Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка системи аналітики прогресу фізичних навантажень І спортивних результатів з використанням фреймворку Spring

Виконав(ла): студентка 4 курсу, групи СП-41
спеціальності 121
«Інженерія програмного забезпечення»
(шифр і назва спеціальності)

	(підпис)	Сороківська Н.В. (прізвище та ініціали)
Керівник	(підпис)	Мудрик І.Я. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Яцишин В.В. (прізвище та ініціали)

Тернопіль 2023

РЕФЕРАТ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок 60, таблиць 1, рисунків 35, додатків 3, презентація.

Тема: Розробка системи аналітики прогресу фізичних навантажень і спортивних результатів з використанням фреймворку Spring

В кваліфікаційній роботі бакалавра висвітлено алгоритми розробки програмного продукту, використовувані для реалізації шаблони та методології, а також результати проектування та імплементації вибраних стратегій у програмі. Описано та аргументовано вибір засобів для втілення ідеї у системне рішення.

Ключові слова: програмне рішення, інформаційна система, застосунок, аналіз, аналітика, проектування, розробка.

ANNOTATION

Bachelor's certification work. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2023. Pages 60, tables 1, figures 35, appendices 3, presentation.

Topic: Development of a system for analyzing physical activity progress and sports results using Spring framework.

The bachelor's attestation work highlights the algorithms used for developing the software product, as well as the patterns and methodologies used for its implementation. It`s also describes and justifies the selection of tools for realizing the idea in a program solution.

Keywords: Software solution, information system, application, analysis, analytics, design, development.

ЗМІСТ

ВСТУП

Наразі, в усіх сферах та галузях використовуються програми для автоматизації та спрощення зберігання і аналізу об’ємів даних, так, для кожної групи завдань, використовуються наявні ресурси аби спростити та покращити взаємодію користувачів із інструментами для досягнення бажаних результатів, не виключенням є і сфера здоров’я і спорту.

Основні вимоги для практичності та доцільності застосування якогось продукту це його масштабованість для широкого сегменту індивідів, доступність незалежно від місця перебування чи використовуваних платформ та зрозумілість для різних категорій від користувачів до професіоналів. Проектуючи ці пункти на пропонований до реалізації продукт, виокремлюються наступні критичні аспекти: необхідність забезпечення функціоналу, для отримання за запитом користувача, ефективно та у зручній формі відповідей, обробляти та зберігати інформацію у базі даних на віддаленому пристрої, в сучасному світі виконання процесів з обробки та зберігання файлів механічно, тобто вручну, не буде ефективним та продуктивним, тому людство у своєму розвитку наблизилось до того, що Інтернет став основним генератором світових тенденцій; уможливлення доступу до широкого спектру варіантів використання застосунку, наявність багатьох платформ, пристроїв, операційних систем, браузерів тощо вимагає від пропонованих рішень, забезпечення їх відображення на кожній із обраних користувачем платформі, відповідність продукту таким вимогам свідчитиме про клієнт орієнтованість та практичність рішення, що забезпечить його популярність; адаптивність до персональних аспектів застосування для кожного із користувачів, не зважаючи на множину тенденцій та їх постійний розвиток у охоплюваних сферах, передбачення практичної більшості варіантів застосування додатку підтримуватиме постійну активність у його використанні; презентабельність та зручність у взаємодії, послідовність й логічність відтворюваних операцій в інформаційній системі для

забезпечення якості й поширення ідеї в різних спільнотах та групах суспільства, без прив’язаності та орієнтованості на окремі спеціалізації чи професійні знання.

Враховуючи вищеописане, розробляється такий підхід, який дозволить зацікавленим у даному рішенні, не перейматись місцем розташування даних, використовуваним обсягом пам’яті та іншими аспектами, що стосуються збереження даних. Також результат – економія часу для роботи із даними та передачею їх між пристроями. Багато процесів можуть використовувати програму одночасно, звертаючись до однієї доступної системи для роботи із функціоналом з власного чи довільного пристрою, забезпечуючи безпеку через механізми автентифікації та авторизації. Інші варіанти виконання передбачають тривалий процес передачі та роботи, що у свою чергу характеризується використанням різних програм, які можуть мати лімітовані значення, зберігати дані в потенційно небезпечному місці, що в крайньому випадку призведе до втрати конфіденційних даних чи іншої чутливої інформації.

Створюване програмне рішення забезпечує механізм для розширення в подальшому, функціоналу інформаційної системи, за вимогами та запитами клієнтів. На основі наявних класів можна створювати похідні та інші сутності, варіанти взаємодії, зв’язки тощо, додавати їх у розроблюваний продукт з попереднім описом у документації, проходженням перевірки якості та етапів тестування автоматизованого та користувацького відповідно до потреб.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих на ринку рішень

Процес автоматизації різноманітних, часто виконуваних завдань, не є новацією в будь якій із сфер, проте варіанти реалізації та функціонал доволі різняться і забезпечують унікальність і гнучкість окремих продуктів. Для втілення ідеї створюваного застосунку проаналізовано наявних засобів та їх функціонал. Для створення порівняння, виокремлено наступні критерії оцінок:

- спектр функціональних можливостей
- зручність у користуванні
- адаптивність й персоналізація інтерфейсу
- сучасність та автентичність стилізації
- забезпечення безпеки даних
- кросплатформенність
- вибір мови інтерфейсу

Оскільки універсального рішення для розроблюваної ідеї не знайдено, вирішено опиратись на додатки та сайти, що надають можливості окремих її компонентів. Першим необхідним функціоналом, виступатиме функціонал запису даних для спостереження, важливість цього компоненту зумовлюється основною ціллю застосування всього продукту: відслідковування змін та аналіз здобутих результатів. Тож для аналізу необхідності та доцільності створення додатку чи сайту, варто переглянути способи втілення у наявні рішення. Пошуки можна здійснити в категоріях здоров’я, спорт, медицина тощо. Огляд найкращих рішень сформовано на базі статті із медіа ресурсу Healthline [1].

JEFIT (фітнес засосунок)

Рисунок 1.1.1 – Логотип застосунку JEFIT

За деякими рейтингами (зокрема наведеної вище статті) додаток подається як найпопулярніший, логотип компанії подано на рисунку 1.1.1. Застосунок покликаний для надання функціоналу керування та запису даних щодо тренувань в зручному форматі у телефоні.

Згідно із сформованим напередодні списком оцінювання програми, варто виокремити її переваги та недоліки. Опираючись на відгуки користувачів на платформах AppStore, Google Play Market та власний досвід користування, до переліку переваг варто віднести: можливість аналізу прогресу фізичних навантажень, формування графіків опираючись на такі дані, створення і відображення тренувальних планів і вправ. З недоліків, користувачі зазначають: втрату записаних даних, неможливість синхронізації й перенесення даних між пристроями, багато реклами та необхідність постійної оплати за більшість функціоналу. Інтерфейс мобільного застосунку подано на рисунку 1.1.2.

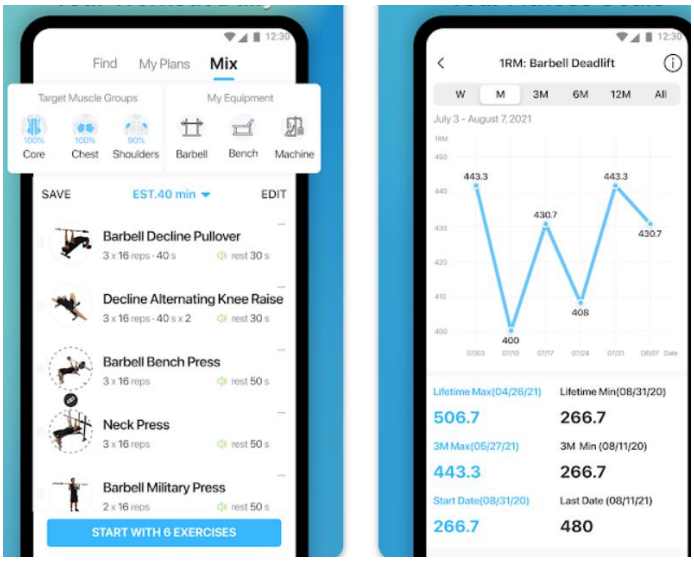


Рисунок 1.1.2 – Інтерфейс мобільного додатку JEFIT

Strava (застосунок і сайт для відстеження спортивної активності через GPS)

Рисунок 1.1.3 – Логотип компанії Strava

Застосунок має функціонал для запису даних багатьох тренувань за стилями, його плюси: безкоштовний варіант програми, аналітика і графіки на основі даних, широкі функції відстеження та вимірювання продуктивності, з мінусів: для використання більшості функцій аналізу даних потрібна місячна підписка, не вистачає силових тренувань і студійних занять, запису для інших важливих для відслідковування прогресу даних, періодична втрата записаних даних. Інтерфейс рішення наведено на рисунку 1.1.4.

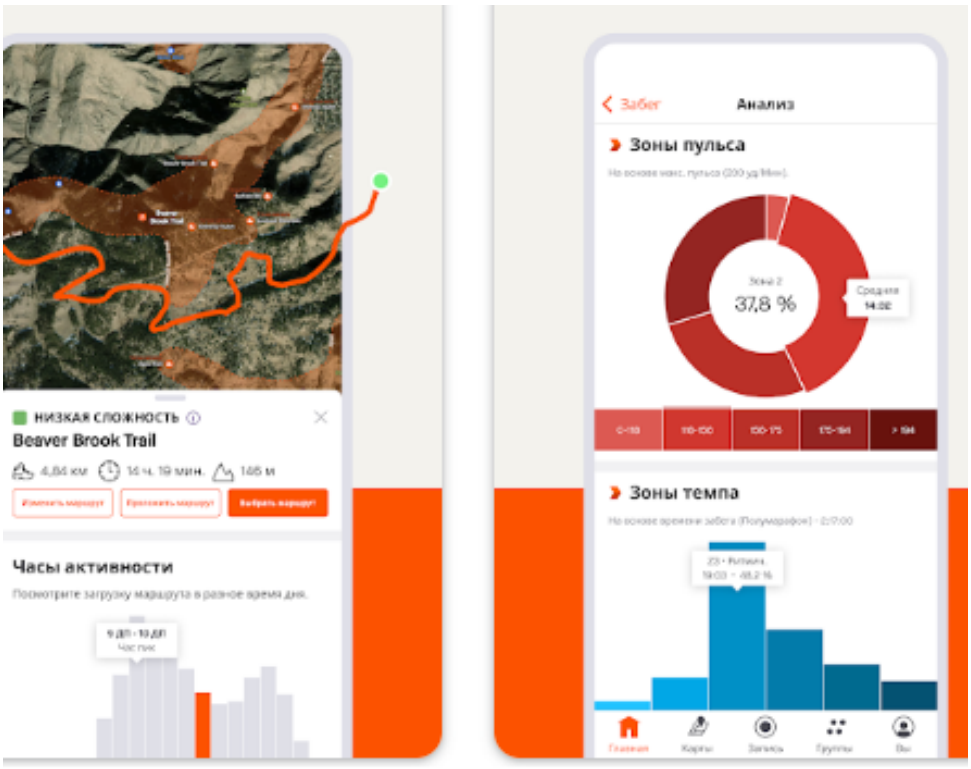


Рисунок 1.1.4 – Інтерфейс додатку Strava

Tablycjakalorijnosti (система щоденного контролю отриманої і витраченої енергії)

Рисунок 1.1.5 – Логотип компанії Tablycjakalorijnosti

Програма дуже зручна для детального запису даних щодо спожитих продуктів, їх поживної цінності і нутрієнтів. Також є можливість відстежувати зміну власних даних для відстеження особистого прогресу в зміні замірів параметрів тіла. Плюси: запис даних харчування, можливість додання страв та рецептів, велика база даних продуктів і зручне подання записаних даних. Мінуси: відсутність можливостей додання даних про тренування, обмежений функціонал конкретними варіантами застосування, періодичні проблеми із доступом до функціоналу додатку, велика кількість реклами. Інтерфейс додатку подано на рисунку 1.1.6.

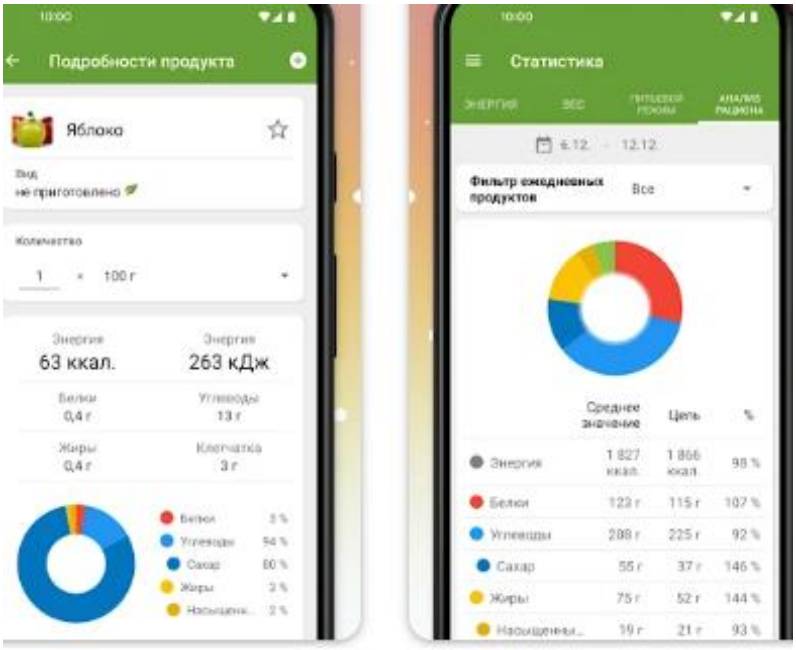


Рисунок 1.1.6 – Інтерфейс програми Tablycjakalorijnosti

Тож можна скласти таблицю із результатами оцінки проаналізованих додатків в категоріях, наприклад за шкалою 1-5, де при оцінках враховано також відгуки користувачів.

Таблиця 1.1 – Оцінка згідно із оголошеними критеріями

Продукт / Критерій	JEFIT	Strava	tablycjakalorijnosti
функціональні можливості	3	3	3
зручність у користуванні	4	3	5
адаптивність й персоналізація	1	3	3
сучасність та автентичність стилізації	3	3	3
кросплатформенність	3	5	3
безпека і зберігання даних	2	3	5

1.2 Обґрунтування вибору напрямку дослідження

У повсякденному житті багато людей слідкують за своїм фізичним здоров’ям та піклуються про його стан. Позитивний ефект фізичної активності при різних хронічних захворюваннях добре вивчений і підтверджений у літературі, також є відносно небагато досліджень, які б систематично кількісно оцінювали співвідношення доза-реакція, між фізичною активністю та результуючими станами хронічного захворювання. Зокрема, систематичні огляди, які вивчали взаємозв’язок доза-реакція, відносно раку молочної залози, діабетом, ішемічною хворобою серця та будь-яким видом раку. Для досягнення користі й ефективності ВООЗ рекомендує принаймні 600 метаболічних еквівалентів (МЕТ) хвилин

загальної активності на тиждень; наприклад, це буде приблизно 150 хвилин на тиждень швидкої ходьби або 75 хвилин на тиждень бігу, також як активність розглядається будь-який інший вид спорту та існує кореляція, між стилем та необхідною тривалістю. В загальному стаття групи вчених із Копенгагена [2] показала, що фізично активні особи, знижують ризик смертності мінімум на 30% в порівнянні з тими, хто не має спортивних навантажень в своєму розпорядку дня. Проте, верхньої межі, де б фізична активність ставала шкідливою для особи не встановлено. Вищезгадана публікація, стверджує, що для 116 000 осіб відстежували активність (повільний біг) впродовж 30 років, і це навантаження було в двічі більше від рекомендованого ВООЗ, результати дослідів виявились ще кращими. Це говорить про однозначну користь від занять певними видами спорту (не стосується травматичних дисциплін і небезпечних видів діяльності). Незважаючи на те, що добре встановлено зв'язок між фізичною активністю та хронічними захворюваннями, включаючи рак молочної залози, рак товстої кишки, діабет, ішемічну хворобу серця та ішемічний інсульт й згадану вище статтю, наявні відомості доволі обмежені щодо того, наскільки ризик зменшується зі збільшенням обсягу загальної активності, тому, при реалізації програмного рішення, важливо надати користувачам можливість записувати різні варіанти відстежуваних даних, аби можна було персоналізувати тренування та підібрати зручні функції.

