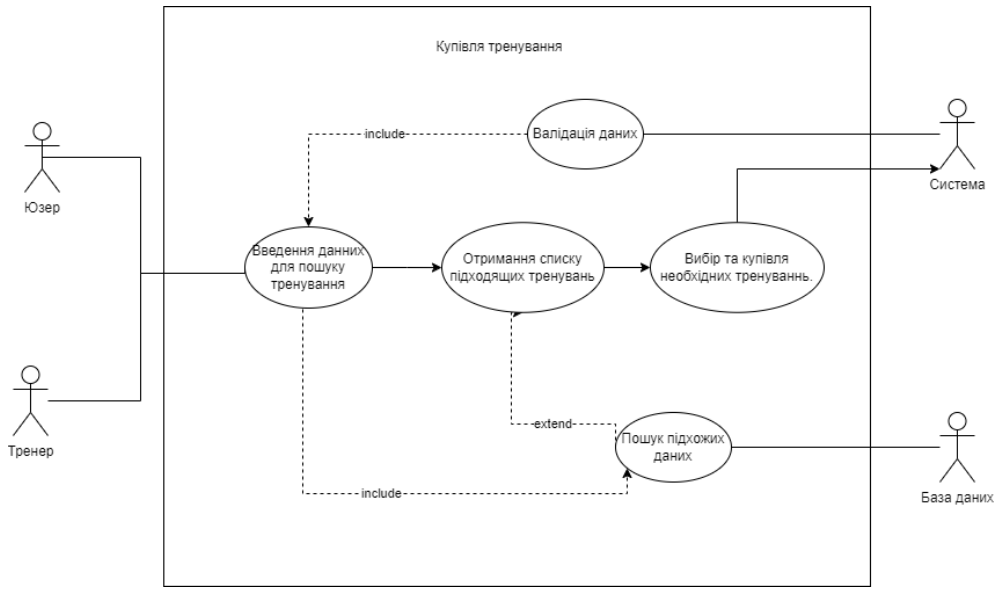

Рисунок 2.6 – діаграма купівлі тренування

Після цього було розроблено фізичну модель бази даних (рисунок 2.7)

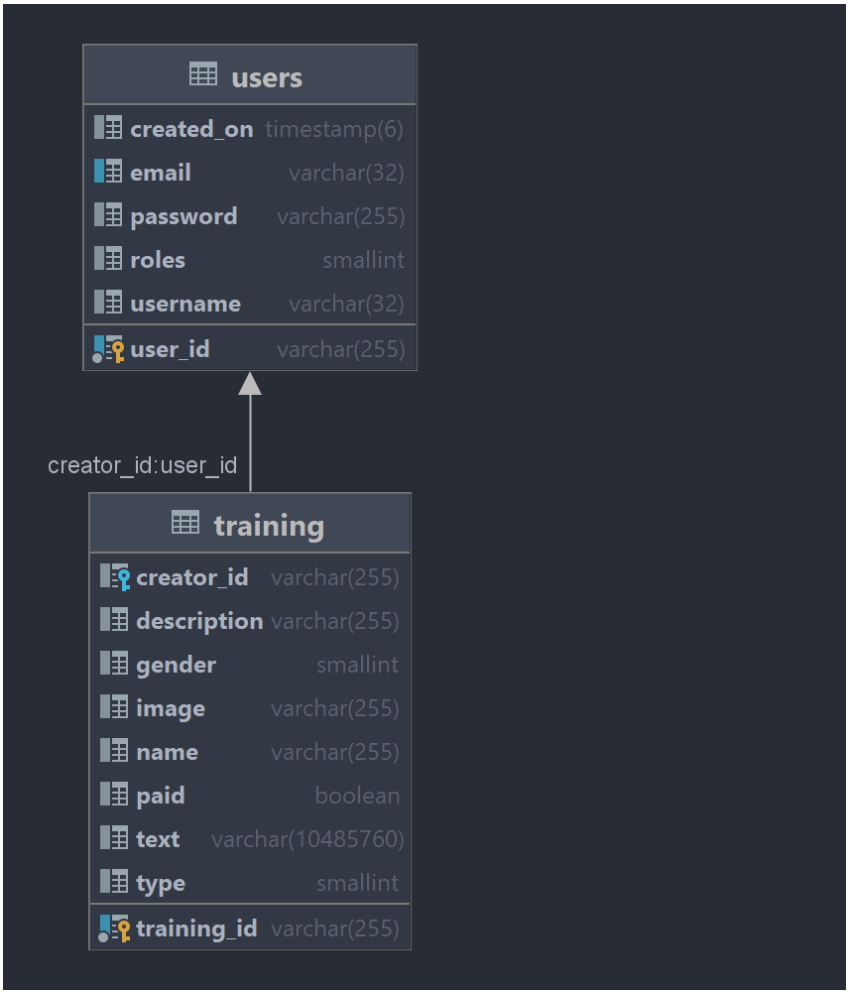

Рисунок 2.7 – Діаграма структури бази даних

Створення IDEF0 та Use Case дозволило значно пришвидшити розробку програмної реалізації та покращити розуміння необхідного додатку. Після їх створення можливо розпочати створення програмного додатку. Для розуміння загальної логіки буде використано IDEF0 діаграму, а реалізація цієї логіки має опиратися на Use Case діаграми. Окрім цього діаграма бази даних допоможе швидко та правильно створити правильні сутності для ініціалізації та роботи з базою даних.

3. Розробка програмного додатку

3.1 Архітектура програмного додатку

Для початку розробки програмного додатку необхідно спочатку визначитися з його архітектурою, для цього необхідно визначитися які технології будуть використані та як вони будуть взаємодіяти між собою.

Для розробки було використано шаблон проектування MVC (Model View Controller) , який означає що в програмному додатку буде окремо виділена бізнес-логіка яку називають Model, реалізація вивід цих даних – View та Controller який виступає посередниками між ними. Для взаємодії через інтернет використано REST API, що дозволяє винести Model на окремий сервер та налаштувати із нею зв’язок через мережу.

Для реалізації шаблону MVC було використано:

- Model та Controller - мову програмування Java разом із Spring framework. Для реалізації Model необхідно створити так звані сервіси, які будуть отримувати дані, робити з ними необхідні операції та повертати результат. Для Controller в Spring framework існує спеціальна анотація @Controller Яка дозволяє створювати обробник запитів за визначеним шляхом.
- Для збереження даних - базу даних PostgreSQL з якою буде взаємодіяти лише сервіси з Java частини.
- За реалізацію View буде відповідати JavaScript з фреймворком React які будуть відповідати за рендер необхідних результатів та сторінок для навігації юзера з яких будуть збиратися необхідні дані та посилатися до контролеру.

Деталі їх взаємодії між собою було представлено в попередньому пункті у вигляді IDEF0 та Use-Case діаграм (рисунок 2.1, 2.2, 2.3, 2.4 ,2.5 та 2.6 відповідно).

3.2 Програмна реалізація

Для розробки програмної реалізації було створено Model та Controller частину. Спочатку необхідно створити додаток який буде обробляти необхідні данні. Для цього необхідно ініціалізувати пустий Spring додаток, для цього було використано можливості IntelliJ IDEA (рисунок 3.1).

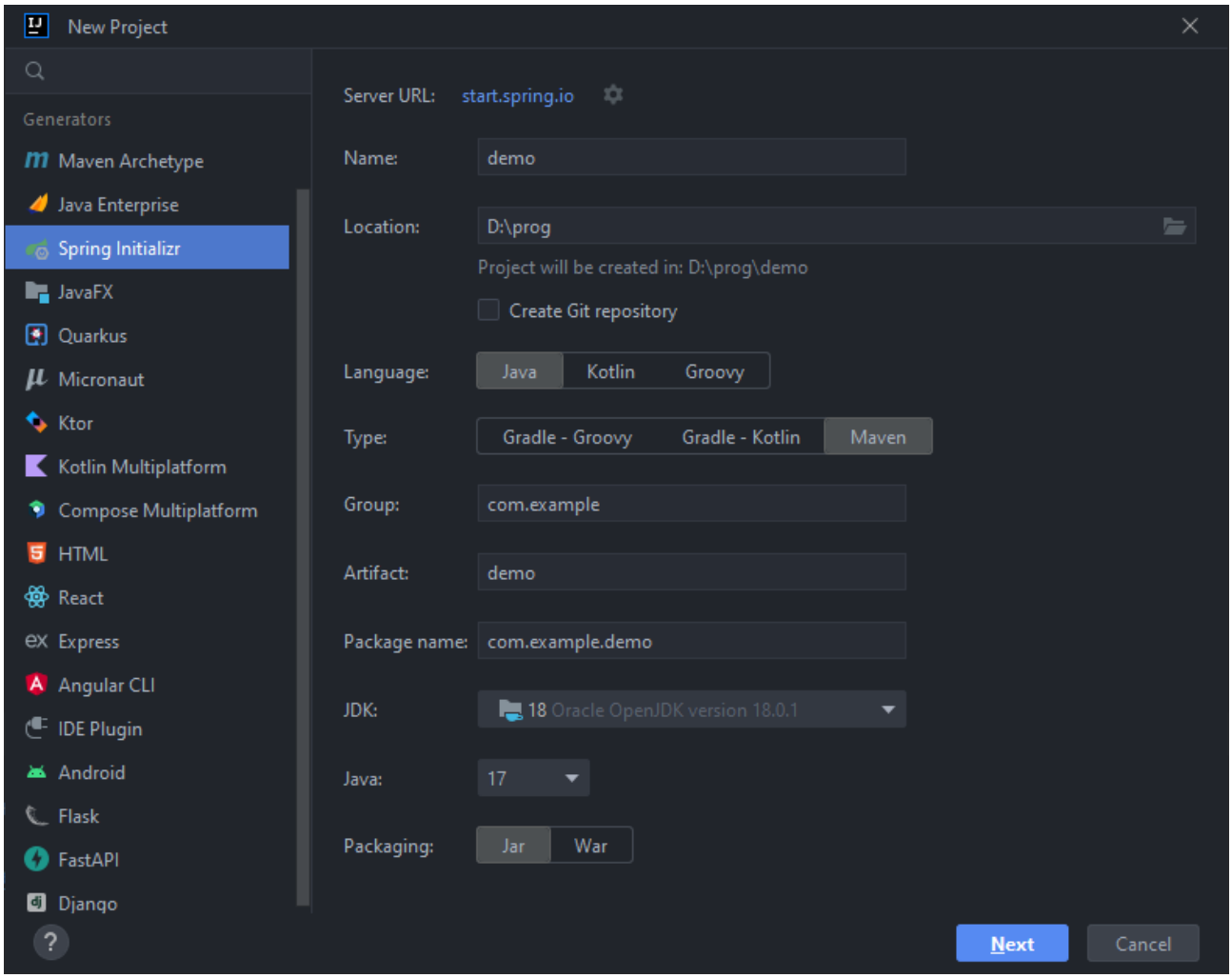

Рисунок 3.1 - ініціалізування проекту

Після цього необхідно конфігурувати pom.xml в якому зазначити загальні налаштування проекту (версія проекту, назва, груп айді, тощо) – рисунок 3.2 та підключити необхідні плагіни та бібліотеки – рисунки 3.3 та 3.4 відповідно.

```xml
<modelVersion>4.0.0</modelVersion>

<groupId>com.oleksandr</groupId>
<artifactId>sport-shop</artifactId>
<version>1</version>
<packaging>jar</packaging>

<name>sport-shop-back</name>

<properties>
  <org.mapstruct.version>1.5.3.Final</org.mapstruct.version>
  <org.projectlombok.version>1.18.24</org.projectlombok.version>
  <org.commons-io.version>2.11.0</org.commons-io.version>
  <maven-compiler.plugin.version>3.8.1</maven-compiler.plugin.version>
  <org.springdoc.version>1.7.0</org.springdoc.version>
  <springboot.version>3.0.5</springboot.version>
  <java.version>17</java.version>
  <maven.compiler.source>${java.version}</maven.compiler.source>
  <maven.compiler.target>${java.version}</maven.compiler.target>
  <org.maven-compiler-plugin.version>3.8.1</org.maven-compiler-plugin.version>
  <nimbus-jose-jwt.version>9.31</nimbus-jose-jwt.version>
  <springdoc-openapi-starter-webmvc-ui.version>2.1.0</springdoc-openapi-starter-webmvc-ui.version>
  <spring-boot-maven-plugin.version>2.7.4</spring-boot-maven-plugin.version>
  <firebase-admin.version>9.1.1</firebase-admin.version>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>${springboot.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Рисунок 3.2 – налаштування проекту

