**АНОТАЦІЯ**

Метою бакалаврської кваліфікаційної роботи було створення програмної системи, яка надає зручний інтерфейс для менеджменту тренувань та дієти та надає засоби для відслідковування прогресу. Система складається з двох частин: мобільного застосунку для користувачів та серверу з базою даних, який оновлюватиме інформацію. Завдяки використанню реактивних технологій мобільний додаток буде надзвичайно швидким і доступним в роботі. Клієнтська частина розроблялась за допомогою фреймворку react-native.

Серверна частина розроблялась на платформі .NET CORE версії 3.0.1 з використанням технологій ASP.NET WEB API, Для збереження бази даних використовувався Microsoft Sql Server 2019.

До системи створено специфікацію вимог. Проектування та розробка програмного продукту проводилась згідно вимог представлених у специфікації та на основі аналізу предметної області.

Робота містить програмну реалізацію системи та пояснювальну записку, загальний розмір якої 78 сторінка, розмір основного матеріалу 53 сторінки.

**ABSTRACT**

The aim of the bachelor's thesis was to create a software system that provides a user-friendly interface for exercise and diet management and provides tools for tracking progress. The system consists of two parts: a mobile application for users and a database server that will update the information. Thanks to the use of react technology, the mobile application will be extremely fast and accessible. The client part was developed using the react-native framework.

The server part was developed on the .NET CORE platform version 3.0.1 using ASP.NET WEB API technologies. Microsoft Sql Server 2019 was used to save the database.

A requirements specification has been created for the system. The design and development of the software product was carried out according to the requirements presented in the specification and on the basis of the analysis of the subject area.

The work contains a software implementation of the system and an explanatory note, the total size of which is 78 pages, the size of the main material is 53 pages.

Зміст

[ВСТУП 9](#_Toc73574832)

[РОЗДІЛ 1. АНАЛІЗ ОСОБЛИВОСТЕЙ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ФІТНЕС ДОДАТКІВ 10](#_Toc73574833)

[1.1. Роль фітнес додатків в сучасному суспільстві 10](#_Toc73574834)

[1.2. Аналіз популярних фітнес додатків 10](#_Toc73574835)

[1.3. Висновок 13](#_Toc73574836)

[РОЗДІЛ 2. ОПИС ЗАВДАННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15](#_Toc73574837)

[2.1. Постановка завдання 15](#_Toc73574838)

[2.2. Специфікація вимог до програмного продукту 15](#_Toc73574839)

[2.2.1. Призначення та мета 15](#_Toc73574840)

[2.2.2. Продукти-аналоги 15](#_Toc73574841)

[2.2.3. Загальний опис 16](#_Toc73574842)

[2.2.3.1. Характеристики продукту 16](#_Toc73574843)

[2.2.3.2. Класи користувачів та їхня характеристика 17](#_Toc73574844)

[2.2.3.3. Середовище функціонування 17](#_Toc73574845)

[2.2.3. Характеристики системи 17](#_Toc73574846)

[2.2.4. Вимоги зовнішніх інтерфейсів 23](#_Toc73574847)

[2.2.4.1. Користувацький інтерфейс 23](#_Toc73574848)

[2.2.5. Нефункційні вимоги 23](#_Toc73574849)

[2.2.5.1. Вимоги продуктивності 23](#_Toc73574850)

[2.2.5.2. Вимоги безпеки 23](#_Toc73574851)

[2.2.6. Атрибути якості програмного продукту 23](#_Toc73574852)

[2.3. Вибір засобів для виконання завдання 23](#_Toc73574853)

[2.4. Висновок 24](#_Toc73574854)

[РОЗДІЛ 3. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ФІТНЕСУ. 25](#_Toc73574855)

[3.1. Архітектура системи 25](#_Toc73574856)

[3.2. Проектування компонент та пакетів серверної частини 26](#_Toc73574857)

[3.3. Проектування бази даних 27](#_Toc73574858)

[3.4. Проектування архітектури нативного веб-застосунку 29](#_Toc73574859)

[3.5. Висновок 31](#_Toc73574860)

[РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-СИСТЕМИ 33](#_Toc73574861)

[4.1. Опис реалізації основних методів програми. 33](#_Toc73574862)

[4.2. Опис роботи системи 35](#_Toc73574863)

[4.3. Тестування програмного забезпечення 37](#_Toc73574864)

[4.3.1. Вступ 37](#_Toc73574865)

[4.3.2. Розробка тестів 37](#_Toc73574866)

[4.3.3. Функціональне тестування 38](#_Toc73574867)

[4.3.4. Навантажувальне тестування 38](#_Toc73574868)

[4.3.5. Критерії успішності виконання тестування 38](#_Toc73574869)

[4.4. Висновок 38](#_Toc73574870)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 39](#_Toc73574871)

[5.1. Економічна характеристика проектного рішення 39](#_Toc73574872)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту 40](#_Toc73574873)

[5.3. Оцінювання та аналізування факторів внутрішнього та зовнішнього середовища 41](#_Toc73574874)

[5.4. Формування стратегічних альтернатив 44](#_Toc73574875)

[5.5. Бюджетування 47](#_Toc73574876)

[5.6. Висновки до економічної частини 52](#_Toc73574877)

[ВИСНОВКИ 53](#_Toc73574878)

[ЛІТЕРАТУРНІ ДЖЕРЕЛА 54](#_Toc73574879)

[ДОДАТОК А. Діаграма варіантів використання, розгортання та компонент 56](#_Toc73574880)

[ДОДАТОК Б. Фізична схема бази даних 58](#_Toc73574881)

[ДОДАТОК В. Скрипт створення бази даних 59](#_Toc73574882)

[ДОДАТОК Г. Варіанти використання 64](#_Toc73574883)

[ДОДАТОК Д. Звіт про навантажувальне тестування 69](#_Toc73574884)

[ДОДАТОК Е. Інструкція користувача 70](#_Toc73574885)

[ДОДАТОК Є. Основні класи та компоненти програми 72](#_Toc73574886)

# ВСТУП

В сучасному ритмі життя досить важко виділити достатньо часу на догляд за своїм здоров’ям та формою. Тому важливо мати постійну мотивацію для виконання щоденних вправ. Фітнес застосунки розробляються компаніями для комфортного та простого слідкування за своїм прогресом.

Ще одне, що визначає доцільність створення такого ПЗ є те, що використання фітнес застосунку значно спрощує моніторинг власної дієти і підрахунок калорій. Особливо якщо при цьому можна бачити скільки калорій користувач спалює при виконанні конкретних вправ.

Мобільний застосунок суттєво економить ваш час, оскільки в ньому зібрані усі корисні інструменти для трекінгу вашої активності і доступні в будь який момент.

Очевидно, що на ринку вже давно існує різноманіття програм для фітнесу. Однак, більшість з них мають дуже обмежений безкоштовний функціонал для користувача і досить складний в освоєні графічний інтерфейс. Аналізуючи ринок для андроїд додатків я помітив що більшість з них не мають програми харчування та вузько направленні на стать чи похудання, що не для кожної людини є її метою.

Метою бакалаврської кваліфікаційної роботи буде розробка програмної системи, яка вирішуватиме ці питання, зокрема буде мати кілька прогам тренувань/дієт та щоденний моніторинг прогресу, а також матиме відео і описові гайди вправ та рецепти. Особливість цієї програми буде в її простоті для розуміння і заохочення користувача в подальшому прогресі.

# РОЗДІЛ 1. АНАЛІЗ ОСОБЛИВОСТЕЙ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ФІТНЕС ДОДАТКІВ

## Роль фітнес додатків в сучасному суспільстві

Практично усі люди знають, що для того щоб бути здоровим варто дотримуватись дієти і робити вправи. Однак, якщо запитати в людини зазвичай вона не зможе сказати як саме варто розподіляти навантаження для свого власного організму або як при цьому правильно харчуватись. Можливо зможуть назвати кілька продуктів, але реальна користь дієти полягає в її збалансованості, а не в окремих продуктах.

Програми надають цінну інформацію, часто в режимі реального часу, але нічого не роблять для того, щоб зробити дані ефективними. Дієта - це неймовірно важлива частина фітнес-програми та здорового способу життя загалом. На жаль, темпи сучасного життя та розповсюдження дешевих закусок можуть стати складною справою для правильного вибору дієти, коли втомилися або поспішаємо. І ще до цього - тренажерні зали, тренери, дієтологи та лікарі здаються занадто недоступними та надто дорогими. Люди все частіше знаходять менше часу у своєму насиченому житті, щоб присвятити себе здоровому способу життя.

Особливістю власної системи я хотів би виділити ще те, що в ній буде можливість додавати калорії які ви з’їли окремо від дієти. Нерідко буває що немає часу щось готувати, а треба бігти на роботу. Тоді ви купуєте, наприклад, бургер в Мак Дональдс. Практично на усіх продуктах пише їхню калорійність. Так, доводиться записувати кожен з’їдений продукт та бачити його калорійність - це шокує. Дрібниці, про які ви ніколи раніше не думали, раптом стають великими міркуваннями.

## Аналіз популярних фітнес додатків

На сьогоднішній день існує багато сервісів для фітнесу та побуту. Один з найбільш популярних в Україні є Seven – 7-хвилинні вправи [1]. Даний сервіс дозволяє приділяти всього 7хв на день для тренувань щоб бути у формі. Рейтинг на Plat Market – 4.4/5.

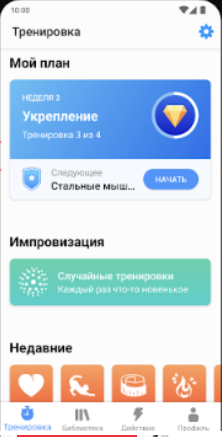


Рис. 1.1. Сервіс «Seven — 7-хвилинні вправи» [1]

При розробці додатку важливо вказувати складність тої чи іншої програми тренувань. Це дозволить новачкам вибрати адекватний рівень навантажень інтуїтивно, а також перетворить процес тренування в гру, створюючи мотивацію для проходження складнішого рівня. Також це має психологічний сенс, оскільки при правильному заохочені користувача програмою в нього формується звичка. Так в додатку Seven – 7-хвилинні вправи представлені такі категорії програм:

* Get Started (підходить для початківців)
* Lose Weight (націлена на спалення калорій)
* Get Fit (на покращення стану тіла)
* Mobility (підходить для людей які займаються мобільними видами спорту, наприклад, бадмінтон чи футбол)
* Be Challenged (підходить для тих хто хоче досягнути свого ліміту)

При виборі певної програми можна побачити набір тренувань. Більшість з них заблоковані оскільки платні. Перевагою даної програми є анімації вправ та голосовий супровід. При спробі вийти з тренування програма спробує вас відмовити. Програма не дає змогу вести статистику спалених калорій.

Ще один фітнес сервіс доступний на Android та iOS Домашні Тренування – без Додаткового Спорядження [2], який у 2018 році отримав нагороду «Найкращий застосунок» за версією Google Play у розділі «Фітнес». Він складається з збірників детально описаних та показаних тренувань із власною вагою для чоловіків та жінок. Користувачу досить внести свої дані й вибрати бажаний рівень підготовки. Доступні тренування направлені на балансування ваги та в основному на збільшення обсягу м’язів різних зон.

Великою перевагою даного сервісу є те що в ньому досить багато безкоштовного функціоналу. А також на відміну від попереднього сервісу має вкладку репорт в якій можна відслідковувати історію тренувань та прогрес ваги.

В даному сервісі є слайд меню, що я вважаю дуже зручним рішенням. Однак сам інтерфейс трохи ускладнений та самі програми більше нагадують якийсь лендінг без чіткої ієрархії.

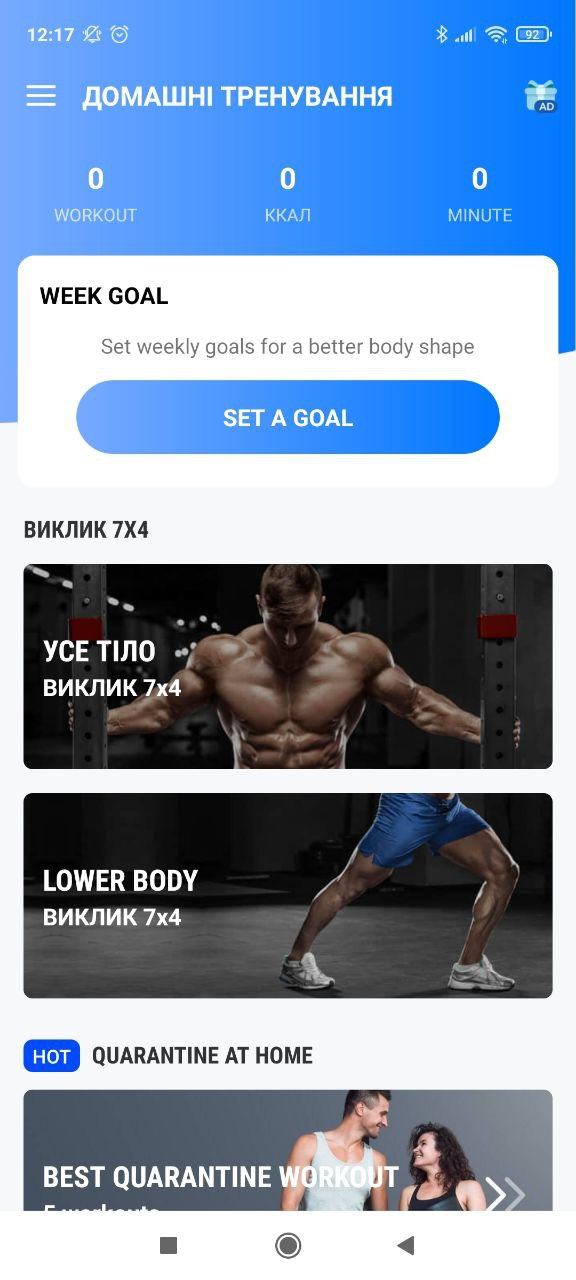
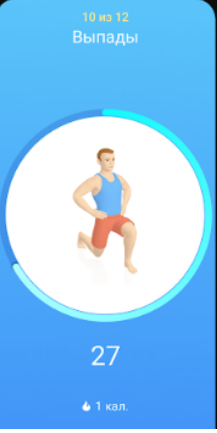


Рис. 1.2, 1.3 Анімації вправ в додатку Seven – 7-хвилинні вправи [1], сервіс «Домашні Тренування – без Додаткового Спорядження» [2]

Серед плюсів описаних вище програм можна зазначити:

* Естетичний інтерфейс і контент наповнення
* Анімації завдань
* Перегляд прогресу
* Експертна база (гайди, підказки і тд.)
* Нагадування
* Можливість реєстрації через Google

Як ви напевне вже помітили, в більшості програм для фітнесу акцент ставиться тільки на тренуваннях, а не на дієті. Це досить велике упущення, оскільки набагато легше досягнути поставлених результатів за допомогою дієти ніж тренувань. Навіть якщо людина буде виконувати достатню кількість роботи на день, але при цьому споживати висококалорійну їжу, то ефективність такого способу буде досить низька. Однак, варто зазначити, що дослідження проведене університетом Колорадо вказує, що саме тренування, а не дієта впливають на закріплення результату і довгострокового збереження форми. "Наші висновки свідчать про те, що ця група успішних людей, що підтримують схуднення, споживає таку ж кількість калорій на день, як особи з надмірною вагою та ожирінням, але, схоже, уникає відновлення ваги, компенсуючи це високим рівнем фізичної активності", - сказала Вікторія А. Катеначчі, доктор медичних наук, лікар з управління вагою та дослідник медичного містечка CU Anschutz.

