КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теоретичної кібернетики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ З ВИКОРИСТАННЯМ NODE.JS ТА БІБЛІОТЕКИ REACT

Виконав студент 4-го курсу

Олег ГУБЕРНАТОР

(підпис)

Науковий керівник:

професор, кандидат фіз.-мат наук,

Юрій КРАК

(підпис)

Засвідчую, що в цій роботі нема запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики
«1» червня 2022р.
протокол № 11
Завідувач кафедри
Юрій КРАК

(підпис)

Київ — 2022

РЕФЕРАТ

Обсяг роботи: 40 сторінок, 23 ілюстрації, 26 використаних джерел

SPA, SINGLE-PAGE APLIKACE, JAVASCRIPT, ANGULAR, REACT, VUE, NODE.JS, EXPRESS, KOA.JS, WEB-APP.

Метою курсової роботи є створення клієнт-серверного програмного забезпечення яке буде давати змогу тренерам підготувувати плани тренувань для спортсменів, а ті в свою чергу зможуть прочитати ці плани та звітувати про процес тренувань. Для забезпечення роботи буде побудований веб-додаток на React із зручним користувацьким інтерфейсом та додатковими функціями.

Visual Studio Code був обраний як інструмент програмування – IDE розроблений компанією Microsoft для Windows, зручний для багатоплатформної розробки веб-застосунків. Редактор надає функції, пов’язані з Git, підсвічування синтаксису та інструменти для покращення коду.

Мова програмування JavaScript.

Методи розробки: клієнт-серверний підхід, створення front-end частини.

Результати роботи: було досліджено найпопулярніші фреймворки для створення фронтенду, платформу Node.js та її фреймворк Express. У ході виконання був побудований веб-додаток який допомагає покращити взаємодію тренера і спортсмена та покращити ефективність тренувань.

Можливості програмного продукту можуть бути розширені та доповнені в майбутньому. Користувачі отримають користь від веб-додатка, оскільки він дає щось нове та корисне як для тренера так і для його підопічного.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

Від англ. – розшифрування абревіатури на англійській мові;

HTML (від англ. Hypertext Markup Language) –це мова розмітки яка використовується для створення структури веб-сторінок

API (від англ. Application Programming Interface) – це набір протоколів взаємодії, підпрограм, та інструментів розробки програмного забезпечення

IDE (від англ. Integrated Development Environment) – інтегроване середовище розробки

JS – (від англ. Java Script) – динамічна, об'єктно-орієнтована прототипна мова програмування.

HTTP – (від англ. HyperText Transfer Protocol) –протокол передачі даних

DOM – (від англ. Document Object Model) – інтерфейс прикладного програмного забезпечення для взаємодії зі структурованими документами.

JSX – (від англ. JavaScript XML) – це розширення синтаксису JavaScript

JSON – (від англ. JavaScript Object Notation) – це формат тексту який використовується для обміна даними

SASS – (від англ. Syntactically Awesome Style Sheets) –метамова, яку інтерпретують в CSS

SPA – (від англ. Single Page Application) – це односторінковий застосунок

XML – (від англ. eXtensible Markup Language) – розширювана мова розмітки

CSS – (від англ. Cascading Style Sheets) – це спеціальна мова стилю сторінок

ВСТУП

Оцінка сучасного стану об’єкта розробки. Зараз існує великий вибір веб-додатків для різних видів спорту, наприклад для тренувань з плавання або вимірювання кількості кроків тощо. Ці програми мають різні налаштування і функції та допомагають тренерам і спортсменам покращувати свої результати.

Після ряду досліджень було виявлено, що прямих аналогів немає. Саме тоді з’явилася ідея веб-додатка для тренувань. Зараз існує велика кількість бібліотек для програмування інтерфейсів. У цьому дослідженні розглядається побудова веб-інтерфейсів за допомогою парадигми реактивного програмування, яка сьогодні є популярним методом розробки. І розробка серверної частини на асинхронній подієво-орієнтованої програмній платформі, яка значно підвищує продуктивність сервера за рахунок неблокування введення та виведення, а також спрощує розробку сервера.

Актуальність роботи полягає у відсутності необхідних фунцій веб-додатків для тренувань без потреби щось встановлювати.

Мета й завдання роботи. Метою роботи є розробка веб-додатку, який був би корисним для тренерів і спортсменів та простим для використання. Завдання для досягнення цієї мети:

- проаналізувати найбільш популярні фрейморки Angular, Vue і бібліотеку React для сторення SPA додатків
- проаналізувати розробку серверної частини на асинхронній подієво-орієнтованій програмній платформі;
- розробити технічне завдання;
- спроектувати та розробити клієнт-серверне програмне забезпечення.

Об’єктом дослідження є розробка веб-додатку для тренувань яке буде давати змогу тренерам підготувувати плани тренувань для спортсменів, а ті в свою чергу зможуть прочитати ці плани та звітувати про процес тренувань.

Предметом дослідження є технології створення клієнт-серверного

програмного забезпечення

Можливі сфери застосування. Цей веб-додаток для тренувань може застосовуватись як для самостійного використання спортсменом так і для використання в моделі тренер-спортсмен, можлива інтеграція в систему залу для використання в загальноспортзальній системі

РОЗДІЛ 1 ТЕХНОЛОГІЇ СТВОРЕННЯ ВЕБ-ДОДАТКІВ ТА ЇХ ОСОБЛИВОСТІ

1.1 SPA

Спроби перемістити десктопні програми почалися давно. Випробовувалися різні рішення та технології з різним ступенем успіху. Це були, наприклад, аплети Java, IFrames, Adobe Flash та інші. Хоча це були зовсім протилежні технології, у них була спільна мета – спроба створити платформену програму, схожу на настільну програму, яка працюватиме у браузері. Односторінкові програми також прагнуть до цієї мети, але з тією різницею, що використовуються лише технології, які підтримуються безпосередньо браузером, тобто без необхідності встановлення плагінів у браузер. Першою технологією, яка змінила наше уявлення про веб-додатки, був AJAX - асинхронна технологія JavaScript і XML. Ця технологія дозволила надсилати запити на сервер у фоновому режимі без перезавантаження сторінки. Ці дані, отримані від сервера, можуть бути вбудовані в об’єктну модель документа (DOM), тобто об’єктну модель, яка складає сторінку. Це дало змогу динамічно змінювати вміст сторінки.

1.1.1 Принцип SPA

В односторінковій програмі (SPA) вся веб-сторінка працює як одна програма. У цьому випадку інтерфейс та сервер програми, так звані front-end і back-end, повністю розділені, де SPA опікується клієнтською стороною.

При стандартному дизайні веб-сайту необхідність оновлення даних, що відображаються на сторінці, надсилається на сервер запитів, так званий реквест. Сервер обробляє цей запит, генерує отриманий HTML-код, який відправляється назад у браузер. Браузер відтворює цей код і відображає його користувачеві. Тому сервер також піклується про інтерфейс.

У випадку SPA інтерфейс розташований у браузері клієнта. Коли сторінка спочатку завантажується, інструменти, необхідні для візуалізації сторінок, завантажуються, і відображається перше представлення, так зване подання. Якщо потрібно створити нове представлення даних, наприклад, користувач натискає посилання, яке перенаправляє його в інше місце в програмі, це подання відображається локально та динамічно вставляється на сторінку DOM. Запит надсилається на сервер (якщо потрібно) під час відтворення нового подання. Дані зазвичай повертаються із сервера у форматі XML або JSON. Потім вони обробляються та додаються до подання.

Результатом є перенаправлення та відтворення нової частини сторінки з оновленими даними, без необхідності повного завантаження в браузері.

