МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.
(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: Android-додаток «Фітнес-щоденник»_____

Виконавець: _____ Гребеник Артем Романович
(підпис) **(ПІБ)**

Керівник:_____ Гамаюн Володимир Петрович
(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович
(підпис) **(ПІБ)**

Київ 2021

ВСТУП

Дуже швидке зростання ринку персональних цифрових пристроїв, що можуть виконувати різноманітні функції допомоги людині-власнику, спричинює їх тотальне впровадження у всі сфери життя сучасного індивідуума. Так, програми допомагають у вивченні іноземних мов та математики, орієнтуванні на місцевості та пошуку потрібних місцезнаходжень, навіть у замовленні продуктів та таксі в реальному світі. Не виключенням є і сфера фізичної культури та спорту: тут навіть для пересічного громадянина (не професійного спортсмена) можуть знадобитися певні стандартні речі-функції такого асистуючого програмного забезпечення. Відповідно, розробка таких програмних продуктів, як фітнес-щоденник, є актуальною для фахівця в галузі ІТ, тому їй і присвячена дана робота.

Метою роботи є підвищення зручності процесу інформаційного супроводу тренувань з фітнесу. При умові використання проектованого додатку, людині більше не потрібно буде запам’ятовувати плани тренувань та інші різноманітні їх аспекти (або писати все це на папірцях), а можна зберігати усю інформацію, пов’язану із тренуваннями, в одному місці. Тим більше у такому зручному місці, як сучасний смартфон, що завжди «під рукою».

Частинними задачами роботи є:

- аналіз предметної галузі та виділення питань, що підлягають вирішенню саме у даній роботі;

- створення проекту мобільного додатку для інформаційного супроводу занять із фітнесу;

- виконання програмної реалізації спроектованого мобільного додатку.

Об’єктом дослідження є процес інформаційного супроводу занять із фітнесу.

Предметом дослідження є методи та засоби для ведення такого процесу.

Методи досліджень: методи та технології програмування, математичне моделювання, метод аналізу (декомпозиція).

Практичне значення роботи полягає у створення робочого програмного продукту – мобільного додатку для найбільш популярної (по кількості пристроїв, на яких вона встановлена) мобільної операційної системи Android, що може зручно використовуватися людьми, які проводять тренування із фітнесу.

У перспективі можливе розширення розробленого програмного продукту шляхом додавання нової функціональності для тренувань саме із фітнесу, а також залучення інших (споріднених) видів спорту – бодібілдінгу, легкої атлетики і т.п.

1 АНАЛІЗ МОЖЛИВОСТЕЙ ЗАСТОСУВАННЯ СУЧАСНИХ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ГАЛУЗІ ОЗДОРОВЧО-ПРОФІЛАКТИЧНОЇ ФІЗКУЛЬТУРИ ТА СПОРТУ

1.1 Можливі способи використання сучасних гаджетів для цілей фізичної культури

На разі у сучасному суспільстві постала проблема недостатньої фізичної активності людини протягом доби. Така ситуація є однією з вагомих причин більш, чи менш серйозних проблем із здоров'ям. Основним чинником даної проблеми є зловживання сучасними можливостями інформаційно-комунікаційних систем, Інтернету та соціальних мереж, але насправді всі ці засоби можна використати і на користь для здоров'я.

Так, зараз Інтернет простір є місцем де люди не тільки отримують необхідну інформацію, а ще і змагаються один з одним у різних формах будь-то кількість лайків в Instagram чи битви в онлайн іграх. Саме це у нас забирає більшу частину часу та несе за собою втрату фізичної активності і, як наслідок, здоров'я. Доцільним є об'єднання між собою змагань в Інтернет просторі та можливостей по забезпеченню здорового способу життя шляхом створення відповідної системи, у якій для віртуального розвитку буде необхідний і фізичний.

Така система може бути виконана у вигляді додатку на телефоні, який пов'язаний із окремим апаратним пристроєм – фітнес браслетом на руці, що слідкує за кількістю кроків та спалених калорій за день. На основі даних, зібраних з браслету кожного користувача, які повинні надсилатися на відповідний централізований сервер, будується статистика, результатом якої є таблиця переможців, що буде періодично оновлюватися (наприклад, кожного тижня) і на основі якої здійснюватиметься винагороджування переможців. Саме така система може реально стимулювати людей бути більш фізично активними.

У якості нагороди можна впроваджувати як нематеріальні заохочення (наприклад, надавати переможцям можливість прорекламувати на всю аудиторію додатку свою сторінку у соціальній мережі чи власний канал), так і пов’язана з фінансами (наприклад, купони на знижку у магазинах спортивних товарів, або в магазинах здорового харчування, що не лише дасть людині можливість заощадити кошти, але і буде корисно для розвитку малого та середнього бізнесу).

Для підвищення мотивації учасників (говорячи просто, для забезпечення «чесності» змагань), треба поділити гравців на вікові та фізичні категорії щоб кожен мав змогу змагатися з людьми свого рівня. Також для мотивування користувачів можна зробити список щоденних завдань, за виконання яких гравцеві будуть надаватися додаткові бали, наприклад піти додому більш довшим шляхом, провести годину у парку і т. п.. Також у цьому додатку доцільно реалізувати перелік порад щодо здорового способу життя, харчування, цікаві факти про спорт, наводити у приклад людей, які змогли вдосконалити себе фізично. В майбутньому така система могла би перерости у глобальну соціальну мережу фізкультури та спорту, де учасники могли би ділитися своїми досягненнями, порадами, шукати однодумців та реалізовувати будь-яку діяльність, пов’язану із здоровим способом життя.

Ще однією перевагою вказаної ідеї є можливість будувати статистику фізичної активності людей по містам, регіонам, країнам, або навіть глобально - по всьому світу.

Очевидно, що завдяки описаній системі можна покращувати фізичний стан людей по всьому світу (причому з використанням тих речей, які сьогодні навпаки шкодять здоровому способу життя – смартфонів та Інтернет).

Також у сучасний час більшість людей схильні до проблеми малорухливого способу життя, причиною чого в тому числі є надмірне захоплення комп'ютерними технологіями (іграми, соціальними мережами та засобами комунікації, переглядом Інтернет-сайтів). Сучасне покоління все

більше і більше стало переходити в віртуальний світ і забувати про реальні, фізіологічні потреби, однією з яких, очевидно, є достатня кількість фізичної активності протягом дня.

У той же самий час сам засіб, який є джерелом цієї проблеми - сучасний «розумний» мобільний телефон (смартфон) - може допомогти в її вирішенні. Зокрема доцільно використання програмного засобу (реалізованого у вигляді мобільного додатку для найбільш популярних операційних систем Android і iOS), який контролював б рівень фізичної активності людини протягом дня, і, при необхідності, настійно рекомендував би зробити додаткові, контрольовані ним вправи.

В першу чергу такий додаток повинен враховувати вже існуючі особливості організму користувача, що можна реалізувати у вигляді короткого опитування (зріст, вага, наявність хронічних захворювань - на вибір зі списку, і т. д.) при установці в систему. На підставі цієї інформації програма повинна розраховувати обсяги і тип фізичних вправ, рекомендованих для даної людини для виконання протягом дня.

По-друге, програма повинна вести контроль фізичної активності користувача протягом дня, що можливо реалізувати, враховуючи малі розміри смартфона і можливість (яка зазвичай і реалізується) його безперервного носіння в кишені (бажано, у певній кишені - наприклад, нагрудній). Технічно при цьому сигнал може бути отриманий з акселерометра (G-датчик, гіроскоп) смартфона. Очевидно, що в програмі повинні бути при цьому прошиті типові «схеми» зміни прискорень, що відповідають основним рухам людини.

Нарешті, третьою важливою особливістю запропонованого додатка повинна бути можливість обмеження роботи інших додатків, що перешкоджають фізичної активності користувача (мобільні версії клієнтів соцмереж, всі встановлені гри, і т.п. - список таких програм може задаватися при установці або довантажуватися з сервера, що підтримує роботу додатка - при можливості наявності такого). Наприклад, в їх роботі можна

влаштовувати примусові паузи і відстрочки, а у випадку з іграми - взагалі забороняти їх запуск до проведення рекомендованих обсягів вправ.