```xml
<dependencies>
  <!-- Project dependencies -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
  </dependency>

  <!--    Firebase dependency-->
  <dependency>
    <groupId>com.google.firebase</groupId>
    <artifactId>firebase-admin</artifactId>
    <version>${firebase-admin.version}</version>
  </dependency>
```

Рисунок 3.3 – підключення плагінів

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>${spring-boot-maven-plugin.version}</version>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Рисунок 3.4 – підключення бібліотек

Потім можливо перейти до самої розробки. Спочатку було реалізовано структуру бази даних у вигляді класів. Для цього було використано ORM бібліотеку яку було підключено раніше, що дозволяє автоматизувати ініціалізацію бд, та запити до неї. Задля цього було створено окремі пакети в які необхідно додати реалізації(рисунок 3.5).

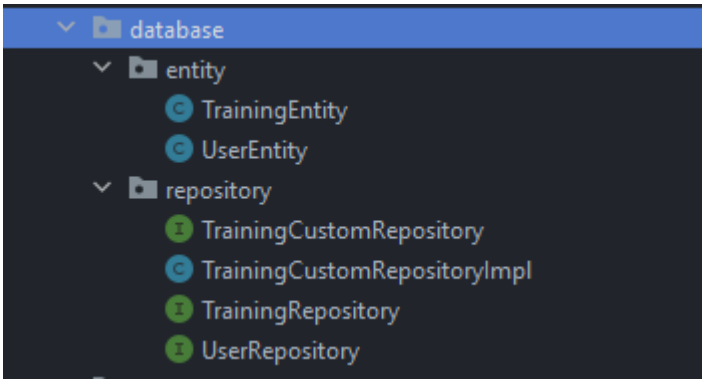

Рисунок 3.5 – структура бд

Приклад реалізації сутності наведено на рисунку 3.6, а репозиторію на рисунку 3.7

```java
public class UserEntity {
  @Id
  @GeneratedValue(generator = "uuid")
  @GenericGenerator(name = "uuid", strategy = "uuid2")
  @Column(name = "user_id")
  private String id;

  @Column(name = "email", length = 32, unique = true)
  private String email;

  @Column(name = "username", length = 32)
  private String username;

  @Column(name = "password")
  private String password;

  @ToString.Exclude
  @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy = "creatorId")
  private List<TrainingEntity> trainings;

  @Column(name = "roles")
  @Enumerated(EnumType.ORDINAL)
  private Roles role;

  @Column(name = "created_on")
  private LocalDateTime createdOn;

  /** Roles. */
  8 usages  Kazlnacheiev
  public enum Roles {
    2 usages
    ROLE_USER,
    3 usages
    ROLE_TRAINER
  }
}
```

Рисунок 3.6 – приклад реалізації сутності UserEntity

```java
/** UserRepository. */
2 usages  Kazlnacheiev
@Repository
public interface UserRepository extends JpaRepository<UserEntity, String> {
  2 usages  Kazlnacheiev
  Optional<UserEntity> getUserEntityByEmail(String email);

  1 usage  Kazlnacheiev
  @Transactional
  @Modifying
  @Query("update UserEntity u set u.password = :newPassword where u.email = :email")
  void findByEmailAndChangePassword(
      @Param(value = "email") String email, @Param(value = "newPassword") String newPassword);
}
```

Рисунок 3.7 – приклад реалізації репозиторію UserRepository

Після цього в файл application.properties було додано строку spring.jpa.hibernate.ddl-auto=update яка буде автоматично створювати/оновлювати структуру бд під час кожного запуску додатку. Окрім цього в цьому ж файлі було указано дані необхідні для підключення до бази(url, username та password).

Далі було створено файли конфігурації, які відповідають за безпеку додатку(хешування паролю, налаштовує ролі для доступу до необхідних ендпоінтів, перевірку токену, необхідні конвертори, тощо) – рисунок 3.8.

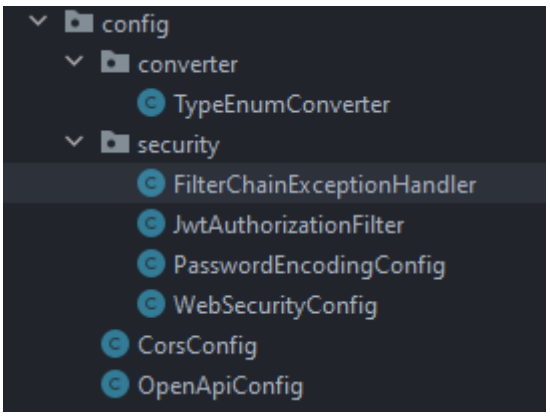

Рисунок 3.8 – файли конфігурації

Реалізація цих конфігурацій дозволяє реалізувати контролери в яких за спеціальним шляхом будуть налаштовані необхідні REST методи. Запити за допомогою спеціального маперу будуть підставлені до спеціальних DTO (Data Type Object) в яких указані правила для валідації даних та при помилках за допомогою exception handler`у який їх обробляє виводити результат із коректними результатами або помилку із 400 статус кодом. Усі ці класи винесено в окремі пакети – рисунок 3.9.

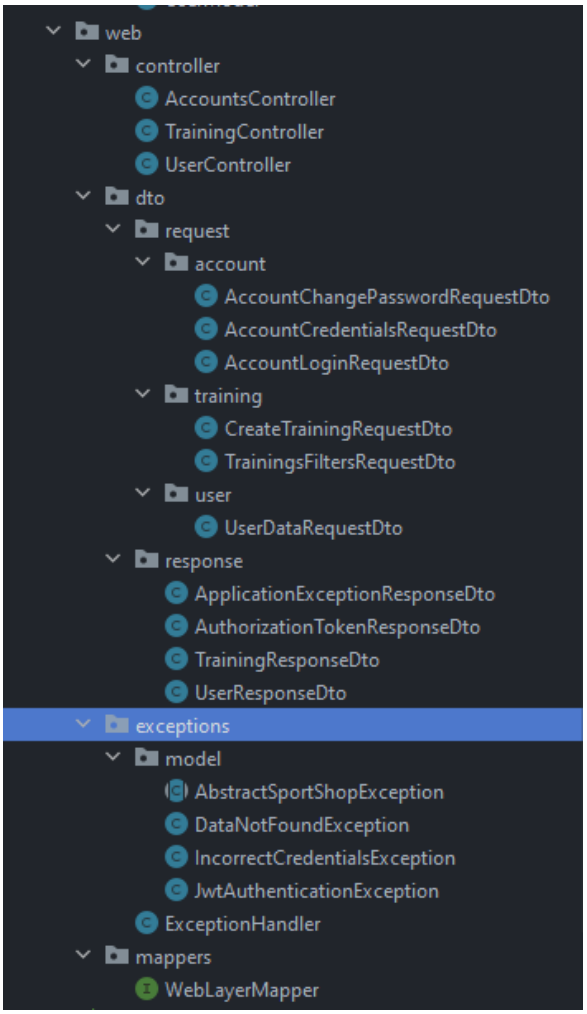

Рисунок 3.9 – структура Controller класів

Після цього було реалізовано сервіси, які конвертували DTO до моделей з якими вони працювали. Такий підхід дозволяє чітко розмежувати дані, за допомогою чого при зміні структури розробники будуть мати можливість швидко змінити необхідні моделі не зважаючи на DTO та навпаки, що сприяє покращенню масштабуванню проекту в майбутньому.

Ці сервіси було також винесено в окремий пакет – рисунок 3.10.

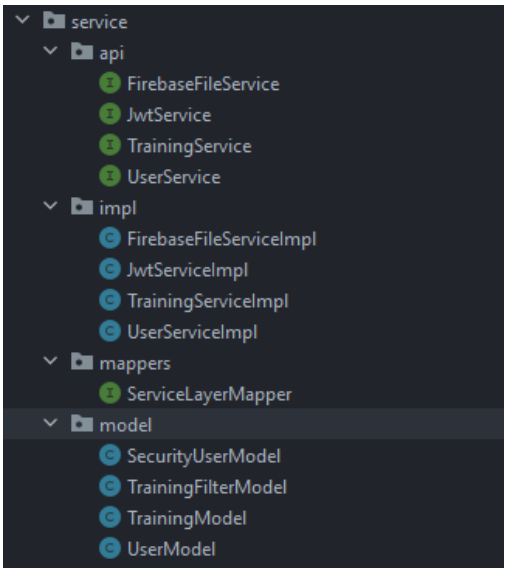

Рисунок 3.10 – структура Model класів

За допомогою підключеного раніше Swagger можна подивитися на отриману структуру проекту разом із необхідними для фронт-енд частини даними, що спростить її розробку(рисунок 3.11).

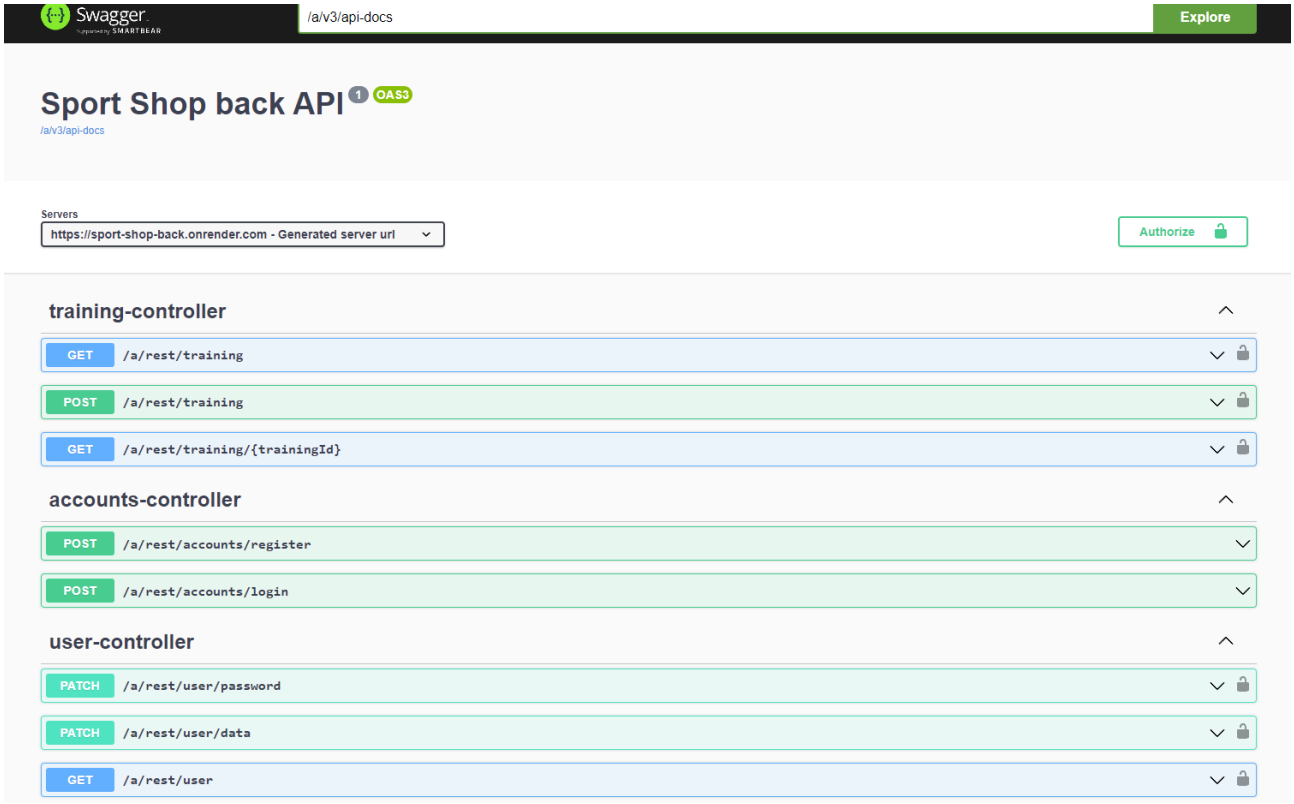

Рисунок 3.11 - Swagger

Маючі цю структуру, можливо розпочати розробку View частини додатку, для цього спочатку необхідно ініціалізувати React проект. Задля цього необхідно виконати команду npx create-react-app sportshop-fe яка створить пусти проект – рисунок 3.12.

