

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ РОЗРАХУНКУ ВАРТОСТІ
РОБОЧОГО ЧАСУ

Виконав: студент 4 курсу, групи КНТ-517
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

Демченко Андрій Сергійович

(прізвище та ініціали)

Керівник Скрупській С.Ю.

(прізвище та ініціали)

Рецензент Шило Г.М.

(прізвище та ініціали)

м. Запоріжжя
2021 рік

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

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

“ ” 2021 року

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

Демченко Андрія Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Комп'ютерна система розрахунку вартості робочого часу

керівник проекту (роботи) Скрупський Степан Юрійович, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “17” березня 2021 року № 81

2. Строк подання студентом проекту (роботи) 06 травня 2021 року

3. Вихідні дані до проекту (роботи) наявні методи та алгоритми пошуку об'єктів на графічному зображенні, час обробки одного регіону розпізнавання має не перевищувати 10 мс, рівень коректних спрацювань не менш 70%

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області;

2) Проектування комп'ютерної системи;

3) Розробка апаратної специфікації та інфраструктури;

4) Розробка програмного забезпечення системи.

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

Блок – схема алгоритму розрахунку вартості часу;

Порівняльна характеристика використаних технологій;

Макети сторінок комп'ютерної системи;

Демонстрація роботи комп'ютерної системи.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
1-4	Скрупський С. Ю., к.т.н., доцент		
нормоконтроль	Щербак Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів пошуку об'єктів, перегляд отриманих результатів, виділення недоліків та переваг	01.03.2021 р.	
2	Розробка структури системи	15.03.2021 р.	
3	Розробка алгоритму роботи системи	20.03.2021 р.	
4	Реалізація системи уточненого розпізнавання	01.04.2021 р.	
5	Дослідження системи уточненого розпізнавання	10.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	01.05.2021 р.	
7	Оформлення графічного матеріалу	10.05.2021 р.	
8	Оформлення допоміжного матеріалу	17.05.2021 р.	

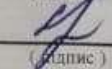
Студент



А.С. Демченко

(ініціали та прізвище)

Керівник проекту (роботи)



С.Ю. Скрупський

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 65 с., 21 рис., 2 таблиці, 25 лістингів, 29 джерел

ANDROID, API, FIREBASE, JAVA, MYSQL, БАЗА ДАННИХ, ВАРТИСТЬ ЧАСУ, РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

Об'єктом проектування є розробка комп'ютерної системи на базі платформи Android для підрахунку вартості часу користувача.

Метою проектування є розробити портативний інструмент для оцінки вартості часу людини.

Методами та стандартами проектування є розробка дизайну та написання коду на мові високого рівня програмування Java.

Результатом проектування є система, що демонструє функціонал оцінки вартості часу та надання статистики на основі зібраних даних.

Отриманий додаток можна підтримувати шляхом додавання нового функціоналу.

ЗМІСТ

Скорочення та умовні позначки.....	7
Вступ.....	8
1 Аналіз предметної області	10
1.1 Аналіз об'єкту розробки	10
1.2 Аналіз іде для розробки клієнтської частини	11
1.3 Аналіз субд та бд	14
2 Проектування комп'ютерної системи.....	16
2.1 Аналіз вимог до додатка	17
2.2 Побудова алгоритму роботи кс	19
2.3 Побудова діаграми класів	20
2.4 Макети сторінок android додатку.....	22
2.5 Структура бази даних.....	25
3 Розробка апаратної специфікації та інфраструктури	27
4 Розробка програмного забезпечення системи	31
4.1 Створення проекту у іде.....	31
4.2 Розробка клієнтської частини додатку	32
4.3 Розробка розмітки фрагментів на основі дизайну.....	35
4.4 Розробка клієнтського функціоналу додатку.....	42
4.4.1 Розробка функціоналу центральної сторінки	48
4.4.2 Розробка функціоналу таймера	50
4.4.3 Розробка функціоналу лівої сторінки.....	52
4.4.4 Розробка функціоналу правої сторінки	55

	6
4.4.5 Налаштування realtime database	57
4.4.6 Випробування системи.....	58
Висновки.....	61
Перелік джерел посилання.....	63

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

КС - комп'ютерна система

ПЗ - пояснювальна записка

СУБД - система управління базами даних

API - інтерфейс програмування додатків, Application Programming Interface

APM - архітектура мобільних процесорів, Advanced RISC Machine

IDE - інтегроване середовище розробки, Integrated Development Environment

SDK - набір інструментів для розробки додатків, Software Development Kit

ВСТУП

У сучасному світі найдорожчою валютою є час. Його обмінюють на гроші, ходячи на роботу, дарують своїм близьким, проживаючи з ними спільні моменти, або вкладають у самих-себе [1]. Навіть за перегляд стрічки Instagram або будь-якого іншого медіа сервісу людина платить своїм часом.

Наглядним прикладом монетизацій є перегляд реклами, де час людини конвертується у заробіток творців контенту та рекламодавців.

Монетизація – це отримання доходу за допомогою своїх веб-ресурсів [2] (надалі монетизація розглядатиметься, як конвертація часу в грошові одиниці).

З цього слідує, що питання раціонального розподілу часу стає перед усім. Кожна людина планує свій день, зокрема, роботу, відпочинок та інші буденні справи, все це займає різну кількість часу та надати кожній справі вагову оцінку, без застосування чітких цифр, важко. Адже потрібно враховувати не тільки час необхідний для виконання поставленої задачі, а і потенціальну суму монетизації свого часу.

Виникає необхідність в інструментах які будуть в автоматизованому режимі відповідати на питання: виконати поставлену задачу самостійно, чи делегувати цю справу, заплативши кошти [3].

Якщо дивитись з точки зору оптимізації робочого часу та роботи, то даний інструмент повинен відповідати на друге важливе запитання: скільки коштує робота конкретної людини.

Таке запитання необхідно ставити тому, що наразі йде найбільший розвиток дистанційної роботи, або роботи на фрілансі.

Фрілансер – це людина без постійного і частіше без фізичного місця роботи, де вона на пряму працює з замовником [4].

Стрімкий розвиток цього напрямку роботи спровокувала пандемія 2019-2021рр.

Нові зміни потребують нових рішень. Тож, постає потреба в об'єднанні у повсякденному житті відпочинку та роботи.

Інструмент що задовільнить потреби у цих двох напрямках повинен мати наступні риси:

- мобільність;
- простий та інтуїтивний інтерфейс;
- наочність розрахованих даних;
- наявність статистики.

Для того, щоб здійснити поставлену задачу, необхідно виконати такі пункти:

- зробити аналіз предметної області;
- спроектувати комп'ютерну систему;
- розробити апаратні специфікації та інфраструктуру;
- розробити програмне забезпечення цієї системи та протестувати його.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз об'єкту розробки

Основна ціль алгоритму - розрахувати вартість однієї години користувача, що б на основі цих даних надати йому статистичну інформацію що-до продуктивності робочого дня та допомогти в раціональному плануванні витрат власного часу.

Для реалізації необхідно обрати вид пристрою який завжди буде у швидкому доступі, за для комфортного внесення даних. Під цю характеристику підходять лише смартфон, смарт-годинник та ноутбук. У свою чергу смарт-годинник занадто малий, а ноутбук занадто великий, тож оптимальним варіантом є смартфон. Тому, алгоритм матиме свою реалізацію у вигляді мобільного додатку.

Мобільний додаток – це програмне забезпечення, розроблене для використання на смартфонах, планшетах, та інших мобільних пристроях на базі платформ: Android, IOS та інших, менш поширених платформах [5]. Тож вибір постає між платформою Android та IOS.

Android – це платформа для мобільних технологій, яка забезпечує смарт пристрої від смартфона до телевізорів потужністю та мобільністю операційної системи Linux, а також надійністю та стабільністю мови програмування високого рівня (Java, Kotlin) [6].

IOS – це операційна система корпорації Apple, яка ставилась на усі смартфони лінійки Apple починаючи з iPhone 3, а також на деякі інші пристрої компанії.

При детальному порівняльному аналізі на основі табл. 1.1 фаворитом, для першої версії реалізації алгоритму та першого ринку збуту, виходить платформа Android, адже вона має більший охоплення потенціальних користувачів додатком при менших витратах на виробництво.

Для написання додатків на платформі Android існує дві найпопулярніші мови програмування високого рівня: Java та Kotlin. Обидві мови компілюються до байт-

коду, тож їх можна використовувати одночасно в одному проєкті, якщо це необхідно, адже кожна з них має свої переваги та недоліки.

Java – це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems у 1991 році та офіційно випущена 23 травня 1995 року [7].

Kotlin – це мова програмування компанії JetBrains, яка за 9 років стала міцним конкурентом Java для Android розробки [8].

Тож об'єктом розробки є алгоритм розрахунку вартості часу людини, а саме однієї години, у вигляді мобільного додатку на базі платформи Android.

Таблиця 1.1 – Порівняння платформи Android та IOS для розробки [9]

Порівняльні параметри	Android - пристрій	IOS - пристрій
Де цільова аудиторія	Більш масовий та орієнтований на середній або бюджетний ринок	Орієнтована на преміум користувачів
Який додаток дешевше розробити	Розробка більш дешева	Розробка дорожча
Тип монетизації: покупки у додатку/реклама	Занизька покупна спроможність, монетизація рекламою	Покупна здатність у 5 разів вища, монетизація покупками у додатку

1.2 Аналіз IDE для розробки клієнтської частини

Для розробки під платформу Android є безперечні два фаворити – IDE Android Studio та IDE Eclipse.

IDE – це інтегрована середовище розробки (Integrated Development Environment), комплекс засобів, маюча все необхідне програмісту для створення програмного забезпечення [10].

Eclipse – вільна інтегрована середовище розробки модульних крос платформних додатків [11].

Android Studio – програмне забезпечення, яке являє собою середовище розробки додатків для мобільної платформи Android [12].

Для комфортної розробки необхідна наявність великого набору функцій, які орієнтовані саме під розробку на платформі Android так, чим більш вузько направлена середовище розробки, тим легше в ній розібратись і використовувати всі її можливості на максимум, також велику перевагу має оптимізація, яка буде наявна на основі вище перерахованих параметрів. Що дозволить додатково зекономити кошти адже для розробки буде потрібен працівник з більш низьким рівнем кваліфікації.

Час, який буде використано для розробки додатка, носить найважливіший характер. Дуже сильно прискорює швидкість розробки наявність можливості у режимі реального часу переглядати зовнішній вигляд додатку при його написанні, і можливість зручно переглядати структуру проекту та працювати з нею. Приємним додатком буде наявність можливості швидкого обміну файлами проекту з системою контролю версій, а саме GIT.

GIT – це розподілена система контролю версій, яка надає можливість розробникам відстежувати зміни в файлах і працювати над одним проектом разом з колегами [13].

Наявність цієї платформи пришвидшує роботу, бо збільшується швидкість обміну файлами між програмістами. Додатково покращується рівень безпеки розробки. Якщо щось трапиться із даними на робочій машині програміста, він завжди зможе завантажити останню збережену версію проекту з GIT. Є можливість ділити код на вітки, що дозволяє створити декілька версій коду одночасно та за потреби відмінити зміни у проекті, завантаживши попередню версію.

Кожна з платформ має свої переваги та недоліки, на основі детального порівняльного аналізу, спираючись на табл. 1.2, та порівняння їх з вище перерахованими потребами, для розробки даного додатку більше підходить IDE Android Studio. Вона має великий вибір інструментів, гнучкість в розробці,

можливість тестування, підтримка декількох найпопулярніших мов розробки та має вбудований емулятор, що робить її найкращою у своєму класі.

Таблиця 1.2 – Порівняння IDE Android Studio та IDE Eclipse. [14]

Характеристики	IDE Android Studio	IDE Eclipse
Підтримка компанії Google	Так	Ні
Наявність емулятора	Так	Ні
Вузько спеціалізований напрям	Так	Ні
Підтримка декількох мов розробки	Так	Так
Підтримка мови C/C++	Так	Так
Наявність російської мови інтерфейсу?	Так	Так
Потреби потужності системи	Потребує великого об'єму ресурсів	Потребує малого об'єму ресурсів
Платне чи безплатне програмне забезпечення?	Безкоштовне	Безкоштовне
Наявність системи контролю версій GIT	Так	Ні

1.3 Аналіз СУБД та БД

Для ідентифікації користувачів додатка та зберігання їх статистичної інформації необхідно використання бази даних.

Найпопулярніше рішення у використанні баз даних є СУБД – MySQL з запитамі на мові SQL

СУБД – це система управління базами даних, являє собою комплекс програмного забезпечення, завдяки якому можна створювати бази даних та проводити над ними різні операції [15].

MySQL – це революційна система управління базами даних з відкритим кодом [16].

SQL – це мова програмування структурованих запитів, які використовуються у якості ефективного способу зберігання даних та їх редагування [17].

Є ще не мало різних СУБД під свої БД зі своїми перевагами та недоліками але всі вони виконують роль саме бази даних, зберігають та редагують інформацію, і все, для розширення їх функціоналу потрібно використовувати сервер, який буде виступати посередником між клієнтом і базою даних. Але це ускладнить розробку та потребує багато ресурсів, починаючи з грошей для оплати оренди сервера і закінчуючи додатковими фахівцями та часом розробки, який збільшиться щонайменше у два рази.

Тому необхідно роздивитись інші варіанти, і таким варіантом є сервіс від корпорації Google – Firebase, якій має в собі інструмент Realtime Database та Authentication, інтерфейс взаємодії з яким наявний у Android Studio.

Firebase – це комплексна платформа для розробки веб-додатків і в першу чергу для мобільних додатків, від компанії Google [18].

Realtime Database – це бази даних у реальному часі [18].

Firebase Authentication – інструмент авторизації користувачів у системі Firebase за допомогою різних методів [28].