Іншим важливим аспектом чому важливим є відстеження фізичної активності та її аналіз слугує те, що сучасний спосіб життя помітно відрізняється від способу життя наших предків, і ці відмінності в харчуванні та рівні активності часто є причиною глобальної «пандемії» ожиріння. Про це свідчить дослід про характеристику ожиріння пов'язаного із «західним» способом життя [3]. В досліді порівнювались дані мисливців-збирачів, сучасних людей із розвинених країн та фермерів із країн що розвиваються. Результат відношення між відсотками жирової маси і рівнем фізичної активності показано на рисунку 1.2.1.

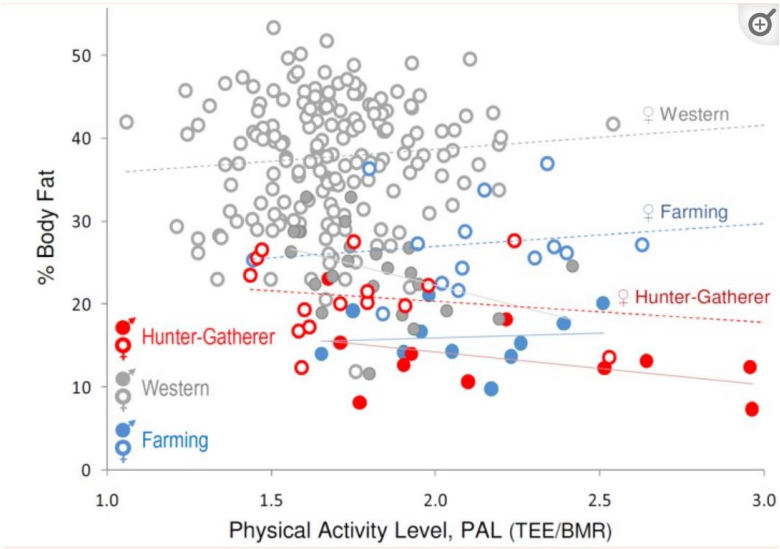


Рисунок 1.2.1 – Відношення відсотку жиру в організмі та фізичною активністю

Тож, для підтримки та вдосконалення власного тіла, необхідно підібрати зручну та ефективну програму тренувань, спосіб харчування, а також ефективні харчові добавки й препарати для збалансування й забезпечення потрібних концентрацій гормонів й інших речовин, які споживаються чи виробляються в людському організмі. Для фізичних результатів часто використовується гормонотерапія та аналоги гормонів, речовини стимуляції певних процесів в організмі. Важливими факторами для досягнення персональних цілей у спортивних результатах чи нормуванні певних медичних показників слугують різноманітні чинники, які необхідно відстежувати водночас, оскільки наявність чи відсутність одного з них впливає на ланцюг подій, що врешті ведуть до результуючих характеристик особи. Для самого ж відстеження та створення аналітики записуваних даних потрібно дослідити зміни певних антропометричних показників таких як: анатомічні окружності, маса тіла, а також таких обстежень, як динамометрія та каліперометрія. Варто звернути увагу й на вплив дієтотерапії та помірних фізичних навантажень на рівень гормонів. Зрозумілим є необхідність наукових досліджень в такій сфері, та недостатність підтверджених даних для формування єдиної стратегії чи послідовності дій для досягнення максимальних результатів, у виконанні, все ж, формовано продукт на основі останніх наукових публікацій і матеріалів, зокрема доповіді про вплив андрогенних гормонів

(тестостерон) на потреби організму та на деякі антропометричні показники [4] та роботи «вплив об'єму та інтенсивності тренувань на покращення м'язової сили та об'єму» [5].

Спираючись на результати досліджень, враховуються наступні параметри для розробки системи аналізу даних: можливість запису даних щодо споживаних калорій, нутрієнтів отримуваних організмом із продуктів; їх кількість (вагу), категорії, функції; можливість додавання даних споживаних харчових добавок та компонентів, таких як спортивні добавки, стимулятори, енергетики, ізотоніки, анаболічні комплекси тощо, а також окремих хімічних елементів, вітамінів та мінералів; запис дозувань, одиниць виміру та характеристик споживаного; їх опис, назву, категорію, функції, час додання; можливість внесення інформації щодо медичних аналізів та інших даних для спостереження та аналітики процесів, що відбуваються впродовж періоду дослідів чи їх етапів; передбачити, врахувати і впровадити варіанти для запису одиниць вимірів, додання медіа публікацій, що міститимуть лабораторні медичні результати чи іншу інформацію, яка може знадобитись користувачам; функціонал запису даних щодо виконаних фізичних навантажень, їх назви, опису, ритму серця в момент навантажень, тривалість, швидкість, дистанцію, довжину, вагу тощо для вправ, категорії й демонстрацію; можливість відстежування даних по виконаних фізичних активностей, тренувальних планів, днів, вправ, підходів і інформацію про дату; самі дані фізичних навантажень: тренувальний план його період виконання, можливість призначення плану для багатьох користувачів, тренувальні дні, кількість повторень групи вправ впродовж тижня їх стиль та категорії, окремі вправи, підхід до виконання, кількість повторів і час перерви між ними, поради та рекомендації, зауваження й застереження.

Реалізація ідеї спрямована також на уможливлення відстеження даних, як у двох попередньо згаданих наукових працях. Наприклад, в кожній з них, спостереження проводилися в кілька етапів, кожен по кілька тижнів і було відображено отримані дані у форматі таблиць, де зафіксовані результати поетапно чи в цілому, та у форматі графіків, на яких відображені зміни даних впродовж

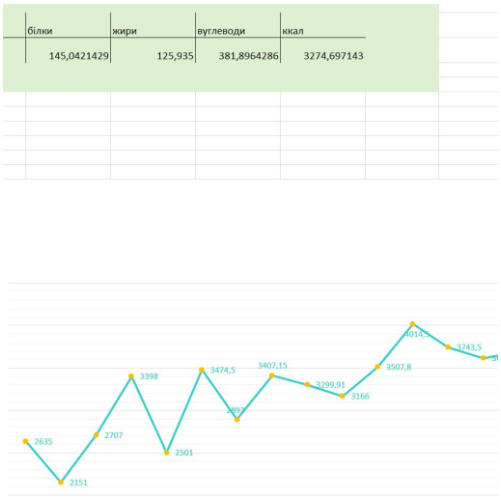
досліду. Приклад графіків, які повинні бути створеними подано на рисунках 1.2.2 та 1.2.3.

Зміни концентрації тестостерону впродовж досліду

Дата	Тестостерон в ннмоль/л
04.07.2022	19.06
29.07.2022	32.45
16.08.2022	438.9
30.08.2022	275.29
04.10.2022	15.49
01.11.2022	14.49
29.11.2022	26.66

Рисунок 1.2.2 – Таблиці дослідження для аналізів

З рисунку видно дані, які важливі користувачам при внесенні даних щодо аналізів: назва спостереження, дата внесення, значення та його одиниці виміру, оскільки вони можуть викривляти фактичні показники, та саме результат.



білки	жири	вуглеводи	ккал
145,0421429	125,935	381,8964286	3274,697143

Рисунок 1.2.3 – Графік та таблиця дослідження спожитих нутрієнтів

Тож, з наведених ілюстрацій, та аналізу наявних рішень чітко видно затребувані функції для здійснення аналітики: формування вибірки з записаних даних та представлення їх у формі таблиць, графіків із такими даними, як кількість,

назва, категорія, одиниці виміру, дата внесення. Враховуючи вищеописані аспекти, як основне завдання майбутньої системи виступає можливість забезпечити користувача функціональністю ведення передачі даних, взаємодії із вивантаженими файлами та створеними сутностями спостереження, активними компонентами та нещодавно доданими елементами, власним профілем. По суті, користувачеві системи надаватиметься практичне знаряддя для роботи із вивантаженими та доступними користувачеві файлами, аналізами, споживаними нутрієнтами, добавками і даними щодо тренувань, виконаної фізичної активності у системі, а також редагування, надійні передача і збереження, видалення персональної інформації. Рішення повинне надавати доступні знаряддя для цих операцій, автоматизацію повторюваних дій в середовищі. Інформаційна система повинна володіти комфортним, доступним для розуміння будь-якій професійній галузі користувачів інтерфейсом, який є адаптивним та може використовуватися на великій кількості різних пристроїв виведення інформації. Розроблюваний додаток повинен гарантувати збереження даних користувача, як особистих що потрібні для автентифікації так і того, що він надав програмі. Оновлення версій повинне здійснюватися в автоматизованому режимі на основі системи контролю версій і системи (сервера). Програмне рішення включатиме й можливість для розширення в подальшому, функціоналу програмного продукту на вимогу та запити клієнтів. Забезпечувати таку можливість, можна при коректному проектуванні системи, де на основі наявних класів можна створювати похідні та інші, споріднені, сутності, варіанти взаємодії, зв’язки. Додавання нового у розроблюваний продукт повинне супроводжуватись описом у документації, проходженням перевірки якості та проходженням етапів тестування автоматизованого та користувацького відповідно до нових функціоналів.

Безпекові вимоги включатимуть: доступ до ресурсу із використанням сторонніх засобів (браузерів для ПК і окремо застосунків для мобільних та планшетів) , а також використати сучасні підходи до хешування паролів і застосування затверджених і рекомендованих способів передачі даних між компонентами створюваного продукту.

Дане завдання потребує розуміння принципів розробки, алгоритмів для ефективної реалізації та оптимізації розроблюваних методів; завдання дає можливість дослідити та проаналізувати доступні рішення частини операцій та інтегрувати відповідні реалізації у власний проєкт, а також вдосконалити професійні навички у розробці та тестуванні функціоналу.

1.3 Основні алгоритми та підхід до реалізації

Алгоритми взаємодії користувача і програми будуть наступними: користувач отримує доступ до створеного акаунту – здійснює вибірку записаних даних – отримує результати аналізу, реєстрація нового акаунту – додання персональних даних – внесення даних для спостереження, оновлення даних входу – доступ до записів – вибірка потрібної інформації – формування звіту.

Для впровадження розробки вибрано методологію RAD, оскільки для процесу важлива швидкість, вимоги суттєво не змінюватимуться впродовж життєвого циклу ПЗ, для створення продукту не буде задіяно великої команди чи команди взагалі.

Опис обраної методології:

Обрана методологія спрямована на максимальну швидкість та зручність процесу створення із особливим акцентом на використовувані засоби та їх відповідність обраним концепціям. Із її використанням, розробники мають змогу ефективно розподіляти час на модулі виконання етапів реалізації продукту та дбати про їх відповідність замовленим.

До переваг можна віднести:

- швидкість створення продукту;
- заощадження вартості процесу;
- збереження якості.

До забезпечення високої якості створюваного продукту, відноситься повне виконання вимог, що включають функціональні та нефункціональні, забезпечення

їх адаптивності та гнучкості в процесі розробки до модифікацій, а також отримання якісної документації для зручності та підтримки системи в майбутньому.

Фази циклів розробки систем із використанням методології RAD можна розподілити наступним чином (рис. 1.2.4): планування (опис аналізу й вимог); проектування; створення та реалізація; інтеграція та використання.

Основні принципи методології:

- розробка додатків ітераціями;
- можливість не завершити повний цикл розробки на окремому етапі;
- інтеграція користувачів до процесу розробки і тестування;
- використання засобів управління конфігурацією;
- тестування та розвиток проекту, паралельно із розробкою продукту.

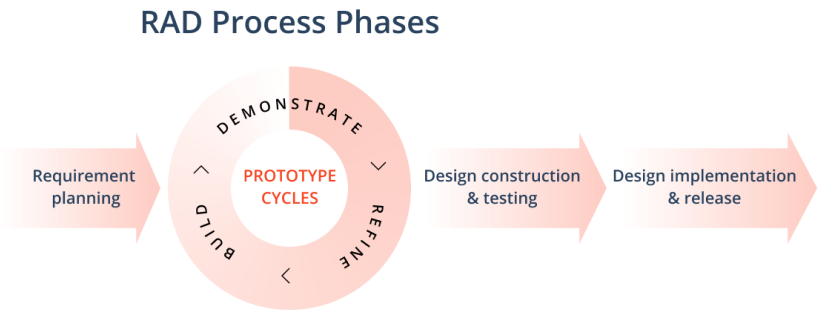


Рисунок 1.2.4 – Схема етапів створення системи із RAD-методологією

Важливість принципів менеджменту в описуваній технології зумовлена їх поширенням на кожен із етапів життєвого циклу створюваних систем: етапу дослідження організації, формування вимог щодо дизайну тощо. Втілення принципів видно в наступних перевагах застосування технології: швидкість виконання робіт, забезпечення якісних характеристик, зниження вартості розробки.

1.4 Опис вимог до проекту

Система повинна задовольняти наступні функціональні вимоги:

- реєстрація у системі
- вхід в систему
- вивантаження даних
- додання користувачів
- додання ролей користувачів
- оновлення токену сесії із браузера
- редагування введених даних
- перегляд окремих даних
- перегляд списків сутностей
- видалення файлів
- видалення профілю
- додання опису для файлів
- перегляд статистики за певними категоріями
- перегляд даних профілю
- налаштування системи перевірки особи користувача

В процесі розробки програмного забезпечення описаний перелік функціоналу та інших можливостей може бути доповнений та розширений.
До нефункціональних вимог відноситься надійність:

- збереження даних. Пропонована система повинна гарантувати збереження даних користувача, як особистих що потрібні для автентифікації так і того, що було представлено програмі, для подальшого опрацювання;
- зручність використання. Інформаційна система повинна володіти структурно правильним, зрозумілим для користування будь-якому сегменту осіб в незалежності від роду зайнятості інтерфейсом, який є адаптивним та може

використовуватися на великій кількості різних пристроїв виведення інформації;

- можливість надання відгуку щодо роботи програми, рекомендацій та побажань для подальшого розвитку системи на вимоги користувачів та враховуючи їх у подальшій модифікації;
- норма дефектів. система повинна бути доступна у цілодобовому режимі та до використання, якщо користувач увійшов (пройшов автентифікацію та авторизацію), але довго не відкривав головну форму програми реалізовано сесію для запиту паролю та логіну користувача для аутентифікації в систему. Час, що йде на обслуговування створеного додатку не повинен перевищувати 3% від часу тривалості роботи загалом.

Продуктивність:

- паралельна робота користувачів. Застосунок має підтримувати велику кількість зв'язків із працюючими користувачами, надаючи їм доступ до спільної бази даних та інформації;
- час відгуку. Очікування на результат запиту, для постійно відтворюваних завдань – до 0,5 секунд, для складніших запитів – приблизно до 1,5 секунд.

Придатність до експлуатації:

- масштабованість. Система має бути здатна підтримувати велику кількість одночасно виконуваних запитів до загальної використовуваної бази даних і забезпечувати можливість збільшити їх кількість;
- оновлення версій. Оновлення версій повинне здійснюватися в автоматизованому режимі на основі системи контролю версій і системи (сервера).

Обмеження проектування:

Вимоги до структури:

В програмному забезпеченні доцільно втілити деякі структурні шаблони згідно із архітектурними принципами взаємодії із компонентами системи:

- для розроблюваної інформаційної системи потрібно реалізувати модульність окремих структурних компонентів (тих, що відповідають за певну логіку взаємодії між класами програми);
- винесення в окремі файли структурних елементів, які використовуються в межах усієї програми та не залежать від конкретного класу;
- зберігання паролів користувачів у безпечному форматі в базі даних (в хешованому вигляді).

Інформація має зберігається в таблицях під керуванням реляційної СУБД. Так як БД є основним джерелом інформації для створюваного застосунку (інформація вноситься, модифікується і видаляється користувачем) то інформація повинна відповідати загальним вимогам:

- повнота інформації характеризує її обсяг, який повинен бути достатнім для прийняття рішення, для діяльності, для вирішення конкретного завдання, а також необхідно звернути увагу на такий критерій як надмірність, тобто необхідно формувати такий інформаційний продукт, який буде відповідати вимогам достатності для прийняття рішення;
- своєчасність: інформація повинна бути оперативною, тобто така, щоб за час її передачі та обробки стан справ не змінився;
- відповідність фактичних збережених даних до описаних, визначає ступінь достовірності інформації;
- релевантність (відповідність форми запиту до пошукового образу форми запиту);
- зручність відображення: забезпечення відповідності до потреб користувачів;
- відповідність використовуваної БД до третьої нормальної форми.

2 РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проектування

Проектування програмного продукту – це процес створення детального плану реалізації, який буде використано в ході подальшої розробки. Основною метою цього етапу є забезпечення можливості, аби описувана інформаційна система відповідала раніше сформованим вимогам та виконувала свої функції відповідно до заданих параметрів. Тож, на основі описаних у попередньому розділі пунктів, формується вектор потенційних для використання технологій і шляхів до реалізації ідеї.

