Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

# Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Мобільний додаток - агент "Помічник спортсмена"»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Даниїл КЛИМЕНКО/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /ст. викл. Ірина ШАПОВАЛ/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

_____ _____ //_____
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ //_____
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ //_____
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ //_____
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

# Explanatory Note
to Bachelor's Thesis

on the topic: «Athlete Assistant mobile application»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group ПЗ1812: /Danyil KLYMENKO/

Scientific Supervisor: /Iryna SHAPOVAL/

Normative controller: /Olena KUROPYATNYK/

Supervisors

| | |
|---|---|
| (Chapter title heading) | / / (position, name, surname) |
| (Chapter title heading) | / / (position, name, surname) |
| (Chapter title heading) | / / (position, name, surname) |
| (Chapter title heading) | / / (position, name, surname) |

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: ~~Факультет~~ «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

**З А В Д А Н Н Я**

на кваліфікаційну роботу бакалавра
студенту Клименко Даниїл Олександрович
1. Тема роботи: «Мобільний додаток - агент "Помічник спортсмена"»
Керівник роботи: Шаповал Ірина Вікторівна, старший викладач
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: 14.06.2022 р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина: збір та аналіз вимог, проектування, опис аналогів, бізнес процеси
4.2 Основна частина: проектування, розробка програми, тестування
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): титулка, зміст, вступ, завдання, огляд програми, висновки

6. Консультанти розділів роботи:

| Розділ | Прізвище, ініціали та посада консультанта | Завдання видав: (підпис консультанта, дата) | Завдання прийняв: (підпис студента, дата) |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі | 31.01.2022-15.02.2022 | |
| 2 | Огляд літератури та аналіз аналогів | 16.02.2022-18.02.2022 | |
| 3 | Прототипування | 19.02.2022-01.03.2022 | 30% |
| 4 | Розробка програми | 02.03.2022-17.03.2022 | |
| 5 | Відлагодження | 18.03.2022-29.03.2022 | 60% |
| 6 | Розробка, узгодження та оформлення даних | 29.03.2022-03.04.2022 | |
| 7 | Подання кваліфікаційної роботи до кафедри | 04.04.2022-12.04.2022 | 100% |
| 8 | Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії | 22.06.2022 | |

Студент _____ Даниїл КЛИМЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ ст. викл. Ірина ШАПОВАЛ
(підпис) (Ім'я ПРІЗВИЩЕ)

# РЕФЕРАТ

Пояснювальна записка до дипломного проєкту: 9 розділів, 122 сторінки, 23 рисунки, 56 таблиць, 4 літературних джерела та 1 додаток.

Об’єктом розроблення є мобільний додаток з клієнт-серверною структурою.

Метою кваліфікаційної роботи є розробка спеціалізованого мобільного додатку для структуризації та контролю тренувального процесу спортсменів.

Задачі:

- проєктування об’єктної моделі мобільного додатку;
- проєктування структури бази даних для мобільного додатку;
- розробка інтерфейсу мобільного додатку;
- розробка функціональної частини мобільного додатку;
- розробка серверної частини;

Ключові слова: мобільний додаток, тренувальний процес, діаграма класів, тест, налагодження.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

# ВСТУП

Програмний засіб «Мобільний додаток - агент "Помічник спортсмена"» призначений для того, щоб задовільнити потреби спортсменів різного рівня підготовки, а саме: фіксувати навантаження; знаходити теоретичні матеріали; планувати тренування; відстежувати досягнення.

Головною метою розробки є оптимізація та організація тренувального і навчального процесу спортсменів. Застосунки, наявні на ринку, зазвичай слабо заглибленні у професійний напрямок та мають низьку ефективність як для початківців, так і для спортсменів високого рівня. Більшість програм мають дуже вузький спектр фунціоналу та не можуть самостійно запропонувати комплексний підхід до спортивного розвитку. Це в свою чергу змушує користувача використовувати декілька несинхронізованих додатків, зменшуючи загальну ефективність користування смарт-засобами для підвищення рівня фізичної підготовки спортсменів.

Експлуатаційним призначенням даного мобільного додатку є підвищення ефективності тренувань за рахунок швидкої та зручної фіксації важливих показників, доступу до достовірних та ефективних планів тренувань, зручного відстеження прогресу. Завдяки охопленню більшої частини функціоналу, потрібного для фіксації та опрацювання розвитку, користувач зможе бачити більшість потрібної інформації в одному додатку. Це дасть змогу скоротити час на занесення та порівняння показників з декількох додатків. Важливим аспектом є розробка універсальної платформи для розвитку в більшості спортивних напрямків з достатньою гнучкістю та адаптивністю для різних видів спорту.

# 1. Збір та аналіз вимог

## 1.1.Методи збору вимог до програмного забезпечення

Для збору вимог було використано метод опитування. Метод опитування – психологічний вербально-комунікативний метод, який полягає у здійсненні взаємодії між інтерв'юером і опитуваними (респондентами) з метою одержання від суб'єкта відповідей на заздалегідь підготовлені запитання. Джерелом інформації в опитуванні є письмові або усні судження респондента. Опитування може проводитись як особисто, коли дослідник безпосередньо контактує з респондентами, так і дистанційно при опосередкованій участі дослідника або ж взагалі без його участі.[1]

Опитування проводилося через анкетування засобами Google Forms. Анкетування є найбільш формалізованою формою опитування. Це метод отримання інформації за допомогою письмових відповідей на систему заздалегідь підготовлених і стандартизованих питань з точно зазначеним способом відповідей. Основним інструментом цього виду опитування є анкета.[1]

Була розроблена анкета для отримання основних вимог до розробки мобільного додатку. Опис запитань та результати анкетування наведені у таблиці 1.1.1 Анкетування пройшли 83 учасники. За результатами опитування було виділено основні вимоги до мобільного додатку.

Таблиця 1.1.1 Запитання в анкеті

| № | Тип запитання | Опис запитання | Текст | Результат |
|---|---|---|---|---|
| 1 | Просте альтернативне запитання | Вибір одного з двох варіантів відповіді | Мобільним пристроєм з якою операційною системою ви користуєтеся?<br>– iOS;<br>– Android. | «iOS» обрано 37 разів, що дорівнює 44,58 відсоткам, «Android» обрано 46 разів, що дорівнює 55,42 відсоткам. |

Продовження таблиці 1.1.1

| № | Тип запитання | Опис запитання | Текст | Результат |
|---|---|---|---|---|
| 2 | Питання з декількома варіантами відповіді | Вибір одного з трьох та більше варіантів відповіді | Вкажіть ваш вік:<br>a) до 14 років;<br>b) від 14 до 18 років;<br>c) від 18 до 25 років;<br>d) 25 років та більше. | a) 21 раз;<br>b) 35 разів;<br>c) 15 разів;<br>d) 11 разів. |
| 3 | Просте альтернативне запитання | Вибір одного з двох варіантів відповіді | Ви зацікавлені у визначенні цілей та контролі їх досягнення?<br>– так;<br>– ні. | Варіант «так» обрали 72 рази, варіант «ні» обрали 11 разів |
| 4 | Просте альтернативне запитання | Вибір одного з двох варіантів відповіді | Ви вважаєте корисним доступ до навчальних статей в мобільному додатку?<br>– так;<br>– ні. | Варіант «так» обрали 58 разів, варіант «ні» обрали 25 разів |
| 5 | Просте альтернативне запитання | Вибір одного з двох варіантів відповіді | Вам потрібна можливість запису деталізації тренувань?<br>– так;<br>– ні. | Варіант «так» обрали 53 разів, варіант «ні» обрали 30 разів |

Продовження таблиці 1.1.1

| № | Тип запитання | Опис запитання | Текст | Результат |
|---|---|---|---|---|
| 6 | Питання з вибірковою підмножиною | Вибір декількох відповідей з наведених варіантів | Оберіть додатки, які ви використовували для ведення обліку тренувань:<br>a) Nike training club;<br>b) Riders;<br>c) Runtastic;<br>d) BetterMe;<br>e) Home workout; | a) 32 рази;<br>b) 10 разів;<br>c) 18 разів;<br>d) 26 разів;<br>e) 31 раз. |
| 7 | Питання з декількома варіантами відповіді | Вибір одного з трьох та більше варіантів відповіді | Який з варіантів ви вважаєте найкращим?<br>a) один користувач має один обліковий запис з різними видами спорту;<br>b) один користувач пов’язаний з декількома обліковими записами для кожного виду спорту;<br>c) один користувач має один обліковий запис з одним видом спорту; | a) 18 разів;<br>b) 52 рази;<br>c) 13 разів. |

## 1.2.Прототипування

Прототипування (prototyping) – це найбільш часто використовуваний сучасний метод виявлення вимог. Програмні прототипи конструюються для візуалізації системи або її частини для замовників з метою отримання їх відгуків.[1]

Було розроблено прототипи запланованих екранів для розділів мобільного додатку рисунок 1.2.1-1.2.6.


Рисунок 1.2.1 Сторінка облікового запису

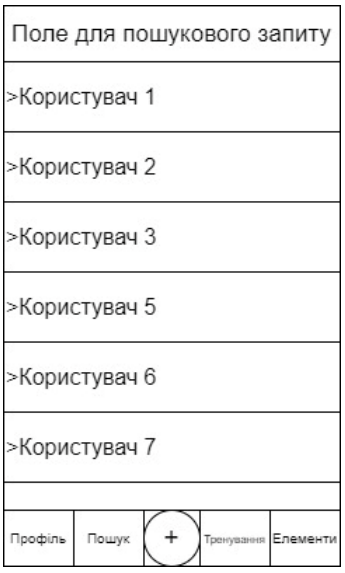


Рисунок 1.2.2 Сторінка пошуку

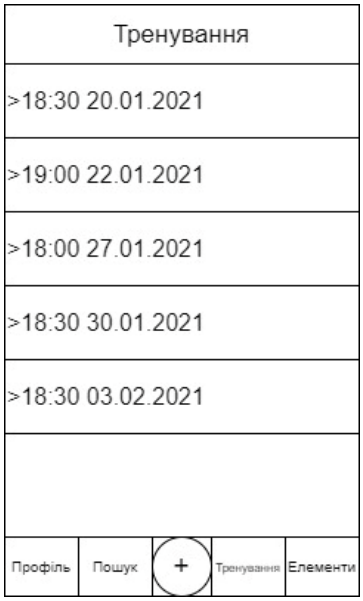


Рисунок 1.2.3 Сторінка списку тренувань


Рисунок 1.2.4 Сторінка тренування

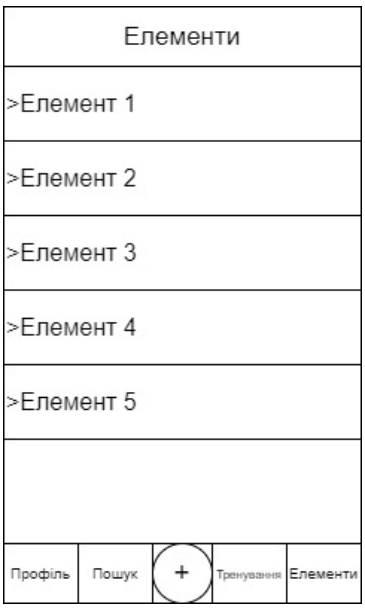


Рисунок 1.2.5 Сторінка списку елементів

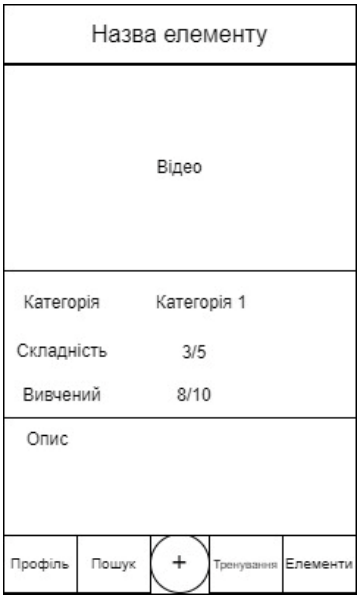


Рисунок 1.2.6 Сторінка елементу

1.3.Бізнес процеси

Головним бізнес процесом цього додатку є тренування. Його схема реалізована на рис. 1.3.1

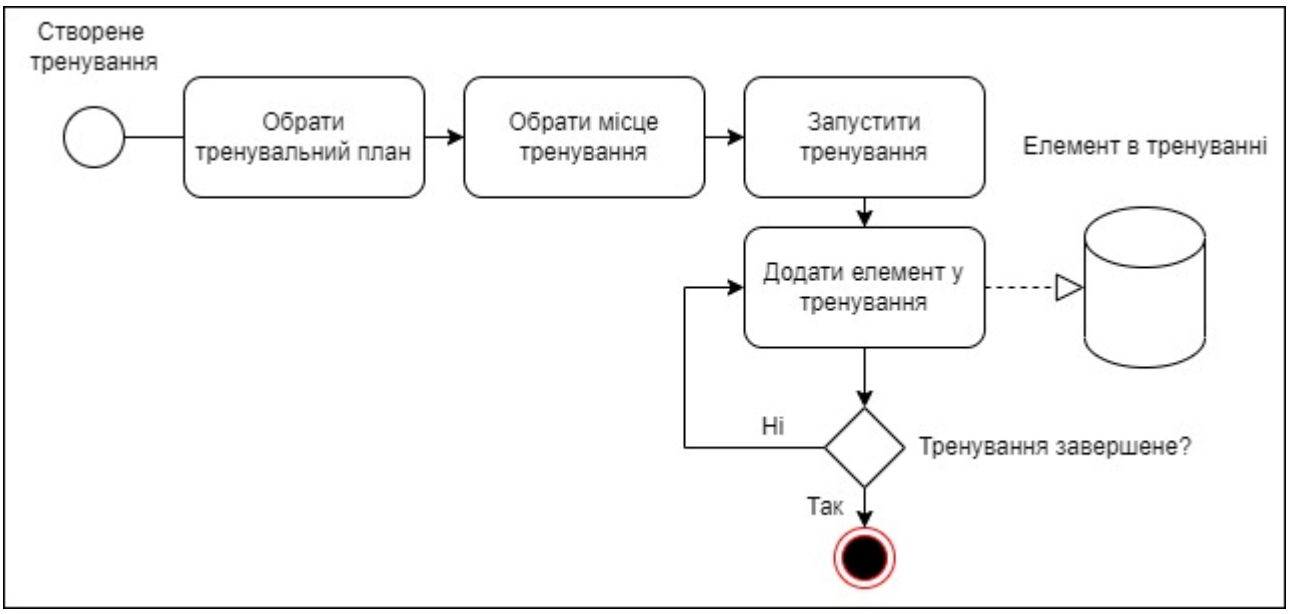


Рисунок 1.3.1 Схема бізнес-процесу «Тренування»

## 1.4.Опис аналогів

Існує велика кількість мобільних додатків створених для оптимізації тренувального процесу. Більшість з них дуже схожі один на одного, дають подібний користувацький досвід та мають спільні недоліки. Основна проблема в недостатній деталізації вивчення нових елементів, цей недолік виходить з того, що більшість з цих додатків орієнтовані на звичайний фітнес чи здоровий образ життя, а це, в свою чергу, означає, що тренувальні програми складаються з елементарних задач.

Відсутність можливості послідовного вивчення «вглибину» робить звичайний блокнот більш привабливим для людей які займаються більшістю видів спорту. З популярних додатків можна відібрати для аналізу ті, які мають найбільшу відмінність один від одного: Riders та Nike Training Club.

### 1.4.1. Riders

Один з найближчих аналогів, який дає можливість бачити зв’язок між елементами і вивчати їх послідовно.

Переваги:

– має розділення з деталізацією для кожного виду спорту;
– відмічені рівні складності та зв’язки між елементами;

– є можливість додавати відео виконання елементів, дивитися відео інших користувачів.

Недоліки:

– відсутня можливість фіксувати окремі тренування;

– відсутня можливість фіксувати виконання елементів;

– застаріла база даних, давно не підтримується розробниками.

1.4.2. Nike Training Club

Цей додаток має найбільш розгалужену соціальну мережу.