Завдяки Firebase Realtime Database є можливість створити такі системи, які будуть оновлювати свої дані у режимі реального часу на усіх пристроях, які будуть під'єднані до цієї системи. Коли один користувач щось змінює в базі даних, інший користувач бачить ці зміни, без оновлення сторінки, у цьому і є головний сенс Realtime Database [18]. При цьому вона містить ті самі можливості сортування даних через запити до неї, як і класична база даних.

Дуже корисною функцією є Firebase Authentication, вона містить в собі декілька найпопулярніших видів ідентифікації користувачів у мережі, наприклад, за номером телефона або за обліковим записом Google та ще багато інших. Це у купі з Realtime Database дозволяє оптимізувати керування даними кожного з користувачів.

Тобто, Realtime Database та Authentication частково замінюють собою сервер, чого буде достатньо для реалізації усього функціоналу додатку.

Отже, після аналізу доступних платформ для роботи системи, було обрано мобільну платформу, адже пристрої які працюють на ній, більшу частину часу, знаходяться біля користувача. Система складається з двох частин: клієнтською та серверною. Для реалізації клієнтської частини було обрано платформу андроїд, а основна мова системи – Java. Середою розробки було обрано IDE Android Studio. Для реалізації серверної частини необхідна база даних та сам сервер для керування БД, але зважаючи на додаткові витрати для реалізації серверу і БД, було обрано альтернативний шлях, їм став сервіс від Google – Firebase. Сервіс бере на себе роль серверу і має в собі базу даних у реальному часі (її особливість у самотійному відправленні даних до клієнта при їх зміні) та Firebase Authentication, яка дозволяє оптимізувати та знизити витрати часу на розробку системи реєстрації користувачів у додатку.

2 ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

Для початку, необхідно визначити, які функції буде виконувати комп'ютерна система, хто буде її потенціальним користувачем та чи є у неї аналоги на ринку і чим вона відрізняється від них.

Суть створюваної комп'ютерної системи у тому, що б надати користувачеві інформацію що до вартості його часу, спираючись на його дохід та кількість відпрацьованих годин на цей дохід. Система повинна полегшити не тільки робочі відносини, але і буденне життя користувача.

Основна ціль системи, на основі розрахованих даних надати йому статистичну інформацію що-до продуктивності робочого дня та допомогти в раціональному плануванні витрати власного часу.

З точки зору роботи, найбільшу користь система несе для «фрілансерів». Люди, які працюють таким чином не раз стикались з проблемою оптимальної оцінки своєї праці. Так, посилаючись на час, який виконавець витратив на виконання поставленої задачі, система розрахує суму до оплати для замовника. Система розрахує вартість години праці виконавця посилаючись на надану стартову інформацію користувачем системи. Після чого з часом використання система відстежує та корегує суму вартості однієї години праці спираючись на доходи користувача. А замовник матиме можливість оцінити продуктивність праці у період виконання роботи та перевірити правдивість наданих даних виконавцем що-до ціни роботи. З цього маємо, що обидві сторони у виграші при використанні цієї системи. Виконавець отримує справедливую оплату, а замовник прозорість ціни за виконання замовлення.

На ринку мобільних додатків не знайдено аналогів даних комп'ютерній системи, але є веб додатки із схожим функціоналом. На відміну від конкурентів система дозволяє відстежувати вартість часу та корегує його у динаміці, знайдені аналоги роблять це один раз при надані користувачем даних. Додатково вона має

рейтингову систему та можливість переглянути статистику свого робочого процесу.

Маючи статистику продуктивності праці та рейтингову, конкурентну таблицю (кожен користувач може переглянути «топ» користувачів системи за власним напрямом роботи або за загальним та побачити свою позицію, що буде спонукати людину до росту, або дозволить оцінити прибутковість напряму посилаючись на реальні та конкретні цифри – реальних користувачів), завдяки чому оптимізувати свою роботу та відстежити аномалії з використанням системи набагато легше. Також система надає можливість переглядати історію зарахування коштів за виконану роботу користувача у зв'язці з історією витраченого часу.

Якщо ж людина працює в будь-якому іншому напрямку і має фіксовану заробітну плату або не має потреби виставляти комусь рахунок за виконану працю, система допоможе оптимізувати день.

Якщо уявити, що розрахована вартість одного часу дорівнює 152 гривні і людині необхідно сьогодні зробити генеральне прибирання квартири. Тож, перед ним постає питання, зробити це своїми зусиллями, витративши, приблизно, 6 годин чи замовити послуги «клінінгової компанії», послуги яких коштують (саме в цьому випадку) 750 гривень. Спираючись на данні розраховані системою стає зрозумілим що вигідніше буде замовити послуги «клінінгової компанії», адже за цей час, 6 годин, що могли бути витрачені на прибирання, заробіток складе 912 гривень, що перевищує вартість послуг «клінінгової компанії» та доводить доцільність прийняття цього рішення, отримавши прибуток у 162 гривні.

2.1 Аналіз вимог до додатка

Додаток повинен містити:

- механізм ідентифікації користувача для зберігання і надання даних;
- механізм відстеження витраченого часу;

- можливість внесення даних для початкових розрахунків;
- можливість перегляду історії роботи;
- можливість перегляду статистики.

Вимоги до дизайну додатку:

- не завантаженість;
- функціональність;
- мінімалістичність;
- інтуїтивність.

Вимоги до функціоналу таймера:

- можливість запуску та зупинки підрахунку часу;
- відображення поточної вартості, що до зазначеного часу на таймері.

Вимоги для історії роботи таймера та оплати праці:

- інформування про дату запису;
- інформування про кількість відпрацьованого часу;
- інформація про розраховану вартість часу.

Вимоги до статистики:

- відображення поточної вартості часу на основі зібраних даних;
- відображення прогресу що до досягнення цілі;
- наявність графіку продуктивності праці.

Вимоги до рейтингу:

- відображення ідентифікатора користувача;
- відображення напрямку праці користувача;
- відображення вартості часу користувача;
- можливість сортування списку рейтингу.

Вимоги до бази даних:

- масштабованість;
- легкість у підключенні та взаємодії;
- можливість здійснювати вибірку даних;
- дешеве обслуговування;
- підтримка кирилиці.

Надійність:

– додаток повинен бути в працездатному стані 24 години на добу та 7 днів на тиждень.

Продуктивність:

– додаток повинен підтримувати одночасне звернення до бази даних великої кількості користувачів.

Безпека:

– додаток повинен гарантувати конфіденційність даних користувача.

2.2 Побудова алгоритму роботи КС

Основний алгоритм системи повинен розраховувати вартість часу людини та надавати цю інформацію користувачеві.

При старті роботи системи, алгоритм очікує надання інформації, для початкових розрахунків, де C – кількість часу витраченого на виконання W роботи у годинах, M – кількість отриманих коштів за виконану W роботу за заданого часу у валюті. Проводиться розрахунок вартості однієї години праці P , $P = C/M$.

Надалі йде перевірка чи не надійшли нові данні за час розрахунку, якщо «ні» то користувач отримує результат, алгоритм очікує нові данні, якщо «так» йде оновлення масиву даних C та M . Виконується розрахунок усіх наявних на цей момент даних: $P = (C_{n-k} + \dots + C_n) / (M_{n-k} + \dots + M_n)$.

Виконується перевірка чи не надійшли нові данні, якщо «ні» користувач отримує результат, якщо «так» алгоритм повертається до кроку з оновленням даних та повторно виконує дії у циклі.

Блок - схема роботи алгоритму зображено на рисунку 2.1.

Таким чином маємо точне значення вартості години праці P у конкретний момент часу, надалі відпрацьовує алгоритм збереження даних.

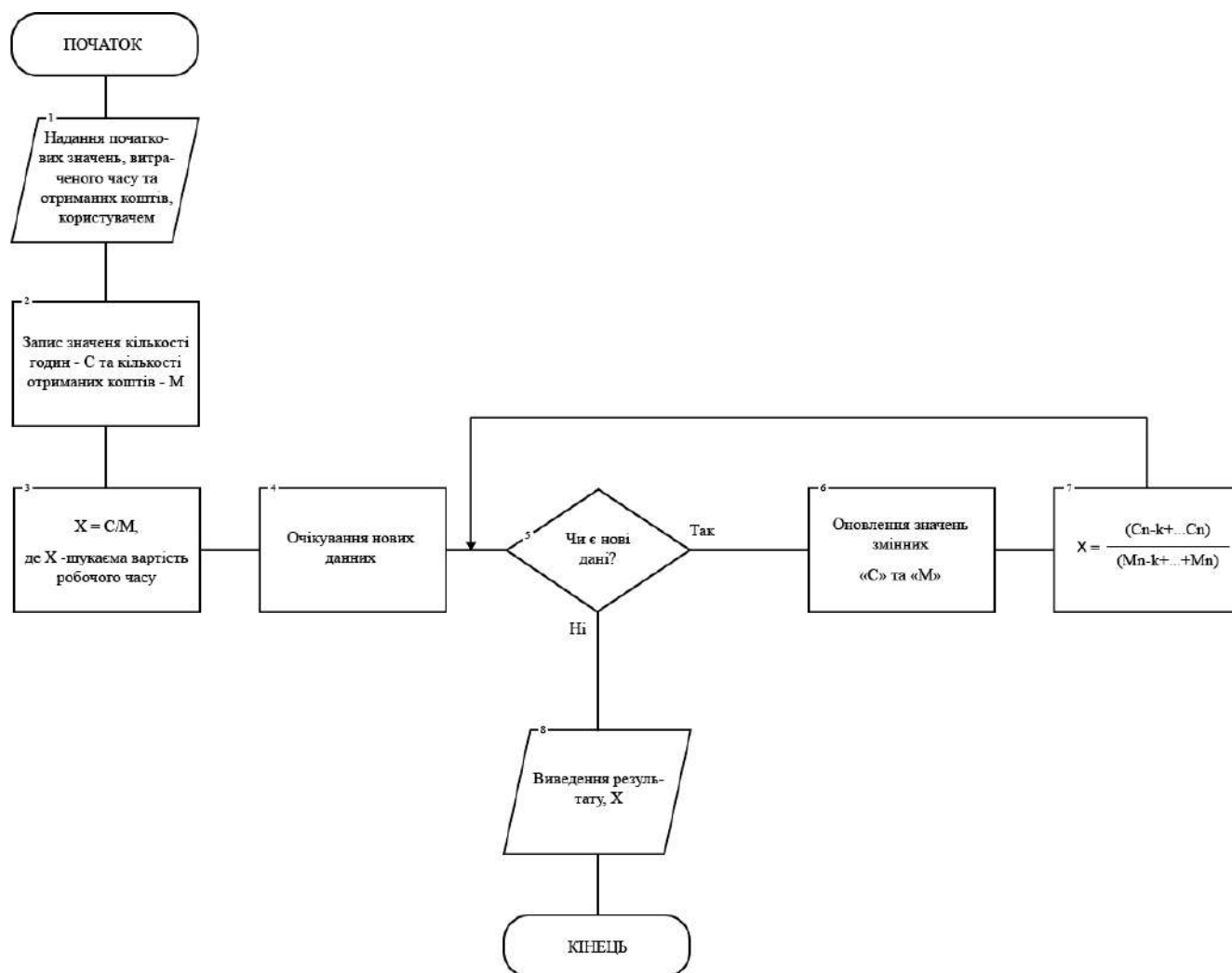


Рисунок 2.1 – Блок - схема алгоритму розрахунку вартості робочого часу

2.3 Побудова діаграми класів

Клас `HomeActivity` відповідає за розміщення та навігацією по сторінкам системи. Також цей клас відповідає за реєстрацію користувачів у системі та передачу стартової інформації про них до бази даних, використовуючи клас `User`.

Клас `User` описує об'єкт, який передається до БД. При формуванні запиту до бази даних, він виступає типом відправлених та отриманих даних з відповідної таблиці (назва таблиці відповідає назві класу). Такі самі функції виконують класи: `HistoryTransaction`, `HistoryTimer` та `Graphix`.

Клас `LeftFragment` представляє собою сторінку історії. Він має в собі ще два класи, відповідає за навігацію по них: `LeftTransactionHFragment`, `LeftTimeHFragment`.

Клас `LeftTransactionHFragment` відповідає за отримання і відправку даних про надходження коштів користувачеві до бази даних, таблиця – `HistoryTransaction` та передає отримані дані до класу- адаптеру `TransactionHAdapter`. Додатково клас відповідає за виведення вікна вводу даних що до зарахувань коштів користувачеві.

Клас `LeftTimeHFragment` відповідає за отримання даних з бази даних, таблиця - `HistoryTimer`, що до історії роботи таймера та передає дані класу - адаптеру `TimerHAdapter`.

Класи - адаптери (`TransactionHAdapter`, `TimerHAdapter`, `AdapterReiting`, `AdapterGraphix`) виконують функцію кур'єра. Вони отримують дані від відповідних класів, структурують і оброблюють їх та відправляють у зворотному напрямку для відображення цих даних на екрані.

Клас `SenterFragment` представляє собою сторінку статистики та відповідає за доступ до класу `TimerFragment`, який розташований у ньому. Робить запити, на додавання та отримання даних, до бази даних, використовуючи клас `Graphix` та `User`. Робить розрахунки з отриманими даним, прослуховує введення нових даних користувачем та при їх наявності повторює розрахунки і оновлює дані у БД. Отримані дані відправляє до класу – адаптеру `AdapterGraphix`.

Клас `TimerFragment` відповідає за відображення та роботу таймера, має у собі функції керування таймером. Прослуховує поточні дані таймера та робить розрахунки і структурує отримані дані, відправляє їх до бази даних використовуючи у якості типу даних клас `HistoryTimer`.

Клас `RightFragment` представляє собою сторінку рейтингу та картки користувача. Клас відповідає за формування запитів відповідно до обраних пунктів фільтру користувачем до БД на отримання даних використовуючи типом даних клас `User`. Опрацьовує отримані дані та відправляє до класу – адаптеру `AdapterReiting`.

