

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТРЕНУВАНЬ В
ТРЕНАЖЕРНОМУ ЗАЛІ
DEVELOPMENT OF A MOBILE APPLICATION FOR GYM TRAINING

Виконав: студент(ка) 4 курсу, групи КНТ-139
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГВОЗДИЛКО Д.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
— ” — 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

ГВОЗДИЛКА Данила Олександровича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проекту (роботи) Розробка мобільного застосунку для тренувань в тренажерному залі.

Development of a mobile application for gym training.

керівник проекту (роботи) Федорончак Тетяна Василівна, доцент,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від —4— квітня 2023 року № 87

2. Строк подання студентом проекту (роботи) 23 травня 2023 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ФЕДОРОНЧАК Т.В. доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 13 » лютого 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Срок виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Данило ГВОЗДИЛКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проекту (роботи)

_____ Тетяна ФЕДОРОНЧАК
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 170 с., 33 рис., 3 табл., 3 дод., 20 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, ТРЕНУВАННЯ В ТРЕНАЖЕРНОМУ ЗАЛІ, ANDROID, ХАРЧУВАННЯ, KOTLIN

Об'єкт дослідження – програмні засоби для тренування в тренажерному залі.

Предмет дослідження – мобільні застосунки для тренування в тренажерному залі.

Мета роботи – створення мобільного застосунку для правильного та безпечного тренування в тренажерному залі.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Kotlin, мобільний пристрій з операційною системою Android.

Результати. Створено мобільний-застосунок, який взаємодіє з користувачем та дає необхідну інформацію про тренування.

Висновки. Розроблено мобільний-застосунок для правильного та безпечного тренування в тренажерному залі.

Галузь використання – особисте використовування для тренування в тренажерному залі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 170 pages, 33 figures, 3 tables, 3 appendixes, 20 sources.

MOBILE APPLICATION, TRAINING IN THE GYM, ANDROID, NUTRITION, KOTLIN.

The object of research is software for training in a gym.

The subject of the research is mobile applications for training in the gym.

The goal of the work is to create a software system for proper and safe training in the gym.

Materials, methods and technical means: object-oriented programming, Kotlin programming language, mobile device with the Android operating system.

Results. A mobile application has been created that interacts with the user and provides the necessary information about training.

Conclusions. A mobile application has been developed for proper and safe training in the gym.

The field of use – personal use for training in the gym.

ЗМІСТ

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Вступ.....	9
1.2 Дослідження програмних засобів для тренувань в тренажерному залі	10
1.3 Висновки за розділом 1	17
2 Матеріали і методи.....	18
2.1 Вибір мови програмування	18
2.2 Вибір архітектури.....	20
2.3 Вибір системи впровадження залежностей.....	23
3 Опис програми.....	27
3.1 Функціональні вимоги до програми та вимоги до інтерфейсу	27
3.2 Розбір UI - екранів.....	28
3.3 Файли та методи.....	29
4 Експлуатація, тестування та експериментальне дослідження програми.....	37
4.1 Призначення й умови застосування програми.....	37
4.2 Вхідні та вихідні дані.....	37
4.3 Тестування та результати роботи за стосунку	38
Висновок	51
Перелік джерел посилання	52
Додаток А Технічне завдання	55
Додаток Б Текст програми	59
Додаток В Слайди презентації.....	164

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API — Application Programming Interface;

ОС — Операційна система;

ПЗ — Програмне забезпечення.

ВСТУП

Розробка мобільних застосунків для тренувань у спортзалі – актуальна і важлива тема в сучасному світі фітнесу та здорового способу життя. Зі зростанням популярності тренажерних залів та інтересу до тренувань багато людей шукають зручні та ефективні способи покращити свою фізичну форму. Розробка мобільних застосунків може стати відмінним рішенням, щоб зробити навчання більш цікавим, ефективним і доступним у будь-який час і з будь-якого місця. Цей засіб може включати в себе комплекси вправ, інформацію про правильне виконання вправ, інформацію про харчування та здоровий спосіб життя тощо. Такий застосунок буде не лише корисним інструментом для тренувань, а й допоможе зберегти статистику тренувань та відстежувати прогрес. Таким чином, призначення даної роботи є створення мобільного застосунку для правильних та безпечних тренувань у тренажерному залі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вступ

Розділ "Аналіз предметної області" є одним з ключових етапів розробки мобільного застосунку для тренування в тренажерному залі. У цьому розділі ми детально дослідимо та проаналізуємо основні аспекти, що стосуються тренувань в тренажерному залі, такі як фізіологічні та психологічні аспекти тренування, типи вправ та тренажерів, які використовуються для різних груп м'язів, а також відповідність між цими вправами та фізичними цілями, які має користувач.

Фізіологічні та психологічні аспекти тренувань мають вирішальне значення для розуміння тренувального процесу та досягнення бажаних результатів. Фізіологічні аспекти пов'язані з реакцією організму на фізичну активність, включно зі змінами в таких процесах, як серцево-судинна і дихальна системи, обмін речовин і робота м'язів. Розуміння фізіологічних аспектів допомагає вибрати найефективніші вправи та обладнання для досягнення фізичних цілей.

Психологічні аспекти охоплюють аспекти, що впливають на успіх тренування, такі як мотивація, концентрація, та емоції. Розуміння психологічних аспектів може допомогти залишатися мотивованим, зосередитися на тренуваннях і долати труднощі з меншими втратами.

Розуміння фізіологічних і психологічних аспектів тренувань може допомогти користувачам вибрати правильні вправи і тренувальні плани, які дадуть найбільш ефективні результати для них.

Існує безліч різних типів вправ і тренажерів, які можна використовувати для розвитку різних груп м'язів. Важливо розуміти, що кожна вправа має свої унікальні переваги і підходить для досягнення конкретних фізичних цілей.

Для розвитку м'язів верхньої частини тіла можуть використовуватися вправи, спрямовані на м'язі грудей, спини й плечей, як-от жим гантелей, жим лежачи, тощо.

Для зміцнення м'язів нижньої частини тіла використовуються вправи, спрямовані на м'язи стегон і сідниць, як-от присідання зі штангою.

Крім того, важливо розуміти, що вправи можуть бути спрямовані на досягнення різних фізичних цілей, таких як збільшення м'язової маси, витривалості та координації. Наприклад, вправи з важкими вагами можуть сприяти на нарощуванню м'язової маси, а вправи з легкими вагами і великою кількістю повторень - витривалості.

Розуміння взаємозв'язку між різними вправами і тренажерами та фізичними цілями допоможе вибрати найкращі вправи для досягнення бажаних результатів.

Після проведення аналізу ми зможемо краще зрозуміти потреби та вимоги наших користувачів та розробити мобільний застосунок, який найбільш ефективно допоможе їм досягати своїх фізичних цілей.

Таким чином, інформація з усіх перелічених попередніх пунктів, обумовлює розробку відповідних програмних засобів.

1.2 Дослідження програмних засобів для тренувань в тренажерному залі

Нині існує значна кількість програмних засобів у вигляді android-застосунків, що використовуються для надання інформації для тренування в тренажерному залі. Для вивчення програмних засобів було обрано різні додатки для тренувань в тренажерному залі.

Застосунок Nike Training Club (рис. 1.1).

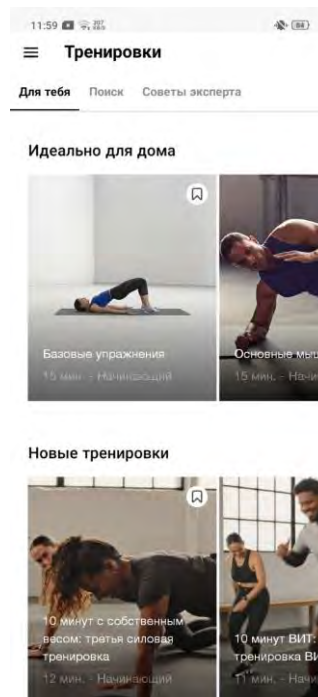


Рисунок 1.1 – Застосунок Nike Training Club

Переваги:

- відсутність вбудованих покупок;
- велика кількість програм тренувань;
- детальний аудіоспровід;
- наявність розділу с корисними статтями;
- привабливий дизайн.

Недоліки:

- необхідність авторизації;
- відсутність довідника вправ;
- велика вага програми.

Застосунок Adidas Training (рис. 1.2).

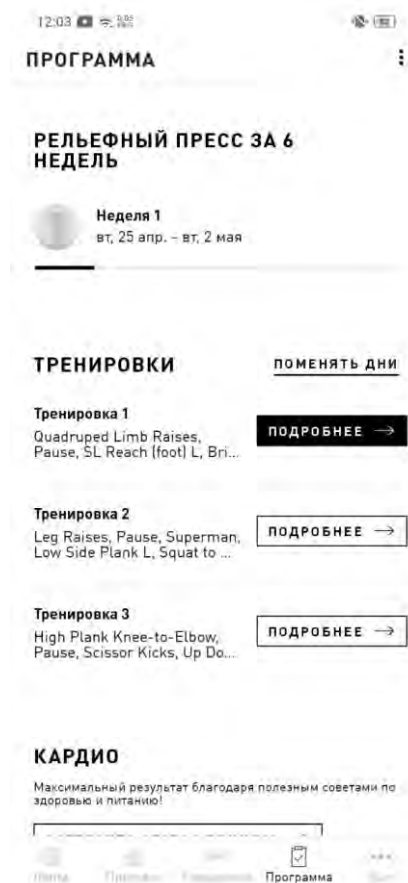


Рисунок 1.2 – Застосунок Adidas Training

Переваги:

- відсутність вбудованих покупок;
- велика кількість програм тренувань;
- детальний аудіоспровід;
- наявність розділу с корисними статтями;
- привабливий дизайн.

Недоліки:

- необхідність авторизації;
- відсутність довідника вправ;
- велика вага програми.

Застосунок GymUp (рис. 1.3).

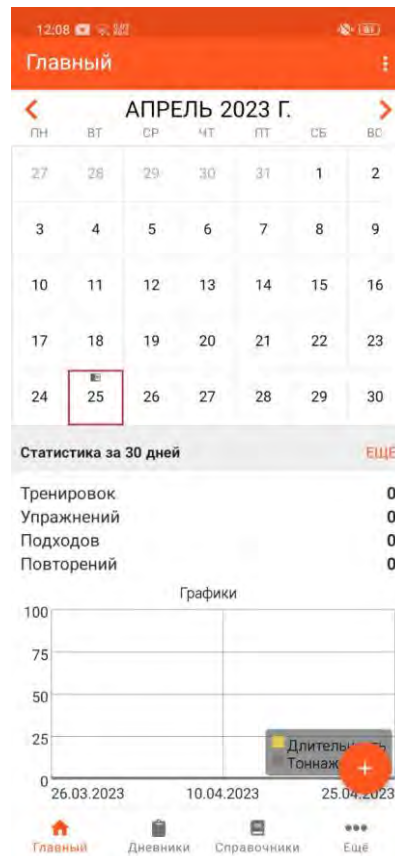


Рисунок 1.3 – Застосунок GymUp

Переваги:

- відсутність вбудованих покупок;
- велика кількість програм тренувань;
- наявність розділу с корисними статтями;
- мала вага додатку.

Недоліки:

- необхідність авторизації;
- відсутність аудіоспроводження.

Застосунок Seven (рис. 1.4).

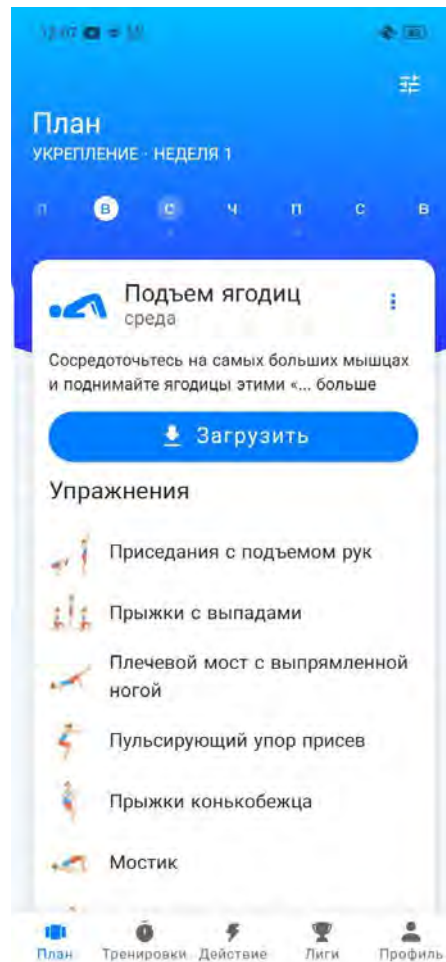


Рисунок 1.4 – Застосунок Seven

Переваги:

- мала вага додатку;
- середній ступінь науковості пояснення вправ;
- детальний аудіоспровід;
- наявність розділу з корисними статтями;
- привабливий дизайн.

Недоліки:

- невелика кількість програм;
- відсутність довідника вправ;
- рекламні матеріали, що відволікають від основного контенту;
- довгий час запуску програми.

Застосунок Workout Trainer (рис. 1.5).

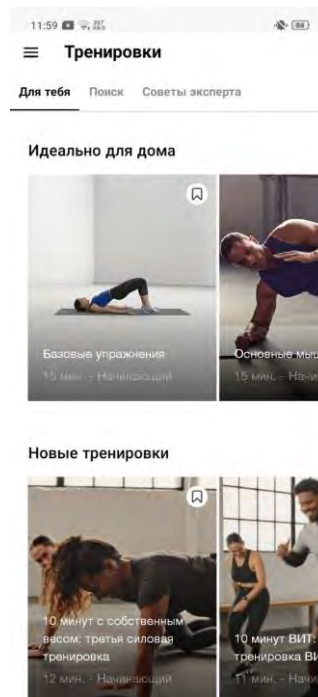


Рисунок 1.5 – Застосунок Workout Trainer

Переваги:

- відсутність вбудованих покупок;
- велика кількість програм тренувань;
- детальний аудіоспровід;
- наявність розділу с корисними статтями;
- привабливий дизайн.

Недоліки:

- необхідність авторизації;
- відсутність довідника вправ;
- велика вага програми.

Порівняльний аналіз програмного забезпечення для тренування в тренажерному залі наведено у табл. 1.1.

Таблиця 1.1 - Порівняльна таблиця застосунків для тренувань в тренажерному залі.

Критерій порівняння	Nike training club	Adidas training	GymUp	Seven	Workout trainer
Підтримка україномовної локалізації	-	-	-	-	-
Аудіоспровід	+	+	-	+	+
Довідник вправ	-	-	+	-	-
Необхідність авторизації	+	+	-	-	+
Наявність розділу з корисними статтями	+	+	+	+	+

З наведеної вище таблиці видно, що не все програмне забезпечення пропонує можливість надання інформації українською мовою, а також надання довіднику вправ. Тому було визначено функціональні вимоги до застосунку, а саме:

- підтримка україномовної локалізації;
- підтримка довідника вправ;
- наявність розділу з корисними статтями;
- без необхідності авторизації;
- наявність рецептів корисних страв;
- можливість відстежувати свої результати;

Тому розробка програмного забезпечення для тренувань в тренажерному залі є доцільною.

1.3 Висновки за розділом 1

Було проаналізовано питання, пов'язані з розробкою програмного забезпечення, що надає інформацію про тренування в тренажерному залі.

Було визначено фізіологічні та психологічні аспекти тренування, типи вправ та тренажерів, які використовуються для різних груп м'язів, а також відповідність між цими вправами та фізичними цілями, які має користувач.

Було проаналізовано програмні засоби у вигляді android-застосунків для надання інформації про тренування в тренажерному залі. Результати показали, що не всі додатки пропонують підтримку україномовної локалізації.

Тому розробка програмного забезпечення у вигляді android-застосунку для тренувань в тренажерному залі є доцільною.

2 МАТЕРІАЛИ І МЕТОДИ

Розробка додатків для Android вже деякий час перебуває на підйомі. Це пов'язано з тим, що Android стала найпопулярнішою операційною системою для смартфонів у світі. З цим новим сплеском попиту багато компаній задаються питанням, яку мову програмування та технології їм слід використовувати для створення передових та ефективних мобільних застосунків.

2.1 Вибір мови програмування

Говорячи про розробку застосунків для Android, багато фахівців одразу ж згадують дві технології, а саме Kotlin і Java, обидві з яких є чудовими та популярними мовами, здебільшого завдяки своїм чудовим функціям і здатності розробляти чудові застосунки.

Kotlin - це мова програмування з відкритим вихідним кодом, розроблена компанією JetBrains у 2010 році. Kotlin можна використовувати в проектах із розроблення застосунків для Android, IOS та комп'ютерних систем. Kotlin забезпечує взаємодію з Java, тому можна використовувати навички та знання Java. Ключові особливості Kotlin:

- сумісність із Java: Kotlin може використовувати наявні навички та знання Java. Це означає, що розробники можуть продовжувати писати додатки для Android, використовуючи свою поточну мову програмування;
- null-safety: Kotlin стійкий до нулів, що запобігає помилкам, які зазвичай виникають через неправильно ініціалізованих параметрів. Важливо переконатися, що типи не використовуються як нульові значення за замовчуванням;
- класи даних: Java розробникам необхідно створити поля і змінні для зберігання даних, конструктори, функції get/set для полів/змінних і низку

інших функцій. Kotlin, з іншого боку, спрощує процес створення класів даних, просто включивши ключове слово `data` у визначенні класу;

- лаконічний синтаксис: відсутність непотрібних лексем і ключових слів робить код більш лаконічним і легким для очей розробника;

- виняткові ситуації: на відміну від програмістів Java, розробникам Kotlin не доводиться проходити через часто гнітючий і трудомісткий процес вилову та оголошення винятків;

- робота з потоками: обидві мови дають змогу створювати кілька фонових потоків для довготривалих процесів, що дає змогу уникнути труднощів для основного потоку під час запуску компонентів, які належать одному за стосунку в Android Studio. У Java це важке завдання, але в Kotlin є просте рішення за допомогою Coroutines.

Java — це об'єктно-орієнтована мова програмування на основі класів, яка розроблена таким чином, щоб мати якомога менше залежностей від реалізації. Він призначений для того, щоб дозволити розробникам програм писати один раз і запускати будь-де (WORA), тобто скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності повторної компіляції [1]. Java була створена компанією Sun Microsystems у 1995 році. Це об'єктно-орієнтований, імперативний, паралельний процес з автоматичним керуванням пам'яттю. Java можна використовувати для проектів для розробки додатків для Android, а також desktop додатків, але це трохи більше, ніж код Kotlin, оскільки він вимагає імпорту класів Java. Він є більш складним.

Ключові особливості Java:

- автоматичне управління пам'яттю: завдяки автоматичному управлінню пам'яттю розробникам не потрібно турбуватися про виділення та звільнення пам'яті у своєму коді, завдяки збирачу сміття;

- підтримка декількох платформ: оскільки Java підтримує кілька платформ, розробники можуть почати писати код на Java, а потім перейти на Kotlin за необхідністю.

Розглянемо особливості мов програмування та порівняємо їх у табл. 2.1.

Таблиця 2.1 - Порівняльна таблиця мов програмування

Параметри	Java	Kotlin
Простота використання	Важко	Легко
Продуктивність	Висока	Висока
Популярність	Висока	Середня
Робота з потоками	Важка робота	Легкий процес
Безпека від NPE	Не має	Присутня
Лаконічність	Поступається	Має перевагу
Швидкість написання коду	Середня	Висока

Як бачимо обидві мови має свої переваги та недоліки, але Kotlin – мова, що швидко розвивається, тому надає розробникам можливість експериментувати з новітніми сучасними парадигмами програмування в розробці для Android, тому мій вибір спадає на Kotlin.

2.2 Вибір архітектури

Розробники завжди вважають за краще розробляти Android-додатки з використанням шаблонів архітектури програмного забезпечення. Архітектурні шаблони надають модульність файлам проекту і гарантують, що весь код покритий модульними тестами. Це полегшує розробникам підтримку програмного забезпечення та розширення функціональності додатка в майбутньому. MVP(Model-View-Presenter), MVVM(Model-View-ViewModel), MVI(Model-View-Intent) – найпопулярніші архітектури в галузі.

Шаблон архітектури MVP забезпечує простий спосіб структурування коду проекту. MVP широко визнаний, оскільки пропонує модульність,

можливість тестування та чистішу і простішу в обслуговуванні кодову базу [2]. Він складається з трьох компонентів:

- Model: рівень зберігання даних. Відповідає за обробку логіки домену(фактичних бізнес-правил) та зв'язок з базою даних і мережевим рівнем;
- View: рівень інтерфейсу користувача. Забезпечує візуалізацію даних, відстежує дії користувача і повідомляє про них presenter;
- Presenter: отримує дані з моделі і застосовує логіку інтерфейсу користувача, щоб визначити, що відображати. Керує станом подання і виконує дії відповідно до повідомлень користувача про вхідні дані з подання.

Діаграма роботи MVP (рис.2.1).

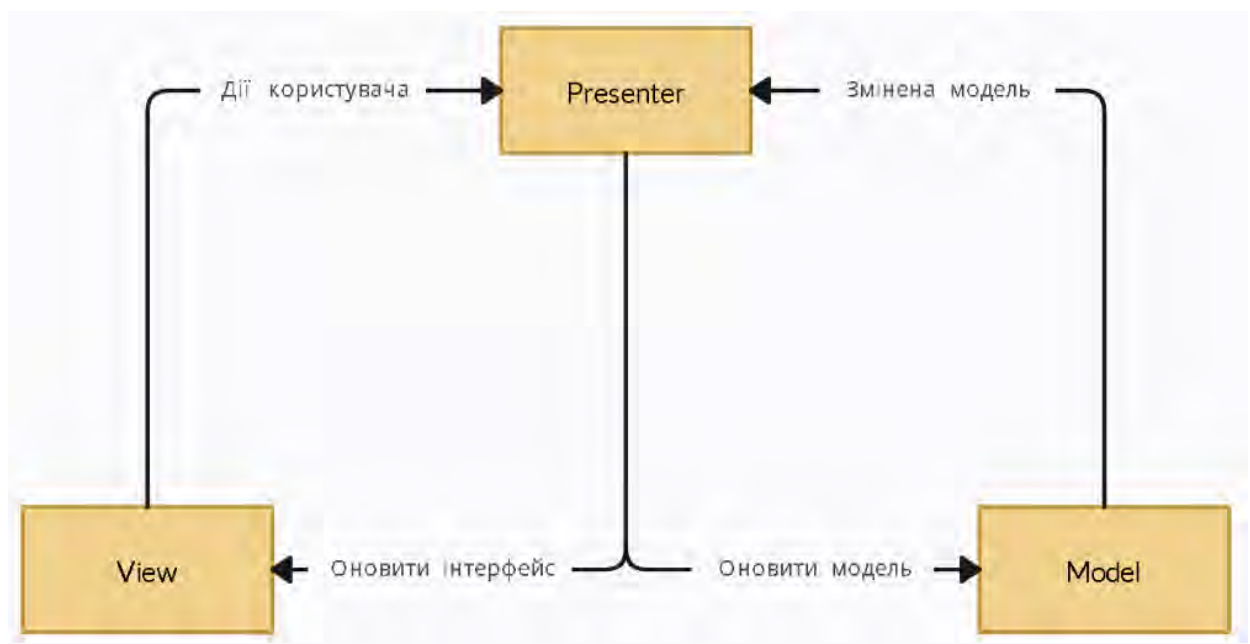


Рисунок 2.1 – Діаграма роботи MVP

Шаблон MVVM схожий на патерн проектування MVP(Model-View-Presenter), де ViewModel виконує роль Presenter у MVP. Однак недоліки шаблону MVP усунуті в MVVM, який пропонує відокремити логіку представлення даних(подання або інтефейс користувача) від основної частини бізнес-логіки додатку [2]. Окремі шари MVVM-коду:

- Model: цей шар відповідає за абстракцію джерел даних. Model та ViewModel працюють разом для отримання та зберігання даних;
- View: метою цього рівня є інформування ViewModel про дії користувача. Цей рівень спостерігає за ViewModel і не містить логіки програми;
- ViewModel: надає потоки даних, пов'язані з View. Крім того, він діє як сполучна ланка між моделлю та представленням.

Діаграма роботи MVVM (рис.2.2).

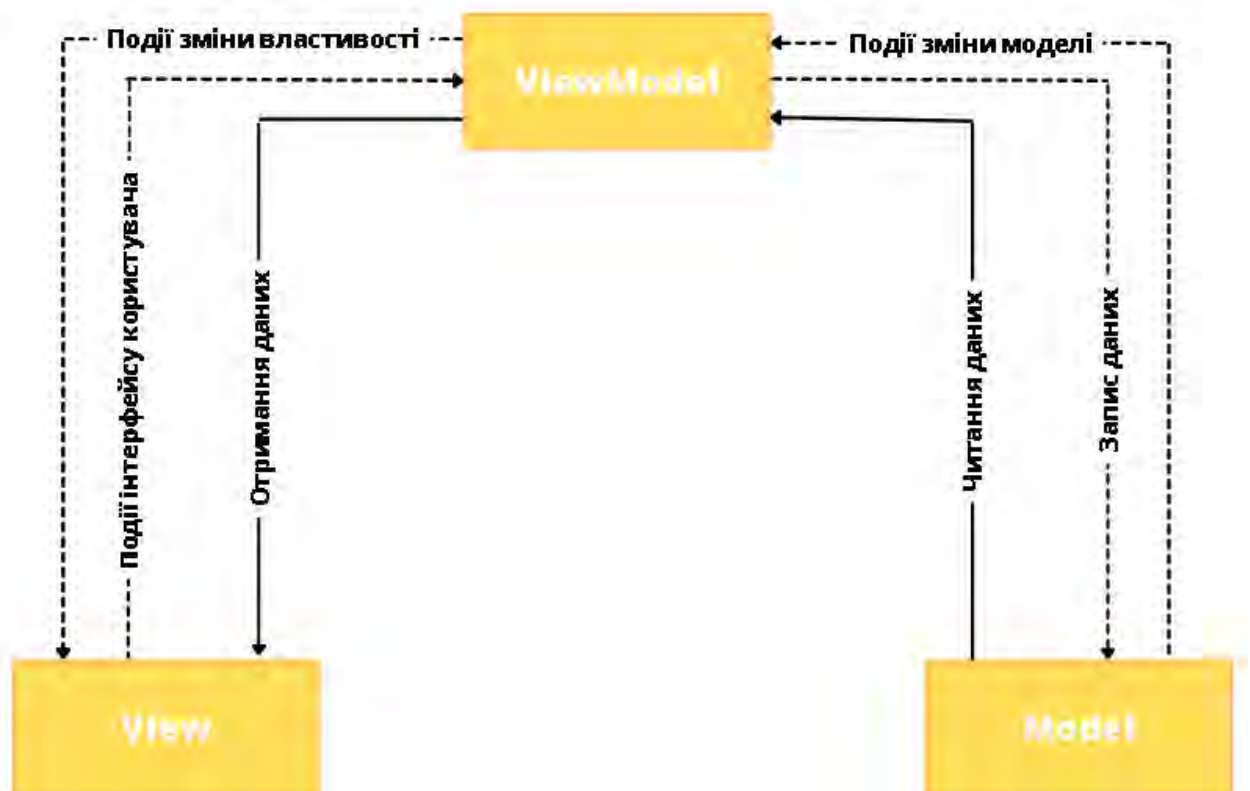


Рисунок 2.2 – Діаграма роботи MVVM

MVI - дуже схожий шаблон на MVVM, але з деякими важливи відмінностями. MVI визначає набір подій, які потрібно обробити, а ViewModel і View відображають потік цих подій по мірі того, як вони відбуваються. Основна відмінність, однак, полягає у зворотному зв'язку з представленням: в той час як в MVVM для кожного фрагмента даних

зазвичай існує окремий публікатор, в MVI публікується єдиний об'єкт, який визначає весь стан представлення.

Діаграма роботи MVI (рис.2.3).