Переваги:

– має багато програм для фітнесу;

– має гарні відео-матеріали для контролю правильності виконання;

– має розвинену соціальну складову: змагання, рейтинги, спілкування у додатку.

Недоліки:

– відсутня можливість вивчення елементів.

1.5.Висновки

1.5.1. Методи збору вимог до програмного забезпечення

Було проаналізовано аудиторію майбутнього додатку. Проведено опитування, в ході якого визначені основні функції, потрібні потенційним користувачам.

1.5.2. Прототипування

На цьому етапі, виходячи з потрібних функцій та на основі приблизного набору сутностей було побудовано прототип інтерфейсу користувача мобільного додатку.

1.5.3. Бізнес процеси

Було побудовано діаграму основного бізнес процесу цього мобільного додтку.

1.5.4. Опис аналогів

Після дослідження та аналізу подібних мобільних додатків, відібраних за популярністю серед опитаних користувачів, було виділено функціональність, потрібну для отримання переваги над наявними на ринку мобільними додатками.

## 2. Проєктування

### 2.1.Зовнішнє проєктування

#### 2.1.1. Функціональне призначення

Функціональним призначенням програми є фіксація, обробка, візуалізація фізичних показників та медіаматеріалів, отриманих від користувачів. Створення соціальної мережі з декількома типами користувачів згідно з їх ролями в процесі спортивного розвитку.

#### 2.1.2. Експлуатаційне призначення

– створення повноцінної екоситеми для професійного розвитку користувача;
– збільшення продуктивності спортивної підготовки;
– покращення теоретичної підготовки спортсменів;
– попередження небезпечних та травматичних ситуацій;
– надання можливості для соціального зростання талановитих та працьовитих спортсменів за рахунок оприлюднення показників їх досягнень.

#### 2.1.3. Функціональні вимоги

Додаток повинен реалізовувати даний функціонал:

– реєстрація користувачів;
– додавання тренування;
– додавання елементів в тренуванні;
– перегляд інформації про завершені тренування;
– пошук і перегляд інформації про користувача;
– додавання елементів для вивчення в профіль.

Діаграма прецедентів подана на рис.2.1

Прецеденти (use case) – це опис послідовностей дій (включаючи їх варіанти), які виконуються системою для того, щоб актор отримав результат, що має для нього певне значення [2].

### 2.1.4. Вхідні дані

Вхідними даними мобільного додатку є:

- параметри авторизації;
  - email;
  - пароль;
- інформація про користувача
  - ім’я;
  - прізвище;
  - по-батькові;
  - номер телефону;
  - дата народження;
  - ім’я користувача;
- фізичні параметри користувача
  - зріст;
  - вага;
  - стать.
- параметри тренування;
  - місце тренування;
  - спорядження для тренування;
  - навчальний матеріал за яким проходить тренування;
  - час тренування(початок та кінець);
  - елементи виконані під час тренування
- параметри елементів у тренуванні;
- елемент(Вибір зі списку);
- ступінь виконання.

### 2.1.5. Вихідні дані

Вихідними даними мобільного додатку є:

– список завершених тренувань;
– інформаційна сторінка «профіль» користувача;
– список наявних записів навчальних матералів;
– список елементів для виконання на тренуванні;
– текстові та відео- матеріали у статті навчальних матеріалів

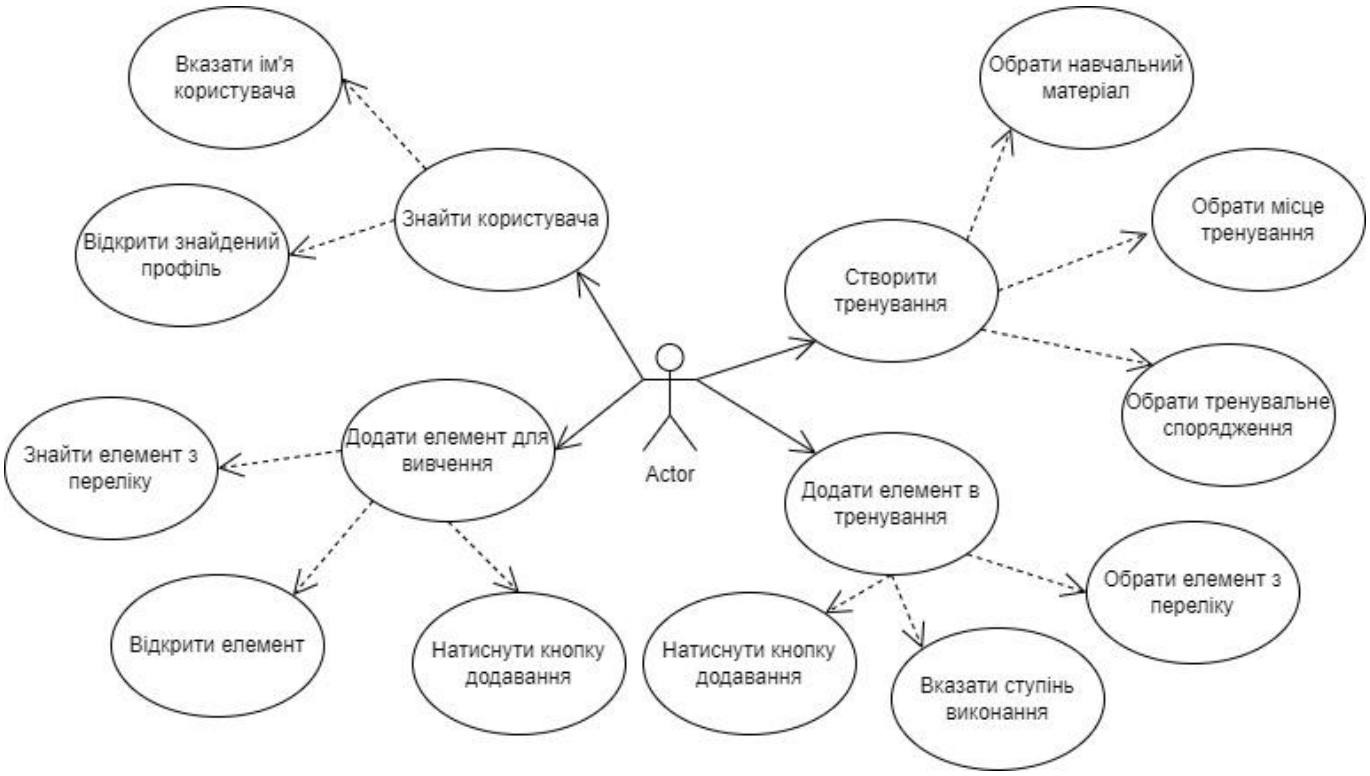


Рисунок 2.1 Діаграма прецедентів

2.1.6. Висновки

В результаті виконання зовнішнього проєктування, відштовхуючись від потреб аудиторії було сформульоване функціональне та експлуатаційне призначення. Також були розроблені функціональні вимоги для формування діаграми прецедентів та отримання переліку дій, які потрібно реалізувати. Виходячи з наявних вимог були сформовані набори вхідних та вихідних даних.

## 2.2.Внутрішнє проєктування

### 2.2.1. Проєктування архітектури системи

Додаток побудований з модулів, які відповідають за взаємодію користувача з екранами перегляду. Нижче наведений перелік модулів:

ProfileActivity – модуль у якому користувач може переглянути інформацію про користувача зі списку зареєстрованих, в залежності від прав доступу користувача до відкритого профілю, є можливість відкрити сторінку редагування профілю(EditProfileActivity), сторінку редагування фізичних показників(EditParamethersActivity), додати користувача в лист вподобань, видалити користувача з листа вподобань, відкрити сторінку з тренуваннями користувача(TrainingsListActivity), відкрити сторінку з елементами доданими користувачем(UserFollowedElementsListActivity).

TrainingsListActivity – сторінка для взаємодії зі списком тренувань, з якої можна відкрити сторінку завершеного тренування (TrainingActivity) або відкрити сторінку з новим тренуванням (CurrentTrainingActivity).

TrainingActivity – сторінка на якій користувач може переглянути інформацію про завершене тренування та перейти на сторінку виконаного під час тренування елементу (CompletedTrainingElementActivity).

SearchUserActivity – модуль пошуку користувачів, з якого можна відкрити сторінку користувача (ProfileActivity).

EditProfileActivity – модуль з формою для внесення даних про користувача.

EditParamethersActivity – модуль з формою для внесення фізичних параметрів користувача.

CurrentTrainingActivity – модуль з інформацією про запущений процес тренування, та з можливістю відкривати сторінки нового елементу (AddTrainingElementActivity) та виконаного елементу (CompletedTrainingElementActivity).

AddTrainingElementActivity – модуль з полями для внесення інформації про новий елемент в тренуванні.

CompletedTrainingElementActivity – модуль з інформацією про елемент зі списку виконаних під час тренування.

UserFollowedElementsListActivity – модуль з відображення у вигляді окремих блоків користувачів, яких відстежує обраний користувач, по натисканню на блок з користувачем відкривається сторінка профілю (ProfileActivity).

UserFollowedElementActivity – модуль з відображенням у вигляді окремих блоків елементів, які користувач додав для вивчення, по натисканню на блок відбувається перехід на сторінку з описом елементу (ElementInfoActivity).

TrainingPlansListActivity – модуль з відображенням наявних у системі тренувальних планів для елементів з можливістю відкриття сторінки тренувального плану (TrainingPlanActivity).

TrainingPlanActivity – модуль з відображенням інформації про елемент, списку елементів включених до тренувального плану з можливістю перейти на сторінку з інформацією про елемент (ElementInfoActivity).

ElementsListActivity – модуль з відображенням у вигляді списку всіх елементів з можливістю перейти на сторінку з інформацією про елемент (ElementInfoActivity).

ElementInfoActivity – модуль з відображенням інформації про елемент та списком рекомендованих тренувальних планів для вивчення цього елементу з можливістю переходу на сторінку плану (TrainingPlanActivity) та додавання елементу в лист для вивчення.

Основні логічні сутності представлені на рис. 2.2

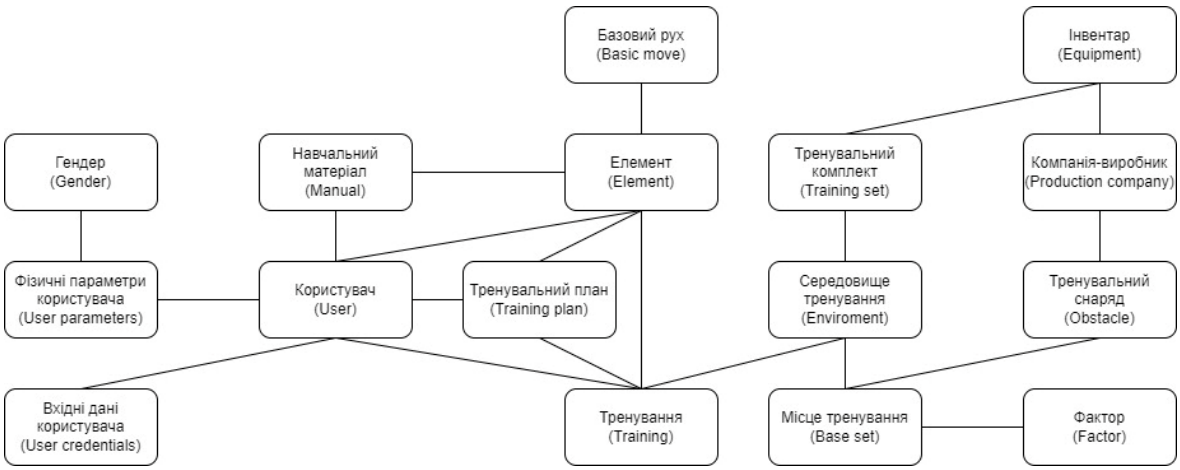


Рисунок 2.2 Діаграма сутностей

2.2.2. Проєктування інтерфейсу користувача

Розроблена діаграма станів користувача програми наведена на рис. 2.3.


Рисунок 2.3 Діаграма станів користувача програми

На основі даних, які потрібно вивести, та діаграми станів був розроблений інтерфейс користувача. Він зображений на рисунках 2.4-2.10

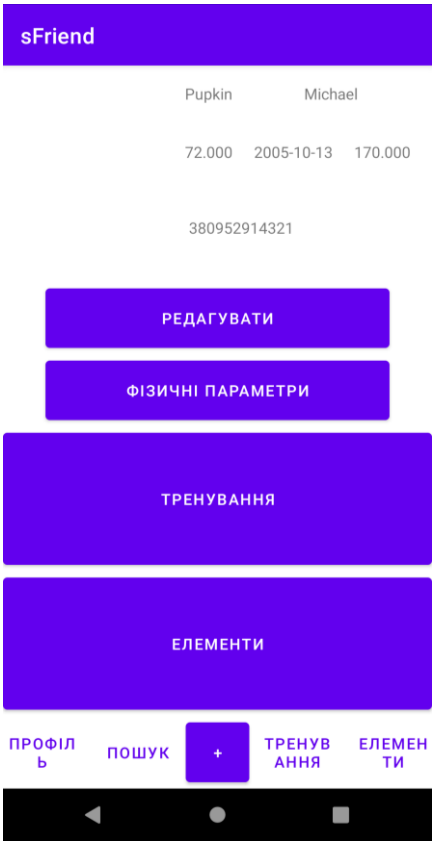


Рисунок 2.4 Інтерфейс сторінки «Профіль»

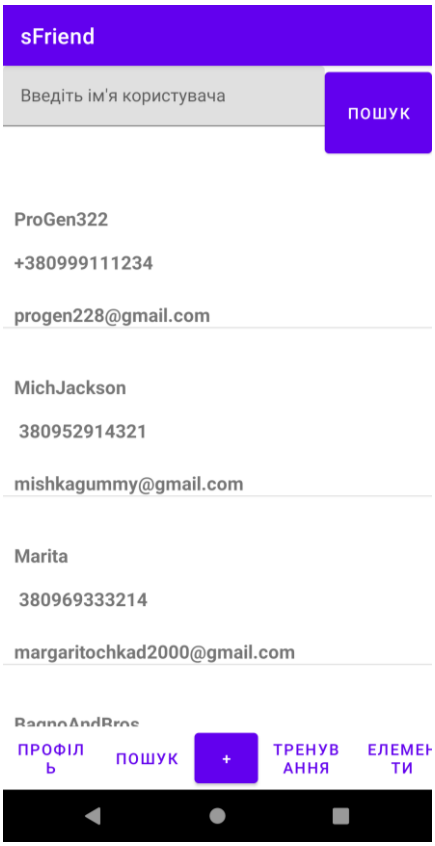


Рисунок 2.5 Інтерфейс сторінки «Пошук»

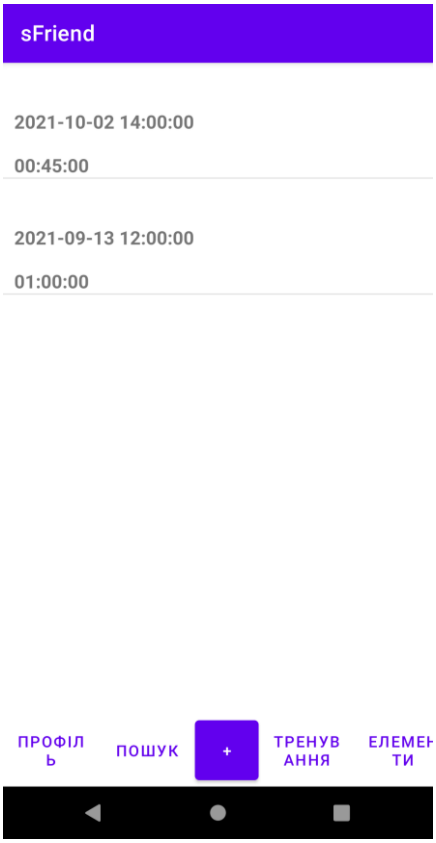


Рисунок 2.6 Інтерфейс сторінки «Список тренувань»


Рисунок 2.7 Інтерфейс сторінки «Тренування»