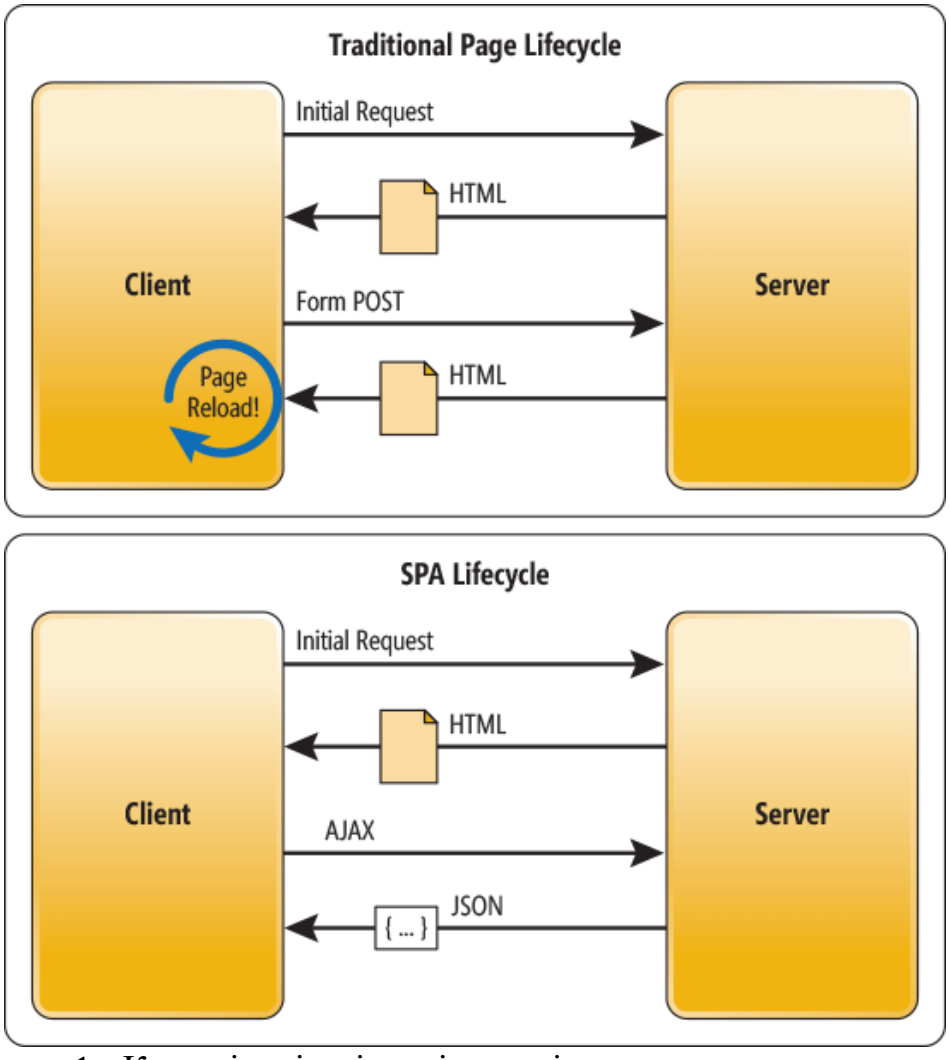

Рисунок 1 - Комунікація між клієнтом і сервером у стандартному та односторінковому додатку

1.1.2 Технології створення SPA

Мета SPA полягала в тому, щоб уникнути необхідності встановлювати плагіни в браузері, тому вони повинні запускатися в будь-якому браузері. Тому для створення SPA достатньо базових технологій, які також використовуються для створення стандартних сторінок – HTML, CSS та Javascript.

1.1.2.1 HTML

HTML – Hyper Text Markup Language. Ця мова розмітки використовується для створення структури веб-сторінок. Для створення цієї структури використовуються елементи HTML, які складаються з тегів. Теги можуть бути парними або непарними. Ми відокремлюємо окремі частини вмісту, наприклад заголовок, розділ таблиці тощо. Ці елементи потім відображаються браузером у графічний дисплей, який бачить користувач. На даний момент останньою версією є версія 5.2.

1.1.2.2 CSS

CSS – каскадні таблиці стилів, використовуються для стилізації елементів HTML. Вони працюють за принципом селекторів, за допомогою яких ми вибираємо потрібний елемент, а потім проектуємо його за допомогою окремих властивостей CSS, таких як font-size, для розміру тексту, або backgorund-color для кольору фону елемента. Сьогодні препроцесори CSS, такі як Sass або LESS, набувають все більшої популярності, додаючи до стандартної мови CSS нові функції, такі як використання змінних, успадкування та більш чітке позначення вкладених селекторів. Поточна версія CSS — версія 3.

1.1.2.3 Javascript

Javascript — це мова програмування сценаріїв, що використовується для створення веб-сайтів. Основним завданням JavaScript є визначення та динамічна зміна поведінки сторінки. За допомогою цієї мови ми можемо маніпулювати елементами HTML, змінювати їх розміри, положення, вміст або властивості CSS. Без JavaScript неможливо було б створювати інші, крім звичайних статичних веб-сайтів. Завдяки цій мові ми сьогодні можемо створювати дуже складні додатки, які неможливо впізнати від настільних. Javascript почався як суто клієнтська мова, сьогодні вже є технології, які дозволяють нам використовувати Javascript і технології Node.js для написання коду, який працює на сервері.

Перша версія мови була створена в 1997 році під офіційною назвою ECMAScript 1. Найбільш суттєві зміни відбулися в ECMAScript 5 (скорочено ES5), у 2009 році, а потім у ECMAScript 6 (ES6), що зробило JavaScript найпопулярнішою мовою. що ми сьогодні знаємо.

1.2 Огляд бібліотеки React

React — це бібліотека JavaScript, яка зосереджена на View в класичній моделі MVC. Вид влаштований як ієрархія компонентів. React сам по собі не є повноцінним фреймворком. Він стає повноцінним каркасом застосунку після додавання інших компонентів, які роблять його повною технологією створення SPA. Назва React походить від слова reactive, що вказує на філософію фреймворку — окремі компоненти реагують на поточний стан програми. Програма в цілому не підтримує свій стан, лише окремі компоненти підтримують свої стани. Якщо ми хочемо додати функціональність для підтримки глобального стану, потрібен зовнішній плагін. React також не надає таких речей, як маршрутизатор або службу HTTP-запитів. Користувач вирішує, що React підтримуватиме, які пакети встановити. Цей принцип

дозволяє зберегти загальний розмір програми невеликим, оскільки містить лише ту функціональність, яка дійсно потрібна програмісту.

1.2.1 Компоненти

Компоненти пропонують спосіб розділити UI на незалежний код для повторного використання. Концептуально компонент є функцією JavaScript, яка приймає вхідні параметри як props (скорочено від слова properties) і повертає елемент React, що описує те, що має бути відтворено в браузері.

Компоненти можна визначити в React як класи або функції. Компоненти пропонують більше функціональних можливостей, ніж класи. Щоб визначити клас як компонент React, він клас повинен розширити клас React.Component.

Кожен компонент підтримує власний локальний стан, що дозволяє нам зберігати, редагувати або видаляти властивості, збережені в компоненті. Зразок компонента в React може виглядати так.

```
class Example extends React.Component {
  constructor(props) {
  super(props);
  this.state = {date: new Date()};
  }
  render() {
  return (
  <div>
  {this.state.date.toLocaleTimeString()}
  </div>
  );
  }
 }
```

Створений клас прикладу розширює клас React.Component. Параметр props надсилається до конструктора компонента, який ініціалізує властивості компонента. Ми призначаємо початковий стан компонента об’єкту this.state, зокрема ми призначаємо поточну дату властивості date. Потім він визначає в методі render (), як ми хочемо, щоб користувач бачив цей стан у своєму браузері.

Як і компоненти Angular, компоненти React мають життєвий цикл.

- render(): єдиний обов'язковий метод в компоненті React. Викликається щоразу, коли компонент оновлюється.
- componentDidMount(): викликається один раз після відображення виводу компонента.
- componentWillUnmount(): викликається один раз перед видаленням компонента з DOM.

1.2.2 JSX

JSX (скорочене розширення JavaScript) є рекомендованою технологією опису дизайну в React. Це розширення синтаксису JavaScript. React не має філософії поділу технологій на власні файли, як у випадку з Angular, де Typescript і Template записуються в HTML у двох окремих файлах. Весь код, включаючи JSX, записується безпосередньо в компонент React. На перший погляд він разюче нагадує мову HTML. Однак у результаті JSX компілюється в код JavaScript. Як приклад, ми можемо перерахувати заголовок у методі render() у компоненті.

```
<h1 className='large'>Hello World</h1>
```

В результаті цей код буде перекладено на JavaScript як

```
React.createElement(
  'h1',
  {className: 'large'},
  'Hello World'
  )
```

Використання JSX не є необхідним , можна написати безпосередньо код JavaScript. Однак використання JSX значно підвищує чіткість коду та зменшує складність навчання, оскільки, ймовірно, більшість веб-програмістів вже стикалися з HTML-кодом, на який JSX дуже схожий.

1.2.3 Redux

Redux — це бібліотека JavaScript з відкритим вихідним кодом, яка забезпечує глобальне керування здоров'ям додатків. Redux забезпечує односторонній потік даних і дотримується кількох принципів.

- Програма має глобальний стан
- Зміна цього стану ініціює оновлення компонентів, на які впливає
- Лише деякі функції можуть змінювати стан
- Ці функції можна виконувати лише користувачем
- Завжди існує лише одна зміна стану.