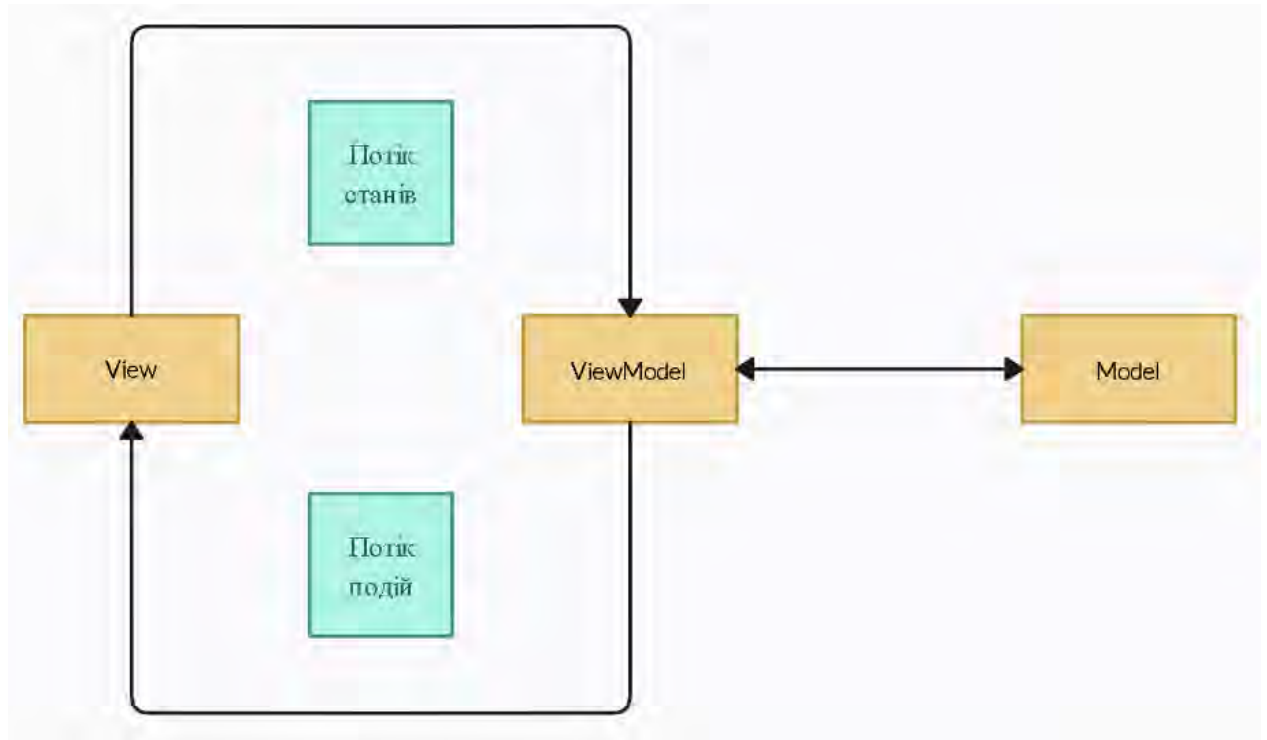


Рисунок 2.3 – Діаграма роботи MVI

Як бачимо, кожен з шаблонів мають схожі риси та відмінності, для свого застосунку я вибрав шаблон MVVM, тому що з цим шаблоном легше оброблювати дії користувача.

2.3 Вибір системи впровадження залежностей

Ін'єкція залежностей – процес надання зовнішніх залежностей програмним компонентам. Це специфічна форма “інверсії контролю”, що застосовується до управління залежностями. Повністю дотримуючись принципу виключної відповідальності, організація делегує турботу про створення необхідних залежностей зовнішньому спеціалізованому

загальному механізму. Використання ін'єкції залежностей може закласти основу для гарної архітектури застосунку. Має наступні переваги:

- код можна використовувати повторно;
- легкість ре факторингу;
- легкість тестування.

Існує три основні фреймворку для використання ін'єкцій залежностей в Android, а саме Dagger, Hilt, Koin.

Dagger – це фреймворк залежностей з відкритим кодом від Google. Він розроблений, щоб допомогти розробникам писати більш зручний и тестований код, відокремлюючи бізнес-логіку від коду фреймворку. Dagger використовує обробку анотацій для генерації коду і добре оптимізований для додатків Android. Однією з головних переваг є те, що він є дуже потужним інструментом. Однак, Dagger не є простим варіантом для початківців, і може бути складним у налаштуванні та використанні. Dagger 2 заснований на Java Specification Request (JSR) 330. Він використовує генерацію коду та базується на анотаціях. Згенерований код дуже легко читати та налагоджувати [8].

Dagger використовує такі анотації:

- `@Module` і `@Provides`: визначають класи та методи, які забезпечують залежності [8];
- `@Inject`: запити залежностей. Може використовуватися в конструкторі, полі або методі [8];
- `@Component`: увімкнути вибрані модулі та використовувати їх для впровадження залежностей [8];
- Dagger використовує згенерований код для доступу до полів, а не відображення. Тому не дозволяється використовувати приватні поля для введення поля [8].

Hilt – це інфраструктура розгортання залежностей, розроблена компанією Google на основі Dagger, але зі спрощеним процесом встановлення та налаштування. Hilt надає набір компонентів та анотацій, які спрощують налаштування Dagger, а також набір засобів для структурування

коду. Hilt розроблений для початківців і може використовуватися як для Android, так і для веб-додатків. Недоліками є те, що він є потенційно більш ресурсоємним і менш конфігурованим, ніж Dagger. Для роботи з Hilt необхідно знати які анотації і для чого потрібно використовувати:

- `@HiltAndroidApp` – розмістіть над класом програми, щоб розпочати магію. Він генерує всі згадані раніше компоненти та навіть деякі модулі, які можуть вам знадобитися. Він потрібен, щоб усе працювало [7];

- `@AndroidEntryPoint` – цей розміщується над діями та фрагментами. Після цього згенеровані модулі конструктора будуть заповнені вашими екранами. Це потрібно для кожного екрана, на який ви хочете вставити. Якщо ви використовуєте ViewModels у своїй програмі, ця анотація гарантує, що ViewModel можна впровадити [7];

- `@Inject` – розміщується над залежністю, яку потрібно вставити, або поруч із нею [7];

- `@Module` - позначає клас як модуль, який забезпечує деякі залежності. Кожен модуль також має бути анотований `@InstallIn`, щоб визначити його область [7];

- `@InstallIn` – приймає один параметр, який є компонентом для встановлення модуля. Компоненти мають різні області дії, тому переконайтеся, що ваш модуль включено лише там, де це необхідно [7];

- `@Provides` – розмістіть над функцією в модулі, яка повертає примірник залежності, яку ви хочете кудись додати [7];

- `@Binds` – також використовується в модулях, але функція може бути абстрактною або функцією інтерфейсу. Замість створення та повернення екземпляра функція має повертати тип залежності (зазвичай інтерфейс) і приймати конкретну реалізацію як параметр [7];

- `@HiltViewModel` – розмістіть над ViewModels, щоб заповнити ViewModelComponent, що дозволяє вводити моделі перегляду [9];

Koin – середовище розподілу залежностей з відкритим вихідним кодом, створене розробниками додатків для Android Kotlin. Koin розроблено,

щоб бути легким, простим у використанні та оптимізованим для додатків Kotlin. Однією з головних переваг є те, що він добре оптимізований для додатків і є найкращим вибором для розробників, які використовують мову Kotlin. Анотації системи впровадження Koin:

- `@Single` – екземпляр, який створюється лише один раз [9];
- `@Factory` - екземпляр фабрики. Для екземплярів, які створюються щоразу, коли вам потрібен екземпляр [9];
- `@KoinViewModel` – екземпляр Android ViewModel [9];

Щоб запустити систему в файлі програми потрібно запустити `startKoin {}` [9].

Розберемо системи впровадження та порівняємо їх у табл. 2.2.

Таблиця 2.2 – Порівняльна таблиця систем впровадження залежностей

Характеристика	Dagger	Hilt	Koin
Ким створена	Google	Google	Спільнота Kotlin
Тип ін'єкції	Конструктор, поля	ViewModel, область дії і фрагментів	ViewModel, область дії і фрагментів
Швидкість	Не використовує reflection, тому працює швидше і безпечніше	Використовує reflection для генерації кода під час компіляції, робить його швидким та безпечним	Використовує reflection для генерації кода під час виконання, робить його повільним та небезпечним
Легкість освоєння	Важка	Легка	Легка

Для свого додатку я обрав систему впровадження залежностей – Hilt, тому що вона легка для освоєння та працює швидко та безпечно.

3 ОПИС ПРОГРАМИ

3.1 Функціональні вимоги до програми та вимоги до інтерфейсу

Розглянемо функціональні вимоги до програмного забезпечення для тренувань у тренажерному залі. Визначемо основні характеристики та функції, які повинно забезпечувати програмне забезпечення для тренувань, щоб забезпечити ефективні та безпечні тренування для користувачів:

- вибір тренувальної програми: користувачі повинні мати можливість вибрати програму тренування, яка найкращим чином відповідає їхнім потребам і цілям. Застосунок має пропонувати різні варіанти тренувань, наприклад силовий, кардіо та функціональний тренінг;
- відстежування прогресу: застосунок має надавати можливість відстежувати прогрес користувача;
- давати всю необхідну інформацію, для безпечного та ефективного тренування, включно із технікою тренування, правильного харчування, спортивних добавок, загальних порад.

Інтерфейс взаємодії з програмою є ключовим елементом, який забезпечує зручну та ефективну роботу користувачів з програмним забезпеченням. Визначено наступні аспекти вимог до інтерфейсу програми:

- ергономічність та інтуїтивність: Інтерфейс програми повинен бути зрозумілим та легким у використанні. Важливо забезпечити, щоб користувач міг швидко зорієнтуватись у програмі, знайти потрібну інформацію без додаткових зусиль;
- графічний дизайн: Важливо розробити привабливий та залучаючий графічний дизайн інтерфейсу. Кольорова палітра, шрифти та іконки повинні бути добре підібраними, щоб створити приємне візуальне сприйняття;
- навігація та структура: Інтерфейс повинен мати зрозумілу та послідовну структуру, щоб користувачі могли ефективно переходити між різними екранами та розділами програми.

3.2 Розбір UI - екранів

Застосунок має основні три екрани, а саме:

- Workout (екран, який відображає типи тренувань);
- DirectoryFragment (екран, який відображає тип інформації, котру користувач може отримати);
- SettingsFragment(екран, який відображає налаштування).

Та додаткові екрани, а саме:

- Profile(екран, який відображає профіль користувача);
- AboutApp(екран, який відображає інформацію, щодо застосунку);
- Measurements(екран, який відображає заміри тіла користувача);
- Exercise(екран, який відображає інформацію, щодо вправи);
- DirectoryInfo(екран, який відображає список типів інформації);
- DetailedInfo(екран, який відображає інформацію обраного типу);
- PowerWorkout(екран, який відображає інформацію силового тренування).

Наступна діаграма відображає всю навігацію застосунку (рис.3.1).

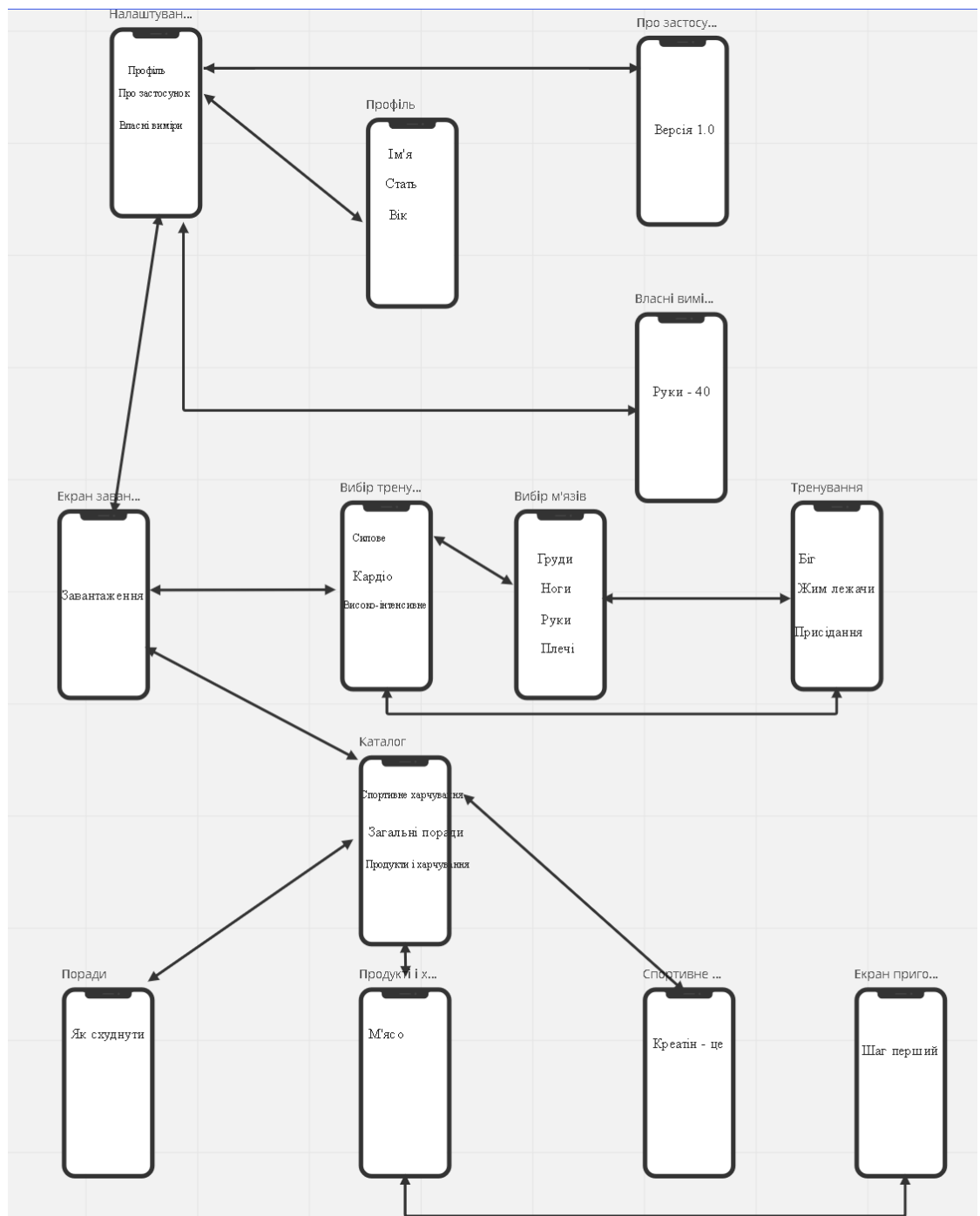


Рисунок 3.1 – Діаграма навігації застосунку

3.3 Файли та методи

Застосунок було розбито на папки, для того, щоб розділити логіку і було легше орієнтуватися в класах, а саме:

- base(знаходяться такі класи які додають додаткову логіку до класів Fragment, ViewModel);

- data(класи, які використовуються для моделювання об'єктів, без додаткової логіки);
- di(класи, які впроваджують залежності в потрібні місця);
- navigation(знаходиться клас, який відповідає за навігацію програми);
- repository(знаходяться класи, які відповідають за бізнес-логіку програми);
- ui(знаходяться класи, які відповідають за відображення екранів і взаємодію користувача з ними);
- usecase(знаходиться клас, який також відповідає за бізнес-логіку додатку);
- util(знаходиться клас, котрий допомагає передати строки в потрібне місце);
- res(знаходяться файли ресурсів застосунку);
- layout(знаходяться файли, що відповідають за графічний інтерфейс застосунку);
- anim(файли, що відповідають за анімацію під час навігації в застосунку).

Розпочнемо з того, що кожен файл-фрагмент має два однакових методи, а саме:

- onCreateView;
- onViewCreated.

onCreateView() рекомендовано лише для роздування макета.

Його можна використовувати для створення компонентів і роздування макета за допомогою методу inflate() об'єкта inflater [18].

onViewCreated() викликається одразу після повернення onCreateView(LayoutInflator, ViewGroup, Bundle), але до відновлення будь-якого збереженого стану в поданні. Це дає підкласам можливість ініціалізувати себе, коли вони знають, що їх ієрархію представлень було повністю створено. Проте ієрархія перегляду фрагмента не приєднана до

його батьківського елемента на даний момент. `onViewCreated()` викликається після `onCreateView()` і гарантує, що кореневий перегляд фрагмента не є нульовим. Тут ми можемо оголосити об'єкти, необхідні для `Context` і в основному використовуються для остаточної ініціалізації (наприклад, зміни елементів інтерфейсу) [18].

Навігація відбувається між пунктами призначення вашого додатка, тобто будь-де у вашому додатку, до якого користувачі можуть перейти. Ці пункти призначення пов'язані за допомогою дій. Навігаційний граф – це файл ресурсів, який містить усі ваші пункти призначення та дії. Графік представляє всі шляхи навігації вашої програми. Кожне місце призначення представлено мініатюрою попереднього перегляду, а дії підключення представлені стрілками, які показують, як користувачі можуть переходити від одного пункту призначення до іншого [18]. Майже до кожного фрагменту був створений файл `Screen`, який відповідає за навігацію до наступного екрану. Кожен клас `Screen`, реалізовує інтерфейс `NavigationComponentScreen`, який реалізовує інший інтерфейс `Screen`. Також є клас `Command`, він має в собі клас `Navigate`, котрий відповідає за команду навігації до наступного класу. Інтерфейс `Router`, відповідає за отримання поточного графу методом `attachNavController`, та функції `execute`, яка віддає необхідну команду для навігації. Було створено граф, `start_graph.xml`, котрий порівнює наші елементи та віддає інформацію до котрого класу необхідно зробити навігацію.

До екранів, які містять в собі якісь списки, було створено класи – `Adapter`, які ініціалізують роботу зі списками. Адаптер містить в собі такі методи і елементи:

- `nBindViewHolder`: пов'язує дані з графічними елементами;
- `onCreateViewHolder`: ініціалізовує графічні елементи списку;
- клас `PowerViewHolder`: який в методі `bind` передає отримані дані в наші `View` – елементи та відображає їх;

– клас `DiffCallback`: має методи `areItemsTheSame` та `areContentsTheSame`, які потрібні для перевірки елементів, що прокручуються в списку на унікальність, за для більш плавного прокручування.

До кожного класу – фрагменту, також було створено класи `ViewModel`, які отримують дані з бізнес логіки або виконують цю логіку та передають їх до фрагментів.

Також, до кожного класу – фрагменту, було створено файли `fragment.xml`, в яких створюється графічний дизайн екрану.

Для роботи з клієнт – серверною частиною застосунок звертається до API [21], та було створено такі класи та інтерфейси:

– інтерфейс `GetNutritionApi`: який надає можливість отримати даних з серверу [21];

– клас `GetNutritionApiImpl`: який є реалізацію для `GetNutritionApi`;

– `NutritionApi`: інтерфейс запиту до серверу [21];

Для впровадження залежностей було створено такі класи:

– `NetworkModule`: модуль, який впроваджує об'єкт `Retrofit` для `GetNutritionApiImpl`. Об'єкт `Retrofit` створює та виконує запит до серверу [21], який відправляє дані у відповідь;

– `DataModule`: модуль, який впроваджує `Context` до `ViewModel`-класів. `Context` – клас, який надає доступ до базових функцій програми: доступ до ресурсів, файлової системи, виклик активності, це потрібно тому що класи `ViewModel` не мають контексту, тому його необхідно впроваджувати;

– `NutritionUserBind`: модуль, який впроваджує наш репозиторій до `ViewModel` класів, для того щоб користувач зміг зробити запит до серверу та отримати необхідні дані.

Всі об'єкти створені завдяки Data-класам - клас даних `Kotlin` використовується лише для зберігання даних і не надає жодних інших функцій, окрім зберігання даних. Існують наступні умови для визначення

класу Kotlin як класу даних: основний конструктор повинен мати принаймні один параметр [17].

В файлі `SplashFragment.kt` в методі `onViewCreated` реалізовано завантаження гіф та виклик функцій з файлу `SplashViewModel.kt`, а саме:

- `loadData`: імітація завантаження даних;
- `navigateToMain`: за допомогою `SplashScreen.kt`, який показує в який граф потрібно перейти, відбувається навігація до `MainFragment`.

В файлі `MainFragment.kt`, в методі `onViewCreated` оброблюється натискання на кнопок, за допомогою батьківського методу нашого View-елементу – `setOnClickListener`. Оброблюємо три варіанти натискання:

- користувач натискає на елемент силове тренування: викликаємо функцію з `MainFragmentViewModel` – `createPowerWorkout`”, який створює об’єкт силового тренування типу `TypeWorkout`, який містить в собі `List<Category>`. `List<Category>` - список об’єктів типу `Category`, який містять в собі такі параметри: `keyCategory` типу `String`, `image` типу `Drawable` та `listExercises` типу `List<Exercises>`. `List<Exercises>` - список об’єктів типу `Exercises`, які містять в собі такі параметри: `name` типу строка, `description` типу строка та `image` типу `Drawable`. Викликаємо функцію з `MainFragmentViewModel.kt` – `navigateToGroupMuscles`, який отримує параметр типу `TypeWorkout`, з попереднього методу, та передає його в метод `navigateToGroupMuscles`”, який відповідає за навігацію до екрану списку груп м’язів за допомогою файлу `MainFragmentScreen.kt`. Проходить перевірка який параметр прийшов, якщо параметр з об’єкту силового тренування, то викликаємо навігацію з файлу нашого графу `navigator.navigate(MainFragmentDirections.actionMainFragmentToPowerWorkoutFragment(type!!))`;

- користувач натискає елемент на елемент кардіо тренування: викликаємо функцію з `MainFragmentViewModel` – `createCardioWorkout`”, який створює об’єкт кардіо тренування типу `TypeWorkout`, та передає його

параметром в метод `navigateToExercise`», який відповідає за навігацію до екрану зі списком вправ, за допомогою `MainFragmentScreen`;

- користувач натискає елемент на елемент високо інтенсивне тренування: викликаємо функцію з `MainFragmentViewModel` – `HIITWorkout`», який створює об'єкт високо-інтенсивного тренування типу `TypeWorkout`, та передає його `keyCategory` в метод `navigateToExercise`», який відповідає за навігацію до екрану зі списком вправ, за допомогою `MainFragmentScreen`.

Файл `PowerWorkoutFragment.kt`, отримує з попереднього класу об'єкт силового тренування типу `TypeWorkout` та передає список `Category` до `PowerWorkoutAdapter.kt`, який відображає список груп м'язів.

Файли `PowerWorkoutViewModel` та `PowerWorkoutScreen` виконують лише роль навігації для класу `PowerWorkoutFragment.kt`.

Файл `ExerciseFragment.kt`, на початку перевіряє аргумент який було передано при навігації та виконує створення необхідного тренування за допомогою `ExerciseViewModel.kt`. Наприклад при навігації було передано строку зі значенням `Груди`». Ми викликаємо із `ExerciseViewModel.kt` метод `createChesttraining()`, який створює тренування для грудей та передає цей список до `ExerciseAdapter.kt`, який відображає ці вправи. Така ж логіка дотримується до інших груп м'язів, кардіо та високо-інтенсивного тренування.

Файл `SettingsFragment.kt`, має метод `onClickListeners`, в котрому оброблюються всі натискання на кнопки.

Якщо натискається на елемент профіль – відбувається навігація до екрану `Профіль`», за допомогою файлів `SettingsScreen.kt` та `SettingsViewModel.kt`.

Файл `ProfileFragment.kt` має метод `setUi`, який виконує логіку з заповненням даних, відображенням та їх зберігання. Зберігання відбувається за допомогою об'єкту `SharedPreferences`. Один із найцікавіших варіантів зберігання даних, які Android надає своїм користувачам, — це «Спільні

налаштування». Спільні параметри — це спосіб, за допомогою якого можна зберігати та отримувати невеликі обсяги примітивних даних як пари ключ/значення у файлі на сховищі пристрою, наприклад String, int, float, Boolean, які складають ваші параметри у файлі XML у програмі на сховище пристрою. Спільні налаштування можна розглядати як словник або пару ключ/значення [19].

Файл AboutFragment.kt, відображає інформацію з файлу ресурсів строк.

Файл MeasurementsFragment.kt, як і ProfileFragment.kt, виконує таку ж саму логіку.

Файл DirectoryFragment.kt, відображає три елементи з корисною інформацією. В методі onCreateView відбувається обробка натискання на елемент, а вже файли DirectoryScreen та DirectoryViewModel роблять навігацію до обраного елементу.

Файл DirectoryInfoFragment.kt, має два адаптери:

- DirectoryInfoAdapter: використовується для відображення списку спортивних добавок або загальних порад;
- NutritionAdapter: використовується для відображення списку страв, та обробки на натискання якогось конкретного елементу для навігації на наступний екран з детальною інформацією.

В методі onCreateView(), перевіряється який параметр був переданий, та після чого відбувається створення елементів. Наприклад було передано строку «Загальні поради», викликається метод з DirectoryInfoViewModel – createGeneralTipsModel(), який створює список об'єктів Directory, який містить в собі name типу String та description типу String. Така ж логіка виконується, якщо параметр дорівнює «Спортивне харчування». Якщо параметр дорівнює «Продукти і харчування», викликаємо метод «getNutrition()» із DirectoryInfoViewModel. В цьому методі ми створюємо корутину, об'єкт в котрому ми виконуємо роботу асинхронно на іншому потоці, за для того, щоб користувач під час завантаження даних мог

далі працювати з за стосунком. У об'єкту типу `GetNutritionUseCase`, викликаємо `getNutrition()`. Який викликає метод `getNutrition()` з репозиторію `GetNutritionApi`. Після чого отримані дані передаються в об'єкт `LiveData`, об'єкт на котрий можливо підписатися та отримати необхідні дані в фрагменті, після чого ці дані передаються до адаптеру за допомогою `submitList()`, та відображаються на екрані.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для безпечного та ефективного тренування в тренажерному залі. Програмне забезпечення виконує наступні функції:

- дає вибір тренувальної програми: користувачі повинні мати можливість вибрати програму тренування, яка найкращим чином відповідає їхнім потребам і цілям. Застосунок має пропонувати різні варіанти тренувань, наприклад силовий, кардіо та функціональний тренінг;
- дає можливість відстежування прогресу: застосунок має надавати можливість відстежувати прогрес користувача;
- дає інформацію про найпопулярніші спортивні добавки, які допомагають досягнути мети;
- пропонує список страв, які підходять для виконання поставленої мети;
- дає відповіді на найпопулярніші питання в сфері фітнесу;
- пропонує ергономічний та інтуїтивно зрозумілий інтерфейс.

Для того, щоб запустити програмне забезпечення необхідне наступне апаратне забезпечення:

- пристрій має бути на операційній системі Android;
- мінімальна версія операційної системи Android має бути 9;
- максимальна версія операційної системи Android має бути 13;
- 36 MB вільного місця на пристрої.

4.2 Вхідні та вихідні дані

Вхідними даними мобільного додатку для отримання інформації для безпечного та ефективного тренування є дані, введенні користувачем та

кнопки, які користувач натискає. Вихідними даними є саме інформація про вправи, техніку їх виконання.

4.3 Тестування та результати роботи за стосунку

Для того, щоб запустити застосунок, необхідно на пристрій завантажити файл з розширенням .apk.

Після запуску за стосунку перше, що бачить користувач – екран завантаження (рис 4.1).

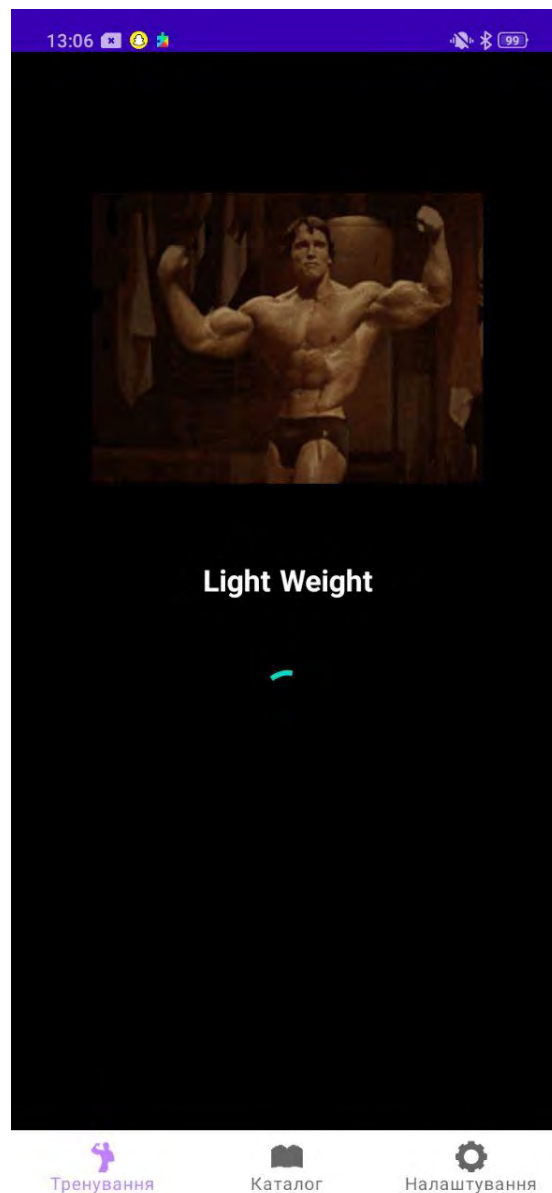


Рисунок 4.1 – Вигляд екрану завантаження

Після цього користувач потрапляє на головний екран – екран тренування (рис 4.2).