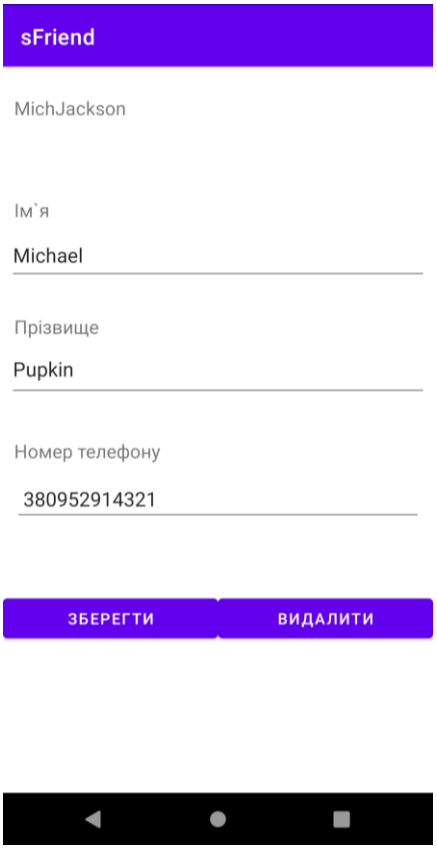


Рисунок 2.8 Інтерфейс сторінки «Редагування користувача»

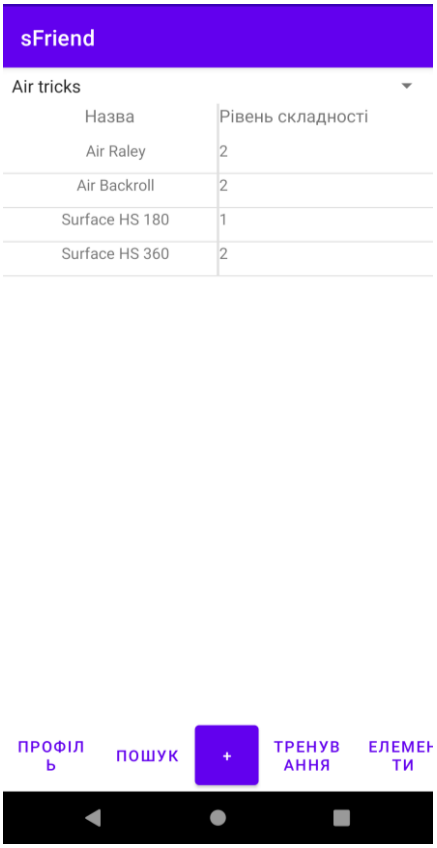


Рисунок 2.9 Інтерфейс сторінки «Список елементів»

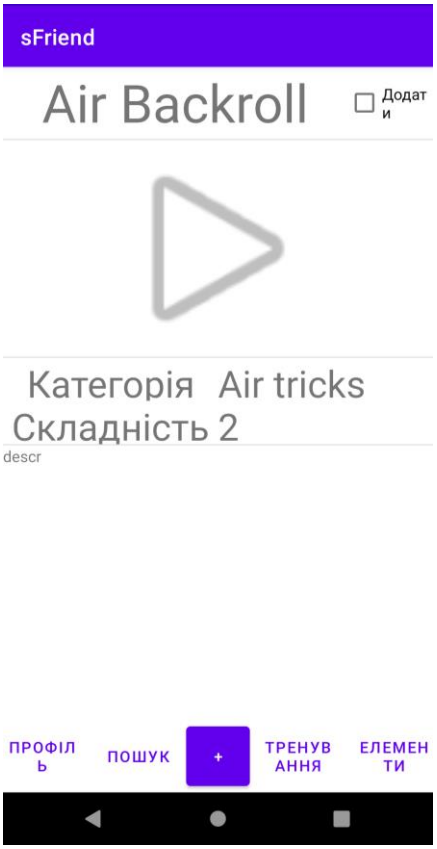


Рисунок 2.10 Інтерфейс сторінки «Елемент»

2.2.3. Проєктування динаміки системи

Динаміка системи об‘єктно-орієнтованих систем проєктується на основі аналізу застосування системи (Use Case Diagram) та первинної класифікації.[1]

Діаграми послідовностей використовуються для уточнення діаграм прецедентів більш детального опису логіки сценаріїв використання.[2]

Розроблена діаграма послідовності для прецеденту «Додати елемент для вивчення» рисунок 2.11

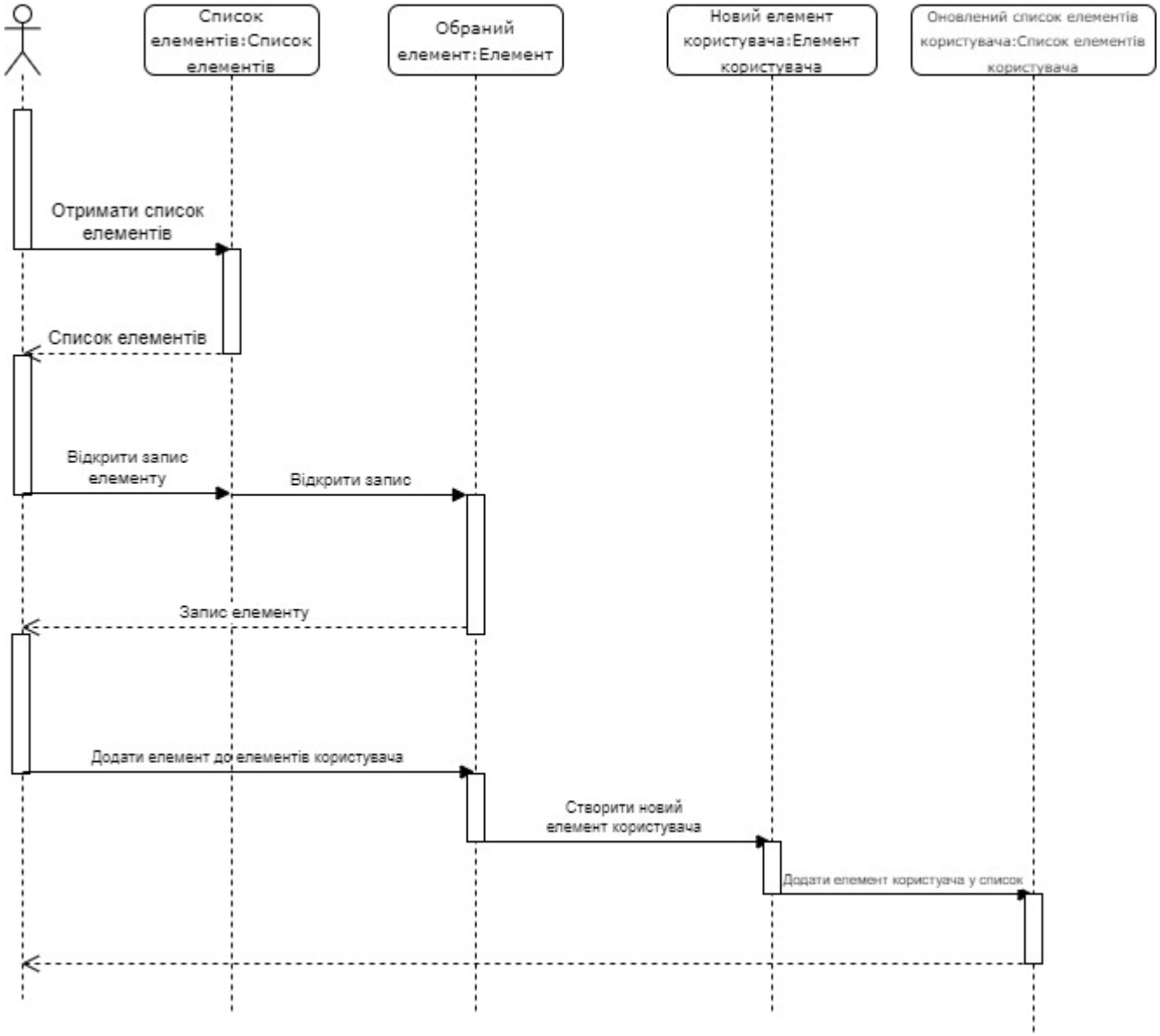


Рисунок 2.11 Діаграма послідовності для прецеденту «Додати елемент для вивчення»

2.2.4. Проєктування системи на фізичному рівні

Виходячи з діаграми сутностей (рис. 2.2) та функціональних вимог була побудована діаграма артефактів серверної та клієнтської частин додатку. Вона зображена на рисунку 2.12

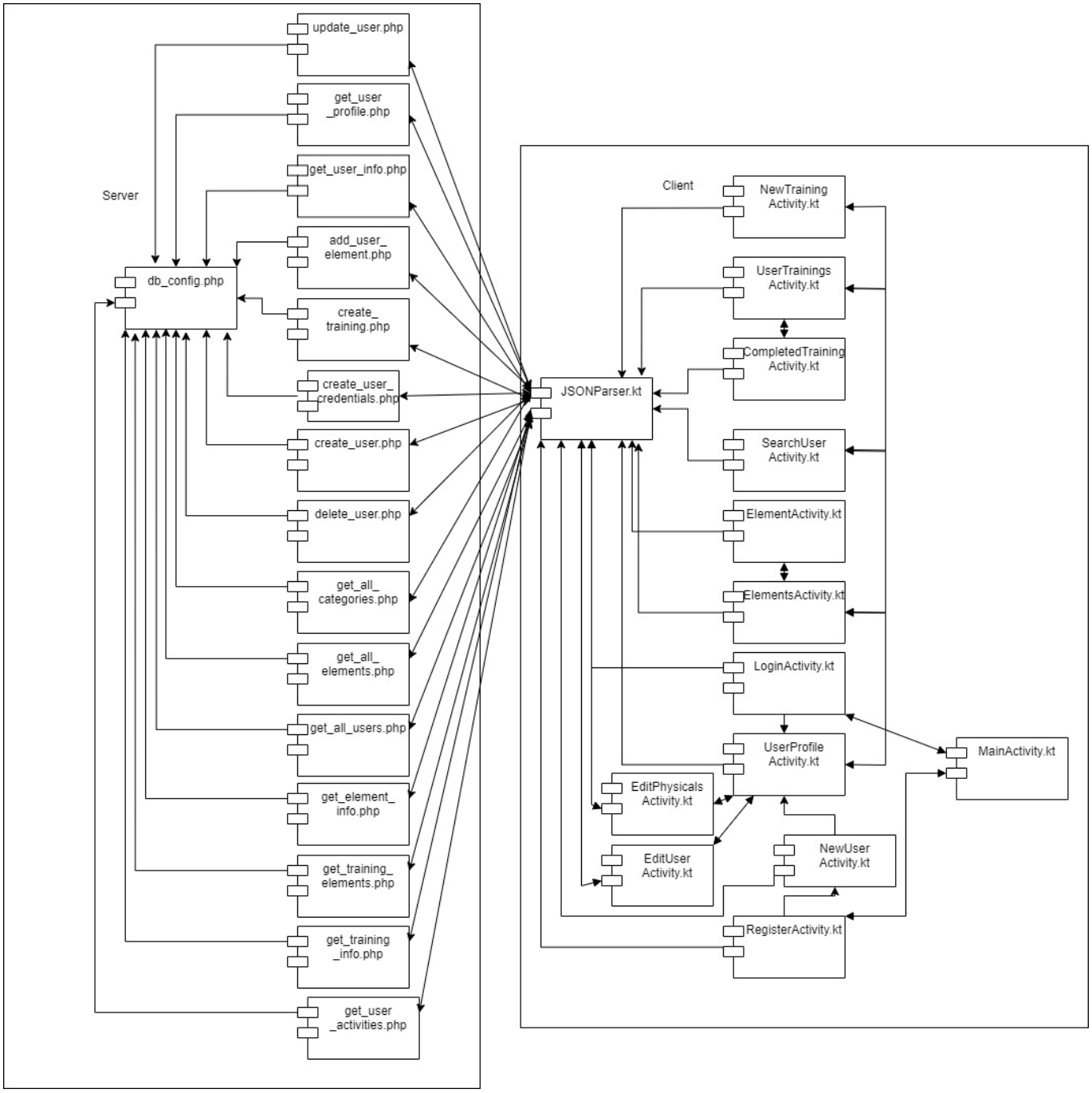


Рисунок 2.12 Діаграма артефактів(компонентів)

2.2.5. Проєктування бази даних

В програмі передбачено використання таблиць MySQL, на основі діаграми сутностей (рис. 2.2) було побудовано таблиці бази даних (таблиці 2.1-2.26)

Таблиця 2.1 – Таблиця «base_factor»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| base_set_id | Ідентифікатор місця тренування | int(11) | + |
| factor_id | Ідентифікатор фактора | int(11) | + |
| factor_value | величина фактора | double | - |

Таблиця 2.2 – Таблиця «base_set»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| base_set_id | Ідентифікатор запису набору обставин для тренування | int(11) | + |
| base_set_name | Назва набору | varchar(50) | - |
| location | Місцезнаходження | varchar(100) | - |

Таблиця 2.3 – Таблиця «base_set_obstacles»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| base_set_id | Ідентифікатор запису набору обставин для тренування | int(11) | + |
| obstacle_id | Ідентифікатор снаряду | int(11) | + |
| position | Розміщення снаряду | varchar(10) | - |

Таблиця 2.4 – Таблиця «basic_moves»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| move_id | Ідентифікатор руху | int(11) | + |
| move_name | Назва руху | varchar(50) | - |
| move_type | Тип руху | varchar(50) | - |
| move_level | Рівень складності | int(11) | - |

Таблиця 2.5 – Таблиця «categories»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| category_id | Ідентифікатор категорії | int(11) | + |
| name | Назва категорії | varchar(50) | - |

Таблиця 2.6 – Таблиця «elements»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| element_id | Ідентифікатор елементу | int(11) | + |
| el_name | Назва елементу | varchar(50) | - |
| start_mov_id | Ідентифікатор початкового руху | int(11) | + |
| h_level | Рівень складності | int(11) | - |
| video_link | Посилання на відео | varchar(100) | - |
| descr | Опис | text | - |
| parts_count | Кількість складових рухів | int(11) | - |
| category_id | Ідентифікатор категорії | int(11) | + |

Таблиця 2.7 – Таблиця «element_moves»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| element_id | Ідентифікатор елементу | int(11) | + |
| move_id | Ідентифікатор руху | int(11) | + |
| index_number | Порядковий номер руху в елементі | int(11) | + |

Таблиця 2.8 – Таблиця «environment»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| enviroment_id | Ідентифікатор тренувального середовища | int(11) | + |
| training_set_id | Ідентифікатор запису набору інвентаря для тренування | int(11) | + |
| base_set_id | Ідентифікатор запису набору обставин для тренування | int(11) | + |

Таблиця 2.9 – Таблиця «equipment»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| equipment_id | Ідентифікатор інвентарю | int(11) | + |
| equipment_type | Тип інвентарю | varchar(50) | - |
| equipment_size | Розмір | int(11) | - |

Продовження таблиці 2.9

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| equipment_name | Назва | varchar(50) | - |
| equipment_model | Модель | varchar(50) | - |
| equipment_prod_year | Рік випуску | int(11) | - |
| production_company_id | Ідентифікатор компанії-виробника | int(11) | + |
| equipment_description | Опис | varchar(400) | - |

Таблиця 2.10 – Таблиця «equipment_training_set»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| equipment_id | Ідентифікатор інвентарю | int(11) | + |
| training_set_id | Ідентифікатор тренувального комплекту | int(11) | + |

Таблиця 2.11 – Таблиця «factors»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| factor_id | Ідентифікатор фактору | int(11) | + |
| factor_name | Назва | varchar(50) | - |
| factor_type | Тип | varchar(50) | - |
| impact_level | Рівень впливу | double | - |

Таблиця 2.12 – Таблиця «genders»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| gender_id | Ідентифікатор статі | int(11) | + |
| name | Назва | varchar(50) | - |

Таблиця 2.13 – Таблиця «manual»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| post_id | Ідентифікатор навчального матеріалу | int(11) | + |
| title | Заголовок | varchar(100) | - |
| element_id | Ідентифікатор елемента | int(11) | + |
| author_id | Ідентифікатор користувача-автора | int(11) | + |