Цей глобальний стан не може ініціювати далі дії самі по собі, лише вхідні дані. Це значно полегшує підтримання цього стану та запобігає його небажаним змінам.

У цьому глобальному стані, наприклад, можуть зберігатися відповіді сервера, кешовані дані або дані, створені локально. Redux дозволяє вам маніпулювати та підтримувати ці дані.

1.2.4 Віртуальний DOM

Використовується Virtual DOM - кешована версія DOM, яку вона зберігає в пам'яті пристрою, на якому вона працює. Бібліотека ReactDOM піклується про своєчасність цього подання в пам'яті з відтвореним виглядом. Перевага цього принципу в тому, що вони завжди оновлюються лише ті частини сторінки/компонента, які необхідні, тобто змінився їх статус. Це позбавляє від необхідності повторно відображати всю сторінку.

1.2.5 Застосування стилів до компонентів

React дає велику свободу у виборі стилю для окремих компонентів. Перший варіант — створити файл, який містить код стилю, а потім імпортувати в компонент. Цей файл може бути типу CSS, де ми

використовуємо класичний CSS код, або можна використовувати будь-який з препроцесорів, наприклад LESS або SASS.

Інший варіант — створити стилі як об’єкт JavaScript, а потім призначити стилів з цього об’єкта безпосередньо в шаблон JSX. Це рішення приносить користь, наприклад ми контролюємо, де які стилі використовуються за кількістю посилань на об’єкт в якому розташовані стилі, або прості зміни стилю на основі значення змінних у компоненті. Однак у цій програмі ми також втрачаємо переваги препроцесорів, таких як змінні CSS або композиція окремих селекторів.

1.3 Порівняння фреймворків

Для порівняння фреймворків були розроблені прості версії мого додатку з однаковим функціоналом. Завдяки цьому можна чітко побачити, як однотипні задачі вирішуються в окремих технологіях. Це дозволяє порівнювати об’єктивні властивості, такі як швидкість завантаження окремих програм або їх результуючий розмір, який вони будуть займати на сервері.

1.3.1 Порівняння розмірів додатків

Спочатку потрібно створити локальну версію програми. У цьому режимі доступні деякі функції, які надлишково збільшують кінцевий розмір програми або можуть становити загрозу безпеці. Це, наприклад, функціональність Source Maps. Ці файли відображають отриманий код JavaScript у вихідні файли програми. Це дає можливість налагоджувати програму безпосередньо в браузері. Ще одна функція — це автоматичне оновлення представлення у браузері після збереження змін у коді. Це позбавляє від необхідності оновлювати сторінку вручну після кожної зміни.

Продуктова збірка вилучає цю функціональність. Через видалення source maps у браузері доступний лише мінімізований код JavaScript і CSS. Основне завдання - зменшити отримані файли і підвищити його

продуктивність. Однак вони мають своє виключення, коли розмір програми не є проблемою. Бувають ситуації, коли якась частина програми не працює лише в певному середовищі, наприклад, у продакшені. Без них налагодження практично неможливе. Тому зберігання вихідних карт має оцінюватися для кожної ситуації окремо.

1.3.1.1 Angular

Щоб створити робочу збірку програми, необхідно запустити команду «ng build –prod» за допомогою терміналу в папці проекту. Крім створення продуктового пакету, ця збірка перевіряє програму більш детально, ніж у режимі розробки. Якщо у додатку є помилки, які проходять через робочу збірку, але через продуктову не пройдуть. Наприклад, виклик неіснуючої функції або виклик функції з неправильною кількістю параметрів із шаблону компонента. Причиною пропуску цього поглибленого контролю в режимі розробника є прискорення окремих збірок. Продуктові збірки займають на порядок більше часу і значно уповільнюють процес розробки програми. Після успішного завершення збірки в папці проекту створюється нова папка, яка називається dist. У ньому знаходиться отриманий проект, готовий для відправки на сервер або хостинг. Якщо ви хочете включити вихідні карти до робочої збірки, ви повинні ввімкнути цей параметр у файлі angular.json. Остаточний розмір програми, створеної в цій роботі без вихідних карт, становить 1,2 МБ. Якщо ввімкнути source maps, розмір збільшиться до 7,63 МБ

1.3.1.2 Vue

Початок розробки в технології Vue здійснюється за допомогою команди «npm run build». Ця команда створює папку dist, вміст якої готовий до хостингу. Однак за замовчуванням ця збірка автоматично додає source maps до

```
module.exports = { productionSourceMap: false
};
```

збірки. Тому їх потрібно відключати вручну. Для цього вам потрібно створити новий файл у кореневому каталозі проекту під назвою vue.config.js. Крім усього іншого, можна вимкнути генерацію цих карт.

Кінцевий розмір виробничого пакету становить 5,75 МБ. Після пропуску створення вихідної карти розмір впав до 1,21 МБ.

1.3.1.3 React

React має ідентичний процес створення локального пакету. React автоматично створює вихідні карти і навіть не має офіційного способу видалити їх зі збірки. Найпростіший варіант, навіть рекомендований розробниками, — це просто видалити їх після збірки. Пакет із source maps має розмір 3,23 МБ, без них зменшується до 656 КБ

1.3.1.4 Порівняння результатів

Порівнюючи результати, ми бачимо, що Angular найгірший у цьому порівнянні. Для пакетів без карт це те саме з Vue, але він створює набагато більші вихідні карти. Найменшу програму створює фреймворк React в обох випадках, з картами і без них

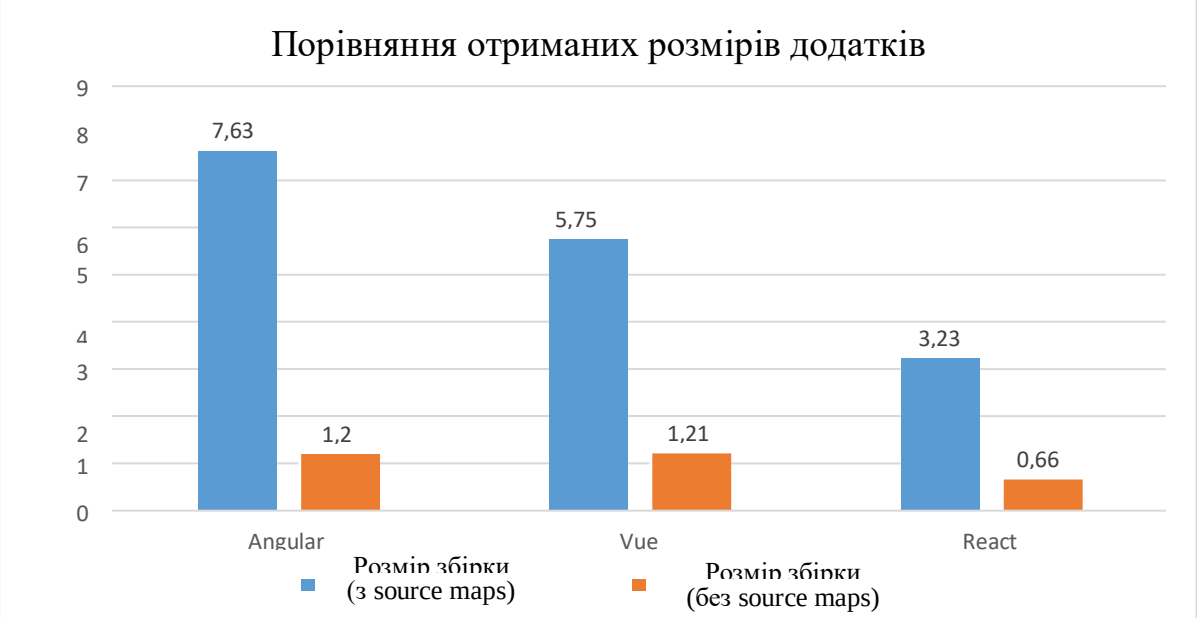

Рисунок 2 - Отриманні розміри програми

1.3.2 Порівняння швидкості додатків

Я буду використовувати інструменти консолі розробника, надані Google Chrome, щоб порівняти швидкість. Вкладка «Performance» дає змогу записувати хід завантаження програми та розбивку тривалості кожного кроку.

Усі тести проводилися локально на одному комп'ютері, а програми були підключені до одного локального сервера. Ці кроки звели до мінімуму зовнішні чинники, які можуть вплинути на результати. Через те, що це односторінкові програми, вони завантажуються повністю, щойно вони вперше потрапляють на сторінку. Цю функціональність можна обійти, а початковий розмір сторінки зменшити за допомогою функції, яка називається «lazy loading». Використовуючи цю техніку, окремі сторінки програми завантажуються лише після того, як користувач потрапляє до заданої частини. Однак це відбувається у фоновому режимі, і сторінка в браузері не оновлюється.