Отже, найпродуктивнішим та найшвидшим методом досягнути результату є сумістити дієту та тренування. Оскільки кожна частина дуже важлива, буде доцільно створити ПЗ, яке включатиме в себе переваги фітнес застосунків та розписані дієти на тиждень.

## Висновок

Отже, під час огляду та аналізу особливостей фітнес додатків я дослідив деякі статті про те чи може фітнес додаток створювати мотивацію та які аспекти важливі при побудові правильного програмного комплексу. Як видно з висновків на ринку досі немає ідеального рішення. Усі програми розглянуті мною мають вузьконаправлену специфіку та деякі недоліки.

На підставі проведеного аналізу продуктів аналогів та визначення їх особливостей можна сказати, що на ринку існує велика кількість програм орієнтованих на вирішення поставленої задачі, але більшість з них мають платний функціонал і не включають в себе програми дієт.

Отже, в цей момент в суспільстві існує потреба в системі, що б полегшувала відстежування своїх тренувань, а також давала б чуткі інструкції по дієті. Існуючі аналоги або не задовольняють усіх потреб, або мають надлишкову функціональність, через що збільшується час для інтуїтивного ефективного використання системи.

# РОЗДІЛ 2. ОПИС ЗАВДАННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1. Постановка завдання

Поставлене завдання – розробити програмне забезпечення, основне ціль якого надати користувачам зручний та простий в освоєнні інтерфейс для менеджменту власних тренувань та дієти.

Дана система повинна надати інтерфейс реєстрації та авторизації. Для зареєстрованих користувачів повинна бути доступна вкладка з щоденними вправами, де вони можуть вибрати вправу подивитись інструкцію або просто почати відлік і робити її, вкладка з інформацією про себе де можна поміняти ріст, вагу, вік, тощо. Також вкладка з статистикою де можна порівняти скільки калорій на день користувач спалює, можна подивитись передбачення ваги через якийсь час. Вкладка програми, де користувач може вибрати для себе програму, почитати її опис, водночас користувач може бути підписаний тільки на 1 програму, програма визначає порядок і інтенсивність тренувань, а також збільшує або зменшує кількість денних калорій. Такий підхід значно полегшує розуміння програми користувачем. Користувач може в будь який момент змінити підписку на іншу програму. Також має бути вкладка з корисними туторіалами.

За допомогою даного програмного забезпечення користувачі матимуть змогу легко почати вести здоровий спосіб життя. Програмне забезпечення має коректно відображатись на мобільних пристроях.

## 2.2. Специфікація вимог до програмного продукту

## 2.2.1. Призначення та мета

Програмне забезпечення “Fittrain” призначене для надання зручного інтерфейсу користувачам для менеджменту власних тренувань та дієти, а також як збірник корисних туторіалів і як трекінг прогресу тренувань.

## 2.2.2. Продукти-аналоги

На сьогоднішній день існує дуже багато мобільних додатків, що надають користувачам платні або безкоштовні набори вправ та слугують тренерами. Кожен з них відрізняється чимось особливим. Наприклад, в деяких є програми тренувань направлені на конкретні групи м’язів.

* Seven – 7-хвилинні вправи [1] – мобільний застосунок, який дозволяє вибрати свою фітнес ціль (бути в формі, накачатись, скинути вагу) і надає велику кількість різних тренувань, анімовані вправи, круглий дизайн та голос диктора типічний для таких додатків. Також його можна використовувати як щоденник активності.
* Домашні Тренування – без Додаткового Спорядження [2] –мобільний додаток, який визнано найкращим додатком в 2018 році в категорії «фітнес». Має велику кількість програм націлених на силові тренування впродовж довгого періоду часу, ідеально підійде для культуристів.

Проте, мною не було знайдено системи, яка б давала користувачу менеджмент дієти і тренувань одночасно.

## 2.2.3. Загальний опис

## 2.2.3.1. Характеристики продукту

Система володіє наступними характеристиками:

* Авторизація
* Реєстрація
* Робота з тренуваннями
* Робота з дієтою
* Перегляд статистики
* Перегляд передбачень
* Зміна особистих даних
* Вихід з акаунту
* Вибір програми тренувань
* Перегляд туторіалів

Діаграма прецедентів знаходиться в додатку А.

## 2.2.3.2. Класи користувачів та їхня характеристика

Система “Fittrain” орієнтована на один клас користувачів:

* Користувач мобільного додатку – користуються усім функціоналом додатку описаним вище: Вибирають програму тренувань, виконують вправи, переглядають рецепти, слідкують за калоріями в статистиці і тд.

## 2.2.3.3. Середовище функціонування

Серверна частина програмного продукту передбачає наступні мінімальні апаратні та програмні вимоги:

* Веб-сервер для сайту IIS Express.
* Операційна система Linux або Windows.
* Платформа .NET Core 3.0.1.
* Об'єм ОЗП не менше 4096МБ.
* Сервер баз даних – MS Sql.

Для повноцінної роботи програмного забезпечення необхідний постійний доступ в Інтернет.

Оскільки клієнтська частина розробляється нативною, для неї потрібен буде пристрій, наприклад смартфон на операційній системі Android або його емулятор.

## 2.2.3. Характеристики системи

1. **Авторизація**

***Опис і пріоритет***

Данна характеристика відповідає за змогу користувача увійти в систему, так користуватись її функціоналом. Для входу користувач зобов’язаний мати валідний логін та пароль. Система забезпечує відповідний інтерфейс входу.

Пріоритет: високий.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку входу в систему.

Відгук: система надає інтерфейс взаємодії з чутливими до набору інформації полями.

Дія: користувач заповнює поля вводу логіну та паролю і натискає на кнопку увійти.

Відгук: система передає данні на сервер, отримує токен авторизації і авторизує користувача.

***Функціональні вимоги***

REQ 1.1: Отриманий токен перевіряється на валідність та актуальність.

REQ 1.2: Результати запиту мають бути у відповідному форматі

REQ 1.3: Якщо зв’язок з сервером встановити не вдається то вивести повідомлення про помилку.

REQ 1.4: Якщо логін чи пароль введено невірно то вивести повідомлення про помилку.

1. **Реєстрація**

***Опис і пріоритет***

Не менш важливою характеристикою є реєстрація нових користувачів в систему. Вона забезпечує авторизацію нового користувача, передає його інформацію на сервер та створює для нього обліковий запис.

Пріоритет: високий.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку реєстрації в систему.

Відгук: система надає інтерфейс взаємодії з чутливими до набору інформації полями.

Дія: користувач заповнює поля вводу логіну та паролю, повторного паролю, віку, статі, ваги, росту і натискає на кнопку зареєструватися.

Відгук: система передає данні на сервер, реєструє користувача, отримує токен авторизації і авторизує користувача.

***Функціональні вимоги***

REQ 2.1: На сторінці реєстрації мають бути поля вводу для усіх параметрів користувача: логін, пароль, ім’я (псевдонім), стать, вік, вага та ріст.

REQ 2.2: Поле вводу паролю має маскувати введені символи.

REQ 2.3: Валідація даних користувача.

REQ 2.4: Обробка помилок при відповіді сервера про неможливість створення акаунту.

1. **Відображення інформації про щоденні вправи**

***Опис і пріоритет***

Поточна характеристика надає змогу підписатись на програму тренувань, після чого на сторінці вправ з’являться вправи на день, а на сторінці дієти рекомендовані страви.

Пріоритет: високий.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку вправ.

Відгук: система показує набір вправ на поточний день.

Дія: користувач вибирає вправу на виконання.

Відгук: система перенаправляє користувача на сторінку виконання вправи.

***Функціональні вимоги***

REQ 3.1: При відсутній підписці на програму тренувань замість вправ має показувати напис «There are no exercises for you yet. You should choose your program first.» з клікабельним посиланням на сторінку з програмами.

REQ 3.2: Програма має показувати тільки ті вправи, що рекомендовані конкретному користувачу в конкретний день тижня, день тижня програма визначає автоматично.

REQ 3.3: При натисканні на вправу здійснюється перехід на сторінку виконання вправи.

REQ 3.4: На сторінці мусять бути: рівень складності вправи, час виконання для кожої вправи.

1. **Навігаційна бокова панель.**

***Опис і пріоритет***

Дана характеристика надає можливість відкрити бокову панель. Це досить важливо оскільки це єдиний спосіб навігації в мобільному даному додатку. На боковій навігаційній панелі мають бути вкладки натискаючи на які користувач переходить на відповідні сторінки.

Пріоритет: високий.

***Послідовність дія/вігук***

Дія: користувач нажимає кнопку меню.

Відгук: система відкриває бокове меню.

Дія: користувач повторно нажимає кнопку меню.

Відгук: система закриває бокове меню.

***Функціональні вимоги***

REQ 4.1: Можливість відкрити бокове меню через проведення пальцем по екрану зліва направо.

REQ 4.2: Можливість відкрити бокове меню натиснувши на відповідну кнопку в куті екрану.

REQ 4.3: Швидка навігація по програмі.

REQ 4.4: Бокове меню закривається автоматично після переходу на сторінку.

1. **Відображення інформації про щоденні страви**

***Опис і пріоритет***

Завдяки цій характеристиці користувачу відображаються рекомендовані страви на день.

Пріоритет: середній.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку дієти.

Відгук: система показує набір страв на поточний день.

Дія: користувач вибирає страву для перегляду.

Відгук: система перенаправляє користувача на сторінку детального опису страви.

***Функціональні вимоги***

REQ 5.1: При відсутній підписці на програму тренувань замість страв має показувати напис «There are no daily menu for you yet. You should choose your program first.» з клікабельним посиланням на сторінку з програмами.

REQ 5.2: Програма має показувати тільки ті страви, що рекомендовані конкретному користувачу в конкретний день тижня, день тижня програма визначає автоматично.

REQ 5.3: При натисканні на страву здійснюється перехід на сторінку з рецептом страви.

1. **Перегляд туторіалів**

***Опис і пріоритет***

Завдяки даній характеристиці користувач програми має змогу зайти на сторінку «Guides» і переглянути статті та відео на різні теми пов'язані з спорядженням, тренуваннями та здоровим способом життя.

Пріоритет: низький.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку туторіалів.

Відгук: система показує список наявних туторіалів.

Дія: користувач вибирає туторіал для перегляду.

Відгук: система перенаправляє користувача на сторінку детального опису статті.

***Функціональні вимоги***

REQ 6.1: При натисканні на заголовок статті програма перенаправляє користувача на сторінку з повним описом статті.

1. **Перегляд та підписка на програми тренувань**

***Опис і пріоритет***

Користувач може відкрити «Programs» вкладку та переглянути інформацію про програму тренувань. А також підписатись на неї.

Пріоритет: середній.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку програм.

Відгук: система показує список наявних програм.

Дія: користувач підписується на вибрану програму.

Відгук: система відправляє запит на сервер з оновленям інформації.

***Функціональні вимоги***

REQ 7.1: Користувач може побачити рівень складності тої чи іншої програми по шкалі від 1-5

REQ 7.2: При підписці на програму на сервер надсилається відповідний запит, після чого інформація про вибрану програму тренувань зберігатиметься в базі.

1. **Зміна інформації користувача**

***Опис і пріоритет***

Зайшовши на вкладку «Settings» завдяки даній характеристиці користувач може змінити особисту інформацію або вийти з акаунту.

Пріоритет: низький.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку налаштувань.

Відгук: система показує список наявних налаштувань.

Дія: користувач вибирає вийти з системи або змінити особисту інформацію.

Відгук: система перенаправляє користувача в першому випадку на сторінку авторизації, в другому на сторінку зміни інформації..

***Функціональні вимоги***

REQ 8.1: Валідація введених даних.

REQ 8.2: При помилці чи успіху змін користувач отримує повідомлення про це.

1. **Перегляд статистики та прогнозувань**

***Опис і пріоритет***

Ця характеристика обумовлюється потребою користувача в перегляді та прогнозуванні своїх результатів.

Пріоритет: середній.

***Послідовність дія/вігук***

Дія: користувач переходить на сторінку статистики.

Відгук: система показує графік виконаних/невиконаних тренувань та графік прогнозу прогресу.

***Функціональні вимоги***

REQ 9.1: Графічне зображення даних.

## 2.2.4. Вимоги зовнішніх інтерфейсів

## 2.2.4.1. Користувацький інтерфейс

Взаємодія клієнтської частини з серверною відбувається за допомогою REST API.

## 2.2.5. Нефункційні вимоги

## 2.2.5.1. Вимоги продуктивності

Запити до серверу повинні оброблятись в межах декількох секунд, оскільки це максимальний час очікування зі сторони UX для користувача.

## 2.2.5.2. Вимоги безпеки

Усі запити до сервера окрім реєстрації та авторизації мають бути виконані при валідному JWT токені. Час дії токену 3 години. Паролі в базі даних збережені в захешованому варіанті.