Рисунок 4.2 – Головний екран застосунку

Головний екран містить в собі:

- силове тренування;

- кардіо або високо-інтенсивне;
- меню каталог;
- меню налаштування.

Силове тренування призначений для відображення списку груп м'язів(рис 4.3).



Рисунок 4.3 – Екран силового тренування

Наступним кроком користувач обирає тренування на конкретну групу м'язів, наприклад: груди (рис.4.4).



Рисунок 4.4 – Екран зі списком вправ

Якщо користувач все ж таки обрав кардіо тренування, він одразу потрапить на екран зі списком вправ (рис 4.4).

Така ж ситуація буде, якщо користувач обере високо-інтенсивне тренування.

Наступне, що може зробити користувач обрати в меню пункт «Каталог», який відповідає за відображення списку тем з корисною інформацією (рис 4.5).



Рисунок 4.5 – Каталог зі списком корисної інформації

Каталог містить в собі такі пункти:

- спортивне харчування;
- загальні поради;
- продукти і харчування;

Екран «Спортивне харчування» містить в собі список добавок, та загальну інформацію про неї (рис 4.6).

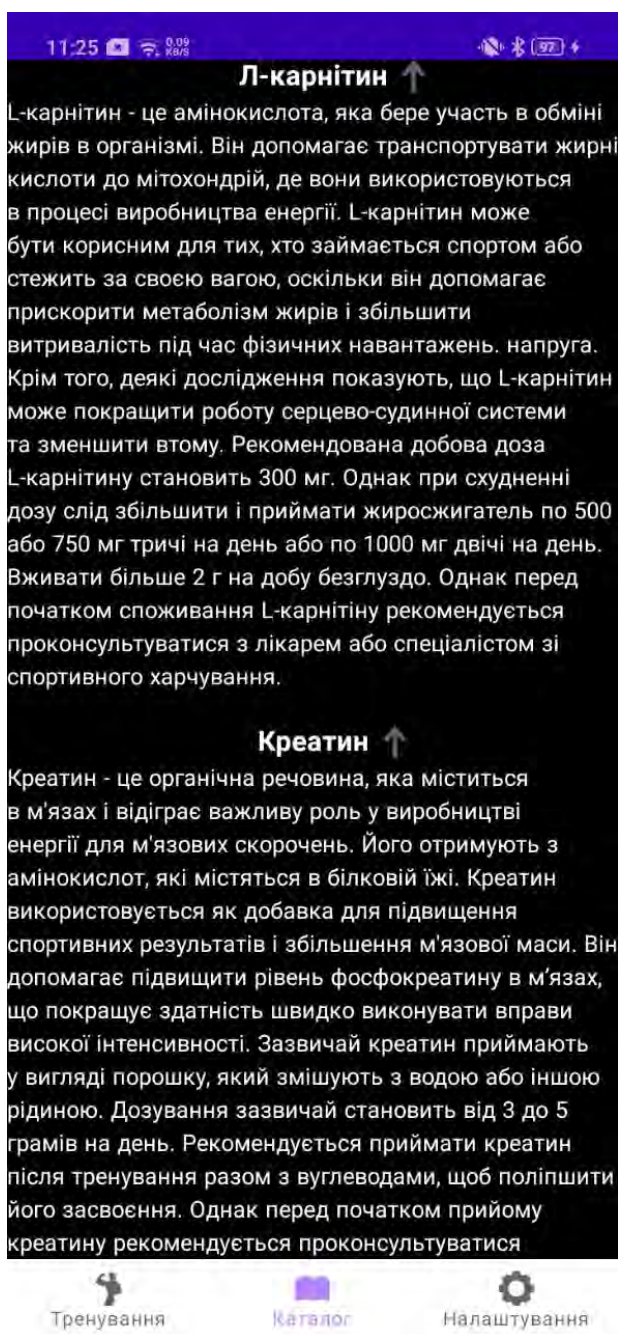


Рисунок 4.6 – Екран спортивного харчування

Екран «Загальні поради» містить в собі список порад, які допоможуть досягнути мети (рис 4.7).



Рисунок 4.7 – Екран загальних порад

Екран «Продукти і харчування» містить в собі список страв, які також допоможуть досягнути мети (рис 4.8).

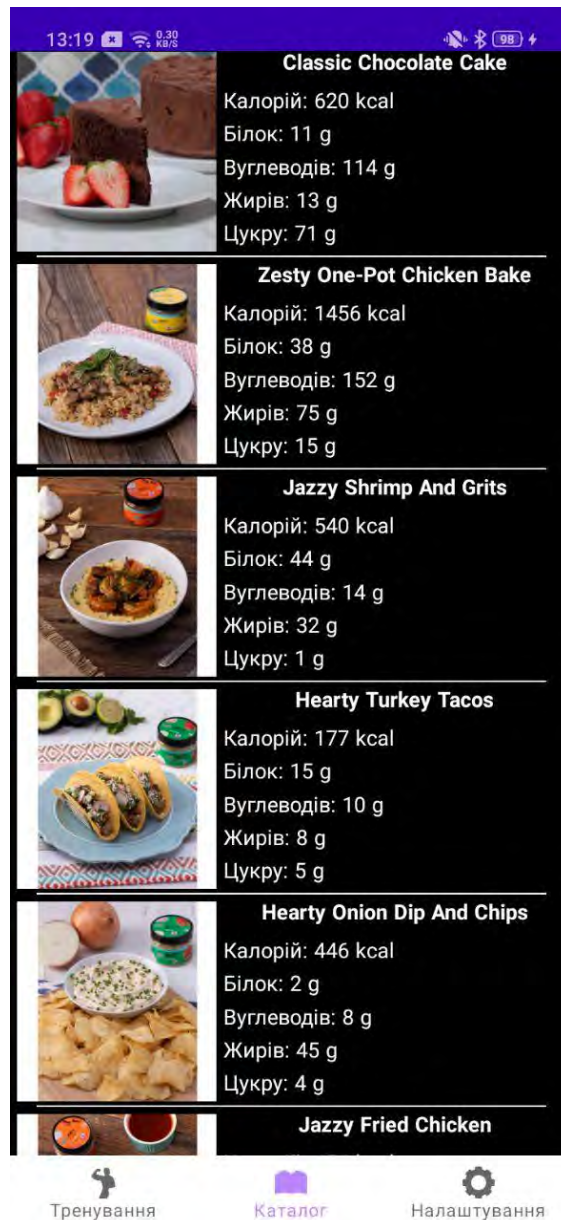


Рисунок 4.8 – Екран продуктів і харчування

Користувач може обрати будь-яку страву, після чого він потрапить на екран, який дасть інформацію, як приготувати цю страву (рис 4.9).

В API, яке використовується для отримання списку страв [21], не має можливості отримати інформацію українською мовою, але воно дає всю необхідну інформацію стосовно страв.



Рисунок 4.9 – Екран детальної інформації про обрану страву

Наступне, що може зробити користувач обрати пункт меню «Налаштування», екран, який містить в собі список налаштувань, профіль, про застосунок, виміри власного тіла (рис 4.10).

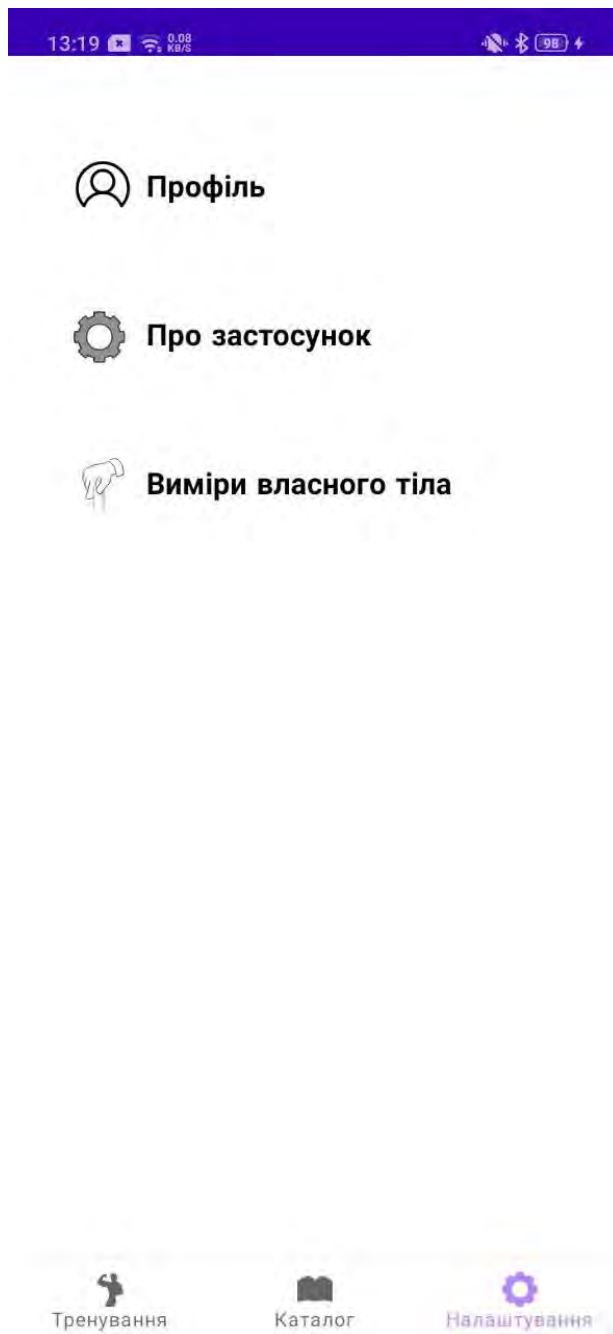


Рисунок 4.10 – Налаштування

Якщо, користувач обере пункт «Профіль», потрапить на екран, де можна додати персональну інформацію (рис. 4.11).



Рисунок 4.11 – Профіль

Якщо, користувач обере пункт «Про застосунок», побачить невелику інформацію про застосунок (рис 4.12).

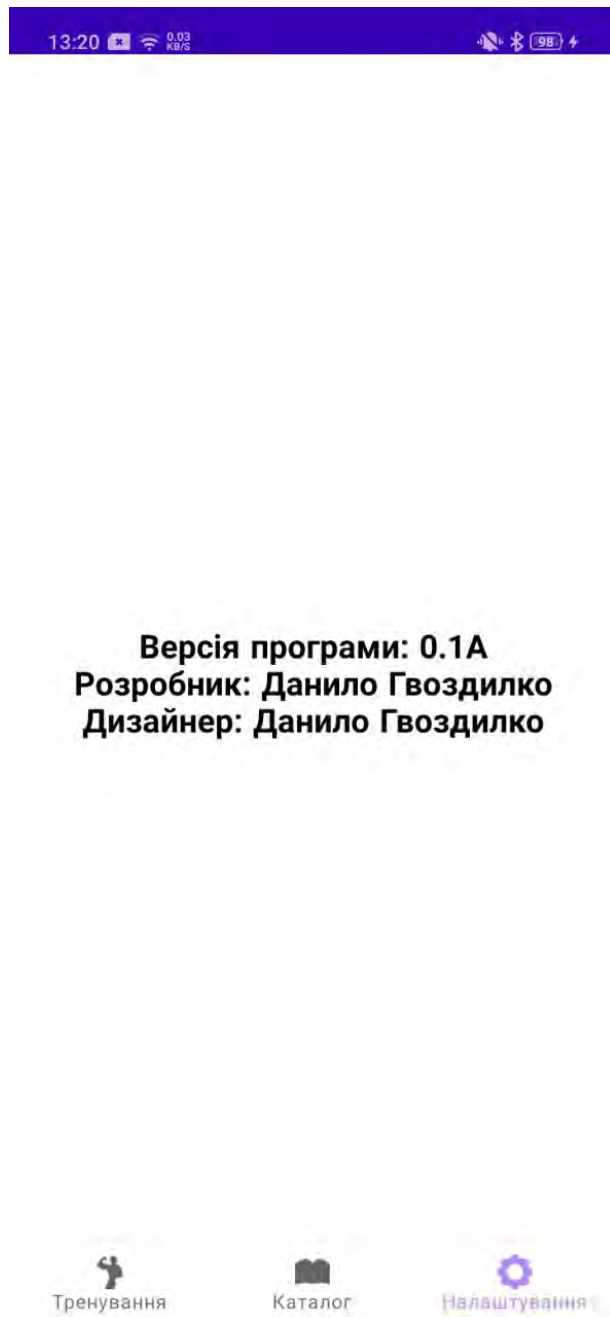


Рисунок 4.12 – Про застосунок

Та останнє, що може зробити користувач обрати пункт «Виміри власного тіла», де користувач може додати свої виміри та у майбутньому порівнювати та бачити різницю (рис 4.13).

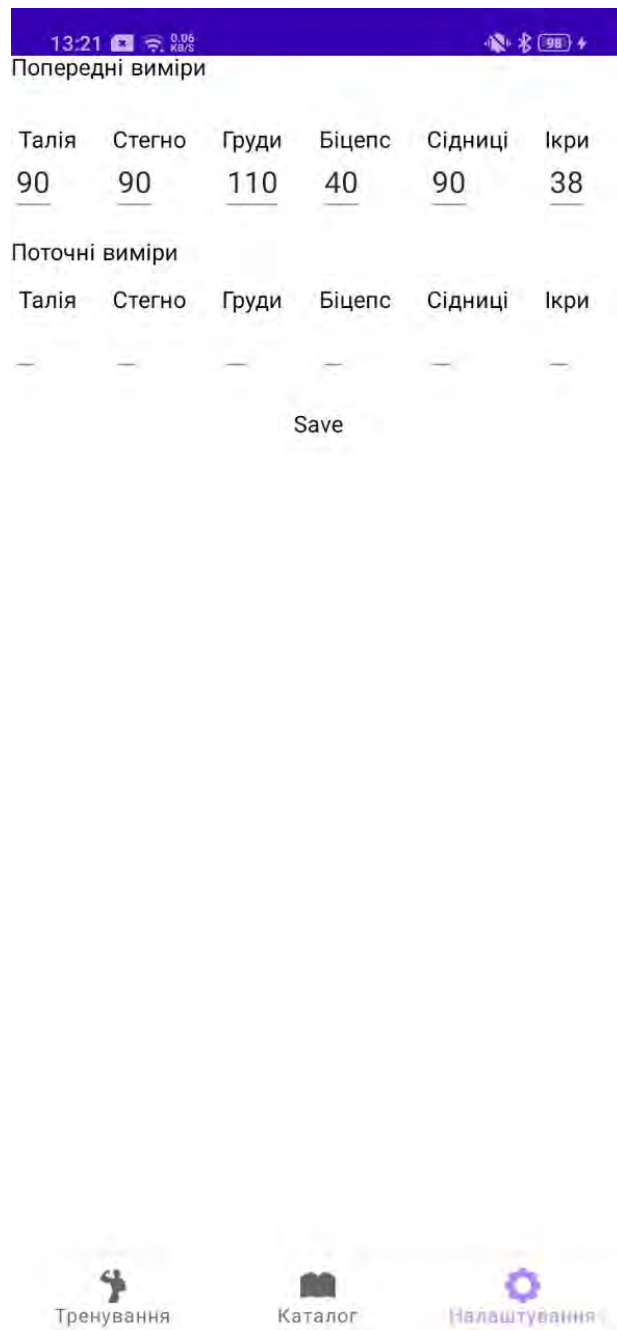


Рисунок 4.13 – Виміри власного тіла

ВИСНОВОК

З метою надання інформації для безпечного та ефективного тренування в тренажерному залі розробка мобільного застосунку є необхідною.

Було проаналізовано існуючі застосунки для тренувань, мови програмувань для написання проекту, архітектури та різні варіанти впровадження залежностей, де було обрано мову – Kotlin, архітектуру – MVVM, систему впровадження залежностей – Hilt.

Було визначено функціональні вимоги та вимоги до інтерфейсу програми, вимоги до складу та параметрів технічного засобу, де було обрано мобільний пристрій з ОС - Android. Застосунок відповідає меті та темі розробки та всім перерахованим вимогам, які були визначені.

Програмний продукт був протестований без помилок. Так як це перша версія застосунку, він може оновлюватися та отримувати нові можливості, наприклад:

- розширення програм тренувань;
- розширення каталогу с корисною інформацією;
- додання відеоспроводження під час виконання вправи;
- покращення дизайну;

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Introduction to Java - GeeksforGeeks. GeeksforGeeks [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-java/>
2. Mishra R. Difference Between MVP and MVVM Architecture Pattern in Android - GeeksforGeeks. GeeksforGeeks [Electronic resource]. / R. Mishra. – Access mode: <https://www.geeksforgeeks.org/difference-between-mvp-and-mvvm-architecture-pattern-in-android/>.
3. Uvarova A. MVVM на Android с компонентами архитектуры + библиотека Koin. Medium [Электронный ресурс]. / A. Uvarova. - Режим доступа: <https://medium.com/nuances-of-programming/mvvm-на-android-с-компонентами-архитектуры-библиотека-koin-e2e77b77950e/>.
4. Sady M. All You Need for MVI is Kotlin. How to Reduce Without a Reducer?. Medium [Electronic resource]. / M. Sady. – Access mode: <https://betterprogramming.pub/all-you-need-for-mvi-is-kotlin-how-to-reduce-without-reducer-5e986856610f/>.
5. Лекция 5 по архитектуре андроид приложений. Паттерн MVP - Fandroid.info. Fandroid.info Уроки створення мобільних додатків [Электронный ресурс]. - Режим доступа: <https://www.fandroid.info/lecture-5-on-the-architecture-of-the-android-application-mvp-pattern/>.
6. Raj A. Dependency Injection: Dagger-Hilt vs Koin. Medium [Electronic resource]. / A. Raj. – Access mode: <https://medium.com/gradeup/dependency-injection-dagger-hilt-vs-koin-ab2f7f85e6c6/>.
7. Android Handbook | Dependency injection / Hilt. Infinum | Building digital products [Electronic resource]. – Access mode: <https://infinum.com/handbook/android/dependency-injection/hilt/>
8. Vogel L. Using Dagger 2 for dependency injection in Android - Tutorial. Eclipse, Android and Java training and support [Electronic resource]. / L. Vogel. – Access mode: <https://www.vogella.com/tutorials/Dagger/article.html/>.

9. Definitions | Koin. Koin - The pragmatic Kotlin Injection Framework - developed by Kotzilla and its open-source contributors [Electronic resource]. Access mode: <https://insert-koin.io/docs/reference/koin-annotations/definitions/>.
10. Dependency injection in Android | Android Developers. Android Developers [Electronic resource]. Access mode: <https://developer.android.com/training/dependency-injection/>.
11. Which is better, Kotlin vs. java? Let's settle the debate. DAC.digital [Electronic resource]. Access mode: <https://dac.digital/kotlin-vs-java/#:~:text=Finally,%20Kotlin%20has%20better%20support,Kotlin%20project%20way%20more%20efficient/>.
12. Vaguez L. Kotlin vs. Java for Android development - LogRocket Blog. LogRocket Blog [Electronic resource]. / L. Vaguez. – Access mode: <https://blog.logrocket.com/kotlin-vs-java-android-development>.
13. Jhonson E. Kotlin vs Java: Which is the Best Choice for Android App Development?. Medium [Electronic resource]. / E. Jhonson. – Access mode: <https://medium.com/javarevisited/kotlin-vs-java-which-is-the-best-choice-for-android-app-development-7c9fc782d2c9/>.
14. What is Java? | IBM. IBM - Deutschland | IBM [Electronic resource]. Access mode: <https://www.ibm.com/topics/java/>.
15. Biet N. Introduction to Kotlin - GeeksforGeeks. GeeksforGeeks. / N. Biet [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-kotlin/>.
16. Kotlin - Data Classes. Online Courses and eBooks Library [Electronic resource]. Access mode: https://www.tutorialspoint.com/kotlin/kotlin_data_classes.htm#:~:text=A%20Kotlin%20Data%20Class%20is,have%20at%20least%20one%20parameter/.
17. Get started with the Navigation component | Android Developers. Android Developers [Electronic resource]. Access mode: <https://developer.android.com/guide/navigation/navigation-getting-started/>.

18. Qadir A. Deep dive into Fragments when to use onCreate(), onCreateView and onViewCreated(). Medium [Electronic resource]. / A. Qadir. – Access mode: <https://medium.com/@abdulqadirtr/deep-dive-into-fragments-when-to-use-oncreate-oncreateview-and-onviewcreated-33faf6454955/>.

19. Shared Preferences in Android with Example - GeeksforGeeks. GeeksforGeeks [Electronic resource]. Access mode: <https://www.geeksforgeeks.org/shared-preferences-in-android-with-examples/>.

20. Dojo A. Tasty API Documentation (apidojo) | RapidAPI. Rapid - The Next Generation API Hub [Electronic resource]. / A. Dojo. – Access mode: <https://rapidapi.com/apidojo/api/tasty/>.

ДОДАТОК А
Технічне завдання

Вступ

Мобільний застосунок про тренування в тренажерному залі може використовуватися для отримання інформації для ефективного та безпечного тренування.

A.1 Підстава для розробки

Розробку виконано на основі завдання на дипломну кваліфікаційну роботу на тему «Розробка мобільного застосунку для тренувань в тренажерному залі».

A.2 Призначення розробки

Мобільний застосунок прагне надати інформацію для ефективного та безпечного тренування.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Розглянемо функціональні вимоги до програмного забезпечення для тренувань у тренажерному залі. Визначемо основні характеристики та функції, які повинно забезпечувати програмне забезпечення для тренувань, щоб забезпечити ефективні та безпечні тренування для користувачів.

- вибір тренувальної програми: користувачі повинні мати можливість вибрати програму тренування, яка найкращим чином відповідає їхнім потребам і цілям. Застосунок має пропонувати різні варіанти тренувань, наприклад силовий, кардіо та функціональний тренінг;

- відстежування прогресу: застосунок має надавати можливість відстежувати прогрес користувача;

- давати всю необхідну інформацію, для безпечного та ефективного тренування, включно із технікою тренування, правильного харчування, спортивних добавок, загальних порад;

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс взаємодії з програмою є ключовим елементом, який забезпечує зручну та ефективну роботу користувачів з програмним забезпеченням. Визначено наступні аспекти вимог до інтерфейсу програми:

- ергономічність та інтуїтивність: Інтерфейс програми повинен бути зрозумілим та легким у використанні. Важливо забезпечити, щоб користувач міг швидко зорієнтуватись у програмі, знайти потрібну інформацію без додаткових зусиль;
- графічний дизайн: Важливо розробити привабливий та залучаючий графічний дизайн інтерфейсу. Кольорова палітра, шрифти та іконки повинні бути добре підібраними, щоб створити приємне візуальне сприйняття.
- навігація та структура: Інтерфейс повинен мати зрозумілу та послідовну структуру, щоб користувачі могли ефективно переходити між різними екранами та розділами програми;

А.3.3 Вимоги до надійності

Мобільний застосунок повинен забезпечувати роботу у виняткових ситуаціях.

А.3.4 Умови експлуатації

Для роботи та надання інформації для ефективного та безпечного тренування, необхідне відповідне апаратне забезпечення з відповідними параметрами.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для того, щоб програмне забезпечення працювало у вигляді мобільного додатку для надання інформації для безпечного та ефективного тренування, необхідні такі параметри:

- пристрій має бути на операційній системі Android;
- мінімальна версія операційної системи Android має бути 9;
- максимальна версія операційної системи Android має бути 13;
- 36 MB вільного місця на пристрої;

А.4 Вхідні дані до роботи

Вхідними даними мобільного додатку для отримання інформації для безпечного та ефективного тренування є дані, введенні користувачем та кнопки, які користувач натискає. Вихідними даними є саме інформація про вправи, техніку їх виконання.