Для більш чіткого розуміння взаємодії класів, необхідно побудувати UML діаграму класів системи, дивитись рис. 2.2.

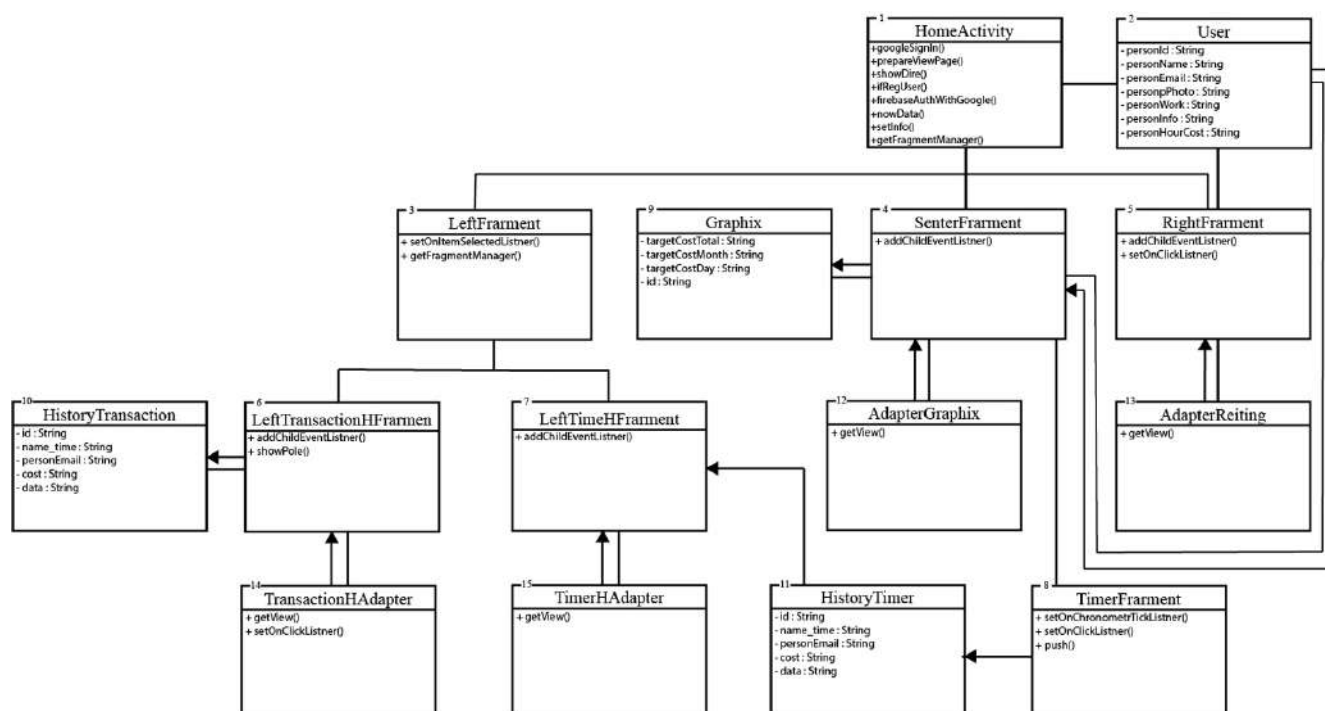


Рисунок 2.2 - UML діаграма класів

2.4 Макети сторінок Android додатку

Для зручного користування додатком необхідно створити макет, у якому буде проаналізована вся ергономіка та передбачене місце для усіх функцій. На базі розробленого макету художник розробить дизайн для кожної сторінки додатку.

Так, як авторизація у додатку буде проходити в автоматичному режимі, необхідно розробити макет трьох сторінок додатку:

- головна сторінка , з відображенням усієї статистичної інформації;
- сторінка відображення історії роботи таймеру та історії надходження оплати за роботу;
- сторінка з особистою інформацією користувача та вікном рейтингу працівників у такому ж напрямку.

Навігація по сторінках додатку за горизонтальним гортанні вліво (до сторінки з історією) чи вправо (до сторінки з рейтингом) від головної (центральної) сторінки.

На центральній сторінці, макет якої зображено на рисунку 2.3, додаток має:

- ключовий елемент відображення поточної вартості часу у вигляді Progress Bar;
- елемент відображення навігації по сторінках додатку;
- блок з графіком та Progress Bar;
- кнопка виклику таймеру.

Ця сторінка виступає лицем додатку, тож має бути приємною на вигляд, інформативною та не перевантаженою. Макет, а після і дизайн цієї сторінки буде базовим для інших сторінок. Так, як додаток має елементи з якими необхідно часто взаємодіяти (перша група) та ті, які будуть нести тільки інформативний характер або дуже рідку взаємодію (друга група елементів), сторінка додатку розділена навігаційним елементом на дві частини, де знизу знаходяться елементи з першої групи, а зверху з другої групи.

На лівій сторінці, макет якої зображено на рисунку 2.4, додаток має:

- анімаційне зображення за для надання інтерактивності додатку та підбадьорювання користувача;
- навігація, як і у центральній сторінці;
- блок з історією роботи таймера та надходження коштів;
- навігаційна панель для переходів між історіями у блоку історії.

На сторінці збережено основний стиль з розділом пунктом меню сторінки на дві області.

На правій сторінці, додаток має:

- блок з інформацією користувача;
- блок рейтингу;
- блок з фільтром для налаштування вибірки у блоку відображення рейтингу;
- навігація (аналогічного іншим сторінкам).

Сторінка зберігає основний стиль, але елементи в ній відносяться переважно до першої групи. Макет сторінки зображено на рисунку 2.5.



Рисунок 2.3 – Макет головної сторінки додатку



Рисунок 2.4 – Макет лівої сторінки додатку



Рисунок 2.5 – Макет правої сторінки додатку

2.5 Структура бази даних

База даних повинна мати чотири таблиці, дивитись рис. 2.6:

- Таблиця User з даними користувача;
- Таблиця HistoryTimer з історією роботи таймера;
- Таблиця HistoryTransaction з історією отримання коштів;
- Таблиця Graphix з даними статистики;

Таблиця User має такі поля:

- personEmail – електронна адреса користувача;
- personHourCost – розрахована вартість часу користувача;
- personId – унікальний ідентифікатор акаунту Google користувача;
- personInfo – додаткова інформація про користувача;
- personName – ім'я користувача;

- personPhoto – фото профіля користувача у Google;
- personWork – назва спеціальності користувача.

Таблиця HistoryTimer має такі поля:

- cost – розрахована вартість нарахованого часу праці;
- data – дата додавання запису;
- id – унікальний ідентифікатор користувача;
- name_time – відпрацьований час.

Таблиця HistoryTransaction має такі ж поля, як і таблиця HistoryTimer. Тільки поле «name_time» містить назву платежу замість відпрацьованого часу, а поле «cost» суму отриманих коштів.

Таблиця Graphix має такі поля:

- targetCostTotal – загальна ціль користувача що до вартість часу;
- targetCostMonth – цільова вартість часу до кінця місяця;
- targetCostDay – цільова вартість часу до кінця дня;
- id – унікальний ідентифікатор користувача.

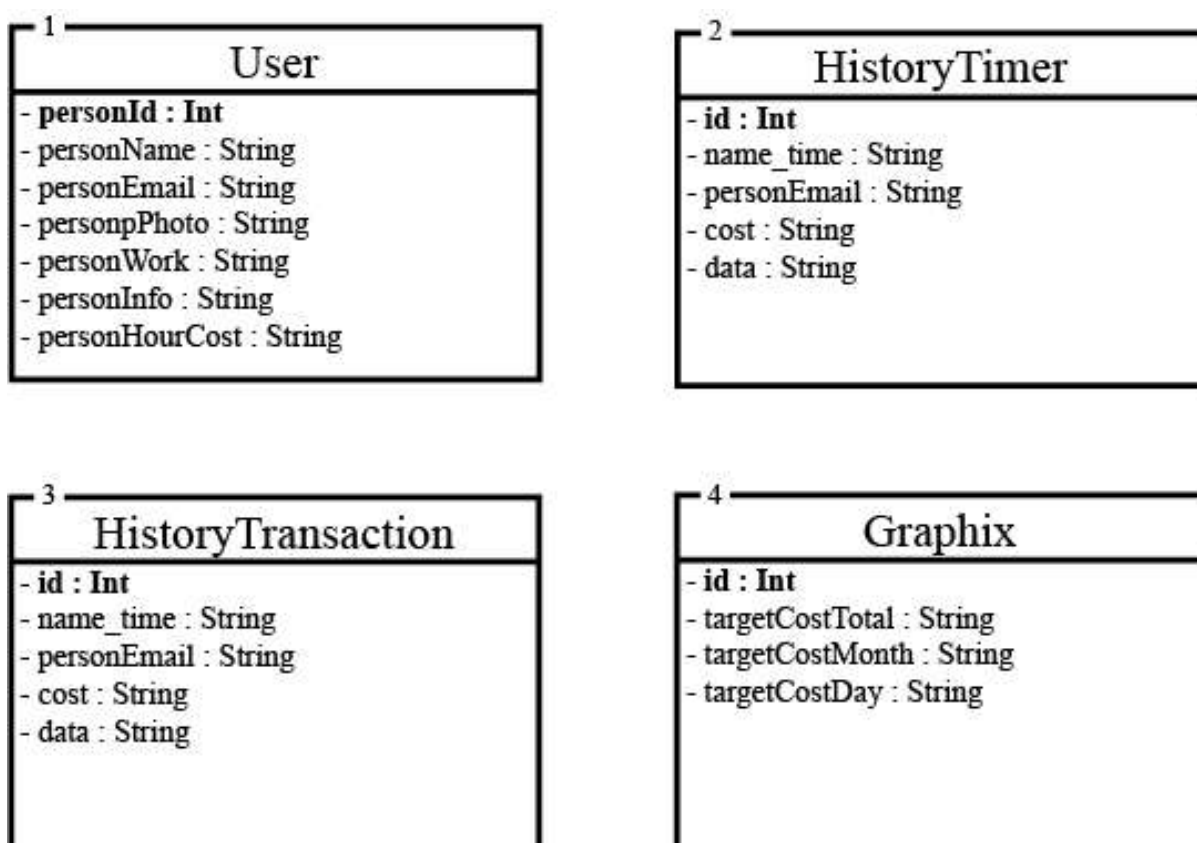


Рисунок 2.6 – Графічне представлення таблиць

Тож, сходячи з усього вище сказаного маємо чітке бачення які функції буде виконувати система, хто буде її користувачем та у чому її особливість у порівнянні з аналогами. Сформовано перелік вимог для додатка. Розроблено основний алгоритм, розрахунку вартості часу, системи. На основі описаних вимог було створено макети сторінок додатку, на основі яких повинен створюватись дизайн додатку. Після закінчення проектування клієнтської частини було розроблено структуру бази даних системи.

3 РОЗРОБКА АПАРАТНОЇ СПЕЦИФІКАЦІЇ ТА ІНФРАСТРУКТУРИ

Для коректної роботи будь-якої програми необхідно мати відповідні до її вимог апаратні специфікації. Необхідно проаналізувати де і як буде використовуватись програмне забезпечення, а головне, що воно буде робити, у глобальному плані.

Є три глобальні напрямки роботи системи:

- робота з графікою;
- робота з алгоритмами та обчисленнями;
- робота з великими об'ємами даних.

Подана система відноситься до другого пункту переліку. Цей пункт може буди зрівняний з перший, який є най більш ресурсна затратний, але у даному випадку алгоритми, які будуть виконуватись не є важкими, тож потреби до апаратного забезпечення будуть незначними.

Головними критеріями для оцінки сумісності системи з програмним забезпеченням є:

- підтримка програмою операційної системи апаратного забезпечення;
- підтримка архітектури процесора;
- наявність мінімально необхідної розрахункової потужності процесора;
- наявність мінімально необхідного розміру оперативної пам'яті для коректного і комфортного розгортання усіх процесів програмного забезпечення;

– наявність необхідної кількості вільного місця у накопичувачі.

Для роботи додатку необхідно мати пристрій, а саме телефон. Під управлінням мобільної операційної системи. Наразі існує лише дві поширені ОС: Android та IOS. На основі порівняння цих двох систем у розділах вище, робимо висновок, що потрібна ОС саме Android. Адже вона має більший охоплення аудиторії та підходящу модель монетизації.

Монетизація за допомоги показу рекламних оголошень на сторінках додатку, реалізована завдяки сервісу та його SDK - Google AdMob.

SDK - це комплект для розробки програмного забезпечення, який допомагає розробникам створювати додатки для конкретних платформ. Це можуть бути комп'ютери, ігрові приставки або мобільні пристрої [25].

Google AdMob – одна з найбільших рекламних мереж, яка дозволяє отримувати дохід від показу оголошень з додатків у які вона інтегрована [26].

Додатково у додаток необхідно інтегрувати SDK OneSignal, воно дозволяє підключити пуш повідомлення за декілька строчок коду, а все управління ними йде через сайт, що дозволяє уникнути написання додаткового інтерфейсу для керування повідомленнями. Додатковим плюсом є велика кількість інструментів на сайті OneSignal для персоналізації оголошень та надання їм додаткового функціоналу, такого, як відкриття просилання по кліку на повідомлення.

OneSignal – це сервіс для доставки Push-повідомлень і автоматизації цього процесу [27].

Платформа Android має версії, мінімальна версія Android для роботи додатку – 5.0 Lollipop. Обрана ця версія тому, що в ній з'явився новий користувацький інтерфейс та підтримка багатьох можливостей по кастомізації та дизайну, на базі якого створені останні версії [21].

Android 5.0 Lollipop – це версія мобільної операційної системи Android, випущена 25 червня 2014 року [21].

Архітектура процесора – мобільна, яку мають усі телефони під керуванням вище описаної версії Android та більш нові [22].

