КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок із підтримки фітнес-тренувань»

Виконала _____
(Підпис)
Борсук Олександра Сергіївна
(прізвище, ім'я, по батькові)

Керівник _к.т.н., доц. Міронова В.Л.
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок із підтримки фітнес-тренувань»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Борсук Олександра Сергіївна	

Назва роботи українською та англійською мовами

Веб-застосунок із підтримки фітнес-тренувань
Fitness training web application

Мета бакалаврської кваліфікаційної роботи, завдання

Мета бакалаврської роботи: розробка веб-застосунку для заохочення та підтримки користувачів при виконанні фізичних навантажень.

План роботи:

1. Сучасні підходи розробки веб-застосунків та їх подальшого впровадження
2. Аналіз архітектурних підходів та вибір стеку технологій для розробки веб-застосунку
3. Програмна реалізація веб-застосунок із підтримки фітнес-тренувань

Міронова В.Л., доц., к.т.н. : _____

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	12
2	13
3	11
Висновки	2
Перелік використаних джерел	4
Додатки	6

				ДП XXXX 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.						
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ (РЕФЕРАТ)

Дипломна робота: 59 с., 16 рис., 36 джерел, 4 дод.

Метою дипломної роботи є розробка веб-застосунку для заохочення та підтримки користувачів при виконанні фізичних навантажень.

Для досягнення мети роботи потрібно вирішити такі **завдання**:

– дослідити теоретичні основи побудови та варіативність архітектури веб-застосунків;

– дослідити сучасні стеки технологій для розробки програмних продуктів та обґрунтувати вибір стеку;

– проаналізувати наявні сучасні програмні продукти сфери охорони здоров’я та спорту для виявлення переваг та недоліків, що складатимуть вимоги до реалізації;

– спроектувати та розробити веб-застосунок.

Об’єкт дослідження.

Популяризація та заохочення користувачів до ведення здорового способу життя шляхом використання веб-застосунку для фітнес-тренувань як інструменту формування комплексу вправ.

Предмет дослідження.

Програмно-технічні засади та концепції розробки веб-застосунків, зокрема програмних продуктів для фітнес-тренувань.

Методи дослідження.

Дослідження та порівняльний аналіз наявних програмно-технічних рішень сфери охорони здоров’я та спорту. Системний аналіз та опис підходів побудови архітектури та розробки веб-застосунків. Наукове моделювання при побудові кастомних діаграм розроблювального програмного продукту. Розробка односторінкового інтерактивного застосунку (SPA) за обраним стеком технологій MERN.

Ключові слова: веб-застосунок, фітнес-тренування, SPA, MERN

ANOTATION (ABSTRACT)

Thesis: 59 pp., 16 figs., 36 sources, 4 appendix.

The aim of the thesis is to create an effective and interactive web application to support fitness training.

To achieve the goal of the work you need to solve the following **tasks**:

- to explore the theoretical foundations of the construction and variability of the architecture of web applications;
- explore modern technology stacks for software development and justify the choice of stack;
- to analyze the available modern software products in the field of health and sports to identify the advantages and disadvantages that will make the requirements for implementation;
- design and develop a web application.

Object of study.

Promoting and encouraging users to lead a healthy lifestyle by using a web application for fitness training as a tool for forming a set of exercises.

Subject of study.

Software and technical principles and concepts for developing web applications, including software products for fitness training.

Research methods.

Research and comparative analysis of existing software and hardware solutions in the field of health and sports. System analysis and description of approaches to building architecture and developing web applications. Scientific modeling in the construction of custom diagrams of software development. Development of a one-page interactive application (SPA) for the selected MERN technology stack.

Keywords: web application, fitness training, SPA, MERN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

англ. – англійська

БД – база даних

ВООЗ – Всесвітня організація охорони здоров'я – спеціалізована установа Організації об'єднаних націй, яка опікується проблемами охорони здоров'я у світовому масштабі.

ІС – інформаційна система

Лендінг – сайт-вітрина, веб-сторінка, яка відкривається при натисканні на рекламне оголошення чи ланку (лінк).

СД – сховище даних

СНД – Співдружність Незалежних Держав – регіональна міжнародна організація, до якої входить низка пострадянських країн.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду.

Фронтенд – (від анг. Front-end) це інтерфейс для взаємодії між користувачем.

API – набір чітко визначених методів для взаємодії різних компонентів

CRUD – 4 основні функції управління даними «створення, читання, оновлення і вилучення»

ВСТУП

Здоров’я нації є одним з ключових напрямів розвитку діяльності кожної країни, що посилився у зв’язку з нинішньої пандемією. Безумовно, на здоров’я кожного окремого громадянина впливає широкий спектр факторів, проте, кожен має можливості для самостійного регулювання, підтримки та профілактики найбільш поширених проблем. Головним профілактичним засобом визначено активний спосіб життя та заняття спортом.

Актуальність дослідження. Сучасний темп життя вимагає гнучкості та вмінню виконувати персональні задачі швидко, якісно та шляхом використання ІС. Тому поняття фізичних навантажень «мігрувало» у програмні продукти та застосунки, що пропонують різні види та способи фізичної діяльності. Розробка веб-застосунку з підтримки різнотипових фітнес-тренувань є доцільною, оскільки перевагою таких продуктів є кросплатформність та доступність з будь-якого гаджету.

Мета дослідження. Розробка веб-застосунку для заохочення та підтримки користувачів при виконанні фізичних навантажень.

Завдання дослідження. Обґрунтування та досягнення мети бакалаврської роботи передбачає визначення та вирішення наступних завдань:

- дослідити теоретичні основи побудови програмних застосунків:
  - розглянути наявні програмні рішення та визначити переваги й недоліки;
  - сформулювати технічне завдання на розробку веб-застосунку;
- здійснити аналіз програмно-технологічного інструментарію для розробки програмних продуктів у сфері веб-програмування;
- здійснити проектування, реалізацію, впровадження застосунку з підтримки фітнес-тренувань;
- описати схему роботи користувацького інтерфейсу.

Об’єкт дослідження. Популяризація та заохочення користувачів до ведення здорового способу життя шляхом використання веб-застосунку для фітнес-тренувань як інструменту формування комплексу вправ.

Предмет дослідження. Програмні засади, принципи побудови веб-застосунку для підтримки фітнес-тренувань.

Методи дослідження. Для розкриття теми та поставлених задач бакалаврської роботи, а також для формування звіту про розробку програмного продукту було використано такі методи:

- метод аналізу застосовано при досліджені наявних програмних рішень у сфері охорони здоров’я та спорту;
- описовий метод використано при формуванні алгоритму розробки та кроків моделювання курсового проекту, описі основних інструментів й технологій розробки, а також при огляді реалізованих можливостей застосунку;
- метод наукового моделювання використано при формуванні діаграм, моделей та інших компонентів, що відображають необхідні структурні елементи бакалаврської роботи та іншого допоміжного інструментарію;
- метод порівняння залучено при роботі наявними програмними рішеннями для визначення переваг та недоліків існуючих програмних систем;
- метод синтезу при вивчені сучасних програмно-технологічних рішень, стеків технологій, популярних фреймворків та бібліотек для вибору найбільш оптимального і доцільного набору розробника веб-застосунку.

Практичне значення одержаних результатів. Розвиток та заохочення громадянського суспільства до занять спортом, навіть в невеликих масштабах, на основі впровадження та використання веб-застосунку для підтримки фітнес-тренувань з сформованими наборами фізичних вправ різного типу та складності.

РОЗДІЛ 1. ПРОГРАМНИЙ ПРОДУКТ В СФЕРІ ЗДОРОВ'Я ТА СПОРТУ

1.1 Проблеми охорони здоров'я, причини виникнення та впровадження профілактичних заходів з допомогою ІТ-сервісів

Проблема людського здоров'я ніколи не втратить актуальності та прагнення до покращення цієї складової комфортного та щасливого життя кожної особи. Щороку провідні країни та міжнародні організації витрачають й інвестують вагомі грошові обсяги в розвиток та дослідження у сфері охорони здоров'я. Рівень витрат, які країна може собі дозволити для фінансування цієї сфери, залежить від інтегрального показника індивідуальних витрат окремо взятого громадянина цієї країни та видатків, сплачених населенням цієї країни, загалом. Кожного року цей показник перераховується, адже на нього впливає широке коло демографічних, соціальних та економічних факторів. Також Європейське регіональне бюро (ЄРБ) ВООЗ додає наступні чинники, що визначають сучасні тренди та тенденції для охорони здоров'я:

- глобалізація;
- вагомі демографічні зміни;
- екологічні проблеми;
- міграція населення;
- соціально-політичні конфлікти;
- нерівність розподілу матеріальних благ для населення.

З початку 2020 року до цього переліку додалась світова пандемія коронавірусу. Ця криза вже більше року викриває усі труднощі медичних систем, які ігнорувались раніше.

Системи умовно поділені на три групи : з переважно державним фінансуванням (країни СНД, скандинавські країни), страхова медицина (США, ОАЕ тощо) та приватна система охорони здоров'я. Проте, жодна з систем не змогла продемонструвати реальні переваги при боротьбі з епідемією: ані

державні, ані приватні медичні заклади, на жаль, не в змозі створити якісний алгоритм лікування пацієнтів в умовах поширення вірусу.

За даними ВООЗ, у період з березня по червень 2020 року 90% країн мала серйозні проблеми з надання медичних послуг.[1] Для боротьби з коронавірусом лікарі були змушені обирати які медичні послуги і кому саме відкласти, аби надати пріоритет хворим COVID-19. Багатьом хворим на хронічні, неінфекційні захворювання відтермінували необхідне лікування. Незважаючи на те, що вірусні та інфекційні захворювання, які класифікують як «гострі захворювання», є більш агресивними і швидше розвиваються, проблема неінфекційних захворювань знаходиться в центрі уваги лікарів усього світу протягом останнього десятиліття.

Для України цей аспект також має першочергове значення, бо, на жаль, в нашій країні досить висока частота поширеності неінфекційних хвороб й інвалідизації та смертності внаслідок цих хвороб.[2] В основі розвитку таких захворювань лежать однакові фактори ризику, до яких належать:

- малорухомий спосіб життя;
- куріння;
- надмірна вага;
- нераціональна харчова поведінка.

Усі описані фактори можливо коригувати і усувати на ранніх етапах розвитку за допомогою профілактичних заходів, що базуються на формуванні в населення медичних знань й установок на здоровий спосіб життя.

За для зниження рівня розвитку неінфекційних захворювань варто розширювати популяційну стратегію – впливати з допомогою сучасних технологій та застосунків на ті особливості способу життя і навколишнього середовища, які збільшують ризик розвитку хвороб серед усього населення, а саме – малорухомий спосіб життя. Саме ця проблема набула найвищого рівня актуальності, одразу після коронавірусу, в останні рік-півтора. Всесвітній карантин змусив адаптуватись до нових реалій дистанційної роботи та навчання,

більше часу проводити у власних домівках і, як результат, спричинив тотальний «сидячий» спосіб життя.

Дослідження Кардіфського університету показали, що 39% робітників, що працюють віддалено, працюють довше, ніж співробітники з офісу.[4] Влітку 2020 року Форбс оприлюднив матеріали власних досліджень про віддалений спосіб роботи та визначив ключові недоліки. Більше 40% опитаних людей заявило, що стан їх здоров'я, в тому числі і психологічний, значно погіршився після початку пандемії.[5]

Малорухливий спосіб життя провокує достатньо велику кількість хвороб, що можуть виснажувати та шкодити організму, різко зростає ризик розвитку захворювань серцево-судинної, дихальної, травної та сечостатевої систем. [3] Також, це гарантований спосіб постарішати на 10 років раніше, ніж більш рухливі ровесники. Такі приголомшливі результати були отримані фахівцями з лондонського Королівського коледжу у ході проведення досліджень, в якому брали участь 2400 пар близнюків. Під час дослідження вчені зробили опитування учасників, де враховували частоту та інтенсивність їхніх фізичних навантажень і виміряли довжину теломер (кінцевих фрагментів хромосом, що не несуть спадкової інформації, захищають ДНК клітини від пошкоджень і деформацій та є своєрідними індикаторами швидкості старіння організму) в клітинах їх крові. Зіставлення отриманих результатів дозволило дослідникам зробити висновок, що однакову довжину теломер мали фізично найбільш активні учасники та ті, хто дотримувався переважно сидячого способу життя, але був на 10 років молодший за перших.