ДОДАТОК Б
Код програми

```

//Base
//BaseFragment
import androidx.fragment.app.Fragment
abstract class BaseFragment<VM : BaseViewModel> : Fragment() {
    protected abstract val viewModel: VM
}

//BaseViewModel
abstract class BaseViewModel : ViewModel()

//Data
//Category
import android.graphics.drawable.Drawable
data class Category(
    val keyCategory: String,
    val image:Drawable,
    val listExercises: List<Exercises>
)

//Directory
data class Directory(val name: String, val description: String)

//Exercises
import android.graphics.drawable.Drawable
data class Exercises(val name: String = "", val description: String = "", val image: Drawable?)

//SplashState
data class SplashState(
    val dataLoaded: Boolean = false,
)

//TypeWorkout
@Parcelize
data class TypeWorkout(
    val typesWorkout: @RawValue List<Category>
):Parcelable

//Components
data class Components(val ingredients: List<ComponentsItem>)

//ComponentsItem
import com.google.gson.annotations.SerializedName

```

```
data class ComponentsItem(
    @SerializedName("ingredient") val ingredients: Ingredients
)
```

//Food

```
import com.google.gson.annotations.SerializedName
data class Food(
    @SerializedName("count") val count: Int,
    @SerializedName("results") val foodList: List<FoodItem>
)
```

//FoodItem

```
import android.os.Parcelable
import com.google.gson.annotations.SerializedName
import kotlinx.android.parcel.Parcelize
import kotlinx.android.parcel.RawValue

@Parcelize
data class FoodItem(
    @SerializedName("nutrition") val nutrition: @RawValue Nutrition,
    @SerializedName("instructions") val instruction: @RawValue List<InstructionsItem>,
    @SerializedName("sections") val sections: @RawValue List<SectionsItem>,
    @SerializedName("thumbnail_url") val thumbnailUrl: String,
    @SerializedName("name") val name: String,
    @SerializedName("total_time_tier") val totalTime: @RawValue TotalTimeTier?
) : Parcelable
```

//Ingredients

```
import com.google.gson.annotations.SerializedName
data class Ingredients(
    @SerializedName("name") val name: String
)
```

//Instructions

```
data class Instructions(val listInstructions: List<InstructionsItem>)
```

//InstructionsItem

```
import com.google.gson.annotations.SerializedName
data class InstructionsItem(
    @SerializedName("temperature") val temperature: Int?,
    @SerializedName("position") val position: Int?,
    @SerializedName("display_text") val displayText: String?,
)
```

```

        @SerializedName("appliance") val appliance: String?
    )

```

//Nutrition

```

import com.google.gson.annotations.SerializedName
data class Nutrition(
    @SerializedName("carbohydrates")
    val carbohydrates: Int? = 0,
    @SerializedName("protein")
    val protein: Int? = 0,
    @SerializedName("fat")
    val fat: Int? = 0,
    @SerializedName("calories")
    val calories: Int? = 0,
    @SerializedName("sugar")
    val sugar: Int? = 0,
)

```

//Sections

```

data class Sections(val components: List<SectionsItem>)

```

//SectionsItem

```

import com.google.gson.annotations.SerializedName
data class SectionsItem(
    @SerializedName("components") val components: List<ComponentsItem>
)

```

//TotalTimeTier

```

data class TotalTimeTier(@SerializedName("display_tier") val displayTier: String)

```

//di

//NutrinionUserBind

```

import com.viggo.lightweight.repository.GetNutritionApi
import com.viggo.lightweight.repository.GetNutritionApiImpl
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent

```

```

@Module
@InstallIn(SingletonComponent::class)
abstract class NutritionUserBind {
    @Binds

```

```

    abstract fun bindNetwork(impl: GetNutritionApiImpl): GetNutritionApi
}

```

//DataModule

```

import android.content.Context
import com.viggo.lightweight.navigation.NavigationComponentRouter
import com.viggo.lightweight.navigation.Router
import com.viggo.lightweight.util.StringsProvider
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

```

```
@Module
```

```
@InstallIn(SingletonComponent::class)
```

```
class DataModule {
```

```
    @Singleton
```

```
    @Provides
```

```
    fun provideNavigationManager(): Router = NavigationComponentRouter()

```

```
    @Singleton
```

```
    @Provides
```

```
    fun provideContext(@ApplicationContext context: Context): StringsProvider =
        StringsProvider(context)

```

```
}
```

//NetworkModule

```

import com.viggo.lightweight.Constants.BASE_URL
import com.viggo.lightweight.repository.NutritionApi
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

```

```

@Module
@InstallIn(SingletonComponent::class)
class NetworkModule {

    @Provides
    fun makeRetrofit(): NutritionApi {
        val retrofit = createRetrofit()
        val api = retrofit.create(NutritionApi::class.java)
        return api
    }

    @Provides
    fun createRetrofit(): Retrofit {
        return Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .client(createOkHttpClient())
            .build()
    }

    @Provides
    fun createOkHttpClient(): OkHttpClient {
        val logging = HttpLoggingInterceptor()
        logging.setLevel(HttpLoggingInterceptor.Level.BODY)
        val httpClient = OkHttpClient.Builder()
        httpClient.addInterceptor(logging)
        return httpClient.build()
    }
}

```

//Navigation

//Router

```

import androidx.navigation.NavController
import javax.inject.Inject

interface Router {
    fun attachNavController(navController: NavController)
    fun execute(command: Command)
}

sealed class Command {

```

```

data class Navigate(val screen: NavigationComponentScreen) : Command()
object PopUp : Command()
}

```

```

interface Screen<T> {

```

```

    fun execute(navigator: T)
}

```

```

interface NavigationComponentScreen : Screen<NavController>

```

```

class NavigationComponentRouter @Inject constructor() : Router {

```

```

    lateinit var navController: NavController

```

```

    override fun attachNavController(navController: NavController) {
        this.navController = navController
    }

```

```

    override fun execute(command: Command) {
        when (command) {
            is Command.Navigate -> command.screen.execute(navController)
            Command.PopUp -> navController.popBackStack()
        }
    }
}

```

```

}

```

//Repository

//GetNutritionApi

```

import com.viggo.lightweight.data.food.Food

```

```

interface GetNutritionApi {
    suspend fun getNutrition(): Result<Food?>
}

```

//GetNutritionApiImpl

```

class GetNutritionApiImpl @Inject constructor(private val nutritionApi: NutritionApi) :

```

```

    GetNutritionApi {
        override suspend fun getNutrition(): Result<Food?> {
            return suspendCoroutine { continuation ->

```

```

                val response = nutritionApi.listFood(

```

```

        from = Constants.FROM,
        size = Constants.SIZE,
        tags = Constants.TAGS,
        api_key = Constants.api_key
    )
    response.enqueue(object : Callback<Food> {
        override fun onResponse(call: Call<Food>, response: Response<Food>) {
            val food = response.body()
            continuation.resume(Result.success(food))
        }

        override fun onFailure(call: Call<Food>, t: Throwable) {
            continuation.resume(Result.failure(t))
        }
    })
}
}
}

```

```

//NutritionApi
import com.viggo.lightweight.data.food.Food
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Header
import retrofit2.http.Query

interface NutritionApi {
    @GET("list")
    fun listFood(
        @Query("from") from: Int,
        @Query("size") size: Int,
        @Query("tags") tags: String,
        @Header("x-rapidapi-key") api_key: String
    ): Call<Food>
}

```

```

//UseCase
//GetNutritionUseCase
import com.viggo.lightweight.data.food.Food
import com.viggo.lightweight.repository.GetNutritionApi
import javax.inject.Inject

```

```

class GetNutritionUseCase @Inject constructor(private val getNutritionApi: GetNutritionApi) {
    suspend fun getNutrition(): Food? {
        getNutritionApi.getNutrition().fold(onSuccess = { result ->
            return@getNutrition result
        }, onFailure = { error ->
            error.printStackTrace()
        })
        return null
    }
}

```

//Uitl

//StringProvider

import android.content.Context

import com.viggo.lightweight.R

import javax.inject.Inject

```

class StringsProvider @Inject constructor(

```

```
    context: Context
```

```
) {
```

```
    val creatineMeaning = context.getString(R.string.creatine_meaning)
```

```
    val bcaaMeaning = context.getString(R.string.BCAA_meaning)
```

```
    val proteinMeaning = context.getString(R.string.protein_meaning)
```

```
    val l_carnitineMeaning = context.getString(R.string.L_carnitin_meaning)
```

```
    val gainerMeaning = context.getString(R.string.gainer_meaning)
```

```
    val glutamineMeaning = context.getString(R.string.glutamine_meaning)
```

```
    val creatine = context.getString(R.string.creatine)
```

```
    val bcaa = context.getString(R.string.BCAA)
```

```
    val protein = context.getString(R.string.protein)
```

```
    val l_carnitine = context.getString(R.string.L_carnitine)
```

```
    val gainer = context.getString(R.string.gainer)
```

```
    val glutamine = context.getString(R.string.glutamine)
```

```
    val gainMuscle = context.getString(R.string.gain_muscles)
```

```
    val weightLoss = context.getString(R.string.weight_loss)
```

```
    val increaseStamina = context.getString(R.string.increase_stamina)
```

```
    val improvePower = context.getString(R.string.improve_power)
```

```
    val gainMuscleTip = context.getString(R.string.gain_muscles_tip)
```

```
    val weightLossTip = context.getString(R.string.weight_loss_tip)
```

```
    val increaseStaminaTip = context.getString(R.string.increase_stamina_tip)
```

```
    val improvePowerTip = context.getString(R.string.improve_power_tip)
```

```

val legsImg = context.getDrawable(R.drawable.img_legs)
val bicepsImg = context.getDrawable(R.drawable.img_biceps)
val tricepsImg = context.getDrawable(R.drawable.img_triceps)
val shouldersImg = context.getDrawable(R.drawable.img_shoulders)
val backImg = context.getDrawable(R.drawable.img_back)
val chestImg = context.getDrawable(R.drawable.img_chest)
val legs = context.getString(R.string.legs)
val biceps = context.getString(R.string.biceps)
val triceps = context.getString(R.string.triceps)
val shoulders = context.getString(R.string.shoulders)
val back = context.getString(R.string.back)
val chest = context.getString(R.string.chest)
val cardio = context.getString(R.string.cardio)
val hiit = context.getString(R.string.hiit)

val hammer = context.getString(R.string.hammer)
val barbellLiftBiceps = context.getString(R.string.barbell_lift_for_biceps)
val bendingArmsWithDumbbells = context.getString(R.string.bending_arms_with_dumbbells)
val blockPullForTriceps = context.getString(R.string.block_pull_for_triceps)
val frenchTricepsPress = context.getString(R.string.french_triceps_press)
val liftingDumbbellsForearm = context.getString(R.string.Lifting_dumbbells_on_the_forearm)
val extensionTriceps =
    context.getString(R.string.extension_of_arms_on_the_simulator_for_triceps)
val barbellSquats = context.getString(R.string.barbell_squats)
val breedingLegs = context.getString(R.string.breeding_legs_in_the_simulator)
val reductionLegs = context.getString(R.string.reduction_of_legs_in_the_simulator)
val legPress = context.getString(R.string.leg_press_in_the_simulator)
val legExtension = context.getString(R.string.leg_extension_in_the_simulator)
val bendingLegs = context.getString(R.string.bending_the_legs_in_the_simulator)
val benchPress = context.getString(R.string.bench_press)
val dumbbellBenchPressHorizontalBench =
    context.getString(R.string.dumbbell_bench_press_on_a_horizontal_bench)
val breedingDumbbellsInclineBench =
    context.getString(R.string.breeding_dumbbells_on_an_incline_bench)
val inclineDumbbellBenchPress = context.getString(R.string.incline_dumbbell_bench_press)
val pullDownUpperBlock = context.getString(R.string.pulldown_of_the_upper_block_to_the_chest)
val dumbbellPullBelt = context.getString(R.string.dumbbell_pull_to_the_belt)
val deadlift = context.getString(R.string.deadlift)
val hyperextension = context.getString(R.string.hyperextension_on_the_bench)
val tBarRowOnTheBlock = context.getString(R.string.T_bar_row_on_the_block)

```

```

val standingDumbbellPress = context.getString(R.string.standing_dumbbell_press)
val standingBarbellPress = context.getString(R.string.standing_barbell_press)
val liftingDumbbellsToTheSides = context.getString(R.string.lifting_dumbbells_to_the_sides)
val liftingDumbbellsInFrontOfYou = context.getString(R.string.lifting_dumbbells_in_front_of_you)

val hammerExecution = context.getString(R.string.hammer_execution)
val barbellLiftBicepsExecution = context.getString(R.string.barbell_lift_for_biceps_execution)
val bendingArmsWithDumbbellsExecution =
    context.getString(R.string.bending_arms_with_dumbbells_execution)
val blockPullForTricepsExecution = context.getString(R.string.block_pull_for_triceps_execution)
val frenchTricepsPressExecution = context.getString(R.string.french_triceps_press_execution)
val liftingDumbbellsForearmExecution =
    context.getString(R.string.Lifting_dumbbells_on_the_forearm_execution)
val extensionTricepsExecution =
    context.getString(R.string.extension_of_arms_on_the_simulator_for_triceps_execution)
val barbellSquatsExecution = context.getString(R.string.barbell_squats_execution)
val breedingLegsExecution = context.getString(R.string.breeding_legs_in_the_simulator_execution)
val reductionLegsExecution =
    context.getString(R.string.reduction_of_legs_in_the_simulator_execution)
val legPressExecution = context.getString(R.string.leg_press_in_the_simulator_execution)
val legExtensionExecution = context.getString(R.string.leg_extension_in_the_simulator_execution)
val bendingLegsExecution =
    context.getString(R.string.bending_the_legs_in_the_simulator_execution)
val benchPressExecution = context.getString(R.string.bench_press_execution)
val dumbbellBenchPressHorizontalBenchExecution =
    context.getString(R.string.dumbbell_bench_press_on_a_horizontal_bench_execution)
val breedingDumbbellsInclineBenchExecution =
    context.getString(R.string.breeding_dumbbells_on_an_incline_bench_execution)
val inclineDumbbellBenchPressExecution =
    context.getString(R.string.incline_dumbbell_bench_press_execution)
val pullDownUpperBlockExecution =
    context.getString(R.string.pulldown_of_the_upper_block_to_the_chest_execution)
val dumbbellPullBeltExecution = context.getString(R.string.dumbbell_pull_to_the_belt_execution)
val deadliftExecution = context.getString(R.string.deadlift_execution)
val hyperextensionExecution = context.getString(R.string.hyperextension_on_the_bench_execution)
val tBarRowOnTheBlockExecution = context.getString(R.string.T_bar_row_on_the_block_execution)
val standingDumbbellPressExecution =
    context.getString(R.string.standing_dumbbell_press_execution)
val standingBarbellPressExecution = context.getString(R.string.standing_barbell_press_execution)
val liftingDumbbellsToTheSidesExecution =
    context.getString(R.string.lifting_dumbbells_to_the_sides_execution)
val liftingDumbbellsInFrontOfYouExecution =

```

```

context.getString(R.string.lifting_dumbbells_in_front_of_you_execution)

val icHammer = context.getDrawable(R.drawable.ic_hammer)
val icBarbellLift = context.getDrawable(R.drawable.ic_barbell_lift)
val icBendingArm = context.getDrawable(R.drawable.ic_bending_arm)
val icBlockPullTriceps = context.getDrawable(R.drawable.ic_block_pull_triceps)
val icForearm = context.getDrawable(R.drawable.ic_forearm)
val icFrenchPress = context.getDrawable(R.drawable.ic_french_press)
val icTricepsInSimulator = context.getDrawable(R.drawable.ic_triceps_simulator)
val icStandDumbbellPress = context.getDrawable(R.drawable.ic_stand_dumbbell_press)
val icStandBarbellPress = context.getDrawable(R.drawable.ic_stand_barbell_press)
val icStandDumbbellToTheSides = context.getDrawable(R.drawable.ic_stand_dumbbell_to_the_sides)
val icDumbbellInFront = context.getDrawable(R.drawable.ic_dumbbell_in_front)
val icSquats = context.getDrawable(R.drawable.ic_squats)
val icReductionLegs = context.getDrawable(R.drawable.ic_reduction_legs)
val icBreedingLegs = context.getDrawable(R.drawable.ic_breeding_legs)
val icLegPress = context.getDrawable(R.drawable.ic_leg_press)
val icLegExtension = context.getDrawable(R.drawable.ic_leg_extension)
val icBendingLegs = context.getDrawable(R.drawable.ic_bending_legs)
val icPullDown = context.getDrawable(R.drawable.icpulldown)
val icDumbbellPullBelt = context.getDrawable(R.drawable.ic_dumbbell_pull_belt)
val icDeadlift = context.getDrawable(R.drawable.ic_deadlift)
val icHyperextension = context.getDrawable(R.drawable.ic_hyperextension)
val icTBar = context.getDrawable(R.drawable.ic_t_bar)
val icBenchPress = context.getDrawable(R.drawable.ic_bench_press)
val icBreedingDumbbell = context.getDrawable(R.drawable.ic_breeding_dumbbell)
val icInclineBenchPress = context.getDrawable(R.drawable.ic_incline_bench_press)
val icHorizontalPress = context.getDrawable(R.drawable.ic_horizontal_press)

val enduranceRunningExecution = context.getString(R.string.endurance_running_execution)
val pullUpsExecution = context.getString(R.string.pull_ups_execution)
val burpeeExecution = context.getString(R.string.burpee_execution)
val enduranceRunning = context.getString(R.string.endurance_running)
val pullUps = context.getString(R.string.pull_ups)
val burpee = context.getString(R.string.burpee)
val warmUp = context.getString(R.string.warm_up)
val warmUpExecution = context.getString(R.string.warm_up_execution)
val icEnduranceRunning = context.getDrawable(R.drawable.ic_endurance_running)
val icPullUps = context.getDrawable(R.drawable.ic_pull_ups)
val icBurpee = context.getDrawable(R.drawable.ic_burpee)

```

```

val treadmill = context.getString(R.string.treadmill)
val exerciseBike = context.getString(R.string.exercise_bike)
val ellipticalTrainer = context.getString(R.string.elliptical_trainer)
val rowingMachine = context.getString(R.string.rowing_machine)
val treadmillExecution = context.getString(R.string.treadmill_execution)
val exerciseBikeExecution = context.getString(R.string.exercise_bike_execution)
val ellipticalTrainerExecution = context.getString(R.string.elliptical_trainer_execution)
val rowingMachineExecution = context.getString(R.string.rowing_machine_execution)
val icTreadmill = context.getDrawable(R.drawable.ic_treadmill)
val icExerciseBike = context.getDrawable(R.drawable.ic_exercise_bike)
val icEllipticalTrainer = context.getDrawable(R.drawable.ic_elliptical_trainer)
val icRowingMachine = context.getDrawable(R.drawable.ic_rowing_machine)
val icWarmUp = context.getDrawable(R.drawable.ic_warm_up)

}

```

```

//App
import android.app.Application
import dagger.hilt.android.HiltAndroidApp

```

```

@HiltAndroidApp
class App : Application()

```

```

//Constants
object Constants {
    const val GIF =
        "https://media.tenor.com/v_qPOJw06Q0AAAAAd/flexing-flex.gif"
    const val CARDIO = "Cardio"
    const val HIIT = "Hiit"
    const val BASE_URL =
        "https://tasty.p.rapidapi.com/recipes/"
    const val api_key = "6854de60bdmsh734fda289feffd5p1db60bjsn900a106c49fc"
    const val FROM = 0
    const val SIZE = 50
    const val TAGS = "schwartz_seasoned_pro"
}

```

```

//MainActivity
import android.os.Bundle
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.NavController

```

```

import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.setupWithNavController
import com.viggo.lightweight.databinding.ActivityMainBinding
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    @Inject
    lateinit var router: Router
    private val viewModel: MainActivityViewModel by viewModels()

    private lateinit var binding: ActivityMainBinding
    private val navController: NavController by lazy {
        (supportFragmentManager.findFragmentById(R.id.container_fragment) as NavHostFragment).navController
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        setupNavigation()

        binding.bottomNavigation.setupWithNavController(navController)
        binding.bottomNavigation.setOnNavigationItemSelectedListener {
            when (it.itemId) {
                R.id.mainFragment -> {
                    if (navController.currentDestination?.id == 2131362241) {
                        it.isCheckedable = false
                    } else {
                        it.isCheckedable = true
                        viewModel.navigateToTrainingFragment()
                    }
                }
                R.id.directoryFragment -> {
                    if (navController.currentDestination?.id == 2131362241) {
                        it.isCheckedable = false
                    } else {
                        it.isCheckedable = true
                        viewModel.navigateToDirectoryFragment()
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    R.id.settingsFragment -> {
        if (navController.currentDestination?.id == 2131362241) {
            it.isCheckable = false
        } else {
            it.isCheckable = true
            viewModel.navigateToSettingsFragment()
        }
    }
}
return@setOnNavigationItemSelectedListener true
}

}

private fun setupNavigation() {
    router.attachNavController(navController)
}

}

//MainActivityScreen
import androidx.navigation.NavController
import com.viggo.lightweight.navigation.NavigationComponentScreen

class MainActivityScreen(private val to: ToFragment) : NavigationComponentScreen {
    override fun execute(navigator: NavController) {
        when (to) {
            ToFragment.HomeFragment -> {
                navigator.navigate(R.id.mainFragment)
            }
            ToFragment.DirectoryFragment -> {
                navigator.navigate(R.id.directoryFragment)
            }
            ToFragment.SettingsFragment -> {
                navigator.navigate(R.id.settingsFragment)
            }
        }
    }
}

```

```

enum class ToFragment {
    HomeFragment, DirectoryFragment, SettingsFragment
}

}

//MainActivityViewModel
import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

@HiltViewModel
class MainActivityViewModel @Inject constructor(val router: Router) : BaseViewModel() {

    fun navigateToTrainingFragment() {
        router.execute(Command.Navigate(MainActivityScreen(MainActivityScreen.ToFragment.HomeFragment)))
    }

    fun navigateToDirectoryFragment() {

router.execute(Command.Navigate(MainActivityScreen(MainActivityScreen.ToFragment.DirectoryFragment)))
    }

    fun navigateToSettingsFragment() {
        router.execute(Command.Navigate(MainActivityScreen(MainActivityScreen.ToFragment.SettingsFragment)))
    }

}

//UI

//AboutFragment
class AboutFragment : BaseFragment<AboutViewModel>() {

    override val viewModel: AboutViewModel by viewModels()
    lateinit var binding: FragmentAboutBinding

    override fun onCreateView(
        inflater: LayoutInflater,

```

```

        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentAboutBinding.inflate(layoutInflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        binding.back.setOnClickListener {
            findNavController().popBackStack()
        }
    }
}

```

```

//AboutViewModel
import com.viggo.lightweight.base.BaseViewModel

```

```

class AboutViewModel:BaseViewModel() {
}

```

```

//DirectoryFragment
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import com.viggo.lightweight.R
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentDirectoryBinding
import dagger.hilt.android.AndroidEntryPoint

```

```

@AndroidEntryPoint
class DirectoryFragment : BaseFragment<DirectoryViewModel>() {

    override val viewModel: DirectoryViewModel by viewModels()
    lateinit var binding: FragmentDirectoryBinding

    override fun onCreateView(
        inflater: LayoutInflater,

```

```

        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentDirectoryBinding.inflate(layoutInflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.icSportNutrition.setOnClickListener {
            viewModel.navigateToDirectoryInfo(getString(R.string.sport_nutrition))
        }
        binding.icGeneralTips.setOnClickListener {
            viewModel.navigateToDirectoryInfo(getString(R.string.general_tips))
        }
        binding.icProduct.setOnClickListener {
            viewModel.navigateToDirectoryInfo(getString(R.string.food_and_nutrition))
        }
    }
}

```

//DirectoryScreen

```

import androidx.navigation.NavController
import com.viggo.lightweight.navigation.NavigationComponentScreen

```

```

class DirectoryScreen(private val type: String) : NavigationComponentScreen {

    override fun execute(navigator: NavController) {
        navigator.navigate(
            DirectoryFragmentDirections.actionDirectoryFragmentToDirectoryInfoFragment(type)
        )
    }

}

```

//DirectoryViewModel

```

import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

@HiltViewModel
class DirectoryViewModel @Inject constructor(private val router: Router) : BaseViewModel() {

    fun navigateToDirectoryInfo(type: String) {
        router.execute(Command.Navigate(DirectoryScreen(type)))
    }
}

//DirectoryInfoAdapter
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.viggo.lightweight.R
import com.viggo.lightweight.data.Category
import com.viggo.lightweight.data.Directory
import com.viggo.lightweight.databinding.ItemDirectoryInfoBinding

class DirectoryInfoAdapter(
    private val onClickProductListener: (Category) -> Unit
):
    ListAdapter<Directory, DirectoryInfoAdapter.GifViewHolder>(DiffCallback()) {

    override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
        holderShop.bind(currentList[position], onClickProductListener)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
        return GifViewHolder(
            ItemDirectoryInfoBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            )
        )
    }
}

```

```

    )
    )
}

class GifViewHolder(private var binding: ItemDirectoryInfoBinding) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(info: Directory, onClickProductListener: (Category) -> Unit) {
        binding.apply {
            var isVisible = false
            imgArrow.setOnClickListener {
                if (!isVisible) {
                    isVisible = true
                    imgArrow.setImageResource(R.drawable.ic_arrow_up)
                    description.visibility = View.VISIBLE
                } else {
                    isVisible = false
                    imgArrow.setImageResource(R.drawable.ic_arrow_down)
                    description.visibility = View.GONE
                }
            }
            name.text = info.name
            description.text = info.description
        }
    }
}

class DiffCallback : DiffUtil.ItemCallback<Directory>() {

    override fun areItemsTheSame(oldItem: Directory, newItem: Directory): Boolean {
        return oldItem == newItem
    }

    override fun areContentsTheSame(
        oldItem: Directory,
        newItem: Directory
    ): Boolean {
        return oldItem == newItem
    }
}

```

```
}
```

//DirectoryInfoFragment

```
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.navArgs
import com.viggo.lightweight.R
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentDirectoryInfoBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class DirectoryInfoFragment : BaseFragment<DirectoryInfoViewModel>() {
    override val viewModel: DirectoryInfoViewModel by viewModels()
    lateinit var binding: FragmentDirectoryInfoBinding
    private val args: DirectoryInfoFragmentArgs by navArgs()
    private val powerWorkoutAdapter = DirectoryInfoAdapter(onClickProductListener = { selectedGif ->

    })
    private val nutritionAdapter = NutritionAdapter(onClickProduct = { selectedProduct ->
        viewModel.navigateToDetailedInfo(selectedProduct)
    })

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
    ): View {
        binding = FragmentDirectoryInfoBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val typeOfInfo = args.type
        if (typeOfInfo == getString(R.string.sport_nutrition)) {
            val modelSportNutritions = viewModel.createSportNutritionModel()
            binding.recycler.adapter = powerWorkoutAdapter
            powerWorkoutAdapter.submitList(modelSportNutritions)
        }
    }
}
```

```

    }

    if (typeOfInfo == getString(R.string.general_tips)) {
        val modelOfGeneralTips = viewModel.createGeneralTipsModel()
        binding.recycler.adapter = powerWorkoutAdapter
        powerWorkoutAdapter.submitList(modelOfGeneralTips)
    }
    if (typeOfInfo == getString(R.string.food_and_nutrition)) {
        binding.progress.visibility = View.VISIBLE
        viewModel.getNutrition()
        viewModel.list.observe(viewLifecycleOwner) {
            if (it != null) {
                binding.progress.visibility = View.GONE
            }
            binding.recycler.adapter = nutritionAdapter
            nutritionAdapter.submitList(it?.foodList)
        }
    }
}

```

```

//DirectoryInfoScreen
import androidx.navigation.NavController
import com.viggo.lightweight.data.food.FoodItem
import com.viggo.lightweight.navigation.NavigationComponentScreen

class DirectoryInfoScreen(private val foodItem: FoodItem) : NavigationComponentScreen {

    override fun execute(navigator: NavController) {
        navigator.navigate(
            DirectoryInfoFragmentDirections.actionDirectoryInfoToDetailedInfo(foodItem)
        )
    }

}

```

```

//DirectoryInfoViewModel
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModelScope
import com.viggo.lightweight.base.BaseViewModel

```

```

import com.viggo.lightweight.data.Directory
import com.viggo.lightweight.data.food.Food
import com.viggo.lightweight.data.food.FoodItem
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import com.viggo.lightweight.usecase.GetNutritionUseCase
import com.viggo.lightweight.utill.StringsProvider
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class DirectoryInfoViewModel @Inject constructor(
    private val stringsProvider: StringsProvider,
    private val useCase: GetNutritionUseCase,
    private val router: Router
) : BaseViewModel() {

    val list = MutableLiveData<Food?>()

    fun createSportNutritionModel(): List<Directory> {
        return listOf(
            Directory(stringsProvider.l_carnitine, stringsProvider.l_carnitineMeaning),
            Directory(stringsProvider.creatine, stringsProvider.creatineMeaning),
            Directory(stringsProvider.protein, stringsProvider.proteinMeaning),
            Directory(stringsProvider.bcaa, stringsProvider.bcaaMeaning),
            Directory(stringsProvider.gainer, stringsProvider.gainerMeaning),
            Directory(stringsProvider.glutamine, stringsProvider.glutamineMeaning),
        )
    }

    fun createGeneralTipsModel(): List<Directory> {
        return listOf(
            Directory(stringsProvider.improvePower, stringsProvider.improvePowerTip),
            Directory(stringsProvider.increaseStamina, stringsProvider.increaseStaminaTip),
            Directory(stringsProvider.gainMuscle, stringsProvider.gainMuscleTip),
            Directory(stringsProvider.weightLoss, stringsProvider.weightLossTip)
        )
    }

    fun getNutrition() {

```

```

viewModelScope.launch {
    val nutritions = useCase.getNutrition()
    list.value = nutritions
}
}

fun navigateToDetailedInfo(food: FoodItem) {
    router.execute(Command.Navigate(DirectoryInfoScreen(food)))
}

}

```

```

//NutritionAdapter
import android.content.Context
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.viggo.lightweight.R
import com.viggo.lightweight.data.food.FoodItem
import com.viggo.lightweight.databinding.ItemNutritionBinding

class NutritionAdapter(
    private val onClickProduct: (FoodItem) -> Unit
) : ListAdapter<FoodItem, NutritionAdapter.GifViewHolder>(DiffCallback()) {

    override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
        holderShop.bind(currentList[position], onClickProduct)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
        return GifViewHolder(
            ItemNutritionBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            ), parent.context
        )
    }
}

```

```

class GifViewHolder(
    private var binding: ItemNutritionBinding,
    private val context: Context,
) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(info: FoodItem, onClickProduct: (FoodItem) -> Unit) {
        with(binding) {
            nameProduct.text = info.name
            carbohydrates.text = context.getString(
                R.string.carbohydrates,
                info.nutrition.carbohydrates.toString()
            )
            protein.text = context.getString(
                R.string.proteinF,
                info.nutrition.protein.toString()
            )
            fat.text = context.getString(
                R.string.fat,
                info.nutrition.fat.toString()
            )
            calories.text = context.getString(
                R.string.calories,
                info.nutrition.calories.toString()
            )
            sugar.text = context.getString(
                R.string.sugar,
                info.nutrition.sugar.toString()
            )
            Glide.with(itemView.context).load(info.thumbnailUrl).into(icProduct)
            itemView.setOnClickListener { onClickProduct.invoke(info) }

        }
    }
}

class DiffCallback : DiffUtil.ItemCallback<FoodItem>() {

    override fun areItemsTheSame(oldItem: FoodItem, newItem: FoodItem): Boolean {
        return oldItem == newItem
    }
}

```

```

        override fun areContentsTheSame(
            oldItem: FoodItem,
            newItem: FoodItem
        ): Boolean {
            return oldItem == newItem
        }
    }
}

//DetailedInfoFragment
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.view.isVisible
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.navArgs
import com.bumptech.glide.Glide
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentDetailedInfoBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class DetailedInfoFragment : BaseFragment<DetailedInfoViewModel>() {

    override val viewModel: DetailedInfoViewModel by viewModels()
    lateinit var binding: FragmentDetailedInfoBinding
    private val args: DetailedInfoFragmentArgs by navArgs()
    private val instructionAdapter = InstructionAdapter()

    private val ingredientsAdapter = IngredientsAdapter()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentDetailedInfoBinding.inflate(inflater, container, false)
        return binding.root
    }
}