## 2.2.6. Атрибути якості програмного продукту

Клієнтська частина не повинна мати випадків при яких вона зупиняється в роботі при умові робочого серверу. Усі запити до серверу здійснюються за допомогою HTTPS протоколу. Задля кращого освоєння нових користувачів буде розроблена інструкція використання.

## 2.3. Вибір засобів для виконання завдання

Для розробки клієнту було використано мову програмування JavaScript. Основою платформи для розробки став node.js, який заснований на рушії V8, який перетворює джаваскрипт в машинний код. Перевагами такого вибору стали надійність, продуктивність, функціональність.

В якості бібліотеки для розробки мобільного додатку була використана React-Native. Ця бібліотека, як і її аналог React, використовується для побудови ефективних SPA застосунків. Вона має характерний компонентний декларативний стиль написання коду. В поєднанні з потужним імперативним JavaScript ця технологія від Facebook стала неймовірно популярною і корисною.

Для зберігання і оновлення даних в мобільному застосунку були використані бібліотеки Redux та AsyncStorage. За допомогою яких можна було уникнути очищення JWT токена кожен раз після закривання додатку.

Для серверної частини я вирішив використати технологію REST API, а саме ASP.NET WEB API на базі .NET CORE 3.1. Основний сенс цієї технології це зв'язок між фронтендом та бекендом за допомогою HTTPS протоколів. Що являється стандартом в інтерфейсі програм.

В якості серверу баз даних я використав SQL Server. Створював базу даних та скрипти я в середовищі MSS Management Studio, а для підключення до бекенду використав технологію Entity Framework.

## Висновок

За результатами аналізу роботи над даним розділом було вирішено розділити систему на клієнтську та серверну частину. Також була створена специфікація вимог до програмного забезпечення. В якій були описані вимоги до кожної з характеристик наведених вище.

Крім того були вибрані інструменти розробки майбутньої системи. А саме для фронтенду: React-Native, JavaScript, Node.js, Redux, AsyncStorage. А для серверу: ASP.NET WEB API, C#, EntityFramework.

# РОЗДІЛ 3. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ФІТНЕСУ.

## 3.1. Архітектура системи

В попередньому розділі було вибрано модель архітектури клієнт-сервер. Клієнт-серверна архітектура - це обчислювальна модель, в якій сервер розміщує, постачає та управляє більшістю ресурсів та послуг, які споживає клієнт. Цей тип архітектури має багато клієнтів, підключених до центрального сервера через локальну мережу або інтернет. Клієнт-серверна архітектура надає високий рівень контролю системи оскільки вона централізована і усі запити обробляються незалежно один від одного. Потік даних зазвичай ініціюється клієнтом, який запитує якісь дані і сервер обробляє запит і відправляє якісь дані назад до клієнта через протокол. Клієнти не можуть безпосередньо спілкуватися між собою [3].

Інтерфейс спілкування сервера з програмою клієнта буде реалізований через принципи RESTful API. Це доволі типове рішення для таких систем.

Першопочатково архітектура REST (Representational State Transfer) задумувалась як набір обмежень, який мав би спростити взаємодію між компонентами в мережі інтернет. По друге, це дозволяло розмежувати клієнт та сервер, залишаючи на клієнті те, що потрібно клієнту, а не на сервері. Спрощення серверної сачтини позитивно впливає на масштабування. Та і взагалом розділення на дві незалежні частини дозволяє їм розвиватись незалежно один від одного. Існує 6 основних обмежень по Філдінгу. Проаналізувавши їх я зробив такі висновки: в період між запитами клієнта на сервері не має зберігатись ніяка інформація, що належить клієнту, крім токену авторизації, бекенд має представляти собою уніфікований стандарт інтерфейсу, який прийматиме та відправлятиме запити в відповідному форматі, на кожен з методів HTTP протоколу має відповідати один або декілька методів CRUD контролера, але кожен метод матиме власний ідентифікатор URL. Тобто не може бути 2 чи більше POST методів з одинаковим URL [4].

Ще одна особливість REST принципів – це відділення даних від сервера. На самому сервері в постійній основі не зберігаються дані, які запитує клієнт. Тобто сервер наприклад бере дані з бд і повертає їх в JSON форматі клієнту при цьому сам JSON об’єкт не зберігається на сервері.

Логічне представлення архітектури відображено на діаграмі розгортання (Додаток А)

Також була розроблена діаграма компонент для кращого розуміння інтерфейсів компонент. (Додаток А)

## 3.2. Проектування компонент та пакетів серверної частини

При проектуванні серверної частини використано такі модулі:

* AuthControler – авторизація та взаємодія з сервісом AspNetCore Authorization.
* ExerciseControler – відповідає за сервіс вправ і повертає вправи по айді користувача.
* ProgramControler – відповідає за вибір програми.
* DishControler – відповідає за сервіс страв і повертає страви по айді користувача.
* UserControler – відповідає за оновлення інформації про користувача.
* GuideControler – відповідає за повернення інформації про туторіали.
* Models – набір класів які відповідатимуть реальним таблицям в базі даних, потрібен для співпраці з сервісом EntityFramework.
* DBContext – імплементації інтерфейсу IDBSet, потрібен для співпраці з сервісом EntityFramework.
* Startup – модуль, що відповідає за налаштування сервісів, зокрема CORS політик.

Для розробки серверу буде використано наступні програмні пакети

* .NET Core – платформа для написання WEB API
* Entity Framework Core – пакет для звязування даних в базі даних та моделей на сервері в .NET Core.
* AspNetCore Authorization – пакет для реалізації авторизації на API

На відміну від .NET Framework платформа .NET Core є модульною, що означає, що її пакети підключаються і оновлюються незалежно через менеджер пакетв NuGet.

.NET Core в сучасних реаліяї є одною з найкращих платформ для розробки середніх та великих проектів, оскільки має величезне різноманіття технологій та потужну бібліотеку класів. Також вона кросплатформена, що дозволяє розробляти додатки на будь якій платформі.

## 3.3. Проектування бази даних

Коректно спроектована база даних є дуже важливим етапом в розробці програмного продукту. База даних мусить бути нормалізована до того рівня який потрібен на проекті. Найоптимальніший варіант це 3 нормальна форма, при цьому кількість таблиць не буде космічною, швидкість і простота запитів будуть оптимізовані.

Вибір реляційної бази даних багато в чому пояснюється її стабільністю та швидкістю роботи.

Фізична модель бази даних наведена в додатку Б.

Отож, було створено наступні таблиці: “users”, “dishes”, “exercises”, “guides”, “programs”, “programDishes”, “programExercises”.

**Users** – містить інформацію про користувачів системи:

* id – унікальний ідентифікатор;
* name – ім’я користувача;
* email – емейл користувача;
* password – пароль користувача у вигляді хешу;
* age – вік користувача;
* weight – вага користувача;
* height – ріст користувача;
* sex – стать користувача (0-чол, 1-жін);
* programId – ключ посилання на програму на яку підписаний користувач.

**Programs** – містить інформацію про програми тренувань:

* id – унікальний ідентифікатор;
* name – назва програми;
* description – коротки опис програми;
* complexity – складність програми.

**Guides** – містить інформацію про туторіали:

* id – унікальний ідентифікатор;
* name – назва туторіалу;
* description – коротки опис туторіалу;
* photoURL – посилання на фото;
* videoURL – посилання на відео;
* text – текст туторіалу.

**Exercises** – містить інформацію про вправи:

* id – унікальний ідентифікатор;
* name – назва вправи;
* description – коротки опис вправи;
* photoURL – посилання на фото;
* videoURL – посилання на відео;
* time – час виконання в секундах;
* complexity – складність вправи.

**Dishes** – містить інформацію про страви:

* id – унікальний ідентифікатор;
* name – назва страви;
* description – коротки опис страви;
* photoURL – посилання на фото;
* videoURL – посилання на відео;
* kkal – кількість ккал.;
* fats – кількість жирів;
* hydrocarbonates – кількість вуглеводів;
* netto – маса нетто;
* receipt – рецепт;
* tag – тег страви.

**ProgramDishes** – службова таблиця багато до багатьох, містить інформацію про страви які входять в програму на кожен день:

* id – унікальний ідентифікатор;
* programId – ключ посилання на програму;
* dishId – ключ посилання на страву;
* day – день тижня.

**ProgramExercises** – службова таблиця багато до багатьох, містить інформацію про вправи які входять в програму на кожен день:

* id – унікальний ідентифікатор;
* programId – ключ посилання на програму;
* exerciseId – ключ посилання на вправу;
* day – день тижня.

В додатку В представлений скрипт створення бази даних.

## 3.4. Проектування архітектури нативного веб-застосунку

Для проектування нативних застосунків було вирішено використовувати технологію React-Native. А реакт має досить просту і в той же час ефективну архітектуру SPA (Single Page Application). Її суть в тому, що в один єдиний html документ вставляється код інших компонент, які в свою чергу використовують ще якісь компоненти. В цьому суть компонентного підходу. При переході зі сторінки на сторінку насправді нічого не перезагружаться, просто міняється наповення документу.

Для проектування таких застосунків часто використовується Flux архітектура (рис. 3.1.).

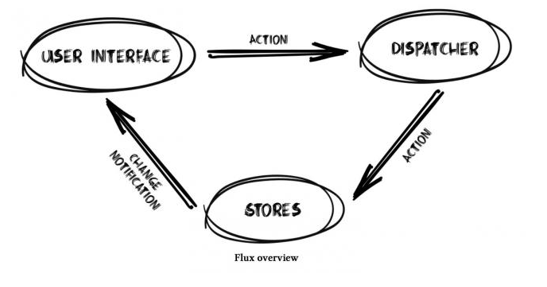


Рис. 3.1. Схема Flux архітектури

Flux – це такий архітектурний підхід або набір патернів програмування для створення користувацького інтерфейсу веб застосунків побудований на однонаправлених потоках даних. Він добре суміщається з принципами реактивного програмування. Суть в тому, що представлення у цій архітектурі самі по собі не здатні змінювати дані. Для цього вони викликають сповіщувач, передаючи в нього конкретні наперед відомі дії. Завдяки такому принципу розділення компонент від даних можна легко сповіщати потрібні компоненти про те, що їм варто ререндеритись. Не треба робити складні взаємодії через «батька - сина» [5]. Для даного проекту була використана бібліотека Redux, оскільки Flux це ідея, а Redux – її реалізація. (рис. 3.2.)

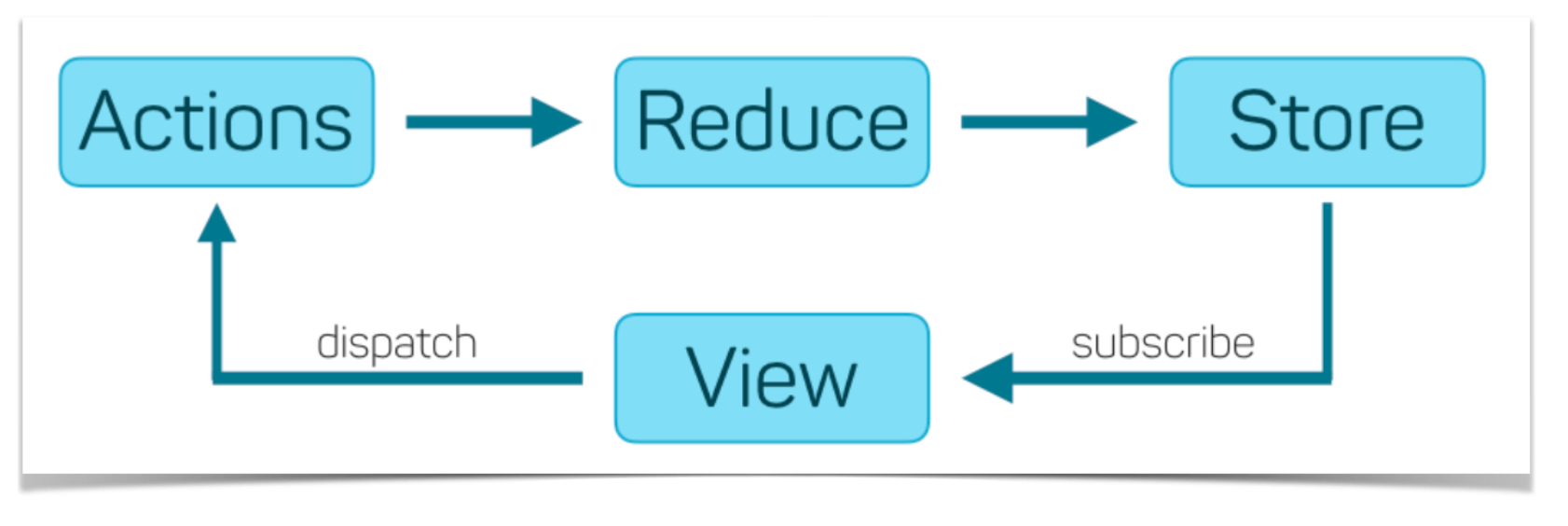


Рис. 3.2. Схема роботи Redux

Суть редаксу досить проста і поміщається всього на 7 стрічках коду. Редакс вирішує проблему заплутаності. Оскільки сховище стає глобальним, дані не треба нікуди передавати. Єдиний спосіб змінити дані це викликати дію. Дія це звичайна функція яка має назву і параметр. Редюсер її отримує і обробляє повертаючи нове значення. В реакті існують допоміжні інструменти, які ще більше полегшують роботу з редаксом. Хуки це функції які предоставляє окрема бібліотека для сумісності з реактом. Варто зазначити, що редакс хоч і створений тими ж людьми з компанії Facebook, які створили реакт, сам він не має прямого відношення до цієї технології. Його використовують і з іншими технологіями, наприклад Angular.

Таким чином використовуючи редакс можна отримати легкий та гнучкий котроль над даними.

## 3.5. Висновок

Метою цього розділу було проаналізувати найбільш адекватні архітектурні рішення для програмної системи. А також спроектувати базу даних та їх взаємодію з кожним рівнем системи.

В даному розділі було визначено та проаналізовано, що REST архітектура найкраще підходить для виконання поставленого завдання. Вона має багато переваг, наприклад структуризація і легке розширення та підтримка функціоналу.

Також була розроблена архітектура бази даних. Спроектована клієнтська та серверна частини продукту. Клієнт змодельований на базі архітектури FLUX з використанням технологій Redux. Разом усі ці технології дозволять побудувати функціональну та розширювальну систему.

# РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-СИСТЕМИ

## 4.1. Опис реалізації основних методів програми.

Для реалізації бокового меню та навігації я використав бібліотеку React Navigation. А саме компоненту Drawer, яка відмальовує бокове меню. Для цього треба імпортувати метод createDrawerNavigator.

Далі, я створюю код для мого бокового меню, де Drawer.Navigator основна компонента навігатора, яка приймає в параметр компоненту бокового меню SideBar. Drawer.Screen в свою чергу виконують роль сторінок на які посилається навігатор. Тут можна задати надпис, іконку та компоненту, яку треба рендерити при переході на цю сторінку.

Компонента яка передається в якості drawerContent являється нашим боковим меню, в параметри передається список сторінок, які треба рендерити. Також тут можна задати стилі. Стилі в React Native не схожі на такі ж у React JS. Вони передаються як обєкт з полями, які можуть мати числові або стрічкові значення.

В минулому розділі я описував технологію Redux. Для її впровадження в проект треба для початку інсталювати дві бібліотеки: Redux та React-Redux. Наступним кроком є створення 3 основних компонентів FLUX архітектури, а саме: Actions, Reducers, Storage.

**Actions** – це дії які описані наступним способом, назва у вигляді стрічки та обєкт параметр.

**Reducers** – це функції які повертають змінений state відповідно до типу дії яка прийшла у якості параметра.

**Storage** – це глобальне сховище для state. Для його стоврення достатньо імпортувати функцію createStore з бібліотеки Redux та використати її передавши у параметр створений раніше редюсер.

Наступним реалізовується модуль реєстрації та авторизації. Наступним кодом являється запит на бекенд про введений пароль та емейл. Якщо ці данні валідні тоді сервер поверне JWT token, який успішно зберігається на фронтенді поки не вийде його термін придатності.

Для того щоб перевірити чи токен ще дійсний відправляється запит на сервер з існуючим токеном, якщо відповідь буде зі статусом 401 тоді перенаправляємо користувача на сторінку авторизації для повторного створення токена. Якщо ж відповідь 200 ОК, тоді перенаправляємо користувача на основну сторінку додатку.

Для реалізації технології Entity Framework для початку нам потрібно створити моделі для кожної з таблиць баз даних та контекст бази даних. Після чого виконати міграції. Усі моделі і контекст зберігається в папці models (Рис. 4.1.).

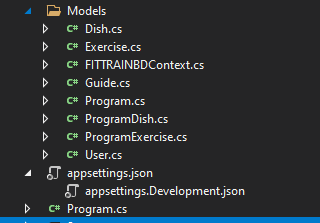


Рис. 4.1. Моделі та контекст бази даних

Для реалізації архітектури RESTfull API я створив контролери, які мають отримувати та повертати данні роблячи вибірку, модифікацію або вставку в базу даних. Наступний контролер повертає данні про вправи для конкретного користувача на поточний день.

Тег [Authorize] означає що до цього методу можуть доступитись тільки користувачі з валідним токеном. Токен створюється на основі секретної стрічки, яка зберігається тільки на сервері. Дізнатись секрет стрічки розкодувавши токен для майбутьніх генерацій фальшивих токенів неможливо. Оскільки токен захешований безпечним алгоритмом SHA256. А як відомо хешування це односторння операція і відновити початкову стрічку маючи хеш практично неможливо на сьогоднішній час.

## 4.2. Опис роботи системи

Система включає три частини: серверну, серевер бази даних та клієнтську. Основна роль сервера бути завжди доступним, його можна розмістити на одному з платних хостингів. Його суть в прийомі та надсилання інформації через https протокол. Зв'язок між сервером та нативним клієнтським застосунком передбачає доступ до глобальної мережі Інтернет в обох застосунках.

Серверна частина містить в собі ASP.NET WEB API застосунок, що реалізує архітектуру RESTful API. Очевидно, що серверна частина мусить мати інтерфейс зв'язку з сервером бази даних, а саме SQL Server. В якості інтерфейсу виступає технологія Entity Framework Core, яка по суті перетворює складний інтерфейс SQL подібних запитів на більш простий для розуміння інтерфейс класів та контексту бази даних. Для цього створюється папка Models, в якій будуть міститись моделі для кожної таблиці бази даних, навіть для службових. Також був створений контекст бази даних. За допомогою нього можна звертатись до інформації в таблицях бази даних як до елементів класів. Після зміни контексту запит буде формуватись до бази даних автоматично, завдяки функції контексту SaveChanges().

Також на серверній частині містяться контролери які обробляють https запити. А саме повертають інформацію про користувача якщо він авторизований, авторизують користувача по логіну і паролю. Повертають щоддений раціон та набір прав для користувача та іншу інформацію. Також реєструють користувача і дозволяють змінювати його особисту інформацію та налаштування.

Я вважаю, що досить логічно було би розмістити і серверний додаток і SQL Server на одному хостингу, тим більше більшість з них підтримують такі функції, наприклад Azure або AWS.

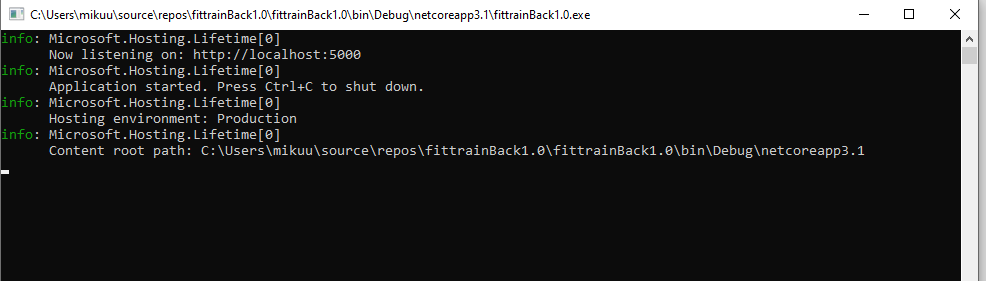


Рис. 4.2. Серверна частина

SQL Server це один з найпопулярніших серверів баз даних. Для роботи додатку необхідно буде встановити його та відновити базу даних додатку. Це можна зробити двома шляхами. Перший це відновити базу даних за допомогою резервного копіювання, наприклад, .bak файлу. Другий це створити нову базу даних виконавши скрипт створення бази даних, потім заповнити базу даних скриптом наповнення.

Клієнтська частина це нативний додаток орієнтований під пристрої на операційній системі Android. Він має зручний та зрозумілий для користувача інтерфейс. Написаний він за допомогою React Native. На головній сторінці користувач матиме змогу побачити які вправи на день в нього є згідно його програми тренувань.

Таким же чином можна побачити рекомендовані страви для користувача. Також клієнтський застосунок має сторінку авторизації. При авторизації введений логін і пароль надсилаються на сервер, де створюється токен авторизації, за допмогою якого наступний раз коли користувач заходитиме в додаток, авторизація пройде автоматично. І так аж поки час дії токену не завершиться, тоді користовач мусть заново авторизуватися.

Очевидно, що зв’язок між сервером і додатком не митєвий, для того щоб користувач не бачив екран авторизації поки система намагається авторизувати його завдяки токену то я зробив завісу загрузки, яка зникає як тільки приходить відповідь з сервера.

Після успішної авторизації користувача направляє на вікно вправ. За допомогою бокового меню користувач зможе переходити на інші сторінки, наприклад на сторінку з стравами, чи програмами. При клацані на вправу, користувач переходить на детальний опис вправи з відео гайдом та таймером. При натисканні на кнопку Start таймер запускається.

Новий користувач однак, побачить трохи іншу картину, в нього не буде підготовлений денний набір вправ, спочатку його скерує на сторінку вибору програми тренувань. При натисненні на посилання користувачу відкриється вікно сторінки вибору програми тренувань (Додаток Ж).

## 4.3. Тестування програмного забезпечення

Тестування є надзвичайно важливим етапом життєвого циклу програмного забезпечення.

### 4.3.1. Вступ

В даному документі наводяться результати тестування системи “Fittrain”. Проводилось функціональне, навантажувальне тестування. Дані види тестування дозволяють перевірити функціонування програмного продукту на рівні реалізації (коду) та на рівні інтерфейсу користувача завдяки спеціальним програмам, що забезпечують імітацію дій користувача з системою.

### 4.3.2. Розробка тестів

Для забезпечення правильної роботи системи було проведено функціональне, навантажувальне тестування.

#### 4.3.2.1. Функціональне тестування

При виконанні функціонального тестування було розроблено дванадцять тестових випадків для додатку клієнта та проведено як позитивне (з коректними вхідними даними), так і негативне (помилкові вхідні дані) тестування.

#### 4.3.2.2. Навантажувальне тестування

Під час тестування серверної частини я провів навантажувальне тестування за допомогою програми утиліти WAPT. Дане програмне забезпечення надає можливість створювати для сайту таке навантаження, ніби його одночасно використовує двадцять користувачів. Вхідні настройки для тестування:

* Максимальна кількість користувачів: 20
* Час виконання тесту: 10 хв

### 4.3.3. Функціональне тестування

Усі тестові випадки пройдено. Функціональне тестування можна вважати успішним.

Варіанти використання представлені в додатку Г

### 4.3.4. Навантажувальне тестування

Навантажувальне тестування пройшло успішно згідно автоматично згенерованого звіту програмою WAPT.

Результати навантажувального тестування представлено в додатку Д.

### 4.3.5. Критерії успішності виконання тестування

При проведенні тестування програмного продукту необхідно було забезпечити:

* Покриття тестовими випадками усіх вимог, представлених в специфікації вимог.
* Варіанти використання повинні забезпечувати позитивне та негативне тестування.
* Усі заплановані тести повинні бути виконані.

Зважаючи на вище представлені пункти можна зробити висновок, що тестування було успішним.

## 4.4. Висновок

Мета данного розділу була створити реалізацію програмної системи. А також провести її тестування.

В даному розділі відбувся опис реалізації основгих класів та компонент системи. Також було розроблено 12 тестових випадків для функціонувального тестування роботи системи і представлено результати її програмної реалізації у додатку Є. Для перевірки коректності роботи програмної системи проводилось два типи тестування: навантажувальне та функціональне. Оскільки всі тестові випадки пройшли успішно, то можна зробити висновок про успіх у виконанні тестування даної системи. Також було створено інструкцію користувача, яка представлена у додатку Е.

# РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

## 5.1. Економічна характеристика проектного рішення

Даний програмний продукт створює умови для менеджменту фітнес навантажень та дієти. Та надає зручний інтерфейс користувачам. Проводить аналіз та прогноз їх успішності.

На сьогоднішній день програми для менеджменту власного часу є досить популярними, оскільки в швидкому темпі сучасного життя люди втрачають відчуття часу і часто забувають робити речі, які запланували. Даний продукт дозволить просто і легко встановити собі режим тренувань і слідкувати за його виконанням. Практично переважна більшість людей бажають мати здоровий та гарний стан тіла. Однак, мало хто ставить собі це як найважливішу задачу на день. Саме тому фітнес додатки користуються великою популярністю, оскільки вони замінюють фізичного тренера, який обходиться досить дорого.

Слід зауважити, що основним важливим критерієм, що впливає на економічну успішність є те що додаток орієнтований на мобільні платформи. Ми добре знаємо, що в Україні, як і в більшості країн світу мобільні пристрої дещо популярніші за настільні комп’ютери чи ноутбуки. Це обумовлюється їх нижчою ціною, поширеністю і побутовою функціональністю. В той час як комп’ютери використовуються швидше для роботи. За статистикою, на 2020 рік з 7.75 мільярдів осіб зазначено 5.19 мільярдів унікальних користувачів телефонів, це сталося за рахунок того, що за останні кілька десятиліть мобільні технології зробили кілька масштабних революцій, що призвело до зниження ціни, збільшення доступності, популяризації нативних додатків, збільшення та покращення функціоналу пристроїв. За статистикою, на мобільні телефони приходиться близько 53.3% інтернет трафіку, в той час як на ноутбуки та настільні комп’ютери 44%, на планшети 2.7%, а на ігрові консолі всього 0.07%. При чому, приріст збільшується тільки на мобільні телефони.[6]

Враховуючи все вищесказане, можна заявити, що основні фактори економічної успішності розробки даного продукту це його велика цільова аудиторія, гнучкість в підтримці та впровадженні нових функцій, зручність використання і високий попит на подібні застосунки.

## 5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту

Індустрія цифрових технологій та додатків для здоров’я і фітнесу процвітає. Оскільки в часи пандемії люди все більше часу проводять вдома, їм потрібно вчитись управляти практично кожним аспектом свого життя. В тому числі фізичне здоров’я та стан свого тіла. Зараз на ринку існують додатки практично на будь яку потребу, будь то додаток для йоги і медитацій чи особистий фітнес тренер.

Найпопулярніші додатки показують те, що люди масово користуються додатками з різними можливостями, будь то йога, фітнес чи харчування. Ось список найпопулярніших додатків в розділі Health and fitness [7].

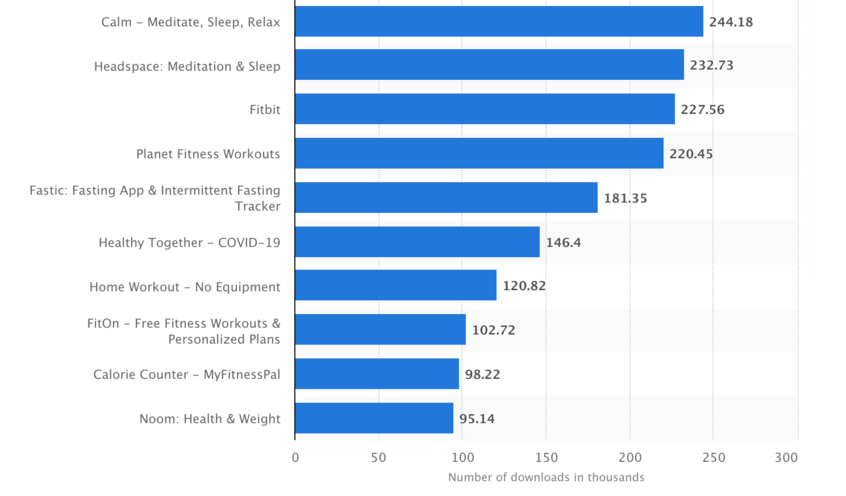


Рис.5.1. Список найпопулярніших програм для фітнесу на 2021 рік [7]

Кількість завантажень показують неймовірну популярність серед користувачів Android та iOS. Найкращі фітнес додатки мають міліони встановлень у всьому світі. Якщо подивитись на рівень доходів від найпопулярніших програм в цій сфері, то видно, що за останні роки пандемії приріст доходів перевищив пів мільярда доларів і далі стрімко росте [8].