Мобільною архітектурою на даний момент є APM (Advanced RISC Machine), розробкою якої займається ARM Limited. Інші компанії ліцензують технологію ARM і на основі цієї ліцензії випускають свої процесори, наприклад, Qualcomm, Nvidia з їх чіпами Tegra, всім відомі Mediatek, чіпи від Apple і, звичайно ж, Samsung з Exynos [22].

Мінімальна кількість ядер процесора для стабільної роботи додатку – два та більше.

Мінімальний об'єм оперативної пам'яті становить 1 гігабайт, цього достатньо адже у додатку не має великої кількості графіки або тяжких розрахункових задач.

Зібраний проект має розмір у 50 мегабайт, додатково необхідно мати хоча б 10 мегабайт вільного простору для зберігання інформації користувача та інших дрібних даних.

Так, як наявна серверна частина системи, необхідне стабільне підключення до мережі під час роботи з додатком. Обсяг обміну даних не великий, тож мінімально необхідна швидкість для комфортної роботи є швидкість, яку надає мережа бездротового покриття «3G» або «H+».

Зважаючи на все вище описане, можна зробити висновок, що жорсткою вимогою є лише стабільне підключення до мережі необхідного рівня та версія Android, так як інші вимоги будуть автоматично виконані, бо ця версія Android йде на апаратне забезпечення з більшим запасом потужності.

При розробці необхідно використовувати бібліотеки сторонніх розробників або фірм. Бібліотеки розширюють стандартний функціонал, або ж надають вже готові методи для швидкої реалізації поставленої задачі. Вони є головним інструментом програміста. Навіть, якщо бібліотека не реалізовує усі потреби на сто відсотків, її можна відредагувати додавши свій функціонал на основі існуючого, або ж додати косметичні правки.

Для розробки деяких елементів інтерфейсу користувача, необхідно використати такі бібліотеки:

- `pl.pawelkleczkowski.customgauge.CustomGauge` – бібліотека дозволяє створити `Progres Bar` формату полу кільце. Стандартний набір інструментів

дозволяє створити їх лише у формі кільця або лінії. Завдяки бібліотеці є можливість швидко інтегрувати прогрес бар, а не прописувати стилі, який нагадує спідометри нових авто, що підкреслює футуристичність та новизну додатку;

– `com.ismaeldivita.chipnavigation:chip-navigation-bar` – бібліотека дозволяє створити анімований Bottom Navigation View. Він включає в себе анімацію зміни написів на форм об'єктів, які інформують про обрану у ньому позицію;

– `pl.droidsonroids.gif:android-gif-drawable` – за допомоги цієї бібліотеки є можливість додавати анімаційні зображення (картинки формату PNG). Стандартні можливості розмітки можуть зробити лише вставку картинки чи відео;

– `de.hdodenhof:circleimageview:3.1.0` – бібліотека робить звичайні картинки круглої форми. Ця бібліотека, за допомоги прописаного у ній стилю з прозорим тілом, окрім кола в центрі, робить уяву, що зображення кругле, насправді ж вона робить маску, ховаючи непотрібну площу картинки.

Для реалізація функціоналу додатку також необхідна велика кількість бібліотек, від тих що запускають потоки до тих що роблять анімацію. Основну бібліотеку, яку необхідно використати для реалізацій головного алгоритму, є Chronometer. У цій бібліотеці вже реалізовані всі функції для роботи таймера, необхідно лише викликати їх.

Інфраструктура даної комп'ютерної системі включає в себе клієнтський інтерфейс, базу даних, стабільне під'єднання до мережі та магазин Google Play, у якому вона буде розміщена. Від їх складеної роботи залежить працездатність системи. У всьому іншому система повністю автономна і не потребує додаткового втручання людини або ресурсів.

Інфраструктура – це комплекс взаємопов'язаних обслуговуючих структур або об'єктів, що становлять і забезпечують основу функціонування системи [23].

Google Play – це магазин додатків, а також ігор, книг, музики і фільмів від компанії Google, що дозволяє стороннім компаніям або самостійним розробникам пропонувати власникам пристроїв з операційною системою Android встановлювати і купувати різні додатки [24].

Тож, маємо перелік факторів сумісності системи з пристроями та перелік їх додаткових інструментів, які необхідно використати, такі як бібліотеки та SDK, для досягнення максимального кола користувачів і максимальної потужності та оптимізації за допомоги розбору елементів інфраструктури системи з виявленням ризиків відмови системи. На базі цього можна починати розробку системи.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Створення проекту у IDE

Після аналізу функціоналу додатка, створення основного алгоритму та макету сторінок, художник надає розробленій дизайн сторінок додатку.

Після проведеного аналізу та вибору IDE у вигляді Android Studio починається розробка.

При створенні проекту у Android Studio обирається шаблон головної сторінки, у нашому випадку Empty Activity, задається назва додатку, ім'я його пакету, вказується місце зберігання проекту, обирається мова розробки - Java та мінімальний рівень API (набір функцій, модулів, бібліотек, класів) – 21, це означає що додаток зможе працювати на пристроях під керуванням Android 5.0 та вище, рис. 4.1. Android Studio конфігурує проект.

Після завершення конфігурації бачимо структуру проекту, стартову сторінку з написом «Hallow World». IDE створило працездатний додаток, при запуску якого, на емуляторі чи фізичному пристрої, відкритись вікно з написом Hallow World».

Переходимо до зміни проекту відносно описаних вище потреб.

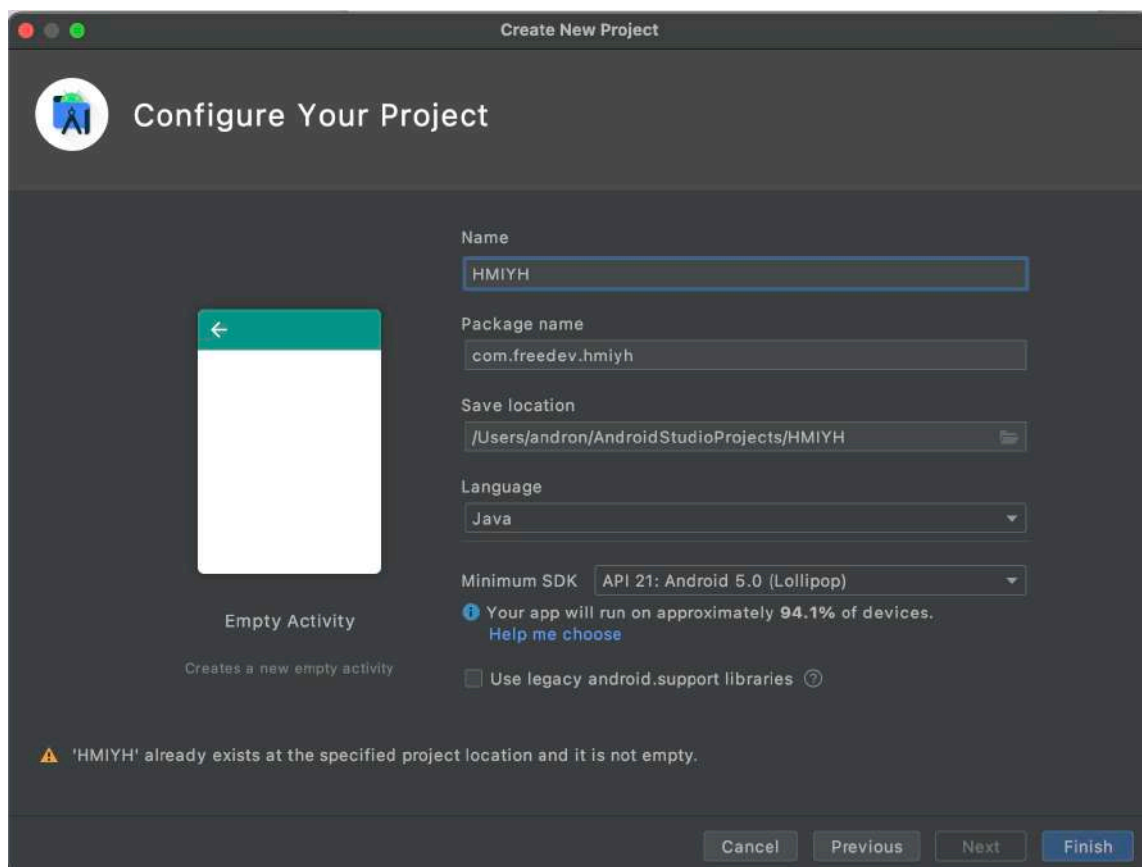


Рисунок 4.1 – Вікно налаштувань проекту

4.2 Розробка клієнтської частини додатку

У першу чергу створюється структура додатку, яка складається з трьох сторінок. Для цього необхідно створити три Fragment та розташувати їх на вже створеній сторінці (активності) HomeActivity.

Fragment – це такий самий компонент, як і Button або TextView, він схожий на активність, але це не зміна активність, він не існує сам по собі, а тільки в складі активностей. У Fragment теж є свій життєвий цикл, але при зміні одного фрагменту на інший вони не перезавантажуються, на відміну від активності. Фрагмент існує доки існує активність або доки його не знищити примусово. Іншими словами, фрагменти допомагають розширити екран пристрою, зробити зі смартфона – планшет (те що помістилось на одній сторінці екрану планшету можна розмістити

на декількох фрагментах на смартфоні листаючи їх або переключаючись). Тож їм набагато краще користуватись при створенні сторінок, між якими потрібно часто переключатись [19].

У дереві проекту створюємо новий пакет (Package) «fragments» у якому створюємо три фрагменти:

- CenterFragment – фрагмент для центральної сторінки додатку;
- LeftFragment – фрагмент для правої сторінки додатку;
- RightFragment – фрагмент для лівої сторінки додатку.

На базі цих трьох фрагментів буде створено додаток. Їх необхідно помістити на активність та зробити можливість переходу між ними, формат переходу – горизонтальне ковзання по «свайпам» вліво або вправо. Для цього у XML файлі класу, у якому лежить вся розмітка інтерфейсу, це дозволяє відокремити код класу від коду розмітки інтерфейсу, необхідно додати контейнер - ViewPager у якому і буде зміна фрагментів, його код наведено у лістингу 4.1, керувати якою буде TabLayout, він представляє собою перемикач, який підтримує заміну фрагментів по «свайпах», його код наведено у лістингу 4.2. Також надана стилізація цьому елементу, на основі розробленого дизайну, у вигляді трьох кругів, які заповнюються при відображенні відповідного фрагмента на екрані.

Лістинг 4.1 – Розмітка контейнеру фрагментів ViewPager

```
<androidx.viewpager.widget.ViewPager
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/view_pager_head"
    android:layout_alignParentStart="true" />
```

Лістинг 4.2 – Розмітка компоненту TabLayout

```
<com.google.android.material.tabs.TabLayout
    android:layout_width="100dp"
    android:layout_height="13dp"
    android:id="@+id/tab_layout"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="270dp"
    app:tabTextColor="@color/black"
    app:tabIndicatorHeight="13dp"
    android:backgroundTint="#00ffffff"
```

```
app:tabMaxWidth="13dp"
app:tabIndicator="@drawable/ic_dot"
app:tabIndicatorColor="@color/white"/>
```

Після створення розмітки – візуальної частини, необхідно запрограмувати їх дії кодом, а саме реалізувати заміну фрагменту по «свайпу», для цього треба написати метод `prepareViewPage()` та клас адаптер – `HomeAdapter` (у структуру проекту додано пакет `adapters`, у якому будуть міститись всі адаптери), код яких наведено у лістингу 4.3 та лістингу 4.4 відповідно, та анімувати іконки пунктів меню. Для реалізації анімації необхідно створити додатковий файл XML, його код наведено у лістингу 4.5, у якому задані параметри анімації та написан код до керування нею.

Лістинг 4.3 – Код методу `prepareViewPage()`

```
private void prepareViewPage(ViewPager viewPager_head) {
    HomeAdapter adapter = new
    HomeAdapter(getSupportFragmentManager());
    LeftFragment leftFragment = new LeftFragment();
    RightFragment rightFragment = new RightFragment();
    CenterFragment centerFragment = new CenterFragment();
    adapter.addFragment(leftFragment);
    adapter.addFragment(centerFragment);
    adapter.addFragment(rightFragment);
    viewPager_head.setAdapter(adapter);
}
```

Лістинг 4.4 – Розмітка компоненту `TabLayout`

```
private class HomeAdapter extends FragmentPagerAdapter {
    List<Fragment> fragmentList = new ArrayList<>();
    public void addFragment(Fragment fragment){
        fragmentList.add(fragment);
    }
    public HomeAdapter(FragmentManager
supportFragmentManager) {
        super(supportFragmentManager);
    }
    @NonNull
    @Override
    public Fragment getItem(int position) {
        return fragmentList.get(position);
    }
    @Override
    public int getCount() {
```

```

        return fragmentList.size();
    }
}

```

Лістинг 4.5 – Код файлу анімації

```

//анімація заповнення
<alpha
    android:duration="1000"
    android:fromAlpha="0.0"
    android:toAlpha="1.0" >
</alpha>

//зворотна анімація
<alpha
    android:duration="1000"
    android:fromAlpha="1.0"
    android:toAlpha="0.0" >
</alpha>

```

4.3 Розробка розмітки фрагментів на основі дизайну

Центральний фрагмент є головною сторінкою у додатку, з нього і починається розробка.

Як було сказано вище кожна сторінка, умовно, поділена на верхню і нижню частину.

У верхній частині знаходиться стилізований ProgressBar за допомогою бібліотеки у вільному доступі – «pl.pawelkleczkowski.customgauge:CustomGauge», який буде відображати прогрес користувача у поставленій цілі що до вартості його години, код наведено у лістингу 4.6.