Необхідний критичний мінімум щоденної рухової активності – 30-хвилинна прогулянка, пробіжка чи будь-яке інше фізичне навантаження. Це допоможе тримати організм в тонусі, уникнути передчасного старіння та низки важких хронічних захворювань. Карантин, безумовно, дещо коригує можливості для активностей: заборона виходити на вулицю без нагальної потреби, закриття спортзалів та комплексів, заборона масових заходів, що включає обмеження по кількості осіб на одній певній території. За таких умов на допомогу людям

приходять технології та застосунки, які значно спрощують, або, навіть, цілком замінюють заняття спортом офлайн.

1.2 Огляд найпоширеніших застосунків та онлайн-систем для тренувань

Мобільні застосунки, сайти, стрім-сервіси та системи стали корисними інструментам в насущній проблемі карантинного життя та діджиталізації всіх сфер людського життя в цілому. Сфера здоров'я та спорту не є виключенням.

Розглянемо світову тенденцію станом на грудень 2020 року на прикладі однієї з груп програмних продуктів – мобільні застосунки. Варто зазначити, що статистичні показники враховують органічність та органічність росту попиту на програмні продукти. [6]

Органічний ріст – природній приріст попиту на продукт, викликаний загальною всесвітньою тенденцією чи модою. Неорганічний ріст попиту – приріст, викликаний витратами на рекламу та маркетинг.

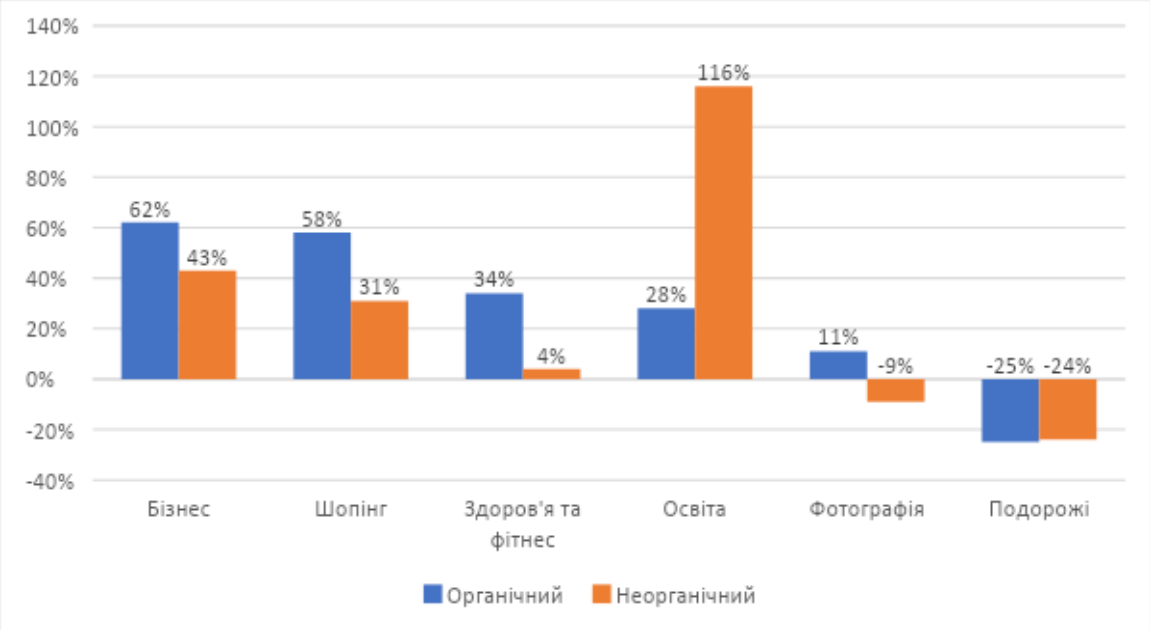

Рис. 1. Відсоткова зміна кількості встановлень за категоріями (2020 vs. 2019)

На діаграмі видно, що категорія «Освіта» збільшила свій попит на 116 % відсотків у 2020 році в порівнянні з 2019 за рахунок підвищення витрат на

рекламу та з метою наздогнати популяризацію самоосвіти на карантині. Органічний приріст даної сфери значно менший. Категорія «Здоров'я та фітнес» підвищила свою популярність на третину органічним зростанням попиту, що демонструє позитивну статистику в бажанні людей слідкувати за своїм здоров'ям та фізичною активністю. Застосунки та інструменти цієї категорії охоплюють велику кількість груп, що дозволяє користувачам широкий вибір програмних продуктів: [7]

1. *Сервіси тренувань на базі «екосистем» виробників мобільних пристроїв.* Двома корпораціями, що ділять між собою основний ринок у цій сфері є Apple та Samsung. [8] У 2020 році Apple представила новий сервіс Apple Fitness+, який був розроблений застосунком для контролю над фізичною активністю, в першу чергу для користувачів Apple Watch, і як додаткова платформа з каталогом тренувань від експертів та звітами щодо стану здоров'я користувача завдяки синхронізації на iPhone, iPad, чи Apple TV. Система представлялась як повний перезапуск вбудованого в смартфон застосунку Fitness (перший випуск – березень 2015 року), революційне рішення, яке зробить кожного користувача iOS здоровішим та більш спортивним. Проте, на жаль, сервіс доступний лише для окремих регіонів, серед яких немає України, тому наші співвітчизники не мають змоги користуватись нативним спортивним сервісом від Apple. Також варто зазначити, що сервіс доступний лише за підпискою. Місяць користування буде коштувати 10 доларів, але якщо передплатити рік, то сума до сплати складатиме 80 доларів. [9] Повним аналогом для користувачів гаджетів виробництва Samsung є нативний сервіс Samsung Health (у більш ранніх версіях – S Health), вперше випущений влітку 2012 року. Цей сервіс має порівняно більше можливостей і позиціонується компанією-розробником як безкоштовний застосунок для відстеження різних аспектів повсякденного життя і сприяння добробуту, таких як сон, споживання води, жіноче здоров'я, дієта та фізична активність. Сервіс аналогічно

синхронізується з смарт-годинником відповідного бренду. Певним недоліком у порівнянні з сервісом від Apple виступає підхід до формування фітнес-тренування. В той час, коли Apple Fitness+ записує персональні відео-уроки, Samsung Health реалізував інтеграцію з іншими популярними сервісами та дублює їх онлайн-тренування.

2. *Мобільні застосунки на платформах Android та iOS.* Саме ця категорія представляє найбільше різноманіття для потенційних користувачів. Для використання необхідно обрати бажаний застосунок у Google Play чи App Store (залежить від операційної системи конкретного смартфону) та завантажити. Більшість сучасних мобільних програмних рішень доступні одразу на обох платформах, щоб охопити більший відсоток користувачів. [10] Останній рік на першому місці в міжнародних рейтингах знаходився застосунок «Beachbody On Demand». Свою високу оцінку він заслужив великим каталогом доступних курсів, які сформовані в комплексні програми тренувань. Тобто, користувач обирає цілий комплекс, який фільтрується за складністю, довготривалістю (від шести тижнів до двох місяців) і кількістю занять на тиждень. Варто зазначити, що застосунок платний і коштує 99 доларів за рік користування. Одним з найкращих безкоштовних застосунків для тренувань в минулому році було обрано «Nike Training Club». В каталозі тренування сортуються за групами м’язів, інтенсивністю та типом тренування, тому користувач може обирати нове тренування кожного заняття спортом. Відповідно до пройдених тренувань, застосунок буде пропонувати нові персональні копмлекси для кожного окремого користувача. Перевагою застосунку є підтримка Android Wear OS, – операційна система, створена для смарт-годинників та інших переносних пристроїв, – яка дозволяє керувати вправами та стежити за пульсом.
3. *Онлайн-платформи з пошуку тренерів.* Ця категорія вирізняється дороговизною, оскільки відбуваються персональні чи групові заняття в

режимі «наживо» і тренер виступає онлайн-вчителем, що слідкує за фізичними активностями клієнта протягом «уроку». [12] Для прикладу розглянемо портал «FYT» - абревіатура від «Find Your Trainer» (укр. «Знайди свого тренера»). Сервіс позиціонується як помічник в пошуках тренера, що допоможе користувачу досягти нових спортивних звершень. Алгоритм сайту працює наступним чином: користувач проходить опитування, щоб відфільтрувати та надалі рекомендувати конкретних тренерів, обирає потрібного тренера, вказує бажаний час та місце тренування (до карантину були доступні опції «вдома», «в залі», «в парку» тощо). Оскільки даний сервіс є способом заробітку для тренерів, то тренування платні і ціна варіюється від 29 до 166 доларів за одне. Сервіс гарантує якісне надання послуг, адже перед реєстрацію тренера на сайті, модератори перевіряють сертифікацію кожного з них.

4. *Застосунки-трекери для занять бігом.* Ця категорія також відносить до мобільних кросплатформних застосунків, проте має полярно інший принцип роботи. Передбачається, що користувач займається бігом з персональним смартфоном чи смарт-годинником(в останні роки з’явилась можливість синхронізації), а той в свою чергу збирає інформацію про фізичну активність з допомогою геолокації. [13] Прикладом такого застосунку є Adidas Running (Runtastic). Програмний продукт створювався за шаблоном соціальної мережі, користувачі якої ділитимуться з друзями власними біговими досягненнями. Застосунок дозволяє відслідковувати швидкість, темп, кілометраж, зміну висоти рельєфу, яку пробіг користувач, калорії і, навіть, кількість втраченої рідини. Усім цим прогресом користувач може ділитись у новинній стрічці. Перерахований функціонал доступний у безкоштовній версії застосунку. Проте, за 35 доларів на рік користувач отримає бігові комплексні програми, що допоможуть йому підготуватись до марафону чи забігу на довгі дистанції.

1.3 Підходи до розробки веб-застосунків

Темою роботи передбачена розробка саме веб-застосунку, тому, для коректного розуміння мети роботи та процесу девелопменту, виділимо ключові поняття та основні підходи у розробці.

Всі ресурси, які доступні у вільному доступі в Інтернеті, діляться на статичні веб-сайти та інтерактивні застосунки (веб-застосунки). В класичному вигляді, веб-застосунок – це програмний продукт, що складається з клієнтської та серверної частини – фронтенду та бекенду. В свою чергу, веб-сайт являє собою набір статичних сторінок, написаних на мові розмітки HTML, що об'єднанні під одним доменним іменем. Але, наразі, важко знайти повністю статичний веб-сайт. Навіть сучасні лендінги та сайти-візитки розробляють з використанням бази даних та мінімальних інтерактивних функцій. Зважаючи на усе вищезазначене, формально будь-який актуальний сайт підпадає під означення «веб-застосунку». Проте, основною відмінністю веб-застосунку є взаємодія з користувачем. Тому новинні сайти та інформаційні ресурси лишаються веб-сайтами, а, наприклад, Google Gmail в браузері відноситься до веб-застосунків.

Технології та мови програмування для розробки веб-застосунків мають досить широку варіативність. Написання фронтенд частини виконується з використанням HTML, CSS і JavaScript. Спеціальні фреймворки спрощують процес розробки та взаємодії цих трьох «інструментів». [17] Найбільш популярними фреймворками є React, Angular, Vue, Ember.js. Серверну частину можна написати майже будь-якою мовою програмування, але найбільш популярними мовами є Java, Python (з використанням фреймворку Django), PHP (з використанням фреймворку Laravel), JavaScript з використанням платформи Node.js, Go, C# з використанням платформи Asp.net, Ruby тощо. [18] Щодо баз даних, то найбільш використовуваними є Oracle, MySQL, MongoDB, PostgreSQL та SQLite.