```

```

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onCreateView(view, savedInstanceState)
    binding.recyclerIngredients.adapter = ingredientsAdapter
    binding.recyclerInstruction.adapter = instructionAdapter

    Glide.with(requireContext()).load(args.food.thumbnailUrl).into(binding.icProduct)
    if (args.food.totalTime?.displayTier.isNullOrEmpty()) {
        binding.cookingTime.isVisible = false
        binding.icCircle.isVisible = false
        binding.icCircleGradient.isVisible = false
    } else {
        binding.cookingTime.text = args.food.totalTime?.displayTier
    }
    val listInstructions = args.food.instruction
    val listIngredients = args.food.sections
    instructionAdapter.submitList(listInstructions)
    ingredientsAdapter.submitList(listIngredients)
}
}

```

//DetailedInfoViewModel

```

import com.viggo.lightweight.base.BaseViewModel
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

@HiltViewModel
class DetailedInfoViewModel @Inject constructor() : BaseViewModel() {
}

```

//IngredientsAdapter

```

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.viggo.lightweight.data.food.SectionsItem
import com.viggo.lightweight.databinding.ItemIngredientsBinding

class IngredientsAdapter(
) : ListAdapter<SectionsItem, IngredientsAdapter.GifViewHolder>(DiffCallback()) {
}

```

```

override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
    holderShop.bind(currentList[position])
}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
    return GifViewHolder(
        ItemIngredientsBinding.inflate(
            LayoutInflater.from(parent.context),
            parent,
            false
        ),
    )
}

class GifViewHolder(
    private var binding: ItemIngredientsBinding,
) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(info: SectionsItem) {
        with(binding) {
            ingredientName.text = info.components.map { it.ingredients.name }.toString()
        }
    }
}

class DiffCallback : DiffUtil.ItemCallback<SectionsItem>() {

    override fun areItemsTheSame(
        oldItem: SectionsItem,
        newItem: SectionsItem
    ): Boolean {
        return oldItem == newItem
    }

    override fun areContentsTheSame(
        oldItem: SectionsItem,
        newItem: SectionsItem
    ): Boolean {
        return oldItem == newItem
    }
}

```

```

    }
}

}

//InstructionsAdapter
import android.content.Context
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.viggo.lightweight.R
import com.viggo.lightweight.data.food.InstructionsItem
import com.viggo.lightweight.databinding.ItemInstructionBinding

class InstructionAdapter(
) : ListAdapter<InstructionsItem, InstructionAdapter.GifViewHolder>(DiffCallback()) {

    override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
        holderShop.bind(currentList[position])
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
        return GifViewHolder(
            ItemInstructionBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            ), parent.context
        )
    }
}

class GifViewHolder(
    private var binding: ItemInstructionBinding,
    private val context: Context,
) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(info: InstructionsItem) {
        with(binding) {
            numStep.text = context.getString(R.string.step, info.position.toString())
        }
    }
}

```

```

        instruction.text = info.displayText
    }
}
}

class DiffCallback : DiffUtil.ItemCallback<InstructionsItem>() {

    override fun areItemsTheSame(
        oldItem: InstructionsItem,
        newItem: InstructionsItem
    ): Boolean {
        return oldItem == newItem
    }

    override fun areContentsTheSame(
        oldItem: InstructionsItem,
        newItem: InstructionsItem
    ): Boolean {
        return oldItem == newItem
    }
}

}

//MeasurementsFragment
import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.widget.doOnTextChanged
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentMeasurementsBinding

class MeasurementsFragment : BaseFragment<MeasurementsViewModel>() {

    override val viewModel: MeasurementsViewModel by viewModels()
    lateinit var binding: FragmentMeasurementsBinding

```

```

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    binding = FragmentMeasurementsBinding.inflate(inflater, container, false)
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    setUi()
}

private fun setUi() {
    with(binding) {
        val sharedPref = context?.getSharedPreferences("MEASUREMENTS", Context.MODE_PRIVATE)
        var oldWaist = sharedPref?.getString("oldWaist", "")
        var oldHip = sharedPref?.getString("oldHip", "")
        var oldChest = sharedPref?.getString("oldChest", "")
        var oldBiceps = sharedPref?.getString("oldBiceps", "")
        var oldButtocks = sharedPref?.getString("oldButtocks", "")
        var oldCalves = sharedPref?.getString("oldCalves", "")
        var newWaist = sharedPref?.getString("newWaist", "")
        var newHip = sharedPref?.getString("newHip", "")
        var newChest = sharedPref?.getString("newChest", "")
        var newBiceps = sharedPref?.getString("newBiceps", "")
        var newButtocks = sharedPref?.getString("newButtocks", "")
        var newCalves = sharedPref?.getString("newCalves", "")
        oldEdtWaist.setText(oldWaist)
        oldEdtHip.setText(oldHip)
        oldEdtChest.setText(oldChest)
        oldEdtBiceps.setText(oldBiceps)
        oldEdtButtocks.setText(oldButtocks)
        oldEdtCalves.setText(oldCalves)
        newEdtWaist.setText(newWaist)
        newEdtHip.setText(newHip)
        newEdtChest.setText(newChest)
        newEdtBiceps.setText(newBiceps)
        newEdtButtocks.setText(newButtocks)
        newEdtCalves.setText(newCalves)
    }
}

```

```

oldEdtWaist.doOnTextChanged { text, start, before, count ->
    oldWaist = text.toString()
}
oldEdtHip.doOnTextChanged { text, start, before, count ->
    oldHip = text.toString()
}
oldEdtChest.doOnTextChanged { text, start, before, count ->
    oldChest = text.toString()
}
oldEdtBiceps.doOnTextChanged { text, start, before, count ->
    oldBiceps = text.toString()
}
oldEdtButtocks.doOnTextChanged { text, start, before, count ->
    oldButtocks = text.toString()
}
oldEdtCalves.doOnTextChanged { text, start, before, count ->
    oldCalves = text.toString()
}

newEdtWaist.doOnTextChanged { text, start, before, count ->
    newWaist = text.toString()
}
newEdtHip.doOnTextChanged { text, start, before, count ->
    newHip = text.toString()
}
newEdtChest.doOnTextChanged { text, start, before, count ->
    newChest = text.toString()
}
newEdtBiceps.doOnTextChanged { text, start, before, count ->
    newBiceps = text.toString()
}
newEdtButtocks.doOnTextChanged { text, start, before, count ->
    newButtocks = text.toString()
}
newEdtCalves.doOnTextChanged { text, start, before, count ->
    newCalves = text.toString()
}

save.setOnClickListener {
    val editor: SharedPreferences.Editor? = sharedPref?.edit()
    if (newWaist.isNullOrEmpty()) {

```

```

        editor?.putString("oldWaist", oldEdtWaist.text.toString())
        editor?.putString("oldHip", oldEdtHip.text.toString())
        editor?.putString("oldChest", oldEdtChest.text.toString())
        editor?.putString("oldBiceps", oldEdtBiceps.text.toString())
        editor?.putString("oldButtocks", oldEdtButtocks.text.toString())
        editor?.putString("oldCalves", oldEdtCalves.text.toString())
        editor?.putString("newWaist", newEdtWaist.text.toString())
        editor?.putString("newHip", newEdtHip.text.toString())
        editor?.putString("newChest", newEdtChest.text.toString())
        editor?.putString("newBiceps", newEdtBiceps.text.toString())
        editor?.putString("newButtocks", newEdtButtocks.text.toString())
        editor?.putString("newCalves", newEdtCalves.text.toString())
        editor?.commit()
        editor?.apply()
    } else {
        editor?.putString("oldWaist", newEdtWaist.text.toString())
        editor?.putString("oldHip", newEdtHip.text.toString())
        editor?.putString("oldChest", newEdtChest.text.toString())
        editor?.putString("oldBiceps", newEdtBiceps.text.toString())
        editor?.putString("oldButtocks", newEdtButtocks.text.toString())
        editor?.putString("oldCalves", newEdtCalves.text.toString())
        editor?.putString("newWaist", "")
        editor?.putString("newHip", "")
        editor?.putString("newChest", "")
        editor?.putString("newBiceps", "")
        editor?.putString("newButtocks", "")
        editor?.putString("newCalves", "")
        editor?.commit()
        editor?.apply()
    }
    findNavController().popBackStack()
}
}
}
}
}

```

```

//MeasurementsViewModel
import com.viggo.lightweight.base.BaseViewModel
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

```

```

@HiltViewModel
class MeasurementsViewModel @Inject constructor() : BaseViewModel() {
}

//ProfileFragment
import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.widget.doOnTextChanged
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentProfileBinding

class ProfileFragment : BaseFragment<ProfileViewModel>() {

    override val viewModel: ProfileViewModel by viewModels()
    lateinit var binding: FragmentProfileBinding

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentProfileBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setUi()
    }

    private fun setUi() {
        with(binding) {

```

```

val sharedPref = context?.getSharedPreferences("PROFILE", Context.MODE_PRIVATE)
var yourName = sharedPref?.getString("name", "")
var yourAge = sharedPref?.getString("age", "")
var yourGender = sharedPref?.getString("gender", "")
var yourWeight = sharedPref?.getString("weight", "")
name.setText(yourName)
age.setText(yourAge)
gender.setText(yourGender)
weight.setText(yourWeight)
txtBack.setOnClickListener {
    findNavController().popBackStack()
}
name.doOnTextChanged { text, start, before, count ->
    yourName = text.toString()
}
age.doOnTextChanged { text, start, before, count ->
    yourAge = text.toString()
}
gender.doOnTextChanged { text, start, before, count ->
    yourGender = text.toString()
}
weight.doOnTextChanged { text, start, before, count ->
    yourWeight = text.toString()
}
save.setOnClickListener {
    val editor: SharedPreferences.Editor? = sharedPref?.edit()
    editor?.putString("name", name.text.toString())
    editor?.putString("age", age.text.toString())
    editor?.putString("gender", gender.text.toString())
    editor?.putString("weight", weight.text.toString())
    editor?.commit()
    editor?.apply()
}
}
}
}

```

//ProfileViewModel

```

import com.viggo.lightweight.base.BaseViewModel
import dagger.hilt.android.lifecycle.HiltViewModel

```

```
import javax.inject.Inject
```

```
@HiltViewModel
```

```
class ProfileViewModel @Inject constructor() : BaseViewModel()
```

```
    //SettingsFragment
```

```
import android.os.Bundle
```

```
import android.view.LayoutInflater
```

```
import android.view.View
```

```
import android.view.ViewGroup
```

```
import androidx.fragment.app.viewModels
```

```
import com.viggo.lightweight.base.BaseFragment
```

```
import com.viggo.lightweight.databinding.FragmentSettingsBinding
```

```
import dagger.hilt.android.AndroidEntryPoint
```

```
@AndroidEntryPoint
```

```
class SettingsFragment : BaseFragment<SettingsViewModel>() {
```

```
    override val viewModel: SettingsViewModel by viewModels()
```

```
    lateinit var binding: FragmentSettingsBinding
```

```
    override fun onCreateView(
```

```
        inflater: LayoutInflater,
```

```
        container: ViewGroup?,
```

```
        savedInstanceState: Bundle?
```

```
): View {
```

```
    binding = FragmentSettingsBinding.inflate(inflater, container, false)
```

```
    return binding.root
```

```
}
```

```
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
```

```
        super.onViewCreated(view, savedInstanceState)
```

```
        onClickListeners()
```

```
}
```

```
    private fun onClickListeners() {
```

```
        with(binding) {
```

```
            profile.setOnClickListener {
```

```
                viewModel.navigateToProfile()
```

```
            }
```

```
            aboutApp.setOnClickListener {
```

```

        viewModel.navigateToAboutApp()
    }
    measurements.setOnClickListener {
        viewModel.navigateToMeasurements()
    }
}
}
}

```

//SettingsScreen

```

import androidx.navigation.NavController
import com.viggo.lightweight.R
import com.viggo.lightweight.navigation.NavigationComponentScreen

class SettingsScreen(private val toFragment: ToFragment) : NavigationComponentScreen {
    override fun execute(navigator: NavController) {
        when (toFragment) {
            ToFragment.Profile -> {
                navigator.navigate(R.id.action_settingsFragment_to_profileFragment)
            }
            ToFragment.AboutApp -> {
                navigator.navigate(R.id.action_settingsFragment_to_aboutApp)
            }
            ToFragment.Measurements -> {
                navigator.navigate(R.id.action_settingsFragment_to_Measurements)
            }
        }
    }
}

enum class ToFragment {
    Profile, AboutApp, Measurements
}
}

```

//SettingsViewModel

```

import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.lifecycle.HiltViewModel

```

```
import javax.inject.Inject
```

```
@HiltViewModel
```

```
class SettingsViewModel @Inject constructor(val router: Router) : BaseViewModel() {
```

```
    fun navigateToProfile() {
```

```
        router.execute(Command.Navigate(SettingsScreen(SettingsScreen.ToFragment.Profile)))
```

```
    }
```

```
    fun navigateToAboutApp() {
```

```
        router.execute(Command.Navigate(SettingsScreen(SettingsScreen.ToFragment.AboutApp)))
```

```
    }
```

```
    fun navigateToMeasurements() {
```

```
        router.execute(Command.Navigate(SettingsScreen(SettingsScreen.ToFragment.Measurements)))
```

```
    }
```

```
}
```

//SplashFragment

```
import android.os.Bundle
```

```
import android.util.Log
```

```
import android.view.LayoutInflater
```

```
import android.view.View
```

```
import android.view.ViewGroup
```

```
import androidx.fragment.app.viewModels
```

```
import androidx.lifecycle.LifecycleScope
```

```
import androidx.navigation.fragment.findNavController
```

```
import com.bumptech.glide.Glide
```

```
import com.viggo.lightweight.Constants
```

```
import com.viggo.lightweight.base.BaseFragment
```

```
import com.viggo.lightweight.databinding.FragmentSplashBinding
```

```
import dagger.hilt.android.AndroidEntryPoint
```

```
import kotlinx.coroutines.launch
```

```
@AndroidEntryPoint
```

```
class SplashFragment : BaseFragment<SplashViewModel>() {
```

```
    override val viewModel: SplashViewModel by viewModels()
```

```
    lateinit var binding: FragmentSplashBinding
```

```
    override fun onCreateView(
```

```
        inflater: LayoutInflater,
```

```
        container: ViewGroup?,
```

```

        savedInstanceState: Bundle?
    ): View {
        binding = FragmentSplashBinding.inflate(layoutInflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        viewModel.loadData()
        observeViewModel()
        Glide.with(requireContext()).load(Constants.GIF).into(binding.imageSplash)
        Log.v("SPLASH", "${findNavController().currentDestination?.id}")
    }

    private fun observeViewModel() {
        lifecycleScope.launch {
            viewModel.state.collect { state ->
                if (state.dataLoaded) {
                    viewModel.navigateToMain()
                }
            }
        }
    }
}

```

//SplashScreen

```

import androidx.navigation.NavController
import com.viggo.lightweight.R
import com.viggo.lightweight.navigation.NavigationComponentScreen

```

```

class SplashScreen : NavigationComponentScreen {

    override fun execute(navigator: NavController) {
        navigator.navigate(R.id.action_splashFragment_to_mainFragment)
    }

}

```

//SplashViewModel

```

import androidx.lifecycle.viewModelScope
import com.viggo.lightweight.base.BaseViewModel

```

```

import com.viggo.lightweight.data.SplashState
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

```

```
@HiltViewModel
```

```

class SplashViewModel @Inject constructor(private val router: Router) : BaseViewModel() {
    val splash = MutableStateFlow(0)
    val state = MutableStateFlow(SplashState())

    fun loadData() {
        viewModelScope.launch {
            while (splash.value < 9) {
                splash.value = splash.value.plus(1)
                delay(500)
            }
            state.value = state.value.copy(dataLoaded = true)
        }
    }

    fun navigateToMain() {
        router.execute(Command.Navigate(SplashScreen()))
    }
}

```

//ExerciseAdapter

```

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.viggo.lightweight.R
import com.viggo.lightweight.data.Exercises
import com.viggo.lightweight.databinding.ItemExerciseBinding

```

```

class ExerciseAdapter(
    private val onClickProductListener: (Exercises) -> Unit
) : ListAdapter<Exercises, ExerciseAdapter.GifViewHolder>(DiffCallback()) {

    override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
        holderShop.bind(currentList[position], onClickProductListener)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
        return GifViewHolder(
            ItemExerciseBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            )
        )
    }
}

class GifViewHolder(private var binding: ItemExerciseBinding) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(workout: Exercises, onClickProductListener: (Exercises) -> Unit) {
        binding.apply {
            var isVisible = false
            icExercise.setOnClickListener {
                if (!isVisible) {
                    isVisible = true
                    imgArrow.setImageResource(R.drawable.arrow_up)
                    descriptionExercise.visibility = View.VISIBLE
                } else {
                    isVisible = false
                    imgArrow.setImageResource(R.drawable.arrow_down)
                    descriptionExercise.visibility = View.GONE
                }
            }
            nameExercise.text = workout.name
            descriptionExercise.text = workout.description
            icExercise.setImageDrawable(workout.image)
        }
    }
}

```

```

class DiffCallback : DiffUtil.ItemCallback<Exercises>() {

    override fun areItemsTheSame(oldItem: Exercises, newItem: Exercises): Boolean {
        return oldItem == newItem
    }

    override fun areContentsTheSame(oldItem: Exercises, newItem: Exercises): Boolean {
        return oldItem == newItem
    }
}

```

```

//ExerciseFragment
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.navArgs
import com.viggo.lightweight.Constants
import com.viggo.lightweight.R
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentExerciseBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class ExerciseFragment : BaseFragment<ExerciseViewModel>() {

    override val viewModel: ExerciseViewModel by viewModels()
    lateinit var binding: FragmentExerciseBinding
    private val args: ExerciseFragmentArgs by navArgs()
    private val exerciseAdapter =
        ExerciseAdapter(onClickListener = { selectedMuscleGroup ->
        })

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentExerciseBinding.inflate(inflater, container, false)
    }
}

```

```

        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.recycler.adapter = exerciseAdapter
        binding.nameGroup.text = args.selectedMuscleGroup
        createTraining(args.selectedMuscleGroup)
    }

    private fun createTraining(selectedGroup: String) {
        when (selectedGroup) {
            getString(R.string.chest) -> {
                val chestTraining = viewModel.createChestTraining()
                exerciseAdapter.submitList(chestTraining)
            }
            getString(R.string.biceps) -> {
                val bicepsTraining = viewModel.createBicepsTraining()
                exerciseAdapter.submitList(bicepsTraining)
            }
            getString(R.string.triceps) -> {
                val tricepsTraining = viewModel.createTricepsTraining()
                exerciseAdapter.submitList(tricepsTraining)
            }
            getString(R.string.back) -> {
                val backTraining = viewModel.createBackTraining()
                exerciseAdapter.submitList(backTraining)
            }
            getString(R.string.legs) -> {
                val legsTraining = viewModel.createLegsTraining()
                exerciseAdapter.submitList(legsTraining)
            }
            getString(R.string.shoulders) -> {
                val shouldersTraining = viewModel.createShouldersTraining()
                exerciseAdapter.submitList(shouldersTraining)
            }
            Constants.CARDIO -> {
                val cardio = viewModel.createCardioWorkout()
                exerciseAdapter.submitList(cardio)
            }
            Constants.HIIT -> {
                val hiit = viewModel.createHIITWorkout()

```

```

        exerciseAdapter.submitList(hiit)
    }
}
}
}

```

```

//ExerciseViewModel
import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.data.Exercises
import com.viggo.lightweight.utill.StringsProvider
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

@HiltViewModel
class ExerciseViewModel @Inject constructor(private val stringsProvider: StringsProvider) :
    BaseViewModel() {

    fun createBicepsTraining(): List<Exercises> {
        return listOf(
            Exercises(
                name = stringsProvider.warmUp,
                description = stringsProvider.warmUpExecution,
                image = stringsProvider.icWarmUp
            ),
            Exercises(
                stringsProvider.hammer,
                description = stringsProvider.hammerExecution,
                stringsProvider.icHammer
            ),
            Exercises(
                name = stringsProvider.barbellLiftBiceps,
                description = stringsProvider.barbellLiftBicepsExecution,
                image = stringsProvider.icBarbellLift
            ),
            Exercises(
                name = stringsProvider.bendingArmsWithDumbbells,
                description = stringsProvider.bendingArmsWithDumbbellsExecution,
                image = stringsProvider.icBendingArm
            ),
            Exercises(
                name = stringsProvider.liftingDumbbellsForearm,
                description = stringsProvider.liftingDumbbellsForearmExecution,

```

```

        image = stringsProvider.icForearm
    )
}

```

```

fun createTricepsTraining(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.blockPullForTriceps,
            description = stringsProvider.blockPullForTricepsExecution,
            image = stringsProvider.icBlockPullTriceps
        ),
        Exercises(
            name = stringsProvider.frenchTricepsPress,
            description = stringsProvider.frenchTricepsPressExecution,
            image = stringsProvider.icFrenchPress
        ), Exercises(
            name = stringsProvider.extensionTriceps,
            description = stringsProvider.extensionTricepsExecution,
            image = stringsProvider.icTricepsInSimulator
        )
    )
}

```

```

fun createShouldersTraining(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.standingDumbbellPress,
            description = stringsProvider.standingDumbbellPressExecution,
            image = stringsProvider.icStandDumbbellPress
        ),
        Exercises(

```

```

        name = stringsProvider.standingBarbellPress,
        description = stringsProvider.standingBarbellPressExecution,
        image = stringsProvider.icStandBarbellPress
    ),
    Exercises(
        name = stringsProvider.liftingDumbbellsToTheSides,
        description = stringsProvider.liftingDumbbellsToTheSidesExecution,
        image = stringsProvider.icStandDumbellToTheSides
    ),
    Exercises(
        name = stringsProvider.liftingDumbbellsInFrontOfYou,
        description = stringsProvider.liftingDumbbellsInFrontOfYouExecution,
        image = stringsProvider.icDumbellInFront
    )
)
}

```

```

fun createLegsTraining(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.barbellSquats,
            description = stringsProvider.barbellSquatsExecution,
            image = stringsProvider.icSquats
        ),
        Exercises(
            name = stringsProvider.breedingLegs,
            description = stringsProvider.breedingLegsExecution,
            image = stringsProvider.icBreedingLegs
        ),
        Exercises(
            name = stringsProvider.reductionLegs,
            description = stringsProvider.reductionLegsExecution,
            image = stringsProvider.icReductionLegs
        ),
        Exercises(
            name = stringsProvider.legPress,
            description = stringsProvider.legPressExecution,

```

```

        image = stringsProvider.icLegPress
    ),
    Exercises(
        name = stringsProvider.legExtension,
        description = stringsProvider.legExtensionExecution,
        image = stringsProvider.icLegExtension
    ),
    Exercises(
        name = stringsProvider.bendingLegs,
        description = stringsProvider.bendingLegsExecution,
        image = stringsProvider.icBendingLegs
    )
)
}

fun createBackTraining(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.pullDownUpperBlock,
            description = stringsProvider.pullDownUpperBlockExecution,
            image = stringsProvider.icPullDown
        ),
        Exercises(
            name = stringsProvider.dumbbellPullBelt,
            description = stringsProvider.dumbbellPullBeltExecution,
            image = stringsProvider.icDumbbellPullBelt
        ),
        Exercises(
            name = stringsProvider.deadlift,
            description = stringsProvider.deadliftExecution,
            image = stringsProvider.icDeadlift
        ),
        Exercises(
            name = stringsProvider.hyperextension,
            description = stringsProvider.hyperextensionExecution,
            image = stringsProvider.icHyperextension
        ),
    )
}

```

```

    Exercises(
        name = stringsProvider.tBarRowOnTheBlock,
        description = stringsProvider.tBarRowOnTheBlockExecution,
        image = stringsProvider.icTBar
    )
)
}

fun createChestTraining(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.benchPress,
            description = stringsProvider.benchPressExecution,
            image = stringsProvider.icBenchPress
        ),
        Exercises(
            name = stringsProvider.dumbbellBenchPressHorizontalBench,
            description = stringsProvider.dumbbellBenchPressHorizontalBenchExecution,
            image = stringsProvider.icHorizontalPress
        ),
        Exercises(
            name = stringsProvider.breedingDumbbellsInclineBench,
            description = stringsProvider.breedingDumbbellsInclineBenchExecution,
            image = stringsProvider.icBreedingDumbbell
        ),
        Exercises(
            name = stringsProvider.inclineDumbbellBenchPress,
            description = stringsProvider.inclineDumbbellBenchPressExecution,
            image = stringsProvider.icInclineBenchPress
        )
    )
}

fun createHIITWorkout(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,

```

```

        description = stringsProvider.warmUpExecution,
        image = stringsProvider.icWarmUp
    ),
    Exercises(
        name = stringsProvider.enduranceRunning,
        description = stringsProvider.enduranceRunningExecution,
        image = stringsProvider.icEnduranceRunning
    ),
    Exercises(
        name = stringsProvider.pullUps,
        description = stringsProvider.pullUpsExecution,
        image = stringsProvider.icPullUps
    ),
    Exercises(
        name = stringsProvider.burpee,
        description = stringsProvider.burpeeExecution,
        image = stringsProvider.icBurpee
    )
)
}

```

```

fun createCardioWorkout(): List<Exercises> {
    return listOf(
        Exercises(
            name = stringsProvider.warmUp,
            description = stringsProvider.warmUpExecution,
            image = stringsProvider.icWarmUp
        ),
        Exercises(
            name = stringsProvider.treadmill,
            description = stringsProvider.treadmillExecution,
            image = stringsProvider.icTreadmill
        ),
        Exercises(
            name = stringsProvider.exerciseBike,
            description = stringsProvider.exerciseBikeExecution,
            image = stringsProvider.icExerciseBike
        ),
        Exercises(
            name = stringsProvider.ellipticalTrainer,
            description = stringsProvider.ellipticalTrainerExecution,
            image = stringsProvider.icEllipticalTrainer
        )
    )
}

```

```

    ), Exercises(
        name = stringsProvider.rowingMachine,
        description = stringsProvider.rowingMachineExecution,
        image = stringsProvider.icRowingMachine
    )
)
}
}

```

```

//PowerWorkoutAdapter
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.viggo.lightweight.data.Category
import com.viggo.lightweight.databinding.ItemPowerWorkoutBinding

class PowerWorkoutAdapter(
    private val onClickProductListener: (Category) -> Unit
) : ListAdapter<Category, PowerWorkoutAdapter.GifViewHolder>(DiffCallback()) {

    override fun onBindViewHolder(holderShop: GifViewHolder, position: Int) {
        holderShop.bind(currentList[position], onClickProductListener)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GifViewHolder {
        return GifViewHolder(
            ItemPowerWorkoutBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            )
        )
    }
}

class GifViewHolder(private var binding: ItemPowerWorkoutBinding) :
    RecyclerView.ViewHolder(binding.root) {

    fun bind(workout: Category, onClickProductListener: (Category) -> Unit) {
        binding.apply {

```

```

        category.text = workout.keyCategory
        image.setImageDrawable(workout.image)
    }
    itemView.setOnClickListener {
        onClickProductListener.invoke(workout)
    }
}
}

class DiffCallback : DiffUtil.ItemCallback<Category>() {

    override fun areItemsTheSame(oldItem: Category, newItem: Category): Boolean {
        return oldItem.keyCategory == newItem.keyCategory
    }

    override fun areContentsTheSame(oldItem: Category, newItem: Category): Boolean {
        return oldItem == newItem
    }
}
}

```

```

//PowerWorkoutFragment
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.navArgs
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentPowerWorkoutBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class PowerWorkoutFragment : BaseFragment<PowerWorkoutViewModel>() {
    override val viewModel: PowerWorkoutViewModel by viewModels()
    lateinit var binding: FragmentPowerWorkoutBinding
    private val powerWorkoutAdapter =
        PowerWorkoutAdapter(onClickProductListener = { selectedMuscleGroup ->
            viewModel.navigateToExercise(selectedMuscleGroup.keyCategory)
        })
}

```

```
private val args: PowerWorkoutFragmentArgs by navArgs()
```

```
override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    binding = FragmentPowerWorkoutBinding.inflate(inflater, container, false)
    return binding.root
}
```

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding.recycler.adapter = powerWorkoutAdapter
    powerWorkoutAdapter.submitList(args.powerWorkout.typesWorkout)
}
```

```
}
```

//PowerWorkoutScreen

```
import androidx.navigation.NavController
import com.viggo.lightweight.data.Category
```