Ідеальним варіантом для визначення швидкості програми є завантаження домашньої сторінки програми після оновлення сторінки браузера. Це перевіряє швидкість завантаження, візуалізації та основних функцій, таких як створення запиту API та обробка цих даних. Він містить найбільшу кількість дочірніх компонентів. Ця сторінка також майже не містить частин із зовнішніх бібліотек, тому продуктивність цих доповнень не вплине на результати. Вкладка «Performance» містить можливість почати запис продуктивності програми та автоматично оновити сторінку. Під час цього тесту важливо вимкнути кешування, що може спотворити результати.

Однак слід додати, що це лише початкове завантаження програми, яке користувач відчує лише один раз під час свого відвідування або при подальшому оновленні сторінки. В якості іншого тесту ми можемо використовувати вкладку Audit, яка містить інструмент Lighthouse, який виконує комплексний аналіз сайту з кількох точок зору, наприклад,

продуктивності. Цей тест автоматизований, що виключає втручання людини, робить його більш точним.

1.3.2.1 Angular

Додаток, написаний на Angular, був першим протестованим. Використовуючи доступні інструменти, ми можемо додатково обмежувати діапазон вимірювань, щоб він не містив зайвого часу, коли під час вимірювання нічого не відбувалося. В основному це час між завершенням завантаження сторінки та вимкненням швидкого завантаження вручну.

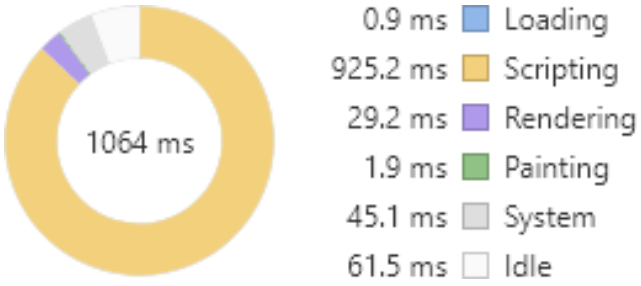

Рисунок 3 - Результати тесту продуктивності програми в технології Angular.

Завантаження сторінки зайняло майже одну секунду. Найдовше зайняло сценарії. Цей елемент в основному містить обробку браузером файлу main.js. Він містить мінімований код всієї програми.

Рисунок 4 - Результат аудиту програми Angular

Відповідно до інструменту Lighthouse, після оновлення сторінки програма відтворюється через 1,4 секунди

1.3.2.2 Vue

Як іншу програму, я тестував програму у фреймворку Vue

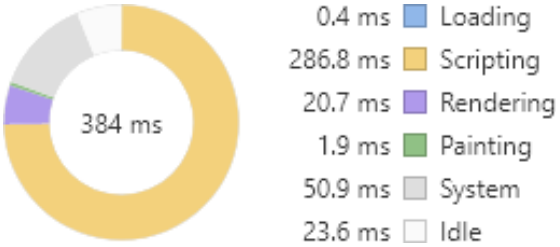

Рисунок 5 - Результати тесту продуктивності програми Vue.

Завантаження програми у фреймворк зайняло менше 400 мс. Це в основному швидше, ніж у попередньому випадку. У цьому випадку скрипт зайняв 287мс

Рисунок 6 - Результат аудиту програми Vue.

Попереднє вимірювання також підтверджується аудитом, який виміряв перший рендеринг і зручність використання програми через 0,7 секунди.

1.3.2.3 React

Останнім тестованим додатком був додаток React.

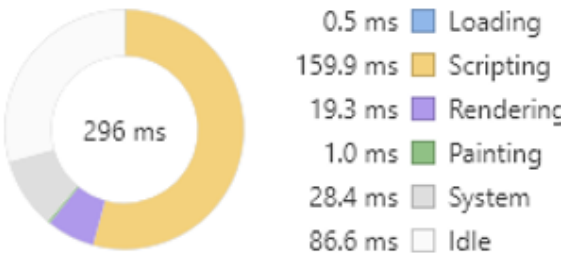

Рисунок 7 - Результати тесту продуктивності програми React

Технологія React справляється з найшвидшим завантаженням. Скрипти тривали менше ніж 160 мс, що менше третини в порівнянні з Angular.

Рисунок 8 - Результат аудиту програми React

Згідно з аудитом, початкове завантаження та отримання займає 0,6 секунди. Цей результат узгоджується з попередніми вимірюваннями, де ручне вимірювання продуктивності було приблизно на 100 мс швидше, ніж програми Vue.

1.3.3 Порівняння зручності та складності розробки

Фреймворки жодним чином не обмежують зручність використання. Ці технології не обмежують то, що можна створити, і я не зіткнувся з обмеженнями в будь-якому з фреймворків при розробці додатків.

З точки зору об’єктивних показників, таких як розмір і швидкість програми, фреймворк React найбільш хороший. Він стабільно тримав лідерство у всіх випробуваннях, які були зроблені.

Складність розробки багато в чому суб'єктивна. Це залежить від уподобань і попереднього досвіду розробника. Я вважаю, що фреймворк React є простою і хорошою технологією для вивчення. Розробка в рамках цієї технології інтуїтивно зрозуміла, в ній використовуються звичайні технології, з якими вже знайомі багато розробників. React можна розглядати як перетин Vue і Angular, від обох технологій він запозичує певну функціональність, зберігаючи при цьому невеликий загальний розмір і високу швидкість.

Angular є найбільш повним і надійним рішенням. Відразу після встановлення він готовий до створення програми без необхідності встановлення додаткових плагінів та бібліотек. Він також найсуворіше дотримується архітектури і у розробника не так багато варіантів вирішення певного типу задач, як у випадку з React. У цієї системи є свої переваги і недоліки. Основним недоліком є те, що рішення, яке віддає перевагу Angular, не завжди може бути найоптимальнішим. Це також видно з проведених у цій роботі тестів. Однак значення недоліку більш тривалого початкового навантаження зменшується з розміром проекту. Якщо розробник приходить до добре налагодженого проекту Angular, він може швидко освоїти цей проект. Тому я вважаю найбільш підходящим типом проекту для цієї технології великі та складні проекти, над якими працюють великі команди розробників з різним досвідом.

React дає розробнику вільний вибір для вирішення всіх проблем. Структура є модульною, окремі проекти можна об’єднати та спроектувати так, щоб вони були максимально ефективними. Ця свобода та модульність означає, що технологія сильно залежить від початкової архітектури та вибору правильних модулів програми, теоретично кожен проект, написаний на React, може виглядати сильно по-різному. Це може призвести до проблем із залученням нових членів до команди розробників. React підходить як технологія для проектів, над якими працюють розробники головною метою яких є досягнення максимальної продуктивності та ефективності.

1.4 Огляд платформи Node.js та її фрейморку Express.js

1.4.1 Node.js

Node.js — це платформа, розроблена для написання високомасштабованих веб-серверів. Використовує керовану подіями, неблокуючу модель введення/виводу, що робить її придатною для додатків реального часу з інтенсивним використанням даних, що працюють на

розподілених пристроях. Node.js побудовано на механізмі візуалізації, який використовує мову програмування JavaScript. У результаті програми, написані на цій платформі, використовують подібний синтаксис, що й інтерфейсні програми JavaScript, включаючи об’єкти та функції

Особливості:

- Асинхронна і керована подіями – вся бібліотека API Node.js є асинхронна, тому неблокуюча. В принципі, це означає, що сервер ніколи не чекає повернення даних API. Весь вхід/вихід таких операції, як підключення до бази даних або робота з файлами не блокують основний потік, а продовжуються, лише коли отримують дані від іншої сторони.
- JavaScript – JavaScript подібний синтаксис Node.js надає опцію мати як інтерфейсний додаток, так і бекенд на одній мові програмування. Може не бути чергування між технологіями, також можна використовувати той самий код або бібліотеки в усій програмі.

Node.js від інших середовищ і серверів відрізняє те, як він реагує на вхідні події. Більшість традиційних серверів (наприклад, запис файлів або запити до бази даних) сприймають виклики як блокування. Таким чином, програма переривається і чекає, поки виклик поверне результат, і таким чином завершується. Після цієї події програма знову продовжується. Поточні сервери вирішують цю проблему, створюючи потік для кожної події, що відбувається, але це рішення не є оптимальним, якщо мова йде про програму для кількох користувачів. Node.js вирішує цю проблему по-різному.