З використанням такого додатка людина буде менше відволікатися, відкладати заняття на потім і мотивувати себе на виконання корисного для неї фізичного навантаження, що, очевидно, має сприяти профілактиці захворювань, які є наслідками малорухливого способу життя.

Відмітимо, що попередні запропоновані рішення мають складну структуру та являють собою комплекси як мінімум із клієнтської та серверної частин (причому серверну частину слід десь розміщувати і супроводжувати, на що потрібні гроші), а в останньому випадку взагалі – програмно-апаратну систему. Існують і більш прості способи застосування мобільних програмних продуктів для цілей фізичної культури та спорту. Так, очевидно, що для забезпечення достатньої ефективності заняття мають відбуватися у відповідності до певного плану, схеми тренування, які мають десь зберігатися. Традиційним способом тут є варіант фіксації плану тренувань та/або детальних описів ходу їх проведення на паперових носіях (зошитах, блокнотах, щоденниках). При цьому спортсмен стикається з рядом усіх недоліків паперового документообігу:

- паперовий носій має завжди бути присутнім у спортсмена (що не завжди реалізується, оскільки людина іноді змінює плани і прямує на тренування із інших місць, а не із дому, а взагалі завжди носити з собою такий щоденник очевидно недоцільно);

- не дуже мала маса та габарити паперового носія вносять незручність у його постійному транспортуванні на місце тренувань та назад додому;

- паперовий носій має обмежений обсяг і, залежно від ступеня деталізації записів у ньому, рано чи пізно закінчується, що спричинює втрату ранніх записів або необхідність збереження багатьох томів такого щоденника;

- при необхідності знаходження якихось записів (за певне число, або із особливими записами про певний стан самопочуття) пошук інформації на

паперовому носії може здійснюватися десятки хвилин і більше (залежно від обсягів та кількості томів паперової інформації, що вже накопичилися);

- відсутність можливості розширення функціональності паперового блокнота, адже комп'ютерну програму можна легко навчити виконувати певні нові функції (наприклад, таймер для фіксації тривалості занять), а можливості паперових носіїв із очевидних причин строго обмежені, і т.д.

Ведення обліку занять в електронній цифровій формі на смартфоні позбавляє від усіх перерахованих недоліків (оскільки смартфон і так завжди поруч з користувачем, відносно компактний, має практично необмежені обсяги пам'яті, та має значну кількість можливостей: подання аудіо та відеосигналів, зчитування інформації із численних сенсорів, що може бути корисним для спортсмена, і т.п.) та відкриває нові цікаві можливості для особи, що його використовує. Таким чином, доцільним є використання програм для мобільних пристроїв – асистентів, що допомагають фіксувати хід ведення тренувань та, можливо, мають певні додаткові, корисні для занять можливості.

1.2 Аналіз вимог до програмних продуктів, що можуть бути корисними для спортивних занять

Слід сказати, що на сьогоднішній день існує чимала кількість рішень по базовій архітектурі програмного продукту, що може представляти собою фітнес-щоденник. Такий програмний продукт может бути реалізований у вигляді:

а) настільного Desktop-додатку для ОС Windows (оскільки саме ця операційна система є найбільш поширеною серед персональних комп'ютерів). Даний варіант є найменш зручним, оскільки вимагає доступу до ПК для того, щоб внести якусь нову інформацію у щоденник, або продивитися існуючу. Очевидно, що займаючись на природі, або у фітнес-центрі (спортзалі) такого доступу людина не має. Відповідно, щоби

користуватися настільним додатком слід запам’ятовувати усю необхідну інформацію та вносити її в систему тільки коли користувач добиратиметься до комп’ютера. Кінцево однозначно можна сказати, що такий варіант організації фітнес-щоденику є найменш зручним і розглядатися у даній роботі не буде;

б) веб-додатку, який має більшу гнучкість, оскільки є доступним із будь-якого пристрою, підключеного до мережі Інтернет. Цей варіант може використовуватися як зі стаціонарного ПК та ноутбуку, так і зі смартфону, що має доступ в Інтернет. Загальновідомо, що існують ситуації, коли доступ до Інтернету з мобільного пристрою може незабаром припинятися (проблеми з бездротовою GSM мережею, слабкий сигнал у лісі, де часто проводяться спортивні заняття, у підвалах, де часто розміщують спортивні зали і раптові закінчення грошей на рахунку). Зважаючи на усі ці можливі проблеми, архітектура програмного забезпечення типу «веб-додаток» для організації фітнес-щоденника також не дуже підходить (хоча і набагато краще, аніж настільний додаток);

в) вбудованого програмно-апаратного рішення, такого, як фітнес-браслет або фітнес-трекер. Очевидно, що такий варіант із використанням спеціально призначеного пристрою є найбільш зручним, але і потребує фінансових затрат, що не є прийнятним для широких верств населення (особливо, зважаючи на те, що приблизно ідентичну функціональність можна отримати абсолютно безкоштовно за допомогою рішення із наступного пункту). Зважаючи на порівняно високу вартість, такий підхід до зведення фітнес-щоденника також розглядати не будемо;

г) мобільного додатку, що є найбільш зручним рішенням для створення фітнес-щоденника. По-перше, це рішення може бути виконано абсолютно безкоштовним. По-друге, сам пристрій, на якому буде завантажуватися програмне забезпечення є компактним, зручним у використанні і уже є присутнім майже у всіх осіб, які за віком та звичками можуть (хоча б потенційно займатися фітнесом). По-третє, по самій своїй суті смартфон

завжди присутній разом з людиною, отже ідеально підходить до створення моментальних нотаток та їх читання/переглядання.

Таким чином, програмний продукт фітнес-щоденнику має бути мобільним додатком, причому готовим для використання на найбільш поширених операційних системах, аналізу чого ще буде присвячено окреме дослідження нижче.

Вказаний програмний продукт повинен мати якомога більш простий інтерфейс, що обумовлене направленістю додатку на широке коло користувачів, більшість з яких не є професіоналами у сфері IT, або принаймні досвідченими користувачами. Отже, простота інтерфейсу додатку є ще однією вимогою до відповідного програмного продукту.

Також важливою вимогою є орієнтація на найбільш поширені та відомі засоби розробки. Це необхідно для того, щоби у подальшому, за такої необхідності, підтримувати проект якомога довше, а крім того – різними розробниками (тобто вимогою є поширеність обраних засобів розробки).

1.3 Аналіз існуючих програмно-технічних рішень для галузі спорту та фізкультури

Встановивши доцільність використання мобільних додатків для допомоги у проведенні занять з фізичної культури і, зокрема, фітнесу, проаналізуємо програмні продукти, що вже наявні на ринку чи у відкритому доступі і можуть бути використані для таких цілей.

При виконанні відповідного пошуку у Google Play по словам «Fitness Diary» випадає певна кількість програм-аналогів, які розглянемо докладніше.

Так, програма Fitness Diary мета використання якої в точності відповідає цілям даної роботи, втім має очевидний недолік: автори програми хотіли максимально перекрити усі можливі потреби користувачів і включили до її складу настільки багато функціоналу, що вона у значній мірі стала

схожою на середовище для розробки, а не на програму для пересічного користувача – рис. 1.1.

Ще однією програмою, націленою на фіксацію ходу та особливостей тренувань є FitNotes Supporter – рис. 1.2.

Очевидно, що інтерфейс програми значно простіший, ніж у попереднього варіанту, але вона є платною і продається по ціні 109,99 грн. Зважаючи на те, що у програмі відсутні оригінальні (секретні) алгоритми обробки інформації чи інші нестандартні (інноваційні) рішення, взагалі кажучи, немає потреби платити за програмне забезпечення, якщо є можливість використати альтернативний безкоштовний варіант.

Ще один аналогічний мобільний додаток Simple Workout Log поєднує недоліки двох попередніх, лише трохи послаблюючи їх: з одного боку його інтерфейс дещо простіший за Fitness Diary – рис. 1.3, однак він є платним (хоча і дешевше за FitNotes Supporter) і коштує 69,99 грн.