```
import com.viggo.lightweight.navigation.NavigationComponentScreen
```

```
class PowerWorkoutScreen(private val selectedMuscleGroup: String) : NavigationComponentScreen {
```

```
    override fun execute(navigator: NavController) {
        navigator.navigate(
            PowerWorkoutFragmentDirections.actionPowerWorkoutFragmentToExerciseFragment(
                selectedMuscleGroup
            )
        )
    }
```

```
}
```

```
}
```

//PowerWorkoutViewModel

```
import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.data.Category
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject
```

```
@HiltViewModel
```

```
class PowerWorkoutViewModel @Inject constructor(val router: Router) : BaseViewModel() {

    fun navigateToExercise(selectedMuscleGroup: String) {
        router.execute(command = Command.Navigate(PowerWorkoutScreen(selectedMuscleGroup)))
    }
}
```

```
//MainFragment
```

```
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.viggo.lightweight.base.BaseFragment
import com.viggo.lightweight.databinding.FragmentMainBinding
import dagger.hilt.android.AndroidEntryPoint
```

```
@AndroidEntryPoint
```

```
class MainFragment : BaseFragment<MainFragmentViewModel>() {
    override val viewModel: MainFragmentViewModel by viewModels()
    lateinit var binding: FragmentMainBinding

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentMainBinding.inflate(inflater, container, false)
        return binding.root
    }
}
```

```

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding.icPower.setOnClickListener {
        val powerWorkout = viewModel.createPowerWorkout()
        viewModel.navigateToGroupMuscles(powerWorkout)
    }
    binding.icCardio.setOnClickListener {
        val notPower = viewModel.cardioWorkout()
        viewModel.navigateToExercise(notPower.keyCategory)
    }
    binding.icHIIT.setOnClickListener {
        val notPower = viewModel.HIITWorkout()
        viewModel.navigateToExercise(notPower.keyCategory)
    }
}

}

//MainFragmentScreen
import androidx.navigation.NavController
import com.viggo.lightweight.data.TypeWorkout
import com.viggo.lightweight.navigation.NavigationComponentScreen

class MainFragmentScreen(
    private val type: TypeWorkout? = null,
    private val notPower: String? = "",
    private val to: To

) : NavigationComponentScreen {

    override fun execute(navigator: NavController) {
        when (to) {
            To.EXERCISE_FRAGMENT -> {
                navigator.navigate(MainFragmentDirections.actionMainFragmentToExercise(notPower!!))
            }
            To.POWER_WORKOUT_FRAGMENT -> {
                navigator.navigate(
                    MainFragmentDirections.actionMainFragmentToPowerWorkoutFragment(
                        type!!
                    )
                )
            }
        }
    }
}

```

```

    }

}

enum class To {
    EXERCISE_FRAGMENT, POWER_WORKOUT_FRAGMENT
}

```

```

//MainFragmentViewModel
import com.viggo.lightweight.Constants
import com.viggo.lightweight.base.BaseViewModel
import com.viggo.lightweight.data.Category
import com.viggo.lightweight.data.TypeWorkout
import com.viggo.lightweight.navigation.Command
import com.viggo.lightweight.navigation.Router
import com.viggo.lightweight.utill.StringsProvider
import dagger.hilt.android.lifecycle.HiltViewModel
import javax.inject.Inject

@HiltViewModel
class MainFragmentViewModel @Inject constructor(
    private val router: Router, private val stringsProvider: StringsProvider
) : BaseViewModel() {

    fun createPowerWorkout(): TypeWorkout {
        return TypeWorkout(
            listOf(
                Category(
                    stringsProvider.chest, stringsProvider.chestImg!!, emptyList()
                ),
                Category(
                    stringsProvider.shoulders, stringsProvider.shouldersImg!!, emptyList()
                ),
                Category(
                    stringsProvider.biceps, stringsProvider.bicepsImg!!, emptyList()
                ),
                Category(
                    stringsProvider.triceps, stringsProvider.tricepsImg!!, emptyList()
                ),
                Category(

```

```

        stringsProvider.back, stringsProvider.backImg!!, emptyList()
    ),
    Category(
        stringsProvider.legs, stringsProvider.legsImg!!, emptyList()
    ),
)
)
}

fun cardioWorkout(): Category {
    return Category(Constants.CARDIO, stringsProvider.chestImg!!, emptyList())
}

fun HIITWorkout(): Category {
    return Category(Constants.HIIT, stringsProvider.chestImg!!, emptyList())
}

fun navigateToGroupMuscles(type: TypeWorkout) {
    router.execute(Command.Navigate(MainFragmentScreen(type, "", To.POWER_WORKOUT_FRAGMENT)))
}

fun navigateToExercise(notPowerWorkout: String) {
    router.execute(
        Command.Navigate(
            MainFragmentScreen(
                null, notPowerWorkout, To.EXERCISE_FRAGMENT
            )
        )
    )
}

}
}

```

```

//AndroidManifest
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.viggo.lightweight">

    <uses-permission android:name="android.permission.INTERNET" />

```

```

<application
    android:name=".App"
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@drawable/logo_army"
    android:label="@string/app_name"
    android:roundIcon="@drawable/logo_army"
    android:supportsRtl="true"
    android:theme="@style/Theme.LightWeight"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
        </activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>

    //res

    //anim

    //slide_in_left
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate android:fromXDelta="-50%p" android:toXDelta="0"
        android:duration="@android:integer/config_mediumAnimTime"/>
    <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>

```

//slide_in_right

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate android:fromXDelta="50%p" android:toXDelta="0"
    android:duration="@android:integer/config_mediumAnimTime"/>
  <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
    android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

//slide_out_left

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate android:fromXDelta="0" android:toXDelta="-50%p"
    android:duration="@android:integer/config_mediumAnimTime"/>
  <alpha android:fromAlpha="1.0" android:toAlpha="0.0"
    android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

//slide_out_right

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate android:fromXDelta="0" android:toXDelta="50%p"
    android:duration="@android:integer/config_mediumAnimTime"/>
  <alpha android:fromAlpha="1.0" android:toAlpha="0.0"
    android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

//layout**//activity_main**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/container"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:background="?android:attr/windowBackground"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent">
```

```

app:layout_constraintRight_toRightOf="parent"
app:menu="@menu/bottom_navigation" />

```

```

<fragment
    android:id="@+id/container_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:defaultNavHost="true"
    app:layout_constraintBottom_toTopOf="@id/bottom_navigation"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:navGraph="@navigation/start_graph"
    tools:ignore="FragmentTagUsage" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//fragment_about

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.about.AboutFragment">

```

```

<TextView
    android:id="@+id/back"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginStart="5dp"
    android:layout_marginTop="5dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/version"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/version_app_0_1a"
    android:textColor="@color/black"

```

```

    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/developer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/developer"
    android:textColor="@color/black"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="@id/version"
    app:layout_constraintStart_toStartOf="@id/version"
    app:layout_constraintTop_toBottomOf="@id/version" />

```

```

<TextView
    android:id="@+id/designer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/designer"
    android:textColor="@color/black"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="@id/developer"
    app:layout_constraintStart_toStartOf="@id/developer"
    app:layout_constraintTop_toBottomOf="@id/developer" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//fragment_detailed_info

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black">

```

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".ui.directory.directoryInfo.detailedInfo.DetailedInfoFragment">
```

```
<ImageView
    android:id="@+id/ic_product"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="centerCrop"
    android:src="@drawable/ic_products"
    app:layout_constraintDimensionRatio="1:1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="ContentDescription" />
```

```
<View
    android:id="@+id/ic_circle"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:layout_margin="6dp"
    android:background="@drawable/ic_point"
    app:layout_constraintBottom_toBottomOf="@id/ic_product"
    app:layout_constraintEnd_toEndOf="@id/ic_product" />
```

```
<View
    android:id="@+id/ic_circle_gradient"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:background="@drawable/circle_gradient"
    app:layout_constraintBottom_toBottomOf="@+id/ic_circle"
    app:layout_constraintEnd_toEndOf="@+id/ic_circle"
    app:layout_constraintStart_toStartOf="@+id/ic_circle"
    app:layout_constraintTop_toTopOf="@+id/ic_circle" />
```

```
<TextView
    android:id="@+id/cooking_time"
    android:layout_width="56dp"
    android:layout_height="56dp"
    android:autoSizeTextType="uniform"
    android:gravity="center"
```

```

        android:textColor="@color/black"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@id/ic_circle"
        app:layout_constraintEnd_toEndOf="@id/ic_circle"
        app:layout_constraintStart_toStartOf="@id/ic_circle"
        app:layout_constraintTop_toTopOf="@id/ic_circle"
        tools:text="30 min" />

```

```
<TextView
```

```

        android:id="@+id/txt_ingredients"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:text="@string/ingredients"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/ic_product" />

```

```
<androidx.recyclerview.widget.RecyclerView
```

```

        android:id="@+id/recycler_ingredients"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4sp"
        android:nestedScrollingEnabled="false"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txt_ingredients"
        tools:listitem="@layout/item_ingredients" />

```

```
<TextView
```

```

        android:id="@+id/txt_instruction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:text="@string/instruction"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@id/recycler_instruction"

```

```

        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/recycler_ingredients" />

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_instruction"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="4dp"
    android:nestedScrollingEnabled="false"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txt_instruction"
    tools:listitem="@layout/item_instruction" />

</androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>

//fragment_directory
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_gradient">

    <ImageView
        android:id="@+id/ic_sport_nutrition"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:scaleType="fitXY"
        android:src="@drawable/ic_sport_nutrients"
        app:layout_constraintBottom_toTopOf="@id/ic_general_tips"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_weight="1" />

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"

```

```

    android:gravity="center"
    android:text="Спортивне харчування"
    android:textColor="@color/white"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_sport_nutrition"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/ic_sport_nutrition" />

```

```
<ImageView
```

```

    android:id="@+id/ic_general_tips"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_general_tips"
    app:layout_constraintBottom_toTopOf="@id/ic_product"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/ic_sport_nutrition"
    app:layout_constraintVertical_weight="1" />

```

```
<TextView
```

```

    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Загальні поради"
    android:textColor="#FFFFFF"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_general_tips"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/ic_general_tips" />

```

```
<ImageView
```

```

    android:id="@+id/ic_product"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_products"
    app:layout_constraintBottom_toBottomOf="parent"

```

```

app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/ic_general_tips"
app:layout_constraintVertical_weight="1" />

```

```
<TextView
```

```

    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Продукти і харчування"
    android:textColor="@color/white"
    android:textSize="35sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_product"
    app:layout_constraintEnd_toEndOf="@+id/ic_product"
    app:layout_constraintStart_toStartOf="@+id/ic_product"
    app:layout_constraintTop_toTopOf="@+id/ic_product" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

//fragment_directory_info

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:background="@color/black"
```

```
    tools:context=".ui.directory.directoryInfo.DirectoryInfoFragment">
```

```
<androidx.recyclerview.widget.RecyclerView
```

```
    android:id="@+id/recycler"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

```
    tools:listitem="@layout/item_power_workout" />
```

```
<ProgressBar
```

```
    android:id="@+id/progress"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```

        android:layout_marginTop="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:visibility="gone"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//fragment_exercise
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```
<TextView
```

```

    android:id="@+id/name_group"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="15dp"
    android:textColor="@color/black"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```
<androidx.recyclerview.widget.RecyclerView
```

```

    android:id="@+id/recycler"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="12dp"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    app:layout_constraintTop_toBottomOf="@id/name_group"
    tools:listitem="@layout/item_exercise" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//fragment_main
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

```

```

android:layout_width="match_parent"
android:layout_height="match_parent">

```

```

<ImageView
    android:id="@+id/ic_power"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_power"
    app:layout_constraintBottom_toTopOf="@id/ic_cardio"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_weight="1" />

```

```

<TextView
    android:id="@+id/txt"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="СИЛОВЕ"
    android:textColor="@color/white"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_power"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/ic_power" />

```

```

<ImageView
    android:id="@+id/ic_cardio"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_cardio"
    app:layout_constraintBottom_toTopOf="@id/ic_HIIT"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/ic_power"
    app:layout_constraintVertical_weight="1" />

```

```

<TextView

```

```

    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Кардио"
    android:textColor="@color/white"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_cardio"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/ic_cardio" />

```

```
<ImageView
```

```

    android:id="@+id/ic_HIIT"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_hiit"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/ic_cardio"
    app:layout_constraintVertical_weight="1" />

```

```
<TextView
```

```

    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Високо-інтенсивне"
    android:textColor="@color/white"
    android:textSize="35sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_HIIT"
    app:layout_constraintEnd_toEndOf="@+id/ic_HIIT"
    app:layout_constraintStart_toStartOf="@+id/ic_HIIT"
    app:layout_constraintTop_toTopOf="@+id/ic_HIIT" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
    //fragment_measurements
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```

xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".ui.measurements.MeasurementsFragment">

```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/previos_measurements"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/old_txt_waist"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="5dp"
    android:layout_marginTop="50dp"
    android:text="@string/waist"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<EditText
    android:id="@+id/old_edt_waist"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/old_txt_waist" />

```

```

<TextView
    android:id="@+id/old_txt_hip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/hip"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_chest"

```

```

app:layout_constraintStart_toEndOf="@id/old_txt_waist"
app:layout_constraintTop_toTopOf="parent" />

```

```
<EditText
```

```

    android:id="@+id/old_edt_hip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_hip"
    app:layout_constraintTop_toBottomOf="@id/old_txt_hip" />

```

```
<TextView
```

```

    android:id="@+id/old_txt_chest"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/chest"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_biceps"
    app:layout_constraintStart_toEndOf="@id/old_txt_hip"
    app:layout_constraintTop_toTopOf="parent" />

```

```
<EditText
```

```

    android:id="@+id/old_edt_chest"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_chest"
    app:layout_constraintTop_toBottomOf="@id/old_txt_chest" />

```

```
<TextView
```

```

    android:id="@+id/old_txt_biceps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/biceps"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_buttocks"
    app:layout_constraintStart_toEndOf="@id/old_txt_chest"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<EditText
    android:id="@+id/old_edt_biceps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_biceps"
    app:layout_constraintTop_toBottomOf="@id/old_txt_waist" />

```

```

<TextView
    android:id="@+id/old_txt_buttocks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/buttocks"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_calves"
    app:layout_constraintStart_toEndOf="@id/old_txt_biceps"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<EditText
    android:id="@+id/old_edt_buttocks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_buttocks"
    app:layout_constraintTop_toBottomOf="@id/old_txt_waist" />

```

```

<TextView
    android:id="@+id/old_txt_calves"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/calves"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@id/old_txt_buttocks"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<EditText

```

```

    android:id="@+id/old_edt_calves"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_calves"
    app:layout_constraintTop_toBottomOf="@id/old_txt_waist" />

```

```
<TextView
```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="15dp"
    android:text="@string/current_measurements"
    android:textColor="@color/black"
    app:layout_constraintBottom_toTopOf="@id/new_txt_waist"
    app:layout_constraintStart_toStartOf="parent" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_waist"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="5dp"
    android:layout_marginTop="50dp"
    android:text="@string/waist"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```
<EditText
```

```

    android:id="@+id/new_edt_waist"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/new_txt_waist" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_hip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

    android:layout_marginTop="50dp"
    android:text="@string/hip"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_chest"
    app:layout_constraintStart_toEndOf="@id/old_txt_waist"
    app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```
<EditText
```

```

    android:id="@+id/new_edt_hip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_hip"
    app:layout_constraintTop_toBottomOf="@id/new_txt_hip" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_chest"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/chest"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_biceps"
    app:layout_constraintStart_toEndOf="@id/old_txt_hip"
    app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```
<EditText
```

```

    android:id="@+id/new_edt_chest"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_chest"
    app:layout_constraintTop_toBottomOf="@id/new_txt_chest" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_biceps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/biceps"

```

```

    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_buttocks"
    app:layout_constraintStart_toEndOf="@id/old_txt_chest"
    app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```
<EditText
```

```

    android:id="@+id/new_edt_biceps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_biceps"
    app:layout_constraintTop_toBottomOf="@id/new_txt_waist" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_buttocks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/buttocks"
    android:textColor="@color/black"
    app:layout_constraintEnd_toStartOf="@id/old_txt_calves"
    app:layout_constraintStart_toEndOf="@id/old_txt_biceps"
    app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```
<EditText
```

```

    android:id="@+id/new_edt_buttocks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_buttocks"
    app:layout_constraintTop_toBottomOf="@id/new_txt_waist" />

```

```
<TextView
```

```

    android:id="@+id/new_txt_calves"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/calves"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="parent"

```

```

app:layout_constraintStart_toEndOf="@id/old_txt_buttocks"
app:layout_constraintTop_toBottomOf="@id/old_edt_waist" />

```

```

<EditText
    android:id="@+id/new_edt_calves"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:maxLines="1"
    app:layout_constraintStart_toStartOf="@id/old_txt_calves"
    app:layout_constraintTop_toBottomOf="@id/new_txt_waist" />

```

```

<TextView
    android:id="@+id/save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:text="Save"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/new_edt_calves" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//fragment_power_workout

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.workout.power.PowerWorkoutFragment">

```

```

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler"
    android:layout_width="match_parent"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    android:layout_height="match_parent"
    tools:listitem="@layout/item_power_workout" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//fragment_profile
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white">
```

```
<TextView
    android:id="@+id/txt_back"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginStart="5dp"
    android:layout_marginTop="5dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:backgroundTint="@color/white"
    android:gravity="center"
    android:hint="Имя"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/age"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
```

```

android:backgroundTint="@color/white"
android:gravity="center"
android:hint="Bik"
android:textColor="@color/black"
android:textColorHint="@color/black"
android:textSize="18sp"
android:textStyle="bold"
app:layout_constraintEnd_toEndOf="@id/name"
app:layout_constraintStart_toStartOf="@id/name"
app:layout_constraintTop_toBottomOf="@id/name" />

```

<EditText

```

android:id="@+id/gender"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="40dp"
android:backgroundTint="@color/white"
android:gravity="center"
android:hint="Стать"
android:textColor="@color/black"
android:textColorHint="@color/black"
android:textSize="18sp"
android:textStyle="bold"
app:layout_constraintEnd_toEndOf="@id/age"
app:layout_constraintStart_toStartOf="@id/age"
app:layout_constraintTop_toBottomOf="@id/age" />

```

<EditText

```

android:id="@+id/weight"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="40dp"
android:backgroundTint="@color/white"
android:gravity="center"
android:hint="Поточна вара"
android:textColor="@color/black"
android:textColorHint="@color/black"
android:textSize="18sp"
android:textStyle="bold"
app:layout_constraintEnd_toEndOf="@id/gender"
app:layout_constraintStart_toStartOf="@id/gender"
app:layout_constraintTop_toBottomOf="@id/gender" />

```

```

<Button
    android:id="@+id/save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20sp"
    android:text="Зберегти"
    android:textAllCaps="false"
    android:textColor="@color/black"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="@id/weight"
    app:layout_constraintStart_toStartOf="@id/weight"
    app:layout_constraintTop_toBottomOf="@id/weight" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//fragment_settings

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white">

```

```

<TextView
    android:id="@+id/profile"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="80dp"
    android:text="Профіль"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toEndOf="@id/ic_profile"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<ImageView
    android:id="@+id/ic_profile"
    android:layout_width="40dp"

```

```

    android:layout_height="40dp"
    android:layout_marginStart="45dp"
    android:layout_marginEnd="15dp"
    android:src="@drawable/profile"
    app:layout_constraintBottom_toBottomOf="@id/profile"
    app:layout_constraintEnd_toStartOf="@id/profile"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@id/profile" />

```

```
<TextView
```

```

    android:id="@+id/about_app"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:text="Про застосунок"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toEndOf="@id/ic_about"
    app:layout_constraintStart_toStartOf="@id/profile"
    app:layout_constraintTop_toBottomOf="@id/profile" />

```

```
<ImageView
```

```

    android:id="@+id/ic_about"
    android:layout_width="35dp"
    android:layout_height="35dp"
    android:layout_marginEnd="15dp"
    android:src="@drawable/settings"
    app:layout_constraintBottom_toBottomOf="@id/about_app"
    app:layout_constraintStart_toStartOf="@id/ic_profile"
    app:layout_constraintTop_toTopOf="@id/about_app" />

```

```
<TextView
```

```

    android:id="@+id/measurements"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:text="Виміри власного тіла"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toEndOf="@id/ic_measurements"

```

```

app:layout_constraintStart_toStartOf="@id/about_app"
app:layout_constraintTop_toBottomOf="@id/about_app" />

```

```

<ImageView
    android:id="@+id/ic_measurements"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_marginEnd="15dp"
    android:src="@drawable/meter"
    app:layout_constraintBottom_toBottomOf="@id/measurements"
    app:layout_constraintStart_toStartOf="@id/ic_about"
    app:layout_constraintTop_toTopOf="@id/measurements" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//fragment_splash

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".ui.splash.SplashFragment">

```

```

<ImageView
    android:id="@+id/image_splash"
    android:layout_width="300dp"
    android:layout_height="300dp"
    android:layout_marginTop="70dp"
    android:src="@drawable/logo_army"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/name_app"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"

```

```

    android:text="@string/app_name"
    android:textColor="@color/white"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/image_splash" />

```

```

<ProgressBar
    android:id="@+id/progress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="50dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/name_app" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

//item_directory_info

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

```

```

<ImageView
    android:id="@+id/img_arrow"
    android:layout_width="25dp"
    android:layout_height="25dp"
    android:layout_marginStart="5dp"
    android:src="@drawable/ic_arrow_down"
    app:layout_constraintBottom_toBottomOf="@id/name"
    app:layout_constraintStart_toEndOf="@id/name"
    app:layout_constraintTop_toTopOf="@id/name" />

```

```

<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

    android:layout_marginTop="25dp"
    android:text="BCAA"
    android:textColor="@color/white"
    android:textSize="17sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/description"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:textColor="@color/white"
    android:visibility="gone"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/name" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

//item_exercise

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

```

```

<ImageView
    android:id="@+id/ic_exercise"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_products"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="ContentDescription" />

```

```

<View

```

```

    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:background="@drawable/background_gradient"
    app:layout_constraintBottom_toBottomOf="@id/ic_exercise"
    app:layout_constraintTop_toTopOf="@id/ic_exercise" />

```

```
<TextView
```

```

    android:id="@+id/name_exercise"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:textColor="@color/white"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/ic_exercise"
    app:layout_constraintEnd_toStartOf="@+id/img_arrow"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />

```

```
<ImageView
```

```

    android:id="@+id/img_arrow"
    android:layout_width="25dp"
    android:layout_height="25dp"
    android:layout_margin="5dp"
    android:src="@drawable/arrow_down"
    app:layout_constraintBottom_toBottomOf="@id/ic_exercise"
    app:layout_constraintEnd_toEndOf="parent"
    tools:ignore="ContentDescription" />

```

```
<TextView
```

```

    android:id="@+id/description_exercise"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:textColor="@color/black"
    android:visibility="gone"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/name_exercise" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//item_ingredients
```

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="6dp"
    tools:background="@color/black">

    <View
        android:id="@+id/point"
        android:layout_width="6dp"
        android:layout_height="6dp"
        android:background="@drawable/ic_point"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/ingredient_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="6dp"
        android:textColor="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/point"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

//item_instruction

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="6dp"
    tools:background="@color/black">

    <TextView
        android:id="@+id/num_step"

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Step 1" />

```

```
<TextView
```

```

    android:id="@+id/instruction"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginVertical="3dp"
    android:textColor="@color/white"
    android:textSize="16sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/num_step" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//item_nutrition
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

```

```
<TextView
```

```

    android:id="@+id/name_product"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@color/white"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/icProduct"
    app:layout_constraintTop_toTopOf="@+id/icProduct" />

```

```

<ImageView
    android:id="@+id/icProduct"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:layout_margin="5dp"
    android:scaleType="centerCrop"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="ContentDescription" />

<TextView
    android:id="@+id/calories"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="3dp"
    android:textColor="@color/white"
    app:layout_constraintBottom_toTopOf="@id/protein"
    app:layout_constraintStart_toEndOf="@+id/icProduct"
    app:layout_constraintTop_toBottomOf="@+id/name_product" />

<TextView
    android:id="@+id/protein"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginVertical="3dp"
    android:textColor="@color/white"
    app:layout_constraintBottom_toTopOf="@id/carbohydrates"
    app:layout_constraintStart_toEndOf="@+id/icProduct"
    app:layout_constraintTop_toBottomOf="@+id/calories" />

<TextView
    android:id="@+id/carbohydrates"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginVertical="3dp"
    android:textColor="@color/white"
    app:layout_constraintBottom_toTopOf="@id/fat"
    app:layout_constraintStart_toEndOf="@+id/icProduct"

```

```
app:layout_constraintTop_toBottomOf="@+id/protein" />
```

```
<TextView
```

```
    android:id="@+id/fat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginVertical="3dp"
    android:textColor="@color/white"
    app:layout_constraintBottom_toTopOf="@id/sugar"
    app:layout_constraintStart_toEndOf="@+id/icProduct"
    app:layout_constraintTop_toBottomOf="@+id/carbohydrates" />
```

```
<TextView
```

```
    android:id="@+id/sugar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginVertical="3dp"
    android:textColor="@color/white"
    app:layout_constraintBottom_toBottomOf="@+id/icProduct"
    app:layout_constraintStart_toEndOf="@+id/icProduct"
    app:layout_constraintTop_toBottomOf="@+id/fat" />
```

```
<View
```

```
    android:layout_width="0dp"
    android:layout_height="1dp"
    android:layout_marginHorizontal="20dp"
    android:layout_marginVertical="3dp"
    android:background="@color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/icProduct" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
    //item_power_workout
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
```

```
android:layout_height="wrap_content">
```

```
<ImageView
```

```
    android:id="@+id/image"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:scaleType="fitXY"
    android:src="@drawable/ic_sport_nutrients"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_weight="1" />
```

```
<TextView
```

```
    android:id="@+id/category"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textColor="@color/white"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/image"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/image" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
//menu
```

```
//bottom_navigation
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto">
```

```
    <item
```

```
        android:id="@id/mainFragment"
        android:icon="@drawable/logo_army"
        android:title="Тренування"
        app:showAsAction="withText|ifRoom" />
```

```

<item
    android:id="@id/directoryFragment"
    android:icon="@drawable/directory"
    android:title="Каталог"
    app:showAsAction="withText|ifRoom" />

<item
    android:id="@id/settingsFragment"
    android:icon="@drawable/settings"
    android:title="Налаштування"
    app:showAsAction="withText|ifRoom" />

</menu>

//navigation

//start_graph
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/start_graph"
    app:startDestination="@id/splashFragment">

    <!-- Splash-->
    <fragment
        android:id="@+id/splashFragment"
        android:name="com.viggo.lightweight.ui.splash.SplashFragment"
        android:label="SplashFragment"
        tools:layout="@layout/fragment_splash">
        <action
            android:id="@+id/action_splashFragment_to_mainFragment"
            app:destination="@id/mainFragment"
            app:enterAnim="@anim/slide_in_right"
            app:exitAnim="@anim/slide_out_left"
            app:popEnterAnim="@anim/slide_in_left"
            app:popExitAnim="@anim/slide_out_right" />
        </fragment>

    <!-- Main -->

    <fragment

```

```

    android:id="@+id/mainFragment"
    android:name="com.viggo.lightweight.ui.workout.MainFragment"
    android:label="MainFragment"
    tools:layout="@layout/fragment_main">
    <action
        android:id="@+id/action_mainFragment_to_PowerWorkoutFragment"
        app:destination="@id/powerWorkout"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
    <action
        android:id="@+id/action_maintFragment_to_Exercise"
        app:destination="@id/exerciseFragment"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
</fragment>

<!-- Settings -->

<fragment
    android:id="@+id/settingsFragment"
    android:name="com.viggo.lightweight.ui.settings.SettingsFragment"
    android:label="Settings"
    tools:layout="@layout/fragment_settings">
    <action
        android:id="@+id/action_settingsFragment_to_aboutApp"
        app:destination="@id/about_app"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
    <action
        android:id="@+id/action_settingsFragment_to_Measurements"
        app:destination="@id/measurements"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
    <action

```

```

        android:id="@+id/action_settingsFragment_to_profileFragment"
        app:destination="@id/profile"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />

</fragment>

<fragment
    android:id="@+id/profile"
    android:name="com.viggo.lightweight.ui.profile.ProfileFragment"
    android:label="Profile"
    tools:layout="@layout/fragment_profile" />