Він обробляє кожен запит з одного потоку. Код, що виконується в цьому потоці, виконується синхронно, але щоразу, коли виникає новий запит, запит буде переміщено до циклу подій разом із функцією зворотного виклику. Основний потік не сплячий і може продовжувати обробляти запити. Після завершення попереднього запиту цикл подій запускає функцію зворотного виклику.

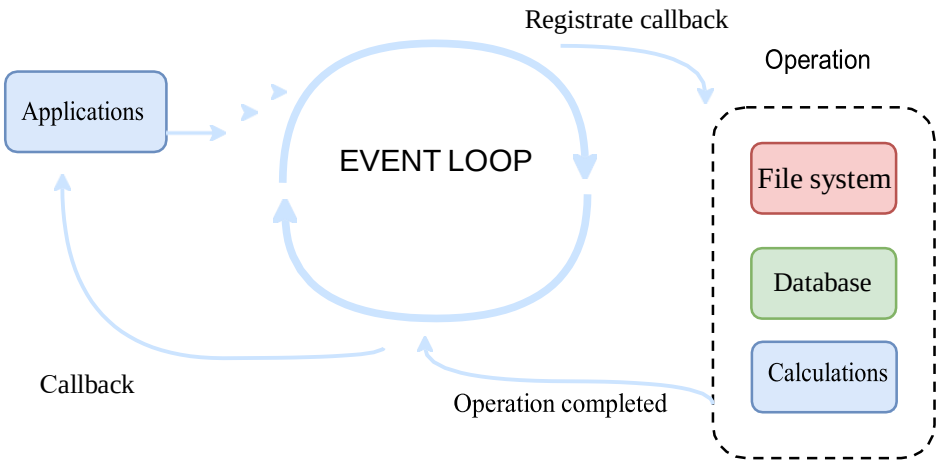

Рисунок 9 - Цикл подій

NPM або Node Package Manager використовується для поширення коду JavaScript серед розробників. Такі коди називаються packages або іноді модулями. Якщо в проекті використовується такий «чужий» пакет, то за допомогою NPM дуже легко перевірити, чи доступні якісь оновлення, і, якщо так, оновити його.

Packages — це в основному каталог, який містить один або кілька файлів, а також файл package.json з інформацією про пакет. Основна ідея npm полягає в тому, що з цих маленьких блоків, які завжди вирішують одну проблему, можна побудувати велику кількість функціональних можливостей програми.

1.4.2 Express.js

Express.js — це веб-фреймворк, який використовується для розробки серверної програми. Сам Node.js спочатку не був розроблений для створення веб-сайтів і тому не містить достатньої функціональності для цієї потреби. Express.js надає це як так зване проміжне програмне забезпечення. Як правило, це програмне забезпечення, яке надає функції, які зазвичай не передбачені оригінальною платформою. Наприклад, це вже створені компоненти JavaScript. Вони дозволяють розробникам використовувати вже створений і перевірений код для свого проекту, що вирішує більшість основних проблем,

таких як аналіз HHTP-запитів або файлів cookie5, створення маршрутів (англійських маршрутів), видалення параметра з URL-адреси, обробка помилок тощо. Express заснований на модулі http в Node.js і компонентах фреймворка Connect.js.

Проміжне програмне забезпечення (middleware) – це функції, які використовуються, наприклад, для виконання коду, для внесення змін у відповіді та об’єкти запиту, для завершення циклу виконання іншого проміжного програмного забезпечення (тобто для запобігання виконання даної функції функції next ()) або продовжити виконання функцій зі стека. Крім об’єктів response (res) і request (req), вони також мають доступ до функції next (), яка використовується для запуску іншого проміжного програмного забезпечення, якщо це необхідно. Кожного разу, коли надсилається запит HTTP, виконуються всі функції проміжного програмного забезпечення, які містяться в так званому стеку функцій проміжного програмного забезпечення. Проміжне програмне забезпечення можна підключити до програми за допомогою методів app.use () або app.GET і app.POST.

Програма в Express.js зазвичай складається з наступних послідовних частин:

- Вставка залежностей або. сторонні бібліотеки
- Створення об’єкта Express.js
- Налаштування програми, встановлення змінних програми
- Підключення до бази даних
- Визначення функцій проміжного програмного забезпечення
- Створення маршрутів
- Запуск самої програми, встановлення порту та імені

1.5 Порівняння та переваги Express з іншими фреймворками

Ми дослідимо відмінності між двома найпопулярнішими фреймворками в Node.js: Koa та Express.

Express - є мінімальною та гнучкою структурою веб-додатків Node.js, яка забезпечує надійний набір функцій для веб та мобільних додатків, вона є проміжним програмним забезпеченням для управління серверами та маршрутизацією.

Express - це, мабуть, найпопулярніший фреймворк для Node.js, і існує безліч інших популярних фреймворків, побудованих на Express.

Переваги:

- Майже стандарт для веб-проміжного програмного забезпечення Node.js.
- Простий, мінімалістичний, гнучкий та масштабований.
- Швидка розробка додатків.
- Повністю настроюваний.
- Проста інтеграція сторонніх служб та проміжного програмного забезпечення.
- Переважно зосереджений на браузерах, що робить шаблонування та рендеринг майже з коробки.
- Велика Express спільнота

Недоліки

Незважаючи на те, що Express.js є дуже зручним і простим у використанні фреймворком, він має деякі незначні недоліки, які можуть вплинути на процес розробки.

- Організація повинна бути дуже чіткою, щоб уникнути проблем при підтримці коду.
- У міру збільшення розміру коду рефакторинг стає дуже складним .
- Потрібна велика кількість ручної праці, оскільки потрібно створити всі кінцеві точки.

Koa була розроблена тією ж командою, яка стоїть і за Express, вона має на меті стати меншою, виразнішою та надійнішою основою для веб-додатків та API. Використовуючи функції асинхронізації, Koa дозволяє відмовитися від зворотних викликів і значно збільшити обробку помилок. Koa не поєднує жодних проміжних програм у своєму ядрі, і він пропонує елегантний набір методів, які роблять написання серверів швидким та приємним.

Додаток Koa - це об’єкт, що містить масив функцій проміжного програмного забезпечення, які складаються та виконуються у формі стека за запитом.

Переваги

- Koa покращує взаємодію, надійність та робить написання проміжного програмного забезпечення набагато приємнішим.
- Має велику кількість корисних методів, але зберігає невеликий розмір, оскільки ніяке проміжне програмне забезпечення не входить у комплект.
- Koa дуже легкий, лише 550 рядків коду.
- Має дуже хороший користувацький досвід.
- Краща обробка помилок за допомогою функції try / catch.
- Поток керування на основі згенерованих данних.
- Більш чистіший та читабельніший асинхронний код.

Недоліки

- Новий фреймворк, тому пільнота навколо Koa порівняно невелика.
- Не сумісний із проміжним програмним забезпеченням у стилі Express.
- Koa використовує генератори, які не сумісні з будь-яким іншим типом проміжного програмного забезпечення Node.js.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ДОДАТКУ

2.1. Постановка задач

1. Проектування клієнт-серверної частини веб-додатку
2. Розробка серверної частини
3. Розробка інтерфейсу користувача
4. Налагодження взаємодії клієнт-сервер

Метою роботи є розробка веб-додатку, яка надасть спортсменам і тренерам базові корисні функції та приємний інтерфейс. При реалізації використовуються сучасні технології для розробки, що сприяє легкому масштабуванню та підтримці сервісу.

2.2. Загальна структура та технології веб-додатку.

Розроблюваний веб-додаток складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних.

Клієнтська частина веб-додатку – це інтерфейс для взаємодії користувача з сайтом, клієнтська частина також формує запити для взаємодії з сервером. При розробці сервісу для тренувань, після порівняння фреймворків, для програмування інтерфейсу була обрана бібліотека React.

Серверна частина – це процеси на сервері для обробки запитів користувача та взаємодії з даними.

База даних - програмне забезпечення на стороні сервера, яке зберігає та отримує дані у відповідний момент. Вона зберігається на сервері зберігається.

Серверна частина веб-додатка звертається до бази даних, витягуючи дані, які необхідні для формування сторінки, запитаної користувачем.