Продовження таблиці 2.13

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| description | Опис | text | - |
| video_link | Посилання на відео | varchar(100) | - |

Таблиця 2.14 – Таблиця «obstacles»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| obstacle_id | Ідентифікатор снаряду | int(11) | + |
| obstacle_name | Назва | varchar(50) | - |
| obtacle_type | Тип | varchar(50) | - |
| difficulty_level | Рівень складності | int(11) | - |
| obstacle_description | Опис | text | - |
| production_company_id | Ідентифікатор компнії-виробника | int(11) | + |

Таблиця 2.15 – Таблиця «plan_element»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| element_id | Ідентифікатор елементу | int(11) | + |
| plan_id | Ідентифікатор плану | int(11) | + |
| el_index | Порядковий номер елементу в плані | int(11) | + |

Таблиця 2.16 – Таблиця «production_companies»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| production_company_id | Ідентифікатор компанії-виробника | int(11) | + |
| product_company_name | Назва компанії | varchar(100) | - |
| location | Місцезнаходження | varchar(100) | - |
| product_company_email | Електронна пошта | varchar(50) | - |

Таблиця 2.17 – Таблиця «training»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| activity_id | Ідентифікатор тренування | int(11) | + |
| plan_id | Ідентифікатор плану тренування | int(11) | + |
| env_id | Ідентифікатор середовища | int(11) | + |

Продовження таблиці 2.18

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| date_time | Дата та час початку тренування | datetime | - |
| duration | Тривалість тренування | time | - |

Таблиця 2.18 – Таблиця «training_element»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| training_element_id | Порядковий номер елемента в тренуванні | int(11) | + |
| training_id | Ідентифікатор тренування | int(11) | + |
| element_id | Ідентифікатор елементу | int(11) | + |

Таблиця 2.19 – Таблиця «training_plan»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| plan_id | Ідентифікатор плану | int(11) | + |
| plan_auth_id | Ідентифікатор автора плану | int(11) | + |
| plan_name | Назва плану | varchar(50) | - |

Таблиця 2.20 – Таблиця «training_set»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| training_set_id | Ідентифікатор запису набору інвентаря для тренування | int(11) | + |
| training_set_name | Назва набору | varchar(50) | - |

Таблиця 2.21 – Таблиця «users»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор користувача | int(11) | + |
| user_name | Логін користувача | varchar(50) | - |
| first_name | Ім’я користувача | varchar(50) | - |
| surname | Прізвище | varchar(50) | - |
| last_name | По-батькові | varchar(50) | - |
| birthdate | Дата народження | date | - |

Продовження таблиці 2.21

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| phone_num | Мобільний номер | char(13) | - |

Таблиця 2.22 – Таблиця «users_training»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор запису користувача | int(11) | + |
| activity_id | Ідентифікатор тренування | int(11) | + |

Таблиця 2.23 – Таблиця «user_credentials»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор запису набору інвентаря для тренування | int(11) | + |
| email | Назва набору | varchar(50) | + |
| password | Пароль | varchar(100) | + |

Таблиця 2.24 – Таблиця «user_element»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор запису користувача | int(11) | + |
| element_id | Ідентифікатор елементу | int(11) | + |
| progress | Прогрес | int(11) | - |

Таблиця 2.25 – Таблиця «user_parameters»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор запису набору інвентаря для тренування | int(11) | + |
| user_parameters_id | Ідентифікатор запису параметрів | int(11) | + |
| gender_id | Ідентифікатор статі | int(11) | + |
| height | Зріст | double(6,3) | - |
| weight | Вага | double(6,3) | |

Таблиця 2.26 – Таблиця «user_training_plan»

| Назва поля | Опис поля | Тип поля | Ключ |
|---|---|---|---|
| user_id | Ідентифікатор запису набору інвентаря для тренування | int(11) | + |
| plan_id | Ідентифікатор плану | int(11) | + |

Схема зв’язків таблиць зображена на рис. 2.13

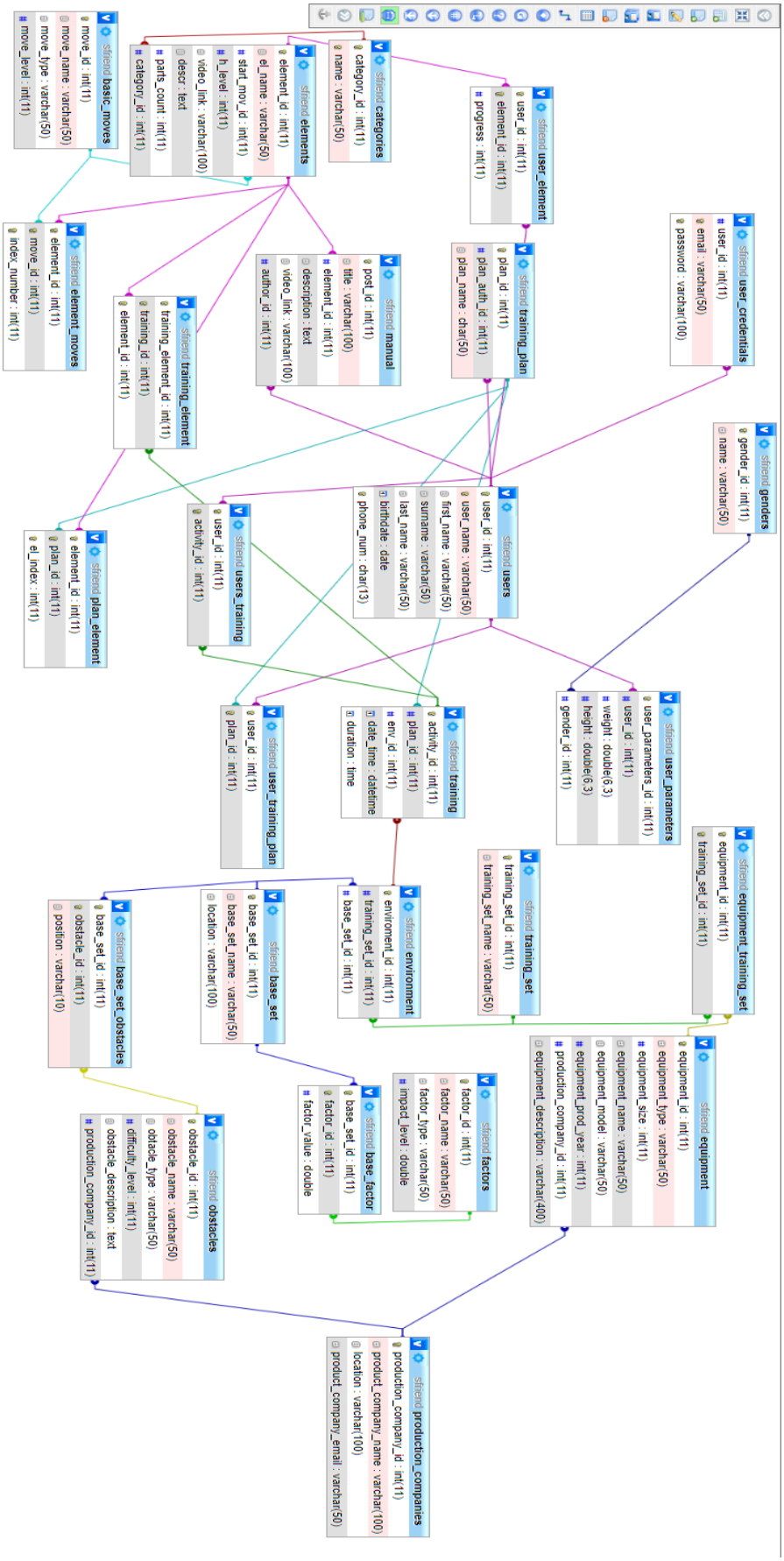


Рисунок 2.13 Схема зв’язків таблиць

## 2.3.Висновки

В цьому розділі описано структуру додатку та бази даних. Реалізовано інтерфейс користувача та базу даних. Підготовлена основа для розробки функціональності.

# 3. РОЗРОБКА ПРОГРАМИ

Для розробки мобільного додатку було обрано мову програмування Kotlin. Серверна частина додатку написана мовою PHP. Основні алгоритми сконструйовані навколо передачі даних між сервером та клієнтським додатком.

Для додавання запису додаток генерує порожню форму та виводить її користувачеві. Після заповнення на збереження дані з форми збираються у хеш таблицю та передаються в модуль формування запиту на сервер, де в залежності від форми створюється певне посилання з методом «POST» та потрібними параметрами. За сформованим посиланням виконується запит, в результаті якого клієнтський додаток отримує відповідь щодо успішності запиту, а сервер створює потрібний запис в базі даних.

Для наповнення інтерфейсу даними використовується то самий модуль, тільки дані він отримує від статичних елементів інтерфейсу, які користувач не редагує напряму, наприклад кнопок чи сторінок під час їх завантаження. Отримавши посилання та потрібні параметри модуль відправляє на сервер запит «GET», у відповідь на який отримує JSON файл з інформацією про один або декілька записів. Сервер у свою чергу, отримавши запит «GET» робить вибірку з бази даних або за окремим об’єктом, або одразу за декількома однотипними записами і повертає файл зі списком знайдених записів.

## 3.1.Опис алгоритмічних структур

Алгоритм (algorithm) – це система формальних правил, які чітко й однозначно окреслюють послідовність дій обчислювального процесу від початкових даних до шуканого результату[1].

Для деталізації алгоритмів потрібних у додатку було створено декілька етапів їх опису.

### 3.1.1. Графічний опис алгоритмів

Для графічного зображення прикладу алгоритму була побудована блок-схема (рис. 3.1)

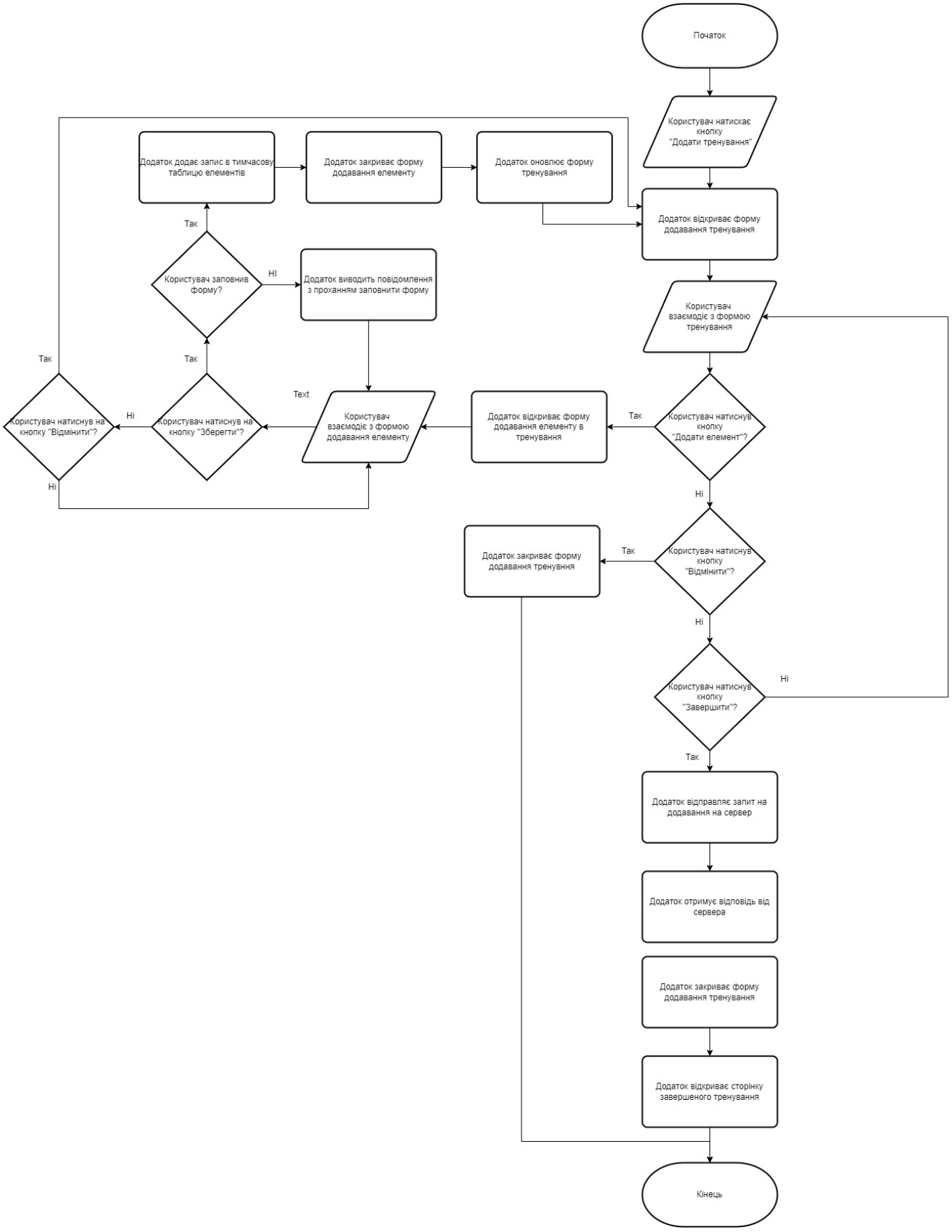


Рисунок 3.1 Блок-схема алгоритму додавання тренування

## 3.2.Метод покрокової деталізації

Метод покрокової деталізації – один з методів декомпозиції. Його суть полягає у розробці алгоритму зверху вниз. На першому кроці формулюється задача в цілому, визначаються формати вхідних та вихідних даних, складається якомога повніша специфікацію. На другому виділяються основні під задачі. На наступних кроках ці під задачі розбиваються ще на кілька під задач. Розбиття виконується доти, доки програміст не матиме повного уявлення як писати програмний код.[1]

Для деталізації було обрано відкривання запису з інформацією про елемент.

Перепишемо подану задачу у вигляді специфікації вхідних та вихідних даних.

Вхід:

- натискання користувачем на кнопку відкриття розділу елементів;
- натискання користувачем на строку з потрібним елементом.

Вихід:

- виведення на екран інформації про обраний елемент.