<fragment
    android:id="@+id/about_app"
    android:name="com.viggo.lightweight.ui.about.AboutFragment"
    android:label="AboutApp"
    tools:layout="@layout/fragment_about" />

<fragment
    android:id="@+id/measurements"
    android:name="com.viggo.lightweight.ui.measurements.MeasurementsFragment"
    android:label="AboutApp"
    tools:layout="@layout/fragment_about" />

<!-- Directory -->

<fragment
    android:id="@+id/directoryFragment"
    android:name="com.viggo.lightweight.ui.directory.DirectoryFragment"
    android:label="Directory"
    tools:layout="@layout/fragment_directory">
    <action
        android:id="@+id/action_directoryFragment_to_directoryInfoFragment"
        app:destination="@id/directoryInfo"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />

```

```
</fragment>
```

```
<!-- PowerWorkout -->
```

```
<fragment
```

```
    android:id="@+id/powerWorkout"
    android:name="com.viggo.lightweight.ui.workout.power.PowerWorkoutFragment"
    android:label="PowerWorkout"
    tools:layout="@layout/fragment_power_workout">
    <action
        android:id="@+id/action_powerWorkoutFragment_to_exerciseFragment"
        app:destination="@id/exerciseFragment"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
    <argument
        android:name="powerWorkout"
        app:argType="com.viggo.lightweight.data.TypeWorkout" />
</fragment>
```

```
<!-- DirectoryInfo -->
```

```
<fragment
```

```
    android:id="@+id/directoryInfo"
    android:name="com.viggo.lightweight.ui.directory.directoryInfo.DirectoryInfoFragment"
    android:label="DirectoryInfo"
    tools:layout="@layout/fragment_directory_info">
    <argument
        android:name="type"
        app:argType="string" />
    <action
        android:id="@+id/action_directoryInfo_to_detailedInfo"
        app:destination="@id/detailedInfo"
        app:enterAnim="@anim/slide_in_right"
        app:exitAnim="@anim/slide_out_left"
        app:popEnterAnim="@anim/slide_in_left"
        app:popExitAnim="@anim/slide_out_right" />
</fragment>
```

```
<fragment
```

```
    android:id="@+id/exerciseFragment"
```

```

        android:name="com.viggo.lightweight.ui.workout.exercise.ExerciseFragment"
        android:label="ExerciseFragment"
        tools:layout="@layout/fragment_exercise">
        <argument
            android:name="selectedMuscleGroup"
            app:argType="string" />
    </fragment>

    <fragment
        android:id="@+id/detailedInfo"
        android:name="com.viggo.lightweight.ui.directory.directoryInfo.detailedInfo.DetailedInfoFragment"
        android:label="DetailedInfo"
        tools:layout="@layout/fragment_detailed_info">
        <argument
            android:name="food"
            app:argType="com.viggo.lightweight.data.food.FoodItem" />
    </fragment>

</navigation>

```

//strings

```

<resources>
    <string name="app_name">Light Weight</string>

    <!-- Name of Exercise for arms -->
    <string name="hammer">Молотки</string>
    <string name="barbell_lift_for_biceps">Підйом штанги на біцепс</string>
    <string name="bending_arms_with_dumbbells">Згинання рук з гантелями</string>
    <string name="block_pull_for_triceps">тяга блоку на трицепс</string>
    <string name="french_triceps_press">французький жим на трицепс</string>
    <string name="Lifting_dumbbells_on_the_forearm">Підйом гантелей на передпліччі</string>
    <string name="extension_of_arms_on_the_simulator_for_triceps">розгинання рук на тренажері для
трицепса</string>

    <!-- Name of Exercise for legs -->

    <string name="barbell_squats">присідання зі штангою</string>
    <string name="breeding_legs_in_the_simulator">розведення ніг в тренажері</string>
    <string name="reduction_of_legs_in_the_simulator">зведення ніг в тренажері</string>
    <string name="leg_press_in_the_simulator">жим ногами в тренажері</string>
    <string name="leg_extension_in_the_simulator">розгинання ніг в тренажері</string>

```

```

<string name="bending_the_legs_in_the_simulator">згинання ніг в тренажері</string>

<!-- Name of Exercise for pectoral muscles -->

<string name="bench_press">жим лежачи</string>
<string name="dumbbell_bench_press_on_a_horizontal_bench">жим гантелей на горизонтальній лаві</string>
<string name="breeding_dumbbells_on_an_incline_bench">розведення гантелей на похилій лаві</string>
<string name="incline_dumbbell_bench_press">жим гантелей в нахилі</string>

<!-- Name of Exercise for back -->

<string name="pulldown_of_the_upper_block_to_the_chest">притягування верхнього блоку до грудей</string>
<string name="dumbbell_pull_to_the_belt">тяга гантелі до пояса</string>
<string name="deadlift">Станова тяга</string>
<string name="hyperextension_on_the_bench">гіперекстензія на лаві</string>
<string name="T_bar_row_on_the_block">Тяга T-bar штанги</string>

<!-- Name of Exercise for shoulders -->

<string name="standing_dumbbell_press">жим гантелей стоячи</string>
<string name="standing_barbell_press">жим штанги стоячи</string>
<string name="lifting_dumbbells_to_the_sides">розведення гантелей в сторони</string>
<string name="lifting_dumbbells_in_front_of_you">піднімання гантелей перед собою</string>
<!-- TODO: Remove or change this placeholder text -->
<string name="hello_blank_fragment">Hello blank fragment</string>
<string name="version_app_0_1a">Версія програми: 0.1A</string>