Функції веб-додатку були розроблені на основі аналізу існуючих спортивних веб-додатків, та спілкування з можлививими користувачами додатку. Як людина яка використовує спортивні додатки, можу сказати що це найкращий спосіб знайти практичні можливості для будь-якого програмного

забезпечення. У наступному розділі наведено функції мого веб-додатка. Тренер і спортсмен мають власний інтерфейс у цьому веб-додатку на основі своїх вимог. У тренера є інформаційну панель для відстеження інформації про кожного зі своїх спортсменів, тобто: персональні дані, статистика тренувань, коментар та результати кожного тренування. Також для кожного спортсмена підготовлений календар, куди тренер може додавати нові тренування та відстежувати результати тренувань. Інтерфейс спортсмена містить інформаційну панель з інформацією про свої тренування, статистику тренування та більш детальний опис кожного тренування, методику виконання. Інтерфейс для спортсменів також пропонує календар із тренуваннями для їх відстеження та звітності. На основі дослідження існуючих технологій створення веб-сервісів було обрано для клієнт-серверної комунікації – архітектуру REST, для створення клієнтської частини веб-додатка – JavaScript бібліотеку React, для створення серверної частини веб-додаток - платформа Node.js і фреймворк Express.js, для зберігання інформації про користувачів, вправи, плана тренувань, списка друзів була використана реляційна система керування базами даних SQLite.

Для реалізації серверної частини була використана платформа Node.js та її фрейморк Express, так як вона є цікавою для вивчення та показує високу продуктивність виконання.

Для ефективної розробки веб-додатку необхідно притримуватися певних принципів, зокрема елементи інтерфейсу повинні бути організованими, об’єднаними у групи логічно пов’язаних елементів, вирівняними, оформленими у єдиному стилі, і, крім того, обов’язковим є наявність вільного простору для розмежування інформаційних блоків, щоб зосередити увагу користувача на чомусь одному. Якщо інтерфейс розроблений за всіма правилами, то це значно підвищує ефективність ресурсу і робить його конкурентноспроможним.

Реалізація веб-інтерфейсу системи була реалізована як під мобільні пристрої, так і десктпоні з адаптацією під будь-який екран. В якості шрифту був використаний Roboto – шрифтова гарнітура без засічок від Google, з базовим розміром у 16 пікселів. Щоб користувачеві було комфортно, при створенні дизайну додатку було використано не надто яскраві кольори.

Структура сайту має такі сторінки: головна, вхід/реєстрація, персональна сторінка, календар тренувань, сторінка з клієнтами, план тренувань, програми тренувань, перелік вправ, статистика, підтримка

РОЗДІЛ 3 МОЖЛИВОСТІ ПРОГРАМНОГО ПРОДУКТУ

3.1. Концепція

Користувачі цього веб-додатка поділяються на тренерів і спортсменів. Кожен користувач має доступ до певних функцій і сторінок веб-додатка. Метою роботи було розроблення та впровадження практичного веб-додатку для планування спортивних тренувань, де тренери та спортсмени мають інтерфейс відповідно до своїх потреб.

3.2. Процес роботи веб-додатку

Першою сторінкою, яка відображається користувачеві після запуску програми, є головна сторінка (Рис.10).

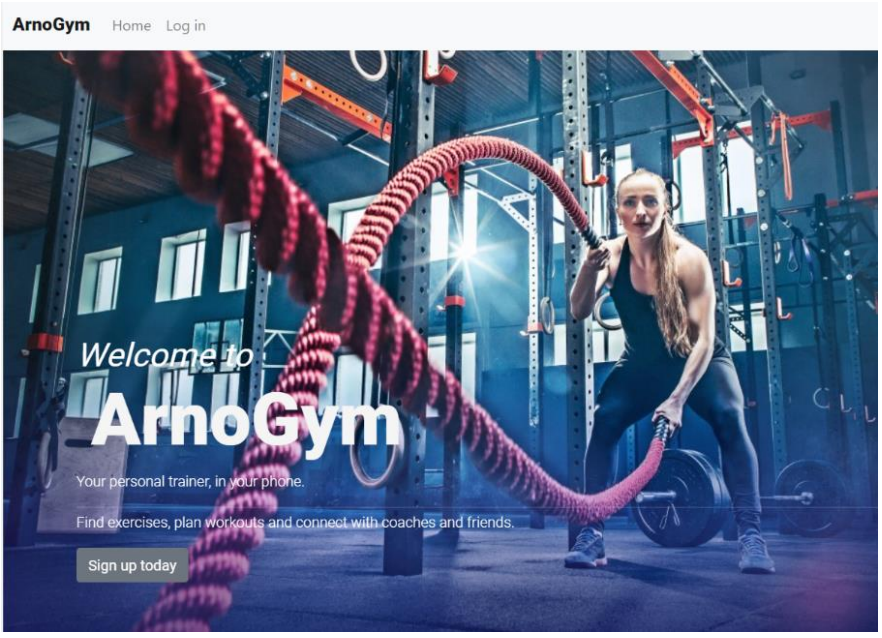

Рисунок 10 – Головна сторінка

Далі є два можливих варінти для нового користувача натиснути на «Sign up today» і зареєструватися або для постійного користувача натиснути на «Log in» та зайти до програми через сторінку входу/реєстрації (Рис.11). Вона дозволяє отримати доступ до веб-програми, ввівши адресу електронної пошти та пароль. У разі збігу даних користувача сервер формує веб-токен JSON, який

є стандартом, що визначає метод безпечної передачі інформації між клієнтом і сервером у вигляді об’єкта JSON. Сервер надсилає клієнту повідомлення разом з інформацією про користувача. Якщо користувача буде перенаправлено на інший сайт, програма розуміє, що він є авторизованим користувачем, і не потребує повторного входу.

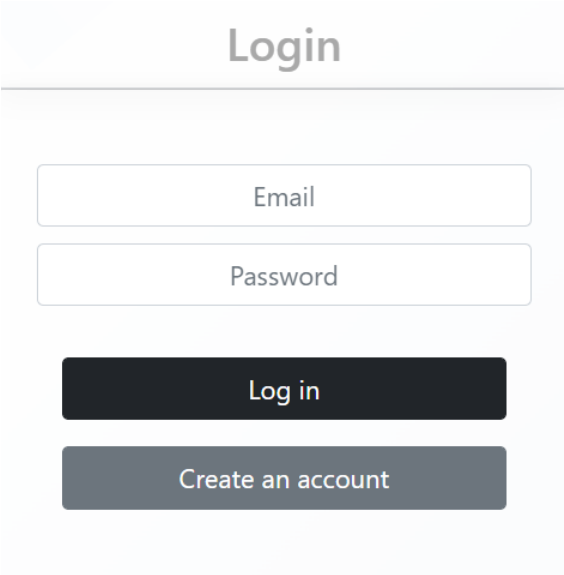

Рисунок 11 – Форма авторизації

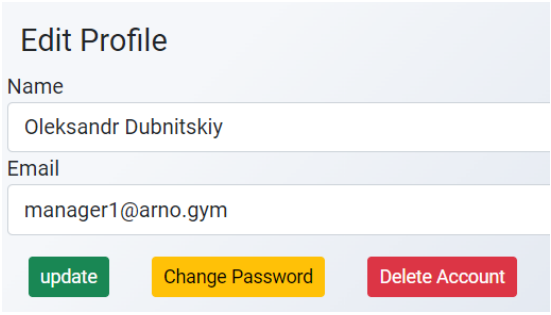

Рисунок 13 – Редагування профілю

Після входу ми потрапляємо до особистого кабінету в якості тренера або спортсмена (Рис.12). Тут можна відредагувати профіль, змінити пароль або взагалі видалити аккаунт (Рис.13)

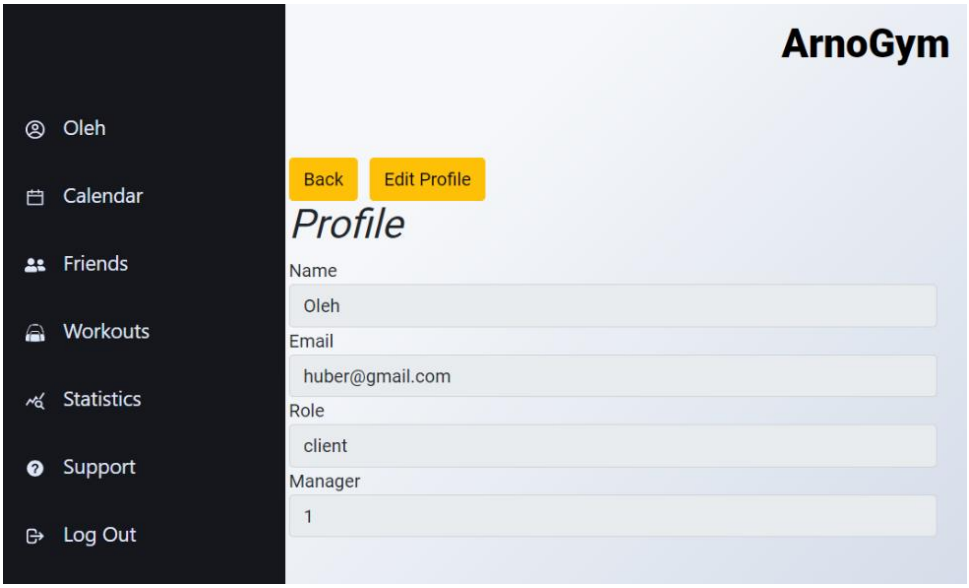

Рисунок 12 – Аккаунт спортсмена

Тренер має профіль кожного спортсмена (Рис.14), який відображає особисту інформацію спортсмена - ім'я, прізвище, адресу електронної пошти та можливість активувати/деактивувати його профіль.