```
$ npx create-react-app sportshop-fe

Creating a new React app in C:\Users\vovas\OneDrive\Рабочий стол\диплом\sportsho
p-fe.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1434 packages in 1m

234 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 62 packages, and changed 1 package in 20s

234 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1496 packages in 3s

234 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created sportshop-fe at C:\Users\vovas\OneDrive\Рабочий стол\диплом\spo
rtshop-fe
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd sportshop-fe
  npm start

Happy hacking!
```

Рисунок 3.12 – ініціалізація React проекту

В проекті було використано бібліотек Material UI, яка надає можливість використовувати готові компоненти з можливістю їх змінювати. Для її використання достатньо виконати команду npm install @mui/material з необхідними елементами. Для того, щоб запустити проект необхідно виконати команду npm start, яка запустить проект та буде оновлювати зміни в реальному часі.

Спочатку було редаговано файл index.js з головними налаштуваннями проекту, в ньому було створено спеціальну тему яка використовувалася на всьому проекті – рисунок 3.13.

```
const theme = createTheme({
    palette: {
        primary: {
            main: "#FF464C",
            contrastText: "#fff",
        },
    },
    overrides: {
        MuiButton: {
            contained: {
                color: "#FF464C",
                backgroundColor: "#fff",
                "&:hover": {
                    backgroundColor: "#FF6E74",
                    "@media (hover: none)": {
                        backgroundColor: "#fff",
                    },
                },
            },
        },
    },
});

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
    <BrowserRouter>
        <ThemeProvider theme={theme}>
            <App />
        </ThemeProvider>
    </BrowserRouter>
);
```

Рисунок 3.13 - index.js

Та після цього налаштовано App.js, в якому уточнено необхідні маршрути для проекту – рисунок 3.14