2.1.1 Вибір мови та середовища розробки

Для реалізації серверної частини розроблюваної системи використано об’єктно-орієнтовану мову програмування Java та Spring Framework. Цей каркас складається з кількох компонентів, які надають великий інструментарій(Графічне представлення подано на рисунку 2.1.1):

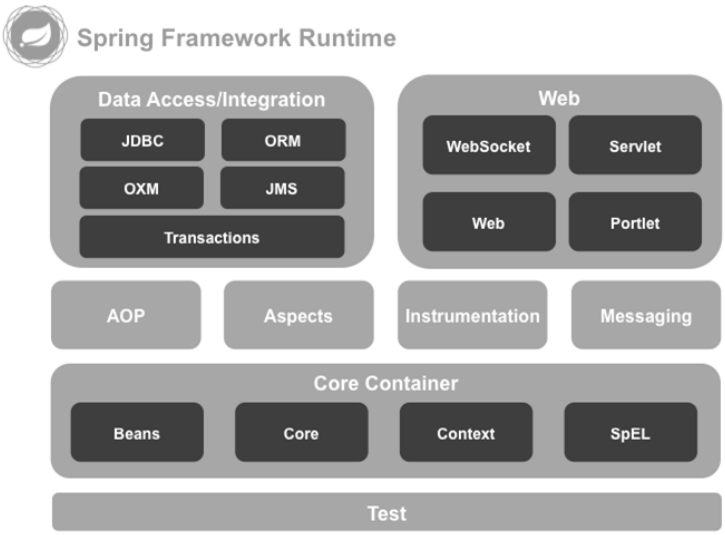


Рисунок 2.1.1 – Модулі використовуваного фреймворку

Також для реалізації серверної частини (бекенду) програми, окрім мови програмування та фреймворку потрібно використати базу даних для безпосереднього збереження даних внесених користувачами. Спираючись на поширеність та надійність наявних систем для управління реляційними базами даних, було вибрано PostgreSQL.

Рисунок 2.1.2 – Логотип СКБД PostgreSQL

Зв’язування бази даних та коду програми відбувається із вказуванням фреймворку потрібного драйвера, та включенням залежностей для роботи із нею. Для ефективного збереження і можливості оновлення таблиць із бази даних доцільно використовувати систему контейнеризації, наприклад, Docker.

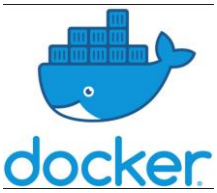


Рисунок 2.1.3 – Логотип інструментарію Docker

Docker виконує операції з управління ізольованими Linux-контейнерами і дозволяє відокремити програми потрібні для запуску від розроблюваного застосунку, а також використовує дещо більш високо рівневі способи взаємодії між компонентами, необхідними для розгортання програм, що надає можливості керувати окремими процесами ізольовано від решти. Наприклад, Docker дозволяє не враховувати вміст контейнера для запуску окремих процесів в ізольованому режимі і, зрештою, транспортувати і дублювати сформовані для цих процесів контейнери на сторонні сервери. Таким чином розробник делегує обов’язки зі

створення, обслуговування і підтримки контейнерів Docker-у. Скориставшись інструментарієм створення відображень та контейнерів, тестування та швидкого розгортання коду, можна значно зменшити затримку між написанням коду та його запуском.

Використовувані можливості докера:

- створення контейнерів, із складними програмними стеками, завдяки зв'язуванню існуючих контейнерів між собою. Створений мережевий тунель забезпечує взаємодію між компонентними контейнерами, що дає можливість уникнути затримок, які були б у разі об’єднання вмісту останніх.
- можливість збереження даних (контейнерів та образів) та передача їх разом із змінами в стані на інші пристрої використовуючи персональний обліковий запис у Docker.
- завантаження із docker-hub і розгортання необхідних компонентів системи (бази даних PostgreSQL) не встановлюючи таку на пристрій, з якого здійснюється запуск та розробка;
- можливість запуску кількох контейнерів та образів одночасно;
- робота на різних операційних системах та незалежно від фізичних характеристик пристроїв, на яких здійснюється розгортання інструментарію.

Інтерфейс інструментарію подано на рисунку 2.1.4.

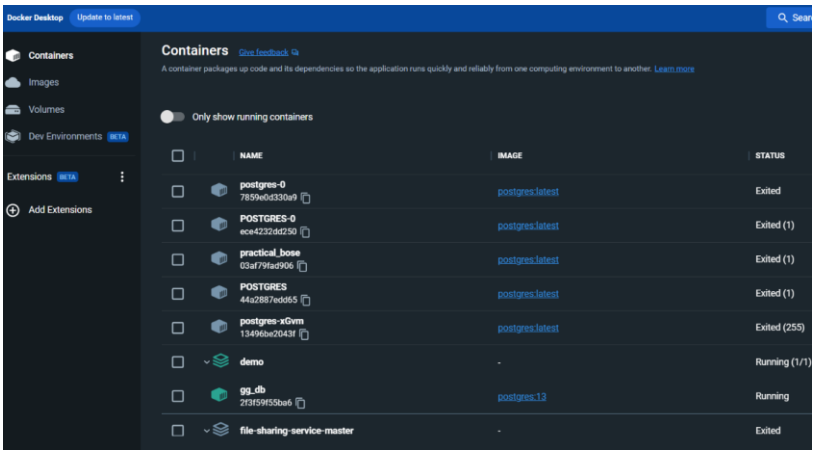


Рисунок 2.1.4 – Інтерфейс програми Docker

2.1.2 Розробка моделей предметної області

Ключовими для розуміння композиції застосунку є моделі предметної області, які виступають абстрактними моделями, що використовуються в процесі розробки інформаційної системи для представлення основних концепцій, понять та взаємозв'язків, які наявні в предметній області. Їх використання доцільне для фіксації, уточнення й визначення функціональних вимог до продукту.

Моделі предметної області можуть бути розроблені на різних рівнях деталізації та включати різні аспекти додатку. Наприклад, вони можуть містити інформацію про об'єкти їх атрибути, процеси та послідовність, взаємодії між об'єктами та ін.

Результатом розробки моделей предметної області може бути створення діаграм використання, послідовностей, прецедентів та інших видів діаграм, які бути використані для подальшого проектування програмного продукту та його реалізації і документації. Створені моделі предметної області можуть змінюватися впродовж всього процесу розробки додатку, залежно від зміни вимог до продукту та виявлення нових.

Згідно з визначенням, для кращого візуального сприйняття зв’язків між акторами та варіантами використанням можна створити діаграму варіантів використання. Актори та варіанти використання, які зображені у вигляді діаграми, можна побачити той функціонал, який необхідно реалізувати в системі.

Для проектованого застосунку основним варіантом використанням є перегляд та додавання даних спостереження. Зображення варіантів використання подано на рисунку 2.1.5

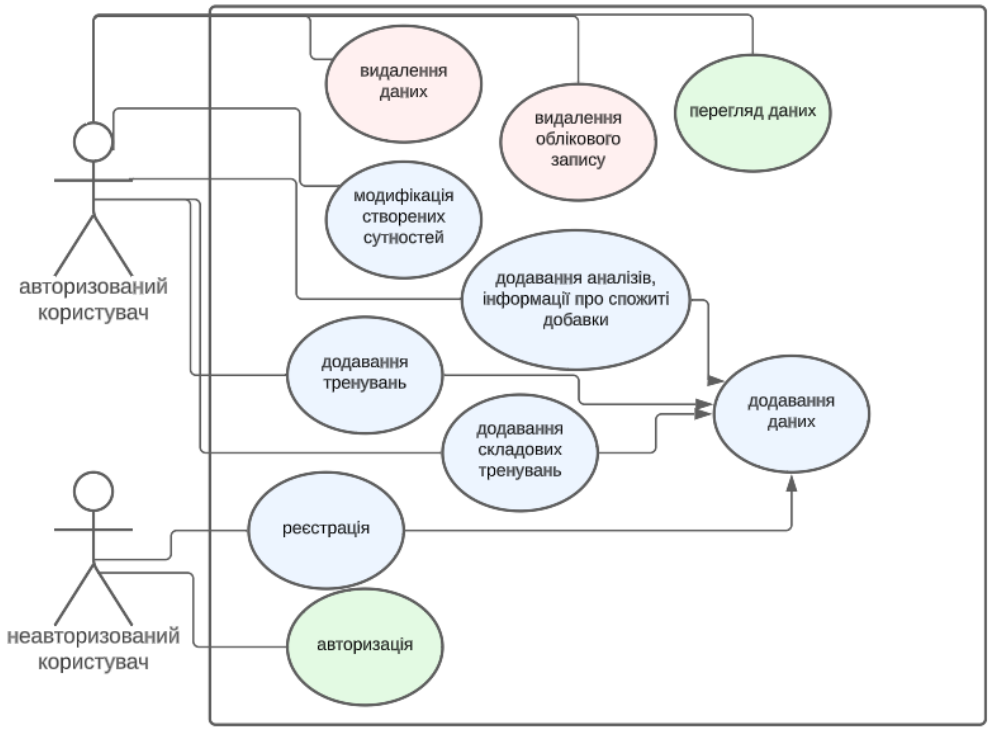


Рисунок 2.1.5 – Діаграма використання застосунку

На діаграмі з рисунку видно, що основними акторами виступають авторизований та неавторизований користувачі. Для кожного з них відведено різний стек можливостей. Так перед неавторизованим постає можливість обрати опцію входу чи реєстрації, оскільки в системі не передбачено дій, які можуть бути представлені таким акторам. Після входу, або ж реєстрації, користувачеві надається інструментарій взаємодії із різноманітними сутностями, додавання, видалення чи модифікація створеного й перегляд раніше доданого із відповідними представленнями, сортованими за датою чи вибраними категоріями атрибутів компонентів застосунку.

Наступним етапом відображення роботи системи, проектування коректного представлення та ланцюгу передавання даних в межах цілого програмного рішення слугує діаграма діяльності, які змальовують в часовому просторі взаємодію між компонентами у застосунку, також у діаграмі зображають обмін повідомленнями між класами, вказується їх порядок та напрям передачі, та їх послідовність.

Вертикальні лінії вказують на тривалість процесів в програмі та об’єктів , які існують паралельно та в один час. Горизонтальні лінії описують процес надсилання запиту (повідомлення) між описаними компонентами. Для процесу авторизації користувача існують компоненти користувач, контролер, сервіс безпеки, сервіс опрацювання і бази даних, транзакція запитів відбувається із опрацюванням даних в кожному із описаних класів у окремо виділеному часовому зрізі. Приклад подано на рисунку 2.1.6

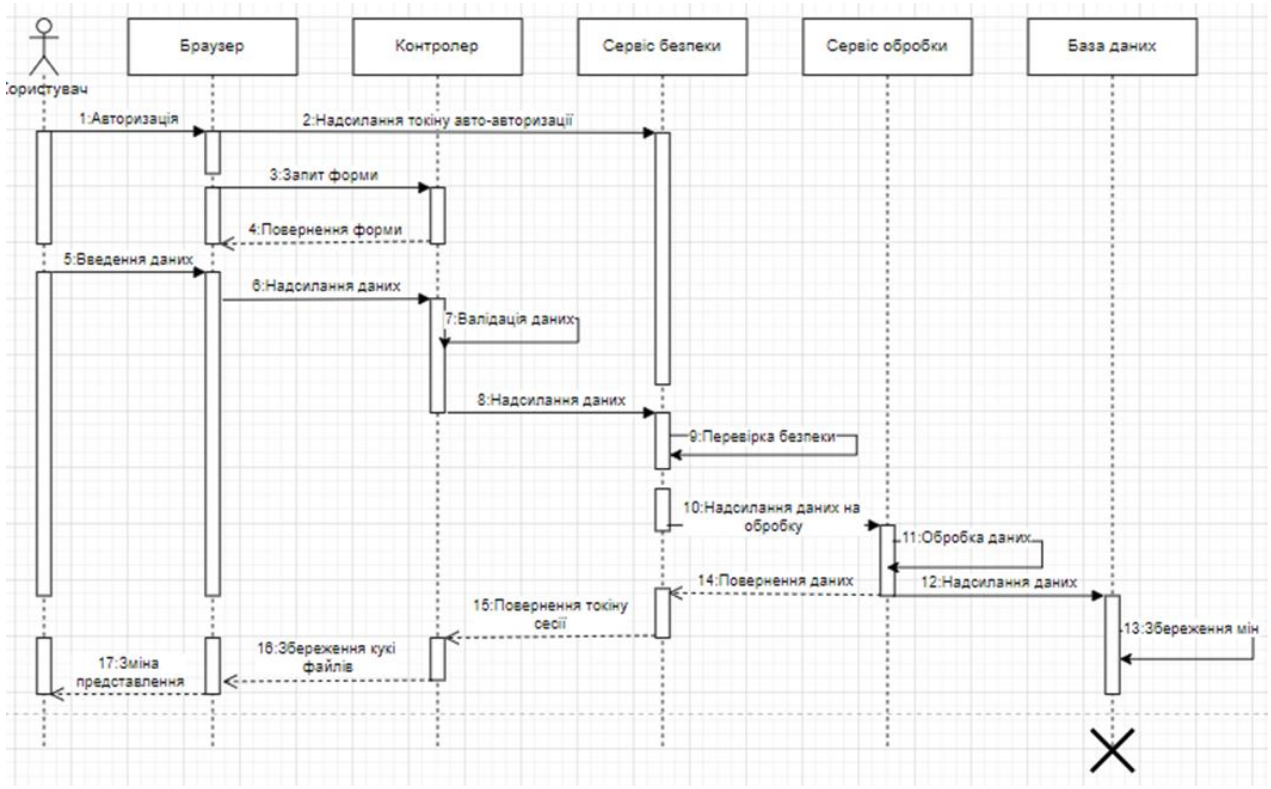


Рисунок 2.1.6 – Діаграма послідовності у програмі

З діаграми авторизації видно, що коли користувач заходить на сервіс, його дані, якщо вони наявні, надсилаються на сервіс безпеки авторизації, де вони перевіряються, або ж середовище запитує та відображає форму авторизації, надіслану контролером, після ведення користувачем даних, здійснюється їх опрацювання та звірення із базою даних. В результаті актор процесу отримує доступ до своїх даних та можливість здійснювати подальші операції.

2.1.3 Проектування архітектури

Маючи всі відомості про систему з першого розділу і попередніх пунктів, можна приступити до проектування архітектури проекту, а саме: визначення, якими класами буде володіти система. Визначивши всі класи застосунку, побудуємо діаграму класів за допомогою мови UML.

В об’єктно-орієнтованих, високорівневих, мовах програмування клас слугує основною структурою створюваного додатку. Це поняття присутнє і в описах системи за допомогою UML-діаграм, що дозволяє в деяких середовищах генерувати програмний код або виконувати реінжиніринг.

Тож, можна виявити наступні класи предметної галузі:

Сутність User – призначена для збереження інформації про користувачів. Це одна з основних моделей, що містить у собі основні дані для авторизації, інформація для формування звітності, отримання даних про тренування, дані щодо вживаних добавок та категорій. Дані щодо паролю визначеного користувачем збережені у хешованому вигляді, для унеможливлення взлому чи перехоплення цієї надважливої персональної інформації, що може збігатись із паролями особи в інших застосунках і, можливо, піддати небезпеці надважливі дані, що можуть бути використані зловмисниками у різних цілях проти волі нашого клієнта.

Сутність UserAuthority – призначена для збереження інформації про права, які надаються певному користувачеві, це впливатиме на подальші дії та відображення компонентів системи, дій, які може здійснювати цей користувач, права та дані щодо користувача який їх отримав, для цілісності та правильності асоціювання із даними в таблицях баз даних розроблювальної інформаційної системи.

Сутність Category – призначена для збереження інформації про категорії тренувань та вживаних продуктів і добавок, так як кожен користувач має свої певні категорії та їх опис, в цьому класі також містяться дані щодо опису категорії,

користувача, до якого ця категорія відноситься та дані, які зберігались у визначеній категорії.

Сутність TrainingPlan – призначена для представлення структури тренувальних планів, які користувачі використовують в певний проміжок часу а містить інформацію про тренувальні дні, дати початку і завершення, списки користувачів, які мають доступ та опис.

Клас TrainDay – спрямований на забезпечення цілісності передачі даних щодо тренувальних днів у базу даних. Зберігає атрибути із переліком вправ, їх описом, стилем тренувань, посилання на користувача, що додав вправу (для уможливлення додавання тренувальних днів, незалежних від тренувальних планів.

Клас TrainDayExercise – містить інформацію про тренувальний день, для якого додається вправа, опис вправи, опис стилю виконання, підходів, їх кількості та дані про час перерви між ними.

Сутність ActivityObservationData – призначена для представлення структури відстежуваних змін в прогресі певного виду діяльності (вправ), вбачає можливість запису вибраних критеріїв відслідковування, наприклад кількості, ваги, швидкості, часу виконання, дистанції та ін.