Існує три основних підходи у розробці веб-застосунків. Перший – SPA або Single Page Application – це односторінковий інтерактивний застосунок. Ключовою особливістю такого підходу є те, що користувач при використанні

сайту і переключенні між вкладками лишається на одній і тій самій сторінці. Тобто, користувачу завантажуються і оновлюються лише необхідні частини контенту з допомогою AJAX, а не сторінки цілком. Це надає SPA перевагу у швидкості роботи. Прикладом такого підходу є Google Gmail. [16] Те що застосунок справді односторінковий можна помітити при переключені між списками листів – адреса сторінки сайту в цей час не змінюється.

Базовою мовою для розробки таких веб-застосунків є JavaScript. Для створення невеликого інтерактиву зазвичай використовують бібліотеку jQuery. Проте, це не найбільш оптимальний варіант для більших проектів. В таких випадках краще використовувати фреймворки Vue, React чи Angular. Перевагами SPA є швидкість оновлення/завантаження, як вже зазначалось вище, швидкість розробки з допомогою фреймворків, код такого застосунку легко використовувати для розробки мобільних застосунків (особливо React Native). Як недоліки можна виділити лише необхідність ввімкненого JavaScript в браузері, для коректного відображення контенту, та певні проблеми при впровадженні SEO-оптимізації для застосунку.

Наступним підходом в розробці є MPA або Multi Page Application – це традиційні багатосторінкові веб-застосунки, коли користувач завантажує нові HTTP-сторінки в процесі взаємодії. Тому процес обміну даними в таких застосунках дещо повільніший, аніж в SPA, особливо коли відсутнє стабільне інтернет-з’єднання чи проблеми з доступом до хостингу. Яскравим прикладом Multi Page Application є такі інтернет-магазини як Rozetka чи Amazon. Основною перевагою такого підходу є масштабованість – така архітектура чудово підходить для сайтів з великою кількістю сторінок. Робота з SEO-оптимізацією теж буде комфортнішою, оскільки кожну сторінку можна окремо оптимізувати під різноманітні запити та додати мета-теги. До недоліків можна віднести нижчу швидкість взаємодії, складність та коштовність розробки, оновлення та підтримки таких об’ємних програмних продуктів.

Третім підходом виділяють PWA або Progressive Web Application. Це порівняно новий підхід в розробці, який часто називають новим, ефективнішим,

більш функціональним SPA. Такий прогресивний веб-застосунок нагадує за своїми функціональними можливостями нативні десктопні або мобільні застосунки. Основними відмінностями і перевагами є те, що такий сайт можна додати на головний екран мобільних застосунків, він вміє відправляти push-повідомлення, взаємодіяти з гео-локацією і працювати в режимі офлайн. Яскравим прикладом слугує Google Docs, який за своїм змістом є онлайн-офісом, але користувач може використовувати застосунок і в автономному режимі. Також до переваг варто додати кросплатформність та захищеність – PWA працює тільки за HTTPS протоколом. Наразі, основним недоліком є відсутність підтримки на старих версіях мобільних ОС, тому розробку програмного продукту з таким підходом важко назвати вдалим з точки зору покриття всіх користувачів. Але, PWA тільки розпочинає заходити на ринок девелопменту і в найближчому майбутньому ситуація обов'язково зміниться.

1.4 Формування завдання на розробку

Основним завданням дипломної роботи полягає, як було визначено вище, у побудові архітектури та розробці веб-застосунку для фітнес-тренувань. Враховуючи розглянуті підходи до розробки та зважаючи на описані переваги та недоліки, оптимальним патерном для розробки веб-застосунку є SPA – односторінковий інтерактивний застосунок. Тому для якісного виконання визначеного завдання необхідно дослідити та виконати наступні задачі:

- визначити стек технологій для розробки застосунку;
- описати ключові частини функціоналу, що будуть розроблятись, для отримання загального уявлення про розроблюваний програмний продукт, а також деталізовано описати безпосередньо кожну розроблювану одиницю функціоналу (фічу) з визначенням пріоритету на розробку;
- створити діаграму для демонстрації загальної взаємодії системи;
- описати середовище розробки та кінцевої роботи веб-застосунку;

- визначити обмеження на дизайн та реалізацію;
- розробити користувацькі сценарії (Use Cases);
- визначити нефункціональні вимоги до програмного продукту, а саме: вимоги до продуктивності, атрибути якості програмного забезпечення, вимоги до безпеки застосунку.

Для детального опису та моделювання архітектури програмної системи застосувати мову UML (Unified Modeling Language) – уніфіковану мову для об’єктного моделювання та його графічного опису. [19] Розробити наступні діаграми:

- діаграму компонентів для демонстрації розбиття програмної системи на структурні компоненти та зв’язків між ними;
- діаграму композитної структури, щоб показати внутрішню структуру класів і взаємодію внутрішніх елементів;
- діаграму розгортання для моделювання працюючих вузлів (апаратних засобів), необхідних для коректної роботи веб-застосунку;
- діаграму прецедентів, щоб відобразити відносини між акторами (користувачами системи) та прецедентами (елементами системи);
- діаграму послідовності, щоб показати впорядковану в часі взаємодію об’єктів і повідомлень, якими вони обмінюються.

Оскільки розроблюваний програмний продукт буде працювати в браузері і є веб-застосунком, то патерном, для представлення даних, буде MVC - Model View Controller.

Висновки до розділу

Станом на 2021 рік системи охорони здоров’я більшості країн світу кинули всі свої сили та ресурси на боротьбу з вірусом, що змушує вводити карантинні міри. Проте, це не зменшує небезпеку неінфекційних захворювань людей, викликаних станом навколишнього середовища та персональним способом

життя кожного з них. В деяких випадках карантин провокує розвиток таких пагубних звичок як малорухомий спосіб життя. Наведено статистичні дані, що підтверджують необхідність посилення профілактичних заходів і популяризації фізичних навантажень. Враховуючи обставини, визначено, що актуальним сучасним підходом є проведення тренувань в режимі онлайн.

Наведено дані, що демонструють органічний ріс попиту застосунків категорії «Здоров'я і спорт» у 2020 році в порівнянні з попереднім. Також описано розподілення на основні групи застосунків: сервіси тренувань на базі «екосистем» виробників мобільних пристроїв, мобільні застосунки на платформах Android та iOS, онлайн-платформи з пошуку тренерів та застосунки-трекери для занять бігом. В кожній з категорій наведені приклади застосунків з описом ключових можливостей, переваг та недоліків кожного з них.

Підсумовуючи наведений опис, виділено, що більшість застосунків є мобільними, тому визначена мета розробки веб-застосунку є актуальною. Наведено три основних підходи у розробці веб-застосунків, проаналізовано переваги та недоліки кожного та виділено найбільш оптимальний для подальшого девелопменту програмного продукту.

Визначено завдання та задачі для подальшого процесу реалізації веб-застосунку.

РОЗДІЛ 2. ІНСТРУМЕНТАРІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Опис мов програмування та засобів для розробки веб-застосунку

Базовою тріадою веб-програмування є поєднання HTML, CSS та JavaScript, інтерпретація яких дозволяє користувачам бачити звичні для них сайти у будь-якому браузері. [20] При цьому, в загальному представлені HTML відповідає за структуру сторінки в браузері, CSS - за стилізацію та відображення сторінки користувачу, а JavaScript відповідає за динамічність і поведінку елементів веб-сторінки.

Технічно HTML і CSS не є мовами програмування, це структурована інформація про каркас і стилі сторінки. Проте, вони утворюють базис, без якого неможливе існування сайтів і веб-застосунків в їх сучасному представлені.

HTML (від англ. HyperText Markup Language) - мова гіпертекстової розмітки, що використовується для створення веб-сторінок. В першу чергу, є інструментом для створення логічної розмітки сторінки за допомогою тегів.

Тег - спеціальний маркер, що позначає кожен елемент сторінки і допомагає браузеру правильно інтерпретувати, а відповідно і відображати, бажаний вміст сторінки. Зміст і відображення тегу матиме різний інтерфейс відображення в браузері залежно від того, яким тегом вони огорнуті.

HTML має довгу історію створення та декілька версій, що мають свої особливості. З 2014 року актуальною версією лишається HTML5 - це наступна суттєва редакція стандарту HTML, що замінює HTML 4.01, XHTML 1.0 та XHTML 1.1. Остання версія представила широкий спектр оновлень додавши нові елементи та атрибути для побудови сучасних веб-застосунків та сайтів. До основних нововведень належать:

- нові семантичні елементи, які надають змогу створити правильно структуровану логічну розмітку сторінки: <header>, <main>, <aside>, <footer> та інші;

- розширений список атрибутів для тегу <input>, що представляє нові можливості у створенні веб-форм;
- поява тегу <canvas>, що підтримує двовимірну панель для малювання фігур шляхом написання скриптів на JavaScript;
- теги <audio> та <video> для нативної підгрузки аудіо та відео файлів, без використання сторонніх плагінів;
- функціонал геолокації за вимогою користувача - тепер користувачі можуть обирати чи поширювати власне місцезнаходження з активним веб-застосунком.

Головною перевагою HTML є те, що будь-яку сторінку, написану цією мовою, можна без особливих проблем переглянути у будь-якому веб-браузері на різних платформах.

CSS (від англ. Cascading Style Sheets) - каскадні таблиці стилів або просто стилі - це набір параметрів форматування, що застосовуються до елементів у документах для змін їх зовнішнього виду. [21] До основних можливостей та переваг CSS належать:

- економія часу розробки - один написаний стиль можна застосовувати до необмеженої кількості елементів та сторінок;
- швидше завантаження сторінок - стилі описані в окремому файлі і не має потреби постійно прописувати їх в самому HTML для кожного тегу;
- проста підтримка і можливість змін - для внесення глобальних змін варто змінити стилі і вони автоматично оновлять елементи на веб-сторінках;
- ширші можливості в стилізації елементів - CSS має набагато більший перелік атрибутів, ніж HTML;
- сумісність на декількох пристроях - CSS дозволяють оптимізувати вміст веб-сторінок для різних типів пристроїв.

Коли браузер відображає документ, то має погодити його вміст (HTML розмітка) зі стилями. [22] Цей процес проходить в кілька етапів:

1. Коли браузер завантажує HTML-файл, він бере написаний код, аналізує його та починає інтерпретувати його построково в майбутню HTML-сторінку.
2. Під час інтерпретації браузер перетворює HTML в DOM (від англ. Document Object Model) - це уявлення сторінки в пам'яті комп'ютера у вигляді дерева з визначенням батьківських та дочірніх елементів.
3. Також, при аналізі вихідного HTML коду, браузер знаходить файли стилів і починає інтерпретувати CSS.
4. Синтаксичний аналіз CSS складається з двох кроків - вирішення конфліктних оголошень стилів та обробка кінцевих значень.
5. Опрацьований CSS також зберігається у древовидній структурі, схожій на DOM - CSS Object Model.
6. Інтерпретовані HTML та CSS разом складають дерево рендерингу. Цим етапом завершується робота над файлами.
7. Модель візуального форматування - інструмент, використовуваний браузерами для візуалізації веб-сторінок. Цей алгоритм визначає блоки і їх макет для кожного елементу у дереві рендерингу, для визначення кінцевого макету сторінки.
8. Після моделі візуального форматування сторінка нарешті відображається користувачу на екрані його девайсу.