Ще один аналогічний додаток «GymRun Журнал Тренировок и Фитнес-трекер» також надає доступ до всієї функціональності тільки на платній основі, а інтерфейс його простим також назвати не можна – рис. 1.4.

Окрім розглянутих, у системі Google Play присутня ще чимала кількість додатків із схожим призначенням:

- Progression Workout Tracker (інтерфейс складнуватий, а також певні елементи функціональності слід покупати);

- Progression Body & Weight Log – схожа по призначенню програма, але яка більше направлена не на контроль процесу тренувань, а на очевидні

результати від нього типу динаміки зміни ваги тіла спортсмена; також певну функціональність треба купувати;

- My Work in Progress – аналогічна програма, що має більшу направленість на бодібілдеров та пауерліфтеров, так як дозволяє відстежувати динаміку ваг, якими працює спортсмен; окремі можливості програми слід купувати;

- FitHero - Gym Workout Tracker & Exercises Log – інтерфейс програми є складнуватим для сприйняття, окремі корисні функції можна використовувати тільки після покупки;

- тощо.

Таким чином, аналіз існуючих рішень показує два основних їх недоліки:

- у значної кількості додатків дуже складний інтерфейс користувача, що обумовлене, ймовірно, бажанням розробників включити до них якомога більше функціональності та налаштувань, однак однозначно є негативним фактором для широких мас користувачів, більшість з яких є поверхневими користувачами системи Android;

- майже всі додатки є або повністю платними, або важливу частину свого функціоналу надають за плату.

Зважаючи на вищенаведене, доцільною є розробка власного програмного продукту – мобільного додатку для системи Android, який являє собою фітнес-щоденник, що і буде виконано в рамках даної роботи.

1.4 Постановка задачі роботи

Таким чином, беручи до уваги усю інформацію, що наведена вище, можна сформулювати технічне завдання на розробку програмного продукту, що реалізує фітнес-щоденик.

Тип продукту: записна книжка органайзер для фітнес-діяльності.

Назва продукту: URHealthy (читається як «You Are Helathy», тобто «Ви – здорові»).

Призначення продукту: підвищення ефективності роботи користувачів по їхній фітнес-діяльності.

Цільова програмно-апаратна платформа: мобільні пристрої типу смартфонів та планшетів.

Цільова операційна система: ОС Android, версії не нижче 4.1 Jelly Bean.

Сховище даних: найпростіше з точки зору системних вимог.

Мінімальні системні вимоги пристрою, на якому буде завантажуватися мобільний додаток:

- процесор 1 ГГц;
- оперативна пам’ять 1 Гб;
- 50 Мб на диску;
- сенсорний екран 426x320 пікселів.

Функціональність:

- створення записів-повідомлень, пов’язаних із фітнес-діяльністю;
- перегляд записів-повідомлень.

Програма повинна мати зручний сучасний інтерфейс, стандартний для пристроїв на базі ОС Android.

Розробка програмного продукту ведеться в рамках написання кваліфікаційної роботи і буде захищатися під час представлення результатів її створення.

Керуючись даними вимогами можна переходити до процесу проектування програмного продукту, вибору засобів розробки та проведення його програмної реалізації.

1.5 Висновки по розділу

Таким чином, у даному розділі проаналізована предметна область використання сучасних інформаційно-комунікаційних технологій для цілей

допомоги у процесі фізичних тренувань, зокрема із фітнесу. Встановлено, що оскільки заняття фітнесом відбуваються переважно не у домашніх умовах, то використання настільних програмних продуктів не є обґрунтованим у даному випадку. Веб-додатки із зрозумілих причин не можуть функціонувати за відсутності Інтернет-з’єднання, а вбудовані рішення мають достатньо високу вартість. Відповідно, єдиним підходом, що задовольняє вимоги даної предметної галузі, є розробка мобільного додатку.

У розділі проведено аналіз існуючих мобільних додатків, що мають схоже призначення та функціональність. Встановлено, що усі вони або мають перевантажений інтерфейс користувача, або суттєву частину своєї функціональності надають користувачеві за додаткові кошти (тобто не є повністю безкоштовними).

Відповідно, встановлено доцільність розробки власного програмного продукту у вигляді мобільного додатку, що має простий зрозумілий візуальний інтерфейс та вільні умови використання. Розроблено розділ постановки задачі на розробку. Керуючись цією інформацією, можна переходити до подальшого процесу проектування мобільного додатку «Фітнес-щоденник».

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ДОПОМОГИ В ОРГАНІЗАЦІЇ ЗАНЯТЬ З ФІТНЕСУ

2.1 Вибір цільової платформи для розробки

У попередньому розділі було встановлено, що найбільш ефективним способом підвищення ефективності використання систем допомоги у галузі фізичної культури є запровадження власного мобільного додатку, що дозволяв би у значно простішій мірі виконувати типові (перераховані вище) операції на фітнес-заняттях.

Мобільні пристрої, для яких створюються відповідні рішення, діляться на декілька класів, що залежить, в першу чергу, від типу цільової системи, на якій цей додаток буде використовуватися. Так, на сьогоднішній день апаратно мобільні системи можна поділити на смартфони та планшети (зокрема, які повинні бути обладнані функцією доступу в Інтернет).

Переважна більшість планшетів на сьогоднішній день працюють під управлінням ОС Android, а альтернативні системи майже не використовуються (якщо розглядати у відсотковому вираженні).

Серед смартфонів же встановилася чітка тенденція, що значно полегшує розробникам їх роботу в частині вибору цільової платформи. На ринку зараз присутні пристрої під управлінням всього лише двох «великих» ОС: Android та iOS. Інші операційні системи (Windows Mobile, Blackberry OS, Symbian, і т.д.) представлені вкрай мало, тому на старті розробки можуть не розглядатися (версії для цих систем можна розробляти, коли програмні продукти для двох основних згаданих ОС вже повністю готові, відпрацьовані та давно запущені у широке використання).

Отже, Android від Google та iOS від компанії Apple – ці дві системи є абсолютно несумісними, мають різну архітектуру і програмне забезпечення для них розробляється шляхом застосування різних підходів (та, відповідно, засобів розробки). Програмні продукти для Apple розміщуються в Apple

iStore, додатки для Android розміщуються в Google Play Market. Додатки для iOS зводяться у середовищі XCode (яке до речі доступне тільки для OS X, тобто може завантажуватися лише на Mac-системах) мовою Swift або C++. У розробників для ОС Android більше різноманітних інструментів та мов програмування: починаючи від Java [4] і закінчуючи C# [5]. Беручи до уваги усі відмінності між двома цими ОС, розробникам слід визначитися із цільовою платформою і тут слід урахувати відносну поширеність цих продуктів на вітчизняному ринку. Слід відмітити, що, незважаючи на певне переважання «престижності» продуктів від компанії Apple, завдяки кращій ціновій політиці, більшій лінійці доступних продуктів різних фірм (в першу чергу, такого гіганта, як Samsung, але і інших виробників мобільних телефонів: HTC, Huawei, Meizu, і т.д.) можна із впевненістю констатувати, що ОС від Google використовується на значно більшій кількості пристроїв в Україні, ніж Apple (різниця, ймовірно, становить сотні відсотків, тобто «в рази»). Таким чином, якщо обирати між цими двома альтернативами, то більш доцільним на даний момент є створення програмних продуктів саме для Android. Крім того, слід ураховувати, також іміджевий момент, який полягає у тому, що власники пристроїв компанії Apple зазвичай вважають себе більш заможними, або принаймні більш стильними, а для таких осіб у меншій мірі буде властивим використання власних фітнес-щоденників (такі люди частіше будуть шукати послуги «живих» оплачуваних фітнес-тренерів, функцією яких і є слідкування за процесом тренувань клієнта, але не за допомогою часозатратного введення інформації власноруч у програму-щоденник).