Клас AnalysisData – містить дані про аналіз, його одиниці виміру, час вивантаження у застосунок, інформацію про користувача, та можливість додання фотознімку лабораторного результату.

Клас FoodIntake – описує дані щодо спожитих продуктів: їх назва, час споживання, поживну харчову цінність, вміст нутрієнтів, категорію споживання, можливість додання функції (опису бажаних цілей, що очікуються для досягнення із вживанням продукту).

Клас SupplementIntake – спрямований для опису інформації щодо спожитих додаткових компонентів, що не є продуктами харчування, проте потрібні для досягнення поставлених результатів, наприклад, поширених спортивних добавок, мінералів, вітамінів, медикаментів тощо. Містить опис та назву речовини, кількість(дозування), одиниці виміру, час вживання, можливе додання фотознімків для кращого розуміння препарату в майбутньому відстеженні.

Оскільки для зберігання всіх описаних атрибутів класу при реалізації об’єкту виділяється пам’ять, варто організувати класи так, аби не було надлишкових даних, при тому не упустити потрібних атрибутів для зручності у роботі із додатком. При тому у деяких об’єктів буде зв’язок із екземплярами класів системи, вони можуть бути організовані як атрибути чи винесені в окрему таблицю в базі даних із внесенням лише посилання на ідентифікатор в полі класу. Виходячи з визначених сутностей системи, розпочинаємо визначення класів системи:

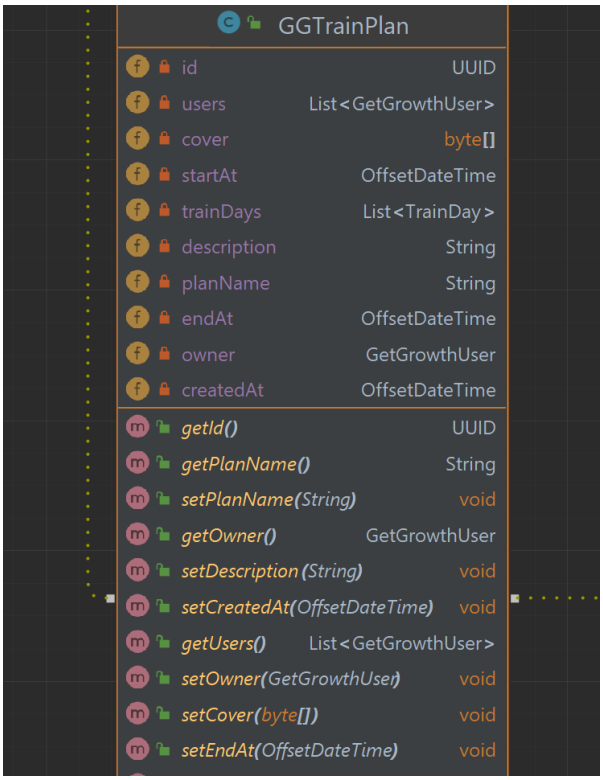


Рисунок 2.1.7 – Клас опису тренувального плану

Користувач може добавляти тренувальні плани, редагувати, або видалити їх.

Клас TrainPlan (рис. 2.1.7) відповідає за створені в інформаційній системі екземпляри цього типу. Він містить у собі наступні деталі: ідентифікаційний номер (унікальний ключ доступу до екземпляру в базі даних), список користувачів, для яких відкритий доступ для читання плану, обкладинку, що відображається для зручності представлення окремих планів, дати початку і кінця, «власника» цього екземпляру, тренувальні дні, опис, час створення.

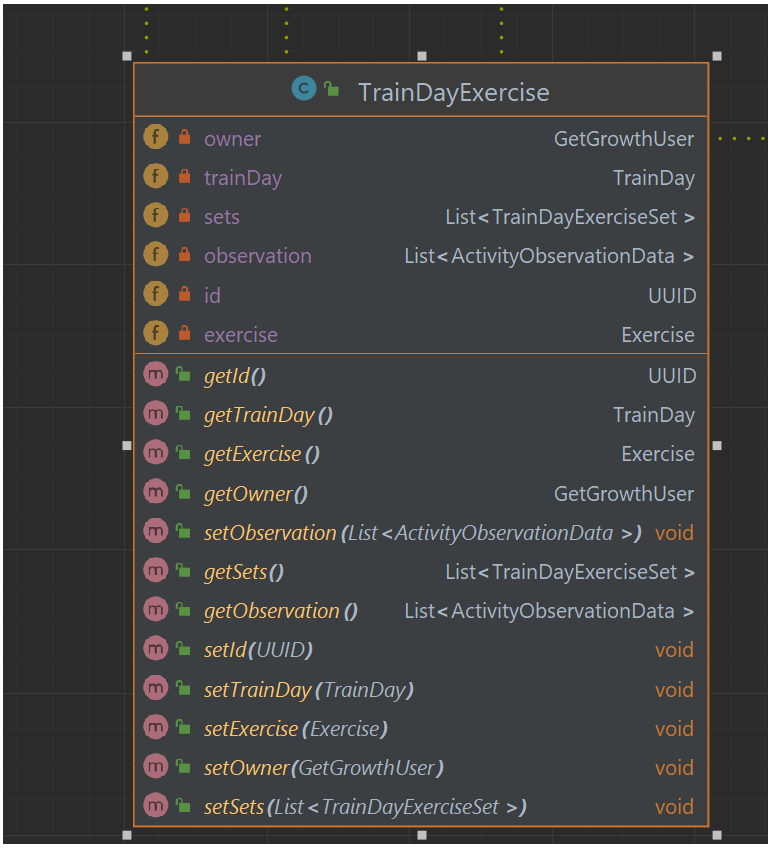


Рисунок 2.1.8 – Представлення класу тренувальних вправ

Клас, представлений на рисунку 2.1.8, містить дані, які описують сутність тренувальної вправи, яка відноситься до певного тренувального дня, містить посилання на детальний опис вправи, користувача, який додав цей екземпляр, додані дані спостереження для цього класу, дані про підходи для даної вправи та методи для доступу і редагування цих атрибутів.

Методи get спрямовані на повернення даних із створеного унікального екземпляру сутності, а set – для призначення чи перезапису окремих полів, не модифікуючи в той ж час інші наявні атрибути.

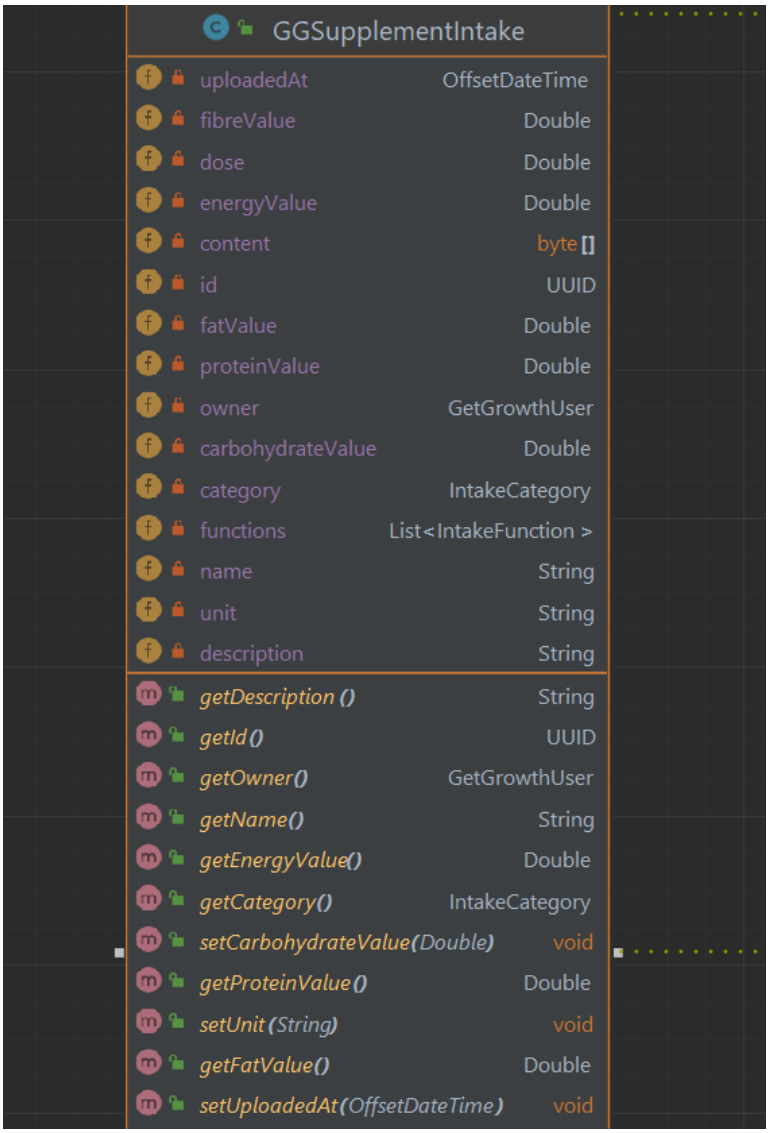


Рисунок 2.1.9 – Опис класу спожитих добавок

Клас, представлений на рисунку 2.1.9, містить дані, які описують сутність спожитих добавок чи інших препаратів, інформацію із назвою, опису, про дозування, одиниці виміру, дані про поживні цінності, час додання, функції, до яких належать: для сну, для збалансування гормонального фону, для набору маси, для підвищення рівня енергії, для зміцнення імунітету тощо, а до списку категорій відносяться: медикаменти, вітаміни, мінерали, харчі, добавки та ін.

Схожий вигляд мають класи, які описують такі сутності як спожиті продукти харчування та доданих медичних аналізів, відмінними для цих класів є зв’язки з іншими класами та деякі атрибути-характеристики.

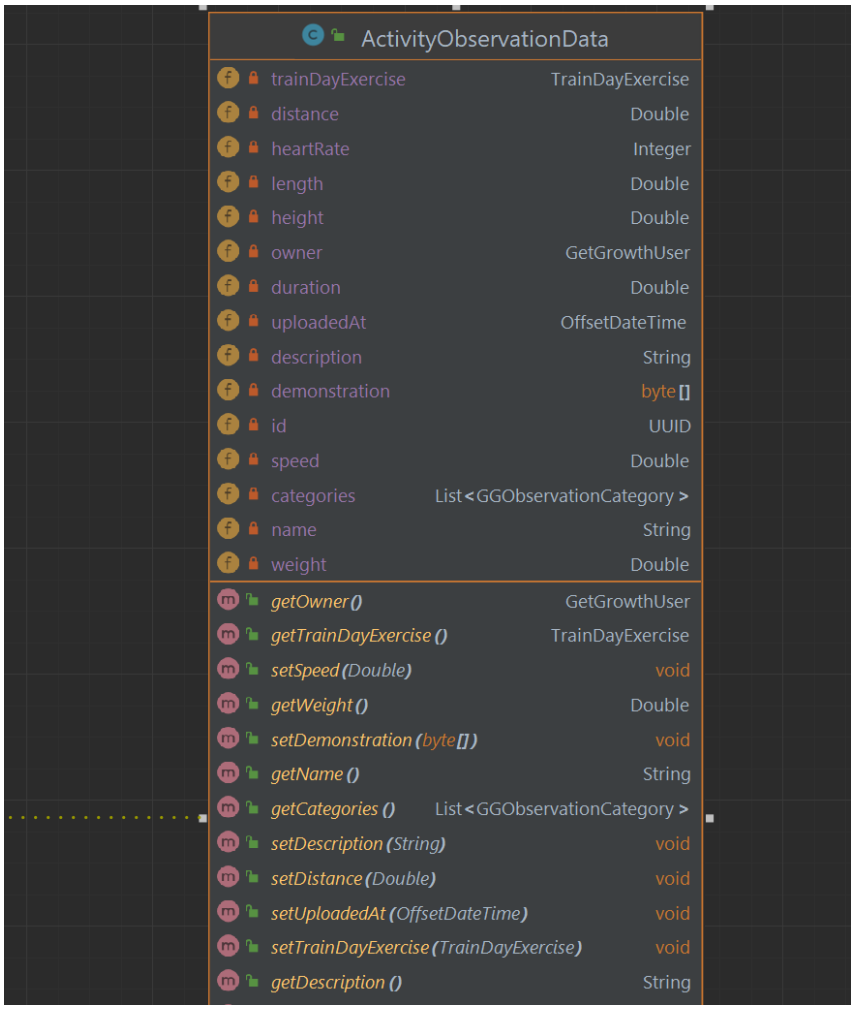


Рисунок 2.1.10 – Структура класу даних спостереження

Загальний функціонал класів, які спрямовані на представлення даних спостереження містить назву і опис, категорію спостереження, час додання публікації і користувача, який додає цей екземпляр.

Сутність спостереження даних фізичних активностей містить також специфіковані поля, для запису потенційно використовуваних характеристик: швидкість, вага, тривалість утримання навантаження, ритм серця, дистанцію та ін., посилання на тренувальну вправу, із якою пов’язаний даний екземпляр даних спостереження.

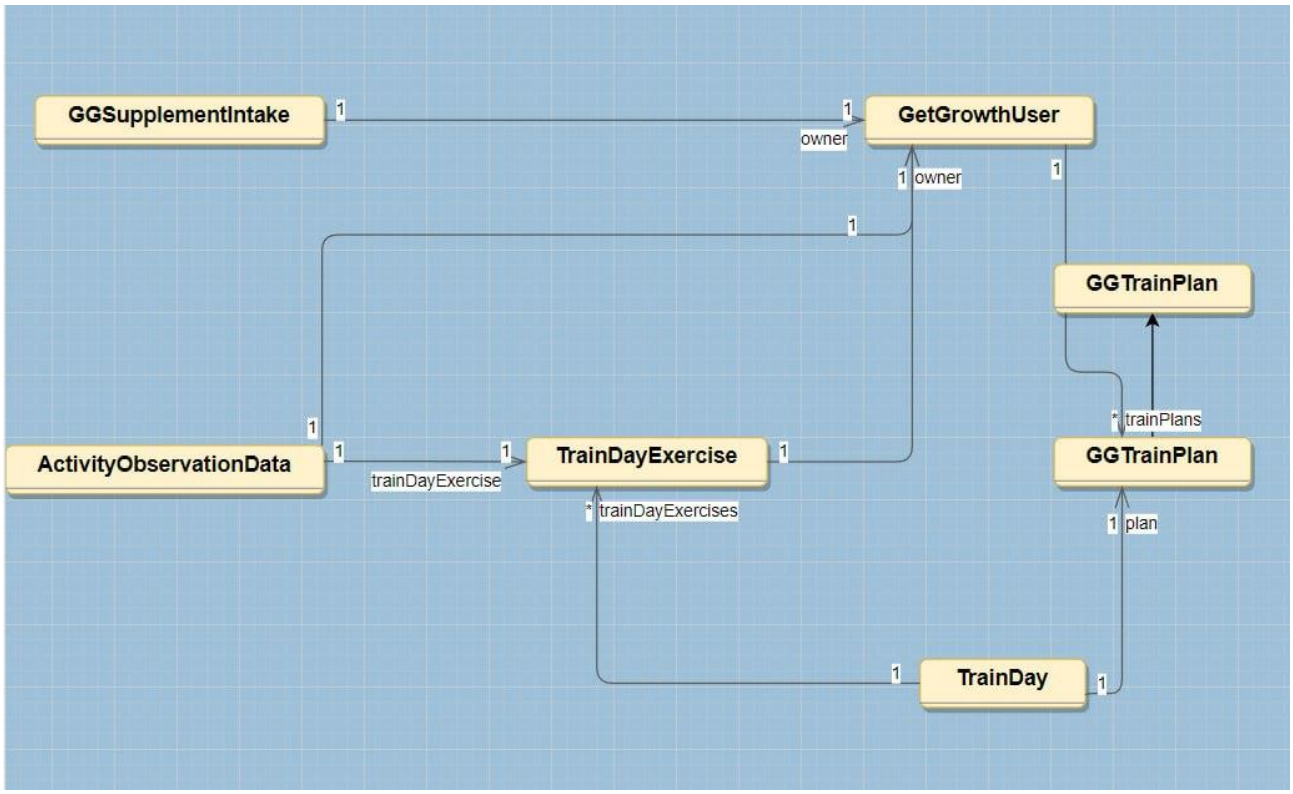


Рисунок 2.1.11 – Діаграма основних класів предметної області

На рисунку 2.1.11 представлено основні класи із проектованого застосунку, між ними видно чіткі взаємозв’язки, що передбачують правильні залежності, дозволяють отримувати доступ у зручному режимі до компонентів системи при цьому зберігаючи модульність класів.