Схематичне відображення роботи браузера для відображення стилізованого HTML-файлу подано на рисунку 2. [23]

JavaScript або просто JS - це мова програмування на основі логіки, яку можна використовувати для модифікації вмісту веб-сайту та змусити його поводитися по-різному у відповідь на дії користувача. Поширене використання JavaScript включає поля підтвердження, активні елементи сайтів з власною поведінкою та додавання нових ідентифікаційних даних до існуючої інформації. [25] Більша частина динамічної поведінки, яку ви побачите на веб-сторінці, завдяки JavaScript, який покращує елементи керування та поведінку браузера за замовчуванням.

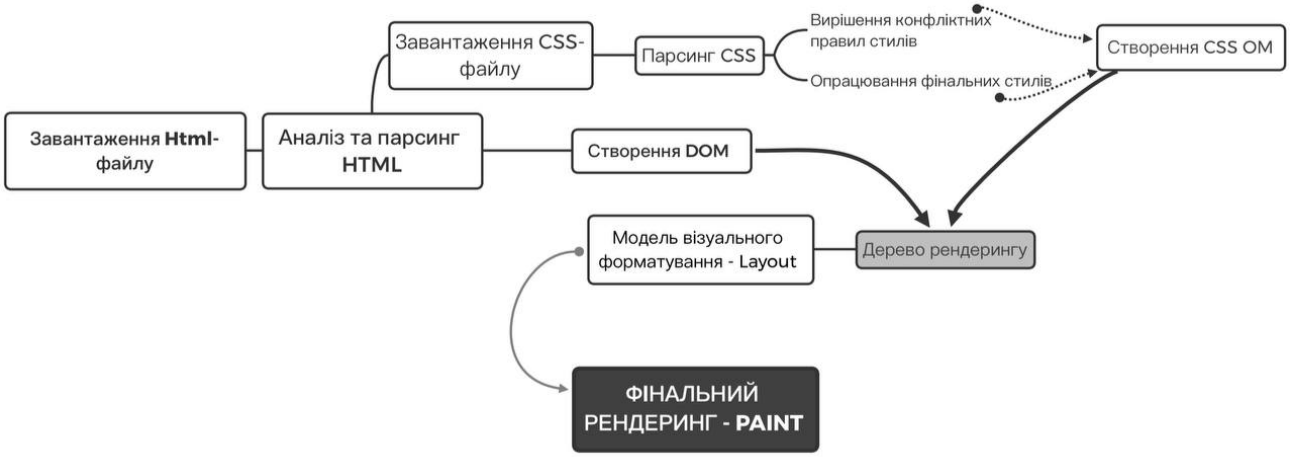

Рис. 2. Загальна схема роботи браузера при рендерінгу типової веб-сторінки

Ключовими перевагами цієї мови програмування є:

- Javascript входить в топ-3 найпопулярніших мов програмування у світі, і це сильно впливає на вибір досвідчених програмістів та початківців;
- Javascript є широко застосовуваним, він підтримується кожним сучасним веб-браузером на будь-яких типах девайсів, і тому для вивчення Javascript немає потреб в додаткових налаштуваннях середовищ розробки. Наприклад, Chrome, Mozilla Firefox, Safari та кожен веб-переглядач, який є на ринку сьогодні, підтримує Javascript;
- одним з останніх великих нововведень в JavaScript є поширення цієї мови на розробку мобільних додатків, розробку настільних додатків та розробку ігор. Це відкриває багато можливостей для програміста, що обрав Javascript ключовою прикладною навичкою для побудови своєї кар’єри;
- чудовим у Javascript є те, що існує та доступно безліч вже розроблених фреймворків та бібліотек, які можна використовувати безпосередньо при розробці програмного забезпечення, щоб скоротити час виходу застосунку на ринок. Топ фреймворків та

бібліотек складають наступні - Angular, React, jQuery, Vue.js, Node.js. [24]

Для сучасного веб-застосунку JS відіграє вагому роль, адже керує усією поведінкою елементів, що забезпечує якісну взаємодію з користувачем, а саме: може валідувати введення даних користувачем з клавіатури та надсилати їх на сервер для збереження, маніпулювати сторінками, завантажувати дані з серверу в фоновому режимі з допомогою бібліотеки Ajax, надсилати користувачам сповіщення тощо.

Веб-сайт може бути статичним чи динамічним. Проте, для будь-якого веб-застосунку з мінімальною складністю необхідна система управління базами даних (скорочена як **СУБД**). Наразі виділяють 3 системи управління базами даних, які зазвичай використовуються для веб-застосунків у сучасній розробці.

Перше місце у цій вершині займає добре відома та перевірена СУБД - MySQL. Це СУБД з відкритим кодом для реляційних баз даних. Часто використовують в поєднанні з PHP. Входить в стек LAMP, що позначає Linux (як операційну систему для сервера), Apache (як веб-сервер), MySQL (як СУБД) та PHP (як мову програмування). Мабуть, найбільшим проектом, який використовує його сьогодні, є WordPress. [26]

Свій шлях на ринку MySQL починав як швидке рішення для баз даних і початково не підтримувались розширені функції баз даних, які доступні користувачам сьогодні (зовнішні ключі, обмеження перевірки та транзакції). MySQL дозволяє користувачеві вибирати з декількох механізмів обробки даних. Є 2 основні варіанти:

- MyISAM: швидший через обмеження можливостей роботи і застосовується при розробці застосунків, що вимагають велику кількість звернень на читання з БД в таблицях та повнотекстовий пошук.
- InnoDB: двигун , що забезпечив відсутні елементи (ключі, транзакції тощо) та став популярнішим з часом, поки його не придбав Oracle.

Багато розробників перейшли на PostgreSQL, найдосконалішу реляційну СУБД з відкритим кодом. Він надає всі функціональні можливості, які можуть знадобитися з бази даних, будучи продуктом, що підтримується спільнотою. Це як Debian СУБД. Став дуже популярним з 2005 року, коли Ruby on Rails випустив свою першу версію, використовуючи цю СУБД за замовчуванням. Rails підтримує деякі корисні функції PostgreSQL, що полегшує розробку. Наприклад, Rails 4 пропонував підтримку масивів та хешів PostgreSQL нестандартно. Це дозволило забути про створення проміжних таблиць для цього і використовувати функцію безпосередньо зі своєї СУБД.

Проте, частина розробників, що звикли до графічних інтерфейсів для роботи з СУБД, виділяють відсутність останнього як головний недолік PostgreSQL. Якщо є необхідність чи бажання працювати саме з цією СУБД, то варто звикати до командного рядка і вчити відповідні команди.

Закриває топ-3 СУБД MongoDB. Це, мабуть, найпопулярніша СУБД NoSQL із відкритим кодом. Замість реляційного дизайну з таблицями, MongoDB зберігає дані як колекції документів, що в рази спрощує дизайн та архітектуру застосунку, щоб легко масштабуватися за допомогою кластерів (насправді це автоматично відбувається з використанням автоматичного розподілення).

Це дуже цінують розробники JavaScript, оскільки таблиці замінюються документами, подібними до JSON. Ці документи не потребують визначення структури, ми можемо просто додати їх із будь-якою структурою, і її форма буде динамічно адаптуватися. Деякі складні типи даних також можна легко зберігати, наприклад масиви.

Цей сучасний підхід до зберігання даних, більш природний для програмістів, змусив багатьох перейти на цю СУБД NoSQL, але в багатьох випадках це було помилкою, оскільки вони хочуть просто замінити одну систему іншою. MongoDB не призначений для безпосередньої заміни реляційних баз даних.

Останнім інструментом розробки, який відносять до базових, є система контролю версій, зокрема **Git**. Процес розробки передбачає роботу з великою

кількістю файлів, а відповідно і з їх зміною, тому системи контролю версій - це необхідні інструменти, що дозволяють зробити розробку більш гнучкою та адаптивною до змін. Зокрема, Git є однією з найбільш ефективних, популярних, надійних і високопродуктивних систем керування версіями. Вона надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. [28]

Система спроектована як набір програм, спеціально розроблених з урахуванням їхнього використання у скриптах. Це дозволяє зручно створювати спеціалізовані системи керування версіями на базі Git або користувацькі інтерфейси. Тому більшість сучасних редакторів коду мають вбудовані можливості роботи з GIT.

2.2 Стек технологій як комплексний підхід у веб-розробці

Стек (від англ. Stack - стопка) технологій - це набір інструментів, що застосовується при роботі в проектах і включає мови програмування, фреймворки, системи управління базами даних, компілятори тощо. Від обраного розробником стека технологій залежать продуктивність роботи, вимоги до апаратних ресурсів, надійність роботи програмного забезпечення.

Розробка веб-застосунків активно застосовує цей принцип для формування патернів, що можуть використовувати розробники, для створення модулів, взаємодія яких складатиме повний функціонуючий застосунок. Критеріями, які враховується при виборі технологій та формуванні стеку розробки, є: розмір і тип проекту, складність проекту, швидкість розробки, вартість та доступність фахівців, доступні інструменти розробки, наявність готових рішень, вартість підтримки, вимоги до навантажень, кросплатформеність та можливості інтеграції з іншими рішеннями.

2.2.1 Огляд сучасних стеків технологій

Станом на 2021 рік найбільш використовуваними є наступні стеки:

- MEAN,
- MERN,
- LAMP,
- Ruby on Rails Tech Stack. [30]

Опишемо та визначимо ключові особливості кожного для аналізу та порівняння.

Експерти вважають технологію MEAN найкращою для веб-розробки завдяки різним перевагам. Він складається з MongoDB (NoSQL DB), Express.js (фреймворк веб-застосунків, що реалізує сервер веб-середовища), Angular (інтерфейсний фреймворк) і Node.js (кросплатформний сервер з відкритим кодом), і може бути використаний для розробки веб-застосунків різного рівня складності.

Єдиною мовою, що використовується у цьому стеку, є JavaScript. Його компоненти чудово вміють взаємодіяти з даними у форматі JSON і передавати останні з доступом до бібліотеки модулів. Це означає, що веб-розробники можуть повторно використовувати цей код у всьому додатку, дотримуючись DRY принципу розробки. Знання JavaScript, HTML та CSS достатньо для роботи з цим стеком технологій веб-розробки. Стек допомагає розробляти швидкі, високоефективні та масштабовані програмні застосунки.

MERN майже ідентичний MEAN із невеликою кількістю технологічних змін, де Angular замінюється React. Головною перевагою використання MERN є інтеграція React, його потужна бібліотека та можливість одночасного використання коду на серверах та браузерах. Крім того, він має феноменальні можливості повнорозмірної розробки (інтерфейс та серверний інтерфейс). React використовує JavaScript XML та Virtual DOM, і ці компоненти працюють та впроваджують зміни без проблем.

React - це популярний фреймворк, відомий своєю гнучкістю та підходом, орієнтованим на продуктивність, що дозволяє створювати топові односторінкові

програми з інтерактивними інтерфейсами. Стек технологій MERN постачається з великим набором інструментів тестування та відкритим кодом із підтримкою спільноти. Це другий за популярністю стек веб-технологій 2021 року.

LAMP - це старий класичний галузевий стандарт, якщо йдеться про перевірені часом стеки веб-розробки, що включає MySQL (Реляційне управління базами даних), Linux (Операційна система), PHP (Мова програмування) та Apache (HTTP-сервер). Він відкритий і доступний. Стек ефективно працює на всіх операційних системах. При веб-розробці це забезпечує веб-сайту найкращу продуктивність, економічну ефективність та гнучкість. Його компоненти можуть бути замінені або змінені в межах одного стека.

Ruby on Rails Tech Stack (RoR) - це стек веб-розробки, зручний для розробників. Він є відкритим, об’єктно-орієнтованим і використовує динамічну мову програмування під назвою Ruby. RoR сприяє розробці легких додатків, що збільшують гнучкість. Стек працює в тандемі з HTML, CSS та JavaScript для створення інтерактивних користувацьких інтерфейсів та XML або JSON для передачі даних. Це дозволяє використовувати структури за замовчуванням для веб-сторінок та управління базами даних. Він також надає розробникам детальний журнал помилок для їх найбільш швидкого усунення.