Отже, однозначно можна стверджувати що найбільш доцільною є розробка мобільного додатку саме для ОС Android, тому коротко розглянемо її особливості [6]. Історично ця ОС виникла на базі відкритого ядра ОС Linux та спеціально розробленої віртуальної машини Java (програмний продукт для виконання програм, написаних однойменною мовою Java та скомпільованих у проміжний байт-код; такий підхід дозволяє робити програми одночасно і

досить швидкими у виконанні, і повністю кросплатформенними – головне, щоби для цільової платформи існувала Java-машина). У 2005 р. корпорація Google купила компанію Android, яка на той час знаходилася в процесі розробки першої версії ОС, і доробивши її, випустила у 2008 році першу версію. В цілому еволюція цих систем показана на рис. 2.1, з якого видно, що в середньому Google видає по одній новій версії ОС на рік.

Слід звернути увагу, що на сьогоднішній день, незважаючи на те, що сама компанія Google пропонує використовувати інший засіб (який також вбудований у головне середовище розробки Android Studio) – мову програмування Kotlin, все ж таки використання мови програмування Java для розробки мобільних додатків під ОС Android є набагато більш поширеним, тому розглянемо її переваги докладніше нижче.

2.2 Обґрунтування вибору засобів розробки

2.2.1 Вибір технології програмування

Особливістю мови Java, а отже, і усіх програм для Android є те, що вона фактично не допускає структурної (процедурної) технології програмування, а дозволяє лише об’єктно-орієнтовану розробку. Розглянемо особливості цієї технології докладніше.

Суть технології, або як ще кажуть, методології об’єктно-орієнтованого програмування - в тому, що система розглядається, як сукупність окремих сутностей - об’єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон у більшості багатозадачних ОС).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у яких типом виступає клас об’єкта. Клас - це просто опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт - це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності). Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття (декомпозиція) програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу показана на рис. 2.2. З графіка рис. 2.2 слідує, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

Такими є критерії вибору технології програмування, якщо такий вибір взагалі є. В нашому ж випадку доступною в мові Java є тільки ОО-технологія, тому для розробки нашого додатку обираємо її.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

Крім розбиття (декомпозиції) всієї предметної області на об’єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об’єднання даних (значення властивостей класу у деякого конкретного об’єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об’єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Спортсмен є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Спортсмена є свої специфічні властивості і / або методи, які як раз і

відрізняють його від просто Людини: СереднійРезультат, МісцеРейтингу, ПровестиЗмагання(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Спортсмен - як похід в сауну чи на масаж. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і те саме – відновити сили для подальшого життя.

Поліморфізм є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, в нашій програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Спортсмен, Школяр, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Спортсмен, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Таким чином, будемо використовувати об’єктно-орієнтований підхід як більш сучасний (тобто такий, що краще відповідає віянням у світі програмування) і такий, що підтримується майже всіма сучасними мовами програмування, зокрема, мовою Java, що є майже єдиною поширеною мовою програмування для створення додатків під систему Android.

2.2.2 Вибір мови програмування

Мову Java розробила компанія Sun Microsystems на початку 90-х років XX століття. Провідну роль у створенні мови зіграв канадський інженер Джеймс Гослінг (James Gosling). На ранніх етапах розробки мова називалася Oak, а потім її перейменували на честь сорту кави Java (зв’язок мови з напоєм відбивається також у її логотипі).

Джеймс Гослінг і його однодумці хотіли створити мову з C-подібним синтаксисом. У той же час він повинен бути простішим у порівнянні з C/C++. Творці планували використовувати Java для програмування побутової електроніки, проте практично відразу після випуску версії 1.0 в 1995 мову почали використовувати розробники серверного та клієнтського ПЗ. У 2010 році компанію Sun Microsystems купила Oracle, яка, відповідно, на даний час і займається розвитком цієї платформи (деякі прихильники вільного програмного забезпечення звинувачують Oracle у певній дискримінаційній політиці по відношенню до мови Java, через яку вона поступово втрачає позиції у деяких сегментах галузі розробки програмного забезпечення (зокрема, якраз при розробці мобільних додатків власниця ОС Android корпорація Google все інтенсивніше рекомендує мову Kotlin російської розробки, а не більш звичну Java). Тим не менше, на даний час Java все ще залишає за собою статус лідера як основного засобу для створення мобільних додатків.

Java - мова програмування загального призначення. Відноситься до об'єктно-орієнтованих мов програмування, причому з сильною типізацією. Творці мови реалізували принцип WORA: write once, run anywhere або «пиши

один раз, запускай скрізь». Це означає, що написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлено середовище виконання Java (JRE, Java Runtime Environment), або по-іншому, Java-машина.

Це завдання вирішується завдяки компіляції написаного на Java коду у проміжний байт-код. Цей формат і виконує JVM або віртуальна машина Java. Якщо говорити більш точно, JVM - частина середовища виконання Java (JRE). Байт-код, який виконується віртуальною машиною не залежить від платформи і є однаковим для усіх можливих випадків. Сама ж віртуальна машина дороблює байт-код, проводячи його кінцеву компіляцію у машинні коди саме тієї платформи, на якій працює Java-машина. Такий підхід дозволяє поєднати переваги компільованих (висока швидкість виконання програм) та інтерпретованих (повна кросплатформеність) мов програмування та згладити їхні недоліки. Слід відмітити, що компіляція у проміжний код була застосована у мові Java вперше, але потім ця модель була скопійована і на інші платформи. Наприклад, на сьогодні код C# компілюється у так звану проміжну мову (IL – Intermediate Language), а потім уже на конкретній ЕОМ за допомогою платформи .NET дороблюється до машинних кодів.

В Java реалізований механізм управління пам'яттю, який називається прибиральником сміття або garbage collector. Розробник створює об’єкти, а JRE за допомогою прибиральника очищує пам'ять, коли об'єкти перестають використовуватися.

Як зазначалося вище, синтаксис мови Java схожий на синтаксис інших С-подібних мов. Ось його деякі особливості:

- чутливість до регістру - ідентифікатори User і user в Java є різними сутностями;

- для іменування методів використовується lowerCamelCase. Якщо назва методу складається з одного слова, воно повинно починатися з малої літери. Приклад: firstMethodName ();

- для іменування класів використовується UpperCamelCase. Якщо назва складається з одного слова, воно повинно починатися з великої літери. Приклад: FirstClassName;

- назва файлів програми має точно збігатися з назвою класу з урахуванням чутливості до регістру. Наприклад, якщо клас називається FirstClassName, файл повинен називатися FirstClassName.java;

- ідентифікатори завжди починаються з літери (A-Z, a-z), знака $ або нижнього підкреслення _.

Java відноситься до мов програмування загального призначення. За даними компанії Oracle, програми на Java запускаються на 3 млрд девайсів. Це маркетингове повідомлення складно перевірити, проте точно відомо, що Java широко використовується і входить в число найбільш затребуваних мов.

Наприклад, переважна більшість великих компаній так, чи інакше використовують Java. Дуже багато серверних додатків для корпорацій написані на цій мові, зокрема, таких як програми для фінансових організацій, які забезпечують проведення транзакцій, фіксацію торгових операцій. Широке поширення Java у банках та інших фінансових інституціях обумовлене якраз стабільністю додатків на цій мові (відсутність «небезпечних» указників, які використовуються у мові C/C++, автоматичне прибирання сміття, розширена надійна обробка виключень exceptions і т.п.). Через це на Java написано дуже багато веб-додатків. Популярні фреймворки, в тому числі Spring, Stuts, JSP, використовуються для створення різних додатків в Інтернет: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів. Популярна комп'ютерна гра Minecraft написана на Java.

Мобільна розробка - ще одна широка область використання Java, і цією мовою якразпишуть програми для пристроїв, що працюють під управлінням ОС Android.

На Java створюють клієнтські програми. Простий і близький розробникам приклад: IDE NetBeans написано на Java.

Також Java застосовується для роботи з Big Data, розробки програм для наукових цілей, наприклад, обробки природних мов, програмування приладів - від побутових девайсів до промислових установок.

Тобто на Java можна писати різні типи додатків: веб, мобільний і десктопний софт, ігри і так далі. Традиційно у цієї мови сильні позиції в промисловому програмуванні, в сегменті великих компаній (т.зв. ентерпрайз).