```jsx
export default function App() {
    return (
        <div>
            <Routes>
                <Route path="/" element={<Layout />}>
                    <Route index element={<MainPage />} />
                    <Route path="login" element={<SignInPage />} />
                    <Route path="register" element={<SignUpPage />} />
                    <Route path="trainings" element={<FreeTrainings />} />
                    <Route path="paid-trainings" element={<PaidTrainings />} />
                    <Route
                        path="choose-trainings"
                        element={<ChooseTrainings />}
                    />
                    <Route
                        path="trainings/:trainingId"
                        element={<TrainingsDetails />}
                    />
                    <Route
                        path="trainings/create"
                        element={<CreateTraining />}
                    />
                    <Route path="cart" element={<Cart />} />
                    <Route path="cart-success" element={<CartSuccess />} />
                    <Route path="profile" element={<AccountProfilePage />} />
                    <Route path="*" element={<NoMatch />} />
                </Route>
            </Routes>
        </div>
    );
}
```

Рисунок 3.14 - App.js

Далі було створено спеціальні папки, в яких зберігаються статичні ресурси(рисунки лого, бекграунд, тощо) – рисунок 3.15.

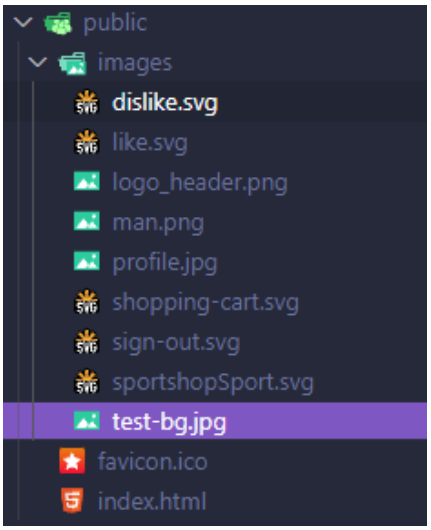

Рисунок 3.15 – структура статичних ресурсів

Для відображення даних в React використовуються спеціальні .jsx файли в яких уточнюється розмітка сайту, та його логіка за допомогою коду. Окрім цього розмітку сайту можливо корегувати за допомогою css файлів, які потім підключаються до розмітки.

Першими було створено розмітку сторінки в якій указана структура сторінки, яка складається з футеру, хедеру, та головною сторінки. Для цього було створено окремі файли в яких указана ця розмітка, та налаштування футеру та хедеру(рисунок 3.16).

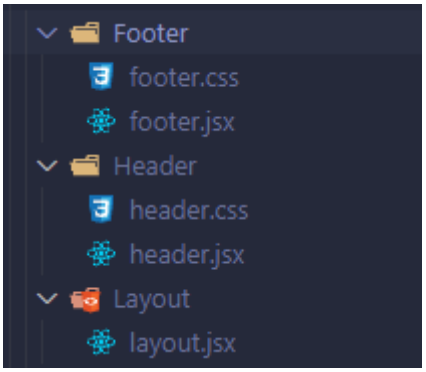

Рисунок 3.16 – розмітка, футер та хедер

Після цього можливо перейти до створення контенту сторінок. Для цього було стоврено окрему папку зі сторінками які необхідні всередині яких задається візуальна структура – рисунок 3.17

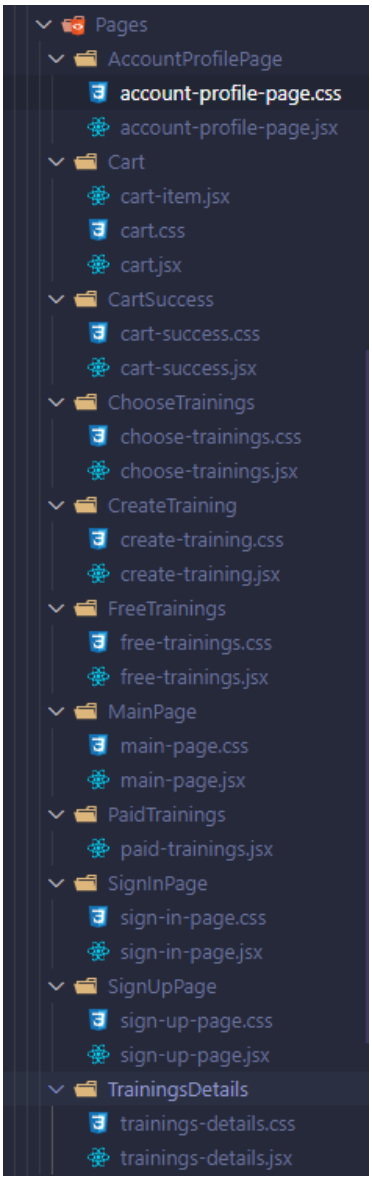

Рисунок 3.17 – структура сторінок

Оскільки ці сторінки мають взаємодіяти із Controller частиною додатку, всередині сторінок було створено запити до розроблених раніше ендпоінтів, та залежно від результатів відображено помилку, або необхідний для юзеру результат.

Після налаштування кожної сторінки було розроблено працюючий додаток який відповідає вимогам зазначеним раніше. Цей додаток має можливість як бути запущеним локально, так і буде розгорнутим на сервері, для цього достатньо лише змінити відповідні дані в Java частині додатку.

3.3 Використання програмного додатку

Під час першого відвідування сайту користувач опиниться на головній сторінці з двома кнопками – авторизації та реєстрації – рисунок 3.18.

Рисунок 3.18 – головна сторінка

Якщо корситувач ще не користувався додатком, йому необхідно зареєструватися, для цього необхідно заповнити відповідну форму – рисунок 3.19.

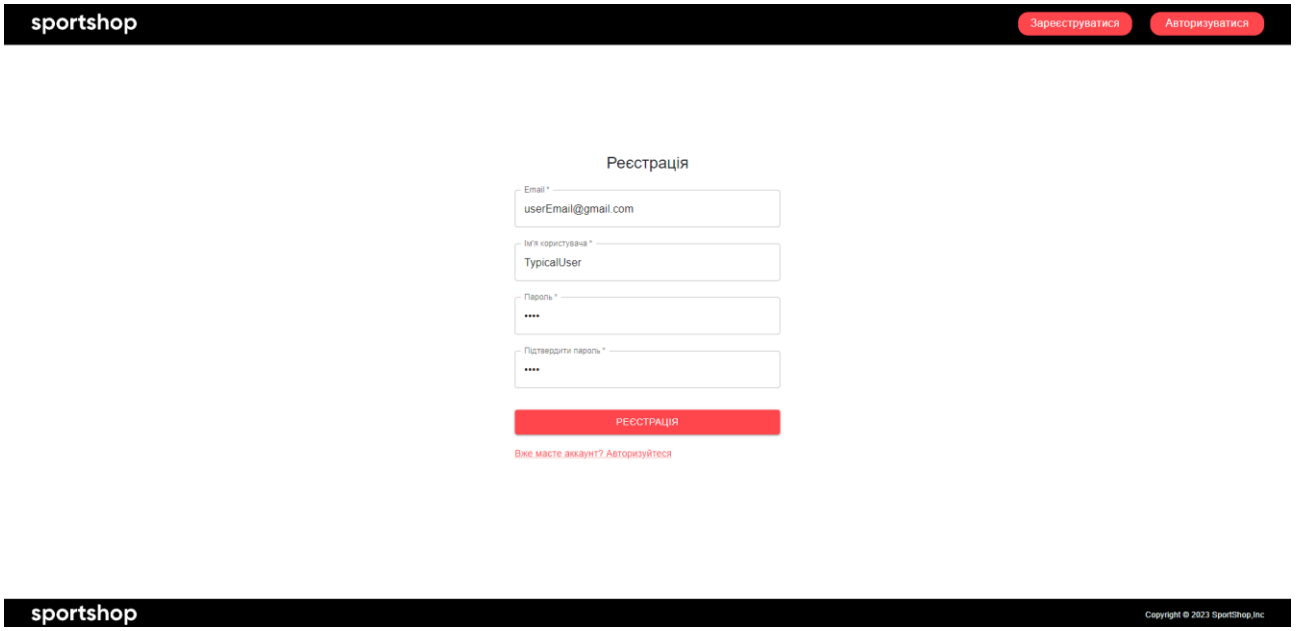

Рисунок 3.19 - реєстрація

Після цього юзер заходить в його аккаунт та перекидає на сторінку з його профілем – рисунок 3.20.

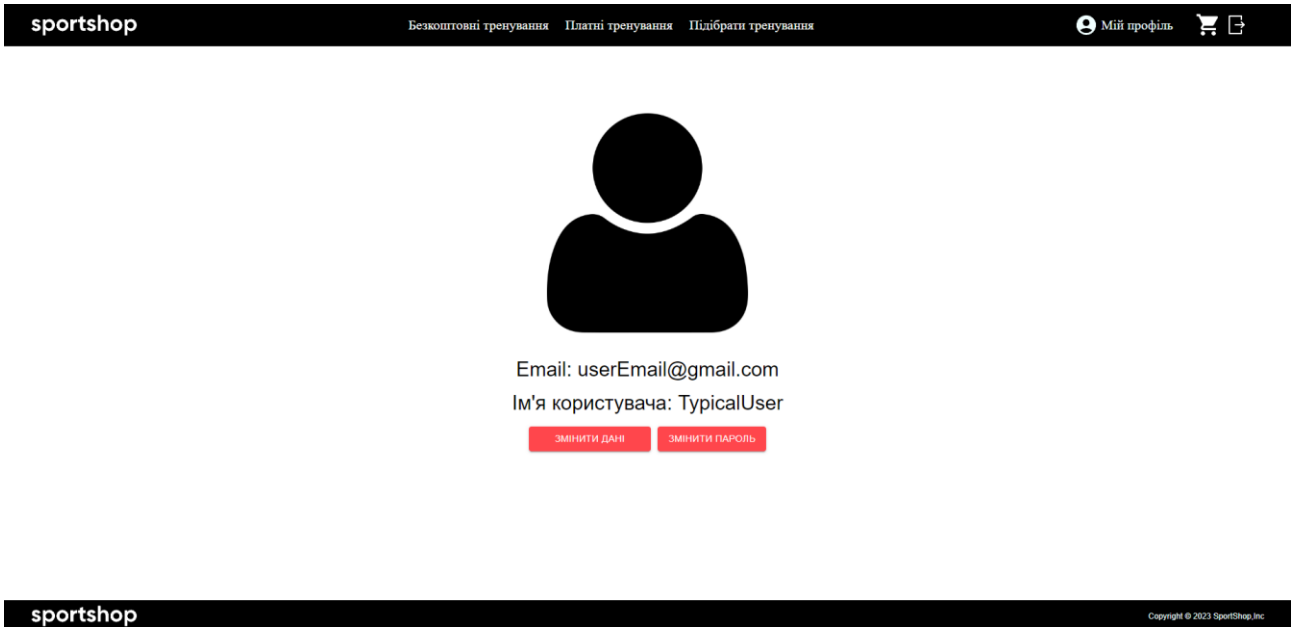

Рисунок 3.20 – профіль користувача

Якщо корсистувач зареєстрований, він має можливість авторизуватися – рисунок 3.21

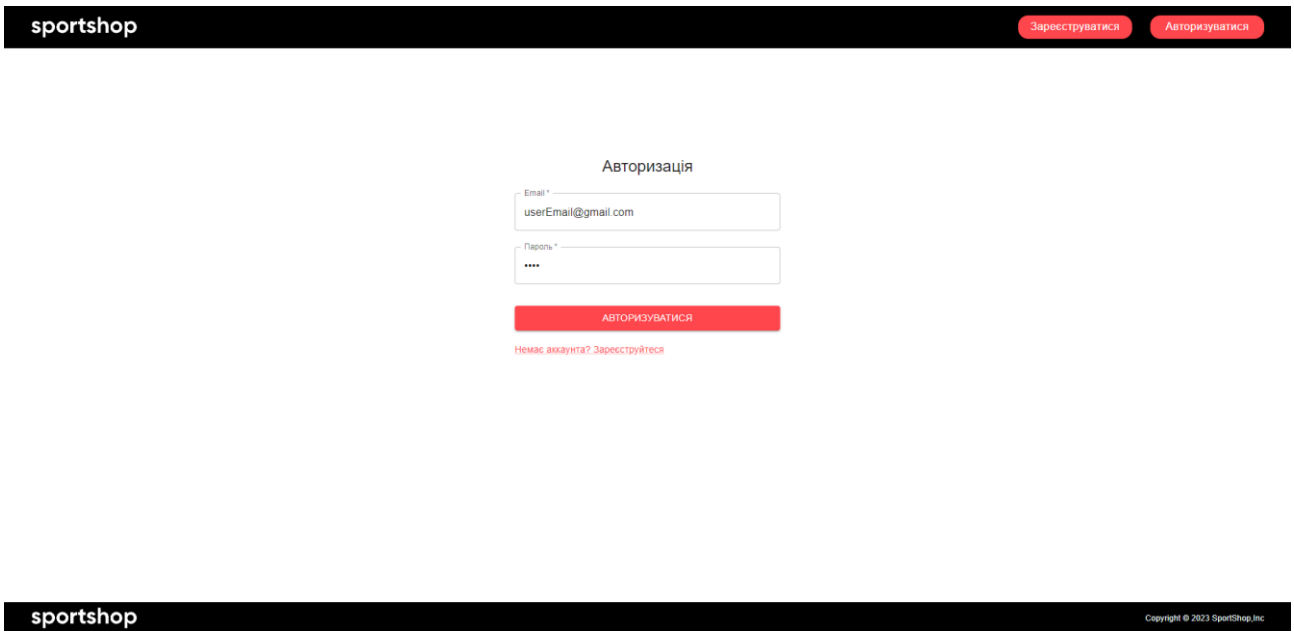

Рисунок 3.21 - авторизація

Після цього для юзера відкривається три вкладки – безкоштовні, платні тренування та підбір тренування (рисунки 3.22, 3.23 так 3.24 відповідно).

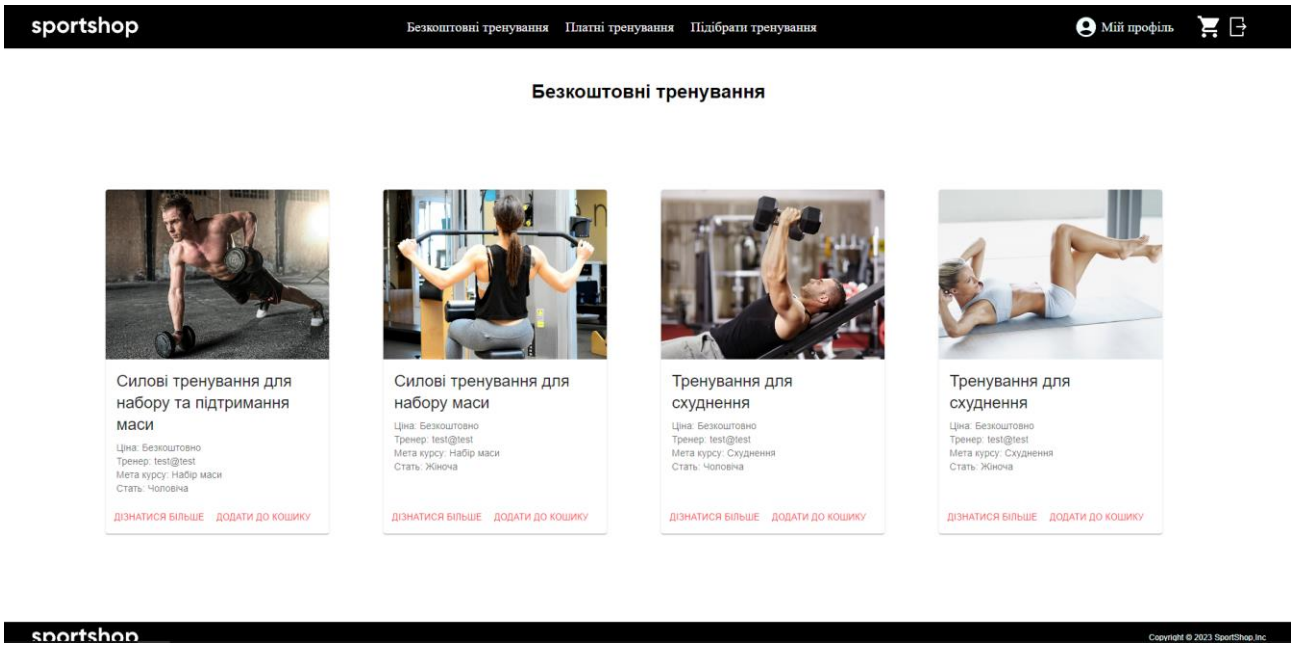

Рисунок 3.22 – безкоштовні тренування

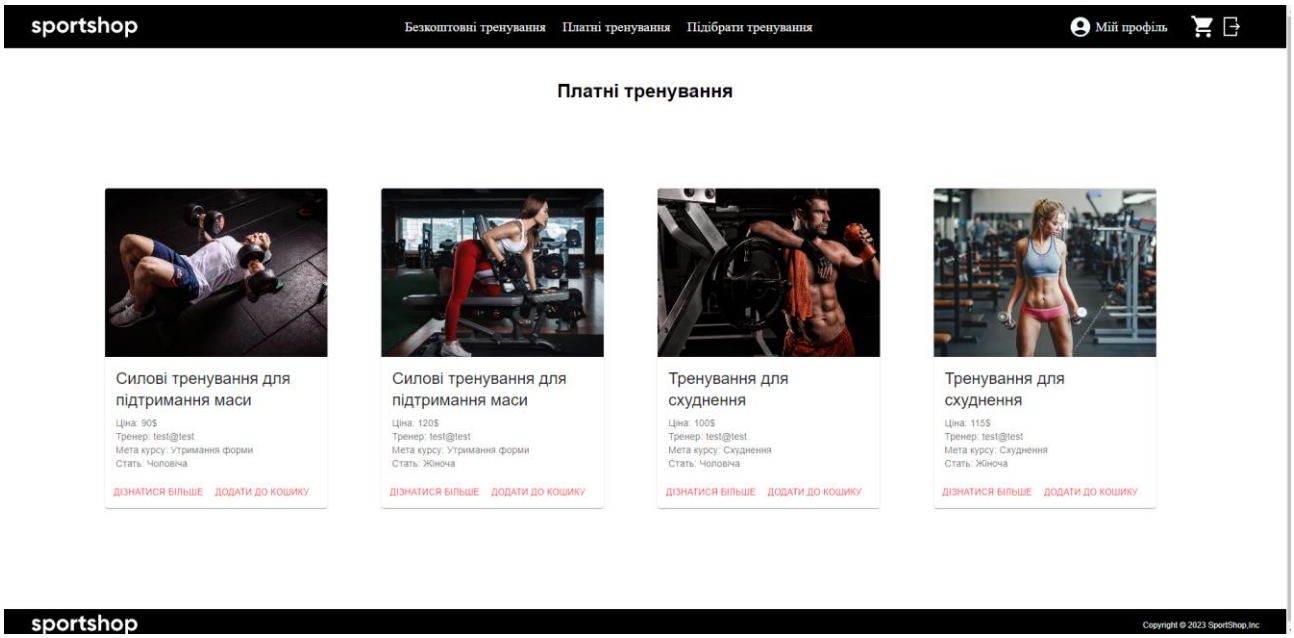

Рисунок 3.23 – платні тренування

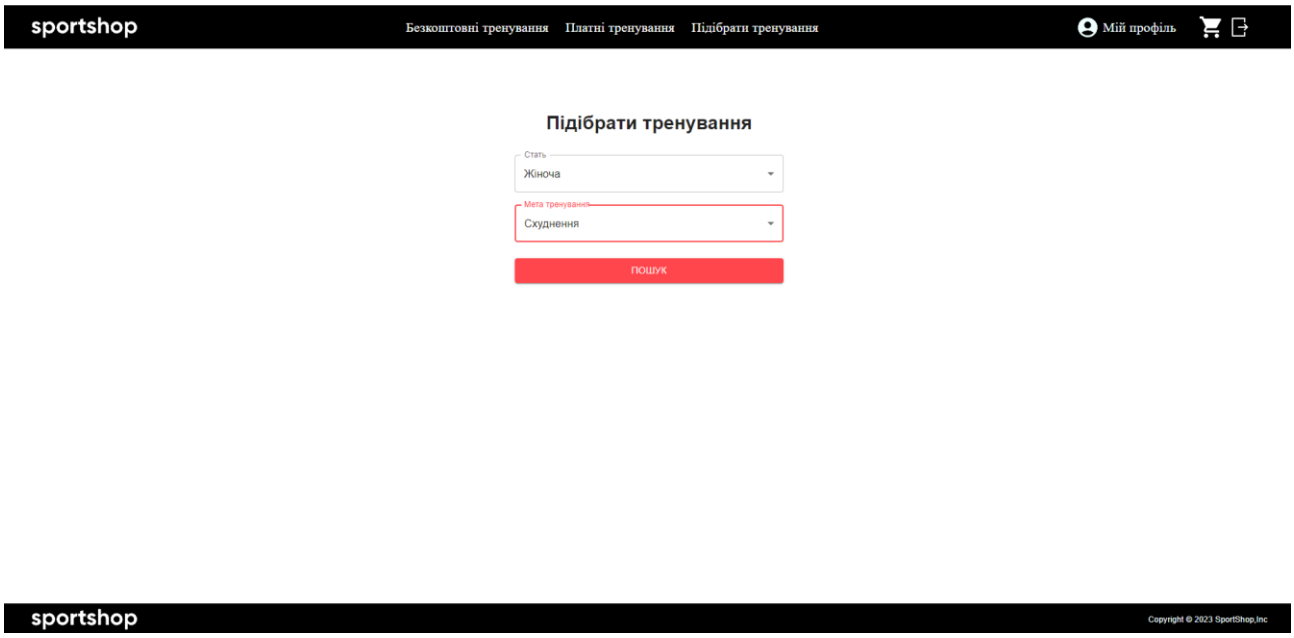

Рисунок 3.24 – підбір тренування

На цих сторінках користувач має можливість подивитися існуючі тренування у вигляді списку, або підібрати необхідне для нього за допомогою спеціальної форми. Після цього юзер має можливість подивитися деталі тренування та додати його до кошику якщо воно підійшло йому(рисунки 3.25)

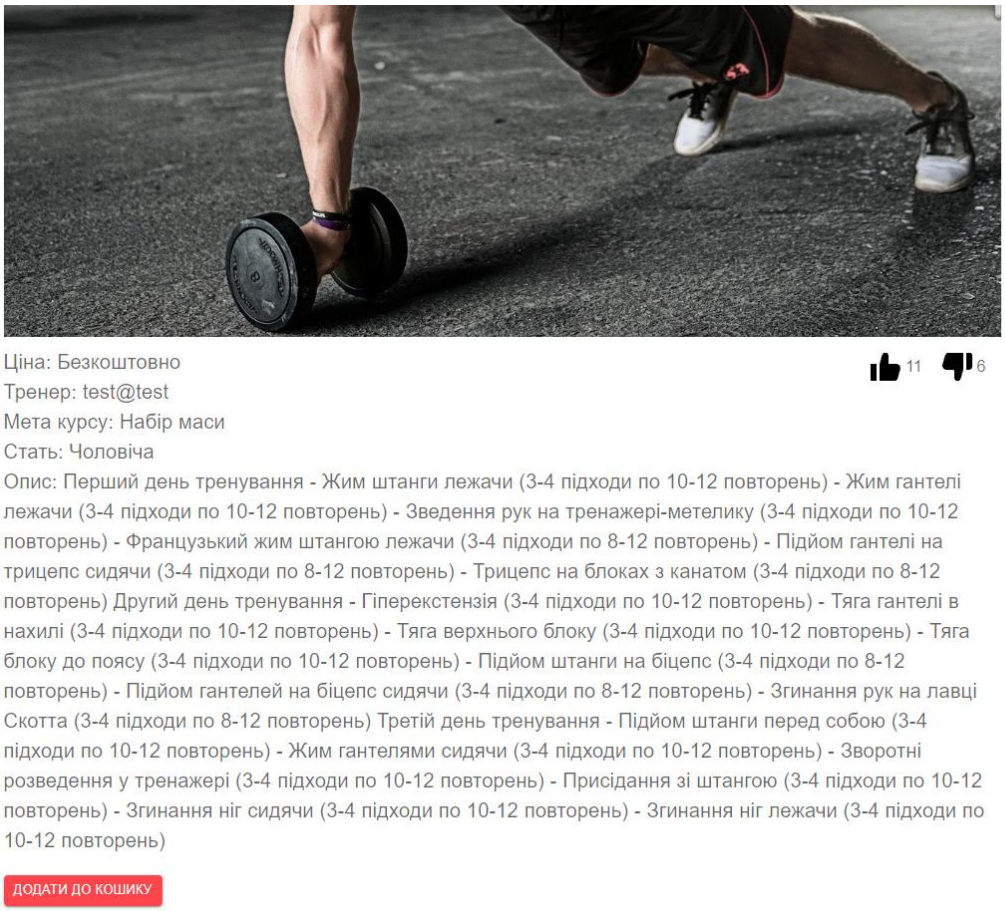

Рисунок 3.25 – деталі тренування

Після завершення юзер має можливість подивитися загальну ціну тренування та замовити їх – рисунок 3.26.

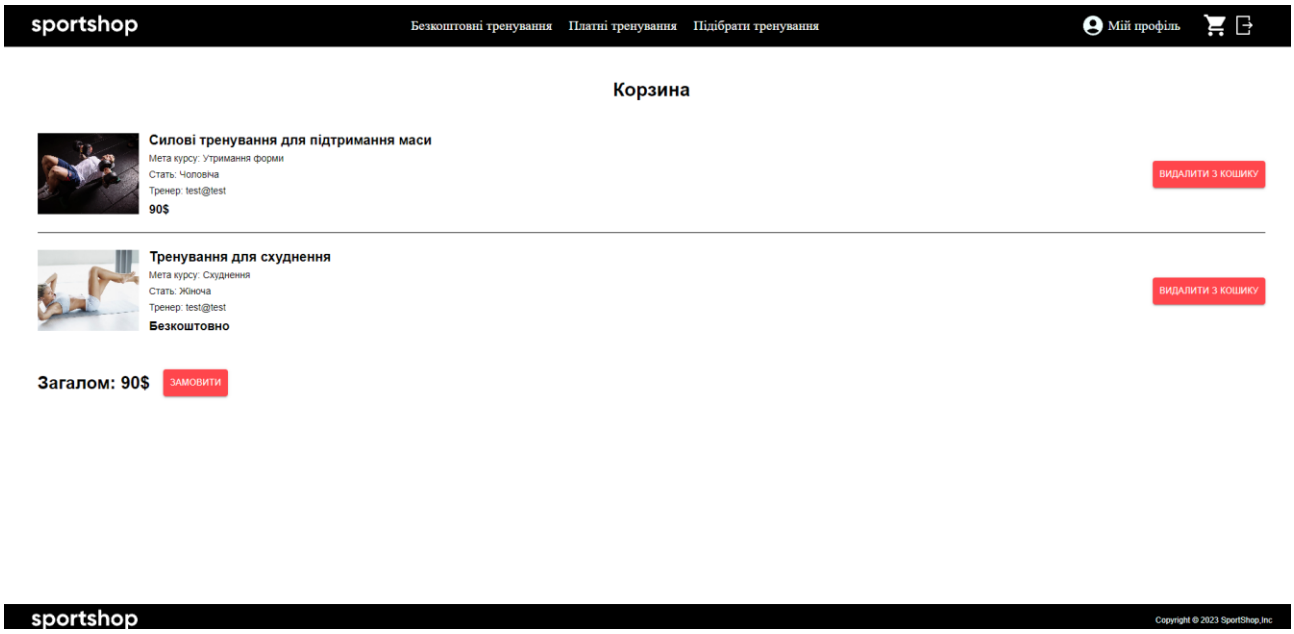

Рисунок 3.26 - корзина

Для тренерів після авторизації стає доступною окрема вкладка з назвою створити тренування, в якій теренер має можливість створити нове тренування, яке потім зможуть купити інші юзери.

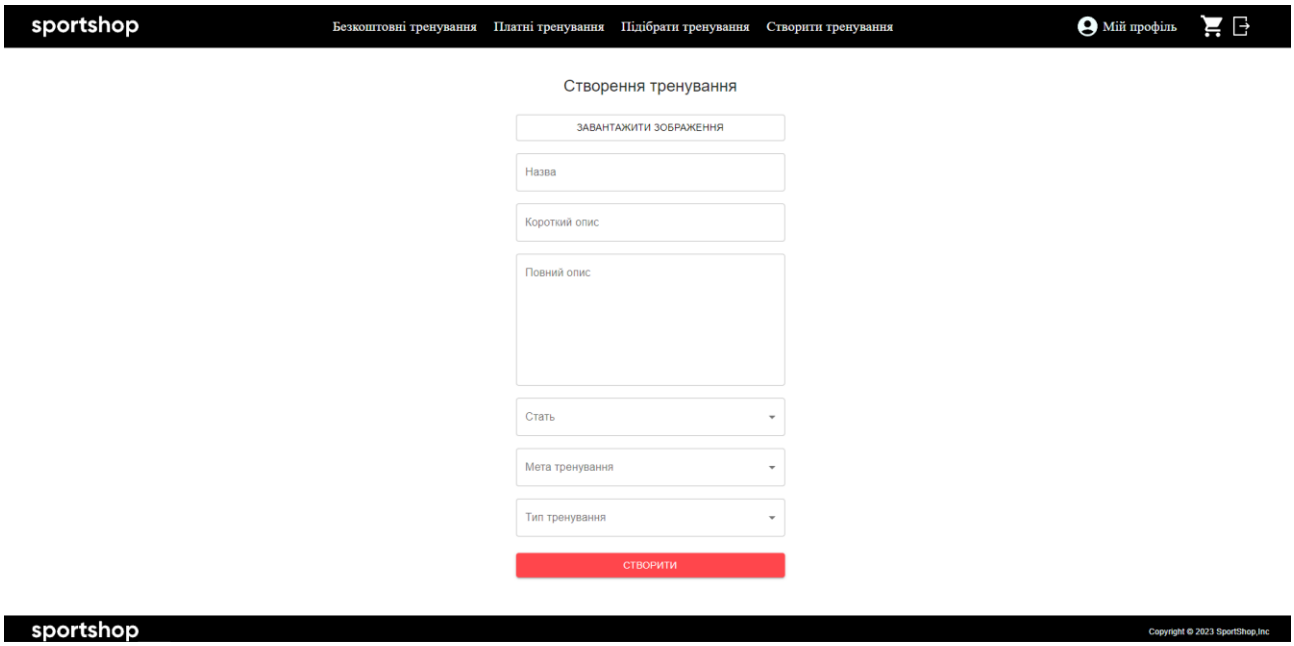

Рисунок 3.27 – створення тренування

Як видно з наведених скріншотів, було створено повноцінний та функціонуючий сайт, який відповідає поставленим вимогам.

ВИСНОВКИ

У ході виконання даної роботи було розроблено web-додаток для планування тренувань, який має актуальність і значущість у суспільстві, де все більше людей усвідомлюють важливість здорового способу життя та фізичної активності. Веб-додаток надає користувачам можливість створювати та налаштовувати персоналізовані тренувальні програми, відстежувати виконання вправ та прогресу, а також отримувати рекомендації для оптимізації тренувального процесу.

У процесі розробки програми було визначено функціональні та нефункціональні вимоги, проведено дослідження технологій, розроблено дизайн та інтерфейс програми, реалізовано функції.

В результаті вийшов інноваційний та корисний веб-додаток, який дозволяє користувачам ефективно планувати свої тренування. Користувачі можуть створювати персоналізовані програми тренувань. Додаток також надає рекомендації та поради для оптимізації тренувального процесу, підвищуючи мотивацію та допомагаючи досягти бажаних результатів.

Проведене дослідження під час виконання цієї роботи підтверджують актуальність і значущість веб-додатків для планування тренувань, а також показують можливості розробки та реалізації таких програм з використанням сучасних технологій. Подальший розвиток та дослідження в цій галузі можуть зосередитися на покращенні алгоритмів рекомендацій, інтеграції з додатковими пристроями та датчиками для більш точного моніторингу фізичної активності, а також на створенні соціальних функцій та можливостей для взаємодії між користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Арлоу, Д. UML 2 та Уніфікований процес. Друге видання/Д. Арлоу, А.Нейштадт. - 2006. - 336 с.

2. Майєрс, Г. Мистецтво тестування програм/Г. Майєрс; пров. з англ. - М.: Фінанси та статистика, 1982. - 176 с.

3. Тестування програмного забезпечення. Фундаментальні концепції управління бізнес-додатків: Пер. з англ. / Сем, Джек Фолк, Енг Кек Нгуєн. - , 2001. - 544 с.

4. Електроний ресурс: https://www.medindia.net/patients/lifestyleandwellness/top-7-benefits-of-using-fitness-apps.htm#benefits-of-fitness-apps

5. Електроний ресурс: https://www.pcmag.com/picks/best-workout-apps

6. Електроний ресурс: https://www.bswhealth.com/blog/how-to-pick-the-best-workout-app-for-your-fitness-goals

7. Сандро Б'янкіні. Порівняльне дослідження методів веб-розробки. Електронний ресурс: https://www.academia.edu/27333629/A_Comparison_Study_of_Web_Development_Methods

8. Атцені П., Мекка Г., Меріальдо П. (1998): Розробка та підтримка веб-сайтів з інтенсивним використанням даних. Proceedings of Extending Database Technology, Valencia: 436–450.

9. Kadlec V. (2004): Гнучке програмування: ефективні методи розробки програмного забезпечення. Комп’ютерна преса, Брно.

10. Зеленка П. (2004): Методи аналізу та проектування веб-додатків. Матеріали докторського семінару Факультет економіки та менеджменту Чеського сільськогосподарського університету, Прага.

11. Д. Хоукрофт, Дж. Керролл, Пропонована методологія веб-розробки. 297.

12. П. Девіс, Survival of thettest, The Computer Bulletin Series 5 2(частина 5) (2000) 28± 29