Лістинг 4.6 – Стилiзований ProgressBar

```

<pl.pawelkleczkowski.customgauge.CustomGauge
    android:id="@+id/headProgressBAQR"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/hour"
    android:layout_centerHorizontal="true"
    app:gaugeEndValue="800"

```

```

app:gaugePointEndColor="#116498"
app:gaugePointSize="77"
app:gaugePointStartColor="#ffffff"
app:gaugeStartAngle="135"
app:gaugeStartValue="200"
app:gaugeStrokeCap="ROUND"
app:gaugeStrokeColor="#bdbdbd"
app:gaugeStrokeWidth="20dp"
app:gaugeSweepAngle="270" />

```

На нижній частини розташований елемент `ListView`, з його допомогою реалізуються списки з ковзанням, код наведено у лістингу 4.7. У ньому знаходяться компоненти стилізованого графіку (`ir.farshid_roohi.linegraph.LineChart`), лістинг 4.8, за аналогію вище описаного `ProgressBar` та стилізований `ProgressBar`, але не за допомогою бібліотеки, а за допомогою написання стиля, його код наведено у лістингу 4.9.

Лістинг 4.7 – Розмітка елементу `ListView`

```

<ListView
    android:id="@+id/list_graphix"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp" />

```

Компонент `ListView` використано за для перспективи додавання на центральну сторінку більше графіків та `ProgressBar`, якщо їх буде більше за максимальну кількість що видно на екрані то треба буде лише прокрутити елемент знизу до гори.

Лістинг 4.8 – Розмітка елементу стилізованого графіку

```

<ListView
    android:id="@+id/list_graphix"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp" />

```

Лістинг 4.9 – Файл розмітки стилізованого ProgressBar

```

<?xml version="1.0" encoding="utf-8"?>
<layer-list
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape android:shape="rectangle">
            <corners android:radius="10dip" />
            <gradient
                android:angle="45"
                android:endColor="#50E7FFFF"
                android:gradientRadius="10dip"
                android:startColor="#50E7FFFF" />
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <scale
            android:drawable="@drawable/pb_custom_progress"
            android:scaleWidth="100%" />
    </item>
</layer-list>

```

На центральній сторінці розташований блок з таймером, який викликається за допомогою натискання на елемент `ImageView`, який знаходиться у самому низу сторінки. Сам блок таймеру являє собою фрагмент, який за допомогою анімації виїжджає на одну третю висоти центральної сторінки та ховається при натисканні на цьому компонента `ImageView`, аналогічного до того, що знаходиться на центральній сторінці. Блок таймеру складається з `Chronometer` (об'єкт що виступає таймером), об'єкта `TextView` для відображення розрахованої вартості часу, згідно часу відрахованого таймером та кнопку старту та зупинки таймеру, яка представлена елементом `TextView` стилізованого під кнопку за допомогою стилів.

На лівій сторінці у верхній частині знаходиться анімаційне зображення, його доцільно додати, як зображення формату PNG ніж прописувати анімацію кодом, за для реалізації цього задумо служить елемент `GifImageView` реалізований за допомогою сторонньої, відкритої бібліотеки, код наведено у лістингу 4.10.

Лістинг 4.10 – Елемент `GifImageView`

```

<pl.droidsonroids.gif.GifImageView
    android:id="@+id/gifImageView"

```

```

android:layout_height="260dp"
android:layout_width="match_parent"
android:src="@drawable/gif1"
android:layout_centerHorizontal="true"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

```

На нижній частині сторінки розташований елемент контейнер `FrameLayout`. Так, як необхідно відобразити на сторінці дві історії (історію таймера та історію зарахування коштів), треба зробити за принципом головних сторінок- використати декілька фрагментів. Треба створити `LeftTimerHFragment` у якому знаходиться історія таймеру та `LeftTransactionHFragment` у якому знаходиться історія зарахувань коштів. Код `FrameLayout` наведено у лістингу 4.11.

Лістинг 4.11 – Елемент `FrameLayout`

```

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/fragment_container"
    android:layout_above="@+id/chip_bar"
    android:layout_marginTop="290dp"/>

```

Під `FrameLayout` розташовано стилізований елемент навігації по створених фрагментах, також за допомогою сторонньої бібліотеки, `ChipNavigationBar`, код наведено у лістингу 4.12.

Лістинг 4.12 – Елемент `ChipNavigationBar`

```

<com.ismaeldivita.chipnavigation.ChipNavigationBar
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:id="@+id/chip_bar"
    android:layout_alignParentBottom="true"
    android:layout_gravity="bottom"
    app:cnb_unselectedColor="@color/white"
    android:background="#191919"
    app:cnb_menuResource="@menu/bottom_menu" />

```

У верхній частині правої сторінки знаходиться область картка користувача, яка містить в собі елементи `TextView` для зображення тексту та стилізований

елемент `CircleImageView` за допомогою сторонньої бібліотеки для зображення аватара користувача, особливість цього елемента у тому що він надає зображенню форму круга при завантаженні прямокутного зображення, код елемента наведено у лістингу 4.13.

Аватар – це фото або інше графічне зображення, яке використовується в обліковому записі користувача для персоналізації [20].

Лістинг 4.13 – Елемент `CircleImageView`

```
<de.hdodenhof.circleimageview.CircleImageView
    android:id="@+id/avatar_user"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:src="@drawable/avatar"
    app:civ_border_color="@color/blue"
    app:civ_border_width="2dp" />
```

Під областю користувача знаходиться елемент `Spinner`, який виконує функцію кнопки з випадаючим списком, на його основі створено фільтри до вікна з рейтингом. Код елемента наведений у лістингу 4.14.

Лістинг 4.14 – Елемент `Spinner`

```
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/spinner"
    android:background="@drawable/style_spinner"
    android:isScrollContainer="true"
/>
```

На нижній частині знаходиться елемент `ListView`, аналогічний до елемента, який використано на центральній сторінці. У ньому знаходяться поля, які відображають список з рейтингом. Для його реалізації треба використати `CircleImageView` для зображення аватара та `TextView` для надання інформації.

Після заповнення сторінок тестовою інформацією, маємо фінальний вигляд сторінок, дивитись рис. 4.2-4.6.



Рисунок 4.2 – Дизайн головної сторінки

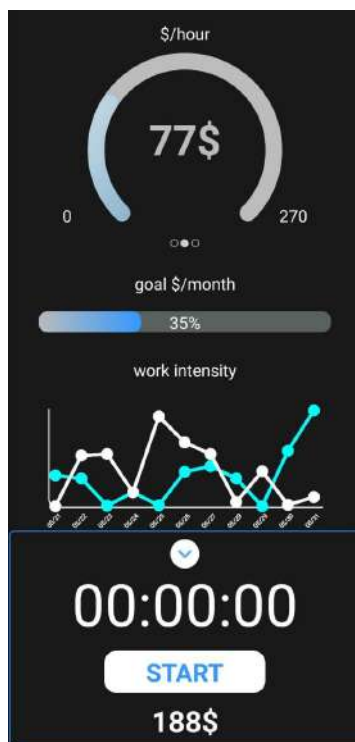


Рисунок 4.3 – Дизайн головної сторінки та таймеру

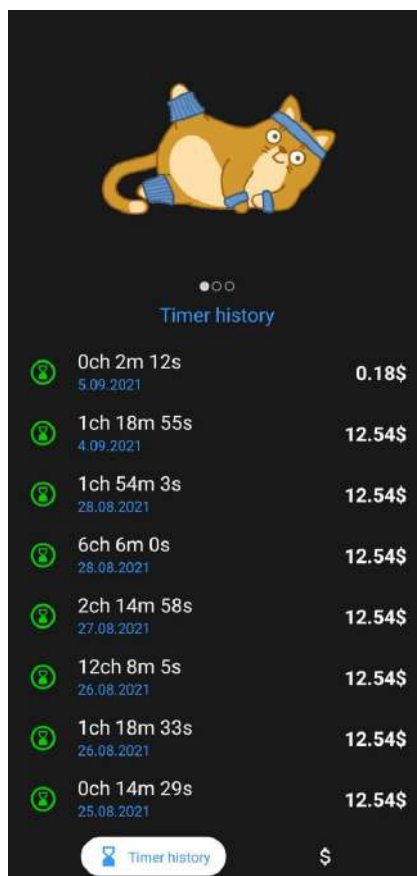


Рисунок 4.4 – Дизайн лівої сторінки історії таймеру

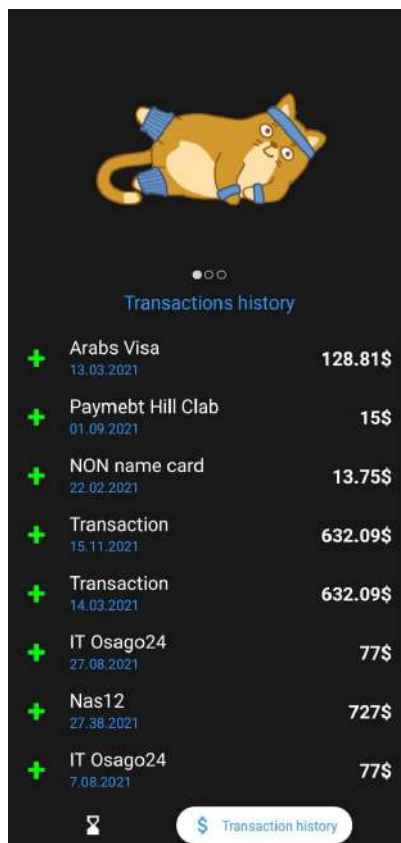


Рисунок 4.5 – Дизайн лівої сторінки історії надходження коштів

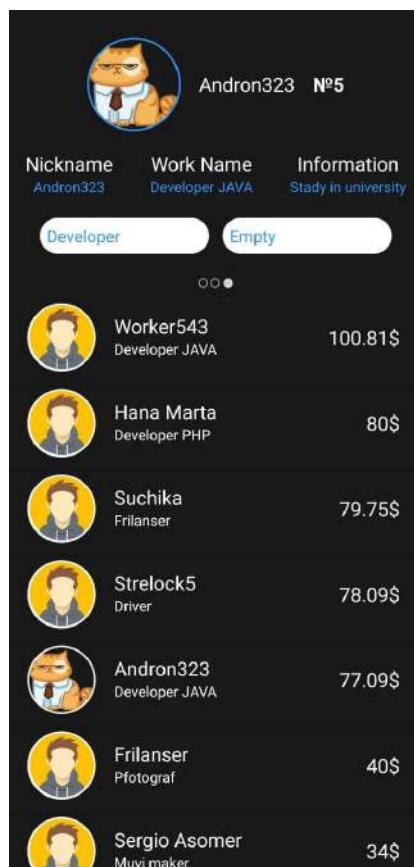


Рисунок 4.6 – Дизайн правої сторінки з рейтингом

4.4 Розробка клієнтського функціоналу додатку

Перш за все, їде ідентифікація користувача серед багатьох інших для коректного відображення інформації.

Для цього необхідно використати сервіс Firebase Authentication. Із списку наданих методів авторизації обрано найпопулярніший та інформативно вмісений – Google Sing-in.

Для його підключення необхідно на головній сторінці Firebase обрати пункт Authentication. У відкритому вікні налаштувань вибрати метод авторизації Google та надати назву додатку, його пакет, обрати данні які необхідно отримати від аканту користувача Google і вказати SHA-1 ключ.

SHA-1 – це стандарт для реалізації "безпечного алгоритму хешування" - односпрямованої криптографічної функції, яка може використовуватися для "підписів" послідовності байтів [29].

У IDE обрати із меню Tools пункт Firebase, обрати необхідний розділ та надати згоду на автоматичне встановлення з'єднання та додавання бібліотек до проекту.

Після успішного поєднання сервісу Firebase та проекту розроблюється логіка роботи реєстрації користувачів на основі наданих функцій та методів Firebase Authentication.

Надалі виконується ініціалізація Firebase Authentication та передаються параметри для ідентифікації проекту.

Перш за все, необхідно перевірити чи зареєстровано поточного користувача у системі, якщо ні – зареєструвати. Перевірка виконується завдяки методу `ifRegUser()`, код методу наведено у лістингу 4.15. Якщо користувача зареєстровано, метод виводить повідомлення про його наявність в системі, якщо не зареєстровано то виводиться діалогове вікно з авторизацією, дивитись рис. 4.7.

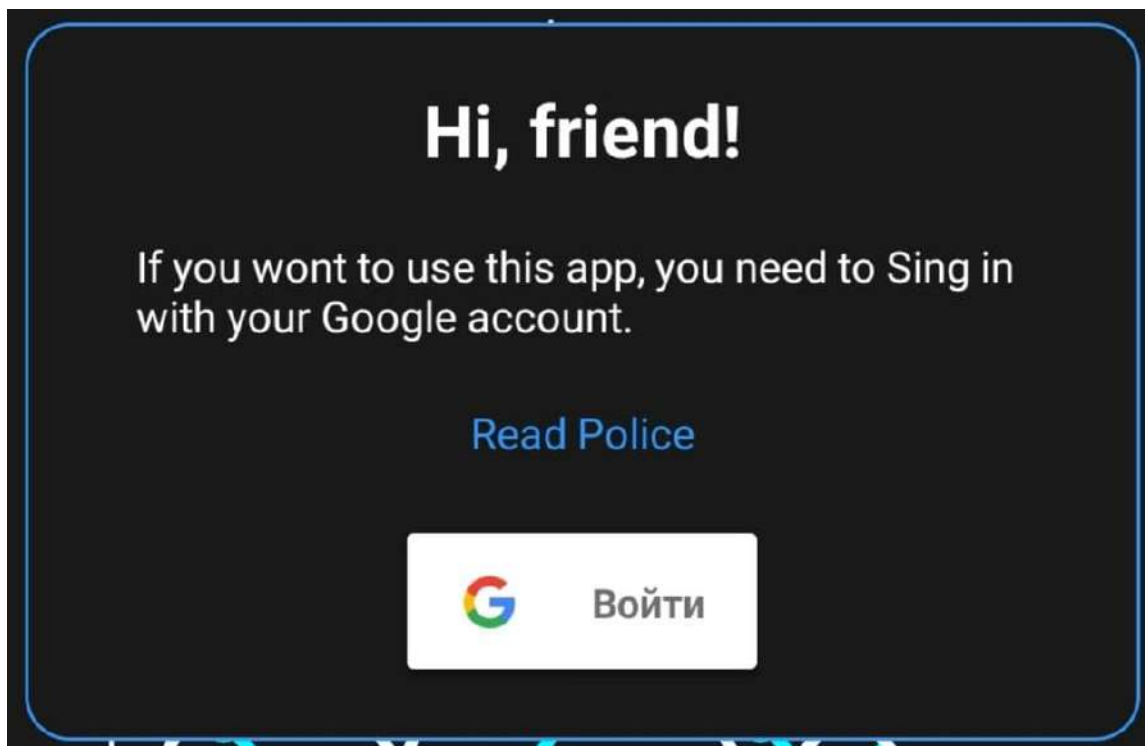


Рисунок 4.7 – Діалогове вікно авторизації

Лістинг 4.15 – Метод перевірки авторизації користувача

```

FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
    FirebaseUser currentUser =
firebaseAuth.getCurrentUser();
    if (currentUser != null) {
        Toast toast =
Toast.makeText(getApplicationContext(),
        "NO NULL", Toast.LENGTH_SHORT);
        toast.show();
    } else {
        regWindow.setVisibility(View.VISIBLE);
        headRelLayHome.setClickable(false);
    }

```

Вікно авторизації містить привітальний напис, посилання на політику конфіденційності та кнопку реєстрації Google, її код наведено у лістингу 4.16. При її натисканні з'являється наступне вікно, рис. 4.8, з переліком облікових записів Google доступних на поточному пристрої для реєстрації у додатку.