Як підсумок можна сказати, що Java - мова програмування загального призначення, яка має С-подібний синтаксис і використовується для створення додатків в різних областях: від Інтернету до розробки ігор, від мобільного ПЗ до програм для корпорацій і наукових інститутів. Саме через ці причини дана мова і обрана у якості основного засобу для розробки мобільного додатку фітнес-щоденника.

2.2.3 Вибір середовища розробки

Слід відмітити, що для мобільної розробки мовою Java існує певний вибір засобів розробки. Так, інтегроване середовище розробки Eclipse, яке до сих пір ще використовується в реальних задачах, поступається потроху, але стабільно середовищу Android Studio від Google, яке, нажаль, має набагато більші системні вимоги. Така ситуація обумовлена тим, що дане середовище є розвитком, адаптацією більш універсального продукту IntelliJ IDEA призначеного для програмування на Java в цілому, до особливостей розробки саме мобільних додатків.

Отже, очевидно, що Android Studio має значно кращі властивості по зведенню проекту додатку, але має і вищі системні вимоги. Оскільки у даному випадку розробник не є лімітованим системними вимогами комп’ютера, приймаємо дане IDE у якості основи для подальшої розробки – рис. 2.3.

2.2.4 Вибір СУБД проекту

Важливим аспектом роботи проектованої системи є алгоритми її роботи з даними – їх обробки та збереження. Відповідно до постановки задачі дослідження сховище даних має бути простим та ефективним, а під ці критерії добре підходить система SQLite.

SQLite - це вбудована реляційна база даних, що поставляється з відкритими вихідними кодами. Вперше випущена в 2000 році, призначена для надання звичних можливостей реляційних баз даних без властивих їм накладних витрат. За час експлуатації встигла заслужити репутацію як переносима, легка у використанні, компактна, продуктивна і надійна база даних.

Вбудовуваність бази даних означає, що вона існує не як процес, окремий від обслуговується процесу, а є його частиною – частиною деякого прикладного застосування. Зовнішній спостерігач не помітить, що прикладний додаток користується СУБД. Додаток просто робить його роботу, движок БД просто міститься всередині. Це позбавляє від необхідності встановлення якогось окремого серверу баз даних (тобто СУБД на зразок MySQL чи Microsoft SQL Server), мережевих налаштувань і адміністрування. Такий підхід дуже сильно полегшує життя програмісту: ніяких фаєрволів, ніяких мережних адрес, ніяких користувачів і конфліктів їх прав доступу. І клієнт, і сервер працюють в одному процесі і це позбавляє від проблем конфігурації, адже усе, чого потребує програміст, уже скомпільовано в його додатку.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті поставки йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина є кросплатформенною утилітою командного рядка.

SQLite можливо використовувати як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.

SQLite підтримує динамічну типізацію даних. Можливі типи значень: INTEGER, REAL, TEXT і BLOB. Так само підтримується спеціальне значення NULL. Розміри значень типу TEXT і BLOB не обмежені нічим, крім константи SQLITE_MAX_LENGTH у налаштуваннях sqlite, яка за умовчанням рівна одному мільярду байтів, чого достатньо для будь-яких даних. Кожне значення в будь-якому полі будь-якого запису може бути будь-якого з цих типів, незалежно від типу, зазначеного при оголошенні полів таблиці. Зазначений при оголошенні поля тип зберігається для довідки у його вихідному написанні, і використовується в якості основи для вибору переваг (так зване «type affinity»: це підхід, що рідко зустрічається в інших СУБД) при виконанні неявних перетворень типів на підставі схожості цієї назви типу на що-небудь, знайоме системі SQLite. В цей алгоритм зашитий великий перелік назв типів даних, що практикуються в інших СУБД. Якщо безпечного перетворення в бажаний тип для значення, що записується, не виходить, то SQLite записує дані у їх початковому вигляді. Для отримання значень з бази є ряд функцій для кожного з типів, і якщо тип значення, що зберігається, не відповідає запитуваному, воно теж, по можливості, перетворюється

Зважаючи на усі ці переваги, SQLite приймається за основне сховище даних для реалізації алгоритмів обробки та збереження інформації проектованого мобільного додатку.

2.3 Розробка структури системи та основних алгоритмів її роботи

Проектована система призначена для створення записів про окремі тренування. У цих записах може міститися попередня інформація про план проведення тренування, цілі та конкретні задачі, одним словом детальний опис майбутнього тренування перед його проходженням. Під час проведення тренування та після нього користувач може передивлятися, додавати або корегувати текстову інформацію про дане тренування, наприклад, описуючи самопочуття або досягнуті результати. При необхідності запис про тренування можна видалити.

Таким чином, перелічені операції, які має реалізовувати програма, підпадають під концепцію CRUD (Create, Read, Update та Delete) – типовий набір операцій із даними. Дана програма, як і взагалі будь-яке ПЗ, створене для цілей обліку, за умовчанням не несе складних алгоритмічних конструкцій. Коли мова іде виключно про облік якихось об’єктів, то процедура (алгоритм) зведення відповідної системи виглядає наступним, досить стандартним чином:

а) створення бази даних, структура таблиць якої є достатньою для збереження інформації про об’єкти, що підлягають обліку, а також необхідні супутні дані. В даному випадку мова іде про досить прості об’єкти – текстові записи про тренування, однак принципово кажучи, необхідною є розробка відповідної БД;

б) проектування процедури внесення у БД інформації про новий об’єкт із сукупності тих, що підлягають обліку – запис про нове тренування. Якщо база даних створювалася для уніфікованих об’єктів одного типу, то ця процедура буде єдиною (принципово майже у всіх випадках можна розглядати різні конкретні способи внесення інформації у довільну БД, однак при описаних особливостях системи, шукати якісь інші способи, окрім одного основного, просто не є доцільним). Таким чином, внесення інформації доцільно виконувати одним SQL-запитом, який частково буде складатися із обов’язкових компонентів, але частково буде формуватися на льоту –

залежно від набору елементів інформації, що описує новий об’єкт, який вноситься у базу.

в) збирання інформації та формування переліку типових операцій зчитування даних із БД. При цьому беруться до уваги різні варіанти пошуку даних: за тим, чи іншим полем, або за певною сукупністю полів.

г) комбінація пунктів *в* та *б* надає ще одну типову операцію, що притаманна будь-якій системі обліку – оновлення (модифікація) даних. Дійсно, для виконання оновлення відомостей про якийсь об’єкт із тих, що зберігаються у системі, спочатку необхідно знайти у БД інформацію про нього (тобто виконати пункт *в*, хоча і без повернення інформації назад користувачеві, але режими пошуку зазвичай є тими ж, що прописані і для простого зчитування даних із бази), а потім виконати пункт *б* (тобто записати у базу відомості про один із об’єктів, що підлягають обліку, але не у кінець БД, а на місце знайденого запису).

д) комбінація пункту *в* та операції видалення записів із БД, тобто спочатку запис слід знайти (за однією із стандартних процедур, які прописані у пункті *в*), а потім стерти.

Переліченим чотирьом типовим операціям б)-д), що в загальному випадку виконує будь-яка програма зберігання однотипних об’єктів, відповідають найбільш вживані команди мови структурованих запитів SQL:

- INSERT – команда внесення даних у базу, причому відбувається саме дописування інформації у кінець вказаної таблиці, яка задіяна у запиті;

- SELECT – найбільш уживана команда SQL, якщо розглянути статистику обробки усіх команд сервером СУБД; використовується для читання інформації із бази даних, яка відноситься до об’єкту, що відповідає певним критеріям пошуку; як уже сказано вище, у практичних задачах використовуються десятки різних запитів типу SELECT, залежно від наявної висхідної інформації для здійснення пошуку; на відміну від трьох інших типових команд, що тут описуються, ця команда зазвичай застосовується для роботи із інформацією з різних таблиць, а не однієї;

- UPDATE – команда оновлення значень у певних полях тих записів таблиці (або одного запису), що відповідає повній умові цієї команди (секція WHERE);

- DELETE – команда повного видалення із заданої таблиці запису, що відповідає повній умові цієї команди (секція WHERE).