13. Tim O'Reilly What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. URL: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

14. Пелещишин А.М. Веб 2.0 та Семантичний Веб: порівняльний аналіз перспективних тенденцій розвитку WWW / А.М. Пелещишин, О.Л. Березко // Східно-Європейський журнал передових технологій. – Харків: Технологічний центр, 2006. – 6/2 (24) 2006. – С. 43–51.

15. Пелещишин А.М. Позиціонування сайтів у глобальному інформаційному середовищі: Монографія / А.М. Пелещишин. – Львів: Видавництво Національного університету "Львівська політехніка", 2007. – 260 с. 3. Webometrics. Ranking Web of World Universities [Electronic resource] / Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

16. CybermetricsLab, CCHS – CSIC // – Mode of access: WWW.URL: http://www.webometrics.info – Title from the screen. 4. Page L.S. The PageRank Citation Ranking: Bringing Order to the Web [Electronic resource] / L.S. Page, S. Brin, R. Motwani, T. Winograd // Technical report, Stanford Digital Library Technologies Project. – 1998. – Mode of access: WWW/URL: http://www-db.stanford.edu/~backrub/pageranksub. ps – Title from the screen.

ДОДАТОК А.

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Web-додаток підбору тренувань»

ПОГОДЖЕНО:

Доцент кафедри Інформаційних технологій

_____ Парфененко Ю.В.

Студент групи ІТз-91с

_____ Ігнатенко О.О.

Суми 2023

1. Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Web-додаток має надавати повноцінну інформацію клієнтам про спортивне тренування, можливість придбати та полегшити підбір тренування

1.2 Мета створення web-додатку

Збільшення кількості бажаючих розвивати себе за допомогою використання web-додатку.

1.3 Цільова аудиторія

До цільової аудиторії web-додатку можна віднести практично всіх людей, що зацікавлені у послугах тренера та інформації щодо тренувань. Завдяки широкому списку послуг, які надає web-додаток кожен зможе підібрати саме те, що шукає.

2 Вимоги до web-додатку

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток має бути доступним в мережі Інтернет. Web-додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями. Додаток буде мати такі сторінки з функціоналом: сторінка авторизації, сторінка реєстрації, головна сторінка, сторінка з безкоштовними програмами, сторінка з платними програмами, сторінка кошику, сторінка з підбором програм за введеними параметрами.

2.1.2 Вимоги до персоналу

Від персоналу не має вимагатися особливих технічних навичок для підтримки й експлуатації web-додатку, окрім загальних навичок роботи з персональним комп'ютером і стандартним веб-браузером.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у web-додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MongoDB.

2.1.4 Вимоги до розмежування доступу

Розроблюваний web-додаток має бути загальнодоступним.

Відповідно до прав доступу до інформації у web-додатку, усіх користувачів можна поділити на відвідувачів та адміністратора.

Відвідувачі можуть переглядати усі сторінки web-додатку, ознайомитись з видами послуг та замовити тренування.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

Структура web-додатку являє собою набір сторінок, які також є пунктами головного меню.

Такими розділами є:

Головна – на сторінці зображені головне меню та інформація про тренування з пропозиціями.

Програми – основна інформація про види тренувань та готові програми тренувань.

Тарифи – інформація та ціна індивідуальних програм тренувань

Контакти – список всіх можливих видів зв’язку з адміністратором сайту.

Кабінет – особистий кабінет користувача.

Кошик – інформація про обрану програму тренувань, оформленням послуги та розрахунок орієнтовної вартості замовлення.

2.2.2 Навігація

Відповідно до бажаного дизайну web-додатку, для навігації, у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх

сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку web-додатку.

2.2.3 Наповнення web-додатку (контент)

Заповнення та редагування контенту web-додатку має бути зроблено за допомогою бази даних.

2.2.4 Дизайн та структура додатку

Стиль web-додатку має бути сучасним, приємним для сприйняття, у якості основних кольорів буде використано червоні, білі та чорні відтінки, так як ці кольори, переважають в стилістиці спортклубів.

Основою мають бути фотографії гарної якості, web-додаток має бути інтуїтивно зрозумілим для використання.

Розташування елементів на головній сторінці web-додатку схематично показано на рисунку А.1.

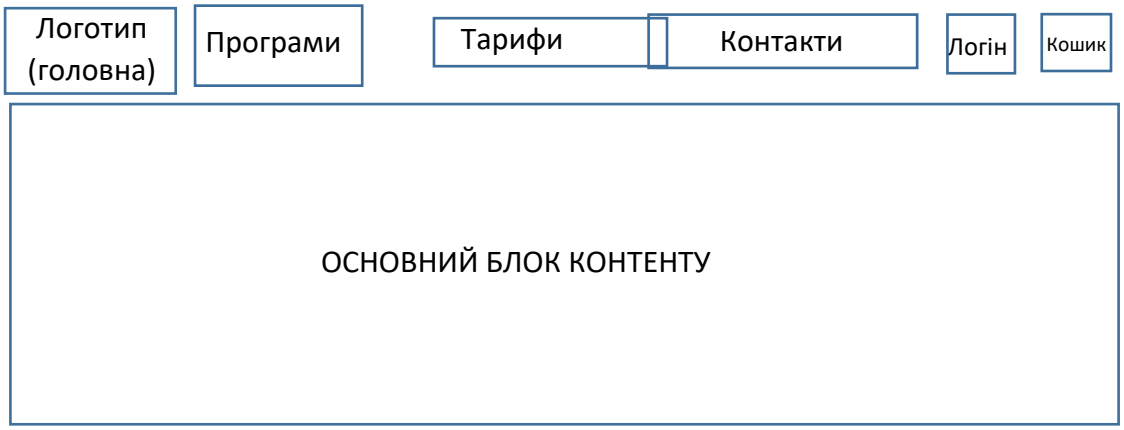

Рисунок А.1 – Схема головної сторінки

2.2.5 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.2.

Рисунок А.2 – Карта web-додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд головної сторінки сайту з корисною інформацією щодо спортивних тренувань	Клієнт
UN-02	Перегляд існуючих спортивних програм	Клієнт
UN-03	Перегляд та можливий вибір платних тренувань	Клієнт
UN-04	Можливість зворотного зв'язку	Клієнт
UN-05	Редагування даних	Адміністратор
UN-06	Можливість входу в особистий кабінет	Клієнт
UN-07	Розрахунок орієнтовної вартості замовлення	Клієнт
UN-08	Перегляд спортивних програм за введеними параметрами клієнта	Клієнт

2.3.2 Функціональні вимоги

Були визначені такі функціональні вимоги:

- реєстрація та авторизація користувачів;
- підбір програми тренування за введеними даними;
- перегляд тренувань;
- перегляд платних послуг;
- можливість оформлення замовлення через кошик;
- адміністрування інформації, зміна інформації в web-додатку, новини.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	Системні вимоги	Пріоритет	Опис
SR-01	Модуль авторизації	M	Надає можливість авторизації для клієнта
SR-02	Модуль реєстрації	M	Надає можливість реєстрації для клієнта
SR-03	Модуль підбору тренувань	M	Надає можливість клієнту обрати тим тренувань
SR-04	Модуль оформлення замовлення	M	Надає можливість оформити замовлення для клієнта

SR-05	Модуль бази даних	M	Надає можливість зберігати дані про тренування

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням:

- ReactJS
- Java Spring
- PostgreSQL

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам: веб-браузер: Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

3 Склад і зміст робіт зі створення web-додатку

Докладний опис етапів роботи зі створення web-додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей необхідних для досягнення певного результату	1 день
2	Складання технічного завдання	3 дні
3	Підготовка прототипу	2 дні
4	Створення макету дизайну web-додатку	3 дні
5	Верстка	3 дні
6	Робота над модулями для web-додатку	2 дні
7	Робота з контентом	2 день
8	Розміщення контенту та каталогів з фото у web-додатку	2 день
9	Перевірка працездатності web-додатку	1 день
10	Завершення роботи	1 день
	Загальна тривалість робіт	20 днів

4 Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Для того, щоб web-додатком могли користуватися клієнти необхідно розмістити його у мережі Інтернет, тому необхідно придбати доменне ім’я та місце на хостингу. На хостинг переноситься web-додаток і наповнення бази даних з подальшою їх доробкою. Для коректного переносу web-додатку на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ.

ДОДАТОК Б. ПЛАНУВАННЯ РОБІТ

Щоб проект був успішним та конкурентоспроможним треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. Абревіатура розшифровується наступним чином: Specific (конкретність та ясність), Measurable (вимірність), Achievable (досяжність), Relevant (узгодженість, важливість), Time-related (визначеність у часі). Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific	Скорочення часу користувача на прийнятті рішень щодо управління компонентами енергетичної системи, підвищення соціальної свідомості громадян.
Measurable	Стабільно працююча мікромережа, яка виробляє лише ту кількість енергії, яку може спожити.
Achievable	Дефіцит енергії зумовив перехід користувачів до ВДЕ, якими потрібно управляти.
Relevant	Для управління власною мікромережею з будь-якої точки планети та прискорення процесу прийняття рішень щодо її роботи.
Time-framed	Є конкретний термін – (13.05.2023)