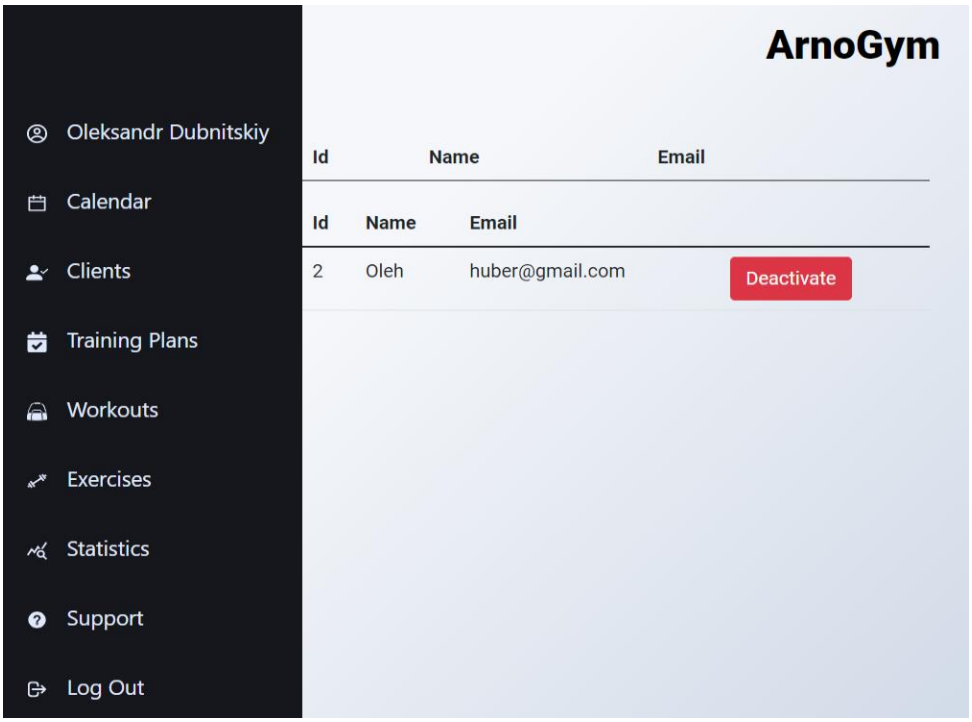

Рисунок 14 – Сторінка підопічних спортсменів

Ще одна цікава функція – «Calendar». Тренер може відстежувати виконання тренуваннь, статистику тренувань, кількість завершених і запланованих тренувань кожного з спортсменів в їх персональних календарях (Рис.15). Спортсмен завдяки цій функції має можливість відстежувати власну щотижневу статистику тренувань, а також переглядати підготовлені та завершені тренування.

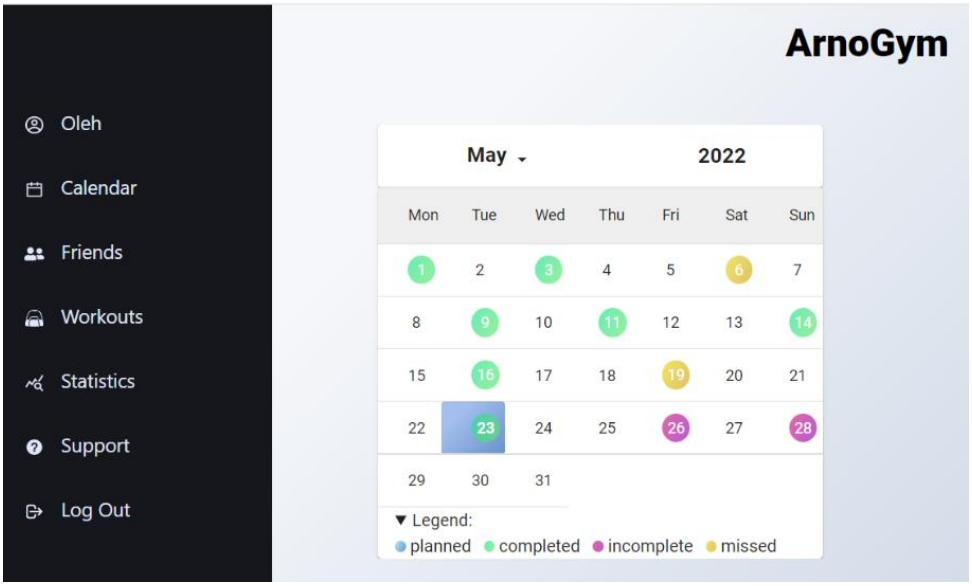

Рисунок 15 – Календар тренувань

Тренер має можливість додати тренування до будь-якої дати в календар певного спортсмена, з своїх вже створених або створивши нове тренування (Рис.16) з вже існуючих вправ, або створити нову (Рис.17). До вправи можна додати відео або опис