Для цих типових операцій можна навести діаграму прецендентів, або як вона дослівно перекладається зі своєї англійською назви діаграму варіантів використання (use case diagram) – рис. 2.4.

Традиційно для мови UML кожному варіанту використання системи має відповідати діаграма активності, яка деталізує спосіб досягнення тієї мети, на яку спрямований кожний варіант використання. Діаграми активності для чотирьох варіантів використання проектованої системи обліку показані на рис. 2.5 – рис. 2.9.

2.4 Висновки по розділу

Таким чином, у даному розділі створено проект мобільного додатку «Фітнес-щоденник», що дозволяє вносити до бази даних записи про майбутні тренування (зокрема їх план), пізніше – переглядати ці записи та вносити до них зміни (зокрема, відмічати результати проведення тренувань).

Також у розділі обґрунтовано використання об’єктно-орієнтованої технології розробки програмного забезпечення, мови програмування Java (яка до сих пір залишається найбільш популярною мовою програмування для

ОС Android) та інтегрованого середовища розробки Android Studio від Google.

Таким чином, керуючись даною інформацією, можна переходити до програмної реалізації мобільного додатку «Фітнес-щоденик».

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ДОПОМОГИ В ОРГАНІЗАЦІЇ ЗАНЯТЬ З ФІТНЕСУ

3.1 Прототипування та реалізація інтерфейсу користувача мобільного додатку

Консультації з клієнтами фітнес-центрів та індивідуальними спортсменами показали, що для них не є важливою наявність гарних продвинутих елементів управління, а для них головною є функціональність продукту, яка полягає у здійсненні зручного (у порівнянні зі звичайним записом у блокноті чи зошиті) зберігання інформації, а також у наявності певних простих додаткових функцій, таких як таймер виконання вправ, тощо. Відповідно, інтерфейс розроблюваного програмного забезпечення може бути наближений до мінімалістичного.

Головним елементом управління буде елемент типу ListView – рис. 3.1, який буде заповнюватися записами-повідомленнями, що знаходяться у локальній базі даних SQLite. Уся організація системи, що проектується, наведена на рис. 3.1.

Як уже зазначено вище, інформація зберігатиметься на пристрої у форматі XML і вносити до неї зміни можна вручну, набираючи усі необхідні теги, відповідно до інструкції користувача системи, або шляхом використання спеціального веб-додатку, що збирає усю необхідну інформацію через форму (використовуючи зручні для людини елементи управління HTML), а потім автоматично формує необхідний XML-файл, розміщуючи введену змістову інформацію по тегам, які введено у даній роботі. Створення такого веб-додатку може бути розширенням даної роботи, яке виконуватиметься у перспективі (якщо розробка отримає реальне практичне впровадження).

Окрім основного екрану, де будуть списком видаватися усі актуальні повідомлення про роботу фітнес центру, також реалізуємо екран налаштувань.

3.2 Проектування бази даних додатку

Зважаючи на особливості предметної галузі база даних проекту є надзвичайно простою і в основному будується на таблиці Trainings із записами про тренування. У дану таблицю впроваджено наступні поля:

- код тренування, первинний ключ таблиці;
- запланована дата тренування;
- фактична дата проведення тренування;
- план тренування, текстове поле;
- результати тренування, текстове поле;
- примітки, текстове поле.

Така таблиця створюється у мобільному додатку, що розробляється, за допомогою наступного коду на мові Java, який включає код на структурованій мові запитів SQL:

```
public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE trainings ("
                + "TrainingID INTEGER PRIMARY KEY,"
                + "TrainingPlanDate TEXT,"
                + "TrainingRealDate TEXT,"
                + "TrainingPlan TEXT,"
                + "TrainingResults TEXT" + ");");
}
```

Оскільки мобільний додаток, що розробляється, призначений для використання тільки однією особою, то впроваджувати якісь відомості для авторизації немає сенсу. Відповідно, на цьому проектування та реалізацію бази даних додатку можна вважати завершеними.

3.3 Особливості реалізації алгоритмів роботи додатку

Відповідно до описаного вище завдання проектування програмна реалізація не міститиме якихось складних алгоритмів по інтелектуальній обробці даних. Суть роботи проектованого додатку полягає у виконанні інформативно-комунікаційної функції: веденні переліку поточної інформації, виведенні її на екран мобільного пристрою. Тому коротко опишемо відповідні прості алгоритми та відповідні програмні складові (реалізацію цих алгоритмів), на основі яких працює додаток.

Отже, розглянемо функції-члени (методи), що реалізують алгоритми (функціональність) проекту. У файлі (класі) `Message`, який уособлює основну базову одиницю запису інформації в проекті – повідомлення (запис про тренування), і є найменшим по об’єму є:

1-4) Функції «геттери» для усіх чотирьох властивостей класу (дати повідомлення, детальної інформації про нього, величини важливості повідомлення та посилання, пов’язаного із ним):

```
public Date getDate() { return date; }
public String getDetails() { return details; }
public double getMagnitude() { return magnitude; }
public String getLink() { return link; }
```

5) Конструктор класу, який є стандартним:

```
public Message(Date _d, String _det, double _mag, String _link) {
    date = _d;
    details = _det;
    magnitude = _mag;
    link = _link;
}
```

6) Функція, необхідна для серіалізації об’єктів класу, тобто така, що зафіксовує особливості переведення об’єкту у текстовий рядок:

```
public String toString() {
    SimpleDateFormat sdf = new SimpleDateFormat("HH.mm");
    String dateString = sdf.format(date);
    return dateString + ": " + magnitude + " " + details;
}
```

Наступним, більш складним по структурі й функціональності класом є клас `Preferences`. Усі поля цього класу мають суто системний характер, і необхідні для посилань (тобто для організації можливості оперування з) елементами управління активності `Preferences`. Методи цього класу мають наступний зміст:

1) Функція-обробник події створення активності:

```
public void onCreate(Bundle icicle) {
```

Тут виконується налаштування активності Preferences перед тим, як її показати та почати використовувати. Тут програмно за допомогою функції `setOnClickListener` встановлюються два обробники для натискань на кнопки OK та Cancel (зважаючи на динамічний характер створення цих обробників у них навіть немає імені).

2) Окремий метод, який заповнює усі елементи управління типу Spinner доступними значеннями, які задаються окремо у ресурсах:

```
private void populateSpinners() {
```

3) Метод для заповнення елементів управління, що містяться на активності Preferences:

```
private void updateUIFromPreferences() {
```

4) Метод для збереження налаштувань:

```
private void savePreferences() {
```

Найскладнішим і основним класом програмного продукту є клас `FitnessCentre`, який задає головну активність додатку. Він містить наступні властивості:

```
ListView messageListView;
ArrayAdapter<Message> aa;
ArrayList<Message> messages = new ArrayList<Message>();

Message selectedMessage;

int minimumMagnitude = 0;
boolean autoUpdate = false;
int updateFreq = 0;
```

Із них:

- `messageListView` – елемент управління для списку усіх повідомлень;
- `selectedMessage` - об’єкт для збереження одного обраного повідомлення;
- три останніх поля – значення налаштувань проекту за умовчанням (без автоматичного оновлення, показувати усі повідомлення, оновлювати кожної хвилини).

Методи цього класу та їх алгоритмічна суть:

1) Функція-обробник події створення активності:

```
public void onCreate(Bundle icicle) {
```

Тут виконується налаштування активності `FitnessCentre` перед тим, як її показати та почати використовувати. Тут програмно за допомогою функції `setOnClickListener` встановлюється обробник для натискання на одне із

повідомлень і видачі детальної інформації про нього (зважаючи на динамічний характер створення цього обробника у нього немає імені).

Тут також завантажуються раніше збережені налаштування додатку – функцією `updateFromPreferences();`.

Тут також присутній виклик функції `refreshMessages();`, яка відображує список усіх повідомлень, що відповідають заданому рівню значимості.