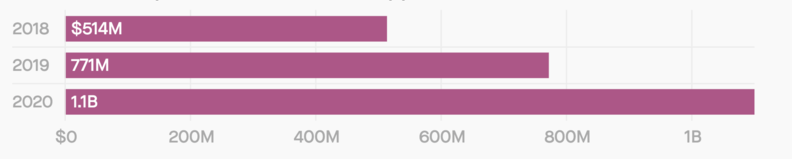


Рис.5.2. Дохід від 100 найкращих фітнес додатків за роки пандемії [8]

Більшість додатків які зараз входять в топ 100 всього кілька років назад були аутсайдерами з невеликою кількістю завантажень. Пандемія дозволила їм стати надзвичайно популярними, а їх власникам заробити немалі гроші. Як показують дослідження «Американської психологічної асоціації» 46% дорослих американців відчувають високий рівень стресу [9], саме тому багато з них шукають додатки для оздоровлення. За аналізом запитів в Google випливає, що близько 30 тисяч людей щомісяця шукають додатки для фітнесу.

Аналізуючи вищенаведену інформацію та твердження я дійшов висновку, що даний продукт завдяки зручному гарному інтерфейсу здобуде прихильність користувачів. Оскільки під час пандемії попит на додатки фітнесу зріс, можна зробити висновок, що створення такого програмного продукту доцільне та має високий шанс на успіх.

## 5.3. Оцінювання та аналізування факторів внутрішнього та зовнішнього середовища

Фактори зовнішнього середовища оцінюються за шкалою [-5; 5], при цьому межі шкали відображають максимальний негативний та позитивний вплив факторів на організацію, 0 демонструє, що фактор впливає на організацію нейтрально.

Фактори внутрішнього середовища оцінюються за шкалою [0;5], при цьому 0 демонструє нерозвиненість, відсутність чи катастрофічний стан фактора внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактора.

Сума вагомостей усіх факторів становить одиницю, тобто рівень вагомості для кожного фактора визначається за допомогою коефіцієнтів. Зважений рівень впливу факторів розраховується як добуток впливу фактора у балах та рівня вагомості [10]. Результати експертних оцінок впливу факторів зовнішнього середовища наведено у табл. 5.1.

Таблиця 5.1

Результати експертних оцінок

|  |  |  |  |
| --- | --- | --- | --- |
| Фактори | Середня експертна оцінка, бали | Середня вагомість факторів | Зважений рівень впливу, бали |
| 1 | 2 | 3 | 4 |
| *Фактори зовнішнього середовища* |  | | |
| Споживачі | 5 | 0,11 | 0,55 |
| Постачальники | 0 | 0,1 | 0 |
| Конкуренти | -3 | 0,1 | -0,3 |
| Державні органи влади | 3 | 0,05 | 0,1 |
| Інфраструктура | 1 | 0,06 | 0,06 |
| Законодавчі акти | 1 | 0,1 | 0,1 |
| Профспілки, партії та інші громадські організації | 0 | 0,05 | 0 |
| Система економічних відносин в державі | 2 | 0,06 | 0,12 |
| Організації-сусіди | -2 | 0,01 | -0,02 |
| Міжнародні події | 4 | 0,01 | 0,04 |
| Міжнародне оточення | 1 | 0,03 | 0,03 |
| Науково-технічний прогрес | 4 | 0,07 | 0,28 |

Продовження таблиці 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Соціально-культурні обставини | 4 | 0,05 | 0,2 |
| Рівень техніки та технологій | 4 | 0,04 | 0,16 |
| Особливості міжнародних економічних відносин | 0 | 0,02 | 0 |
| Стан економіки | 2 | 0,08 | 0,16 |
| Загальна сума | — | 1 | 1,24 |
| *Фактори внутрішнього середовища* |  | | |
| Цілі | 3 | 0,11 | 0,33 |
| Структура | 2 | 0,16 | 0,32 |
| Завдання | 4 | 0,07 | 0,28 |
| Технологія | 5 | 0,2 | 1 |
| Працівники | 2 | 0,21 | 0,42 |
| Ресурси | 1 | 0,25 | 0,25 |
| Загальна сума | — | 1 | 2,6 |

Результати дослідження вказують, що загальний зважений фактор зовнішнього середовища є 1.24, а значить що зовнішнє середовище позитивне. Найбільш позитивно впливають такі фактори: споживачі та науково-технічний прогрес. Розвиток додатку залежить від його успішності, а вона напряму залежить від популярності додатку серед користувачів. Оскільки додаток спрямований на вирішення побутових проблем то і аудиторія в нього значно ширша ніж у інших розрахованих на конкретну аудиторію. Однак слід зазначити, що найпотужнішим негативним фактором зовнішнього середовища є конкуренти на ринку. Це правда, оскільки додатки для фітнесу давно не новинка, а під час пандемії на ринку утворилась справжня війна компаній за користувачів.

Щодо внутрішнього середовища, зважений рівень впливу - 2,6. Найвагоміші фактори впливу тут це технологія розробки яку я використовую та рівень кваліфікації працівників. Проте рівень ресурсів досить низький.

Отже, судячи з вищенаведеного оцінювання умови для створення продукту досить таки сприятливі. Однак, варто зазначити, що існують свої ризики при виході в ринок «червоного океану».

## 5.4. Формування стратегічних альтернатив

Формування стратегічних альтернатив є основою стратегічного планування. Надзвичайно важливо обрати найоптимальніший варіант з урахування її можливостей, недоліків та переваг.

Для першої групи стратегій основними критеріями розділення стратегій є існуючий або новий продукт, а також супутні послуги. На рис. 5.3 відображені існуючі стратегії цієї групи.

**Існуюче**

**Нове**

**Стратегія розроблення нового продукту**

**Стратегія нового продукту з супутніми послугами**

**Стратегія розвитку існуючого продукту**

**Стратегія розвитку існуючого продукту**

**з супутніми послугами**

**Продукт (проектне рішення )**

**Додаткові послуги**

**Наявні**

**Відсутні**

Рис. 5.3. Стратегічні альтернативи

Стратегія розроблення нового продукту (проектного рішення) характеризується створенням абсолютно нового програмного забезпечення, яке дає змогу вирішити новоутворені потреби індивіда, певного колективу чи суспільства загалом, економіки.

Стратегія розвитку існуючого продукту (проектного рішення) означає модифікацію програмного забезпечення та його якісних характеристик.

Стратегія розвитку існуючого продукту (проектного рішення) з супутніми послугами означає запровадження на ринку модифікованого програмного забезпечення із додатковими послугами (встановлення, супроводження, коригування, адаптування до специфіки конкретного підприємства тощо).

Стратегія нового продукту (проектного рішення) з супутніми послугами означає розроблення нового програмного забезпечення та пропонування при його експлуатації додаткових послуг.

Для другої групи критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт.

**Існуючий**

**Новий**

**Стратегія розвитку продукту**

**Диверсифікація**

**Стратегія глибокого проникнення**

**продукту**

**Стратегія**

**розвитку**

**ринку**

**Продукт (проектне рішення )**

**Ринок**

**Новий**

**Існуючий**

Рис. 5.4. Стратегічні альтернативи

Основною метою стратегії глибшого проникнення є використання існуючого продукту задля збільшення частки в існуючому ринку. Така стратегія є прийнятною в тому випадку, якщо компанія володіє достатньою кількістю ресурсів для виробництва власного продукту. Однак, активне зростання на існуючому ринку призведе до акреції конкуренції. Стратегія буде тріумфальною за умови обструкції у ресурсах та потужностях суперників або стрімкому розвитку самого ринку. Варто зазначити, що кожен ринок за об’ємом має свій ліміт і якщо підприємство прагне розвиватись, то воно мусить використовувати інші запропоновані стратегії.

Стратегія розвитку ринку полягає в застосуванні існуючого продукту (програмного забезпечення) або дрібній його модифікації для виходу на свіжий сегмент ринку, весь ринок або іноземний ринок. Ця стратегія є з вищим рівнем ризику, оскільки потрібно виходити на новий ринок, де можуть бути інакші правила гри, запити та смаки споживачів тощо.

Стратегія розвитку продукту полягає в створенні нового продукту (програмного забезпечення) для існуючого сектору ринку. Ця стратегія є ризиковою, так як вимагає створення нового продукту для існуючого сегменту споживачів. Проте, якщо ринок затіває зменшувати обсяги та наявний продукт є на етапі зрілості та падіння, тоді доречно застосовувати стратегію розвитку продукту.

Стратегія диверсифікації реалізується шляхом виходу на нові сектори бізнесу. Тобто збільшення асортименту товарів, послуг тощо

Беручи до уваги вищенаведені факти я обрав для розвитку програмного продукту з першої групи стратегічних альтернатив стратегію розроблення нового продукту з супутніми послугами. Оскільки даний продукт суміщає в собі функціонал кількох повязаних додатків, він не має абсолютних аналогів на даному секторі ринку.

Щодо другої сукупності стратегічних альтернатив, треба зосередитися на стратегії розвитку продукту. Таке рішення можна аргументувати тим, що розроблюване програмне забезпечення не є паралельним до інших додатків, які існують на ринку, а включає багато нових можливостей. Отже, продукт є конкурентоздатним, а обрана стратегія розвитку сприятиме попиту серед існуючого сегменту споживачів.

## 5.5. Бюджетування

Бюджетування є комплексною обгрунтованою системою розрахунку та планування витрат, пов’язаних з випуском та реалізацією продукції, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукції, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій.

Під час планування бюджету потрібно врахувати витрати на оплату праці. Для розробки даного програмного забезпечення необхідно залучити програміста, тестера та менеджера проекту.

Таблиця 5.2

Бюджет витрат матеріалів та комплектуючих виробів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва матеріалів та комплектуючих | Марка, тип, модель | Фактична кількість, шт. | Ціна за одиницю, грн. | Разом, грн. |
| Ноутбук | Ноутбук ASUS ZenBook UM425IA-AM078 Lilac Mist | 1 | 25550 | 25550 |
| Мишка | L300 безпровідна | 1 | 250 | 250 |
| Ручка | Lexi чорна | 3 | 10 | 30 |
| Папір | XEROX, А4, 80г/м2, 500арк | 1 | 90 | 90 |
| Разом: | | | | 25920 |

Таблиця 5.3

Бюджет витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада,  спеціальність | Кількість працівників, осіб | Час роботи, дні | Денна заробітна плата працівників, грн. | Сума витрат на оплату праці, грн. |
| Основна заробітна плата | | | | |
| Програміст | 1 | 24 | 450 | 10800 |
| Тестер | 1 | 15 | 300 | 4500 |
| Менеджер | 1 | 24 | 300 | 7200 |
| Разом: | | | | 22500 |

Таблиця 5.4

Бюджет обов’язкових відрахувань та податків

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Посада, спеціальність | Сума основної заробітної плати | Сума додаткової заробітної плати | Разом витрат на оплату праці | Сума військового збору, грн. | Сума податку з доходів фізичних осіб, грн. |
| Програміст | 10800 | - | 10800 | 162 | 1836 |
| Тестер | 4500 | - | 4500 | 75 | 765 |
| Менеджер | 7200 | - | 7200 | 108 | 1224 |
| Разом: | 22500 | - | 22500 | 345 | 3825 |

Таблиця 5.5

Бюджет загальновиробничих витрат

|  |  |
| --- | --- |
| Статті витрат | Сума, грн. |
| Змінні загальновиробничі витрати, у т.ч.: | |
| Заробітна плата допоміжного персоналу | 1500 |
| Витрати на МШП | 100 |
| Витрати на електроенергію та технологічні цілі | 600 |
| Разом змінних витрат: | 2200 |
| Постійні загальновиробничі витрати, у т.ч.: |  |
| Комунальні послуги | 800 |
| Витрати на оренду | 5000 |
| Витрати на ремонт | 300 |
| Разом постійних витрат: | 6100 |
| Разом загальновиробничих витрат: | 8300 |

Таблиця 5.6

Бюджет адміністративних витрат та витрат на збут

|  |  |
| --- | --- |
| Статті витрат | Сума, грн. |
| Адміністративні витрати, у т.ч.: | |
| Заробітна плата адміністративного персоналу | 1500 |
| Витрати на МШП | 100 |
| Витрати на ремонт | 300 |
| Витрати на сплату податків і зборів | 3599 |
| Знос адміністративного обладнання | 245 |
| Разом адміністративних витрат: | 5744 |
| Витрати на збут, у т.ч.: |  |
| Заробітна плата менеджерів зі збуту | 1600 |
| Витрати на гарантійний ремонт | 200 |
| Витрати на налагодження і експлуатацію | 2340 |
| Витрати на рекламу | 15000 |
| Разом витрат на збут: | 17340 |

Таблиця 5.7

Зведений кошторис витрат на розробку проектного рішення

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статті витрат | Одиниці виміру | Фактична кількість, шт. | Ціна одиниці, грн. | Разом, грн. |
| Сировина і матеріали  Ноутбук  Мишка  Ручка  Папір | шт | 5  1  1  3  1 | 25550  250  10  90 | 25920 |
| Купівельні напівфабрикати та комплектуючі вироби | Грн | - | - | - |
| Зворотні відходи (вираховуються) | Грн | - | - | - |
| Паливо та електроенергія на технологічні цілі | кВт/год | 400 | 0,99 | 396 |
| Основна заробітна плата | Грн | - | 22500 | 22500 |
| Додаткова заробітна плата | Грн | - | - | - |
| Відрахування на соціальне страхування | Грн | - | 5170 | 5170 |
| Витрати на утримання й експлуатацію устаткування | Грн | - | 200 | 200 |
| Загальновиробничі витрати, у т.ч.: |  |  |  |  |
| - змінні; | Грн | - | 2200 | 2200 |
| - постійні; | Грн | - | 6100 | 6100 |
| Разом виробничих витрат: | Грн | - | - | 62486 |
| Адміністративні витрати | Грн | - | - | 5744 |
| Витрати на збут | Грн | - | - | 17340 |

Продовження таблиці 5.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Інші операційні витрати | Грн | - | - | - |
| Разом виробничих і операційних витрат: | Грн | - | - | 85570 |

Для визначення фінансових результатів, необхідно розрахувати вартість (ціну) продукту, який розробляється. Ціна визначається на основі суми виробничих і операційних витрат з урахуванням рентабельності виробництва.