Розіб’ємо задачу на під задачі:

1) Виведення на екран розділу зі списком елементів
    a. Запит до серверу для отримання всіх записів елементів
    b. Отримання файлу зі списком елементів
    c. Формування блоків списку елементів
    d. Виведення блоків на екран у вигляді вертикального списку
2) Обрання потрібного елементу
    a. Зчитування обраного блоку зі списку
    b. Запит до серверу на отримання деталей про обраний елемент
    c. Отримання файлу з полями елементу
    d. Формування сторінки елементу
    e. Виведення на екран сторінки з інформацією про елемент

Представлення цієї ж задачі можна побудувати у вигляді дерева (рис.3.2)

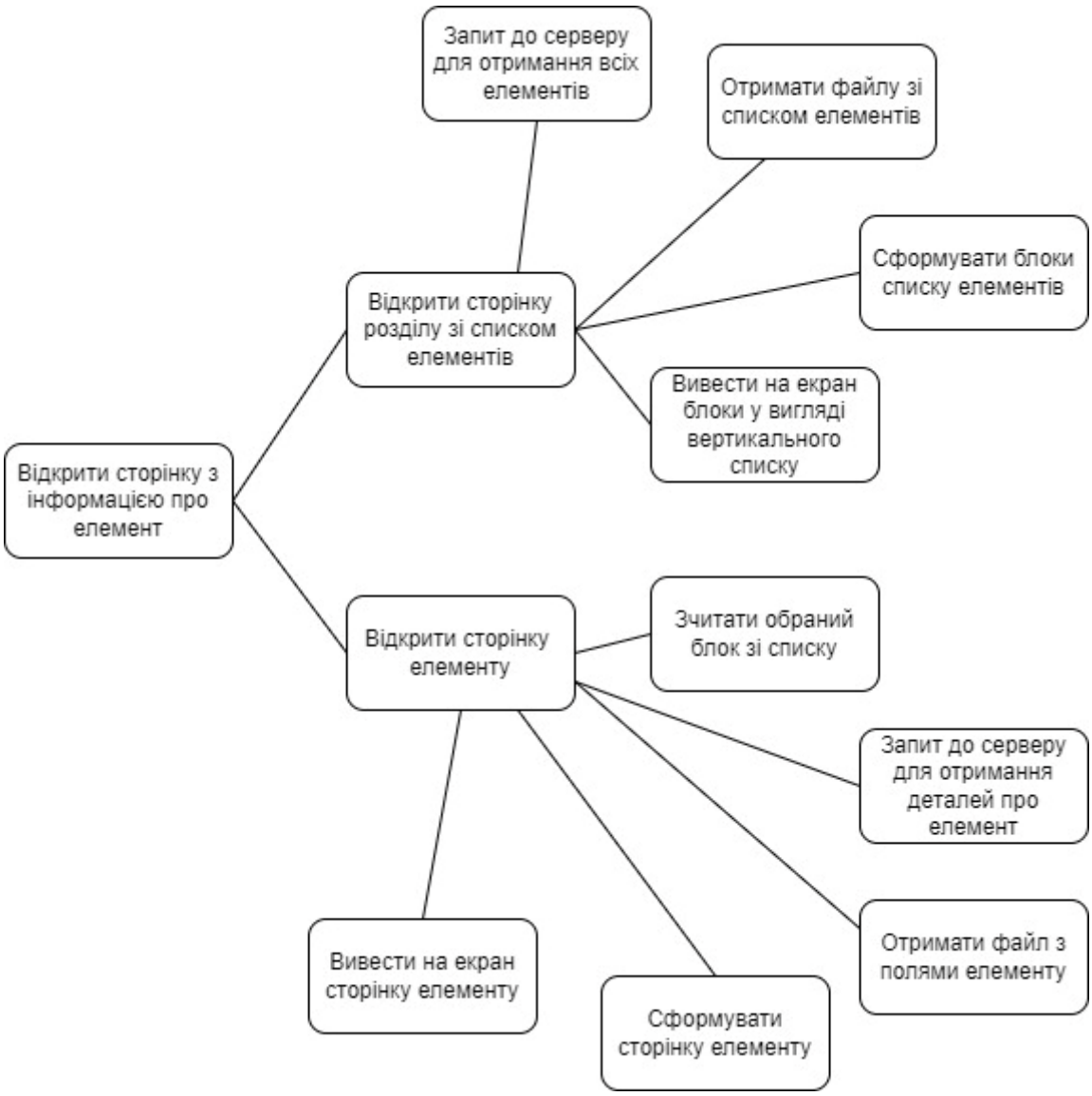


Рисунок 3.2 Дерево покрокової деталізації

## 3.3.Висновки

В цьому розділі було сформовано і описано приклади основних алгоритмів, які будуть реалізовані в мобільному додатку. Різні варіанти представлення зручно використовувати для деталізації різних видів алгоритмів, але, незалежно від обраного варіанту опису алгоритму, цей етап робить розробку більш контрольованою та простою.

# 4. Тестування та налагодження

Тестування – процес виконання програми (або її частини) з метою перевірки її відповідності встановленим вимогам, зазначеним у специфікації, і виявлення дефектів (помилок).[1]

## 4.1. Тестування методом білої скриньки

Проведемо тестування методу makeHttpRequest, який є основним методом модуля JSONParser.kt

### 4.1.1. Специфікація

Метод makeHttpRequest(url:String?,method:String,

params:MutableList<Pair<String, String?>>) відповідає за виконання запитів до серверу з поверненням відповіді про стан запиту та отриманих даних.

Вхід: в явному вигляді приймає посилання для запиту url типу String, метод запиту method типу String та параметрів запиту params типу MutableList<Pair<String, String?>>, в неявному вигляді параметрів не приймає.

Вихід: в явному вигляді повертає об’єкт типу JSONObject з результатом запиту.

Текст методу наведено на рис. 4.1

```
fun makeHttpRequest(url: String?, method: String, params: MutableList<Pair<String, String?>>): JSONObject? {
    var currentUrl = url
    try {//Оператор 1
        if(method === "GET") {//Оператор 2
            if (params.size>0) {//Оператор 3
                currentUrl+="?"
                for (param in params) {//Оператор 4
                    currentUrl+=param.first+ "="+param.second+"&"
                }
            }
            val connection = URL(currentUrl).openConnection()
            connection.setRequestProperty("Accept-Charset", "UTF-8")
            `is` = connection.getInputStream()
        }
        if (method === "POST") {//Оператор 5
            val connection = URL(currentUrl).openConnection()
            connection.setRequestProperty("Accept-Charset", "UTF-8")
            var postData = ""
            if (params != null) {//Оператор 6
                for (param in params) {//Оператор 7
                    if (postData !== "") {//Оператор 8
                        postData += "&"
                    }
                    postData += param?.first + "=" + param?.second
                }
            }
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")
            connection.setRequestProperty("Content-Length", postData.length.toString())
            connection.doOutput = true
            DataOutputStream(connection.getOutputStream()).use { it.writeBytes(postData) }
            `is` = connection.getInputStream()
        }
    } catch (e: UnsupportedEncodingException) {e.printStackTrace()}//Оператор 8
    catch (e: IOException) { e.printStackTrace()}//Оператор 9
    try {//Оператор 10
        val reader = BufferedReader(InputStreamReader(`is`, "iso-8859-1"), 8)
        val sb = StringBuilder()
        var line: String?
        while (reader.readLine().also { line = it } != null) {//Оператор 11
            sb.append(
                """
                $line

                """.trimIndent()
            )
        }
        `is`?.close()
        json = sb.toString()
    } catch (e: Exception) {Log.e("Buffer Error", "Error converting result $e")}//Оператор 12
    try {//Оператор 13
        jObj = JSONObject(json)
    } catch (e: JSONException) { Log.e("JSON Parser", "Error parsing data $e")}//Оператор 14
    return jObj
}
```

Рисунок 4.1 Метод makeHttpRequest

### 4.1.2. Тестування методу

Тести наведені у таблиці 4.1

Таблиця 4.1 Тести для методу makeHttpRequest

| № | Тест | Вхідні дані | Вихідні дані |
|---|---|---|---|
| 1 | Запит на отримання всіх користувачів. | url = «http://192.168.0.103/www/php/ get_all_users.php» method = «GET» params = NULL | {"users":[{"user_id":"1","user_name":"ProGen322","first_name":"Evgen","surname":"Proliskov","phone_num":"+380999111234","email":"progen228@gmail.com"},{"user_id":"2","user_name":"MichJackson","first_name":"Michael","surname":"Pupkin","phone_num":"380952914321","email":"mishkagummy@gmail.com"} ],"success":1} |
| 2 | Запит на отримання запису користувача. | url = «http://192.168.0.103/www/php/ get_user_profile.php» method = «GET» params = [(user_id,1)] | {"success":1,"user":[{"user_id":null,"user_name":null,"first_name":"Evgen","surname":"Proliskov","phone_num":"+380999111234","weight":"90.000","height":"182.000","birthdate":"1983-05-10"}]} |
| 3 | Запит на отримання запису користувача без передачі параметру. | url = «http://192.168.0.103/www/php/ get_user_profile.php» method = «GET» params = NULL | {"success":0,"message":"Required field(s) is missing"} |

Продовження таблиці 4.1

| № | Тест | Вхідні дані | Вихідні дані |
|---|---|---|---|
| 4 | Запит на отримання запису користувача з неіснуючим значенням параметру | url = «http://192.168.0.103/www/php/ get_user_profile.php» method = «GET» params = [(user_id,9)] | {"success":0,"message":"No user found"} |
| 5 | Запит на зміну запису користувача | url = «http://192.168.0.103/www/php/ update_user.php» method = «POST» params = [(user_id,2),(user_name,MicgJackson),(first_name,Misha),(surname,Pupkin),(phone_num,380952914321)] | {"success":1,"message":"User successfully updated."} |

4.1.3. Таблиця покриття операторів: табл. 4.2

Таблиця 4.2 Покриття операторів

| Оператор / Тест | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | * | * | | | | | | | | | * | * | | * | |
| 2 | * | * | * | * | | | | | | | * | * | | * | |
| 3 | * | * | | | | | | | | | * | * | | * | |
| 4 | * | * | * | * | | | | | | | * | * | | * | |
| 5 | * | | | | * | * | * | * | | | * | * | | * | |

4.1.4. Таблиця покриття умов: табл. 4.3

Таблиця 4.3 Покриття умов

| Умова | method === "GET" | | params.size>0 | | method === "POST" | | params != null | | postData !== "" | | reader.readLine().also { line = it } != null | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Тест | + | - | + | - | + | - | + | - | + | - | + | - |
| 1 | * | | * | | | * | | | | | * | |
| 2 | * | | | * | | * | | | | | * | |
| 3 | * | | * | | | * | | | | | * | |
| 4 | * | | | * | | * | | | | | * | |
| 5 | | * | | | * | | * | | * | * | * | |

# ВИСНОВКИ

В ході розробки дипломного прєкту було закріплено і використано набуті під час навчання знання про проєктування та розробку програмних застосунків.

Був проведений аналіз предметної області та визначені основні потреби потенційних користувачів. Розроблені сценарії та дизайн додатку.

При проєктуванні системи було описано вимоги до вихідного продукту, розроблена функціональність та побудована структура майбутнього додатку.

Для розробки було обрано мови Kotlin та PHP. Було пройдено повний процес розробки мобільного Android-додатку з вивченням механізмів внутрішньої взаємодії. Реалізована база даних та серверний додаток, який виконує операції над нею. При поєднанні двох частин додатку було вивчено механізм клієнт-серверної взаємодії.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп’ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с.

2. Майерс, Г., Баджетт, Т., Сандлер К. Искусство тестирования программ, 3-е изд.: пер. с англ. – М.: ООО “И.Д. Вильямс”, 2012. – 272 с.:ил.

3. Інженерія програмного забезпечення [Текст] : навчальний посібник / В. І. Шинкаренко, О. В. Горбова, О. П. Іванов, В. О. Андрющенко, В. Я. Нечай; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Дніпро, 2019. – 140 с.

4. Програмне забезпечення систем [Текст]: методичні вказівки до виконання дипломного проекту / уклад. В. В. Скалозуб, В. І. Шинкаренко, В. О. Андрющенко, Ю. М. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2013. – 41 с.

**Додаток А**

Код програми:

Мобільний додаток:

CompletedTrainingActivity.kt:

```
package com.example.sfriend

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors

class CompletedTrainingActivity : AppCompatActivity(){
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())



    private var date_value: TextView? = null
    private var duration_value: TextView? = null
    private var elements_count_value: TextView? = null
```

```
        // JSON параметры
        private val TAG_SUCCESS = "success"
        private val TAG_TRAINING_ID = "training_id"
        private val TAG_TRAINING = "training"
        private val TAG_ELEMENT = "element"
        private val TAG_DATE = "date_value"
        private val TAG_DURATION = "duration_value"
        private val TAG_ELEMENTS_COUNT = "elements_count_value"
        private val TAG_ELEMENT_NUBBER = "element_number"
        private val TAG_ELEMENT_NAME = "element_name"
        private var elementsList: ArrayList<HashMap<String, String>>? = null


        private var training_id = ""

        private var jsonParser = JSONParser()

        // url для получения одного продукта
        private var urlTrainingDetails = ""
        private var urlTrainingElements = ""



        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            urlTrainingDetails                                                    =
getString(R.string.OpenServerIp)+"get_training_info.php"
```

```
        urlTrainingElements                                                    =
getString(R.string.OpenServerIp)+"get_training_elements.php"
        setContentView(R.layout.completed_training_layout)
        val i = intent
        training_id = i.getStringExtra(TAG_TRAINING_ID)!!

        elementsList = ArrayList()



        // Получение полной информации о продукте в фоновом потоке
        getTrainingDetails()
        loadTrainingElements()
    }

    private fun getTrainingDetails() {
        myExecutor.execute {
            val success: Int
            try {
                // Список параметров
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair("training_id", training_id) )

                // получаем продукт по HTTP запросу
                val json: JSONObject? =
                    jsonParser.makeHttpRequest(urlTrainingDetails, "GET", params)
                Log.d("Single Training Details", json.toString())
                success = json!!.getInt(TAG_SUCCESS)
                if (success == 1) {
                    // Успешно получинна детальная информация о продукте
```

```
                    val trainingObj = json.getJSONArray(TAG_TRAINING)

                    // получаем первый обьект с JSON Array
                    val training = trainingObj.getJSONObject(0)

                    date_value = findViewById<View>(R.id.date_value) as TextView
                    duration_value = findViewById<View>(R.id.duration_value) as
TextView

                    myHandler.post {
                        date_value!!.text = training.getString(TAG_DATE)
                        duration_value!!.text = training.getString(TAG_DURATION)
                    }
                } else {
                    // продукт с pid не найден
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    }
    private fun loadTrainingElements(){
        myExecutor.execute {
            // Будет хранить параметры
            val params: MutableList<Pair<String,String?>> = ArrayList()
            val i = intent
            params.add(Pair(TAG_TRAINING_ID,
i.getStringExtra(TAG_TRAINING_ID)))
            // получаем JSON строк с URL
```

```
                val json: JSONObject? = jsonParser.makeHttpRequest(urlTrainingElements, "GET", params)
                Log.d("All Elements: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val elements = json.getJSONArray(TAG_ELEMENT)

                        // перебор всех продуктов
                        for (i in 0 until elements.length()) {
                            val c: JSONObject = elements.getJSONObject(i)

                            // Сохраняем каждый json елемент в переменную
                            val element_name = c.getString(TAG_ELEMENT_NAME)
                            val element_number = c.getString(TAG_ELEMENT_NUBBER)
                            val training_id = c.getString(TAG_TRAINING_ID)

                            // Создаем новый HashMap
                            val map = HashMap<String, String>()

                            // добавляем каждый елемент в HashMap ключ => значение
                            map[TAG_ELEMENT_NAME] = element_name
                            map[TAG_ELEMENT_NUBBER] = element_number
                            map[TAG_TRAINING_ID] = training_id

                            // добавляем HashList в ArrayList
```

```
                    elementsList?.add(map)
                }
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
        myHandler.post {
            // обновляем UI форму в фоновом потоке
            runOnUiThread {
                val adapter: ListAdapter = SimpleAdapter(
                    this@CompletedTrainingActivity, elementsList,
                    R.layout.training_element_item, arrayOf(
                        TAG_ELEMENT_NAME,
                        TAG_ELEMENT_NUBBER,
                        TAG_TRAINING_ID
                    ), intArrayOf(R.id.element_name, R.id.element_number,
R.id.training_id)
                )
                // обновляем listview
                val listView = findViewById<View>(R.id.list) as ListView
                listView.adapter = adapter
            }
        }
    }
}

    private var resultLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
```

```
            // There are no request codes
            val data: Intent? = result.data
            finish()
            startActivity(data)
        }
    }
}
```

EditPhysicalActivity.kt

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors
import android.os.Handler
import android.os.Looper


class EditPhysicalsActivity : AppCompatActivity() {
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())
    private var inputWeight: EditText? = null
```

```
private var inputHeight: EditText? = null
private var btnSave: Button? = null

private var userId: String? = null

private var jsonParser = JSONParser()

// url для получения одного продукта
private var urlUserDetails = ""

// url для обновления продукта
private var urlUpdateUser = ""

private val TAG_SUCCESS = "success"
private val TAG_USER = "user"
private val TAG_USER_ID = "user_id"
private val TAG_USERNAME = "user_name"
private val TAG_FIRST_NAME = "first_name"


override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    urlUserDetails = getString(R.string.OpenServerIp)+"get_user_phys.php"
    urlUpdateUser = getString(R.string.OpenServerIp)+"update_user_phys.php"
    setContentView(R.layout.activity_edit_user)
    btnSave = findViewById<View>(R.id.btnSave) as Button

    // показываем форму про детальную информацию о продукте
    val i = intent
```

```
        // получаем id продукта (pid) с формы
        userId = i.getStringExtra(TAG_USER_ID)

        // Получение полной информации о продукте в фоновом потоке
        getUserDetails()