Також варто зазначити, що цей стек самий юний з описаних, тому робота з ним може бути дещо важчою в порівнянні з іншими за рахунок дефіциту інформації про досвід роботи інших розробників.

Безумовно, кожен стек має особисті переваги та недоліки, на які варто зважати при виборі технологій для розробки, щоб побудувати веб-застосунок відповідно до вимог. Але, обравши будь-який представлений стек, можна розробити дійсно якісне та сучасне ПЗ.

2.2.2 Обґрунтування вибору стеку технологій

Оскільки визначено, що кінцевим результатом розробки має бути односторінковий інтерактивний застосунок, то стеком ключових технологій обрано MERN.

До переваг цього стеку також можна віднести рендеринг та продуктивність користувацького інтерфейсу загалом, а також - легкість в роботі та налаштуванні взємодії серверної та інтерфейсної частини, що так важливо при розробці невеликих інтерактивних веб-застосунків. Як зазначалось, MERN складається з чотирьох основних інструментів розробки.

MongoDB - це база даних NoSQL, в якій кожен запис бази є окремим документом (колекцією), що складається з пар ключ-значення, подібних до об'єктів JSON (JavaScript Object Notation). MongoDB є гнучкою і дозволяє своїм користувачам створювати схеми, бази даних, таблиці тощо. Документи, які можна ідентифікувати за допомогою первинного ключа, складають основну одиницю MongoDB.

Після встановлення MongoDB користувачі також можуть використовувати оболонку MongoDB Compass. Оболонка MongoDB Compass надає інтерфейс JavaScript, за допомогою якого користувачі можуть взаємодіяти та виконувати операції (наприклад: запити, оновлення записів, видалення записів).

Переваги MongoDB:

- Масштабованість - обробляти великі дані можна, розділивши їх на кілька машин.
- Використання JavaScript - MongoDB використовує JavaScript, що є найбільшою перевагою.
- Без схеми - будь-який тип даних в окремому документі.
- Дані, що зберігаються у формі JSON - JSON має широкий спектр сумісності з браузерами.
- Спільний доступ до даних - дані будь-якого розміру та типу (відео, аудіо) можуть легко передаватися.

- Просте налаштування навколишнього середовища - його дуже просто налаштувати MongoDB.
- Гнучка модель документа - MongoDB підтримує модель документа (таблиці, схеми, колонки та SQL), яка швидша та простіша.

MongoDB надзвичайно добре працює з Node.js і робить зберігання, маніпулювання та представлення даних JSON на кожному рівні програми неймовірно простим. Для власних хмарних додатків MongoDB Atlas робить це ще простішим, надаючи можливість автоматичного масштабування кластера MongoDB на обраному хмарному провайдері в кілька кліків.

Express - це фреймворк Node.js. Замість того, щоб писати код за допомогою Node.js та створювати навантаження модулів Node, Express спрощує написання внутрішнього коду. Express допомагає у розробці серверної частини веб-додатків та API, підтримує багато проміжних програм, що робить код коротшим та простішим. Перевагами, що надають таку популярність цьому фреймворку, є : асинхронність та однопоточність, ефективність, швидкість та масштабованість, має найбільшу спільноту для Node.js, сприяє повторному використанню коду завдяки вбудованому маршрутизатору, має надійний API.

React - це бібліотека JavaScript, яка використовується для побудови користувальницьких інтерфейсів. React використовується для розробки односторінкових додатків та мобільних додатків завдяки своїй здатності обробляти швидкозмінні дані і створювати незалежні компоненти інтерфейсу, що можуть імплементувати одне одного. До характерних особливостей відносять:

1. Віртуальний DOM - віртуальний DOM-об'єкт є поданням об'єкта DOM. Віртуальний DOM насправді є копією оригінального DOM. Будь-які зміни у веб-програмі призводять до того, що весь інтерфейс повторно відображає віртуальний DOM. Потім порівнюється різниця між початковим DOM і цим віртуальним DOM і вносяться зміни відповідно до вихідного DOM.

2. JSX - означає JavaScript XML. Це розширення HTML / XML JavaScript, яке використовується в React. Спрощує та спрощує написання компонентів React.
3. Компоненти - ReactJS підтримує Компоненти. Компоненти - це будівельні блоки інтерфейсу користувача, де кожен компонент має логіку та вносить свій внесок у загальний інтерфейс користувача. Ці компоненти також сприяють повторному використанню коду та полегшують розуміння загальної веб-програми.
4. Висока продуктивність - такі функції, як Virtual DOM, JSX та Components, роблять це набагато швидшим, ніж інші фреймворки.

Node.js надає середовище JavaScript, яке дозволяє користувачеві запускати свій код на сервері поза браузером. Менеджер пакетів вузлів, тобто npm, дозволяє користувачеві вибрати з тисяч безкоштовних пакетів (модульних вузлів) для завантаження. Реалізує потокове передавання даних, побудований на двигуні JavaScript Chrome від Google Chrome, тому має швидке виконання коду.

Гарною практикою у роботі з Node.js та MongoDB є використання пакету mongoose - це ORM (об'єктно-реляційне відображення) для баз даних mongodb для роботи з вузлами. Представляє собою низькорівневу бібліотеку, яка є посередником при звернені до БД. ORM дозволяє створювати моделі та залежності між ними та використовувати спеціальний API для роботи з базою. Це дещо зменшує гнучкість системи, проте компенсує це кращою роботою з кодом та його подальшою підтримкою та структуризацією. [34]

Висновки до розділу

Розробка сучасних веб-застосунків нараховує велику кількість етапів. Першочерговим є вибір мов технологій з допомогою яких буде вестись процес імплементації застосунку. Визначено, що ключові інструменти для розробки складаються з HTML, CSS, JavaScript, бази даних та системи контролю версій.

Виділено базові характеристики кожної мови та інструменту, наведено топ-3 найбільш використовуваних баз даних у сфері веб-програмування та обґрунтовано переваги використання систем контролю версій.

Описано найбільш популярні стеки технологій для веб-застосунків, визначено переваги та особливості кожного окремого стеку. Наведено описову характеристику стеків MEAN, MERN, LAMP, Ruby on Rails Tech Stack.

Обрано та обґрунтовано стек технологій, бібліотек для розробки та побудови архітектури програмного застосунку. Окреслено переваги фундаментальних технологій обраного стеку технологій.

РОЗДІЛ 3. ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ ВЕБ-ЗАСТОСУНКУ

3.1 Загальний опис розроблюваної системи

Вагому частину ресурсів при розробці інтерактивного програмного застосунку у сфері веб займає саме розробка користувацького інтерфейсу (від англ. user interface, UI). [35] Користувацький інтерфейс - це точка взаємодії між користувачем та цифровим пристроєм чи продуктом. Щодо веб-сайтів та програм, дизайн інтерфейсу враховує зовнішній вигляд, відчуття та інтерактивність продукту. Вся справа в тому, щоб переконатися, що користувальницький інтерфейс продукту є якомога інтуїтивнішим та дружнім до кінцевого користувача (від англ. user-friendly), а це означає ретельно продумати кожен візуальний інтерактивний елемент, з яким може зіткнутися користувач.

В допомогу розробникам створено спеціальні генератори палітр та шаблонізатори для формування базових лейаутів (від англ. Layout - макет) сторінок. [36] Для допомоги у визначені кольорової теми було залучено сервіс Coolors. Сайт надає можливість обрати палітру з вже існуючих або створити свою персональну. Результатом роботи стала наступна кольорова тема:

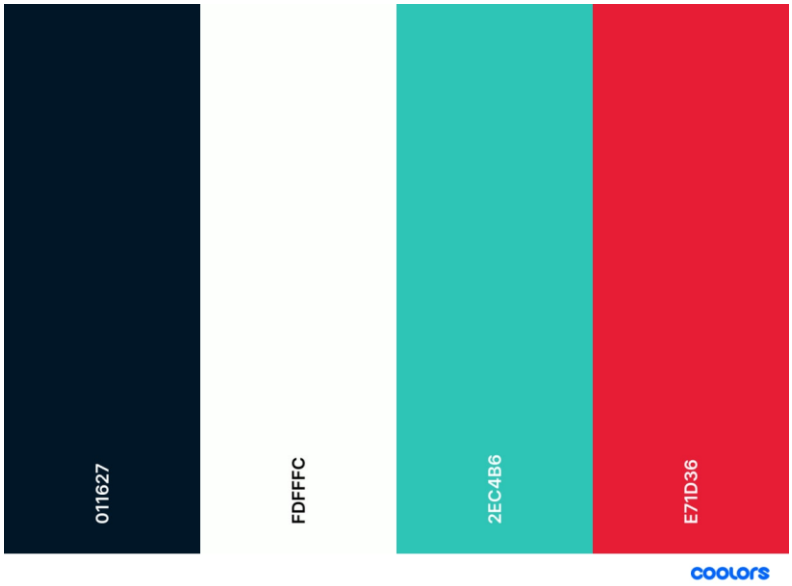

Рис. 3.1 Кольорова палітра розроблюваного веб-застосунку

Оскільки стеком технологій обрано MERN, то за розробку користувацької частини відповідає фреймворк React. Він дозволяє збирати складний UI з маленьких ізольованих шматочків коду, що називаються «компонентами». У зв'язку до React підключено бібліотеку компонентів Material-UI. [38] Бібліотека має широкий вибір компонентів за замовчуванням, доступних за персональним API. Також ця бібліотека надає можливість кастомізації при розробці програмних продуктів.

app > src > theme > index.js > ...

```
1   // Core
2   import { createMuiTheme } from '@material-ui/core';
3
4   export const theme = createMuiTheme({
5     palette: {
6       primary: {
7         main: '#e71d36',
8       },
9       secondary: {
10        main: '#2ec4b6',
11      },
12      inherit: {
13        main: '#fdfffc',
14      },
15      textPrimary: {
16        main: '#011627',
17      },
18      textSecondary: {
19        main: '#f5f5f5',
20      },
21    },
22  });
23
```

Рис. 3.2 Представлення обраної кольорової палітри у коді застосунку

Для цього у корені проекту необхідно створити папку з перевизначенням основних кольорів, до яких будуть звертатись компоненти. Material-UI потребує наступних властивостей:

1. Основний колір (primary) - використовується для представлення основних елементів інтерфейсу для користувача. Це колір, який відображається найчастіше на екранах і компонентах вашого додатка.

2. Другорядний колір (secondary) - використовується для представлення вторинних елементів інтерфейсу для користувача. Він надає більше способів акцентувати увагу та відрізняти ваш товар. Наявність його необов’язкова.
3. Колір для текстових компонентів (text) – використовується для представлення елементів, що складають текстову частину контенту, що відображається.

Відповідно до сформованого завдання на розробку, визначеного стеку технологій та паттерну архітектури програмного продукту розроблено композитну діаграму (рис. 3.3) для деталізованого опису структури і створення прототипу карти веб-застосунку.