2) Функція, яка безпосередньо зчитує вхідний XML-файл з повідомленнями і відображує їх у списку:

```
private void refreshMessages() {
```

3) Функція додавання нового повідомлення у загальний список:

```
private void addNewMessage(Message _message) {
```

4) Функція-обробник події створення меню налаштувань:

```
public boolean onCreateOptionsMenu(Menu menu) {
```

5) Функція-обробник вибору однієї із доступних опцій програми:

```
public boolean onOptionsItemSelected(MenuItem item) {
```

6) Функція створення діалогового вікна із детальною інформацією про вибране повідомлення:

```
public Dialog onCreateDialog(int id) {
```

7) Функція підготовки вікна відповідного діалогу та заповнення його текстових та інших властивостей:

```
public void onPrepareDialog(int id, Dialog dialog) {
```

8) Функція, що здійснює оновлення роботи програми (зокрема, списку повідомлень) після виходу із активності налаштувань, відповідно до нових налаштувань:

```
private void updateFromPreferences() {
```

9) Функція-обробник події повернення із меню налаштувань:

```
    public void onActivityResult(int requestCode, int resultCode, Intent data)
{
```

Таким чином, розглянуто основні алгоритмічні дії, що виконуються у даному програмному забезпеченні. Ще раз відмітимо, що якоїсь складної серйозної обробки даних технічне завдання не передбачає, тому всі алгоритміні конструкції зводяться до викликів розглянутих функцій-методів та виконання їх ділянок коду.

Спираючись на основні алгоритмічні конструкції та їх програмні реалізації, розглянуті вище, можна переходити до розгляду конкретної структури проекту мобільного додатку для фітнесу – рис. 3.2.

На рисунку видно, що у складі проекту присутні наступні програмні складові:

а) Пакет `com.paad.fitnesscentre`, у якому виділено три класи:

- `FitnessCentre`, який відповідає головній активності розроблюваного додатку;
- `Message`, який містить інформацію про одну базову одиницю інформації, з якою працює програма – повідомлення, запис про тренування;
- `Preferences`, що відповідає окремій активності, яка з'являється, коли користувач обирає пункт Preferences у головній активності, і яка відображує усі доступні для налаштувань опції програми.

б) Далі у складі проекту слідує розділ `gen`, який генерується автоматично самим середовищем на основі відомостей про ресурси проекту. Цей розділ не слід змінювати вручну, оскільки будь-які виправлення будуть перезаписані.

в) Наступним пунктом іде системна папка Android 4.2 – відповідно до версії SDK, на якій збирається код додатку (в процесі проектування вирішено використати досить стару версію Software Development Kit – сьому – для того, щоби програмний продукт можна було поширювати навіть на дуже старих смартфонах з операційною системою версії не нижче Android 4.2; також це обумовлене тим, що програмний продукт не використовує і не потребує якоїсь складної функціональності, що з'являлася у нових версіях Android, тому із міркувань підвищення переносимості продукту обрано досить стару версію ОС). Цей пункт також змінювати не потрібно.

г) Ще одна системна папка Android Dependencies містить інформацію про програмні залежності даного продукту від інших системних компонентів Android.

д) Порожня папка assets, у якій мають розміщуватися ті файли ресурсів, які не включаються до системної змінної R (в нашому проекті таких файлів немає).

е) Папка bin, що містить двійкові файли проекту. Тут інтерес представляє сам виконуваний файл проекту FitnessCentre.apk, який можна поширювати.

є) Папка з ресурсами, які включаються до складу змінної R. Тут містяться XML-файли з текстовими рядками проекту.

Таким чином, розглянуто структуру проекту розробленого мобільного додатку і можна переходити до опису методики роботи з ним.

3.4 Тестування та аналіз результатів роботи додатку

В першу чергу, згадаємо існуючі варіанти поняття тестування програмного забезпечення в цілому і мобільних додатків зокрема.

По-перше, різними можуть бути цілі самого тестування. Очевидно, найпершим по важливості у переліку цілей є належне виконання програмою задач, покладених на неї розробником, іншими словами – належне виконання

власних функцій. Таке тестування, відповідно, називають функціональним. Наступними цілями, що відносять процес тестування до нефункціонального класу, є:

а) тестування продуктивності, адже одна програма може виконувати певну задану обробку вхідних даних протягом хвилини, а інша – за десять хвилин. В той же час перша програма при цьому може затребувати для своєї роботи 8 Гб оперативної пам’яті, а друга – всього лише 1 Гб. Очевидно, що продуктивність сильно залежить від алгоритмів, які були застосовані розробниками при реалізації програмного продукту, в т.ч. і другорядних, тобто таких, що не були окреслені у технічному завданні (ТЗ) на розробку програми. Саме на цьому етапі також уточнюються системні вимоги програми, тобто мінімальні характеристики ПК, на якому ще буде забезпечуватися виконання програмою своєї функціональності при адекватних часах обробки даних. Зазвичай, мінімальні системні вимоги прописуються у ТЗ, але вони можуть бути переузгоджені після проведення тестування продуктивності, причому як у меншу, так і у більшу сторони.

б) тестування інтерфейсу користувача (User Interface, UI), при якому оцінюється виконання розробниками стандартних (загальноприйнятих) правил створення графічних інтерфейсів користувача, або у випадку застосування незвичайних (інноваційних) елементів – комфортність їх використання «середнім» користувачем. Тут оцінюються кольорові схеми, розміри елементів управління, їх взаємне розташування, та інші аспекти зовнішнього вигляду програми.

в) тестування доступності функцій ПЗ та зручності користування ними для «середнього» користувача, яке включає перевірку інтуїтивності та зрозумілості елементів інтерфейсу програми, а при використанні незвичайних рішень – їх адекватність функціональності самого ПЗ (наприклад, для дитячої пізнавальної програми використання елементу управління «рожева повітряна куля, що стрибає» є надзвичайно ефективним, а для корпоративного ПЗ – недопустимим, і т.п.).

г) тестування захищеності ПЗ, тобто його здатність протидіяти як випадковим, так і зловмисним діям користувача, що можуть викликати збої в роботі ПЗ. В першу чергу, тут мають тестуватися усі введення, що здійснює користувач, особливо на традиційну помилку переповнення буферу (SQL-ін'єкцію, якщо використовується база даних, і т.п.).

д) тестування процесу встановлення програми, на якому слід упевнитися, що при усіх можливих варіантах опцій, виставлених користувачем, а також допустимих конфігураціях апаратного забезпечення ПК процес інсталяції проводиться адекватно і, не викликаючи помилок, приводить до бажаного результату.

е) тестування сумісності, яке в першу чергу стосується різних версій цільової операційної системи (наприклад, Windows 7, 8, 10), а також бібліотек, драйверів та іншого стороннього програмного забезпечення, яке використовується програмою.

є) тестування надійності, яка, як і у випадку технічних систем, описує роботу програми на тривалих інтервалах часу (в першу чергу, слід встановити, чи не накопичуються помилки при довгому використанні програми).

ж) тестування локалізації, направлене на перевірку якості та адекватності перекладу усього тексту, вбудованого у програму. Очевидно, має виконуватися людиною, що добре знає цільову іноземну мову, а краще – її носієм.

Також види тестування можуть розрізнятися за ступенем автоматизації при проведенні тестування. Очевидно, що робота по тестуванню може бути повністю ручною, мануальною, коли усі дії по перевірці програми виконує людина. В той же час, не має фізичної можливості людині перевірити, наприклад, тисячі варіантів вводу користувача у певне числове поле (а таких полів, зазвичай, у програмі є досить багато!), але за допомогою засобів автоматизації процесу тестування така перевірка стає можливою: необхідно лише, щоби завдання цього числа виконувалося спеціально написаною

програмою, яка би і перевіряла отриманий після цього результат. Або задавати вхідні дані може одне програма тестування, а перевіряти, наприклад, файли-результати може інша програма.

Третьою ознакою для класифікації є орієнтування на нормальні, чи незвичайні (екстремальні) умови роботи з програмою. Позитивною називають перевірку, направлену на оцінку дій програми у стандартних умовах її використання. Негативним є тестування програми у нестандартних умовах, коли поведінка користувача, або інших впливових зовнішніх факторів відрізняється від звичайних їх значень.