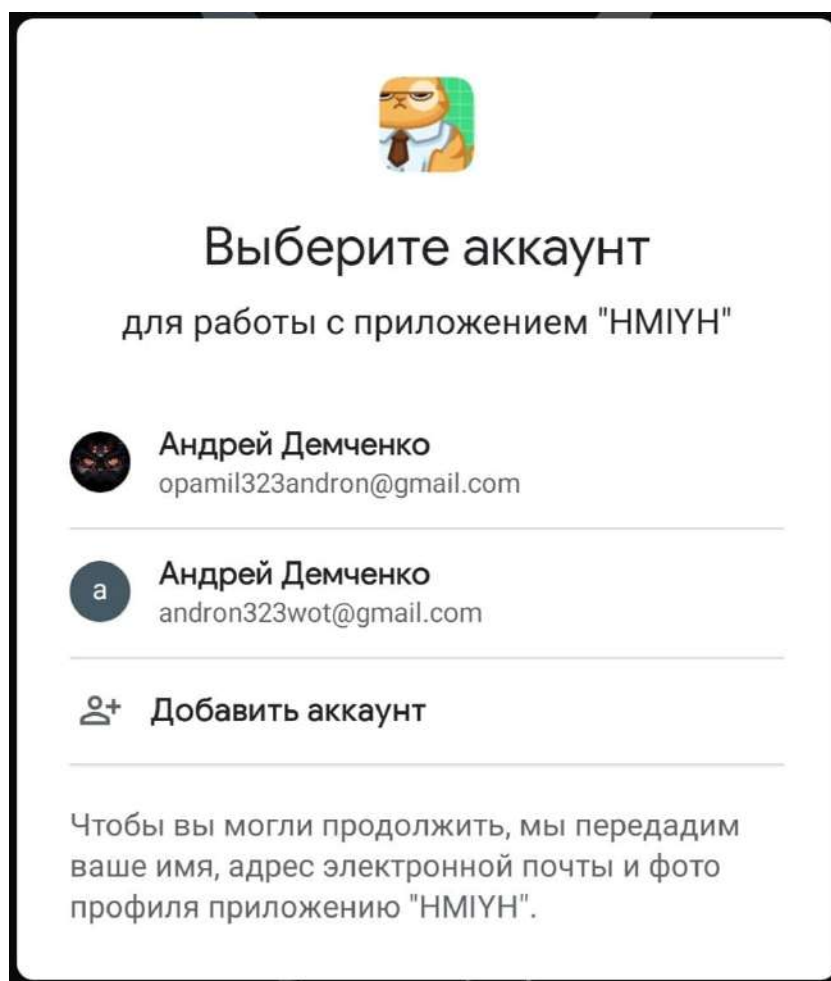


Рисунок 4.8 – Діалогове вікно переліку облікових записів Google

Лістинг 4.16 – Кнопка Google

```
<com.google.android.gms.common.SignInButton
    android:id="@+id/out_button"
    android:layout_width="wrap_content"
    android:layout_toRightOf="@+id/sign_in_button"
    android:layout_marginLeft="30dp"
    android:visibility="gone"
    android:layout_height="wrap_content" />
```

Після вибору облікового запису запускається метод `firebaseAuthWithGoogle()`, код наведено у лістингу 4.17, у якому виконується додавання користувача до бази даних Firebase для поточного додатка та робиться запит на отримання таких даних:

- ім'я користувача поточного облікового запису Google;
- email користувача;
- унікальний id облікового запису користувача Google;
- фото облікового запису Google користувача.

Лістинг 4.17 – Метод `firebaseAuthWithGoogle()`

```
private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {
    AuthCredential authCredential =
    GoogleAuthProvider.getCredential(acct.getIdToken(), null);

    mAuth.signInWithCredential(authCredential).addOnCompleteListener(
        this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult>
            task) {
                if (task.isSuccessful()) {
                    Toast toast =
                    Toast.makeText(getApplicationContext(),
                                "isSuccessful",
                                Toast.LENGTH_SHORT);
                    toast.show();
                    FirebaseUser currentUser =
                    mAuth.getCurrentUser();
                    updateUI(currentUser);
                } else {
                    Toast toast =
                    Toast.makeText(getApplicationContext(),
                                "Failed", Toast.LENGTH_SHORT);
                    toast.show();
                }
            }
        })
}
```

```
    });  
}
```

Одразу після успішного виконання методу `firebaseAuthWithGoogle()` запускається метод `updateUI()`, який і отримує данні що до облікового запису Google, зберігає унікальний id користувача за допомоги методу `SharedPreferences`, код якого наведено у лістингу 4.18, та виводить останнє діалогове вікно, у якому просить користувача надати додаткову інформацію для початкових розрахунків:

- назву виду діяльності;
- інформацію про хобі;
- інформацію про кількість робочих днів на місяць;
- інформацію про кількість робочих годин на день;
- інформація про місячну заробітну плату або приблизний дохід.

Лістинг 4.18 – Метод `SharedPreferences`

```
mSettings = getSharedPreferences(APP_PREFERENCES,  
Context.MODE_PRIVATE);  
SharedPreferences.Editor editor = mSettings.edit();  
editor.putString(APP_PREFERENCES_NAME, personId);  
editor.apply();
```

Вікно містить поля формату `EditText` для вводу інформації та об'єкт `TextView`, який представляє собою кнопку з написом «Save», дивитись рис. 4.9. При натисканні кнопки закривається вікно, виконується розрахунок стартової вартості часу згідно наданих даних та йде конфігурування запиту на збереження даних до `Realtime Database`.

Для збереження даних необхідно виконати підключення до неї використовуючи вбудовані інструменти, аналогічно до підключення `Firebase Authentication`.

Так, як клієнт `Firebase` вже ініціалізовано, необхідно створити екземпляр бази даних та звернутись до неї, задавши ім'я таблиці і передати данні завдяки методу `push()`, все це виконує функція `setInfo()`, код якої наведено у лістингу 4.19.

Для структурування даних і уникнення поетапного запису даних у базу даних формату ключ – значення, необхідно створити клас User, який містить усі необхідні поля та передати його екземпляр до методу push(). У структурі проекту необхідно додати новий пакет «classes» і у ньому створити клас User.

Після виконання всього вище описаного користувач стає зареєстрованим та отримує доступ до додатку, а у базі даних з'являється запис, рис. 4.10, з переліком усіх необхідних даних користувача.

Лістинг 4.19 – Метод setInfo()

```
public static void setInfo(String className, User classInfo){
    FirebaseDatabase database =
    FirebaseDatabase.getInstance();
    DatabaseReference myRef =
    database.getReference(className);

    myRef.push().setValue(classInfo);
}
```

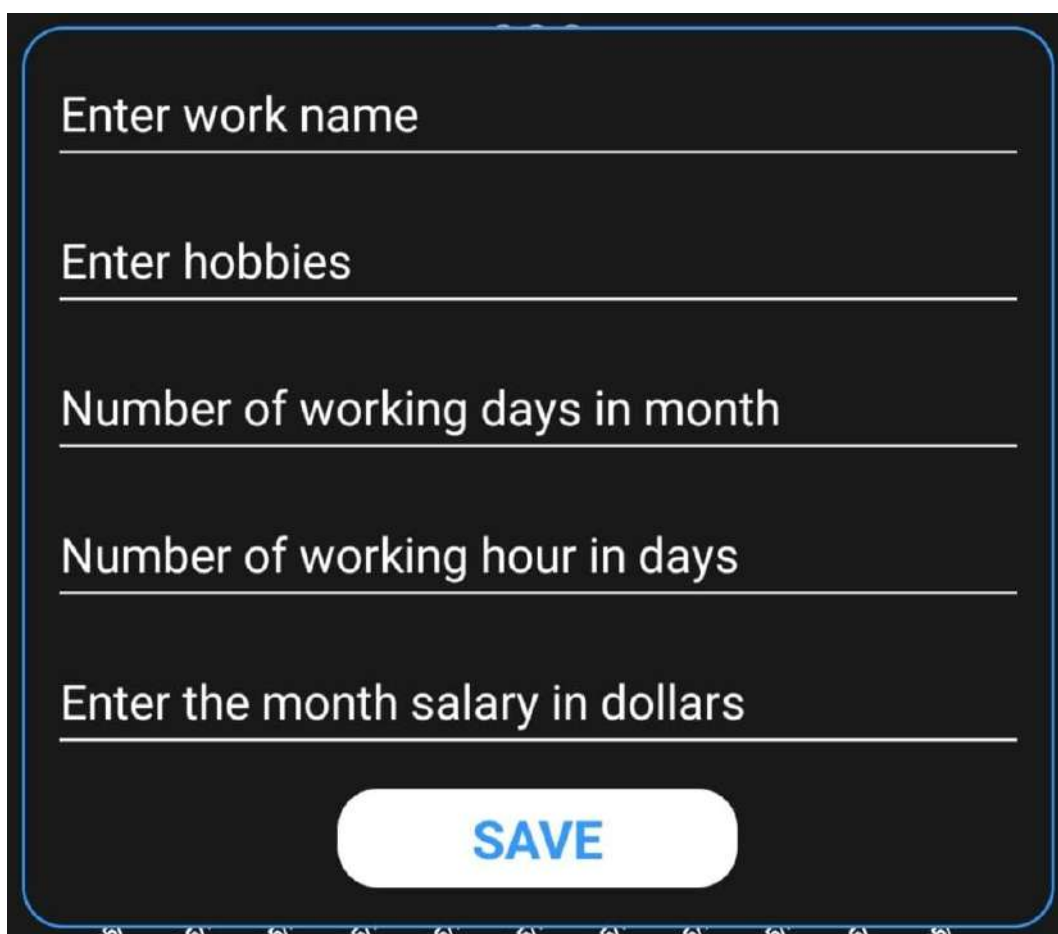


Рисунок 4.9 – Діалогове вікно надання додаткової інформації



Рисунок 4.10 – Перелік даних користувача у базі даних

4.4.1 Розробка функціоналу центральної сторінки

У верхній частині знаходиться головний елемент усього додатка, стилізований ProgressBar, - CustomGauge, в середині якого знаходиться об'єкт TextView.

Ця конструкція містить в собі таку змінну інформацію:

- поточне значення вартості часу користувача;
- цільове значення вартості часу користувача (за замовчення параметр має значення подвоєної вартості часу).

Для її отримання необхідно виконати запит Query до таблиці User з додатковим параметрами сортування:

- id користувача, отриманого за допомогою функції SharedPreferences;
- параметр сортування за ім'ям поля таблиці (personHourCost).

Отримання даних виконується завдяки методу addChildEventListener(), за допомогою методів Getter дістаємо дані з об'єкту User та передаємо їх до CustomGauge та TextView у ньому.

У нижній частині сторінки знаходиться ListView, для його наповнення необхідно створити клас, який буде описувати об'єкти, які надходять, та буде виступати типом отриманих даних.

У списку будуть відображатись графіки та прогрес бари (ProgressBar), для них необхідно передати таку інформацію:

- Ім'я заголовку прогрес бара;
- Текст напису на прогрес барі;
- Значення прогресу;
- Ім'я заголовку графіка;
- Масив координат першої лінії графіку;
- Масив координат другої лінії графіку;
- Масив підпису вісь ординат – X.

Тож клас (Graphix) повинен містити в собі сім полів, його код наведено у лістингу 4.20.

Лістинг 4.20 – Клас Graphix

```
public class Graphix {
    public String name_progress;
    public String onProgSpeed;
    public int progress;
    public String name_graphic;
    public float[] graphic1;
    public float[] graphic2;
    public String[] name_x_graphic;

    public Graphix(String name_progress, String onProgSpeed, int
progress, String name_graphic, float[] graphic1, float[]
graphic2, String[] name_x_graphic) {
        this.name_progress = name_progress;
        this.onProgSpeed = onProgSpeed;
        this.progress = progress;
        this.name_graphic = name_graphic;
        this.graphic1 = graphic1;
        this.graphic2 = graphic2;
        this.name_x_graphic = name_x_graphic;
    }
}
```

Надалі необхідна наявність адаптера, щоб зв'язати дані з елементом управління, у даному випадку елементом управління виступає ListView. Дані представлені у вигляді масиву строк, тому необхідно використати ArrayAdapter, адаптер бере масив даних та розміщує їх по порядку у ListView та, якщо необхідно,

виконує дії з View елементами ListView чи масиву даних. Для цього необхідно створено AdapterGraphix.