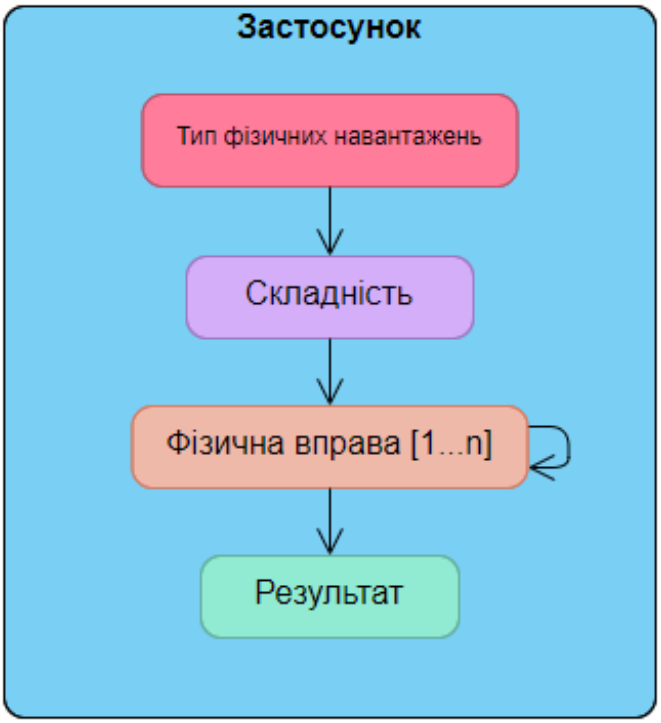

Рис. 3.3 Проста композитна структура роботи застосунку

Для створення якісного та дружнього користувацького веб-застосунку для фітнес-тренувань було визначено реалізувати можливість вибору типу тренування та рівня складності відповідно до потреб кожного окремого користувача. Відповідно, в залежності від персоналізованого вибору варіюватиметься кількість вправ для кожного окремого тренування.

Враховуючи проведений аналіз аналогів, визначені особливості, переваги та недоліки, можливості вибору типу тренування у програмному продукті варіюється наступними позиціями: йога (yoga), аеробіка (aerobics) та силові навантаження (workout). Таким чином, перераховані типи є основними компонентами розробки веб-застосунку. Кожен компонент в свою чергу є вузлом навігації та фізичною сторінкою відображення для користувацького інтерфейсу, тому навігаційна панель описана наступним чином:

- домашня сторінка;
- тип тренування 1;
- тип тренування 2;
- тип тренування 3.

app > src > navigation > index.js > ...

```
1   // Core
2   import React from 'react';
3   // Components
4   import { Route, Switch, Redirect } from 'react-router-dom';
5   import { Home } from '../pages/Home';
6   import { Aerobics } from '../pages/Aerobics';
7   import { Workout } from '../pages/Workout';
8   import { Yoga } from '../pages/Yoga';
9
10  // Instruments
11  import routes from './routes';
12
13  export const Navigation = () => (
14    <Switch>
15      <Route exact component={Home} path={routes.home} />
16      <Route exact component={Aerobics} path={routes.aerobics} />
17      <Route exact component={Workout} path={routes.workout} />
18      <Route exact component={Yoga} path={routes.yoga} />
19      <Redirect to={routes.home} />
20    </Switch>
21  );
22
```

Рис 3.4 Реалізація навігації сторінок у веб-застосунку

Базуючись на загальноприйнятих стандартах розробки програмних продуктів у сфері веб до компонентів обов’язкової реалізації варто віднести хедер та футер сайту, інформаційні сторінки та модальні вікна з застереженнями

для кінцевого користувача. Макет загального вигляду сторінки веб-застосунку представлено на рисунку 3.5.

Рис. 3.5 Макет інтерфейсу сторінки активного тренування

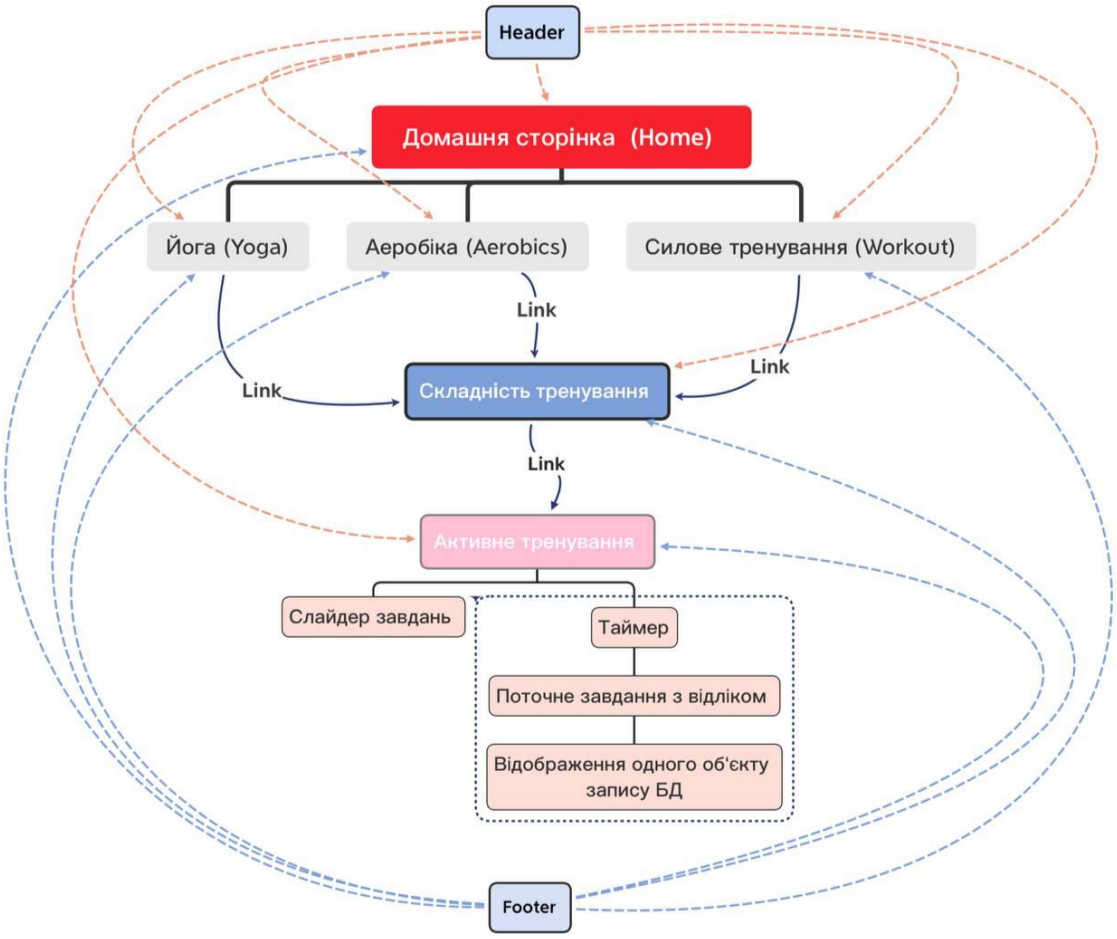

Рис. 3.6 Композитна структура веб-застосуноку

Головним компонентом застосунку є вікно активного тренування з описом фізичної вправи, картинкою відповідної вправи та таймером виконання. Зважаючи на обсяг даних, які має отримувати та відображати застосунок за одиницю часу, то використання бази даних MongoDB і зберігання цих даних на хмарному сервісі є обґрунтованим та доречним. Для компоненту таймеру використано відкриту додаткову бібліотеку «react-countdown-circle-timer». Взаємодію та доступ компонентів один до одного зображено на рисунку 3.6.

3.2 Архітектура серверної частини веб-застосунку

База даних – це сукупність певної кількості текстових структур, основною метою яких є зберігання великих обсягів даних та різноманітних модулів, що здійснюють управління, сортування, видалення та будь-яке інше керування цими даними, що керується сервером застосунку. Для досягнення поставленої мети бакалаврської роботи та розробки якісного програмного продукту було створено базу даних помірного обсягу, що зберігається не локально, а базується віддалено з допомогою MongoDB Atlas.

MongoDB Atlas – хмарний сервіс, що розгортає та масштабує базу даних на AWS, Azure та Google Cloud, в залежності від потреб розробника. MongoDB Atlas дозволяє легко контролювати доступ до бази даних. Екземпляри бази даних розгортаються в унікальній віртуальній приватній хмарі (VPC), щоб забезпечити ізоляцію мережі. Інші функції безпеки включають білий список IP-адрес або VPC Peering, постійно ввімкнену автентифікацію, шифрування в спокої та шифрування під час передачі, складне управління рольовим доступом тощо.

Рольовий доступ включає можливість додавання користувачів з встановленням приватних паролів, методів аутентифікації та обмежень доступу до ресурсів (одна база даних чи кілька).

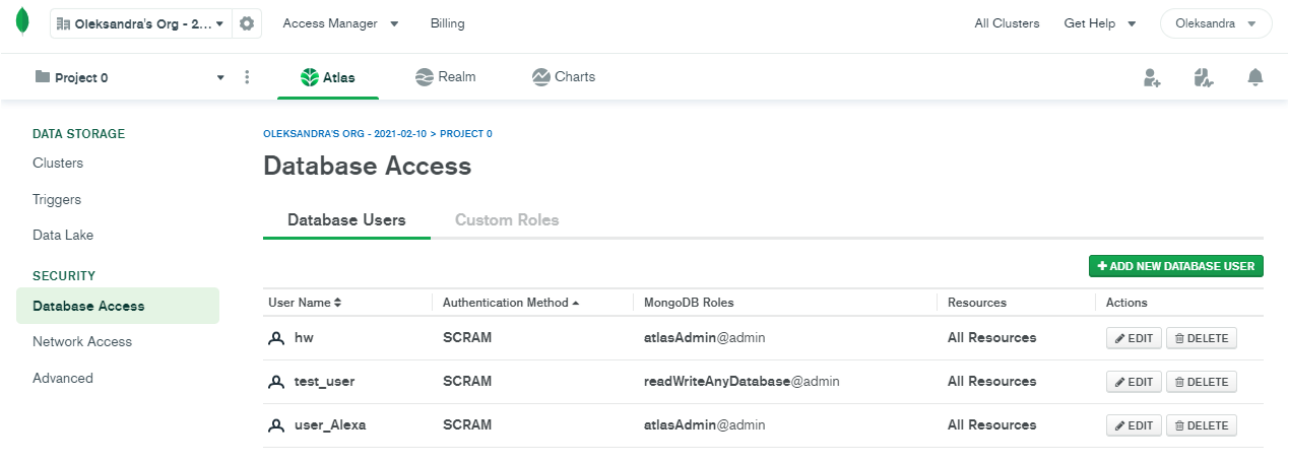

Рис. 3.7 Налаштування безпеки до кластеру баз даних через рольові доступи

Робота з СУБД MongoDB та mongose передбачає створення моделі для взаємодії з даними.

app > server > model > exercise-model.js > ...

```
1  const mongoose = require('mongoose');
2
3  const exerciseSchema = new mongoose.Schema({
4    name: {
5      type: String,
6    },
7    description: String,
8    type: {
9      type: String,
10   },
11   time: {
12     type: Number,
13   },
14   image: {
15     data: Buffer,
16     contentType: String,
17   },
18 });
19
20 module.exports.Exercise = mongoose.model('Exercise', exerciseSchema);
21
```

Рис. 3.8. Модель вправи, що складає одиницю запису колекції бази даних

Враховуючи особливості розроблюваного програмного продукту, база даних складатиметься з однієї колекції, що міститиме перелік усіх вправ. Кожен запис в базі даних має наступні поля:

- Назва вправи,
- Опис вправи,

- Тип вправи,
- Час виконання,
- Зображення вправи.

Тип вправи має стандартні значення на вибір, що дублюють можливості вибору типу тренування у користувача. Час вправи – значення, що зберігається у секундах для реалізації зворотного відліку таймеру при активному тренуванні.

Формат зберігання даних у базі наведено на рисунку 3.9. Окремо варто відзначити формат зберігання даних зображення, адже зберігається не посилання, а сам фізичний файл у бінарній інтерпретації. Відповідно, для правильної трансляції картинки у браузері обов'язковим є написання функції перетворення бінарного рядка у файл з розширенням png.