За рівнем доступу до коду ПЗ тестування може розділятися на наступні типи:

- по схемі чорного ящика, коли тестувальник взагалі немає доступу до коду ПЗ, а працює лише задаючи різні вхідні дані, та оцінюючи результат;

- по схемі білого ящика, коли людина-тестувальник має повний доступ до коду;

- по схемі сірого ящика, коли доступу до коду немає, але тестувальник добре представляє структуру програми, склад її модулів, класів, функцій чи менших блоків коду (без надання самого вихідного тексту).

За рівнем блоку коду, що підлягає тестуванню, виділяють наступні варіанти:

- модульне або юніт-тестування, при якому перевірці підлягає один блок, як, наприклад, клас, чи функція, а іноді – окремий модуль програми;

- інтеграційне тестування дозволяє перевірити взаємодію окремих модулів, адже навіть при їх ідеальному відлагодженні, можлива наявність неузгодженості між даними, що передаються, і в результаті програма працюватиме невірно;

- системне тестування направлене на перевірку роботи всієї програми в цілому;

- приймальне тестування передбачає перевірку того, чи відповідає розроблена програма усім пунктам наявного ТЗ.

Різні виконавці тестування визначають його розділ на альфа та бета тестування. Альфа-тестування виконують самі розробники і воно направлене на виключення усіх очевидних помилок, які змогли знайти власне програмісти. Після його завершення розробники вважають, що програмний продукт уже завершений і передають його іншим особам – бета-тестувальникам, які можуть утворювати певну фокус-групу, або бути окремими віддаленими добровольцями, і т.п.

Також тестування може розділятися за формальністю цього процесу, що, взагалі кажучи, визначається ступенем підготовки тестувальника до самого процесу. Тут можна виділити:

- проведення роботи по тестам, що зарані розроблені (можливо, одразу після формування ТЗ, або пізніше) та утворюють певний банк тест-кейсів;

- дослідницьке тестування, що характеризується одночасною розробкою тестів та їх негайним використанням;

- вільне тестування, що проводиться без розробки спеціальних тестів, а виключно у режимі користувача, який просто використовує ПЗ для своїх різноманітних цілей (якість цього варіанту сильно залежить від досвіду тестувальника, та, взагалі кажучи, її неможливо перевірити, тому користуватися цим типом тестування слід із обережністю, виключно із повністю перевіреними тестувальниками).

Нарешті, тестування також може розділятися на класи по важливості тієї функціональності, що підлягає перевірці під час даного процесу:

- димове тестування, коли оцінюється лише виконання самих важливих складових функціональності;

- тестування критичного шляху перевіряє типові послідовності дій, які виконують у повсякденній роботі з програмою звичайні її користувачі;

- розширене тестування – повна і найбільш глибока перевірка усієї функціональності програми.

Таким чином, беручи до уваги можливі варіанти проведення процесу тестування, можна сказати що у даній роботі доцільно було провести системне бета-тестування, яке і описується у подальшому викладі.

Робота додатку, традиційно при розробці програмного забезпечення для Android, тестувалася в рамках емуляторів, що є доступними для середовища розробки. Для виконання емуляції мобільного пристрою доступною є досить велика кількість опцій повний перелік яких наведено на рис. 3.3.

За необхідності властивості віртуального пристрою можуть бути відредаговані – рис. 3.4. Те ж саме вікно викликається при створенні нового віртуального пристрою.

Тестування на наявних в системі віртуальних машинах (рис. 3.5) показало стабільну роботу додатку у різних вхідних умовах. Спостерігалося швидке оновлення інформації при змінах у висхідному XML-файлі.

Окрім тестування на віртуальній машині, було також проведено дослідження стабільності роботи додатку на реальному пристрої під керуванням ОС Android. Для цього, по-перше, слід перейти до режиму розробника, щоби відкрилися додаткові пункти в меню налаштувань телефону. У більшості моделей (від версії Oreo 8 та вище) для цього треба виконати наступні кроки:

- зайти у меню «Налаштування» > «Система» > «Про телефон»;

- знайти номер зборки і клацати по ньому пальцем багато разів підряд;

- слід припинити клацати, коли з’явиться напис, що «Тепер Ви є розробником»;

- повернутись в меню «Налаштування» > «Система»;

- тепер у ньому має з’явитися новий пункт «Параметри розробника».

У даному меню слід знайти та включити пункт «Режим налагодження по USB». Після цього слід приєднати звичайним USB-кабелем телефон до

ПК і запустити середовище Android Studio, яке в автоматичному режимі має «підхопити» смартфон і він стане доступним для проведення на ньому налагодження (DEBUG) мобільного додатку – рис. 3.6.

Запуск мобільного додатку на телефоні також показав його адекватну та стабільну роботу у відповідності до технічного завдання на розробку. Екран розробленого мобільного додатку у робочому стані можна бачити на рис. 3.7 – рис. 3.8. Таким чином, робота розробленого програмного продукту відповідає необхідним вимогам до мобільного програмного забезпечення, встановленим у підрозділі 1.4.

3.5 Висновки по розділу

Таким чином, у розділі описано проведення програмної реалізації мобільного додатку фітнес-щоденнику. Програмний продукт створений для сучасної і надзвичайно поширеної операційної системи Android (причому через спеціальним чином виконані налаштування проекту, він може бути завантажений навіть на дуже стару версію ОС 4.2, яка все ще є поширеною серед користувачів мобільних пристроїв).

У розділі описується розробка інтерфейсу користувача додатку, який із самого початку було вирішено робити мінімально необхідним (для спрощення його сприйняття звичайними користувачами, які не є професіоналами у галузі ІТ). Наводиться аналіз бази даних, що покладена в основу роботи додатку, розглядається структура проекту програмного продукту, а також його складові: властивості та, в основному, методи.

Також у розділі описано результати тестування розробленого мобільного додатку, яке виконувалося у двох режимах:

- спочатку (під час розробки) мобільний додаток тестувався на віртуальному емуляторі ОС Android;

- після завершення розробки тестування було також проведено і на реальному смартфоні.

В обох випадках було встановлено відповідність отриманого програмного продукту вимогам розділу постановки задачі даної роботи.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Таким чином, у даній роботі виконано проектування та реалізацію мобільного програмного забезпечення на базі поширеної ОС Android, призначене для людей, що займаються фітнесом, зокрема в рамках їх роботи у фітнес-центрах, а також по індивідуальним програмам. Спочатку проаналізовано потреби кінцевих потенційних користувачів продукту та виявлено, що найбільш доцільними діями є запис планів майбутніх тренувань, а після їх проведення – опис отриманих результатів та інших відомостей, що відносяться безпосередньо до тренування. Також користувачі висловили побажання отримати в рамках даного програмного продукту деякі невеликі додаткові можливості, такі як, наприклад, підрахунок часу тренування (типу таймера).

Для вказаних можливостей розроблено алгоритмічні складові, а саме діаграми мови UML, на основі яких можна проводити подальшу реалізацію алгоритмів додатку.

У якості основних методів та інструментальних засобів для розробки мобільного додатку фітнес-щоденника обрано об’єктно-орієнтовану технологію програмування, а також мову програмування загального призначення Java, яка є традиційною для розробки програм під ОС Android. Розробку вирішено вести безпосередньо у інтегрованому середовищі Android Studio від компанії Google (яка на сьогодні і має права на дану операційну систему).

У роботі виконано розробку інтерфейсу користувача додатку, описується зведення його бази даних, що створена за допомогою інструменту SQLite, що дозволяє вбудовувати процеси по роботі з базою даних безпосередньо у сам мобільний додаток (не потрібно встановлювати окрему СУБД). Також описується проектування програмних складових додатку, в якому реалізовано 3 основних класи, докладно описуються їх методи.

Проведено тестування створеного мобільного додатку, причому як на віртуальному емуляторі Android, так і на реальному фізичному пристрої-смартфоні, і воно показало стабільність роботи додатку та належне виконання ним його функціональності відповідно до постановки задачі дослідження. В цілому програма може застосовуватися в реальній фітнес-діяльності як при індивідуальних заняттях, так і в рамках тренувань у фітнес-центрах.