<!-- Sport nutrition -->

<string name="creatine_meaning">Креатин - це органічна речовина, яка міститься в м'язах і відіграє важливу роль у виробництві енергії для м'язових скорочень. Його отримують з амінокислот, які містяться в білковій їжі. Креатин використовується як добавка для підвищення спортивних результатів і збільшення м'язової маси. Він допомагає підвищити рівень фосфокреатину в м'язах, що покращує здатність швидко виконувати вправи високої інтенсивності. Зазвичай креатин приймають у вигляді порошку, який змішують з водою або іншою рідиною. Дозування зазвичай становить від 3 до 5 грамів на день. Рекомендується приймати креатин після тренування разом з вуглеводами, щоб поліпшити його засвоєння. Однак перед початком прийому креатину рекомендується проконсультуватися з лікарем або фахівцем в області спортивного харчування.</string>

```

<string name="protein_meaning">Білок (протеїн) є одним з основних поживних речовин, необхідних для здорового функціонування організму. Він відіграє важливу роль у зростанні та відновленні тканин, забезпечує підтримку імунної системи та бере участь у процесах метаболізму та транспортування кисню в крові. Білок можна отримати з різних джерел, таких як м'ясо, риба, яйця, молочні продукти, горіхи та бобові. Крім того, протеїн також продається у формі спеціальних порошків і батончиків для спортсменів і любителів фітнесу. Щоб прийняти протеїновий порошок, зазвичай його необхідно змішати з водою, молоком або коктейлем і випити. Дозування протеїну може змінюватися залежно від цілей і потреб людини, але зазвичай рекомендується споживати від 1 до 2 грамів на кілограм ваги тіла. Однак перед початком прийому протеїну рекомендується проконсультуватися з лікарем або фахівцем зі спортивного харчування.</string>

<string name="L_carnitin_meaning">L-карнітин - це амінокислота, яка бере участь в обміні жирів в організмі. Він допомагає транспортувати жирні кислоти до мітохондрій, де вони використовуються в процесі виробництва енергії. L-карнітин може бути корисним для тих, хто займається спортом або стежить за своєю вагою, оскільки він допомагає прискорити метаболізм жирів і збільшити витривалість під час фізичних навантажень. Крім того, деякі дослідження показують, що L-карнітин може покращити роботу серцево-судинної системи та зменшити втоми. Рекомендована добова доза L-карнітину становить 300 мг. Однак при схудненні дозу слід збільшити і приймати жиросжигатель по 500 або 750 мг тричі на день або по 1000 мг двічі на день. Вживати більше 2 г на добу безглуздо. Однак перед початком споживання L-карнітину рекомендується проконсультуватися з лікарем або спеціалістом зі спортивного харчування.</string>

<string name="glutamine_meaning">Глютамін - це амінокислота, яка є важливим будівельним блоком білків в організмі. Це одна з 20 незамінних амінокислот, які використовуються в процесі синтезу білка. Глютамін відіграє ключову роль у різних процесах організму, включаючи здоров'я імунної системи, роботу кишечника та відновлення м'язів після фізичних вправ. Глютамін можна приймати як дієтичну їжу добавки або у формі порошку та капсул. Дозування зазвичай залежить від індивідуальних потреб організму та мети прийому. Рекомендується дотримуватись інструкцій на упаковці або проконсультуватися з лікарем або дієтологом перед прийомом глютаміну.</string>

<string name="BCAA_meaning">BCAA (від англійського амінокислоти з розгалуженим ланцюгом) - це суміш трьох амінокислот: лейцину, ізолейцину та валіну. Вони є важливими будівельними блоками білків в організмі та беруть участь у регуляції метаболізму м'язової тканини. BCAA необхідні для підтримки здоров'я та росту м'язової тканини, зменшення втоми та підвищення витривалості під час тренувань, а також прискорення відновлення після тренувань. Це також може допомогти зменшити швидкість катаболізму (розпаду) м'язової тканини під час інтенсивних тренувань або дієт. BCAA можна приймати у формі порошку або капсул. Дозування зазвичай залежить від індивідуальних потреб організму та мети прийому. Рекомендується дотримуватись інструкцій на упаковці або проконсультуватися з лікарем або дієтологом перед прийомом BCAA.</string>

<string name="gainer_meaning">Гейнер – це дієтична добавка, яка містить білки і швидкозасвоювані вуглеводи і, таким чином, забезпечує необхідне харчування для людей, які займаються активними видами

спорту, дозволяючи худим спортсменам набирати м'язову масу. Гейнери не тільки забезпечують організм збалансованим співвідношенням вуглеводів і білка, але і дозволяють розвантажити травну систему, крім того, забезпечують м'язи енергією, сприяють зростанню м'язової маси, підвищують ефективність тренувального процесу, сприяють оптимальному відновленню. Гейнер для набору ваги можна розбавляти будь-якою рідиною (вода, сік, молоко), за винятком гарячої води, через високу ймовірність згорання білка. Добову дозу необхідно приймати в декілька прийомів, обов'язково після тренування, що дозволяє зменшити ефект катаболізму. Схема прийому визначається режимом тренувань. У вільні дні приймати добавку потрібно двічі – вранці і вдень, але не перед сном. У дні занять в тренажерному залі продукт приймають вранці, за півгодини до і після тренування. Гейнер для набору ваги необхідний людям худорлявої статури. Завдяки їх хорошему метаболізму не відбувається збільшення м'язових волокон, а прийом гейнера насичує організм легкозасвоюваними вуглеводами і відсутніми калоріями. Однак без силових навантажень ефекту не буде, також необхідно стежити за достатньою кількістю білкової їжі з розрахунку 2–3 грами чистого білка на кілограм ваги. Розрахунок дозування проводиться на кожні 10 кг ваги, тому чоловікам при чотирьох разових основних прийомах їжі необхідно від 14 до 16 грамів суміші. Жінкам при подібному режимі прийому основної їжі достатньо до 14 грамів продукту. Прийом гейнера для набору м'язової маси без занять спортом може призвести до утворення значних прошарків жиру.

<string name="creatine">Creatine</string>

<string name="protein">Protein</string>

<string name="BCAA">BCAA</string>

<string name="L_carnitine">L-carnitine</string>

<string name="glutamine">Glutamine</string>

<string name="gainer">Gainer</string>

<!-- PFC -->

<string name="proteinF">Білок: %1s g</string>

<string name="fat">Жирів: %1s g</string>

<string name="carbohydrates">Вуглеводів: %1s g</string>

<string name="sugar">Цукру: %1s g</string>

<string name="calories">Калорій: %1s kcal</string>

<string name="step">Шаг: %1s</string>

<!-- Type of info -->

<string name="sport_nutrition">Спортивне Харчування</string>

<string name="general_tips">Загальні поради</string>

<string name="food_and_nutrition">Продукти і харчування</string>

<!-- General tips -->

<string name="gain_muscles">Напощення м'язів</string>

<string name="weight_loss">Втрата ваги</string>

<string name="improve_power">Покращення сили</string>

<string name="increase_stamina">Підвищення витривалості</string>

<string name="gain_muscles_tip">Харчування: Правильне харчування є одним з найважливіших факторів набору м'язової маси. Ви повинні споживати достатню кількість білків, вуглеводів і жирів, щоб забезпечити свій організм необхідними поживними речовинами для росту м'язів. Рекомендована кількість білка становить приблизно 1–1,5 г на 1 кг ваги на день, тоді як вуглеводів має становити 2–3 г на 1 кг ваги на день. Тренування: щоб набрати м'язову масу, необхідно регулярно займатися спортом, не менше 3–4 разів на тиждень. Тренування повинні включати вправи, спрямовані на всі групи м'язів. Почніть з базових вправ, таких як присідання, жими та підтягування, потім ви можете додати ізолюючі вправи, такі як різні скручування. Режим: Відпочинок між тренуваннями також дуже важливий. М'язам потрібен час, щоб відновитися і вирости, тому потрібно дати їм достатньо часу для відновлення. Рекомендується відпочивати 48 годин між тренуваннями однієї і тієї ж групи м'язів. Збільшення навантаження: Щоб стимулювати ріст м'язів, необхідно поступово збільшувати навантаження. Це можна зробити, збільшивши вагу, кількість повторень або зменшивши час відпочинку між вправами. Моніторинг: важливо відстежувати свій прогрес, записуючи тренування та результати. Це допоможе вам відстежувати свій прогрес і коригувати свої тренування та харчування для отримання максимальних результатів. Важливо пам'ятати, що набір м'язової маси – це процес, який вимагає часу, терпіння та наполегливості. Необхідно бути постійним і наполегливим у своїх зусиллях, щоб досягти бажаних результатів.</string>

<string name="weight_loss_tip">Схуднути правильно - не означає голодувати і обмежувати себе в їжі. Це процес, який має базуватися на здоровому способі життя та змінах у дієті та фізичній активності. Ось кілька порад щодо правильного схуднення: Харчування: зменшіть споживання калорій, щоб створити дефіцит калорій. Однак не моріть себе голодом і не виключайте повністю певні групи продуктів. Намагайтеся харчуватися збалансовано, з достатньою кількістю білків, вуглеводів, жирів, вітамінів і мінералів. У той же час їжте більше овочів і фруктів, а також обмежте споживання жирної та солодкої їжі. Фізична активність - додайте фізичні вправи до свого розпорядку дня. Це може бути будь-яка вправа, яка вам подобається і яку ви можете виконувати регулярно. Наприклад, це може бути біг, ходьба, їзда на велосипеді, плавання, йога або силові тренування. Намагайтеся займатися спортом не менше 30 хвилин на день, не менше 5 разів на тиждень. Режим: Відпочинок і сон. Нестача сну може призвести до підвищення рівня гормону голоду, що може сприяти переїданню та зниженню фізичної активності. Рекомендується спати 7–8 годин на добу. Моніторинг: відстежуйте свій прогрес і результати, записуючи свою вагу та щоденні досягнення. Це допоможе вам відстежувати свій прогрес і коригувати свої тренування та харчування для отримання максимальних результатів. Поступово: не прагніть до швидких результатів. Втрата ваги має бути поступовою та стабільною, щоб ваше тіло могло адаптуватися до змін і уникнути повернення втраченої ваги.</string>

<string name="improve_power_tip">Якщо ви хочете підвищити свої силові показники, вам необхідно систематично займатися тренуваннями з обтяженнями. Ось кілька порад, які можуть вам у цьому допомогти: Поступове збільшення ваги: щоб збільшити силу, необхідно поступово збільшувати навантаження на м'язи. Почніть з невеликих ваг і поступово збільшуйте вагу в міру зростання сили та здібностей. Тренування високої інтенсивності: щоб розвинути силу, вам потрібно тренуватися з високою інтенсивністю. Це означає використання об'єктів, які дозволяють виконувати від 6 до 10 повторень за

підхід. Регулярні тренування: щоб збільшити свою силу, вам потрібно тренуватися регулярно, принаймні два-три рази на тиждень. Це допоможе вашим м'язам адаптуватися до навантаження та розвиватися. Вибір правильних вправ: вибирайте вправи, які є найбільш ефективними для розвитку сили. Це можуть бути базові вправи з використанням штанги і гантелей, такі як жим лежачи, присідання, станова тяга, підтягування і т. д. Правильне харчування: для розвитку і збільшення силових показників необхідно отримувати достатню кількість білків, вуглеводів і жирів. Ваш раціон має містити високоякісні джерела білка, такі як м'ясо, риба, яйця, молочні продукти, горіхи та бобові. Відпочинок і відновлення: відпочинок і відновлення також важливі для розвитку сили. Дайте своїм м'язам достатньо часу для відновлення між тренуваннями та приділяйте достатньо уваги сну та харчуванню. Постійний прогрес: намагайтеся постійно покращувати свої результати, збільшуючи вагу та кількість повторень у вправах. Відстежуйте свій прогрес і коригуйте свої тренування

<string name="increase_stamina_tip">Якщо ви хочете підвищити свою витривалість, вам необхідно займатися регулярними кардіотренуваннями, які дозволять організму адаптуватися до підвищених навантажень і поліпшать роботу серцево-судинної системи. Ось кілька порад, які допоможуть вам збільшити свою витривалість: Регулярні фізичні вправи: щоб збільшити свою витривалість, вам потрібно регулярно займатися спортом, принаймні три-чотири рази на тиждень. Це допоможе вашому організму адаптуватися до підвищеного навантаження та покращить роботу серцево-судинної системи. Інтервальні тренування: інтервальні тренування, під час яких ви чергуєте високоінтенсивні вправи з періодами відпочинку, дуже ефективні для розвитку витривалості. Вибір правильної вправи: Для підвищення витривалості найкраще підходять такі кардіотренування, як біг, їзда на велосипеді, плавання, скандинавська ходьба тощо. Виберіть вправи, які вам подобаються і які ви можете виконувати регулярно. Збільшення тривалості ваших тренувань: щоб збільшити свою витривалість, поступово збільшуйте тривалість ваших кардіотренувань. Почніть з 20–30 хвилин і поступово збільшуйте до 45–60 хвилин. Правильна дієта: щоб підвищити свою витривалість, вам необхідно отримувати достатню кількість вуглеводів, які є джерелом енергії для вашого організму. Включіть у свій раціон складні вуглеводи, такі як фрукти, овочі, злаки тощо. Відпочинок і відновлення: відпочинок і відновлення також дуже важливі для розвитку витривалості. Дайте своєму тілу достатньо часу для відновлення між тренуваннями, приділіть достатньо уваги сну та харчуванню.</string>

<!-- Group Muscles -->

<string name="cardio">Кардіо</string>

<string name="hiit">Високо-інтенсивне</string>

<string name="chest">Груди</string>

<string name="legs">Ноги</string>

<string name="biceps">Біцепс</string>

<string name="triceps">Трицепс</string>

<string name="shoulders">Плечі</string>

<string name="back">Спина</string>

<!-- execution technique -->

<string name="hammer_execution">Вправи на зовнішній біцепс виконуються аналогічно класичним підйомам гантелей. Єдина відмінність полягає в тому, що щітки під час підйому і опускання снаряда

розташовані вертикально. Ця особливість руху допомагає підкреслити навантаження на зовнішню частину м'язів-розгиначів, сприяючи формуванню красивого та рельєфного плеча. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="barbell_lift_for_biceps_execution">Візьміть штангу вузьким хватом так, щоб долоні були повернені вгору і були на ширині плечей. Встаньте прямо, з обтяженням плечей, м'язів преса і сідниць. Повільно підніміть штангу до плечей, скорочуючи біцепси. Не дозволяйте ліктям рухатися під час руху. Коли штанга досягне ваших плечей, затримайтеся на мить, потім повільно опустіть її вниз. Повторіть вправу відповідно до своїх цілей і рівня фізичної підготовки. \nЗробіть 3–4 підходи по 10 – 12 повторень</string>

<string name="bending_arms_with_dumbbells_execution">Початкове положення: встаньте з гантелями з боків, долонями вперед. Правильна постава – спина пряма, ноги на ширині плечей. Як виконувати вправу: Зосередьтеся на згинанні ліктів, повільно піднімаючи гантелі до плечей. При цьому забезпечуйте постійний контроль рухів і уникайте пошкодження м'язів. Стисніть біцепс у верхній частині руху, затримайтеся на секунду, потім повільно поверніть гантелі у вихідне положення. Контроль дихання: вдихніть, опускаючи гантелі, і видихніть, піднімаючи гантелі. Кількість повторень і підходів: це рекомендовано виконувати вправу 8–12 повторів у 3–4 підходи. Відпочинок між підходами: відпочивайте 1–2 хвилини між підходами, щоб дати м'язам відновитися перед наступним підходом. Правильна техніка: не використовуйте м'язи плечей для підйому гантелі. Орієнтуйтеся на те, щоб навантаження приймали тільки двоголові м'язи. Щоб забезпечити стійкість тіла, важливо тримати спину прямо, а ноги на ширині плечей. Також важливо уникати навантаження на шию, тому не нахилитися. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="block_pull_for_triceps_execution">Початкове положення: сядьте на тренажер перед блоком і візьміться за ручки знизу, зігнувши лікті під кутом 90 градусів. Ручки потрібно взяти так, щоб відстань між ними була невеликою. Виконання вправи: розставте ноги на ширині плечей і трохи нахиліться спиною вперед. Витягніть руки вниз, згинаючи лікті, зберігаючи постійний контроль над рухом. Стисніть трицепс у верхній частині руху, затримайтеся на секунду, потім повільно поверніть руки у вихідне положення. Контроль дихання: вдихніть, опускаючи руки, і видихніть, піднімаючи руки. Кількість повторень і підходів: це рекомендовано виконувати вправу 8–12 повторів у 3–4 підходи. Відпочинок між підходами: відпочивайте 1–2 хвилини між підходами, щоб дати м'язам відновитися перед наступним підходом. Правильна техніка: не використовуйте м'язи плечей для підйому ваги. Орієнтуйтеся на те, щоб навантаження сприймали тільки триголові м'язи. Щоб забезпечити стійкість тіла, важливо тримати спину рівною, а м'язи корінців затиснутими. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="french_triceps_press_execution">Установка на лаву: ляжте на лаву, візьміть гантелі або штангу так, щоб долоні дивилися в стелю. Правильна техніка: при виконанні французького жиму важливо дотримуватися правильної техніки і форми. Лікті повинні бути вгорі і нерухомі, а долоні дивитися в стелю. Опустіть гантелі або штангу за голову, зігнувши лікті під кутом 90 градусів. Потім повільно поверніться у вихідне положення, згинаючи лікті до повної прямої лінії. Дихання: вдих під час опускання та видих під час підйому. Контроль: контролюйте рухи, не дозволяйте гантелям або штанзі тремтіти або рухатися вбік. Відпочинок: відпочинок між підходами і не забувайте про важливість відновлення м'язів після тренування. Поступове збільшення навантаження: коли ваша фізична форма покращується, ви можете збільшувати вагу гантелей або штанги, але робіть це поступово, щоб уникнути травм. \nЗробіть 3–4 підходи по 10–12 повторень</string>

<string name="Lifting_dumbbells_on_the_forearm_execution">Початкове положення: станьте прямо з гантелями в руках, долонями вниз. Розмістіть гантелі перед собою на відстані більше ширини плечей, зігніть

лікті і підніміть гантелі до рівня передпліч. Вправа: повільно піднімайте гантелі на максимальну висоту, згинаючи зап'ястя. У верхній точці руху затримайтеся на секунду, стискаючи м'язи передпліччя, а потім повільно поверніть гантелі у вихідне положення. Контроль дихання: при опусканні гантелей вдих, підйом - видих. Кількість повторень і підходів: рекомендується виконувати вправу 8–12 повторів у 3–4 підходи. Відпочинок між підходами: відпочивайте 1–2 хвилини між підходами, щоб дати м'язам відновитися перед наступним підходом. Правильна техніка: не піднімайте гантелі занадто швидко і не використовуйте м'язи плечей, щоб підняти гантелі. Орієнтуйтеся на те, щоб навантаження сприймали тільки м'язи передпліччя. Також важливо не перенапружуватися. Зробіть 3–4 підходи по 10–12 повторень

extension_of_arms_on_the_simulator_for_triceps_execution Початкове положення: сядьте на тренажер і візьміть штангу над головою, поклавши руки на перекладину на ширині плечей. Витягніть лікті так, щоб штанга опускалася за голову, а лікті залишалися в статичному положенні біля вух. Як виконувати вправу: Зосередьтеся на розгинанні ліктів, коли ви повільно піднімаєте штангу у вихідне положення. При цьому забезпечуйте постійний контроль рухів і уникайте пошкодження м'язів. Стисніть трицепс у верхній частині руху, затримайтеся на секунду, потім повільно поверніть штангу у вихідне положення. Контроль дихання: вдихніть, опускаючи штангу, і видихніть, піднімаючи штангу. Кількість повторень і підходів: це рекомендовано виконувати вправу 8–12 повторів у 3–4 підходи. Відпочинок між підходами: відпочивайте 1–2 хвилини між підходами, щоб дати м'язам відновитися перед наступним підходом. Правильна техніка: не використовуйте м'язи плечей для підйому ваги. Зосередьтеся на тому, щоб навантаження сприймали тільки трицепси. Зробіть 3–4 підходи по 10–12 повторень

barbell_squats_execution Почніть з того, що розмістіть штангу на стійках на такому рівні, щоб ви могли легко підняти її на плечі. Встаньте біля штанги та покладіть її на задню частину плечей, тримаючи її долонями вгору та тримаючи штангу пальцями. Зробіть крок назад і поверніться до передньої частини залу. Розставте ноги на ширині плечей і поверніть пальці ніг трохи всередину. Почніть присідати, згинаючи коліна і опускаючи таз, поки стегна не будуть паралельні підлозі. Тримайте спину прямо, груди вперед і голову вгору. Видихніть, піднімаючи штангу, випрямляючи ноги та повертаючись у вихідне положення. Повторіть вправу відповідно до своїх цілей і рівня фізичної підготовки. Зробіть 3–4 підходи по 7–10 повторень

breeding_legs_in_the_simulator_execution Початкове положення: сісти на тренажер, випрямити стегна і зігнути коліна під кутом 90 градусів. Ваші ноги повинні бути на ширині плечей, а стопи повинні бути на одному рівні з платформою. Виконання вправи: розведіть ноги в сторони до зближення колін. Зробіть секундну паузу в положенні максимального розгинання, а потім поверніть ноги у вихідне положення. Контроль дихання: вдихніть, розкриваючи ноги, і видихніть, повертаючи їх у вихідне положення. Кількість повторів і підходів: рекомендується виконайте вправу 10–15 повторів у 3–4 підходи. Відпочинок між підходами: Відпочиньте 1–2 хвилини між підходами, щоб дати м'язам відновитися перед наступним підходом. Правильна техніка: не робіть рухів тазом, щоб не пошкодити стегон і крижів. Також важливо не згинати ноги при виконанні вправи, так як це може знизити навантаження на м'язи. Зосередьтеся на використанні м'язів ніг. Зробіть 3–4 підходи по 10–12 повторень

reduction_of_legs_in_the_simulator_execution Регулювання машини: відрегулюйте машину відповідно до свого росту та розміру тіла. Встаньте на платформу обличчям до спинки сидіння. Поставте ноги на спеціально відведені підставки, сидниці чітко розташуйте на площині сидіння, вільно поверніться за спину і встановіть опори. Вихідне положення: взявшись за ручки тренажера і зігнувши руки в ліктях,

втягніть корпус назад, навіть якщо корпус трохи нахилений вперед, а ноги стоять рівно, щоб стегна залишалися на одній лінії з тазом. Виконуючи вправу: напружуючи м'язи стегон і живота, спочатку плавно піднімайте ноги вгору вздовж стегна. лінії до максимальної точки напруги, а потім плавно повернути ноги у вихідне положення. Контроль дихання: під час підйому ніг на вдиху і при поверненні у вихідне положення на видиху. Кількість повторень і підходів: рекомендовано виконувати 10–15 повторів у 3–4 підходи. Відпочинок між підходами: відпочивайте 1–2 хвилини між підходами, щоб м'язи відновилися перед наступним підходом. Правильна техніка: не допускайте згинання колін під час виконання вправи і не нахиліться вперед у попереку, оскільки це може знизити ефективність вправи. \nЗробіть 3–4 підходи по 10–12 повторень</string>

<string name="leg_press_in_the_simulator_execution">Відрегулюйте тренажер під свій зріст і зручне положення. Сидячи на сидінні тренажера, вам повинно бути зручно, а ваші ноги повинні стояти на платформі. Прикріпіть ремені до вільних ваг, щоб забезпечити безпеку та стабільність під час виконання вправи. Поставте ноги на платформу на ширині плечей і направте їх вперед. Повільно стискайте платформу ногами до повного випрямлення ніг. При цьому не розводьте стопи в сторони. Затримайтеся на частку секунди у верхній точці, зосереджуючись на нарузі м'язів. Повільно поверніться у вихідне положення, зберігаючи контроль над рухом. Повторіть вправу, зосередившись на правильній техніці та повільних, контрольованих рухах. \nЗробіть 3–4 підходи по 7–10 повторень</string>

<string name="leg_extension_in_the_simulator_execution">Відрегулюйте тренажер на правильну висоту. Сидіння має бути на одній висоті з вашими стегнами, а коліна повинні бути на рівні основи тренажера. Сядьте на тренажер у правильній поставі. Спина повинна бути прямою, а ноги покласти на спеціальні подушки, які відповідають за комфорт і стійкість. Починайте рух від стегна. Поступово розгинаючи ноги, ви повинні відчувати напругу в м'язах стегна і сідниць. Не робіть різких рухів і не дозволяйте машині рухатися швидко. Слідкуйте за правильним диханням. Ви повинні вдихнути, коли ви розгинаєте ноги, і видихати, коли повертаєте їх у вихідне положення. Повторіть вправу з потрібною кількістю повторень і підходів залежно від вашої фізичної форми та тренувальних цілей. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="bending_the_legs_in_the_simulator_execution">Візьміться руками за спеціальні ручки або краї лавки. На видиху зігніть гомілки, намагаючись максимально наблизити валик до сідниць. Передня поверхня стегон притиснута до лави. На вдиху розігніть гомілки, плавно опустивши вагу. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="bench_press_execution">Ляжте на лаву для жиму. Візьміть штангу щільним хватом так, щоб ваші долоні були повернуті всередину та трохи менше ширини плечей. Зніміть штангу зі стійки та тримайте її прямо над грудьми. Це буде ваше вихідне положення. Повільно опустіть штангу до грудей, згинаючи лікті. Під час руху намагайтеся тримати лікті близько до тіла. Коли штанга торкнеться ваших грудей, затримайтеся на короткий час, потім повільно поверніться у вихідне положення, розгинаючи лікті. Повторіть вправу відповідно до ваших цілей і рівня фізичної підготовки. Спробуйте зосередитися на скороченні біцепса, коли піднімаєте штангу. \nЗробіть 3–4 підходи по 6–10 повторень.</string>

<string name="dumbbell_bench_press_on_a_horizontal_bench_execution">Почніть з вибору правильної ваги гантелей. Виберіть вагу, яку ви можете підняти 8–10 разів без зайвих зусиль. Занадто легкі гантелі не дадуть потрібного навантаження, а занадто важкі можуть призвести до травми. Ляжте на горизонтальну лаву з гантелями в кожній руці. Ноги повинні бути прямими, а стопи опущені на підлогу. Тримайте гантелі над грудьми, зігніть лікті під кутом 90 градусів. Повільно опустіть гантелі до рівня грудей. Лікті повинні

рухатися паралельно підлозі і знаходитися приблизно на рівні грудей. При виконанні жиму гантелей важливо зберігати правильне положення тіла. Зберігайте природний вигин хребта, не піднімайте ноги, не згинайте спину. Лікті повинні рухатися під кутом 45 градусів до тіла. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="breeding_dumbbells_on_an_incline_bench_execution">Почніть з вибору правильної ваги гантелей. Виберіть вагу, яку ви можете підняти 8–10 разів без зайвих зусиль. Занадто легкі гантелі не дадуть потрібного навантаження, а занадто важкі можуть призвести до травми. Відрегулюйте нахил лаву під потрібним кутом. Зазвичай для розведення гантелей використовується кут нахилу від 30 до 45 градусів. Сядьте на похилу лаву з гантелями в руках. Покладіть гантелі на стегна, потім підніміть їх до плечей за допомогою ніг. Ляжте на похилу лаву на спину і підніміть гантелі над собою, зігніть лікті під кутом 90 градусів. Це буде ваше вихідне положення. Розведіть гантелі в сторони, опустивши їх до рівня плечей. Лікті повинні бути злегка зігнуті, а руки злегка зігнуті в ліктях. При підйомі гантелей на похилій лаві важливо зберігати правильне положення корпусу. Зберігайте природний вигин хребта, не піднімайте ноги і не згинайте спину. Повільно поверніть гантелі у вихідне положення над грудьми. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="incline_dumbbell_bench_press_execution">Поставте похилу лаву під кутом приблизно 30–45 градусів. Ляжте на лаву з гантелями в кожній руці. Гантелі повинні бути біля ваших плечей, а лікті зігнуті під прямим кутом. Рухом рук підніміть гантелі до точки прямо над грудьми. У цей момент дихайте повільно та стримано. Поступово опускайте гантелі у вихідне положення, контролюючи рухи та дихання. Повторюйте вправу для заданої кількості повторень або тривалості тренування. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="pulldown_of_the_upper_block_to_the_chest_execution">Відрегулюйте блок на потрібну висоту, приблизно на рівні грудей. Сядьте перед блоком, стисніть коліна та добре прилягайте до сидіння. Витягніть руки вгору та злегка зігніть лікті. Повільно опустіть руки вниз, збільшуючи кут між руками та ліктями, потягнувши важіль до грудей. Затримайтеся на одну-дві секунди внизу, де м'язи найбільш напружені. Повільно поверніть руки у вихідне положення, знову збільшуючи кут між руками та ліктями. Повторіть це вправи для вказаної кількості повторень або тривалості тренування. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="dumbbell_pull_to_the_belt_execution">Встаньте на ширині плечей і злегка зігніть коліна, взявшись за гантелі. Нахиліться вперед, злегка нахиливши тулуб, і поверніть гантелі вниз. Розведіть лікті в сторони, опустіть плечі вниз і назад. Потягніть гантелі вгору до себе, поперек, відводячи плечі назад. Затримайтеся на одну-дві секунди внизу, де м'язи найбільше напружені. Повільно поверніть гантелі у вихідне положення, контролюючи рухи. Повторіть цю вправу з указаною кількістю повторень або тривалістю. Виконуючи цю вправу, дуже важливо дотримуватися правильної техніки і контролювати рухи. Щоб отримати максимальну віддачу від тренування, виконуйте цю вправу повільно та зосереджено. Також не забудьте підібрати вагу гантелей відповідно до свого рівня фізичної підготовки. \nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="deadlift_execution">Початкове положення: встаньте перед штангою, ноги повинні бути на ширині плечей або трохи ширше, а штанга на рівні середини стопи. Руки повинні бути покладені на перекладину на ширині плечей, а пальці долоні повинні бути спрямовані вперед. Опустіться на штангу: нахиліться вперед і зігніть коліна, поки руки не торкнуться штанги. Під час прогину тримайте спину рівно, а

погляд дивіться вперед. Підготовка до підйому штанги: глибоко вдихніть, відчувши напругу в м'язах грудей і спини. Крім того, ви повинні відчувати напругу в біцепсі, щоб уникнути пошкодження зв'язок. Підніміть штангу: повільно піднімайте штангу, розгинаючи коліна та стегна. Підніміть штангу, спираючись на ноги, не згинаючи коліна. Затримайтеся на секунду: затримайтеся на секунду в положенні у верхній частині, де м'язи будуть найбільш напруженими. Поверніть штангу у вихідне положення: повільно опустіть штангу, нахилившись ваші коліна та стегна. Важливо дотримуватися контрольованих рухів, уникати стрибків.\nЗробіть 3–4 підходи по 4–6 повторень.</string>

<string name="hyperextension_on_the_bench_execution">Початкове положення: ляжте на живіт на лаву для гіперекстензії так, щоб стегна були зафіксовані під ріжучою кромкою лави, а ноги були злегка зімкнуті. Тримайте пряму лінію від шиї до стопи, щоб забезпечити правильне положення тіла. Підготовка до підйому: візьміться за ручки, згинаючись у стегнах, зберігаючи правильне положення голови та шиї. Зміцнійте м'язи грудей і спини, необхідні для підтримки стабільного положення. Підйом корпусу: повільно піднімайте корпус вгору, використовуючи м'язи попереку і сідничні м'язи. Підніміть тіло, поки спина не опиниться на одній лінії з ногами. Слідкуйте за контрольованим рухом, уникайте зіткнення з лавою. Затримайтеся на секунду: Затримайтеся на секунду у верхній позиції, де м'язи будуть найбільш напруженими. Не вигинаючи спину, дозволяйте їй бути прямою та встановіть контроль. Поверніться у вихідне положення: повільно опустіть тіло у вихідне положення, контролюючи рух тіла та повертаючись у вихідне положення.\nЗробіть 3–4 підходи по 10 – 12 повторень</string>

<string name="T_bar_row_on_the_block_execution">Регулювання ваги: встановіть вагу блоку, який вам підходить, і підійдіть до нього. Встановіть Т-подібну штангу на блок. Початкове положення: сядьте перед лавою, спина пряма, ноги на підлозі. Візьміть Т-подібну штангу в обидві руки, розставте їх на ширині плечей і почніть тягнути штангу на себе. Робіть це, доки не досягнете верхньої частини грудей. Підтягування Т-подібної штанги: затримайтеся на секунду, коли штанга досягне грудей і м'язи спини напружені. Виконуйте контрольований рух. Поверніться у вихідне положення: повільно поверніть гриф у вихідне положення, розслабляючи м'язи спини та зберігаючи контроль над рухом.\nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="standing_dumbbell_press_execution">Почніть з вибору ваги гантелей, достатньої для виконання вправи, але не надто важкої для вас. Встаньте прямо, тримаючи гантелі в руках, долонями дивіться вперед. Ноги повинні бути на ширині плечей, коліна злегка зігнуті. Підніміть гантелі на рівень плечей, зігнувши лікті. При цьому лікті повинні бути високо, а долоні – зверху. Видихніть, а потім повільно випряміть руки, зведіть гантелі разом над головою. При цьому стежте, щоб лікті не торкалися. Затримайтеся на мить у верхній точці, потім повільно поверніть гантелі у вихідне положення, згинаючи лікті. Повторюйте вправу з заданою кількістю повторів. Важливо не прогинатися в нижній частині вправи, щоб уникнути травм. Крім того, не забувайте про правильне дихання: видихайте при підйомі гантелей і вдихайте при поверненні їх у вихідне положення.\nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="standing_barbell_press_execution">Сядьте на лаву і відрегулюйте стійку зі штангою на потрібну висоту. Сидячи на лаві, злегка зігніть коліна і опустіть ступні на підлогу. Візьміть штангу, що лежить на стійці, і тримайте її на рівні плечей вузьким хватом. Лікті повинні бути зігнуті, а долоні дивитися вперед. Видихніть, а потім повільно підніміть штангу вгору, випрямляючи руки, але не змикаючи їх. При цьому долоні повинні бути спрямовані вперед. Затримайтеся на мить у верхній точці і знову повільно поверніть штангу у вихідне положення, згинаючи лікті. Повторіть вправу з заданою кількістю

повторень.\nЗробіть 3–4 підходи. 10–12 повторень</string>

<string name="lifting_dumbbells_to_the_sides_execution">Розведення гантелей у сторони Візьміть пару гантелей і встаньте на ширині плечей, відведіть плечі назад і злегка зігніть лікті. Повільно розводьте гантелі в сторони, поки ваші руки не стануть паралельними підлозі. Долоні повинні бути спрямовані вниз. Затримайтеся на мить у верхній точці, а потім повільно опустіть гантелі у вихідне положення. Повторіть вправу з указаною кількістю повторень. Ось кілька додаткових порад щодо виконання цієї вправи: Не використовуйте занадто багато вага, оскільки це може призвести до травми плеча. Не використовуйте маховик для посилення підйому гантелі в сторони. Піднімайте гантелі повільно і контрольовано. Не піднімайте руки занадто високо. Підніміть гантелі до рівня плечей. Не розгойдуйте корпус під час виконання вправи.\nЗробіть 3–4 підходи по 10–12 повторень.</string>

<string name="lifting_dumbbells_in_front_of_you_execution">Візьміть пару гантелей і встаньте на ширині плечей, відведіть плечі назад і злегка зігніть лікті. Повільно підніміть гантелі вперед до рівня плечей, тримаючи руки прямими та долонями вниз. Затримайтеся на мить у верхній точці, а потім повільно опустіть гантелі у вихідне положення. Повторіть вправу для вказаної кількості повторень. Ось кілька додаткових порад щодо виконання цієї вправи: Не використовуйте надто велику вагу, оскільки це може призвести до травми плеча. Не використовуйте маховик, щоб збільшити підйом гантелей. Піднімайте гантелі повільно та стримано. Не піднімайте гантелі занадто високо. Підніміть гантелі до рівня плечей і затримайте на мить, перш ніж опустити їх у вихідне положення. Не розгойдуйте тіло під час виконання вправи.\nЗробіть 3–4 підходи по 10–12 повторень.</string>

<!-- Hiit-->

<string name="endurance_running_execution">Біг на витривалість (5–10 хвилин): перейдіть до бігу на витривалість на біговій доріжці або на відкритому повітрі. Виберіть інтенсивність, яку ви зможете підтримувати протягом 5–10 хвилин.</string>

<string name="pull_ups_execution">Підтягування (3–4 підходи з максимальною кількістю повторень): переходьте до підтягувань на турніку або на турніку-платформі. Виконуйте їх з максимальною кількістю повторів у кожній серії.</string>

<string name="burpee_execution">Берпі (3–4 підходи по 10–15 повторень): переходьте до бурпі, ця вправа включає стрибки, віджимання та присідання. Виконуйте кожну серію по 10–15 повторень.</string>

<string name="endurance_running">Біг на витривалість</string>

<string name="pull_ups">Підтягування</string>

<string name="burpee">Берпі</string>

<!-- Cardio-->

<string name="treadmill">Бігова доріжка</string>

<string name="exercise_bike">Велотренажер</string>

<string name="elliptical_trainer">Еліптичний тренажер</string>

<string name="rowing_machine">Гребна машина</string>

<string name="warm_up">Розминка</string>

<string name="warm_up_execution">Розминка: 5–10 хвилин на будь-якому кардіотренажері з низькою швидкістю або інтенсивністю, щоб підготувати тіло до більш інтенсивного тренування.</string>

<string name="treadmill_execution">Бігова доріжка: 20–30 хвилин інтенсивного бігу. Радимо почати з повільного темпу, а потім поступово збільшувати швидкість до рівня, який здається вам досить важким, щоб витримати більше 5–10 хвилин, але все ще можна говорити протягом коротких періодів часу.</string>

<string name="exercise_bike_execution">Велотренажер: 20–30 хвилин на велотренажері. Ми радимо вам встановити рівень опору на такому рівні, який здається досить важким, але дозволяє підтримувати темп протягом 20–30 хвилин.</string>

<string name="elliptical_trainer_execution">Еліптичний тренажер: 20–30 хвилин інтенсивного тренування на еліптичному тренажері. Ми рекомендуємо використовувати різні рівні опору та швидкості, щоб підтримувати високу інтенсивність.</string>

<string name="rowing_machine_execution">Гребний тренажер: 10–15 хвилин на гребному тренажері для опрацювання м'язів верхньої та нижньої частини тіла. Ми рекомендуємо використовувати різні інтенсивності та швидкості, щоб ваше тренування було цікавим.</string>

<string name="ingredients">Інгредієнти</string>

<string name="instruction">Інструкція</string>

<string name="previos_measurements">Попередні виміри</string>

<string name="waist">Талія</string>

<string name="hip">Стегно</string>

<string name="buttocks">Сідниці</string>

<string name="calves">Ікри</string>

<string name="current_measurements">Поточні виміри</string>

<string name="developer">Розробник: Данило Гвоздилко</string>

<string name="designer">Дизайнер: Данило Гвоздилко</string>

</resources>

ДОДАТОК В
Слайди презентації

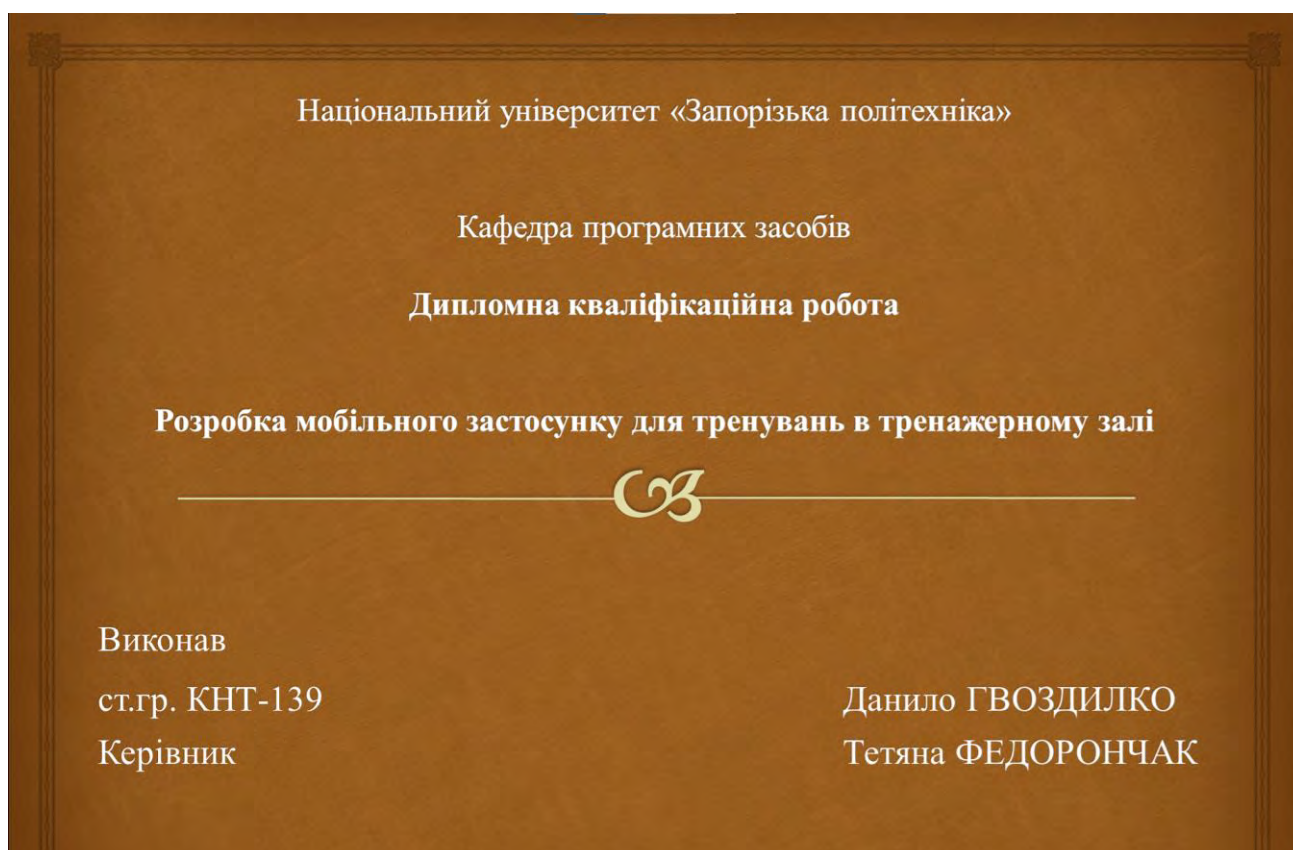


Рисунок В.1 — Слайд 1

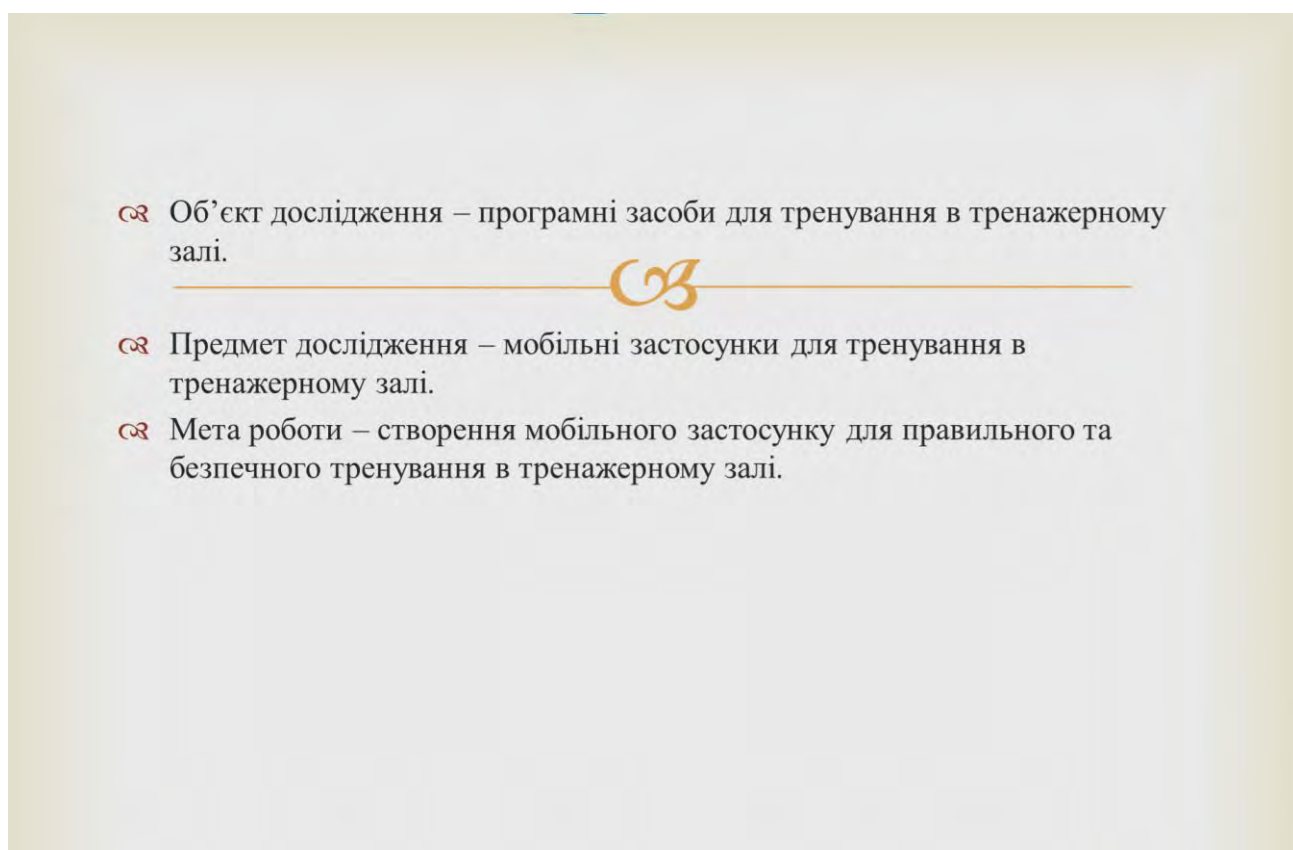


Рисунок В.2 – Слайд 2

Функціональні задачі



- вибір тренувальної програми: користувачі повинні мати можливість вибрати програму тренування, яка найкращим чином відповідає їхнім потребам і цілям. Застосунок має пропонувати різні варіанти тренувань, наприклад силовий, кардіо та функціональний тренінг;
- відстежування прогресу: додаток має надавати можливість відстежувати прогрес користувача;
- давати всю необхідну інформацію, для безпечного та ефективного тренування, включаючи техніку тренування, правильне харчування, спортивні добавки, загальні поради;

Рисунок В.3 – Слайд 3

Аналіз існуючих аналогів



Критерій порівняння	Nike training club	Adidas training	GymUp	Seven	Workout trainer
Підтримка україномовної локалізації	-	-	-	-	-
Аудіоспровід	+	+	-	+	+
Довідник вправ	-	-	+	-	-
Необхідність авторизації	+	+	-	-	+
Наявність розділу з корисними статтями	+	+	+	+	+

Рисунок В.4 – Слайд 4

Вибір мови програмування



Параметри	Java	Kotlin
Простота використання	Важко	Легко
Продуктивність	Висока	Висока
Популярність	Висока	Середня
Робота з потоками	Важка робота	Легкий процес
Безпека від NPE	Не має	Присутня
Лаконічність	Поступається	Має перевагу
Швидкість написання коду	Середня	Висока

Рисунок В.5 – Слайд 5

Вибір системи впровадження залежностей



Характеристика	Dagger	Hilt	Koin
Ким створена	Google	Google	Спільнота Kotlin
Тип ін'єкції	Конструктор, поля	ViewModel, область дії і фрагментів	ViewModel, область дії і фрагментів
Швидкість	Використовує reflection для генерації кода під час компіляції, робить швидким та безпечним	Використовує reflection для генерації кода під час компіляції, робить швидким та безпечним	Використовує reflection для генерації кода під час виконання, робить його повільним та небезпечним
Легкість освоєння	Важка	Легка	Легка

Рисунок В.6 – Слайд 6

Інтерфейс застосунку

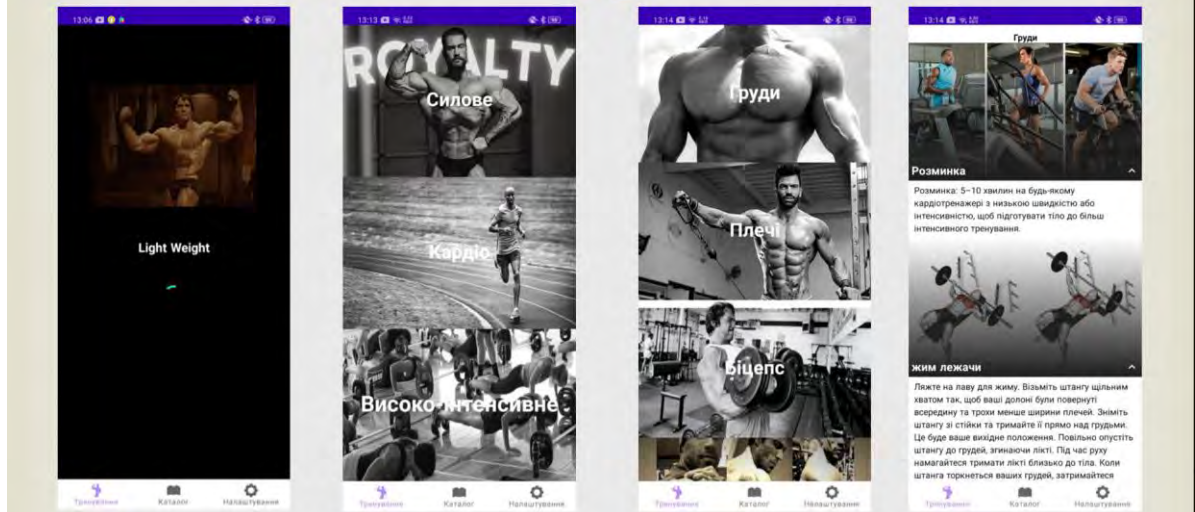


Рисунок В.9 – Слайд 9

Інтерфейс застосунку



Рисунок В.10 – Слайд 10

Висновок



- ❧ З метою надання інформації для безпечного та ефективного тренування в тренажерному залі розробка мобільного застосунку є необхідною.
- ❧ Було проаналізовано існуючі застосунки для тренувань, мови програмувань для написання проекту, архітектури та різні варіанти впровадження залежностей, де було обрано мову – Kotlin, архітектуру – MVVM, систему впровадження залежностей – Hilt.
- ❧ Було визначено функціональні вимоги та вимоги до інтерфейсу програми, вимоги до складу та параметрів технічного засобу, де було обрано мобільний пристрій з ОС - Android. Застосунок відповідає меті та темі розробки та всім перерахованим вимогам, які були визначені.

Рисунок В.11 – Слайд 11

Висновок



- ❧ Програмний продукт був протестований без помилок. Так як це перша версія застосунку, він може оновлюватися та отримувати нові можливості, наприклад:
- ❧ розширення програм тренувань;
- ❧ розширення каталогу с корисною інформацією;
- ❧ додання відеоспроводження під час виконання вправи;
- ❧ покращення дизайну;

Рисунок В.12 – Слайд 12