        // обрабочик на кнопку сохранение продукта
        btnSave!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            saveUserDetails()
        }

    }

    private fun getUserDetails() {
        myExecutor.execute {
            val success: Int
            try {
                // Список параметров
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair("user_id", userId) )

                // получаем продукт по HTTP запросу
                val json: JSONObject? =
                    jsonParser.makeHttpRequest(urlUserDetails, "GET", params)
                Log.d("Single User Details", json.toString())
                success = json!!.getInt(TAG_SUCCESS)
                if (success == 1) {
```

```
                // Успешно получинна детальная информация о продукте
                val userObj = json.getJSONArray(TAG_USER)

                // получаем первый обьект с JSON Array
                val user = userObj.getJSONObject(0)

                inputHeight = findViewById<View>(R.id.input_height) as EditText
                inputWeight = findViewById<View>(R.id.input_weight) as EditText
                myHandler.post {
                    inputHeight!!.setText ( user.getString("height"))
                    inputWeight!!.setText(user.getString("weight"))
                }
            } else {
                // продукт с pid не найден
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }

    }
}

private fun saveUserDetails() {
    myExecutor.execute {
        // получаем обновленные данные с EditTexts
        val height: String = inputHeight?.text.toString()
        val weight: String = inputWeight?.text.toString()
        // формируем параметры
        val params: MutableList<Pair<String, String?>> = ArrayList()
```

```
                params.add(Pair(TAG_USER_ID, userId))
                params.add(Pair("height", height))
                params.add(Pair("weight", weight))
                // отправляем измененные данные через http запрос
                val json: JSONObject? = jsonParser.makeHttpRequest(urlUpdateUser,
"POST", params)

                // проверяем json success тег
                try {
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт удачно обнавлён
                        val i: Intent = intent
                        // отправляем результирующий код 100 чтобы сообщить об
обновлении продукта
                        setResult(100, i)
                        finish()
                    } else {
                        // продукт не обновлен
                    }
                } catch (e: JSONException) {
                    e.printStackTrace()
                }
            }
        }

    }
```

EditUserActivity.kt

```
package com.example.sfriend
```

```
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors
import android.os.Handler
import android.os.Looper


class EditUserActivity : AppCompatActivity() {
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())
    private var userName: TextView? = null
    private var inputFirstName: EditText? = null
    private var inputSurname: EditText? = null
    //private var inputLastName: EditText? = null
    private var inputPhoneNum: EditText? = null
    //private var inputEmail: EditText? = null
    private var btnSave: Button? = null
    private var btnDelete: Button? = null

    private var userId: String? = null
```

```
private var jsonParser = JSONParser()

// url для получения одного продукта
private var urlUserDetails = ""

// url для обновления продукта
private var urlUpdateUser = ""

// url для удаления продукта
private var urlDeleteUser = ""

// JSON параметры
private val TAG_SUCCESS = "success"
private val TAG_USER = "user"
private val TAG_USER_ID = "user_id"
private val TAG_USERNAME = "user_name"
private val TAG_FIRST_NAME = "first_name"
private val TAG_SURNAME = "surname"
private val TAG_LAST_NAME = "last_name"
private val TAG_PHONE_NUM = "phone_num"
private val TAG_EMAIL = "email"

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    urlUserDetails = getString(R.string.OpenServerIp)+"get_user_info.php"
    urlDeleteUser = getString(R.string.OpenServerIp)+"delete_user.php"
    urlUpdateUser = getString(R.string.OpenServerIp)+"update_user.php"
    setContentView(R.layout.activity_edit_user)
```

```
        btnSave = findViewById<View>(R.id.btnSave) as Button
        btnDelete = findViewById<View>(R.id.btnDelete) as Button

        // показываем форму про детальную информацию о продукте
        val i = intent

        // получаем id продукта (pid) с формы
        userId = i.getStringExtra(TAG_USER_ID)

        // Получение полной информации о продукте в фоновом потоке
        getUserDetails()

        // обрабочик на кнопку сохранение продукта
        btnSave!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            saveUserDetails()
        }

        // обработчик на кнопку удаление продукта
        btnDelete!!.setOnClickListener { // удалем продукт в фоновом потоке
            deleteUser()
        }
    }

    private fun getUserDetails() {
        myExecutor.execute {
            val success: Int
            try {
                // Список параметров
```

```
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair("user_id", userId) )

                // получаем продукт по HTTP запросу
                val json: JSONObject? =
                   jsonParser.makeHttpRequest(urlUserDetails, "GET", params)
                Log.d("Single User Details", json.toString())
                success = json!!.getInt(TAG_SUCCESS)
                if (success == 1) {
                   // Успешно получинна детальная информация о продукте
                   val userObj = json.getJSONArray(TAG_USER)

                   // получаем первый обьект с JSON Array
                   val user = userObj.getJSONObject(0)

                   // продукт с pid найден
                   // Edit Text
                   userName = findViewById<View>(R.id.userName) as TextView
                   inputFirstName = findViewById<View>(R.id.inputFirstName) as
EditText
                   inputSurname = findViewById<View>(R.id.inputSurname) as
EditText
                   //inputLastName = findViewById<View>(R.id.inputLastName) as
EditText
                   inputPhoneNum = findViewById<View>(R.id.inputPhoneNumber)
as EditText
                   //inputEmail = findViewById<View>(R.id.inputEmail) as EditText

                   myHandler.post {
```

```
                userName!!.text = user.getString(TAG_USERNAME)
                inputFirstName!!.setText(user.getString(TAG_FIRST_NAME))
                inputSurname!!.setText(user.getString(TAG_SURNAME))
                //inputLastName!!.setText(user.getString(TAG_LAST_NAME))
                inputPhoneNum!!.setText(user.getString(TAG_PHONE_NUM))
            }
            // покаываем данные о продукте в EditText

            //inputEmail!!.setText(user.getString(TAG_EMAIL))
        } else {
            // продукт с pid не найден
        }
    } catch (e: JSONException) {
        e.printStackTrace()
    }

  }
}

private fun saveUserDetails() {
  myExecutor.execute {
    // получаем обновленные данные с EditTexts
    val username: String = userName?.text.toString()
    val firstName: String = inputFirstName?.text.toString()
    val surname: String = inputSurname?.text.toString()
    //val lastName: String = inputLastName?.text.toString()
    val phoneNum: String = inputPhoneNum?.text.toString()
    //val email: String = inputEmail?.text.toString()
```

```
                // формируем параметры
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair(TAG_USER_ID, userId))
                params.add(Pair(TAG_USERNAME, username))
                params.add(Pair(TAG_FIRST_NAME, firstName))
                params.add(Pair(TAG_SURNAME, surname))
                //params.add(Pair(TAG_LAST_NAME, lastName))
                params.add(Pair(TAG_PHONE_NUM, phoneNum))
                //params.add(Pair(TAG_EMAIL, email))

                // отправляем измененные данные через http запрос
                val json: JSONObject? = jsonParser.makeHttpRequest(urlUpdateUser,
"POST", params)

                // проверяем json success тег
                try {
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт удачно обнавлён
                        val i: Intent = intent
                        // отправляем результирующий код 100 чтобы сообщить об
обновлении продукта
                        setResult(100, i)
                        finish()
                    } else {
                        // продукт не обновлен
                    }
                } catch (e: JSONException) {
                    e.printStackTrace()
```

```
            }
        }
    }

    private fun deleteUser() {
        myExecutor.execute {
            val success: Int
            try {
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair("user_id", userId))

                // получение продукта используя HTTP запрос
                val json: JSONObject? =
                    jsonParser.makeHttpRequest(urlDeleteUser, "POST", params)
                Log.d("Delete User", json.toString())
                success = json!!.getInt(TAG_SUCCESS)
                if (success == 1) {
                    // Продукт удачно удален
                    val i: Intent = intent
                    // отправляем результирующий код 100 для уведомления об
удалении продукта
                    setResult(100, i)
                    finish()
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    }
```

```
}
```

ElementActivity.kt:

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.CheckBox
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors

class ElementActivity : AppCompatActivity(){
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())


    private var element_title: TextView? = null
    private var element_category: TextView? = null
    private var element_h_level: TextView? = null
    private var description: TextView? = null
    private var isAdded: CheckBox? = null
```

```
        // JSON параметры
        private val TAG_SUCCESS = "success"
        private val TAG_ELEMENT_ID = "element_id"
        private val TAG_ELEMENT = "element"
        private val TAG_ELEMENT_CATEGORY = "element_category"
        private val TAG_ELEMENT_TITLE = "element_title"
        private val TAG_ELEMENT_H_LEVEL = "element_h_level"
        private val TAG_DESCRIPTION = "description"


        private var element_id = ""

        private var jsonParser = JSONParser()

        // url для получения одного продукта
        private var urlElementDetails = ""



        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            urlElementDetails                                                    =
getString(R.string.OpenServerIp)+"get_element_info.php"
            setContentView(R.layout.element_layout)
            val i = intent
            element_id = i.getStringExtra(TAG_ELEMENT_ID)!!


            // Получение полной информации о продукте в фоновом потоке
```

```
        getElementDetails()
    }

    private fun getElementDetails() {
        myExecutor.execute {
            val success: Int
            try {
                // Список параметров
                val params: MutableList<Pair<String, String?>> = ArrayList()
                params.add(Pair("element_id", element_id) )

                // получаем продукт по HTTP запросу
                val json: JSONObject? =
                    jsonParser.makeHttpRequest(urlElementDetails, "GET", params)
                Log.d("Single Element Details", json.toString())
                success = json!!.getInt(TAG_SUCCESS)
                if (success == 1) {
                    // Успешно получинна детальная информация о продукте
                    val elementObj = json.getJSONArray(TAG_ELEMENT)

                    // получаем первый обьект с JSON Array
                    val element = elementObj.getJSONObject(0)

                    // продукт с pid найден
                    // Edit Text
                    element_title = findViewById<View>(R.id.element_title) as
TextView
                    element_category = findViewById<View>(R.id.element_category) as
TextView
```

```
                    element_h_level = findViewById<View>(R.id.element_h_level) as
TextView
                    description = findViewById<View>(R.id.description) as TextView

                    myHandler.post {
                        element_title!!.text = element.getString(TAG_ELEMENT_TITLE)
                        element_category!!.text =
element.getString(TAG_ELEMENT_CATEGORY)
                        element_h_level!!.text =
element.getString(TAG_ELEMENT_H_LEVEL)
                        description!!.text = element.getString(TAG_DESCRIPTION)
                    }
                    // покаываем данные о продукте в EditText

                } else {
                    // продукт с pid не найден
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    }
}
```

ElementsActivity.kt:

```
package com.example.sfriend

import android.app.Activity
import android.content.Intent
```

```
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors


class ElementsActivity : AppCompatActivity() {

    // Создаем JSON парсер
    private var jParser = JSONParser()

    private var elementsList: ArrayList<HashMap<String, String>>? = null
    private var categoriesList: MutableList<String> = mutableListOf<String>()
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())

    // url получения списка всех продуктов
    private var urlAllElements = ""
    private var urlAllCategories = ""

    // JSON Node names
    private val TAG_SUCCESS = "success"
```

```
        private val TAG_ELEMENTS = "elements"
        private val TAG_CATEGORIES = "categories"
        private val TAG_ELEMENT_ID = "element_id"
        private val TAG_ELEMENT_NAME = "element_name"
        private val TAG_NAME = "name"
        private val TAG_HLEVEL = "h_level"

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            urlAllElements = getString(R.string.OpenServerIp)+"get_all_elements.php"
            urlAllCategories                                                    =
getString(R.string.OpenServerIp)+"get_all_categories.php"
            setContentView(R.layout.activity_all_elements)

            // Hashmap for ListView
            elementsList = ArrayList()
            //categoriesList = mutableListOf<String>()

            // Загружаем продукты в фоновом потоке
            loadAllCategories()
            loadAllElements()

            // получаем ListView
            val lv: ListView = findViewById(R.id.list)
            lv.setOnItemClickListener { _, view, _, _ -> // getting values from selected
ListItem
                val element_id = (view.findViewById(R.id.element_id) as TextView).text
                    .toString()
```

```
                // Запускаем новый intent который покажет нам Activity
                val `in` = Intent(applicationContext, ElementActivity::class.java)
                // отправляем user_id в следующий activity
                `in`.putExtra(TAG_ELEMENT_ID, element_id)

                startActivity(`in`)
            }
        }

        private fun loadAllCategories(){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                val i = intent
                // получаем JSON строк с URL
                val json: JSONObject? = jParser.makeHttpRequest(urlAllCategories,
"GET", params)
                Log.d("All Categories: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val categories = json.getJSONArray(TAG_CATEGORIES)

                        // перебор всех продуктов
                        for (i in 0 until categories.length()) {
                            val c: JSONObject = categories.getJSONObject(i)
```

```
                    // Сохраняем каждый json елемент в переменную
                    val category_name = c.getString(TAG_NAME)

                    categoriesList.add(category_name)
                }
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
        myHandler.post {
            // обновляем UI форму в фоновом потоке
            runOnUiThread {
                val spinner = findViewById<Spinner>(R.id.spinner)
                // Создаем адаптер ArrayAdapter с помощью массива строк и стандартной разметки элемета spinner
                val adapter: ArrayAdapter<String> =
                    ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, categoriesList)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
                // Применяем адаптер к элементу spinner
                // Применяем адаптер к элементу spinner
                spinner.adapter = adapter
            }
        }
    }
}
```

```
        private fun loadAllElements(){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                val i = intent
                // получаем JSON строк с URL
                val json: JSONObject? = jParser.makeHttpRequest(urlAllElements,
"GET", params)
                Log.d("All Elements: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val elements = json.getJSONArray(TAG_ELEMENTS)

                        // перебор всех продуктов
                        for (i in 0 until elements.length()) {
                            val c: JSONObject = elements.getJSONObject(i)

                            // Сохраняем каждый json елемент в переменную
                            val element_id = c.getString(TAG_ELEMENT_ID)
                            val h_level = c.getString(TAG_HLEVEL)
                            val element_name = c.getString(TAG_ELEMENT_NAME)

                            // Создаем новый HashMap
                            val map = HashMap<String, String>()
```

```
                // добавляем каждый елемент в HashMap ключ => значение
                map[TAG_HLEVEL] = h_level
                map[TAG_ELEMENT_ID] = element_id
                map[TAG_ELEMENT_NAME] = element_name

                // добавляем HashList в ArrayList
                elementsList?.add(map)
            }
        }
    } catch (e: JSONException) {
        e.printStackTrace()
    }
    myHandler.post {
        // обновляем UI форму в фоновом потоке
        runOnUiThread {
            val adapter: ListAdapter = SimpleAdapter(
                this@ElementsActivity, elementsList,
                R.layout.element_item, arrayOf(
                    TAG_ELEMENT_NAME,
                    TAG_HLEVEL,
                    TAG_ELEMENT_ID
                ), intArrayOf(R.id.element_name, R.id.h_level, R.id.element_id)
            )
            // обновляем listview
            val listView = findViewById<View>(R.id.list) as ListView
            listView.adapter = adapter
        }
    }
}
```

```
    }

    private var resultLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
            // There are no request codes
            val data: Intent? = result.data
            finish()
            startActivity(data)
        }
    }

}
```