Ц = СБ \* Р + СБ

де Ц – ціна програмного продукту, грн.

СБ – собівартість програмного продукту, грн.

Р – рентабельність, %(25)

Ц = 62486 \* 0,25 + 62486 = 78107,5 грн

Дохід від реалізації = Ц \* кількість = 78107,5 \* 5 = 390537,5 грн

ПДВ = Ц \* 20% = 78107,5 \* 0,2 = 15621,5 грн

Чистий дохід = ДР – ПДВ = 390 537,5 - 15621,5 = 374916 грн

Валовий прибуток = ЧД –СБ = 374916 – 62486 = 312430 грн

Фінансовий результат від операційної діяльності = ВП –АВ –ВнаЗБ = 312430 - 5744-17340= 289346грн

Податок на прибуток = ФР \* 19% = 289346\*0,19 =52082,28 грн

Чистий прибуток = 289346-52082,28= 237263,72 грн

Таблиця 5.8

Бюджет фінансових результатів

|  |  |
| --- | --- |
| Показники | Сума, грн. |
| Дохід від реалізації продукції | 390537,5 |
| Податок на додану вартість (20%) | 15621,5 |
| Чистий дохід від реалізації продукції | 374916 |
| Собівартість реалізованої продукції | 62486 |
| Валовий прибуток | 312430 |
| Операційні витрати | |

Продовження таблиці 5.8

|  |  |
| --- | --- |
| Адміністративні витрати | 5744 |
| Витрати на збут | 17340 |
| Фінансовий результат від операційної діяльності | 289346 |
| Податок на прибуток (19%) | 52082,28 |
| Чистий прибуток (збиток) | 237263,72 |

## 5.6. Висновки до економічної частини

Зробивши економічний аналіз, я зробив ряд висновків про доцільність створення програмного продукту для менеджменту фітнес навантажень та дієти. Зважаючи на бурхливий розвиток технологій для створення мобільних додатків та акреції попиту на фітнес додатки у зв’язку з пандемією, передбачається, що дана система швидко здобуде популярність серед цільової аудиторії при правильній піар кампанії.

Оскільки програмний продукт не повторює інші фітнес додатки абсолютно чи вдосконалює функціонал уже існуючої системи, а має нові можливості, зокрема прогнозування результатів тренувань, то я обрав стратегію розвитку нового продукту з супутнім. Супутні послуги в даному випадку це супровід програмного продукту, додавання нових програм тренувань, розширення функціоналу.

Щодо зовнішніх факторів середовища, що позитивно впливають на прогрес та ініціацію даного програмного забезпечення, особливо важливими є споживачі, загальний стан економіки та рівень розвитку техніки і технологій. Позитивними факторами внутрішнього середовища є цілі та працівники. Тому варто слідувати ідеям та потребам користувачів і покращувати існуючий продукт у цьому напрямі.

Найбільш кардинальний негативний ефект серед факторів зовнішнього середовища займають конкуренти, тому надзвичайно важливим є регулярне вдосконалення системи.

# ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було проаналізовано важливість та актуальність використання мобільних технологій для галузей оздоровлення та фітнесу. Для визначення доцільності розробки мобільного додатку для управляння побутом та фітнес навантаженями було проаналізовано існуючі аналоги. Також було проведено тестування програмного забезпечення та збережено його результати.

Зважаючи на результати дослідження предметної області та аналізу аналогів, було визначено, що на даний момент існує багато схожих систем, але не знайдено систем, які б охоплювали увесь функціонал разом. А зважаючи на те, що з періодом пандемії попит та популярність додатків для фітнесу виросла, було визначено, що створення даної системи є доцільним. У ході проектування було створено спеціифікацію вимог та також обрано архітектуру системи. Слід зазначити, що система продумана так, щоб в майбутньому було легко і доцільно додавати новий функціонал, та робити покращення старого.

Під час програмної реалізації системи використовується архітектурний шаблон клієнт-сервер з використанням технологій ASP .NET WEB API та React Native.

Було проведено два типи тестування, а саме функціональне та навантажувальне. Усі тести пройдено успішно.

# ЛІТЕРАТУРНІ ДЖЕРЕЛА

1. Seven – 7-хвилинні тренування. – [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://play.google.com/store/apps/details?id=com.popularapp.sevenmins&hl=uk&gl=US.
2. Домашні Тренування – без Додаткового Спорядження. – [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://captain-droid.com/uk/apps/sport-fitness/trenirovki-dlya-doma-nikakogo-oborudovaniya.
3. Чистая архитектура. Искуство разработки програмного обеспечения – Роберт Мартін - 2018. – 352с.
4. MarkLogic Server – [Електронний ресурс]. – Режим доступу: https://docs.marklogic.com/guide/rest-dev.pdf.
5. Alex Banks, Eve Porcello. Learning React: Functional Web Development with React and Redux. Printed in the United States of America.Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.; 2017. – First Edition – 320с.
6. Самые важные цифры из глобального отчета Digital 2020 – [Електронний ресурс]. – Режим доступу: <https://www.web-canape.ru/business/internet-2020-globalnaya-statistika-i-trendy/>.
7. Стан індустрії додатків для здоров’я та оздоровлення у 2021 році – [Електронний ресурс]. – Режим доступу: <https://brainberry.ua/uk/newsroom/blog/popularity-of-health-and-wellness-apps>.
8. Why there’s so much money in mindfulness – [Електронний ресурс]. – Режим доступу: <https://qz.com/1958593/covid-19-accelerated-the-worlds-embrace-of-meditation-apps/>.
9. Stress in the Time of COVID-19, Volume One mindfulness – American Psychological Associoation - 2020. – 1-3c.
10. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ВИКОНАННЯ ЕКОНОМІЧНОЇ ЧАСТИНИ ВИПУСКНОЇ БАКАЛАВРСЬКОЇ РОБОТИ - Воськало В.І. канд. екон. наук, доц. – Львів: Національний університет «Львівська політехніка» - 2011. – 14 с.
11. Frank Zammetti Practical React Native. – Pottstown, PA, USA – apress inc. – 2018. – 334с.
12. Bonnie Eisenman. Learning React Native. Printed in the United States of America.Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.; 2017. – Second Edition – 225с.
13. Jonathan Lebensold. React Native Cookbook. Printed in the United States of America.Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.; 2018. – First Edition – 160с.
14. Jamie Kurtz, Brian Wortman. ASP.NET Web API 2: Building a REST Service from Start to Finish. – apress inc. – 2014. – Second Edition – 256с.
15. Лавріщева К. М. Програмна інженерія: підручник / К. М. Лавріщева. – К.: Академперіодика, 2008. – 320с.
16. Кузьмін О.Є., Мельник О.Г. Менеджмент. Конспект лекцій: Навчальний посібник. – Львів: Видавництво Національно Університету “Львівська політехніка”, 2002. - 132с.

# ДОДАТОК А. Діаграма варіантів використання, розгортання та компонент

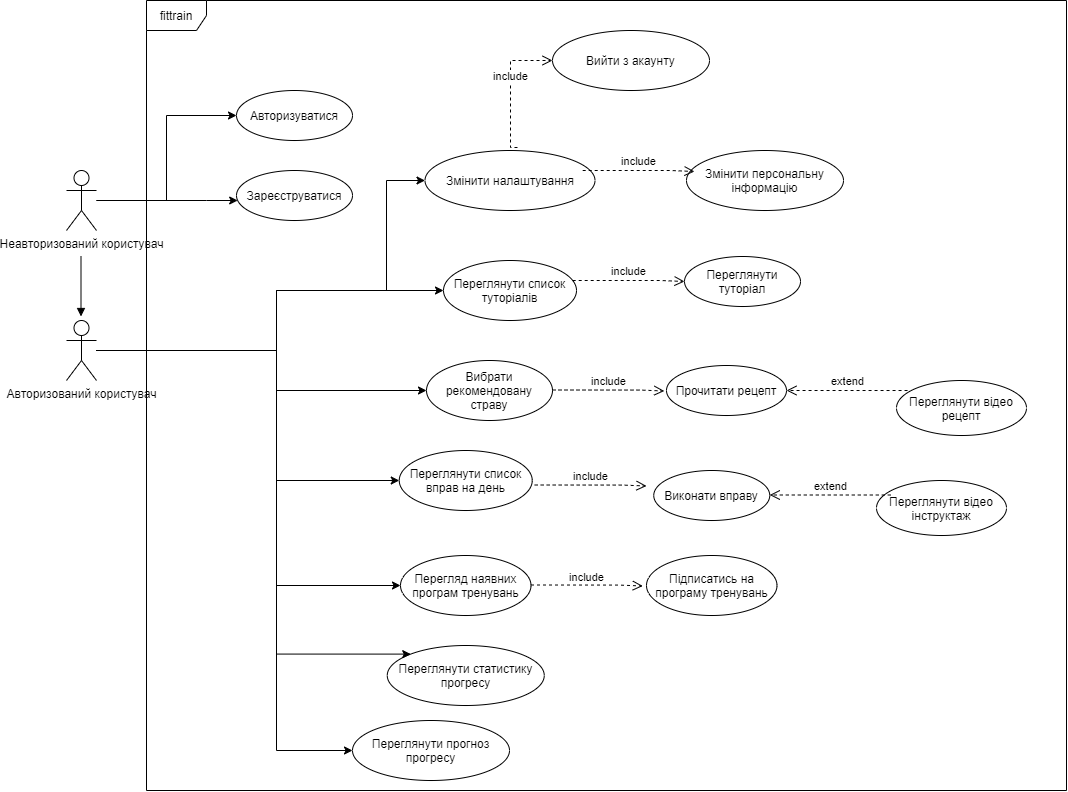


Рис. А.1. Діаграма прецедентів

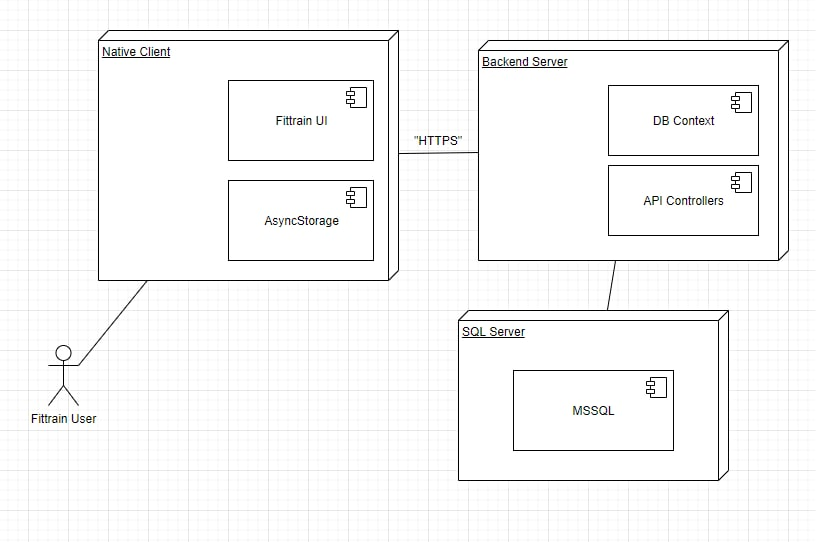


Рис. А.2. Діаграма розгортання

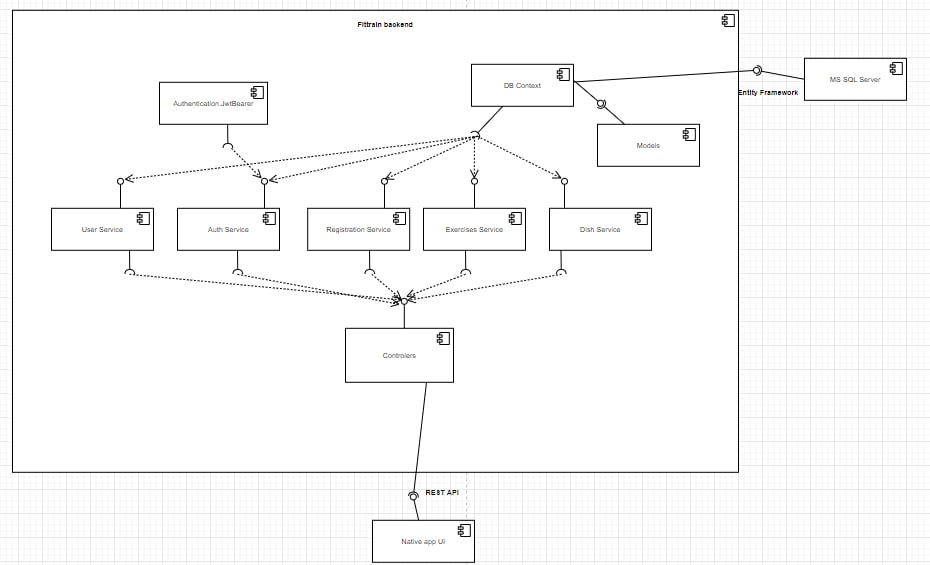
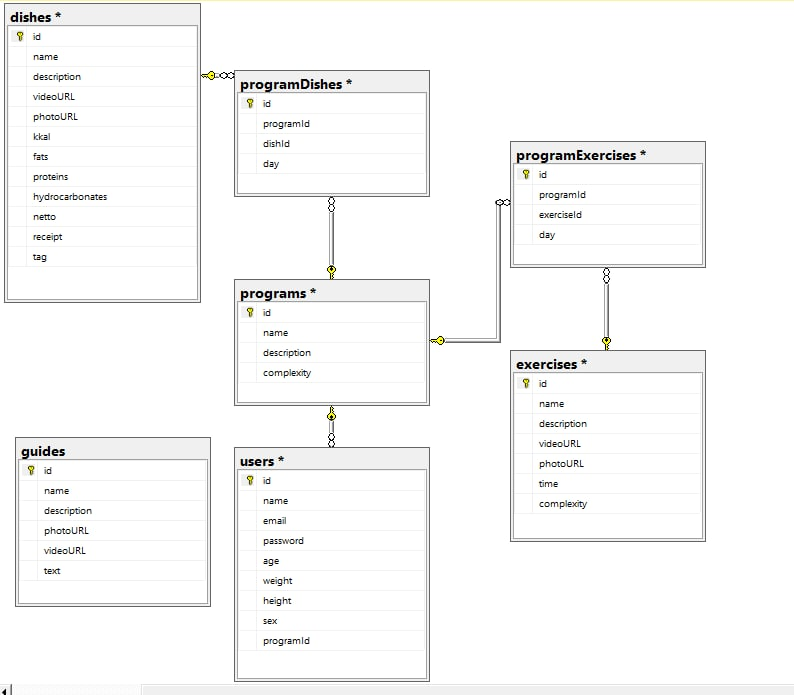