При плануванні робіт необхідно враховувати наступні етапи:

Дослідження та аналіз вимог:

1. Вивчення поточних потреб користувачів у плануванні тренувань.
2. Аналіз існуючих веб-додатків та ресурсів у галузі планування тренувань.
3. Визначення функціональних і нефункціональних вимог для програми, що розробляється.

Проектування архітектури та інтерфейсу:

1. Розробка загальної структури та архітектури додатку.
2. Проектування інтерфейсу користувача, включаючи створення макетів екранів та взаємодії з користувачем.

Вибір технологій та інструментів:

1. Дослідження та вибір необхідних технологій для розробки програми (фронтенд, бекенд, база даних тощо).
2. Визначення інструментів розробки та середовища розробки.

Розробка та реалізація функціональності:

1. Створення базової структури проекту та налаштування середовища розробки.
2. Реалізація функцій планування тренувань, створення програм тренувань та відстеження прогресу.
3. Розробка функцій взаємодії з користувачем, включаючи реєстрацію, авторизацію та обмін даними.

Тестування та налагодження:

1. Проведення модульного тестування для кожної функціональності програми.
2. Виявлення та виправлення помилок та недоліків.
3. Проведення інтеграційного тестування для перевірки працездатності всієї програми.

Дизайн та інтерфейс користувача:

1. Реалізація дизайну інтерфейсу відповідно до розроблених макетів.
2. Оптимізація користувальницького досвіду та інтерфейсу програми.

Розвиток та підтримка:

1. Підготовка програми до розгортання на сервері або хмарній платформі.
2. Розробка механізмів оновлення та підтримки програми.
3. Планування та реалізація майбутніх покращень та додаткових функціональностей.

Оцінка та документація:

1. Проведення оцінки програми відповідно до заданих вимог.
2. Створення документації з розробки, встановлення, налаштування та використання додатка.

Тестування та збір зворотного зв'язку:

1. Проведення тестування програми за участю реальних користувачів.
2. Збір зворотного зв'язку та аналіз її результатів для виявлення потенційних покращень та коригування програми.

Запуск та розгортання:

1. Підготовка до запуску на сервері або хмарній платформі.
2. Розгортання та конфігурація програми для доступу користувачам.

Підтримка та оновлення:

1. Підтримка та оновлення програми після його запуску.
2. Реагування на зворотний зв'язок користувачів і вирішення проблем, що виникають.
3. Внесення покращень і додаткових функцій на основі зворотного зв'язку та потреб користувачів, що змінюються.

Оцінка та документація:

1. Оцінка ефективності програми та досягнення її цілей.

2. Створення заключної документації, що включає опис функціональності, посібника користувача та технічну документацію.

Етапи плану робіт для створення web-додатку планування тренувань є послідовними кроками, які дозволять розробити та впровадити повноцінний та функціональний додаток. План робіт може бути адаптований та доповнений залежно від конкретних вимог та особливостей проекту.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник		Створює шаблони додатку, виконує front-end та back-end розробку.
Проектувальник		Розробляє структуру web-додатку.
Тестувальник		Відповідають за тестування функціоналу та дизайну web-додатку.
Керівник проекту		Формує завдання на розробку проекту, проводить консультації.
Менеджер проекту		Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Відповідальний за супроводження проекту необхідною документацією.

Важливим етапом при плануванні проекту є управління ризиками. Ризик проекту – імовірнісна подія, яка у випадку своєї появи, позитивно або негативно вплине на хід проекту (хоча б один з показників проекту). Під час виконання якісної оцінки ризиків треба визначити ті з них, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику – реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконувати одночасно або окремо, що залежить від ступеня забезпечення

проекту. У таблиці Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Дуже низька Низька	Дуже малий Малий	Прийнятні
2	Середня	Середній	Виправдані
3	Висока Дуже висока	Великий Дуже великий	Неприпустимі

Таблиця Б.6 – Ризики проекту та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
1	Відкритий	Низька кваліфікація розробника для реалізації поставлених задач	Високий	Великий	3	Переглянути додаткову літературу, скористатися онлайн-ресурсами для підвищення знань	Ухилення	Видати допоміжну літературу. Провести консультацію.
2	Відкритий	Хвороба	Дуже високий	Середній	3	Оптимально розподіляти час, не перевантажувати організм та гарно спати. Додати вітаміни до раціону.	Зменшення	Одразу починати лікування, не запускати хворобу. Використати резерви часу, які було залишено для надзвичайних ситуацій.
3	Відкритий	Недотримання календарного плану	Дуже низький	Середній	1	Заздалегідь прорахувати усі можливі затримки та випередження, зробити резерв часу для складних робіт	Зменшення	Переоцінка тривалості виконання кожної роботи, внесення правок в календарний план.
4	Відкритий	Часте внесення змін у ТЗ	Дуже низький	Великий	1	1.Виділити всі необхідні параметри проекту. 2.Чітко описати вимоги до проекту. 3.Обговорити всі технічні засоби виконання проекту та умови реалізації.	Зменшення	Узгодити всі положення з замовником, у разі потреби вносити необхідні зміни та поправки.

5	Відкритий	Вибір неефективної технології розробки	Низький	Великий	2	Провести детальний аналіз існуючих технологій, опираючись на предметну область та специфіку проекту. Порадитися з керівниками. Під час порівняння аналогів звернути увагу, які технології використовувалися під час їх розробки.	Ухилення	Змінювати обрану технологію, детально ознайомившись зі специфікою проекту. Узгодити обрану технологію з керівником.
6	Відкритий	Помилки в документації	Середній	Середній	2	Ознайомитися з прикладами документації перш ніж приступати до самостійного складання.	Ухилення	Детальна перевірка усієї документації декількома людьми.
7	Відкритий	Виникнення непланових робіт	Високий	Середній	2	1.Детальне складання ТЗ 2.Обговорення цілей і задач з керівником	Ухилення	Перепланування календарного плану робіт, внесення правок в ТЗ.
8	Відкритий	Поломка обладнання розробника	Низький	Малий	1	1.Перевірити стан комп'ютера перед початком виконання проекту. 2.Захистити комп'ютер від вірусів та можливих загроз.	Ухилення	Замінити обладнання, залучити спеціаліста для усунення збоїв. Знайти заздалегідь нове робоче місце на час ремонту обладнання
9	Відкритий	Збій програмного забезпечення	Низький	Великий	2	1.Підготувати резерв програмних засобів. 2.Залучити спеціаліста для усунення збоїв.	Зменшення	Замінити програмне забезпечення.

10	Відкритий	Відсутність резервних копій даних	Дуже низький	Дуже великий	3	1.Налаштувати автоматичне збереження даних. 2.Зберігати дані на різних носіях інформації.	Ухилення	Робити копію даних після кожного виконаного етапу.

Для того, щоб знизити негативний вплив ризиків на проект треба виконати планування реагування на них. Планування реагування на ризики – це процес розробки шляхів і визначення дій по збільшенню можливостей і зниженню загрози для мети проекту. Оцінювання наслідків виконується за показниками, що описані в таблиці Б.3. У результаті планування реагування було отримано матрицю ймовірності виникнення ризиків та впливу ризику, результати занесено у таблицю Б.4. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.4 – Матриця ймовірності та впливу згідно проекту

Ймовірність	Вплив загрози(ризику)				
	Дуже малий 0,05	Малий 0,1	Середній 0,2	Великий 0,4	Дуже великий 0,8
0,9			R2(0,18)		
0,7			R7(0,14)	R1(0,28)	
0,5			R6(0,1)		
0,3		R8(0,03)		R5 (0,12), R9(0,12)	
0,1			R3(0,02)	R4(0,04)	R10(0,08)

Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.5. У таблиці Б.6 описано ризики та стратегії реагування на кожен з них.

Таблиця Б.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризик, які входять (номера)
1	Прийнятні	$0,005 \leq R \leq 0,05$	3,4,8
2	Виправдані	$0,05 < R \leq 0,14$	5,6,7,9
3	Недопустимі	$0,14 < R \leq 0,72$	1, 2, 10

ДОДАТОК В.