JSONParser.kt:

```
package com.example.sfriend

import android.util.Log
import org.json.JSONException
import org.json.JSONObject
import java.io.*
import java.net.URL
import kotlin.collections.List

class JSONParser {
    private var `is`: InputStream? = null
    private var jObj: JSONObject? = null
    private var json = ""

    // constructor
```

```
fun JSONParser() {}

// метод получение json объекта по url
// используя HTTP запрос и методы POST или GET
fun makeHttpRequest(url: String?, method: String, params:
MutableList<Pair<String, String?>>): JSONObject? {
    // Створюємо HTTP запит
    var currentUrl = url
    try {
        if(method === "GET") {
            if (params.size>0) {
                currentUrl+="?"
                for (param in params) {
                    currentUrl+=param.first+ "="+param.second+"&"
                }
            }
            val connection = URL(currentUrl).openConnection()
            connection.setRequestProperty("Accept-Charset", "UTF-8")
            `is` = connection.getInputStream()
        }
        // перевіряємо метод HTTP запиту
        if (method === "POST") {
            val connection = URL(currentUrl).openConnection()
            connection.setRequestProperty("Accept-Charset", "UTF-8")
            var postData = ""
            if (params != null) {
                for (param in params) {
                    if (postData !== "") {
                        postData += "&"
```

```
                    }
                    postData += param?.first + "=" + param?.second
                }
            }
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")
            connection.setRequestProperty("Content-Length", postData.length.toString())
            connection.doOutput = true
            DataOutputStream(connection.getOutputStream()).use { it.writeBytes(postData) }
            `is` = connection.getInputStream()
        }
    } catch (e: UnsupportedEncodingException) {
        e.printStackTrace()
    } catch (e: IOException) {
        e.printStackTrace()
    }
    try {
        val reader = BufferedReader(InputStreamReader(`is`, "iso-8859-1"), 8)
        val sb = StringBuilder()
        var line: String?
        while (reader.readLine().also { line = it } != null) {
            sb.append(
                """
                $line

                """.trimIndent()
            )
```

```
            }
            `is`?.close()
            json = sb.toString()
        } catch (e: Exception) {
            Log.e("Buffer Error", "Error converting result $e")
        }

        //
        try {
            jObj = JSONObject(json)
        } catch (e: JSONException) {
            Log.e("JSON Parser", "Error parsing data $e")
        }
        // возвращаем JSON строку
        return jObj
    }
}
```

LoginActivity.kt:

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors
```

```
class LoginActivity : AppCompatActivity() {
    private var jsonParser = JSONParser()
    private val myExecutor = Executors.newSingleThreadExecutor()
    private var inputEmail: EditText? = null
    private var inputPassword: EditText? = null
    private var btnLogIn: Button? = null
    private var urlCheckUser = ""

    private val TAG_SUCCESS = "success"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        urlCheckUser = getString(R.string.OpenServerIp)+"check_user.php"
        setContentView(R.layout.activity_new_user)
        inputEmail = findViewById<View>(R.id.input_email) as EditText
        inputPassword = findViewById<View>(R.id.input_password) as EditText
        btnLogIn = findViewById<View>(R.id.log_in_button) as Button
        btnLogIn!!.setOnClickListener { checkUser() }
    }
    private fun checkUser(){
        myExecutor.execute {
            val email: String = inputEmail?.text.toString()
            val password: String = inputPassword?.text.toString()

            // Заполняем параметры
            val params: MutableList<Pair<String,String?>> = ArrayList()
            params.add(Pair("email", email))
```

```
            params.add(Pair("password", password))

            // получаем JSON объект
            val json: JSONObject? = jsonParser.makeHttpRequest(urlCheckUser,
"GET", params)
            Log.d("Check Response", json.toString())
            try {
               val success = json?.getInt(TAG_SUCCESS)
               if (success == 1) {
                  val userObj = json.getJSONArray("user")

                  val user = userObj.getJSONObject(0)
                  val user_id = user.getString("user_id")
                  (this.application as MyApplication).user_id = user_id
                  val `in` = Intent(applicationContext, UserProfileActivity::class.java)
                  `in`.putExtra("user_id", user_id)
                  startActivity(`in`)

                  // закрываем это окно
                  finish()
               }
               else
               {
                  val err = json?.getString("err_text")
                  val duration = Toast.LENGTH_LONG

                  val toast = Toast.makeText(applicationContext, err, duration)
                  toast.show()
               }
```

```
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    }
}
```

MainActivity.kt:

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val btnLogin = findViewById<Button>(R.id.btn_open_login)
        val btnRegister = findViewById<Button>(R.id.btn_open_reg)

        // обработчик на нажатиЯ кнопки View Users

        // обработчик на нажатиЯ кнопки View Users
        btnLogin.setOnClickListener {
            startActivity(Intent(this, LoginActivity::class.java))
        }
```

```
        btnRegister.setOnClickListener {

            startActivity(Intent(this, RegisterActivity::class.java))
        }
    }
}
```

NewUserActivity.kt

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors


class NewUserActivity : AppCompatActivity() {
    private var jsonParser = JSONParser()
    private val myExecutor = Executors.newSingleThreadExecutor()
    private var inputUserName: EditText? = null
    private var inputFirstName: EditText? = null
    private var inputSurname: EditText? = null
    private var inputLastName: EditText? = null
    private var inputPhoneNum: EditText? = null
    //private var inputEmail: EditText? = null
```

```
    private var urlCreateUser = ""

    private val TAG_SUCCESS = "success"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        urlCreateUser = getString(R.string.OpenServerIp)+"create_user.php"
        setContentView(R.layout.activity_new_user)
        inputUserName = findViewById<View>(R.id.inputUserName) as EditText
        inputFirstName = findViewById<View>(R.id.inputFirstName) as EditText
        inputSurname = findViewById<View>(R.id.inputSurname) as EditText
        inputLastName = findViewById<View>(R.id.inputLastName) as EditText
        inputPhoneNum = findViewById<View>(R.id.inputPhoneNumber) as
EditText
        val btnCreateUser = findViewById<View>(R.id.btnCreateUser) as Button
        btnCreateUser.setOnClickListener { createNewUser() }
    }
    private fun createNewUser(){
        myExecutor.execute {
            val userName: String = inputUserName?.text.toString()
            val firstName: String = inputFirstName?.text.toString()
            val surname: String = inputSurname?.text.toString()
            val lastName: String = inputLastName?.text.toString()
            val phoneNum: String = inputPhoneNum?.text.toString()
            //val email: String = inputEmail?.text.toString()

            // Заполняем параметры
            val params: MutableList<Pair<String,String?>> = ArrayList()
```

```
                params.add(Pair("userName", userName))
                params.add(Pair("firstName", firstName))
                params.add(Pair("surname", surname))
                params.add(Pair("lastName", lastName))
                params.add(Pair("phoneNum", phoneNum))
                val json: JSONObject? = jsonParser.makeHttpRequest(urlCreateUser,
"POST", params)
                Log.d("Create Response", json.toString())
                try {
                    val success = json?.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        val i = Intent(applicationContext, SearchUserActivity::class.java)
                        startActivity(i)
                        finish()
                    }
                } catch (e: JSONException) {
                    e.printStackTrace()
                }
            }
        }
    }
```

RegisterActivity.kt:

```
package com.example.sfriend

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.*
```

```
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors


class RegisterActivity : AppCompatActivity() {
    private var jsonParser = JSONParser()
    private val myExecutor = Executors.newSingleThreadExecutor()
    private var inputEmail: EditText? = null
    private var inputPassword: EditText? = null
    private var btnRegister: Button? = null
    private var urlCheckUser = ""

    private val TAG_SUCCESS = "success"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        urlCheckUser                                                    =
getString(R.string.OpenServerIp)+"check_and_register_user.php"
        setContentView(R.layout.activity_new_user)
        inputEmail = findViewById<View>(R.id.input_email) as EditText
        inputPassword = findViewById<View>(R.id.input_password) as EditText
        btnRegister = findViewById<View>(R.id.register_button) as Button
        btnRegister!!.setOnClickListener { checkUser() }
    }
    private fun checkUser(){
        myExecutor.execute {
            val email: String = inputEmail?.text.toString()
```

```
            val password: String = inputPassword?.text.toString()

            // Заполняем параметры
            val params: MutableList<Pair<String,String?>> = ArrayList()
            params.add(Pair("email", email))
            params.add(Pair("password", password))

            // получаем JSON объект
            val json: JSONObject? = jsonParser.makeHttpRequest(urlCheckUser,
"GET", params)
            Log.d("Check Response", json.toString())
            try {
               val success = json?.getInt(TAG_SUCCESS)
               if (success == 1) {
                  val userObj = json.getJSONArray("user")

                  val user = userObj.getJSONObject(0)
                  val user_id = user.getString("user_id")
                  (this.application as MyApplication).user_id = user_id
                  val `in` = Intent(applicationContext, UserProfileActivity::class.java)
                  `in`.putExtra("user_id", user_id)
                  startActivity(`in`)

                  // закрываем это окно
                  finish()
               }
               else
               {
                  val err = json?.getString("err_text")
```

```
                val duration = Toast.LENGTH_LONG

                val toast = Toast.makeText(applicationContext, err, duration)
                toast.show()
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
    }
  }
}
```

SearchUserActivity.kt:

```
package com.example.sfriend

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors
```

```
class SearchUserActivity : AppCompatActivity() {

    // Создаем JSON парсер
    private var jParser = JSONParser()

    private var usersList: ArrayList<HashMap<String, String>>? = null
    private var tmpUsersList: ArrayList<HashMap<String, String>>? = null
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())
    private var btnSearch: Button? = null
    private var inputSearch: EditText? = null

    // url получения списка всех продуктов
    private var urlAllUsers = ""

    // JSON Node names
    private val TAG_SUCCESS = "success"
    private val TAG_USERS = "users"
    private val TAG_USER_ID = "user_id"
    private val TAG_USER_NAME = "user_name"
    private val TAG_PHONE_NUM = "phone_num"
    private val TAG_EMAIL = "email"
    private var btnTrainings: Button? = null
    private var btnElements: Button? = null
    private var btnAddTraining: Button? = null
    private var btnProfile: Button? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        urlAllUsers = getString(R.string.OpenServerIp)+"get_all_users.php"
        setContentView(R.layout.activity_all_users)
        btnSearch = findViewById<View>(R.id.search_search_button) as Button
        btnTrainings = findViewById<View>(R.id.search_trainings_button) as
Button
        btnElements = findViewById<View>(R.id.search_elements_button) as
Button
        btnProfile = findViewById<View>(R.id.search_profile_button) as Button
        btnAddTraining = findViewById<View>(R.id.search_add_training_button)
as Button
        // Hashmap for ListView
        usersList = ArrayList()
        tmpUsersList = ArrayList()

        // Загружаем продукты в фоновом потоке
        loadAllUsers()

        btnProfile!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            val `in` = Intent(applicationContext, UserProfileActivity::class.java)
            `in`.putExtra(TAG_USER_ID, getString(R.string.current_user_id))
            startActivity(`in`)
        }

        btnTrainings!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            val `in` = Intent(applicationContext, UserTrainingsActivity::class.java)
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_USER_ID, getString(R.string.current_user_id))
```

```
            startActivity(`in`)
        }

        // получаем ListView
        val lv: ListView = findViewById(R.id.list)
        btnSearch!!.setOnClickListener { // запускаем выполнение задачи на обновление продукта
            val searchRequest: String = inputSearch?.text.toString()

            searchUsers(searchRequest)
        }

        // на выбор одного продукта
        // запускается Edit User Screen
        lv.setOnItemClickListener { _, view, _, _ -> // getting values from selected ListItem
            val userId = (view.findViewById(R.id.user_id) as TextView).text
                .toString()

            // Запускаем новый intent который покажет нам Activity
            val `in` = Intent(applicationContext, UserProfileActivity::class.java)
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_USER_ID, userId)

            startActivity(`in`)
        }
    }
```

```
        private fun searchUsers(searchReq: String){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                // получаем JSON строк с URL
                val json: JSONObject? = jParser.makeHttpRequest(urlAllUsers, "GET",
params)
                Log.d("All Users: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val users = json.getJSONArray(TAG_USERS)

                        // перебор всех продуктов
                        for (i in 0 until users.length()) {
                            val c: JSONObject = users.getJSONObject(i)

                            // Сохраняем каждый json елемент в переменную
                            val userId = c.getString(TAG_USER_ID)
                            val userName = c.getString(TAG_USER_NAME)
                            val phoneNum = c.getString(TAG_PHONE_NUM)
                            val email = c.getString(TAG_EMAIL)


                            if(userName.contains(searchReq, ignoreCase = true)) {
                                // Создаем новый HashMap
```

```
                val map = HashMap<String, String>()

                map[TAG_USER_ID] = userId
                map[TAG_USER_NAME] = userName
                map[TAG_PHONE_NUM] = phoneNum
                map[TAG_EMAIL] = email

                // добавляем HashList в ArrayList
                usersList?.add(map)
            }
        }
    } else {
        // продукт не найден
        // Запускаем Add New User Activity
        val i = Intent(
            applicationContext,
            NewUserActivity::class.java
        )
        // Закрытие всех предыдущие activities
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
        startActivity(i)
    }
} catch (e: JSONException) {
    e.printStackTrace()
}
myHandler.post {
    // обновляем UI форму в фоновом потоке
    runOnUiThread {
        val adapter: ListAdapter = SimpleAdapter(
```

```
                    this@SearchUserActivity, usersList,
                    R.layout.list_item, arrayOf(
                        TAG_USER_ID,
                        TAG_USER_NAME,
                        TAG_PHONE_NUM,
                        TAG_EMAIL
                    ),  intArrayOf(R.id.user_id,  R.id.user_firstname,  R.id.phone_num,
R.id.email)
                )
                // обновляем listview
                val listView = findViewById<View>(R.id.list) as ListView
                listView.adapter = adapter
            }
        }
    }
}
private fun loadAllUsers(){
    myExecutor.execute {
        // Будет хранить параметры
        val params: MutableList<Pair<String,String?>> = ArrayList()
        // получаем JSON строк с URL
        val  json:  JSONObject?  =  jParser.makeHttpRequest(urlAllUsers,  "GET",
params)
        Log.d("All Users: ", json.toString())
        try {
            // Получаем SUCCESS тег для проверки статуса ответа сервера
            val success = json!!.getInt(TAG_SUCCESS)
            if (success == 1) {
                // продукт найден
```

```
            // Получаем масив из Продуктов
            val users = json.getJSONArray(TAG_USERS)

            // перебор всех продуктов
            for (i in 0 until users.length()) {
                val c: JSONObject = users.getJSONObject(i)

                // Сохраняем каждый json елемент в переменную
                val userId = c.getString(TAG_USER_ID)
                val userName = c.getString(TAG_USER_NAME)
                val phoneNum = c.getString(TAG_PHONE_NUM)
                val email = c.getString(TAG_EMAIL)

                // Создаем новый HashMap
                val map = HashMap<String, String>()

                // добавляем каждый елемент в HashMap ключ => значение
                map[TAG_USER_ID] = userId
                map[TAG_USER_NAME] = userName
                map[TAG_PHONE_NUM] = phoneNum
                map[TAG_EMAIL] = email

                // добавляем HashList в ArrayList
                usersList?.add(map)
            }
        } else {
            // продукт не найден
            // Запускаем Add New User Activity
            val i = Intent(
```

```
                    applicationContext,
                    NewUserActivity::class.java
                )
                // Закрытие всех предыдущие activities
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
                startActivity(i)
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
        myHandler.post {
            // обновляем UI форму в фоновом потоке
            runOnUiThread {
                val adapter: ListAdapter = SimpleAdapter(
                    this@SearchUserActivity, usersList,
                    R.layout.list_item, arrayOf(
                        TAG_USER_ID,
                        TAG_USER_NAME,
                        TAG_PHONE_NUM,
                        TAG_EMAIL
                    ),  intArrayOf(R.id.user_id,  R.id.user_firstname,  R.id.phone_num,
R.id.email)
                )
                // обновляем listview
                val listView = findViewById<View>(R.id.list) as ListView
                listView.adapter = adapter
            }
        }
    }
```

```
    }

    private var resultLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
            // There are no request codes
            val data: Intent? = result.data
            finish()
            startActivity(data)
        }
    }

}
```

UserProfileActivity.kt:

```
package com.example.sfriend

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors
```

```
class SearchUserActivity : AppCompatActivity() {

    // Создаем JSON парсер
    private var jParser = JSONParser()

    private var usersList: ArrayList<HashMap<String, String>>? = null
    private var tmpUsersList: ArrayList<HashMap<String, String>>? = null
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())
    private var btnSearch: Button? = null
    private var inputSearch: EditText? = null

    // url получения списка всех продуктов
    private var urlAllUsers = ""