З діаграми видно, що клас user відповідає за всіх користувачів в системі. Він містить у собі наступні деталі про користувачів: їх ідентифікатор, ім’я користувача, адресу пошти, в даному випадку це поле також слугуватиме як логін. Самі користувачі можуть редагувати профіль у випадку якщо їхня інформація щодо себе змінилася та потім отримувати змінені дані у відповіді на користувацькі запити. Повна діаграма класів подана в додатку А.

Класи: SupplementIntake, FoodIntake, ActivityObservationData, TrainPlan, TrainDay, TrainDayExercise, CompletedTrainings та ін., підпорядковуються класу користувача, що забезпечує можливість керування такими сутностями та фільтрації їх відповідно до того, хто створював ці екземпляри, кому надано доступ для читання тощо. Також у системі спроектовано відповідні класи для обміну даними,

їх верифікації, опрацювання та збереження в базі даних через відповідні інтерфейси репозиторіїв.

Для зручності та уникнення багатьох помилок у записі типів даних, також для забезпечення можливості ефективної та простої заміни методів, що опрацьовують отримувані від користувачів запити, використовують функціональні класи-записи (record) що дозволяють створити екземпляр із типами даних, не визначаючи методи, конструктори та інші, необхідні для звичайних класів нотації. Таким чином, для створення нового запису в базі даних у контролери системи повинен передаватись запит, наприклад, як на рисунку 2.1.12.

```
6 usages  sorokivski
public record CreateGGObservationRequest(
        @NotNull
        Long ownerId,
        1 usage
        @NullableNotBlank
        String name,
        1 usage
        @NotNull
        @PositiveOrZero
        Double value,
        1 usage
        @NullableNotBlank
        String unit,

        1 usage
        List<GGObservationCategory> categories,
        1 usage
        @NullableNotBlank
        String description
) {
}
```

Рисунок 2.1.12 – Клас запису даних спостереження

2.2 Конструювання

2.2.1 Реалізація ключових класів

Визначивши всі класи застосунку та описавши їх за допомогою діаграм, чітко видно необхідні функції та методи, які повинні бути реалізовані для створюваного застосунку. Для здійснення операцій для опрацювання даних із класів, використовується crud-підхід, що передбачає включення основних операцій над даними, а також restful набору принципів до реалізації виконуваних запитів. Модульність системи включає в себе наступні компоненти: контролери, сервіси, моделі(сутності), репозиторії, конфігураційні класи, ресурси, клас виключень, можливих у системі. Створення окремих модулів представлено на рисунку 2.2.1.

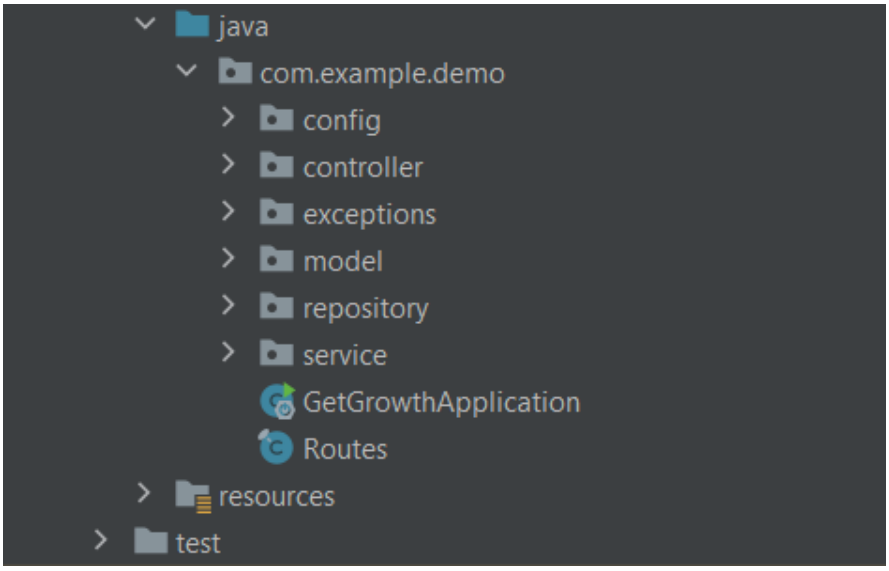


Рисунок 2.2.1 – Директорії системи

В описаних директоріях зберігаються класи, які відповідають певним вимогам та виконують окремі функції над отримуваними даними. Класи контролери відповідають за спрямування запитів користувача в потрібний сервіс та представляють структуру вихідних даних. Сервіси – повинні виконувати операції із перевірки отриманого, формування відповіді та виклику методів із репозиторіїв. Репозиторії – інтерфейси, які виконують запити до бази даних, це можуть бути як

персоналізовані запити (описані безпосередньо мовою SQL) або ж автоматично згенеровані (операції пошуку за ідентифікатором чи якимось із полів, створення та видалення). Пакет конфігурацій містить дані, потрібні для налаштування безпечного доступу в системі.

В пакеті моделей містяться визначення сутностей, наприклад TrainDayExercise може бути описана наступним чином

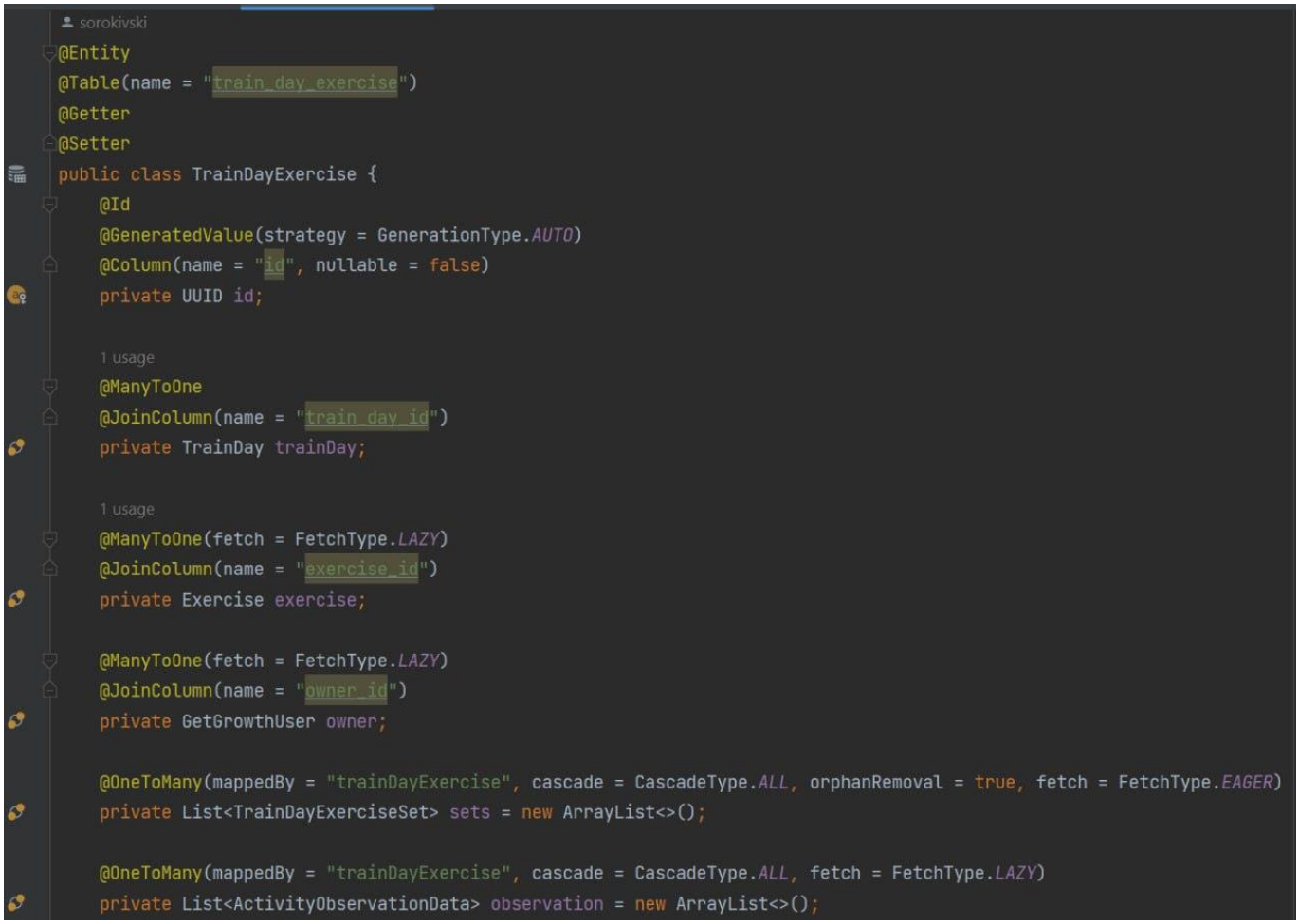


Рисунок 2.2.1 – Опис класу TrainDayExercise

У файлі класу використовуються анотації `@Entity` та `@Table (name = "train_day_exercise"`, які вказують на те, що клас описує сутність з якої створюється таблиця в базі даних, `@ManyToOne, @OneToMany` - вказують на тип зв'язків із іншими сутностями (таблицями) `@JoinColumn (name = "")` – ідентифікує поле із таблиці, за яким відбувається прив'язка до описуваної моделі. `@Getter @Setter` – анотації із пакету pojectlombok, який автоматично створює методи для доступу і модифікації усіх полів класу відповідно до їх типу.

В цій ж директорії містяться раніше описані класи-записи створення екземплярів усіх моделей, а також класи-записи із нотаціями даних із сутності які повертаються користувачеві. Приклад такого клас представлено на рисунку 2.2.2.

Рисунок 2.2.2 – Опис класу-обгортки для відповіді

Ці класи потрібні, аби можна було виключити чи уточнити деякі із атрибутів, які повертаються в результаті опрацювання класу, так анотація `@JsonInclude(JsonInclude.Include.NON_NULL)` вказує, що повертаються тільки визначені й існуючі поля та значення не по замовчуванню, а `@JsonFormat(shape = JsonFormat.Shape.STRING)` визначає новий тип для даних, наприклад дата й час

форматуються в стрічку. Також можна виключити чутливі дані із таких полів, як пароль.

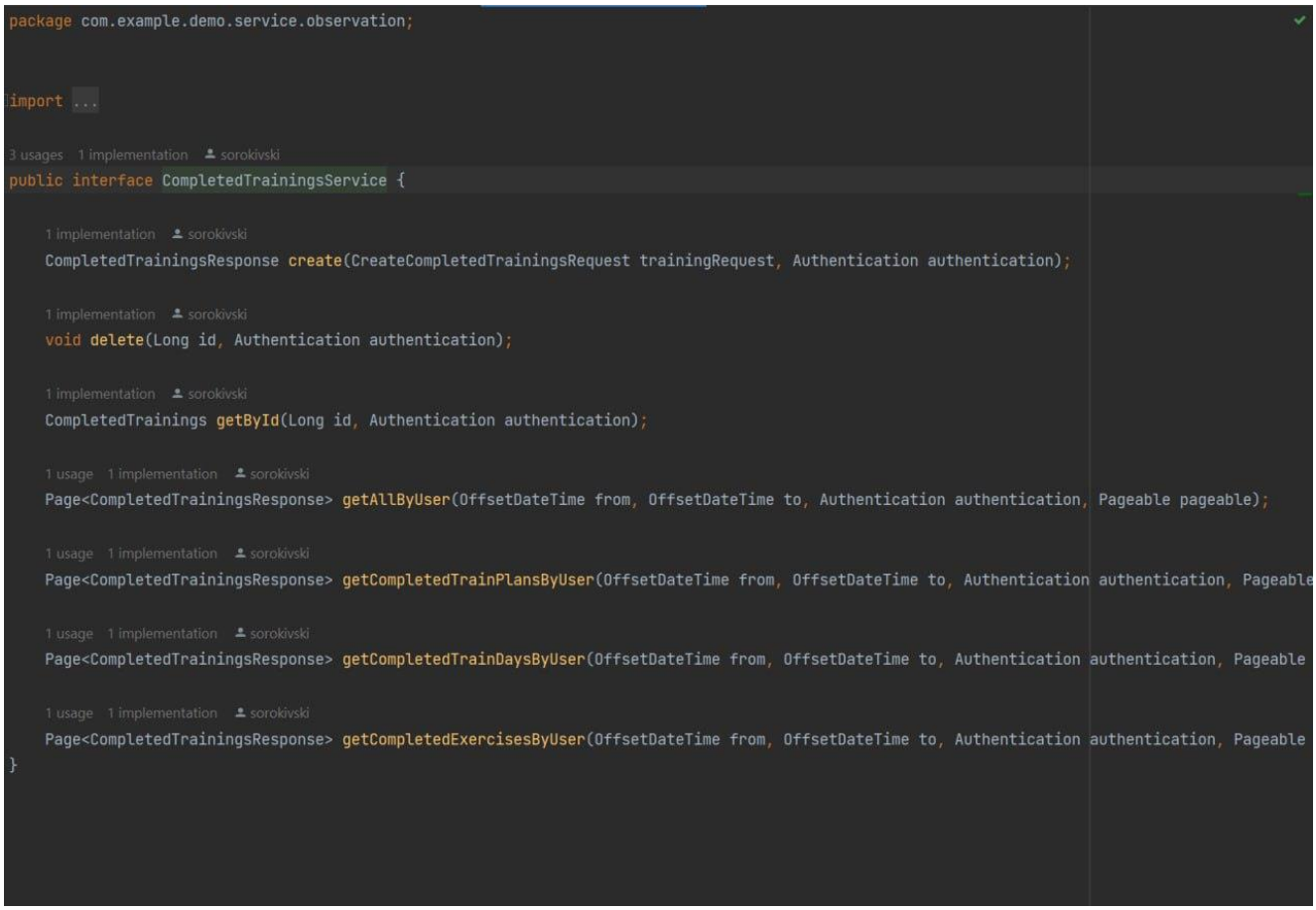
Директорія repository містить інтерфейси, через які здійснюються запити до баз даних. Приклад наведено на рисунку 2.2.3.

```
10 usages  sorokivski *
public interface TrainDayRepository extends JpaRepository<TrainDay, UUID> {

    1 usage  sorokivski
    @Query("select td from TrainDay td join td.repeatOn ro where td.plan = :plan and ro.id = :weekDay")
    Page<TrainDay> findByPlanAndWeekDay(GGTrainPlan plan, WeekDays weekDay, Pageable pageable);


    1 usage  sorokivski
    @Query("SELECT e FROM TrainDay e LEFT JOIN FETCH e.trainDayExercises WHERE e.id = :dayId")
    Optional<TrainDay> findByIdWithTrainDayExercises(@Param("dayId") UUID dayId);

    1 usage  sorokivski
    @Query("SELECT e FROM TrainDay e LEFT JOIN FETCH e.exercises WHERE e.id = :dayId")
    Optional<TrainDay> findByIdWithExercises(@Param("dayId") UUID dayId);

    1 usage  sorokivski
    @Query("SELECT td.plan FROM TrainDay td WHERE td.id = :id")
    GGTrainPlan findPlanById(@Param("id") UUID id);

    1 usage  sorokivski
    @Query("select td from TrainDay td where td.addedBy = :user")
    Page<TrainDay> findByOwner(GetGrowthUser user, Pageable pageable);

    1 usage  sorokivski
    @Query("select td from TrainDay td where td.addedBy = :user or td.available is true")
    Page<TrainDay> findByOwnerAndAvailable(GetGrowthUser user, Pageable pageable);
```

Рисунок 2.2.3 – Інтерфейс із модуля репозиторіїв

В інтерфейсі зазначено, над яким класом (таблицею) в даному репозиторії здійснюються операції і тип ідентифікатора, який використовується. Анотація `@Query("select")` – вказує на персоналізований запит, що виконується над кортежами, які містять дані, можна також здійснювати операції додавання в результат даних з інших таблиць.