Рис. 3.9 Представлення одиниці даних у інтерфейсному застосунку для СУБД – MongoDB Compass

Для підвищення рівня абстракції у роботі з даними на серверній частини запит на базу даних винесено в окремий файл, який являє собою DAO (від англ. data access object) – спеціальний об'єкт доступу до даних. Використання DAO вважається гарною практикою для розподілення бізнес-логіки серверу та запитів, що входять до CRUD. Рисунок 3.10 демонструє код асинхронного запиту до бази даних на отримання масиву об'єктів, який варіюється в залежності від типу

фізичних вправ та складності (характеризується лімітом вправ – кількістю елементів масиву).

```
module.exports.getAllExercise = async (type, limit) => {
    const allExercise = await Exercise.find({type: type}, [], {limit: +limit});
    return allExercise;
};
```

Рис. 3.10 DAO-файл для створення абстрактного інтерфейсу роботи з даними

Сам об'єкт доступу до даних імпортується в контролер, який регулює процес звернення сервера до бази. (Рис. 3.11)

```
app > server > controller > exercise-controller.js > showExercises > module.exports.showExercises
1   const exerciseDao = require('../model/dao/exerciseDao');
2
3   module.exports.showExercises = async (req, res) => {
4       const { type, limit } = req.query;
5       const exercises = await exerciseDao.getAllExercise(type, limit);
6       res.json({ exercises });
7   };
8
```

Рис. 3.11 Серверний контролер запитів для бази даних

Взаємодія користувацького інтерфейсу та серверу програмного продукту контролюється кастомізованим файлом з описаним запитом для Fetch API. Fetch API надає особливий спрощенний інтерфейс JavaScript для опрацювання і роботи з запитами та відповідями на ці запити HTTP. Глобальний метод fetch дозволяє досить легко отримувати ресурси через мережу асинхронно. Рисунок 3.12 є прикладом стандартного базового запиту для отримання масиву об’єктів у вигляді JSON-файлу.

```
export const getExercises = async (type, limit) => {
    const response = await fetch(`${ROOT}?type=${type}&limit=${limit}`);
    const { exercises } = await response.json();
    return exercises;
}
```

Рис. 3.12 Базовий fetch-запит на отримання ресурсів

Для підключення бази даних використовуються налаштування конфігурацій через JavaScript. Відповідний код наведено у додатку А.

3.3 Огляд роботи застосунку

Розроблюваний програмний застосунок є системою підтримки фітнес-тренувань шляхом надання користувачу можливості вибору бажаної програми тренування. Програма (перелік фізичних вправ) формується двома параметрами: типом тренування та рівнем складності.

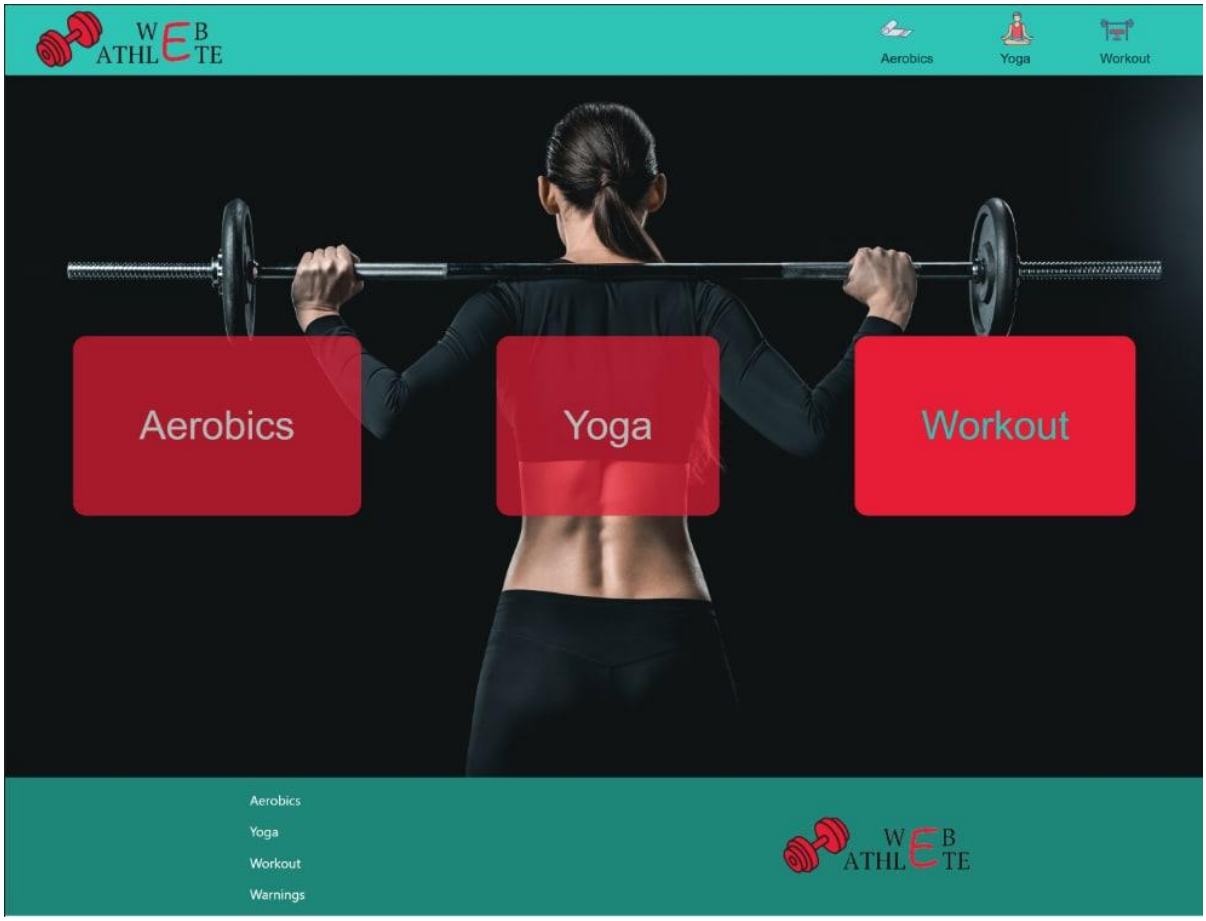

Рис. 3.13 Головна сторінка веб-застосунку з вибором типу тренування

Рівень складності представлений у трьох варіаціях – легкий (easy), середній (medium), hard(складний). Відповідно до рівня кожне тренування формується різною кількістю фізичних вправ. Поточна опція відображена у додатку Г.

Інтерфейс був розроблений відповідно до сучасних вимог, рекомендацій та визначених вимог на розробку. Головним аспектом було розроблення мінімалістичного та дружнього до користувача інтерфейсу. Одним з ключових переваг, які надають веб-застосункам, зокрема SPA, таку перевагу у побудові архітектури та подальшому використанню широкою аудиторією користувачів, є

можливість імплементації адаптивності для подальшої кросплатформності програмного продукту.

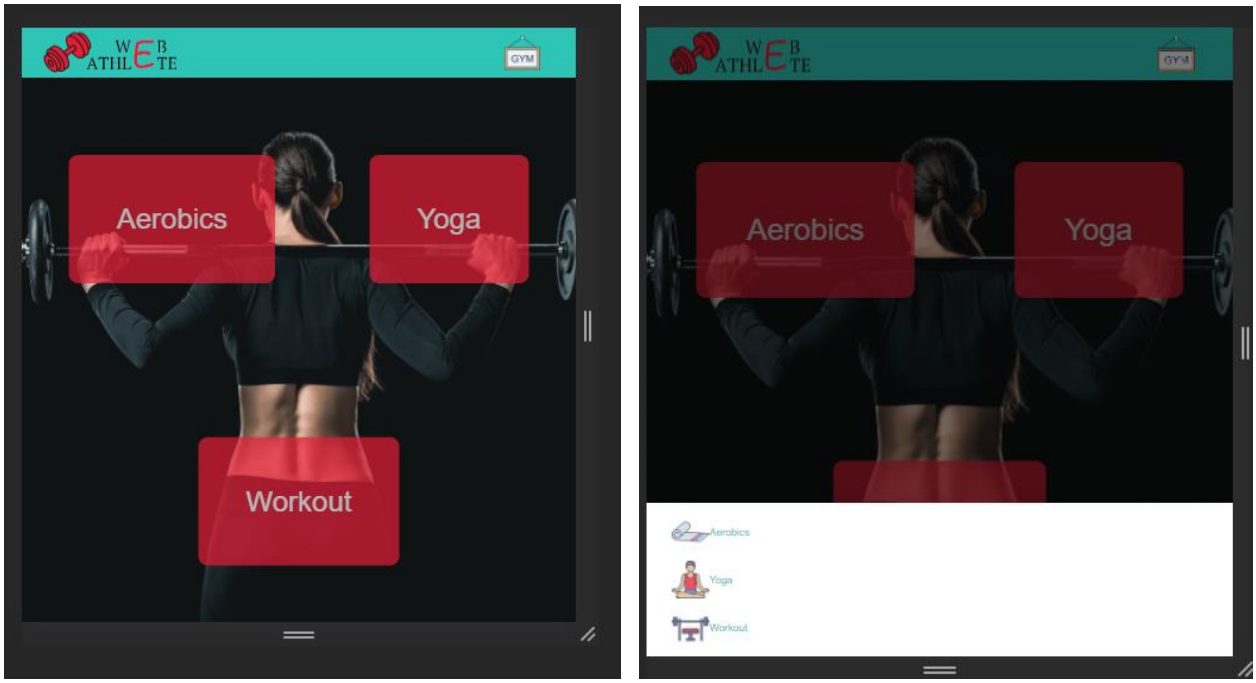

Рис. 3.14 Приклад адаптивності компоненту навігації

Висновки до розділу

Розробка веб-застосунку відповідно до сучасних методологій та концепцій включає велику кількість аспектів, поетапне виконання яких дозволяє досягнути поставленої мети бакалаврської роботи.

В загальному вигляді кожен програмний продукт включає користувацький інтерфейс, серверну частину та базу даних. Відповідно до визначеного стеку технологій розроблено кожен з модулів застосунку. Наслідуючи провідні концепції розробки, базу даних розміщено у хмарному сховищі.

Розроблено та описано макет загального вигляду та композитну діаграму роботи веб-застосунку.

Побудовано програмний продукт відповідно до сформованого завдання на розробку, наведено зображення головного екрану та прикладу адаптивності застосунку.

ВИСНОВКИ

Станом на 2021 рік світові системи охорони здоров’я працюють у посиленому режимі для боротьби з вірусною інфекцією. Проте, це не єдиний виклик, який потребує уваги кожної з держав для збереження власного населення. Наведено статистичні дані, що підтверджують необхідність посилення профілактичних заходів і популяризації фізичних навантажень, оскільки малорухомий спосіб життя провокує величезну кількість хронічних захворювань. Враховуючи обставини, визначено, що актуальним сучасним підходом є проведення тренувань в режимі онлайн.

Опрацьовано теоретичні відомості про популярність застосунків сфери «Здоров’я і спорт», їх органічний приріст та порівняння з попереднім роком. Проведено аналіз для розподілення застосунків на тематичні категорії: сервіси тренувань на базі «екосистем» виробників мобільних пристроїв, мобільні застосунки на платформах Android та iOS, онлайн-платформи з пошуку тренерів та застосунки-трекери для занять бігом. В кожній з категорій наведені приклади застосунків з описом ключових можливостей, переваг та недоліків кожного з них.

Враховуючи наведений опис, виділено, що більшість застосунків є мобільними, тому визначена мета розробки веб-застосунку є актуальною. Наведено три основних підходи у розробці веб-застосунків, проаналізовано переваги та недоліки кожного та виділено найбільш оптимальний для подальшого девелопменту програмного продукту зважаючи на визначені завдання на розробку.

Розробка сучасних веб-застосунків складає певний алгоритм. Першим етапом є вибір мов та технологій, з допомогою яких буде розроблюватись застосунок. Визначено, що ключовими інструментами розробки є HTML, CSS, JavaScript, база даних та система контролю версій.

Виділено базові характеристики кожної мови чи інструменту розробки, описано топ-3 найбільш використовуваних баз даних у сфері веб-програмування та обґрунтовано переваги використання систем контролю версій.