    // JSON Node names
    private val TAG_SUCCESS = "success"
    private val TAG_USERS = "users"
    private val TAG_USER_ID = "user_id"
    private val TAG_USER_NAME = "user_name"
    private val TAG_PHONE_NUM = "phone_num"
    private val TAG_EMAIL = "email"
    private var btnTrainings: Button? = null
    private var btnElements: Button? = null
    private var btnAddTraining: Button? = null
    private var btnProfile: Button? = null
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    urlAllUsers = getString(R.string.OpenServerIp)+"get_all_users.php"
    setContentView(R.layout.activity_all_users)
    btnSearch = findViewById<View>(R.id.search_search_button) as Button
    btnTrainings = findViewById<View>(R.id.search_trainings_button) as
Button
    btnElements = findViewById<View>(R.id.search_elements_button) as
Button
    btnProfile = findViewById<View>(R.id.search_profile_button) as Button
    btnAddTraining = findViewById<View>(R.id.search_add_training_button)
as Button
    // Hashmap for ListView
    usersList = ArrayList()
    tmpUsersList = ArrayList()

    // Загружаем продукты в фоновом потоке
    loadAllUsers()

    btnProfile!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
        val `in` = Intent(applicationContext, UserProfileActivity::class.java)
        `in`.putExtra(TAG_USER_ID, getString(R.string.current_user_id))
        startActivity(`in`)
    }

    btnTrainings!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
        val `in` = Intent(applicationContext, UserTrainingsActivity::class.java)
```

```
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_USER_ID, getString(R.string.current_user_id))

            startActivity(`in`)
        }

        // получаем ListView
        val lv: ListView = findViewById(R.id.list)
        btnSearch!!.setOnClickListener { // запускаем выполнение задачи на обновление продукта
            val searchRequest: String = inputSearch?.text.toString()

            searchUsers(searchRequest)
        }

        // на выбор одного продукта
        // запускается Edit User Screen
        lv.setOnItemClickListener { _, view, _, _ -> // getting values from selected ListItem
            val userId = (view.findViewById(R.id.user_id) as TextView).text
                .toString()

            // Запускаем новый intent который покажет нам Activity
            val `in` = Intent(applicationContext, UserProfileActivity::class.java)
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_USER_ID, userId)

            startActivity(`in`)
        }
```

```
        }

        private fun searchUsers(searchReq: String){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                // получаем JSON строк с URL
                val json: JSONObject? = jParser.makeHttpRequest(urlAllUsers, "GET",
params)
                Log.d("All Users: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val users = json.getJSONArray(TAG_USERS)

                        // перебор всех продуктов
                        for (i in 0 until users.length()) {
                            val c: JSONObject = users.getJSONObject(i)

                            // Сохраняем каждый json елемент в переменную
                            val userId = c.getString(TAG_USER_ID)
                            val userName = c.getString(TAG_USER_NAME)
                            val phoneNum = c.getString(TAG_PHONE_NUM)
                            val email = c.getString(TAG_EMAIL)
```

```
            if(userName.contains(searchReq, ignoreCase = true)) {
                // Создаем новый HashMap
                val map = HashMap<String, String>()

                map[TAG_USER_ID] = userId
                map[TAG_USER_NAME] = userName
                map[TAG_PHONE_NUM] = phoneNum
                map[TAG_EMAIL] = email

                // добавляем HashList в ArrayList
                usersList?.add(map)
            }
        }
    } else {
        // продукт не найден
        // Запускаем Add New User Activity
        val i = Intent(
            applicationContext,
            NewUserActivity::class.java
        )
        // Закрытие всех предыдущие activities
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
        startActivity(i)
    }
} catch (e: JSONException) {
    e.printStackTrace()
}
myHandler.post {
    // обновляем UI форму в фоновом потоке
```

```
                runOnUiThread {
                    val adapter: ListAdapter = SimpleAdapter(
                        this@SearchUserActivity, usersList,
                        R.layout.list_item, arrayOf(
                            TAG_USER_ID,
                            TAG_USER_NAME,
                            TAG_PHONE_NUM,
                            TAG_EMAIL
                        ),  intArrayOf(R.id.user_id,  R.id.user_firstname,  R.id.phone_num,
R.id.email)
                    )
                    // обновляем listview
                    val listView = findViewById<View>(R.id.list) as ListView
                    listView.adapter = adapter
                }
            }
        }
        private fun loadAllUsers(){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                // получаем JSON строк с URL
                val  json:  JSONObject?  =  jParser.makeHttpRequest(urlAllUsers,  "GET",
params)
                Log.d("All Users: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
```

```
if (success == 1) {
    // продукт найден
    // Получаем масив из Продуктов
    val users = json.getJSONArray(TAG_USERS)

    // перебор всех продуктов
    for (i in 0 until users.length()) {
        val c: JSONObject = users.getJSONObject(i)

        // Сохраняем каждый json елемент в переменную
        val userId = c.getString(TAG_USER_ID)
        val userName = c.getString(TAG_USER_NAME)
        val phoneNum = c.getString(TAG_PHONE_NUM)
        val email = c.getString(TAG_EMAIL)

        // Создаем новый HashMap
        val map = HashMap<String, String>()

        // добавляем каждый елемент в HashMap ключ => значение
        map[TAG_USER_ID] = userId
        map[TAG_USER_NAME] = userName
        map[TAG_PHONE_NUM] = phoneNum
        map[TAG_EMAIL] = email

        // добавляем HashList в ArrayList
        usersList?.add(map)
    }
} else {
    // продукт не найден
```

```
                // Запускаем Add New User Activity
                val i = Intent(
                    applicationContext,
                    NewUserActivity::class.java
                )
                // Закрытие всех предыдущие activities
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
                startActivity(i)
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
        myHandler.post {
            // обновляем UI форму в фоновом потоке
            runOnUiThread {
                val adapter: ListAdapter = SimpleAdapter(
                    this@SearchUserActivity, usersList,
                    R.layout.list_item, arrayOf(
                        TAG_USER_ID,
                        TAG_USER_NAME,
                        TAG_PHONE_NUM,
                        TAG_EMAIL
                    ),  intArrayOf(R.id.user_id,  R.id.user_firstname,  R.id.phone_num,
R.id.email)
                )
                // обновляем listview
                val listView = findViewById<View>(R.id.list) as ListView
                listView.adapter = adapter
            }
```

```
            }
        }
    }

    private var resultLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
            // There are no request codes
            val data: Intent? = result.data
            finish()
            startActivity(data)
        }
    }

}
```

UserTrainingsActivity.kt:

```
package com.example.sfriend

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
```

```
import org.json.JSONException
import org.json.JSONObject
import java.util.concurrent.Executors


class UserTrainingsActivity : AppCompatActivity() {

    // Создаем JSON парсер
    private var jParser = JSONParser()

    private var trainingsList: ArrayList<HashMap<String, String>>? = null
    private val myExecutor = Executors.newSingleThreadExecutor()
    private val myHandler = Handler(Looper.getMainLooper())

    // url получения списка всех продуктов
    private var urlAllTrainings = ""

    // JSON Node names
    private val TAG_SUCCESS = "success"
    private val TAG_USERS = "training"
    private val TAG_TRAINING_ID = "training_id"
    private val TAG_USER_ID = "user_id"
    private val TAG_ACTIVITY_DATE = "date_time"
    private val TAG_DURATION = "duration"
    private var training_id = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        urlAllTrainings =
getString(R.string.OpenServerIp)+"get_user_activities.php"
        setContentView(R.layout.activity_all_trainings)

        // Hashmap for ListView
        trainingsList = ArrayList()

        val i = intent
        // получаем id продукта (pid) с формы
        training_id = i.getStringExtra(TAG_USER_ID)!!

        // Загружаем продукты в фоновом потоке
        loadAllTrainings()

        // получаем ListView
        val lv: ListView = findViewById(R.id.list)
        val profileButton: TextView = findViewById(R.id.trainings_profile_button)
        val addButton: TextView =
findViewById(R.id.trainings_add_training_button)
        val btnSearch: Button = findViewById(R.id.trainings_searching_button)
        val elementsButton: TextView =
findViewById(R.id.trainings_elements_button)
        btnSearch!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            val `in` = Intent(applicationContext, SearchUserActivity::class.java)

            startActivity(`in`)
        }
```

```
        profileButton!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            val `in` = Intent(applicationContext, UserProfileActivity::class.java)
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_USER_ID, getString(R.string.current_user_id))

            startActivity(`in`)
        }
        elementsButton!!.setOnClickListener { // запускаем выполнение задачи на
обновление продукта
            val `in` = Intent(applicationContext, ElementsActivity::class.java)

            startActivity(`in`)
        }
        // на выбор одного продукта
        // запускается Edit User Screen
        lv.setOnItemClickListener { _, view, _, _ -> // getting values from selected
ListItem
            val training_id = (view.findViewById(R.id.training_id) as TextView).text
                .toString()

            // Запускаем новый intent который покажет нам Activity
            val `in` = Intent(applicationContext,
CompletedTrainingActivity::class.java)
            // отправляем user_id в следующий activity
            `in`.putExtra(TAG_TRAINING_ID, training_id)

            startActivity(`in`)
        }
```

```
        }

        private fun loadAllTrainings(){
            myExecutor.execute {
                // Будет хранить параметры
                val params: MutableList<Pair<String,String?>> = ArrayList()
                val i = intent
                params.add(Pair(TAG_USER_ID, i.getStringExtra(TAG_USER_ID)))
                // получаем JSON строк с URL
                val json: JSONObject? = jParser.makeHttpRequest(urlAllTrainings,
"GET", params)
                Log.d("All Users: ", json.toString())
                try {
                    // Получаем SUCCESS тег для проверки статуса ответа сервера
                    val success = json!!.getInt(TAG_SUCCESS)
                    if (success == 1) {
                        // продукт найден
                        // Получаем масив из Продуктов
                        val users = json.getJSONArray(TAG_USERS)

                        // перебор всех продуктов
                        for (i in 0 until users.length()) {
                            val c: JSONObject = users.getJSONObject(i)

                            // Сохраняем каждый json елемент в переменную
                            val duration = c.getString(TAG_DURATION)
                            val date_time = c.getString(TAG_ACTIVITY_DATE)
                            val training_id = c.getString(TAG_TRAINING_ID)
```

```
            // Создаем новый HashMap
            val map = HashMap<String, String>()

            // добавляем каждый елемент в HashMap ключ => значение
            map[TAG_DURATION] = duration
            map[TAG_ACTIVITY_DATE] = date_time
            map[TAG_TRAINING_ID] = training_id

            // добавляем HashList в ArrayList
            trainingsList?.add(map)
        }
    }
} catch (e: JSONException) {
    e.printStackTrace()
}
myHandler.post {
    // обновляем UI форму в фоновом потоке
    runOnUiThread {
        val adapter: ListAdapter = SimpleAdapter(
            this@UserTrainingsActivity, trainingsList,
            R.layout.training_item, arrayOf(
                TAG_ACTIVITY_DATE,
                TAG_DURATION,
                TAG_TRAINING_ID
            ), intArrayOf(R.id.date_time, R.id.duration, R.id.training_id)
        )
        // обновляем listview
        val listView = findViewById<View>(R.id.list) as ListView
        listView.adapter = adapter
```

```
                }
            }
        }
    }

    private var resultLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
            // There are no request codes
            val data: Intent? = result.data
            finish()
            startActivity(data)
        }
    }

}
```

Серверна частина:

db_config.php:

```
<?php
    define('DB_USER', "admin"); //логин админа БД
    define('DB_PASSWORD', "admin"); // пароль админа БД
    define('DB_DATABASE', "sfriend"); // база данных
    define('DB_SERVER', "openserv"); // сервер 'localhost'
?>
```

delete_user.php:

```
<?php

$response = array();
```

```
if (isset($_POST['user_id'])) {
    $user_id = $_POST['user_id'];

    require 'db_connect.php';

    $db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or die(mysql_error());

    $result = $db -> query("DELETE FROM users WHERE user_id = $user_id");
    if ($result -> affected_rows() > 0) {
        $response["success"] = 1;
        $response["message"] = "User successfully deleted";

        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No user found";

        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    echo json_encode($response);
}
?>
```

get_all_categiries.php:

```
<?php
```

```
$response = array();

require 'db_config.php';

$db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or die(mysql_error());

$result = $db -> query("SELECT * FROM categories") or die(mysql_error());

if ($result -> num_rows > 0) {
   $response["categories"] = array();

   while ($row = $result -> fetch_array(MYSQLI_ASSOC)) {
      $category = array();
      $category["category_id"] = $row["category_id"];
      $category["name"] = $row["name"];

      array_push($response["categories"], $category);
   }
   $response["success"] = 1;

   echo json_encode($response);
} else {
   $response["success"] = 0;
   $response["message"] = "No categories found";

   echo json_encode($response);
}
```

```
?>
get_all_elements.php:
<?php

$response = array();

require 'db_config.php';

$db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or die(mysql_error());

$result = $db -> query("SELECT
elements.el_name AS element_name,
elements.h_Level AS h_level,
elements.element_id AS element_id,
categories.name AS category
FROM elements
INNER JOIN categories
ON elements.category_id = categories.category_id") or die(mysql_error());

if ($result -> num_rows > 0) {
   $response["elements"] = array();

   while ($row = $result -> fetch_array(MYSQLI_ASSOC)) {
      $element = array();
      $element["element_id"] = $row["element_id"];
      $element["element_name"] = $row["element_name"];
      $element["h_level"] = $row["h_level"];
      $element["category"] = $row["category"];
```

```
            array_push($response["elements"], $element);
        }
        $response["success"] = 1;

        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No elements found";

        echo json_encode($response);
    }
    ?>
```

get_all_users.php:

```
    <?php

    $response = array();

    require 'db_config.php';

    $db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or die(mysql_error());

    $result = $db -> query("SELECT
    users.user_id AS user_id,
    users.user_name AS user_name,
    users.first_name AS first_name,
    users.surname AS surname,
    users.phone_num AS phone_num,
```

```
user_credentials.email AS email
FROM users
INNER JOIN user_credentials
ON users.user_id = user_credentials.user_id") or die(mysql_error());

if ($result -> num_rows > 0) {
    $response["users"] = array();

    while ($row = $result -> fetch_array(MYSQLI_ASSOC)) {
        $user = array();
        $user["user_id"] = $row["user_id"];
        $user["user_name"] = $row["user_name"];
        $user["first_name"] = $row["first_name"];
        $user["surname"] = $row["surname"];
        $user["phone_num"] = $row["phone_num"];
        $user["email"] = $row["email"];
        #$user["created_at"] = $result["created_at"];

        array_push($response["users"], $user);
    }
    $response["success"] = 1;

    echo json_encode($response);
} else {
    $response["success"] = 0;
    $response["message"] = "No users found";

    echo json_encode($response);
}
```

```
?>
get_element_info.php:
<?php

$response = array();
$element = array();

require 'db_config.php';

$db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);
/* Проверить соединение */
if ($db->connect_errno) {
   printf("Соединение не удалось: %s\n", $db->connect_error);
   exit();
}

if (isset($_GET["element_id"])) {
   $element_id = $_GET['element_id'];

   $result = $db -> query("SELECT
      elements.el_name AS element_title,
   elements.h_level AS element_h_level,
   elements.descr AS description,
   elements.element_id AS element_id,
      categories.name AS element_category
FROM elements
INNER JOIN categories
ON elements.category_id = categories.category_id
```

```
WHERE elements.element_id = $element_id");

    if ($result) {
        if ($result -> num_rows > 0) {

            $row = $result -> fetch_array(MYSQLI_ASSOC);



            $element["element_id"] = $row["element_id"];
            $element["element_title"] = $row["element_title"];
            $element["element_category"] = $row["element_category"];
            $element["element_h_level"] = $row["element_h_level"];
            $element["description"] = $row["description"];
            $element["isAdded"] = $row["isAdded"];
            $response["success"] = 1;

            $response["element"] = array();

            array_push($response["element"], $element);

            echo json_encode($response);
        } else {
            $response["success"] = 0;
            $response["message"] = "No element found";

            echo json_encode($response);
        }
    } else {
        $response["success"] = 0;
```

```
        $response["message"] = "No element found";

        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    echo json_encode($response);
}
?>
```