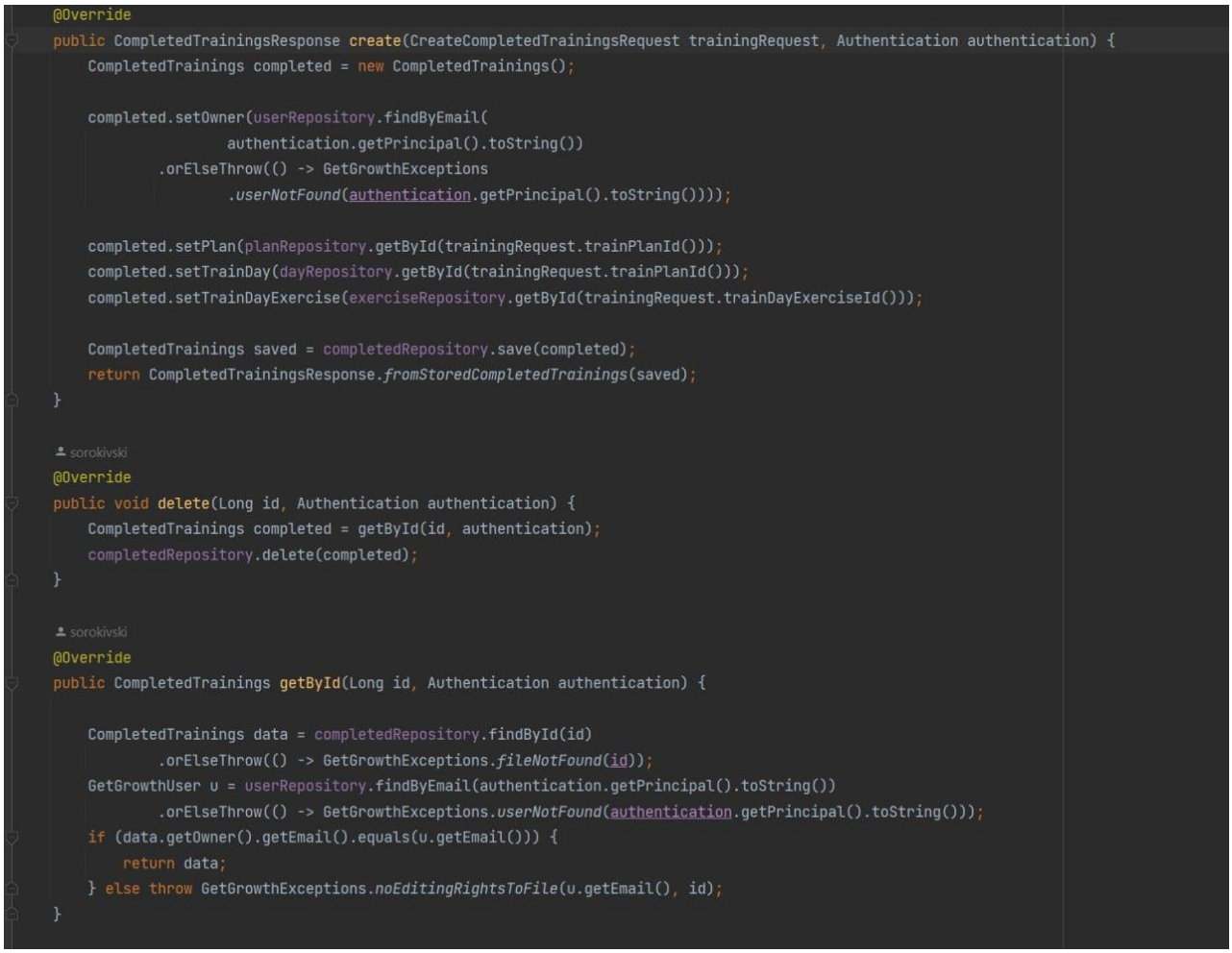
В модулі сервісів визначені інтерфейси із визначенням операцій, які здійснюються над сутностями і класи, в яких реалізовуються ці методи. Приклад подано на рисунку 2.2.4.



```
package com.example.demo.service.observation;


import ...

3 usages  1 implementation  sorokivski
public interface CompletedTrainingsService {

    1 implementation  sorokivski
    CompletedTrainingsResponse create(CreateCompletedTrainingsRequest trainingRequest, Authentication authentication);

    1 implementation  sorokivski
    void delete(Long id, Authentication authentication);

    1 implementation  sorokivski
    CompletedTrainings getById(Long id, Authentication authentication);

    1 usage  1 implementation  sorokivski
    Page<CompletedTrainingsResponse> getAllByUser(OffsetDateTime from, OffsetDateTime to, Authentication authentication, Pageable pageable);

    1 usage  1 implementation  sorokivski
    Page<CompletedTrainingsResponse> getCompletedTrainPlansByUser(OffsetDateTime from, OffsetDateTime to, Authentication authentication, Pageable

    1 usage  1 implementation  sorokivski
    Page<CompletedTrainingsResponse> getCompletedTrainDaysByUser(OffsetDateTime from, OffsetDateTime to, Authentication authentication, Pageable

    1 usage  1 implementation  sorokivski
    Page<CompletedTrainingsResponse> getCompletedExercisesByUser(OffsetDateTime from, OffsetDateTime to, Authentication authentication, Pageable
}
```

2.2.4 – Інтерфейс сервісу для сутності

В інтерфейсі описано, які саме операції можуть здійснюватися для певної сутності із предметної області, а також тип даних, які передають в метод і повертаються в результаті опрацювання. Результати реалізації методів в класі подано на рисунку 2.2.5.

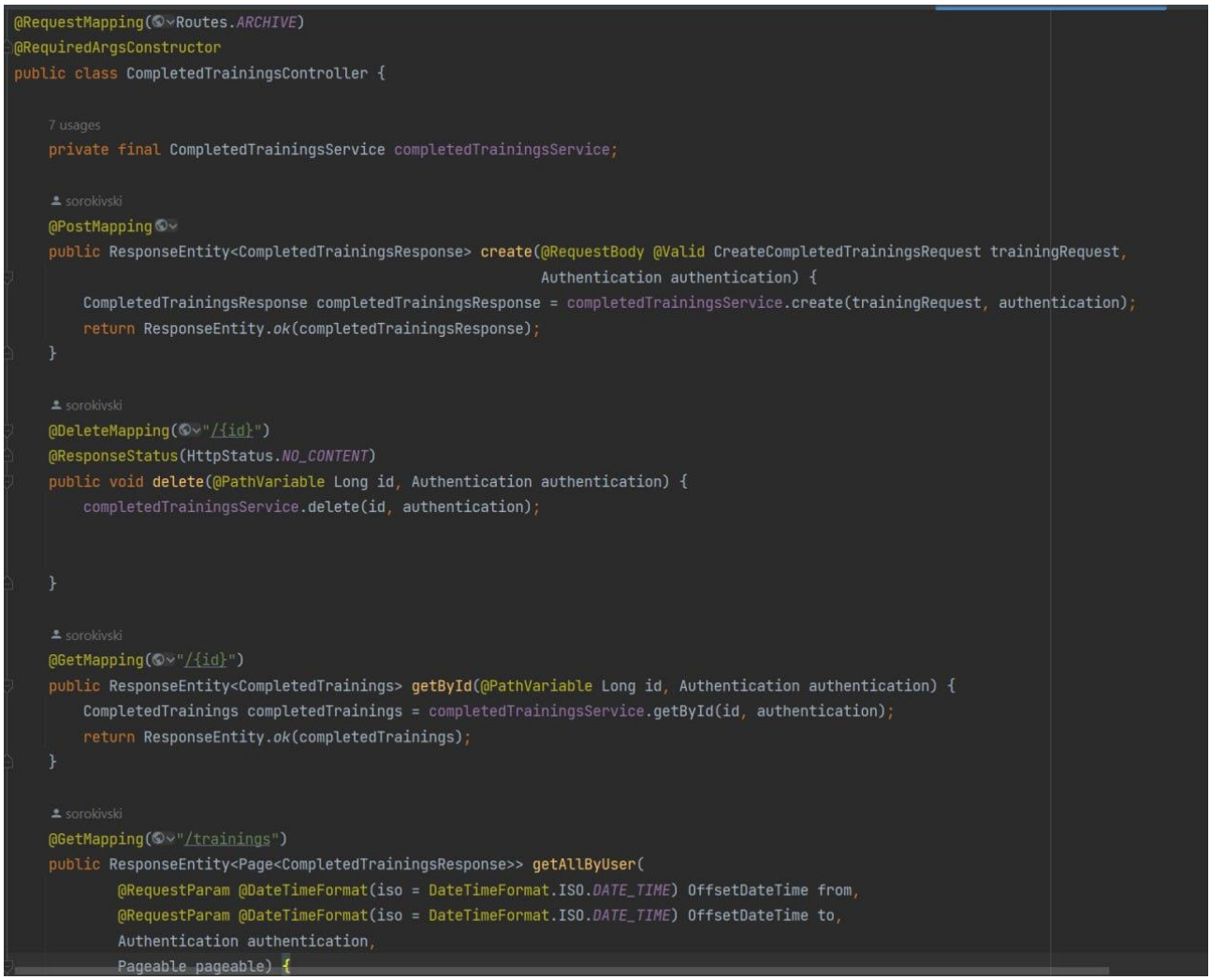


```
@Override
public CompletedTrainingsResponse create(CreateCompletedTrainingsRequest trainingRequest, Authentication authentication) {
    CompletedTrainings completed = new CompletedTrainings();

    completed.setOwner(userRepository.findByEmail(
                    authentication.getPrincipal().toString())
            .orElseThrow(() -> GetGrowthExceptions
                    .userNotFound(authentication.getPrincipal().toString())));

    completed.setPlan(planRepository.getById(trainingRequest.trainPlanId()));
    completed.setTrainDay(dayRepository.getById(trainingRequest.trainPlanId()));
    completed.setTrainDayExercise(exerciseRepository.getById(trainingRequest.trainDayExerciseId()));

    CompletedTrainings saved = completedRepository.save(completed);
    return CompletedTrainingsResponse.fromStoredCompletedTrainings(saved);
}

sorokivski
@Override
public void delete(Long id, Authentication authentication) {
    CompletedTrainings completed = getById(id, authentication);
    completedRepository.delete(completed);
}

sorokivski
@Override
public CompletedTrainings getById(Long id, Authentication authentication) {

    CompletedTrainings data = completedRepository.findById(id)
            .orElseThrow(() -> GetGrowthExceptions.fileNotFound(id));
    GetGrowthUser u = userRepository.findByEmail(authentication.getPrincipal().toString())
            .orElseThrow(() -> GetGrowthExceptions.userNotFound(authentication.getPrincipal().toString()));
    if (data.getOwner().getEmail().equals(u.getEmail())) {
        return data;
    } else throw GetGrowthExceptions.noEditingRightsToFile(u.getEmail(), id);
}
```

Рисунок 2.2.5 – Клас реалізації описаних в інтерфейсі методів

Класи-контролери створені в директорії з відповідними назвами. Ідею таких компонентів системи показано на прикладі контролера сутності CompletedTrainings із викликом описаних в попередньому лістингу методів.

В наступному класі описано методи із анотаціями, що вказують на їх тип(get, post, patch, delete), наприклад `@GetMapping("/train-plans")` – вказує, що описаний метод спрямований на отримання даних, та вказано відносний шлях для доступу до цієї операції (`train-plans`). `Page<CompletedTrainingsResponse>` – декларує тип отримуваних даних, в даному випадку, це сторінка із представленнями виконаних тренувань (планів, днів, вправ), сформований список повертається у вигляді JSON, та може бути представлене на стороні користувача у зручному вигляді.



```
@RequestMapping(Routes.ARCHIVE)
@RequiredArgsConstructor
public class CompletedTrainingsController {

    7 usages
    private final CompletedTrainingsService completedTrainingsService;

    sorokivski
    @PostMapping
    public ResponseEntity<CompletedTrainingsResponse> create(@RequestBody @Valid CreateCompletedTrainingsRequest trainingRequest,
                                                             Authentication authentication) {
        CompletedTrainingsResponse completedTrainingsResponse = completedTrainingsService.create(trainingRequest, authentication);
        return ResponseEntity.ok(completedTrainingsResponse);
    }

    sorokivski
    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void delete(@PathVariable Long id, Authentication authentication) {
        completedTrainingsService.delete(id, authentication);


    }

    sorokivski
    @GetMapping("/{id}")
    public ResponseEntity<CompletedTrainings> getById(@PathVariable Long id, Authentication authentication) {
        CompletedTrainings completedTrainings = completedTrainingsService.getById(id, authentication);
        return ResponseEntity.ok(completedTrainings);
    }

    sorokivski
    @GetMapping("/trainings")
    public ResponseEntity<Page<CompletedTrainingsResponse>> getAllByUser(
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) OffsetDateTime from,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) OffsetDateTime to,
            Authentication authentication,
            Pageable pageable) {
```

Рисунок 2.2.6 – Клас-контролер роботи із CompletedTrainingsService

Такий підхід також уможливлює виокремлення реалізацій серверної частини додатку від графічного інтерфейсу додатку, а також полегшує можливості тестування серверної частини ще до створення представлення інформаційної системи. Анотація `@RequestParam` – потрібна для створення документації та вказання які саме дані повинні бути для виклику методу, після чого із контролеру ця інформація передається у відповідні сервіси для подальшого опрацювання, а `@RequestMapping(Routes.ARCHIVE)` – вказує на шлях, за яким здійснюються запити у системі до описуваного контролеру, даний спосіб задання шляху використано для забезпечення можливості переглядати та/чи модифікувати абсолютні шляхи в окремому файлі, не здійснюючи при цьому пошуку для кожного із контролерів створюваного застосунку окремо.

2.2.2 Тестування створеної системи

Тестування дозволяє оцінити якість продукту, що розроблюється. Якісний продукт повинен відповідати поставленим до нього вимогам. Програмна система повинна реалізовувати всі необхідні варіанти використання і не мати дефектів. Крім того, програмна система повинна бути надійною (не має бути зависань, аварійних відмов і т. п.), безпечною, забезпечувати потрібну продуктивність, бути зручною в експлуатації, розширюваною. Таким чином, тестування представляє собою процес аналізу програмної системи, спрямований на виявлення дефектів і на оцінку властивостей програмної системи. Тож для перевірки на відповідність системи до вимог варто переглянути загальну структуру проєкту та порівняти її із тією, що зазначена у специфікації, зручний інтерфейс для цього доданий у програму через документацію використовуваного фреймворку Spring, цим компонентом виступає openapi swagger-ui (рисунок 2.2.2).

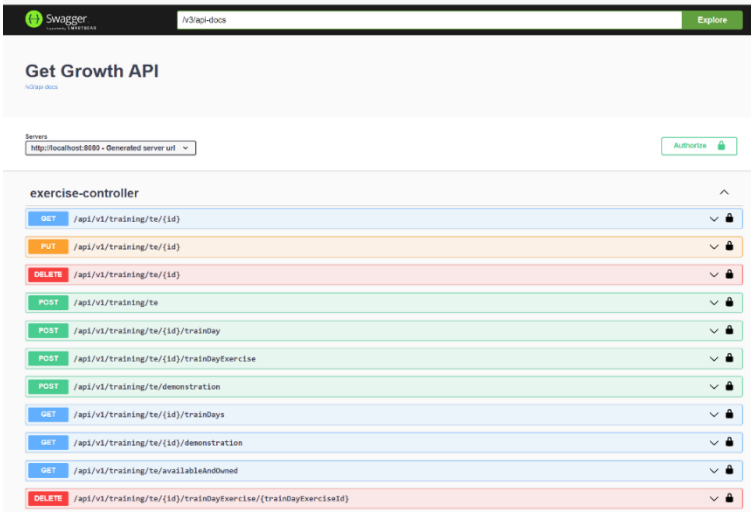


Рисунок 2.2.7 – Порівняння структури проекту із специфікацією

Переглянувши вміст програмного продукту та переконавшись, що всі компоненти заявлені у специфікації присутні у системі, можна здійснювати перевірку на працездатність цих атрибутів та відповідність результатів до

виконуваних запитів. До прикладу здійснення запиту на додавання вживаної добавки повинне супроводжуватись доданням у вже наявну базу даних аналогічного запису.