Рис. А.3. Діаграма компонент

# ДОДАТОК Б. Фізична схема бази даних



# ДОДАТОК В. Скрипт створення бази даних

USE [master]

GO

/\*\*\*\*\*\* Object: Database [FITTRAINBD] Script Date: 31.05.2021 19:23:19 \*\*\*\*\*\*/

CREATE DATABASE [FITTRAINBD]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'FITTRAINBD', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER01\MSSQL\DATA\FITTRAINBD.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'FITTRAINBD\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER01\MSSQL\DATA\FITTRAINBD\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

WITH CATALOG\_COLLATION = DATABASE\_DEFAULT

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [FITTRAINBD].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [FITTRAINBD] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [FITTRAINBD] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [FITTRAINBD] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [FITTRAINBD] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [FITTRAINBD] SET ARITHABORT OFF

GO

ALTER DATABASE [FITTRAINBD] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [FITTRAINBD] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [FITTRAINBD] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [FITTRAINBD] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [FITTRAINBD] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [FITTRAINBD] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [FITTRAINBD] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [FITTRAINBD] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [FITTRAINBD] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [FITTRAINBD] SET DISABLE\_BROKER

GO

ALTER DATABASE [FITTRAINBD] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [FITTRAINBD] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [FITTRAINBD] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [FITTRAINBD] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [FITTRAINBD] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [FITTRAINBD] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [FITTRAINBD] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [FITTRAINBD] SET RECOVERY FULL

GO

ALTER DATABASE [FITTRAINBD] SET MULTI\_USER

GO

ALTER DATABASE [FITTRAINBD] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [FITTRAINBD] SET DB\_CHAINING OFF

GO

ALTER DATABASE [FITTRAINBD] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [FITTRAINBD] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [FITTRAINBD] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [FITTRAINBD] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

ALTER DATABASE [FITTRAINBD] SET QUERY\_STORE = OFF

GO

ALTER DATABASE [FITTRAINBD] SET READ\_WRITE

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[exercises] Script Date: 31.05.2021 19:28:37 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[exercises](

[id] [int] NOT NULL,

[name] [nvarchar](50) NOT NULL,

[description] [nvarchar](2000) NULL,

[videoURL] [nvarchar](250) NULL,

[photoURL] [nvarchar](250) NULL,

[time] [int] NULL,

[complexity] [int] NULL,

CONSTRAINT [PK\_exercises] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[dishes] Script Date: 31.05.2021 19:28:51 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[dishes](

[id] [int] NOT NULL,

[name] [nvarchar](50) NOT NULL,

[description] [nvarchar](2000) NULL,

[videoURL] [nvarchar](250) NULL,

[photoURL] [nvarchar](250) NULL,

[kkal] [decimal](18, 2) NOT NULL,

[fats] [decimal](18, 2) NULL,

[proteins] [decimal](18, 2) NULL,

[hydrocarbonates] [decimal](18, 2) NULL,

[netto] [decimal](18, 2) NULL,

[receipt] [nvarchar](2000) NULL,

[tag] [nvarchar](50) NULL,

CONSTRAINT [PK\_dishes] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[guides] Script Date: 31.05.2021 19:29:11 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[guides](

[id] [int] NOT NULL,

[name] [nvarchar](100) NOT NULL,

[description] [nvarchar](200) NULL,

[photoURL] [nvarchar](250) NULL,

[videoURL] [nvarchar](250) NULL,

[text] [nvarchar](3500) NULL,

CONSTRAINT [PK\_guides] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[programs] Script Date: 31.05.2021 19:29:23 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[programs](

[id] [int] NOT NULL,

[name] [nvarchar](50) NOT NULL,

[description] [nvarchar](2000) NULL,

[complexity] [int] NULL,

[photo\_URL] [nvarchar](250) NULL,

CONSTRAINT [PK\_programs] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[users] Script Date: 31.05.2021 19:29:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[users](

[id] [int] NOT NULL,

[name] [nvarchar](50) NOT NULL,

[email] [nvarchar](200) NOT NULL,

[password] [nvarchar](200) NOT NULL,

[age] [int] NULL,

[weight] [decimal](18, 2) NULL,

[height] [decimal](18, 2) NULL,

[sex] [int] NULL,

[programId] [int] NULL,

CONSTRAINT [PK\_users] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[programExercises] Script Date: 31.05.2021 19:29:54 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[programExercises](

[id] [int] NOT NULL,

[programId] [int] NOT NULL,

[exerciseId] [int] NOT NULL,

[day] [int] NOT NULL,

CONSTRAINT [PK\_programExercises] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [FITTRAINBD]

GO

/\*\*\*\*\*\* Object: Table [dbo].[programDishes] Script Date: 31.05.2021 19:30:02 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[programDishes](

[id] [int] NOT NULL,

[programId] [int] NOT NULL,

[dishId] [int] NOT NULL,

[day] [int] NOT NULL,

CONSTRAINT [PK\_programDishes] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

# ДОДАТОК Г. Варіанти використання

FT1. Авторизація користувача в системі.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач вперше запускає клієнт програми | Відкривається вікно авторизації |
|  | Користувач вводить коректні дані, логін, пароль і натискає кнопку “увійти”. | Закривається вікно авторизації і відкривається головне вікно програми. |

FT2. Авторизація користувача з неправильним логіном.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач вперше запускає клієнт програми | Відкривається вікно авторизації |
| 2. | Користувач вводить не коректні дані, неіснуючий логін і пароль і натискає кнопку увійти. | Користувачу виводиться повідомлення про те, що введені дані недійсні. |

FT3. Авторизація користувача з неправильним паролем.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач вперше запускає клієнт програми | Відкривається вікно авторизації |
| 2. | Користувач вводить не коректні дані, існуючий логін і неіснуючий пароль і натискає кнопку увійти. | Користувачу виводиться повідомлення про те, що введені дані недійсні. |

FT4. Відкриття та закриття бокового меню.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Дія | | Очікуваний результат |
|  | Користувач успішно входить в систему та відкриває бокове меню за допомогою кнопки меню. | Відкривається бокове меню, при повторному натисненні закривається. | |
|  | Користувач повторює спробу відкрити бокове меню, на цей раз за допомогою слайду по екрану зліва направо. | Відкривається бокове меню, при слайді зправа наліво або при натисненні на сіру зону меню закривається. | |

FT4. Виконання вправ.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач входить на вкладку вправ. | Залежно від дня тижня система повертає йому набір вправ для виконання на день |
|  | Користувач вибирає вправу, переглядає відео інструкцію та натискає кнопку «Start» | Відео відтворюється з хостингу Ютуб. При натисненні на кнопку запускається зворотній відлік, при повторному натисканні відлік ставиться на паузу. По закінченню відліку користувач отримує сповіщення. |

FT5. Зміна програми тренувань.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач авторизований та переходить на вкладку вибору програм. | Відкривається вкладка з списком програм. |
|  | Користувач вибирає програму та натискає кнопку «Subscribe». | На сервер відправляється запит на зміну програми, якщо програма є в базі даних при успішній зміні повертається значення true, інакше false. |

FT6. Перегляд інформації про страву.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач входить на вкладку дієти. | Залежно від дня тижня система повертає йому набір страв на день. |
|  | Користувач вибирає страву, переглядає відео рецепт або перечитує інформацію нижче, таку як кількість калорій, опис, рецепт і тд. | Відео відтворюється з хостингу Ютуб. Інформація відображається коректно, якщо відео-рецепту не існує має бути фотографія на його місці. |

FT7 Перегляд інформації про прогрес.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач заходить на вкладку «Stats» | На вкладці представляється інформація по дефіциту калорій та 2 графіки, один з прогнозом, а інший показує кількість виконаних або не виконаних вправ на день. |

FT8. Зміна особистої інформації.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач в вкладці налаштування переходить на зміну особистої інформації. | Відкривається вкладка з полями для зміни особистої інформації. |
| 2. | Користувач заповнює відповідні поля коректною інформацією та натискає зберегти. | На сервер відправляється запит на зміну інформації користувача, якщо запит успішний то виводиться повідомлення про те, що інформація змінена. |

FT9. Зміна особистої інформації некоректна.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач в вкладці налаштування переходить на зміну особистої інформації. | Відкривається вкладка з полями для зміни особистої інформації. |
|  | Користувач заповнює відповідні поля некоректною інформацією та натискає зберегти. | Валідація даних відбувається на клієнті і користувач отримує візуальні підказки де неправильно вписані дані. |

FT10. Переглаяд туторіалів.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач переходить на вкладку туторіалів. | Відкривається івкладка з туторіалами. |
| 2. | Користувач вибирає туторіал. | Відкривається вкладка з детальною інформацією про туторіал та відео плеєром. |

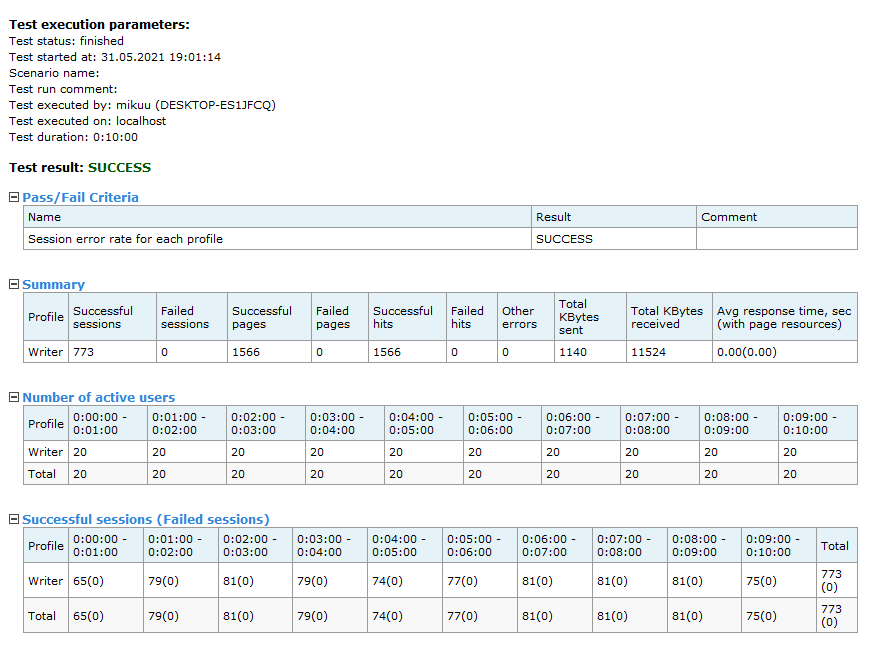
FT11. Автоматична авторизація при наявному валідному токені.

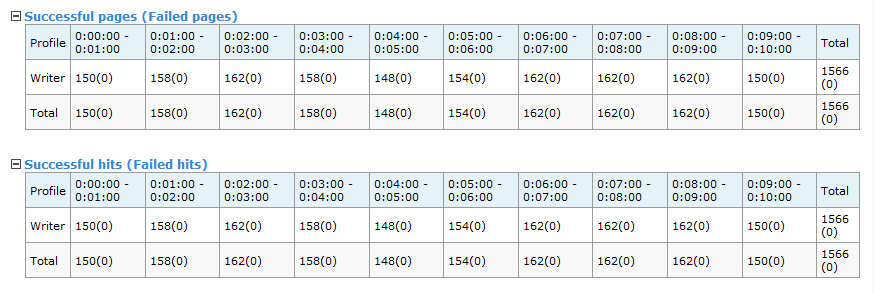
|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач авторизується, закриває програму та відкриває заново. | Система автоматично авторизує користувача. |

FT12. Перехід на сторінку авторизації при невалідному токені.

|  |  |  |
| --- | --- | --- |
|  | Дія | Очікуваний результат |
|  | Користувач авторизується, закриває програму, чекає час придатності токена та відкриває заново. | Система автоматично не авторизує користувача, а переносить його на сторінку авторизації. |

# ДОДАТОК Д. Звіт про навантажувальне тестування





# ДОДАТОК Е. Інструкція користувача

1. **Компоненти програмного забезпечення**

Веб-система розділена на три частини, серверна написана на мові програмування C# на платформі .Net Core 3.0.1 в середовищі програмування Visual Studio 2019 та призначена для виконання на ОС Windows.

Для правильного функціонування серверу комп’ютер повинен володіти наступними характеристиками:

* Операційна система Linux або Windows.
* Платформа .NET Core 3.0.1.
* Об'єм ОЗП не менше 4096МБ.
* Сервер баз даних – MS Sql.

Передбачається, що серверна програма звертатиметься до бази даних розміщеної на локальному сервері, тому необхідно встановлювити Sql Server 2019 на локальній машині.

1. **Встановлення ПЗ**

Для встановлення серверної частини необхідно для початку відтворити базу даних на Sql Server 2019. Для цього можна відновити її з .bak файлу, або перестворити використавший скрипт створення бази даних, а потім скрипт наповнення. Після чого в файлі FITTRAINBDContext.cs необхідно поміняти назву сервера бд в стрічці зв’язку. Для запуску серверу достатньо налаштувати потрібний адрес та заупстити .exe файл. Для використання системи необхідний доступ до інтернету. Для встановлення клієнту достатньо загрузити на телефон apk файл та відкрити його.

1. **Базові функції ПЗ**
   1. **Базові функції фітнес застосунку.**

Для того, щоб користуватись додатком необхідно пройти реєстрацію, вказати свій вік, вагу, ріст, стать, електронну пошту, пароль та ім’я. Після чого, система автоматично авторизує вас. При наступному вході необхідно буде вводити електронну пошту та пароль.