Так, як розмітка для ListView складається з двох елементів і порядок та кількість кожного з них заздалегідь не відома у одній строчці списку, необхідно непотрібні елементи приховувати. Саме тут на допомогу приходить адаптер, при отриманні масиву даних перевіряймо чи наявна назва елемента у поточній позиції масиву, якщо так то виводимо графік або прогрес бар на екран, якщо ж ні – ховаємо той чи інший.

Для виведення даних на графік необхідно зробити два запити до таблиці HistoryTimer, додавши поля фільтрації до них: унікальне id користувача, діапазон дати останнього тижня та минулого.

Після повернення значень від БД, складаємо кількість відпрацьованого часу у кожен з днів тижня та додаємо їх до масива даних першого графіку, аналогічно робимо із другим тижнем. Дати додаються до масиву строк для підпису графіку.

Аналогічним чином рахуються дані для прогрес бара годинної інтенсивності роботи користувача.

Розрахована інформація передається до адаптера та відображається на сторінці користувача.

4.4.2 Розробка функціоналу таймера

Вікно таймера повинно підраховувати час, виводити поточне значення зароблених коштів та передавати зібрані данні у базу даних.

У вікні таймера маємо кнопку, яка одночасно виступає і кнопкою страту і кнопкою зупинки. У початковому стані кнопка має назву – Start, при її натисканні відпрацьовує метод `setOnClickListener()`, який відстежує натиски. Метод має поле типу `boolean` (може приймати тільки значення `true` or `false`), якщо воно дорівнює `true` то кнопка має напис Start.

Якщо кнопка дорівнює Start то вона запускає таймер. Для цього створюється екземпляр класу `Chronometer` і у нього викликається метод `start()`.

Щоб виконувати дії з даними таймера під час його роботи необхідно використати метод `OnChronometerTickListener()`. Завдяки ньому можна виконувати операції зчитування даних та їх підрахунку у реальному часі, кожен такт таймеру.

У цьому методі зчитується поточний нарахований час та рахується поточна вартість часу користувача. Вона передається до об'єкту `TextView`, який відображає її на екрані таймеру.

Якщо кнопка дорівнює `Stop`, виконується зупинка таймера за допомоги методу `stop()`, з методу `OnChronometerTickListener()` беруться дані що до нарахованої вартості часу користувача та поточне значення таймеру. Робиться запит до методу `SharedPreferences` для отримання значення `id` поточного користувача та запит на отримання поточної дати від функції `nowData()`, яка знаходиться у головному `activity`, код функції наведено у лістингу 4.21.

Лістинг 4.21 – Метод `nowData()`

```
public static String nowData(){
    Date currentDate = new Date();
    DateFormat dateFormat = new
SimpleDateFormat("dd.MM.yyyy", Locale.getDefault());
    String dateText = dateFormat.format(currentDate);
    return dateText;
}
```

Після отримання усіх даних формується запит на відправку даних у базу даних. Для цього необхідно створити клас `HistoryTimer`. взяти його екземпляр, наповнити зібраними даними та відправити у базу даних, аналогічно до відправки класу `User`. Результат запису даних у базу даних наведено на рисунку 4.11.

Після зупинки таймеру дані відправляються до бази даних, зміна даних одразу проковує функцію перерахунку усіх наявних даних та оновлює їх на графіках, прогрес барах та на вікні історій.

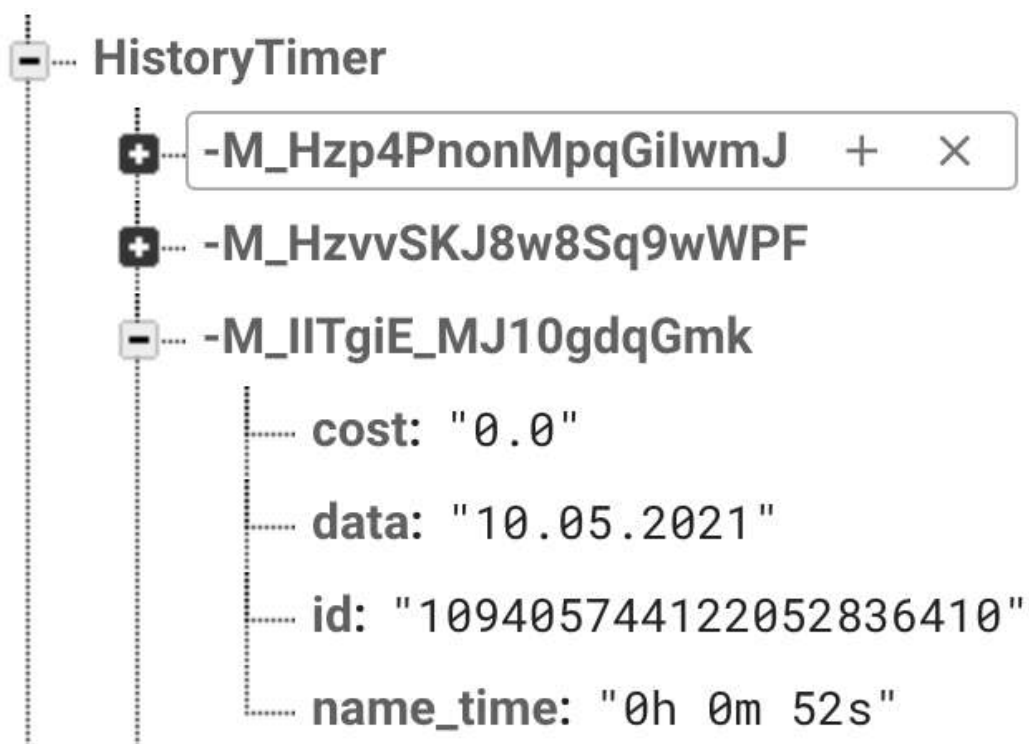


Рисунок 4.11 – Перелік значень таймера у базі даних

4.4.3 Розробка функціоналу лівої сторінки

Ліва сторінка містить лише два елемента, не враховуючі анімаційне зображення, `FrameLayout` – контейнер у якому змінюються 2 фрагмента з історіями, кожен з котрих має свій `ListView`, та `ChipNavigationBar`, який перемикає їх між собою.

Необхідно зв'язати перемикач з контейнером, для цього необхідно застосувати метод перемикача – `setOnItemSelectedListener()`, він дозволяє відстежити яка позиція у перемикача та зв'язати кожен з позицій із відображенням відповідного фрагмента, код наведено у лістингу 4.22.

Лістинг 4.22 – `ChipNavigationBar`

```
ChipNavigationBar chip_bar = v.findViewById(R.id.chip_bar);
chip_bar.setOnItemSelectedListener(new
ChipNavigationBar.OnItemSelectedListener() {
    @SuppressWarnings("NonConstantResourceId")
    @Override
    public void onItemSelected(int i) {
        Fragment selfragment = null;
        switch (i){
            case R.id.time_h:
```

```

        selffragment = new LeftTimerHFragment();
        break;
    case R.id.transak_h:
        selffragment = new
LeftTransactionHFragment();
        break;
    }

getManager().beginTransaction().replace(R.id.fragment_c
ontainer, selffragment).commit();
    }
});
chip_bar.setItemSelected(R.id.time_h, true);

```

Необхідно наповнити ListView кожного з фрагментів даними.

Для фрагмента з історією таймера необхідно отримати дані з бази даних, які туди відправило вікно таймера та вивести їх за допомогою адаптера. Клас для опису об'єкту отриманих даних вже наявний – HistoryTimer. Тож можна виконувати запит до бази даних.

За допомоги описаного вище методу SharedPreferences отримаємо поточне id користувача щоб за його допомоги відсортувати дані в запиті до БД. Запит виконується за допомоги екземпляру класу Query до якого передається екземпляр бази даних з параметрами запиту та через метод addChildEventListener(), код наведено у лістингу 4.23, слухається відповідь бази даних. Отримані дані конвертуються до об'єкту типу HistoryTimer. Після цього передаємо їх в HistoryAdapter адаптер. Він відстежує позицію даних в списку та виводить їх на екран.

Лістинг 4.23 – Метод addChildEventListener()

```

myQuery.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {
        try {
            HistoryTimer historyTimer =
snapshot.getValue(HistoryTimer.class);
            adapter_bottom.add(historyTimer);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        @Override
        public void onChildChanged(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {

        }

        @Override
        public void onChildRemoved(@NonNull DataSnapshot
snapshot) {

        }

        @Override
        public void onChildMoved(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {

        }
        @Override
        public void onCancelled(@NonNull DatabaseError
error) {

        }

    });

```

Для фрагменту з історією зарахувань коштів все аналогічно, але данні до БД відправляються з цього ж фрагменту і далі отримуються та через адаптер виводяться до користувача.

Додавання даних виконується користувачем через натискання кнопки, яка представляє собою елемент списку, що виводиться завжди у нульову позицію списку, дивитись рисунок 4.12.

При натисканні на цей елемент списку відпрацьовує метод `setOnClickListener()` та відкриває діалогове вікно вводу даних отримання коштів:

- назва платежу;
- дата отримання;
- сума.

Отримані дані відправляються до БД, аналогічно описаному методу `push()`, та виводяться на екран через адаптер.

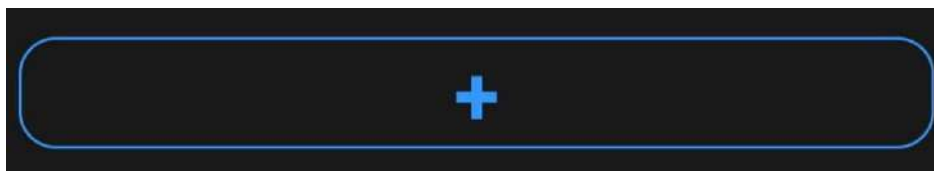


Рисунок 4.12 – Кнопка додавання записів отримання коштів

4.4.4 Розробка функціоналу правої сторінки

У верхній частині сторінки знаходиться картка з інформацією що до користувача. Для її відображення необхідно створити запит Query до таблиці User с БД та відсортовати інформацію за id користувача, отриманого за допомоги функції SharedPreferences, як описувалось раніше.

Метод addChildEventListener() повертає вміст таблиці User для поточного користувача та через методи Getter (гетер) виокремлює інформацію з об'єкту user та присвоює її значення до кожного з TextView за допомоги методу setText().

Одним з зворотних параметрів є посилання на картинку користувача, яку необхідно відобразити у картці користувача. Для цього використовується бібліотека Picasso. Вона дозволяє конвертувати посилання на зображення формату Uri у звичну картинку та помістити в об'єкт відображення зображень - CircleImageView. Додатково, завдяки Picasso задається зображення за замовченням та зображення яке буде відображатись при забогах у завантаженні зображення користувача. Код методу addChildEventListener() наведено у лістингу 4.24.

Лістинг 4.24 – Метод addChildEventListener() для картки користувача

```
//формування запиту з фільтром по id
Query myQuery = myRef.orderByChild("personId").equalTo(id);
    myQuery.addChildEventListener(new ChildEventListener()
    {
        @Override
        public void onChildAdded(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {
            try {
//відповідь БД
                User user = snapshot.getValue(User.class);
                Log.d("$$$$$$$$$$$$$$$$",user.personPhoto);
//Використання бібліотеки Picasso
                Picasso.with(getContext())
                    .load(user.personPhoto)
```

```

        .placeholder(R.drawable.avatar)
        .error(R.drawable.avatar)
        .into/avatar);
        nickname.setText(user.getPersonName());
        data.setText(user.getPersonName());
        data2.setText(user.getPersonWork());
        data3.setText(user.getPersonInfo());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
@Override
public void onChildChanged(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {
}
@Override
public void onChildRemoved(@NonNull DataSnapshot
snapshot) {
}
@Override
public void onChildMoved(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {
}
@Override
public void onCancelled(@NonNull DatabaseError
error) {
}
});

```

Додатково, у верхньому правому кутку картки користувача знаходиться кнопка виходу з облікового запису додатка «Logout», при її натисканні відпрацьовують функція `setOnClickListener()`, її вміст надано у лістингу 4.25, видаляє користувача з БД Firebase, перевантажує головну сторінку додатку, а з нею і усі фрагменти та видаляє всі записані данні через функцію `SharedPreferences`.

Лістинг 4.25 – Вміст методу `setOnClickListener()` для кнопки «Logout»

```

btn_out.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast toast = Toast.makeText(getContext(),
            "Log Out!", Toast.LENGTH_SHORT);
        toast.show();

        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent(getActivity(),
            HomeActivity.class);
    }
});

```

```

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }
});

```

У нижній частині сторінки знаходиться ListView для виведення інформації, а над ним фільтри.

Фільтр є наступним за ваговою складовою об'єктом - Spinner. Для його роботи також необхідна наявність адаптера, адже він теж являє свого роду список, але він приймає лише один масив даних тож створювати окремий клас для його роботи не потрібно.

Аналогічно до описаного вище запиту до БД дістається інформація що до переліку усіх наявних видів професій, представниками якої являються користувачі додатку. Фільтром у запиті виступає додатковий параметр – ім'я поля, у таблиці User, із вмістом назви професії (personWork).

Отриманні дані після вибірки формуються у масив типу String та передаються до адаптера - adapter_spinet, який виводить їх на сторінку.

При виборі одного з елементів списку спрацьовує Listener() та формує запит до БД з обраним параметром фільтрації. Через вище описаний метод addChildEventListener() повертаються дані та передаються до адаптера AdapterReiting, який виводить дані користувачеві.

Функціонал зміни даних користувачем у своїй картці реалізована через функцію changeData(). Вона викликається через метод setOnClickListener() з будь-якого із змінних полів. Метод викликає функцію та передає її ідентифікатор для запиту до БД, до конкретного поля та інформацію яку потрібно ввести. Після внесення змін одразу відпрацьовує метод який виводить дані з БД та оновлює дані у елементі TextView.