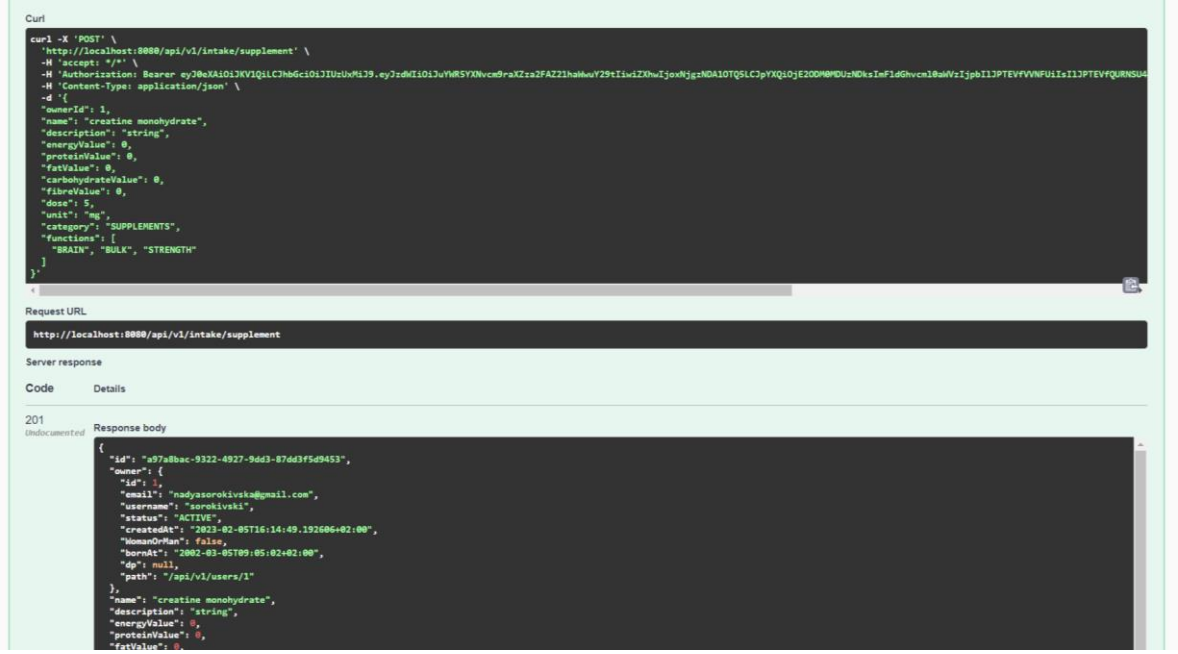


Рисунок 2.2.8 – Здійснення запиту на додання вживаної добавки

На рисунку видно, що при виконанні запиту, повертається JSON-об’єкт із даними, та код операції 200, що підтверджує успішність виконання. Перевірити наявність нового рядку у базі даних можна і безпосередньо, як зображено на рисунку 2.2.4.

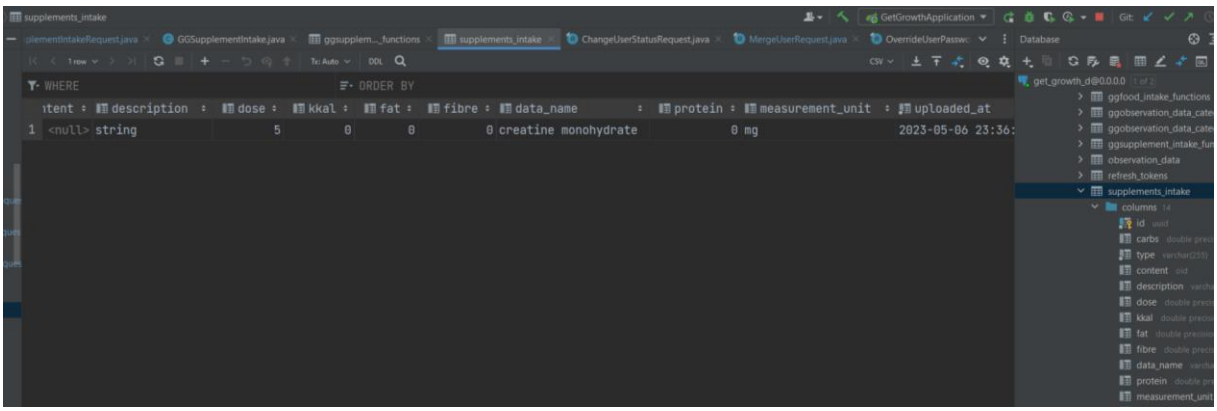


Рисунок 2.2.9 – Перевірка успішності операції із додання рядку

Також для здійснення операцій у системі, користувач повинен пройти процес реєстрації чи авторизації, із використанням swagger-ui цей процес відбувається наступним чином: надсилається запит із даними користувача (логін та пароль) на отримання токену доступу, після чого цей токен доступу використовується як заголовок для кожного із виконуваних запитів аж поки термін його дії не спливе. Приклад надсилання запиту на отримання токену авторизації показано на рисунку.

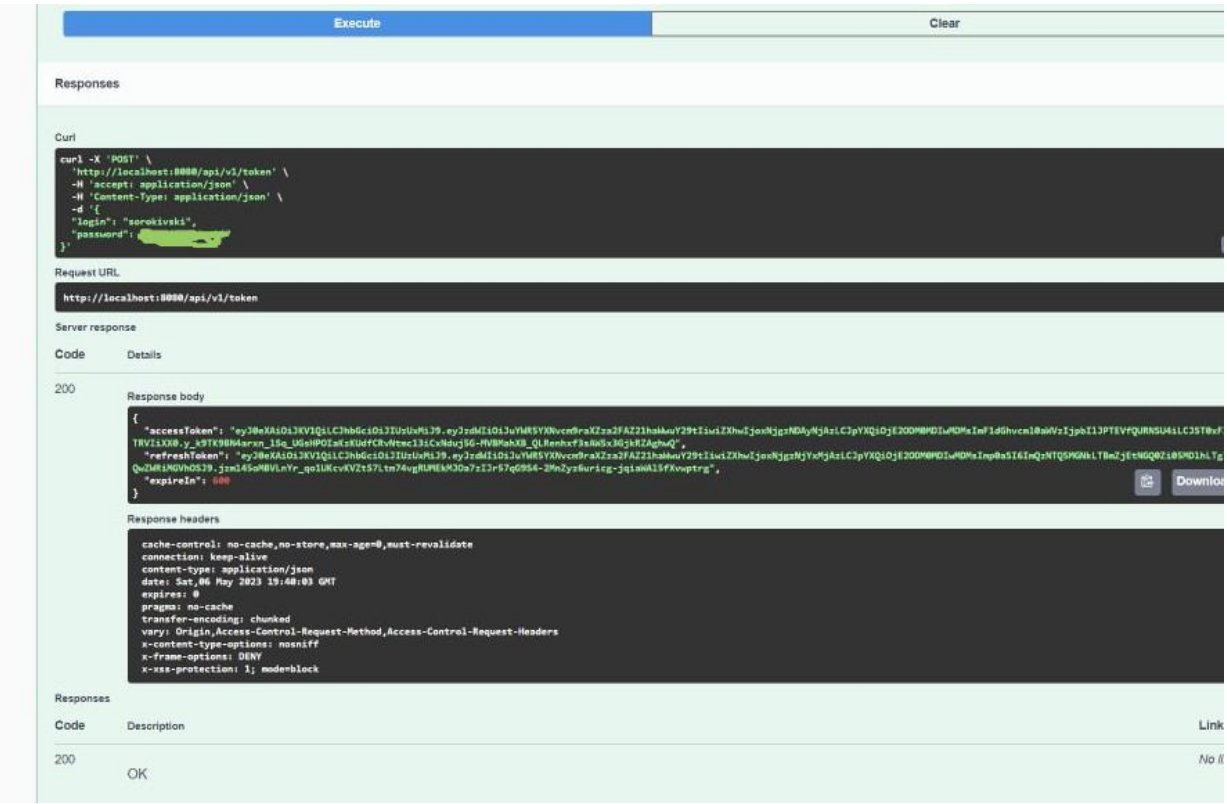


Рисунок 2.2.10 – Запит на авторизацію в програмі

Отож, як видно із цієї роботи – тестування займає не менш важливу частку від загального обсягу виконуваних у процесі розробки робіт, та насамперед допомагає у визначенні та забезпеченні якості розроблюваного програмного продукту та допомагає уникнути багатьох непорозумінь чи труднощів, які б запевне з’являлись у випадку його відсутності.

2.2.3 Графічний інтерфейс системи

Інтерфейс слугує містком між функціоналом/логікою програми та користувачами, та являє собою сукупність засобів для обробки та представлення інформації у зручному для користувачів вигляді; програмні системи використовують графічне представлення із багатовіконним режимом, зміною кольорів, розмірів, видимості (прозорі, напівпрозорі, невидимі фрагменти), розташування, сортування елементів віджетів та екранів, гнучкі налаштування як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступність багатокористувацьких налаштувань. Приклад реалізації інтерфейсу для сторінки профілю в програмі показано на рис 2.2.6. Інші сторінки додатку представлені в додатку Б.

Рисунок 2.2.11 – Графічний дизайн профілю

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Менеджмент безпеки

Менеджмент системи безпеки – це система наукових знань та досвіду їх імплементації в діяльності спрямованій на досягнення безпекових цілей системи безпеки через використання праці, інтелекту та мотивів поведінки людей. Робота відповідальних за цю діяльність полягає у поєднанні та координуванні використання зазначених вище ресурсів для досягнення цілей забезпечення безпеки охорони праці.

Захист персональних даних та інших важливих інформаційних ресурсів необхідний для забезпечення безпеки життєдіяльності як на рівні окремих осіб, так і на рівні організацій та держав. В Україні та Європейському Союзі важливість захисту даних описуються в законодавчих нормах, які регулюють цю сферу. В Україні це закон "Про захист персональних даних"[13], який визначає правила обробки та захисту персональних даних громадян. У Європейському Союзі введено загальний регламент захисту даних (GDPR), який встановлює стандарти безпеки даних та права осіб, чиї дані обробляються.

Безпека даних пов'язана з безпекою життєдіяльності у багатьох аспектах. Перш за все, втрата або незаконне використання конфіденційної інформації може призвести до фінансових збитків, шахрайства, крадіжки особистих даних та ідентичності. Крім того, у разі порушення безпеки даних може постраждати репутація користувачів, що веде до втрати довіри клієнтів і партнерів. Оскільки в розроблюваній системі зберігаються конфіденційні медичні та інші особисті дані, є критичним забезпечення безпеки цієї інформації, тому розглядається питання безпекового характеру. Про важливість цього етапу свідчать постійні проблеми пов’язані із витоком даних з різноманітних ресурсів, тож для систем, що працюють із реальною інформацією, варто додати функціонал для захисту від потенційного витоку зважаючи на документи із врегулюванням прав та правил зберігання, обробки та використання особистих даних європейських інтернет-користувачів

95/46/ЄС, що твердить про те, що персональну інформацію можна опрацьовувати у визначених випадках, а власники онлайн-ресурсів зобов’язані убезпечити її від несанкціонованого доступу.

Ключові аспекти безпеки та менеджменту безпеки в цій сфері:

- Захист прав людини. Згідно з Конституцією України, права людини повинні бути захищені. Додаток повинен дотримуватись правил і політик щодо захисту особистих даних та забезпечення конфіденційності. Користувачі мають мати можливість контролювати свої дані, надавати згоду на їх використання та брати участь у процесі прийняття рішень, пов'язаних з їх даними.
- Автентифікація та авторизація. Використання Spring Security та JWT дозволяє забезпечити автентифікацію користувачів та надання їм прав доступу на основі їх ролей та привілеїв. Коректна ідентифікація користувача та забезпечення необхідних прав доступу є основою безпеки сервісу.
- Захист від викрадення даних. Важливо розробити заходи безпеки, які допоможуть уникнути таких атак, як перехоплення сесій, міжсайтовий скриптінг (XSS), SQL-injection та ін. Використання валідації даних, обмеження параметрів запитів, захист від вразливостей у Spring Security та добре налаштована політика безпеки допоможуть зменшити ризик таких атак.
- Конфіденційність даних. Захист конфіденційності особистих даних користувачів є важливим аспектом. Використання захищеного способу передачі даних, шифрування конфіденційної інформації у базі даних та забезпечення безпеки токенів JWT допомагають забезпечити конфіденційність даних.
- Зберігання даних. Використання надійного і безпечної системи управління даними користувачів, зокрема для зберігання паролів, правильне хешування, обмеження доступу до бази даних та застосування правильних політик зберігання допомагають запобігти витоку інформації.

– Поновлення та усунення вразливостей. Важливо регулярно оновлювати залежності, фреймворки та бібліотеки, щоб усунути вразливості і забезпечити безпеку додатку. Використання останніх версій програмних засобів та регулярних перевірок на вразливості є вимогами для ефективного менеджменту безпеки.

Ефективний менеджмент безпеки даних включає надання пріоритету захисту даних, усвідомлення важливості безпеки серед усіх членів організації, навчання персоналу щодо правильних практик безпеки та встановлення системи постійного моніторингу та оновлення заходів безпеки.

Менеджмент безпеки даних сприяє побудові надійного та безпечного середовища для збереження і обробки інформації, що забезпечує довіру, конфіденційність між користувачем та компанією, що пропонує продукт та недоступність незаконного доступу до особистої інформації стороннім особам чи зловмисникам.

Правильне впровадження і підтримка в робочому стані системи управління охороною здоров’я і безпеки користувачі може бути частиною стратегії належної виробничої практики, яка є ефективним й рентабельним довгостроковим вкладенням засобів у майбутнє продукту. Це, у свою чергу, веде до того, що компанії, які отримали сертифікати на системи управління охороною здоров’я та безпекою життєдіяльності, зокрема і у використанні персональних даних у програмах, вимагають у своїх субпідрядників аби ті також контролювали процеси та управляли ризиками у зазначеній сфері.

3.2 Аналіз небезпек та рекомендації з використання системи

3.2.1 Опис чинників, що спричиняють травматизм

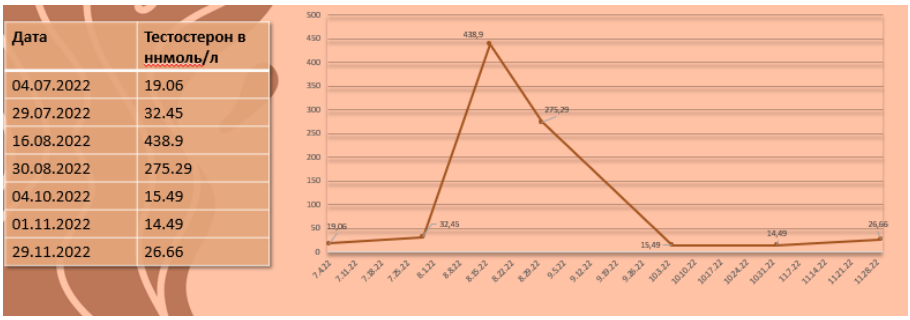
Оскільки у програмному продукті вбачається застосування фізичних навантажень, важливо оцінити можливі травматичні фактори при виконанні тренувань. Перенавантаження м'язів та суглобів може призвести до розтягнень, забоїв, розривів м'язів або зв'язок. Неправильна техніка виконання вправ, надмірне навантаження або недостатній період відновлення можуть бути чинниками, що сприяють таким травмам. Відсутність або недостатня розминка перед тренуванням може збільшити ризик отримання травм. Нерозігріті м'язи та суглоби менш гнучкі, що може призвести до розтягнень або інших травм. Неправильне використання або недоліки у спортивному обладнанні, постійне тренування без достатнього відпочинку може призвести до перенавантаження тіла, зниження імунітету та підвищення ризику травм. Недостатня кількість сну, неправильне харчування та перевтома можуть також сприяти травмам.

3.2.2 Медико-біологічні і психологічні причини травматизму

Особливу небезпеку відносно підвищення спортивного травматизму представляють стимулюючі препарати. Стимулятори нервової системи - похідні фенаміну (аналог гормонів адреналіну та норадреналіну), що призводять до поліпшення спортивних результатів за рахунок усунення охоронного гальмування, можуть призвести до важких наслідків у здоров'ї спортсменів. Надмірне застосування анаболічних стероїдів, на жаль, характерне для ряду видів спорту, здатне призвести до зміни метаболізму сполучної тканини і зниження міцності сухожиль і зв'язок, збільшення ризику їх розривів. Це підтверджується і великою кількістю спонтанних розривів у спортсменів, що спеціалізуються в швидкісно-силових видах спорту. [19]

3.2.3 Інструктаж із прийому гормональних препаратів

Для кожного з застосовуваних препаратів необхідно проводити медичний контроль та відстеження лабораторних аналізів. Наприклад, при прийомі курсу препарату омнадрен, дозою 250мг варто також відслідкувати рівень гормону до проведення курсу і після аби відслідкувати норму рівня та відновлення. Приклад графіку відстеження подано на рисунку 3.2.



Дата	Тестостерон в нмоль/л
04.07.2022	19.06
29.07.2022	32.45
16.08.2022	438.9
30.08.2022	275.29
04.10.2022	15.49
01.11.2022	14.49
29.11.2022	26.66

3.2 – Рисунок результатів відстеження аналізів

Для максимально об’єктивного оцінювання змін потреб організму в калорійності їжі, з раціону варто виключити солодкі та жарені страви, транс-жири, обмежити вживання солі, а також, щоб не стимулювати апетит, обмежити використання спецій (використовуються лише рослинні спеції). Для пришвидшення метаболізму та стимуляції синтезу тестостерону рекомендовані помірні фізичні навантаження. В перші половині першого етапу зазвичай прослідковується зменшення добової калорійності, проте у другій половині калорійність відновлюється. У другому етапі на фоні різкого зростання концентрації гормону калорійність їжі зростатиме майже в двічі. Наприкінці етапу ці показники дещо зменшуватимуться. У третьому етапі добова калорійні ще дещо знизиться.