```
@Component
public class TypeEnumConverter implements Converter<String, Type> {
  @Override
  public Type convert(String value) {
    return Type.create(value);
  }
}

/** Exception handler class for filters. */
@Component
public class FilterChainExceptionHandler extends OncePerRequestFilter {

  @Autowired
  @Qualifier("handlerExceptionResolver")
  private HandlerExceptionResolver resolver;

  /** Method to resolve exception from filters using existing handler. */
  @Override
  protected void doFilterInternal(
      HttpServletRequest request, HttpServletResponse response, FilterChain
filterChain)
      throws ServletException, IOException {

    try {
      filterChain.doFilter(request, response);
    } catch (AbstractSportShopException e) {
      resolver.resolveException(request, response, null, e);
    }
  }
```

```
@RequiredArgsConstructor
public class JwtAuthorizationFilter extends OncePerRequestFilter {
  private static final String HEADER = "Authorization";
  private static final String PREFIX = "Bearer ";
  private final UserService userService;
  private final JwtService jwtService;
  @Override
  protected void doFilterInternal(
      HttpServletRequest request, HttpServletResponse response, FilterChain chain)
      throws ServletException, IOException {
    getAndAuthenticateUserFromJwt(getAndValidateJwtToken(request));
    chain.doFilter(request, response);
  }

  private String getAndValidateJwtToken(HttpServletRequest request) {
    String authenticationHeader = request.getHeader(HEADER);
    if (authenticationHeader == null || !authenticationHeader.startsWith(PREFIX)) {
      throw new JwtAuthenticationException(
          Map.of("token", "Authorization jwt token required"), "Jwt error");
    }

    return authenticationHeader.replace(PREFIX, "");
  }

  private void getAndAuthenticateUserFromJwt(String jwt) {
    try {
      SecurityUserModel securityUserModel =
```

```
        userService.getUserByEmail(jwtService.getEmailFromJwt(jwt));
      UsernamePasswordAuthenticationToken auth =
          new UsernamePasswordAuthenticationToken(
              securityUserModel, "", securityUserModel.getAuthorities());


      SecurityContextHolder.getContext().setAuthentication(auth);
    } catch (IncorrectCredentialsException e) {
      throw new JwtAuthenticationException(Map.of("token", "Jwt verification error"), "Jwt
error");
    }
  }
}


@Configuration
public class PasswordEncodingConfig {


  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }
```

```
@Configuration
@RequiredArgsConstructor
public class WebSecurityConfig {
  private final UserService userService;
  private final JwtService jwtService;
  private final FilterChainExceptionHandler filterChainExceptionHandler;

public class TrainingEntity {
  @Id
  @GeneratedValue(generator = "uuid")
  @GenericGenerator(name = "uuid", strategy = "uuid2")
  @Column(name = "training_id")
  private String id;

  @Column(name = "creator_id")
  private String creatorId;

  @ManyToOne(fetch = FetchType.LAZY, targetEntity = UserEntity.class)
  @JoinColumn(name = "creator_id", insertable = false, updatable = false)
  private UserEntity creator;

  @Column(name = "name")
  private String name;

  @Column(name = "description")
  private String description;
```

```
@Column(name = "text", length = 10485760)
private String text;

@Column(name = "image")
private String image;

@Column(name = "price")
private Integer price;

@Column(name = "paid")
private Boolean paid;

@Column(name = "gender")
@Enumerated(EnumType.ORDINAL)
private Gender gender;

@Column(name = "type")
@Enumerated(EnumType.ORDINAL)
private Type type;
```

```
public class UserEntity {
  @Id
  @GeneratedValue(generator = "uuid")
  @GenericGenerator(name = "uuid", strategy = "uuid2")
  @Column(name = "user_id")
  private String id;

  @Column(name = "email", length = 32, unique = true)
  private String email;

  @Column(name = "username", length = 32)
  private String username;

  @Column(name = "password")
  private String password;

  @ToString.Exclude
  @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"creatorId")
  private List<TrainingEntity> trainings;

  @Column(name = "roles")
  @Enumerated(EnumType.ORDINAL)
  private Roles role;

  @Column(name = "created_on")
  private LocalDateTime createdOn;
```

```
public class TrainingCustomRepositoryImpl implements TrainingCustomRepository {

  private final EntityManager entityManager;


  @Override

  public List<TrainingEntity> findAllTrainingsWithFilters(TrainingFilterModel filterModel)
{

    CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();

    CriteriaQuery<TrainingEntity> select =
criteriaBuilder.createQuery(TrainingEntity.class);

    Root<TrainingEntity> catalogEntityRoot = select.from(TrainingEntity.class);

    select.select(catalogEntityRoot);


    Predicate paidPredicate =

        filterModel.getPaid() != null

            ? criteriaBuilder.equal(catalogEntityRoot.get("paid"), filterModel.getPaid())

            : null;

    Predicate typePredicate =

        filterModel.getType() != null

            ? criteriaBuilder.equal(catalogEntityRoot.get("type"), filterModel.getType())

            : null;
```

```
@Service
public class JwtServiceImpl implements JwtService {
  private final byte[] secretKey =
      System.getenv("SportShopJwtSecret").getBytes(StandardCharsets.UTF_8);
  private final ConfigurableJWTProcessor<SimpleSecurityContext> jwtProcessor =
      new DefaultJWTProcessor<>();
  @PostConstruct
  public void init() {
    JWKSource<SimpleSecurityContext> jweKeySource = new
ImmutableSecret<>(secretKey);
    JWEKeySelector<SimpleSecurityContext> jweKeySelector =
        new JWEDecryptionKeySelector<>(
            JWEAlgorithm.DIR, EncryptionMethod.A128CBC_HS256, jweKeySource);
    jwtProcessor.setJWEKeySelector(jweKeySelector);
  }
  @Override
  public String createJwt(String email) {
    try {
      Date now = new Date();
      Date expDate = new Date(now.getTime() + 1000 * 60 * 60 * 24 * 10); // expires in 10
days
      JWTClaimsSet claims =
          new JWTClaimsSet.Builder()
              .claim("email", email)
              .expirationTime(expDate)
              .notBeforeTime(now)
              .build();
      Payload payload = new Payload(claims.toJSONObject());
```

```
public class TrainingServiceImpl implements TrainingService {
  private final FirebaseFileService firebaseFileService;
  private final TrainingRepository trainingRepository;
  private final TrainingCustomRepository trainingCustomRepository;
  private final UserService userService;

  @Override
  public List<TrainingModel> getAllTrainingsWithFilters(TrainingFilterModel
trainingFilterModel) {
    return ServiceLayerMapper.I.trainingEntitiesToModel(
        trainingCustomRepository.findAllTrainingsWithFilters(trainingFilterModel));
  }

  @Override
  public TrainingModel getTrainingById(String id) {
    return ServiceLayerMapper.I.trainingEntityToModel(
        trainingRepository
            .findById(id)
            .orElseThrow(
                () ->
                    new DataNotFoundException(
                        Map.of("trainingId", "There are no training with such id"),
                        "Training not found")));
  }
```