Проаналізовано найбільш популярні стеки технологій для веб-застосунків та особливості кожного окремого стеку. Наведено описову характеристику таких стеків MEAN, MERN, LAMP, Ruby on Rails Tech Stack.

Обрано та обґрунтовано стек технологій, бібліотек для розробки та побудови архітектури програмного застосунку. Окреслено переваги фундаментальних технологій обраного стеку технологій.

Загальна базова архітектура кожного програмного продукту включає фронтенд (користувацький інтерфейс), серверну частину та базу даних. Відповідно до обраного стеку MERN розроблено кожен з модулів застосунку. Наслідуючи провідні концепції розробки, базу даних розміщено у хмарному сховищі.

Побудовано програмний продукт відповідно до сформованого завдання на розробку, наведено зображення користувацького інтерфейсу та код одного з базових компонентів застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ВОЗ: COVID-19 бьет по системам здравоохранения - как смягчить удар. URL: https://news.un.org/ru/story/2020/08/1384832 (дата звернення: 28.02.2021).
2. Фадєєнко Г.Д., Колеснікова О.В. Основні стратегії профілактики неінфекційних захворювань в Україні. Раціональна фармакотерапія. 2017. Т.43, № 2. С. 5-8. URL: http://rpht.com.ua/ua-issue-article-1615 .
3. Малорухливий спосіб життя. URL: https://kdpu.edu.ua/press-centre/blogs/1145-kafedry/fizychnoi-kultury-ta-metodyky-ii-vykladannia/rekomendatsii-zavkafedry/10396-malorukhlyvyy-sposib-zhyttya.html (дата звернення: 28.02.2021).
4. Marianne Calnan. Almost three-quarters of employees ‘put in more effort’ when working from home. URL: https://www.peoplemanagement.co.uk/news/articles/employees-more-effort-working-from-home (дата звернення: 28.02.2021).
5. Bryan Robinson. Is Working Remote A Blessing Or Burden? Weighing The Pros And Cons. URL: https://www.forbes.com/sites/bryanrobinson/2020/06/19/is-working-remote-a-blessing-or-burden-weighing-the-pros-and-cons/?sh=2ec6292140a9 (дата звернення: 01.03.2021).
6. Shani Rosenfelder. 5 основных трендов, которые сформировали маркетинг мобильных приложений в 2020. URL: https://www.appsflyer.com/ru/blog/top-5-trends-app-marketing/ (дата звернення: 01.03.2021).
7. Amina Lake Abdelrahman, Selina Tedesco. The 20 Best Workout Apps You Should Download in 2021. URL:https://www.goodhousekeeping.com/health-products/g27112869/best-workout-apps/ (дата звернення: 01.03.2021).
8. Apple Fitness+. URL: https://www.apple.com/apple-fitness-plus/ (дата звернення: 01.03.2021).

9. Samsung Health. URL: https://www.samsung.com/ru/apps/samsung-health/ (дата звернення: 01.03.2021).
10. Beach Body on Demand. Programs for every body & schedule. URL: https://www.beachbodyondemand.com (дата звернення: 01.03.2021).
11. Personal Training. At Home or Outdoors. URL: https://findyourtrainer.com/ (дата звернення: 01.03.2021).
12. 7 приложений для бега, проверенных на личном опыте. URL: https://mondayrun.com.ua/running-apps/ (дата звернення: 01.03.2021).
13. Adidas Running от Runtastic и будь активнее вместе с нами. URL: https://www.runtastic.com/ru/ (дата звернення: 01.03.2021).
14. Веб приложение. Разница между сайтом, веб-приложением, spa и pwa. URL: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 02.03.2021).
15. Одностраничные (spa) и многостраничные (pwa) веб-приложения. URL: https://vc.ru/seo/108149-odnostranichnye-spa-i-mnogostranichnye-pwa-veb-prilozheniya (дата звернення: 03.03.2021).
16. Singh. Top 10 Web Development Frameworks of 2021. URL: https://www.techgeekbuzz.com/web-development-frameworks/ (дата звернення: 02.03.2021).
17. The Most Popular Databases 2019. URL: https://www.explore-group.com/blog/the-most-popular-databases-2019/bp46/ (дата звернення: 04.03.2021).
18. The Unified Modeling Language. URL: https://www.uml-diagrams.org/ (дата звернення: 01.03.2021).
19. Lindsay Kolowich Cox. Web Design 101: How HTML, CSS, and JavaScript Work. URL: https://blog.hubspot.com/marketing/web-design-html-css-javascript (дата звернення: 10.03.2021).
20. CSS Tutorial. URL: https://www.tutorialspoint.com/css/index.htm (дата звернення: 14.03.2021).

21. How CSS works. URL: https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps/How_CSS_works (дата звернення: 14.03.2021).
22. Как работает CSS изнутри. URL: https://seo24.kiev.ua/razrabotka/kak-rabotaet-css-iznutri/ (дата звернення: 15.03.2021).
23. Applications of Javascript Programming. URL: https://www.tutorialspoint.com/javascript/index.htm (дата звернення: 18.03.2021).
24. David Morales. The Most Commonly Used Databases for the Web. URL: https://davidmles.medium.com/the-most-commonly-used-databases-for-the-web-268666ae1070 (дата звернення: 19.03.2021).
25. Umesh Singh. Top 5 Database for Web Applications. URL: https://bestinterviewquestion.medium.com/top-5-database-for-web-applications-d71a4229fa37 (дата звернення: 20.03.2021).
26. О системах контроля версий. URL: https://habr.com/ru/company/otus/blog/521290/ (дата звернення: 24.03.2021).
27. Git. Матеріал з Вікіпедії — вільної енциклопедії. URL: https://uk.wikipedia.org/wiki/Git (дата звернення: 24.03.2021).
28. Sneha Das. Top 7 Web Development Technology Stacks for 2021. URL: https://dzone.com/articles/7-top-web-development-technology-stacks-for-2021 (дата звернення: 24.03.2021).
29. MERN Stack. URL: https://www.mongodb.com/mern-stack (дата звернення: 25.03.2021).
30. Nur Islam. The MERN stack: A complete tutorial. URL: https://blog.logrocket.com/mern-stack-tutorial/ (дата звернення: 28.03.2021).
31. MERN Stack. URL: https://www.geeksforgeeks.org/mern-stack/ (дата звернення: 04.04.2021).
32. Используем Mongoose для работы с Mongodb. URL: https://monsterlessons.com/project/lessons/ispolzuem-mongoose-dlya-raboty-s-mongodb (дата звернення: 05.04.2021).

33. Emil Lamprecht. The Difference Between UX And UI Design. URL: https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/ (дата звернення: 14.04.2021).
34. The super fast color schemes generator. URL: https://coolors.co/ (дата звернення: 17.04.2021).
35. React. JavaScript-библиотека для создания пользовательских интерфейсов. URL: https://ru.reactjs.org/ (дата звернення: 19.04.2021).
36. MATERIAL-UI. React компоненты для быстрой и легкой веб-разработки. URL: https://material-ui.com/ru/ (дата звернення: 10.04.2021).

ДОДАТКИ

Додаток А

Файл конфігурацій доступу до бази даних

app > server > index.js > ...

```
const cors = require('cors');

const express = require('express');
const morgan = require('morgan');
const mongoose = require('mongoose');

const exerciseRouter = require('./routes/exercise-router');

const PORT = 8080;

const app = express();
app.use(express.json({ limit: '2MB' }));
app.use(morgan('combined'));

app.use(cors());

app.use('/app', exerciseRouter);

const start = async () => {
    await mongoose.connect(
        'mongodb+srv://test_user:test_user@cluster0.s0nrs.mongodb.net/webAthlete?retryWrites=true&w=m
        {
            useNewUrlParser: true,
            useUnifiedTopology: true,
            useFindAndModify: false,
            useCreateIndex: true,
        }
    );
    app.listen(PORT);
};

start();
```

Додаток Б

Приклад коду компоненту Header

```
// Core
import React, { useState } from 'react';
import { Link } from "react-router-dom";
//Material-UI
import { makeStyles } from '@material-ui/core/styles';
import {
  AppBar,
  Toolbar,
  Typography,
  Hidden,
  Box,
  IconButton,
  Drawer,
  List,
  ListItem,
} from '@material-ui/core';
// Instruments
import routes from '../../navigation/routes';
// Assets
import logo from '../../assets/logo_no_bg.png';
import aerobics from '../../assets/aerobicsLogo.png';
import yoga from '../../assets/yogaLogo.png';
import workout from '../../assets/workoutLogo.png';
import gym from '../../assets/gym.png';
// Styles
import styles from './styles';

const useStyles = makeStyles((theme) => styles(theme));

const navbarPoint = (key, text, path, img) => {
  return { key, text, path, img };
};

const navbarList = [
  navbarPoint("aerobics", "Aerobics", `${routes.aerobics}`, aerobics),
  navbarPoint("yoga", "Yoga", `${routes.yoga}`, yoga),
  navbarPoint("workout", "Workout",`${routes.workout}`, workout),
];

export const Header = () => {
  const classes = useStyles();
  const [open, setOpen] = useState(false);

  return (<AppBar position="static" color="secondary">
```

продовження Додатку Б

```
      <Toolbar className={classes.toolBar}>
        <Link to={routes.home} className={classes.link}>
          <div>
            <img src={logo} alt="Web Athlete logo"/>
          </div>
        </Link>
        <Hidden smDown>
          <Box className={classes.barLinks}>
            {navbarList.map(({ key, text, path, img}) => (
              <ListItem key={key}>
                <Link to={path} className={classes.link}>
                {img && <img src={img} alt={text} height="40px" />}
                <Typography variant="body1" className={classes.bar_text}>
                  {text}
                </Typography>
                </Link>
              </ListItem>
            ))}
          </Box>
        </Hidden>
        <Hidden mdUp>
          <IconButton
            className={classes.link}
            onClick={() => setOpen(!open)}
          >
            <img src={gym} alt="" height="60px" />
          </IconButton>
          <Drawer
            anchor="bottom"
            open={open}
            onClose={() => setOpen(false)}
          >
            <List>
              {navbarList.map(({ key, text, path, img}) => (
                <ListItem key={key}>
                  <Link to={path} className={classes.link_mobile}>
                    <img src={img} alt="" height="60px" />
                    <Typography
                      variant="body1"
                    >
                      {text}
                    </Typography>
                  </Link>
                </ListItem>
              ))}
            </List>
          </Drawer>
```

Додаток В

Приклад коду опису стилів компоненту

```
const styles = (theme) => ({
    box: {
      paddingTop: theme.spacing(8),
    },
    appBar: {
      position: "fixed",
      overflow: "hidden",
      width: "100%",
      top: 0,
    },
    toolBar: {
      display: "flex",
      flexDirection: "row",
      justifyContent: "space-between",
      paddingLeft: "10px",
    },
    barLinks: {
      display: "flex",
      flexDirection: "row",
      justifyContent: "center",
      paddingLeft: "10px",
    },
    bar_text: {
      textDecoration: "none",
      color: theme.palette.textPrimary.main,
      marginLeft: theme.spacing(3),
      marginRight: theme.spacing(3),
      "&:hover,&:focus": {
        color: theme.palette.inherit.main,
      },

    },
    link: {
      textDecoration: "none",
      color: theme.palette.textPrimary.main,
      marginLeft: theme.spacing(3),
      marginRight: theme.spacing(3),
      "&:hover,&:focus": {
        color: theme.palette.inherit.main,
      },
    },
    link_mobile: {
      textDecoration: "none",
      color: theme.palette.secondary.main,
```

продовження Додатку В

```
    marginLeft: theme.spacing(3),
    marginRight: theme.spacing(3),
    display: "flex",
    flexDirection: "row",
    justifyContent: "center",
    alignItems: "center",
  },
});

export default styles;
```

Додаток Г

Інтерфейсне представлення вибору складності фізичних навантажень