Результати досліджень вказують на пряму залежність потреб організму в калоріях від концентрації гормону в організмі. Також вдалось зафіксувати зміни антропометричних даних і їх залежність від концентрації тестостерону.

Результатом першого етапу вважається ефективність застосування дієтотерапії і помірних фізичних навантажень з метою терапії андрогено-дефіциту та гормонального ожиріння у чоловіків. (зростання вільного тестостерону на 70% за місяць). В другому етапі контрольоване введення гормону показує позитивний вплив на фізичні дані піддослідного, а також відсутність побічних ефектів на момент завершення досліду – можуть слугувати доказом безпечності його використання у терапії та спорті. Підсумовуючи результат третього етапу, звертаємо увагу на те, що зміни антропометричних та фізичних даних залишаються стабільними, це може бути корисним для спортивної медицини [4].

Для оцінки психологічного стану при проведені досліджень використовуються опитувальники та проводиться аналіз результатів із порівняннями щодо норми та накладенням на лабораторні результати. Оцінка психічного стану проводиться щотижнево та оцінка рівня гормону – кілька разів впродовж періоду проведення експерименту. Про стан здоров'я можна використовувати опитувальник PHQ-9, опитувальник порушення сну (коротка форма 8a), оцінка знемоги(коротка форма 8a), шкалу загального враження пацієнта щодо важкості симптомів безсоння (PGIS), EQ-5D-5L та стан здоров'я, шкала ASEX, SIGMA, CGI-S, PWC, HAM- A [20].

3.2.4 Профілактика спортивного травматизму

Спорт є не лише джерелом задоволення та активного життя, але й потенційною причиною травм. Травми, отримані під час фізичних навантажень, можуть мати серйозні наслідки для спортсменів, впливати на їхню фізичну активність та загальний стан здоров'я. Однак, шлях до безпечного спортивного досвіду може бути побудований за допомогою ефективної профілактики спортивних травм.

Для отримання додаткової інформації про профілактику спортивних травм, рекомендується консультуватись з фахівцями в галузі спортивної медицини, фізіотерапевтами та вивчати літературу, що присвячена безпеці у спорті.

Ключовим аспектом профілактики спортивних травм є належне навчання техніці та правилам виконання вправ. Користувачі повинні мати глибоке розуміння своєї дисципліни, включаючи коректну постановку ніг, рук, спини та голови при виконанні вправ. Також важливо включати динамічну розтяжку перед тренуваннями, що дозволить організму адаптуватись до фізичних навантажень, розігріти м’язи та сухожилля, що зменшить ризик отримання травм під час тренування. Належна підготовка, та відточування техніки дозволить уникнути неправильних рухів, які можуть призвести до травм впродовж збільшення навантаження в вигляді ваг, тривалості, дальності тощо. Також важливо отримувати відповідну інструкцію та нагляд тренера або складати план самостійно, для контролю виконання вправ та формування рекомендації для покращення техніки на основі здійснюваних навантажень.

Другий аспект профілактики травм – це належні розминка та підготовка перед тренуваннями. Розминка перед спортивними заняттями допомагає підготувати м'язи, суглоби та зв'язки до фізичного напруження, збільшує гнучкість та покращує кровообіг. Це допомагає зменшити ризик розтягнень, надривів та інших м'язово-скелетних травм. Крім того, важливо дотримуватися оптимального режиму тренувань, забезпечуючи не лише інтенсивність, але й відповідний період відновлення між тренуваннями. Недостатній час для відновлення може призвести до перенавантаження тіла та підвищеного ризику травм.

Третій аспект профілактики травм – це використання безпечного спортивного обладнання та захисту. Спортсмени повинні носити відповідний захист для своїх зон ризику, наприклад, шоломи, рукавички, бандажі, лямки та інші захисні пристрої.

ВИСНОВКИ

В результаті виконаної роботи було спроектовано програмний продукт. Програма, задовільняє вимоги до того, що вона повинна бути легкою і зрозумілою у використанні. Ця система спрощує процес відслідковування спортивних результатів, надає інструменти для ведення обліку персональних тренувань, спожитих речовин та ведення звітності щодо власних досягнень і їх аналітики.

Робота системи зосереджена на простій для користувача формі реалізації програми, що надає функціонал для підтримки та вдосконалення власного тіла із можливістю підібрати зручну та ефективну програму тренувань, спосіб харчування, а також ефективні харчові добавки й препарати для збалансування й забезпечення потрібних концентрацій гормонів й інших речовин, які споживаються чи виробляються в людському організмі.

Програма забезпечує можливість відстеження різноманітних чинників водночас, оскільки наявність чи відсутність одного з них впливає на отримувані результати осіб. Для самого ж відстеження та створення аналітики записуваних даних відстежуються зміни певних антропометричних показників таких як: анатомічні окружності, маса тіла, а також таких обстежень, як динамометрія та каліперометрія.

В системі наявні такі ролі: неавторизований користувач, має можливість створити акаунт чи проходити реєстрацію; авторизований користувач: ввести дані персонального профілю, додати дані для відстеження, додати категорії споживаних добавок чи продуктів, додати стилі тренувань, позначення і збереження для подальшого аналізу виконаних фізичних навантажень, відстеження прогресу та перегляд статистики щодо записаних даних. Тож із використанням такого підходу, результатом виконання є застосунок, що задовольняє всім погодженим вимогам.

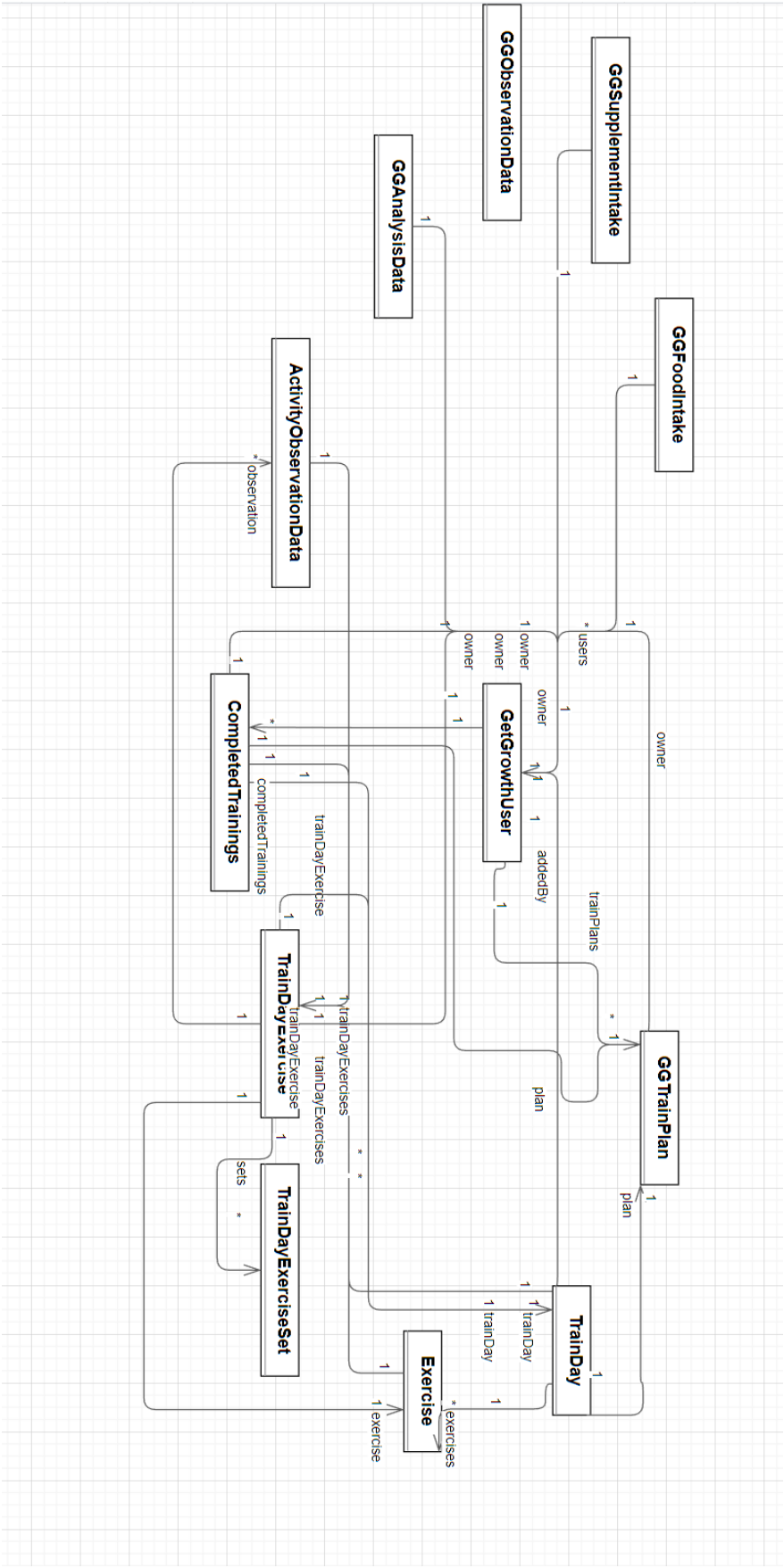
ПЕРЕЛІК ПОСИЛАНЬ

1. Огляд 9 найкращих фітнес-програм [Електронний ресурс] URL: https://www.boomfit.com/en/blog/best-fitness-apps-b22.html
2. Peter Schnohr, James H O'Keefe, Jacob L Marott. Dose of jogging and long-term mortality: the Copenhagen City Heart Study. [Режим доступу] URL: https://pubmed.ncbi.nlm.nih.gov/25660917/
3. Herman Pontzer, David A. Raichlen. Hunter-Gatherer Energetics and Human Obesity [Режим доступу] URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3405064/
4. Сороківський І., Титов В. Вплив андрогенних гормонів (тестостерон) на потреби організму та на деякі антропометричні показники ТНМУ 2023 [Режим доступу] URL: https://drive.google.com/file/d/1PaSBeg9SZoKPpvY8UBvXTRG6MbqxWmY1/view
5. Gerald Tomaro Mangine. Вплив об'єму та інтенсивності тренувань на покращення м'язової сили та об'єму. Spring Term 2015 [Режим доступу] URL: https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=2283&context=etd
6. Сарафенюк Т. Методологія швидкої розробки додатків: досл. стаття. 2013. [Електронний ресурс] URL: https://bit.ly/3IyqbIG
7. Ванюков С. RAD. Гід по методології: навч. стаття, 2020. [Електронний ресурс] URL: https://www.softermii.com/blog/rapid-application-development-model-how-and-when-to-use-it-in-your-software-project.
8. Інформація по Spring Framework [Електронний ресурс] URL: https://docs.spring.io/spring-framework/docs/6.0.0/reference/pdf/spring-framework.pdf .
9. Джонсон Р. Spring Framework – інструкції щодо використання. [Електронний ресурс] URL: https://spring.io/guides/tutorials/rest/
10. Документація по Docker [Електронний ресурс] URL: https://docs.docker.com/get-started/overview/
11. Eric Windmill. Flutter in Action. навч. посібник, Manning Publications Co. USA 2020. [Режим доступу] URL: https://www.manning.com/books/flutter-in-action
12. Документація по Flutter – інструкції щодо застосування. [Електронний ресурс] URL: https://docs.flutter.dev/cookbook
13. Конституція України. Розділ II ПРАВА, СВОБОДИ ТА ОБОВ'ЯЗКИ ЛЮДИНИ І ГРОМАДЯНИНА ст. 23, 31-32 [Електронний ресурс] URL: https://www.president.gov.ua/ua/documents/constitution/konstituciya-ukrayini-rozdil-ii

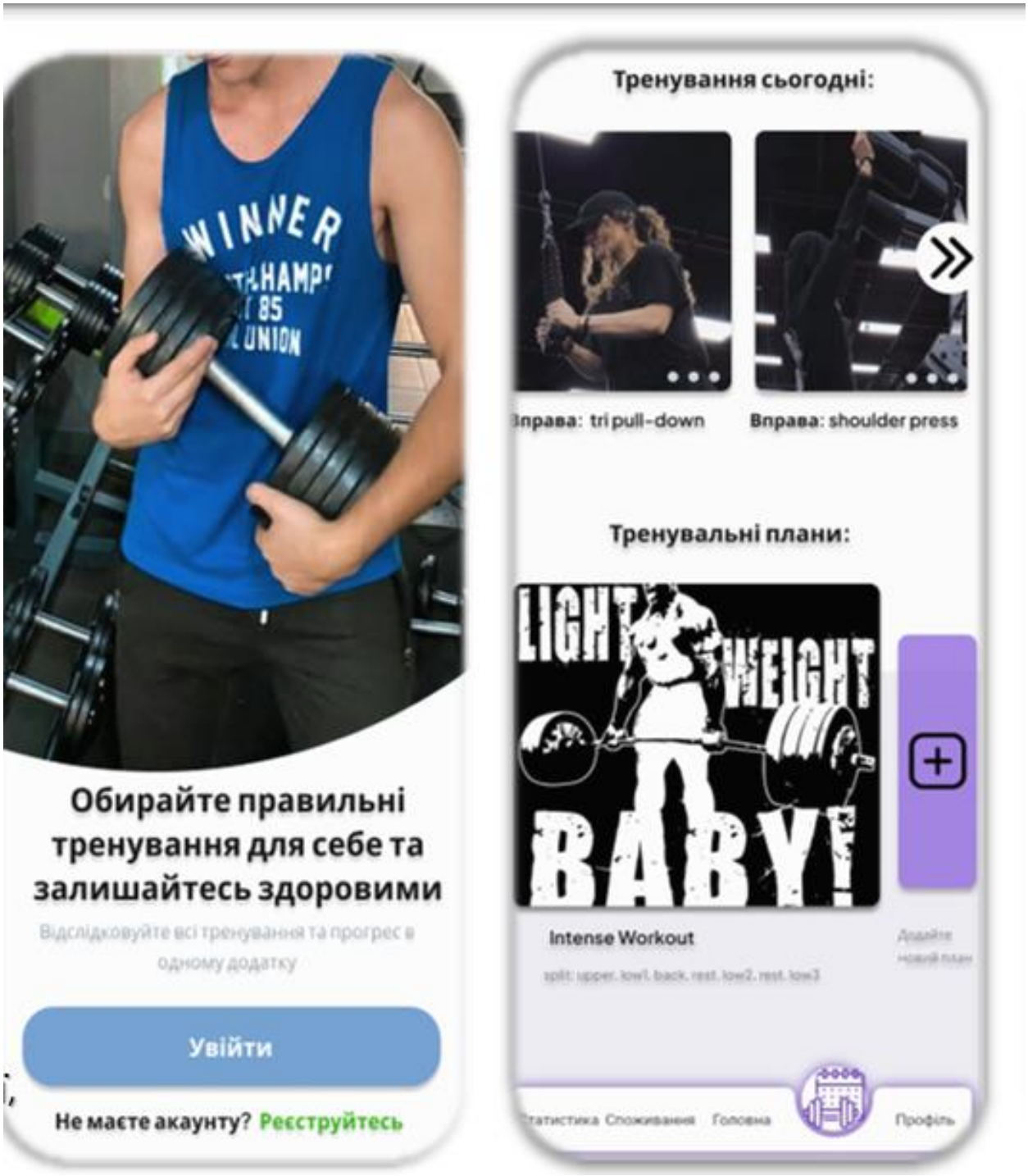
14. Мохняк С.М. Безпека життєдіяльності. Навчальний посібник. – Львів: вид. НУ „Львівська політехніка”, 2009. 264 с. [Режим доступу] URL: https://vlp.com.ua/node/2919?page=0,1
15. Sebastian Peyrott. JWT Handbook, 2016-2018 [Режим доступу] URL: https://auth0.com/resources/ebooks/jwt-handbook/thankyou
16. Закон України «Про охорону праці». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/2694-12
17. Хіменес Х. Р. Травматизм в спорті. [Електронний ресурс] URL: https://repository.ldufk.edu.ua/bitstream/34606048/3741/1/%D0%A2%D0%B5%D0%BC%D0%B0%205%20%D0%A2%D1%80%D0%B0%D0%B2%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%BC%20%D1%83%20%D1%81%D0%BF%D0%BE%D1%80%D1%82%D1%96.pdf
18. Сороківський І., Бучко А., Титов В. Вплив андрогенних гормонів (тестостерон) на психічний стан людини ТНМУ 2023 [Режим доступу] URL: https://drive.google.com/file/d/1PaSBeg9SZoKPpvY8UBvXTRG6MbqxWmY1/view

ДОДАТКИ

Додаток А. Повна діаграма класів інформаційної системи



Додаток Б. Сторінки графічного інтерфейсу



Додаток В. Диск із програмними файлами