```
public class UserServiceImpl implements UserService {

  private final UserRepository userRepository;
  private final PasswordEncoder passwordEncoder;

  @PostConstruct
  private void init() {
    UserEntity testEntity =
        UserEntity.builder()
            .email("test@test")
            .password(passwordEncoder.encode("test"))
            .username("test")
            .createdOn(LocalDateTime.now())
            .role(Roles.ROLE_TRAINER)
            .build();
    if (userRepository.getUserEntityByEmail("test@test").isEmpty()) {
      userRepository.save(testEntity);
    }
  }

  @Override
  public UserModel updateUser(UserModel userModel) {
    return ServiceLayerMapper.I.userEntityToModel(
        userRepository.save(
            ServiceLayerMapper.I.updateUserEntityFromModel(
                userModel,
userRepository.findById(getAuthenticatedUser().getId()).orElseThrow())));
  }
```

```
public class AccountsController {
private final UserService userService;
private final JwtService jwtService;

@PostMapping("/register")
public ResponseEntity<AuthorizationTokenResponseDto> registerAccount(
    @RequestBody @Valid AccountCredentialsRequestDto requestDto) {

userService.registerUser(WebLayerMapper.I.accountCredentialsRequestDtoToUser(request
Dto));
  return ResponseEntity.status(HttpStatus.CREATED)
      .body(
          AuthorizationTokenResponseDto.builder()
              .authorizationToken(jwtService.createJwt(requestDto.getEmail()))
              .build());
}
@PostMapping("/login")
public ResponseEntity<AuthorizationTokenResponseDto> loginInAccount(
    @RequestBody @Valid AccountLoginRequestDto requestDto) {

userService.isCorrectPassword(WebLayerMapper.I.accountLoginRequestDtoToUser(reques
tDto));

  return ResponseEntity.ok(
      AuthorizationTokenResponseDto.builder()
          .authorizationToken(jwtService.createJwt(requestDto.getEmail()))
          .build());
}
```

```
}

public class TrainingController {
  private final TrainingService trainingService;

  @Operation(security = @SecurityRequirement(name = "Authorization"))
  @PostMapping(consumes = {MediaType.MULTIPART_FORM_DATA_VALUE})
  public ResponseEntity<TrainingResponseDto> create(
      @RequestPart(name = "file") MultipartFile file,
      @ModelAttribute @ParameterObject CreateTrainingRequestDto requestDto) {
    return ResponseEntity.status(HttpStatus.CREATED)
        .body(
            WebLayerMapper.I.trainingModelToResponseDto(
                trainingService.saveTraining(
                    file,
WebLayerMapper.I.createTrainingRequestDtoDtoToModel(requestDto))));
  }

public class UserController {
  private UserService userService;
  private JwtService jwtService;

  @GetMapping
  @Operation(security = @SecurityRequirement(name = "Authorization"))
  public ResponseEntity<UserResponseDto> getUser() {
    return ResponseEntity.ok(
        WebLayerMapper.I.userModelToResponseDto(userService.getAuthenticatedUser()));
  }
```

```
  @PatchMapping("/data")
  @Operation(security = @SecurityRequirement(name = "Authorization"))
  public ResponseEntity<AuthorizationTokenResponseDto> changeUserData(
      @Valid @RequestBody UserDataRequestDto requestDto) {

userService.updateUser(WebLayerMapper.I.userDataRequestDtoToUserModel(requestDto)
);
    return ResponseEntity.ok(
        AuthorizationTokenResponseDto.builder()
            .authorizationToken(
                jwtService.createJwt(
                    (requestDto.getEmail() == null)
                        ? userService.getAuthenticatedUser().getEmail()
                        : requestDto.getEmail()))
            .build());
  }

  @PatchMapping("/password")
  @Operation(security = @SecurityRequirement(name = "Authorization"))
  public ResponseEntity<HttpStatus> changeUserPassword(
      @Valid @RequestBody AccountChangePasswordRequestDto requestDto) {
    userService.changeUserPasswordByEmail(
        userService.getAuthenticatedUser().getEmail(), requestDto.getNewPassword());
    return ResponseEntity.ok().build();
  }
}
```

```
const Footer = () => {
  return (
    <footer className="footer">
      <img src="/images/sportshopSport.svg" alt="logo" />
      <p>Copyright © 2023 SportShop,Inc</p>
    </footer>
  );
};

export default Footer;

const Header = () => {
    const [userData, setUserData] = useState({});
    let navigate = useNavigate();

    const token = localStorage.getItem("token");

    const routeChange = (path) => {
        navigate(path);
    };

    const fetchUserInfo = () => {
        fetch("https://sport-shop-back.onrender.com/a/rest/user", {
            headers: {
                Accept: "application/json",
                "Content-Type": "application/json",
                Authorization: `Bearer ${token}`,
```

```
        },
        method: "GET",
    })
        .then((res) => res.json())
        .then((res) => {
            setUserData(res);
        });
};
```

```
                <Link
                    style={{
                        color: "white",
                        textDecoration: "none",
                        marginRight: "25px",
                    }}
                    to="/paid-trainings"
                >
                    Платні тренування
                </Link>
                <Link
                    style={{
                        color: "white",
                        textDecoration: "none",
                        marginRight: "25px",
                    }}
                    to="/choose-trainings"
                >
                    Підібрати тренування
```

```
        </Link>
        {userData?.role === "ROLE_TRAINER" && (
          <Link
            style={{
              color: "white",
              textDecoration: "none",
              marginRight: "25px",
            }}
            to="/trainings/create"
          >
            Створити тренування
          </Link>
        )}
      </div>
    </div>
    <div style={{ display: "flex", alignItems: "center" }}>
      {token && (
        <div
          onClick={handleProfileRedirect}
          style={{
            cursor: "pointer",
            display: "flex",
            alignItems: "center",
            marginRight: "25px",
          }}
        >
          <AccountCircleIcon fontSize="large" />
```

```
            <p style={{ marginLeft: "5px" }}>Мій профіль</p>
        </div>
    )}

    <Link to="/cart">
        <img
            src="/images/shopping-cart.svg"
            alt="logo"
            style={{
                display: token ? "flex" : "none",
                cursor: "pointer",
                marginLeft: "10px",
                width: "32px",
            }}
        />
    </Link>
    <img
        src="/images/sign-out.svg"
        alt="logo"
        onClick={handleLogOut}
        style={{
            display: token ? "flex" : "none",
            cursor: "pointer",
            marginLeft: "10px",
        }}
    />
</div>
```

```
<div style={{ display: !token ? "flex" : "none" }}>
  <Button
    onClick={handleRegisterRedirect}
    style={{
      fontSize: 16,
      textTransform: "none",
      borderRadius: 15,
      padding: "",
    }}
    className="header-sign-up-button"
    variant="contained"
  >
    Зареєструватися
  </Button>
  <Button
    onClick={handleLoginRedirect}
    style={{
      fontSize: 16,
      textTransform: "none",
      borderRadius: 15,
    }}
    className="header-sign-in-button"
    variant="contained"
  >
    Авторизуватися
  </Button>
```

```
            </div>
          </div>
        </header>
    );
};

export default Header;
```

```
export default function AccountProfilePage() {
    const [userInfo, setUserInfo] = useState({
        email: "",
        username: "",
    });

    const [dataFormInfo, setDataFormInfo] = useState({
        email: "",
        username: "",
        password: "",
        confirmPassword: "",
    });

    const [errors, setErrors] = useState({
        email: "",
        username: "",
        password: "",
        passwordMismatch: "",
    });

    const [openDataModal, setOpenDataModal] = useState(false);

    const [openPasswordModal, setOpenPasswordModal] = useState(false);

    let navigate = useNavigate();
    const token = localStorage.getItem("token");
```

```
const handleLoginRedirect = () => routeChange("/login");


const fetchUserInfo = () => {
    fetch("https://sport-shop-back.onrender.com/a/rest/user", {
        headers: {
            Accept: "application/json",
            "Content-Type": "application/json",
            Authorization: `Bearer ${token}`,
        },
        method: "GET",
    })
        .then((res) => res.json())
        .then((res) => {
            if (res.errors) {
                for (const [key, value] of Object.entries(res.errors)) {
                    setError(`${key}`, value, value);
                }
            }
            setUserInfo(() => ({
                email: res.email,
                username: res.username,
            }));
            setDataFormInfo(() => ({
                email: res.email,
                username: res.username,
            }));
        });
```

```
};

const changeUserData = () => {
    if (validateDataFields()) {
        return;
    }
    const body = {};
    if (dataFormInfo.email !== userInfo.email) {
        body.email = dataFormInfo.email;
    }
    if (dataFormInfo.username !== userInfo.username) {
        body.username = dataFormInfo.username;
    }
    fetch("https://sport-shop-back.onrender.com/a/rest/user/data", {
        headers: {
            Accept: "application/json",
            "Content-Type": "application/json",
            Authorization: `Bearer ${token}`,
        },
        method: "PATCH",
        body: JSON.stringify(body),
    })
        .then((res) => res.json())
        .then((res) => {
            if (!res.authorizationToken) {
                for (const [key, value] of Object.entries(res.errors)) {
                    setError(`${key}`, value, value);
```

```
            }
    return (
        <div className="cart">
            <h1
                style={{
                    textAlign: "center",
                    fontWeight: "700",
                    fontSize: "28px",
                    paddingTop: "50px",
                    marginBottom: "25px",
                }}
            >
                Корзина
            </h1>
            {cartItems?.length > 0 ? (
                <>
                    <div
                        style={{
                            display: "flex",
                            flexDirection: "column",
                        }}
                    >
                        {cartItems?.map((el) => (
                            <CartItem
                                key={el.id}
                                cart={cartItems}
                                setCartItems={setCartItems}
```

```
            {...el}
          />
        ))}
      </div>
      <div
        style={{
          margin: "30px 0 50px 50px",
          display: "flex",
          alignItems: "center",
        }}
      >
        <p
          style={{
            fontSize: "26px",
            fontWeight: "600",
            marginRight: "20px",
          }}
        >
          Загалом:{" "}
          {cartItems
            ?.filter((el) => el.paid)
            .reduce((sum, el) => {
              return (el.price += sum);
            }, 0)}
          $
        </p>
        <Button
```

```
                    size="small"
                    sx={{ height: "40px" }}
                    onClick={() => {
                      localStorage.removeItem("cart");
                      routeChange("/cart-success");
                    }}
                    variant="contained"
                  >
                    Замовити
                  </Button>
                </div>
              </>
            ) : (
              <h2
                style={{
                  textAlign: "center",
                  fontWeight: "700",
                  fontSize: "24px",
                  paddingTop: "300px",
                }}
              >
                В корзині пусто!
              </h2>
            )}
          </div>
    );
};
```

```
const CartSuccess = () => {
    return (
        <div className="cart-success">
            <h1
                style={{
                    textAlign: "center",
                    fontWeight: "700",
                    fontSize: "28px",
                    paddingTop: "350px",
                    marginBottom: "25px",
                }}
            >
                Дякуємо за замовлення!
            </h1>
            <p
                style={{
                    textAlign: "center",
                    fontWeight: "600",
                    fontSize: "24px",
                }}
            >
                Наш менеджер скоро зв'яжеться з вами.
            </p>
        </div>
    );
};
```

```
export default CartSuccess;
export default function App() {
  return (
    <div>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<MainPage />} />
          <Route path="login" element={<SignInPage />} />
          <Route path="register" element={<SignUpPage />} />
          <Route path="trainings" element={<FreeTrainings />} />
          <Route path="paid-trainings" element={<PaidTrainings />} />
          <Route
            path="choose-trainings"
            element={<ChooseTrainings />}
          />
          <Route
            path="trainings/:trainingId"
            element={<TrainingsDetails />}
          />
          <Route path="cart" element={<Cart />} />
          <Route path="cart-success" element={<CartSuccess />} />
          <Route path="profile" element={<AccountProfilePage />} />
          <Route path="*" element={<NoMatch />} />
        </Route>
      </Routes>
    </div>
```