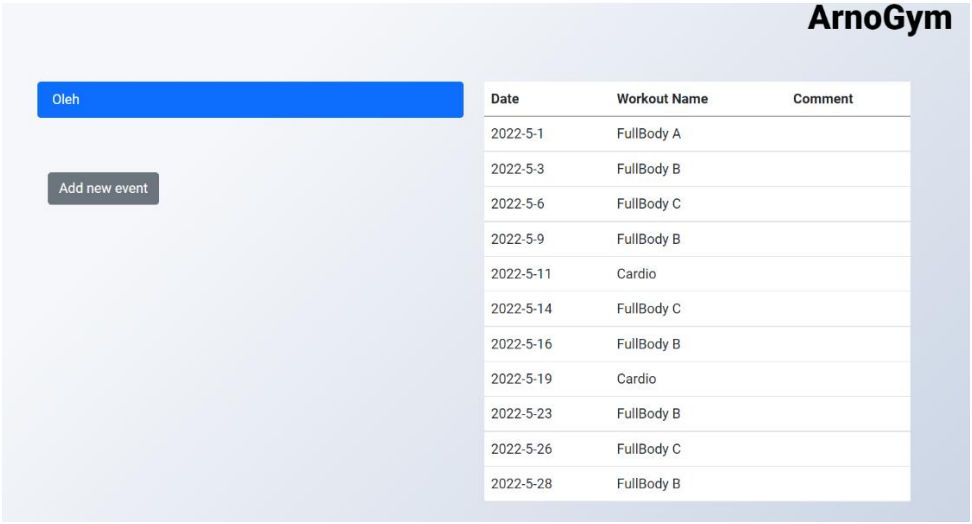

Рисунок 18 – План тренувань

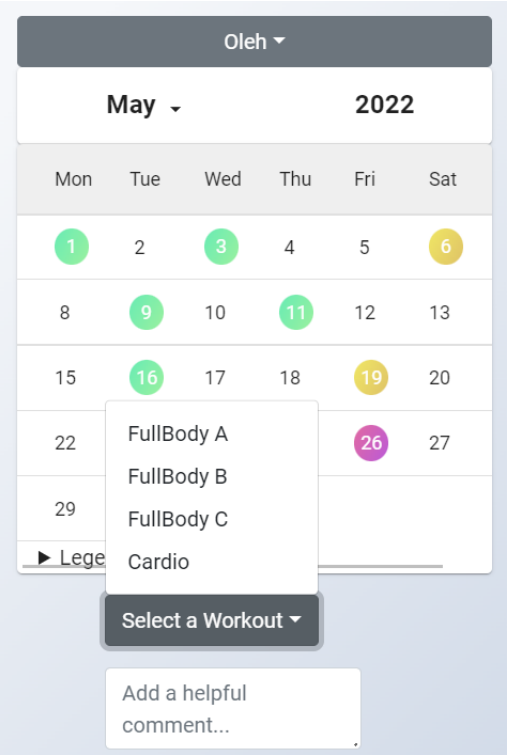

Рисунок 16 – Планування тренуваннь

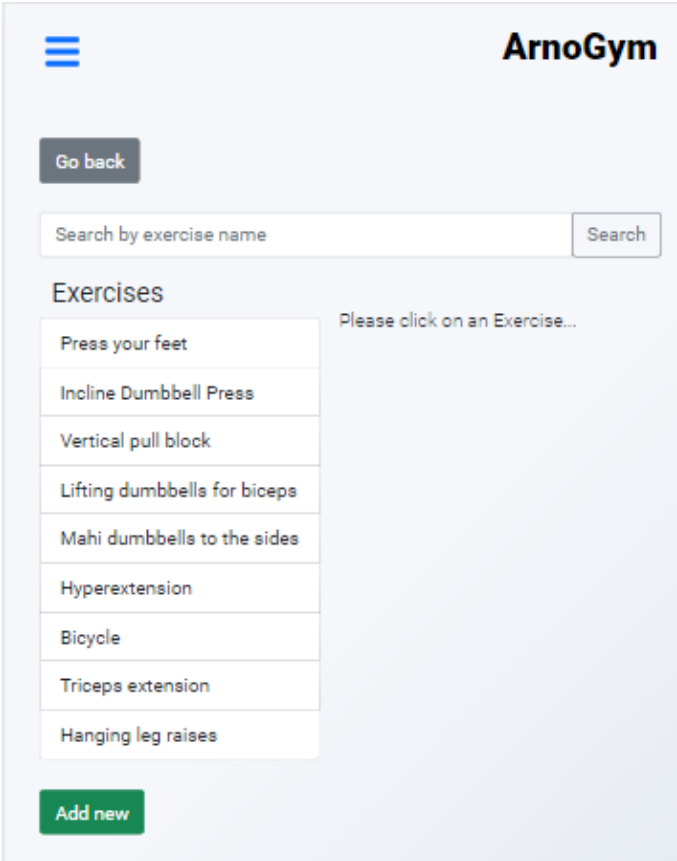

Рисунок 17 – Редагування вправ

Також можна переглянути план тренувань – коли було виконано, які є коментарі від спортсмена або додати якусь подію (Рис.18)

Спортсмен також може додавати в друзі інших спортсменів або тренерів (Рис.19) та після згоди ви можете бачити плани тренувань один одного

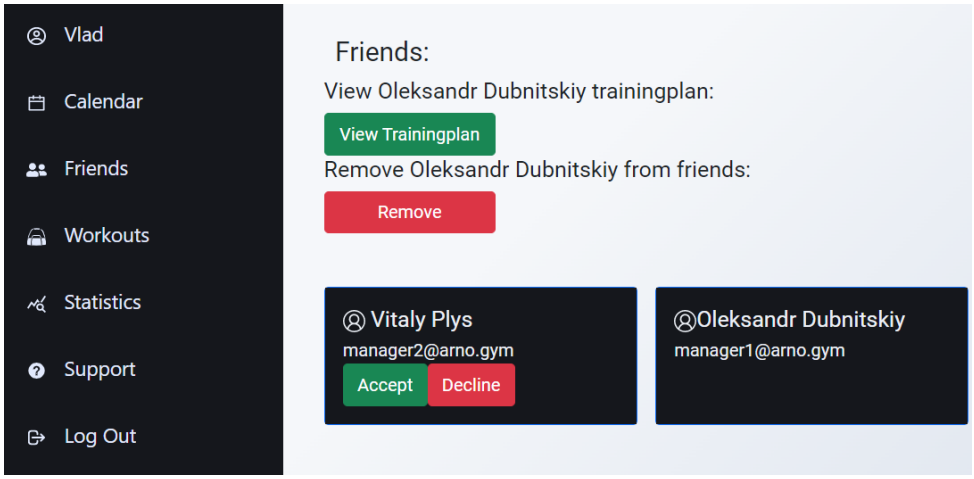

Рисунок 19 – Сторінка друзів

ВИСНОВКИ

У результаті виконання роботи було реалізувано практичне повноцінний веб-додаток, який дозволить тренерам комфортно планувати тренування, а це в свою чергу дозволяє спортсменам читати та звітувати про свої тренування.

У додатку реалізований редактор для створення тренування, крім того, тренер може створити власні бібліотеки тренувань та вправ, що допомагає йому зробити тренування більш ефективними. Спортсмен має можливість писати замітки про свої тренування

Одним із важливих кроків, написаних у цій роботі, була аналітична частина, яка мала місце перед фактичною розробкою веб-додатка і описана в першому розділі. Предметом аналітичної частини було дослідження технології створення веб-додатків та їх особливості, аналіз фрейморків, вибір найбільш підходящих технологій для розробки веб-додатку, а також дослідження існуючих веб-рішень, які допомогли створити власні рішення в веб-додатку.

У майбутньому планується додати нові функції, такі як: елементи соціальної мережі, автоматична синхронізація даних з популярними додатками, якими користуються спортсмени, подальше покращення статистики результатів тренувань і тренувань між спортсменами для створення дружнього спортивного співтовариства в цьому веб-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SCOTT, Emmit A. SPA design and architecture: understanding single-page web applications. Shelter Island, NY: Manning, 2016. ISBN 978-1617292439.
2. ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET [Електронний ресурс]. – Режим доступу : https://msdn.microsoft.com/en-us/magazine/dn463786.aspx
3. BROWN, Tiffany B., Kerry BUTTERS a Sandeep PANDA. HTML5 immediately: [master HTML5 over the weekend]. Brno: Computer Press, 2014. ISBN 9788025142967
4. HTML 5.2 [Електронний ресурс]. – Режим доступу : https://www.w3.org/TR/2017/REC-html52-20171214/
5. GASSTON, Peter. CSS3. Brno: Computer Press, 2016. ISBN 9788025146415.
6. CSS preprocessor [Електронний ресурс]. – Режим доступу : https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor
7. Javascript [Електронний ресурс]. – Режим доступу : https://developer.mozilla.org/en-US/docs/Web/JavaScript
8. The 10 most popular programming languages, according to the 'Facebook for programmers' [Електронний ресурс]. – Режим доступу : https://www.businessinsider.com/the-10-most-popular-programming-languages- according-to-github-2018-10#1-javascript-10
9. WIERUCH, Robin. The Road to learn React: Your journey to master plain yet pragmatic React.js. CreateSpace Independent Publishing Platform,2018. ISBN 978-1986338820.
10. Tutorial: Intro to React [Електронний ресурс]. – Режим доступу : https://reactjs.org/tutorial/tutorial.html#what-is-react

11. Components and Props [Електронний ресурс]. – Режим доступу : https://reactjs.org/docs/components-and-props.html
12. React.Component [Електронний ресурс]. – Режим доступу : https://reactjs.org/docs/react-component.html
13. React Lifecycle Methods – A Deep Dive [Електронний ресурс]. – Режим доступу : https://programmingwithmosh.com/javascript/react-lifecycle-methods/
14. Draft: JSX Specification [Електронний ресурс]. – Режим доступу : https://facebook.github.io/jsx/
15. Fedosejev, Artemij. React.js essentials: a fast-paced guide to designing and buil- ding scalable and maintainable web apps with React.js. First published. Bir-mingham: Packt Publishing, 2015. x, 183 stran. Open source community experience distilled. ISBN 978-1-78355-162-0.
16. A Beginners Guide to Redux [Електронний ресурс]. – Режим доступу : https://www.gistia.com/beginners-guide-redux/
17. The History of React.js on a Timeline [Електронний ресурс]. – Режим доступу : https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/
18. React: A declarative, efficient, and flexible JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу : https://github.com/facebook/react
19. mvc [Електронний ресурс]. – Режим доступу : https://www.upwork.com/hiring/development/express-js-a-server-side-javascript-framework/.
20. Express.js guide using-middleware [Електронний ресурс]. – Режим доступу : http://expressjs.com/uz/guide/using-middleware.html.
21. Node Package Manager [Електронний ресурс]. – Режим доступу : https://docs.npmjs.com/getting-started/what-is-npm.
22. Node.js event-loop [Електронний ресурс]. – Режим доступу :

http:// abdelraoof.com/blog/2015/10/28/understanding-nodejs-event-loop.

23. Mardan, A. Express.js Guid: The Comprehensive Book on Express.js. CreateSpace Independent Publishing Platform, 2014. ISBN 1494269279.
24. Mclaughlin, B. What Is Node? O'Reilly Media, 2011. ISBN 978-1-4493-1005-9
25. Express.js guide basic-routing [Електронний ресурс]. – Режим доступу : http://expressjs.com/en/starter/basic-routing.html.
26. Utility library Lodash [Електронний ресурс]. – Режим доступу : https://lodash.com/.