Для навігації по додатку зробіть свайп вправо, або натисніть на значок меню в верхньому правому куті екрану. Після чого відкриється бокове меню з вкладками: «Exercises», «Diet», «Stats», «Programs», «Guides», «Settings».

Для того, щоб користуватись функціоналом необхідно перейти на сторінку програм тренувань, вибрати програму яка підходить під ваші потреби та підписатись на неї натиснувши кнопку «Subscribe». Після цього в розділі вправи та дієта з’являться щоденні вправи та рекомендовані страви. Інформація оновлюється кожен день сервером автоматично.

Ви можете слідкувати за своїм прогресом та дивитись на прогноз результатів на вкладці «Stats». Вкладка «Guides» надасть вам відео посібники з вправ та інших тем. Для того щоб виконати вправу необхідно вибрати її в вкладці вправ та натиснути на кнопку «Start», яка почне таймер.

Для того щоб вийти з акаунту достатньо зайти на вкладку «Settings» прогорнути вниз та натиснути на «Log out».

1. **Аналіз помилок**

При неправильній роботі програми слід перевірити з’єднання з мережею Інтернет.

# ДОДАТОК Є. Основні класи та компоненти програми

Код навігатора:

import React from "react";

import { Entypo } from "@expo/vector-icons";

import { MaterialIcons } from '@expo/vector-icons';

import { Ionicons } from "@expo/vector-icons";

import { AntDesign } from "@expo/vector-icons";

import { Feather } from "@expo/vector-icons";

import { useSelector, useDispatch } from "react-redux";

import { MaterialCommunityIcons } from "@expo/vector-icons";

import {

ExercisesScreen,

StatsScreen,

SettingsScreen,

GuidesScreen,

} from "../screens";

import DetailScreen from "./DetailScreen";

import DietScreen from "./DietScreen";

import ProgramsScreen from "./ProgramsScreen";

import { Button, View, Text, Image, ActivityIndicator } from "react-native";

import { NavigationContainer, DrawerActions } from "@react-navigation/native";

import { createDrawerNavigator } from "@react-navigation/drawer";

import { createStackNavigator } from "@react-navigation/stack";

import SideBar from "../components/SideBar";

import LoginScreen from "./LoginScreen";

import RegisterScreen from "./RegisterScreen";

import store from "../redux/store.js";

import AsyncStorage from "@react-native-async-storage/async-storage";

const Drawer = createDrawerNavigator();

function MyDrawer() {

return (

<Drawer.Navigator drawerContent={(props) => <SideBar {...props} />}>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<Entypo name="battery" size={24} color={color} />

),

}}

name="Exercises"

component={ExercisesScreen}

/>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<MaterialIcons name="restaurant-menu" size={24} color={color} />

),

}}

name="Diet"

component={DietScreen}

/>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<Ionicons name="ios-stats-chart" size={24} color={color} />

),

}}

name="Stats"

component={StatsScreen}

/>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<MaterialCommunityIcons name="arm-flex" size={24} color={color} />

),

}}

name="Programs"

component={ProgramsScreen}

/>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<AntDesign name="playcircleo" size={24} color={color} />

),

}}

name="Guides"

component={GuidesScreen}

/>

<Drawer.Screen

options={{

drawerIcon: ({ color }) => (

<Feather name="settings" size={24} color={color} />

),

}}

name="Settings"

component={SettingsScreen}

/>

<Drawer.Screen

name="Details"

component={DetailScreen}

initialParams={{

type: "ex",

id: 0,

videoUrl: "",

photoUrl: "",

description: "",

time: 0,

complexity: 0,

}}

/>

</Drawer.Navigator>

);

}

export default function Main() {

const [isLoading, setIsLoading] = React.useState(true);

const logged = useSelector(state => state.userReducer.jwt);

React.useEffect(() => {

const getUserInfo = async (userToken) => {

var myHeaders = new Headers();

myHeaders.append("Content-Type", "application/json");

myHeaders.append(

"Authorization",

`Bearer ${userToken}`

);

var requestOptions = {

method: "POST",

headers: myHeaders,

redirect: "follow",

};

let status = 500;

await fetch("https://localhost:44310/api/login/info", requestOptions)

.then((response) => response.text())

.then((result) => {

if (JSON.parse(result).id) {

console.log("start", result);

const info = JSON.parse(result);

if (info != undefined) {

store.dispatch({

type: "LOG\_IN",

id: info.id,

jwt: userToken,

name: info.name,

email: info.email,

age: info.age,

weight: info.age,

height: info.height,

sex: info.sex,

program: info.programId,

});

AsyncStorage.setItem("userToken", userToken);

console.log("reduxAsync");

status = 200;

} else {

store.dispatch({ type: "LOG\_OUT" });

console.log("reduxout");

AsyncStorage.setItem("userToken", null);

status =401;

}

} else return status;

})

.catch((error) => console.log("error", error));

return status;

};

const retrieve = async () => {

try {

const userToken = await AsyncStorage.getItem("userToken");

console.log("async", userToken);

if (userToken == "null") {

store.dispatch({ type: "LOG\_OUT" });

console.log("asynclogout", userToken);

console.log(store.getState().userReducer.jwt);

} else {

await getUserInfo(userToken);

}

} catch (e) {

console.log(e);

}

};

retrieve();

setTimeout(() => {

setIsLoading(false);

}, 1000);

}, []);

if (isLoading) {

return (

<View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>

<ActivityIndicator size="large" />

</View>

);

}

const AuthStack = createStackNavigator();

return (

<NavigationContainer>

{logged != "" ? (

<MyDrawer></MyDrawer>

) : (

<AuthStack.Navigator

screenOptions={{

headerShown: false,

}}

>

<AuthStack.Screen name={"Login"} component={LoginScreen} />

<AuthStack.Screen

name={"Registration"}

component={RegisterScreen}

/>

</AuthStack.Navigator>

)}

</NavigationContainer>

);

}

# ДОДАТОК Ж. Прототипи інтерфейсу програми

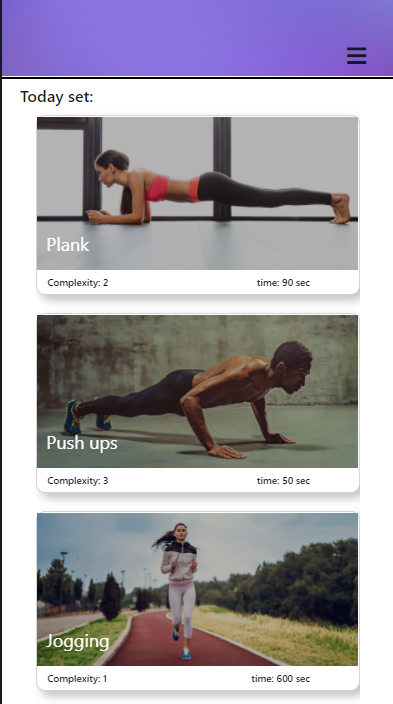
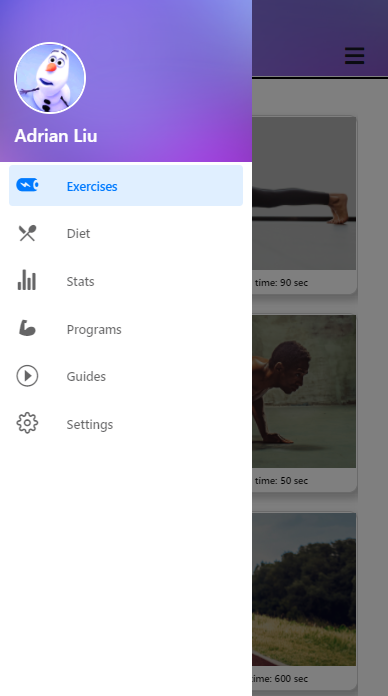


Рис. Ж.1. Вкладка вправ

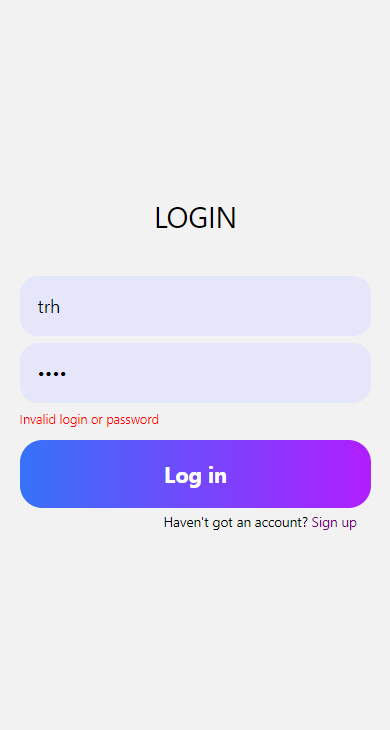
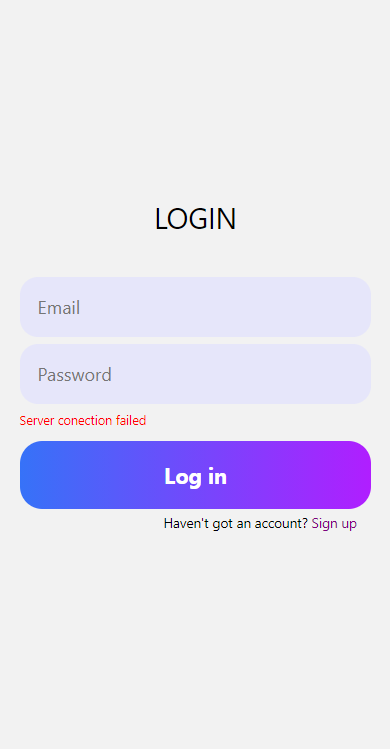
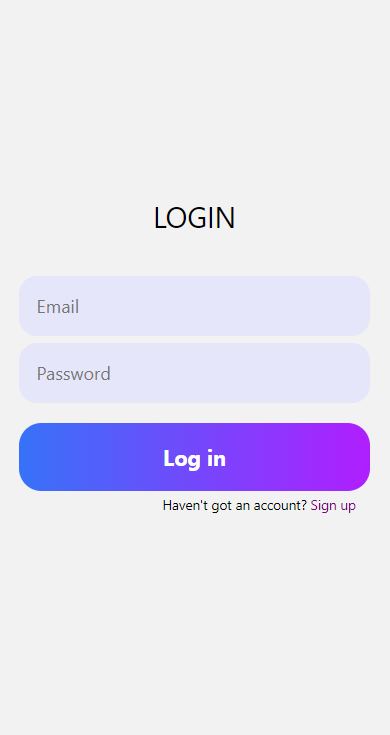


Рис. Ж.2. Вікно авторизації (випадок при відсутньому зв’язку з сервером та при невірних даних)

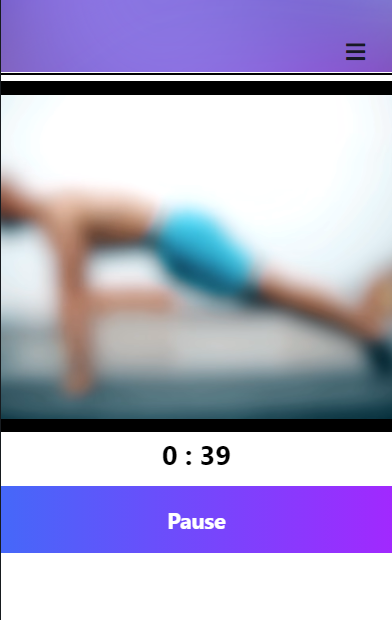
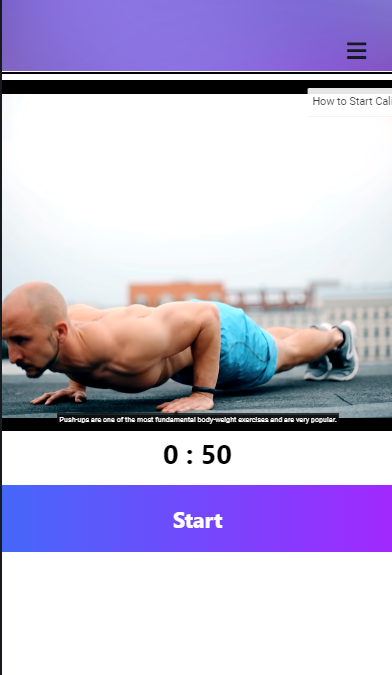


Рис. Ж.3. Сторінка виконання вправи

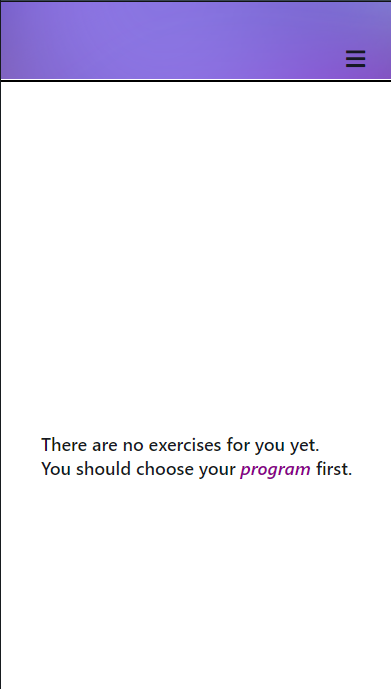


Рис. Ж.4. Вкладка вправ при новому користувачі

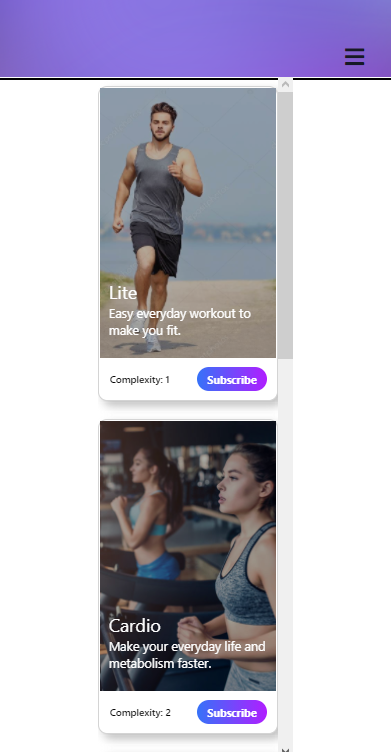


Рис. Ж.5. Вкладка програм тренувань