4.4.5 Налаштування Realtime Database

Для налаштування бази даних у реальному часі від Firebase необхідно перейти на сайт та під'єднати додаток до неї . Для цього необхідно ввести назву проекту та назву пакету додатку, які були задані раніше при розгортанні проекту.

Після цього Firebase надає конфігураційний файл, який необхідно додати до структури проекту та через меню Tools у Android Studio перевірити з'єднання та імпортувати метод для читання і запису інформації.

Повертаємось на сайт та переглядаймо меню налаштувань проекту у Firebase, де знаходимо пункт меню Realtime Database. Переходимо до нього та робимо читання та запис відкритими, база даних готова для роботи, рис. 4.13.

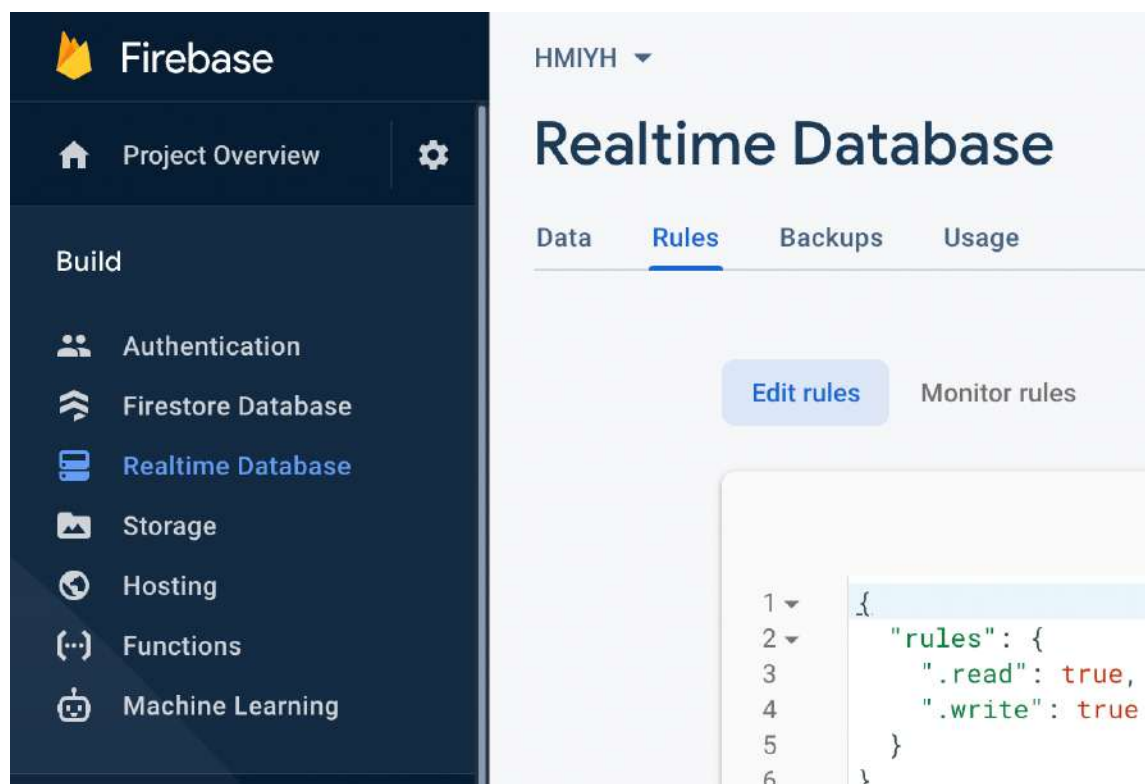


Рисунок 4.13 – Налаштування параметрів доступу до Realtime Database

4.4.6 Випробування системи

Після завершення основних робіт над програмним забезпеченням йде його тестування.

У першу чергу необхідно перевірити коректність роботи бази даних. Зазвичай для цього використовують різні інструменти, але так, як у цьому додатку БД має малі масштаби швидше буде перевірити все у ручному режимі. Додаємо до коду отримання відповіді та запиту до БД методи виводу інформації до консолі – `Log.d()`. Метод приймає два значення, перше це тег або ключ, друге змінна строкового типу.

Бачимо по вихідним даним у консолі, що все працює як слід і можна переходити до тестування інтерфейсу. Це робиться перебором всіх можливих шляхів користувача. Вся інформація виводиться вірно, а анімації працюють як слід.

Останнім етапом є випробування системи на реальному пристрої. Система займає на пристрої, приблизно, 15 мегабайт фізичної пам'яті, рисунок 4.14, використовує за один запуск 60 кілобайт переданих даних, дивитись рисунок 4.15. При повному розсортюванні система використовує до ста мегабайт оперативної пам'яті пристрою. Навантаження на процесор з двома ядрами складає до трьох відсотків. Тож, система використовує менше ресурсів, ніж було закладено, що є безперечним успіхом.

Таким чином, можна зробити висновок, що все працює як слід і дотримано усіх вище описаних вимог. Під час тестування інтерфейсу було виявлено та виправлено декілька текстових помилок, а в усьому іншому, вся система повністю працездатна.

Отже, було створено проект у IDE Android Studio та його структуру. Після чого створено основну сторінку, яка включає в собі три фрагмента, які і відображаються на екрані користувача. На основі розроблених раніше макетів сторінок був створений їх дизайн художником. На основі наданих матеріалів художником було створено користувацький інтерфейс за допомоги файлів розмітки XML. Після заповнення додатку тестовими даними та внесення правок було підключено сервіс Firebase для створення функціоналу додатку таких, як авторизація користувача та робота з базами даних у реальному часі. Виконано налаштування сервісів Firebase на головному сайті. Розроблено методи роботи з даними та їх обробки. У кінці було зроблено тестування додатку та виправлено виявлені недоліки. На виході було отримано повнофункціональний додаток, який відповідає всім вище зазначеним вимогам.



НМІУН
Версія 1.0

ПАМ'ЯТЬ

Всего 15,12 МБ

Приложение 14,88 МБ

Данные 240 КБ

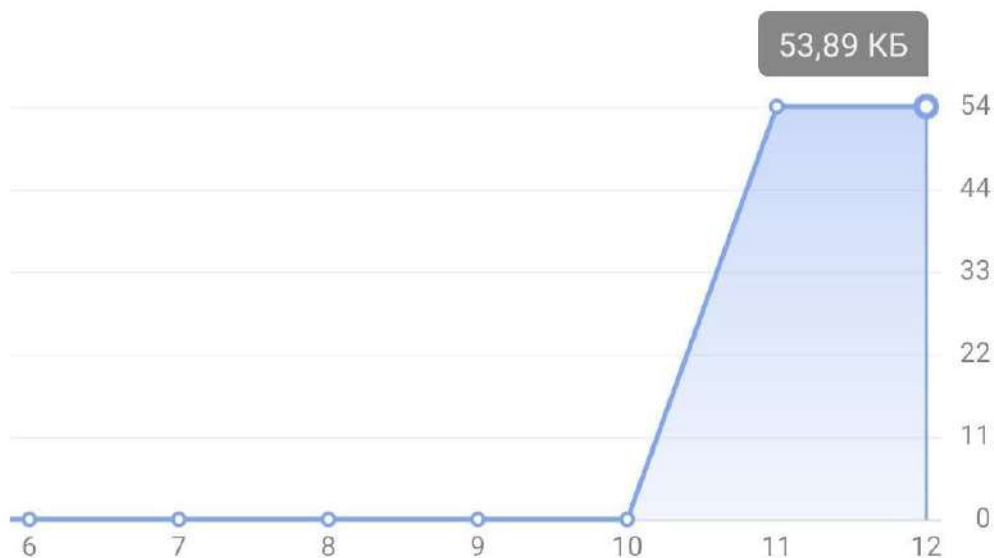
СБРОС

Кэш 133 КБ

ОЧИСТИТЬ КЭШ

Рисунок 4.14 – Використання пам'яті системою

Последние 24 часа ▼



Тарификация, выполняемая оператором, может отличаться от отображаемой здесь.

Использовано: 53,89 КБ, в активном режиме: 53,89 КБ, в фоновом режиме: 0,00 Б

Рисунок 4.15 – Використання мережі системою

ВИСНОВКИ

У результаті роботи був проаналізований об'єкт розробки для чіткого розуміння напрямку розробки та актуальності ідеї.

Були проаналізовані всі необхідні сфери втручання та описані вимоги до комп'ютерної системи.

На базі вимог було обрано платформу розробки, вибір пав на мобільну платформу, адже вона дозволяє знаходитись розробленій системі у постійному доступі у користувача.

Надалі, були проаналізовані операційні системи, які працюють на мобільній платформі. Важливим аспектом є розповсюдженість пристроїв, які працюють на кожній з операційних систем. На підставі цього була обрана операційна система Android.

Обравши платформу та ОС, було обрано мову програмування – Java, адже вона є базовою і має більшу кількість бібліотек.

Після визначення мови програмування, постало питання у виборі IDE для розробки. Після порівняння особливостей двох найпопулярніших IDE, вибір було зроблено на користь IDE Android Studio.

Додатково було проаналізоване апаратне забезпечення, конфігурований мінімальний набір характеристик пристрою користувача для комфортної взаємодії з додатком. У ході цього було з'ясовано, що головним критерієм є версія Android, мінімальне значення якої – п'ять, під назвою Lollipop та підключення до мережі.

Після вирішень питань що до клієнтської частини, були проаналізовані рішення для створення серверної частини. Замість класичної форми реалізації цієї частини, було обрано Realtime Database від компанії Google. Вона дозволяє реалізувати необхідну частину серверного функціоналу без самого сервера, що економить гроші та час розробки.

Після вирішення цього питання було визначено кількість таблиць, перелік полів та даних в них, які буде містити БД.

Спираючись на ідею вирішення поставленої проблеми, що до оцінки часу людини, було продумано та схематично реалізовано, головний алгоритм системи, який і повинен вирішити поставлену проблему.

У ході розробки, спираючись на описані вимоги до системи було продумано функціонал. На основі переліченого функціоналу додатку був розроблений користувацький інтерфейс, на базі якого було створено макет додатку, а після цього намальовано дизайн.

За готовим дизайном був створений інтерфейс користувача, який включав у себе три сторінки та прописана вся функціональність додатку.

Надалі було заповнено графічний інтерфейс тестовими даними для перевірки коректного відображення інформації. Після успішної перевірки було завершено роботу над функціоналом.

Для зберігання та керування даними було створено базу даних у реальному часі. На базі описаних вище параметрів було створено чотири таблиці, визначено поля та наповнено їх значеннями за замовченням. Після чого було зареєстровано додаток на сайті Firebase та встановлено з'єднання бази даних з додатком за допомогою конфігураційного файлу, створеного на сайті, та стандартних інструментів Android Studio.

Останнім етапом було випробування системи, у ході якого були виявлені та виправлені незначні недоліки додатку. Додатково, додаток було запущено на різних пристроях та перевірена адаптивність інтерфейсу користувача додатка до розмірів екрану пристрою. Після проходження усіх програмних тестів, додаток став готовий до використання.

Тож на виході було отримано повнофункціональний додаток, який являє собою відмінний інструмент для розрахунку вартості часу та облегує не тільки буденний вибір дій на його основі але і полегшує робочі відносини між замовником та виконавцем так, як виконавець має можливість точно розрахувати вартість виконаної їм роботи та надати цю інформацію замовнику. У додатку до всього вище описаного, додаток формує рейтинг користувачів за спеціальністю та цим спонукає користувачів досягати нових звершень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Основная валюта нашей жизни – это время URL: <http://edinstvo-gomel.by/index.php/2013-02-22-06-29-06/2505-2018-04-12-21-35-01.html> (дата звернення: 20.02.2021).
- 2 Монетизация URL: <https://promopult.ru/library> (дата звернення: 20.02.2021).
- 3 Сколько стоит время URL: <https://fincalculator.ru/skolko-stoit-vremya> (дата звернення: 20.02.2021).
- 4 Что такое фриланс: как работать на себя URL: <https://www.work.ua/ru/articles/jobseeker/654/> (дата звернення: 20.02.2021).
- 5 Мобильное приложение URL: <https://indicator.ru/label/mobilnoe-prilozhenie> (дата звернення: 23.02.2021).
- 6 О платформе Android URL: <https://oracle-patches.com/coding/3683-о-платформе-android#> (дата звернення: 23.02.2021).
- 7 Java URL: <http://progopedia.ru/language/java/> (дата звернення: 23.02.2021).
- 8 Что такое Kotlin URL: <https://thecode.media/kotlin/> (дата звернення: 23.02.2021).
- 9 Разработка приложений IOS или Android: какую платформу выбрать URL: <https://vc.ru/dev/162780-razrabotka-prilozheniy-ios-ili-android-kakuyu-platformu-vybrat> (дата звернення: 23.02.2021).
- 10 Что такое интегрированная среда разработки? URL: <https://www.kv.by/archive/index2008361105.htm> (дата звернення: 25.02.2021).
- 11 Eclipse IDE URL: https://ru.bmstu.wiki/Eclipse_IDE (дата звернення: 25.02.2021).
- 12 Android Studio: среда разработки мобильных приложений URL: <https://arduinoplus.ru/android-studio/> (дата звернення: 25.02.2021).

13 Git и GitHub: что это такое и в чем разница URL: <https://tproger.ru/translations/difference-between-git-and-github/> (дата звернения: 25.02.2021).

14 Android Studio vs Eclipse: преимущество каждой из сред URL: <https://kitapp.pro/android-studio-vs-eclipse-preimushhestvo-kazhdoj-iz-sred/> (дата звернения: 26.02.2021).

15 СУБД (система управления базами данных) URL: <https://itglobal.com/ru-ru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/> (дата звернения: 29.02.2021).

16 MySQL - система управления базами данных URL: <https://web-creator.ru/articles/mysql> (дата звернения: 29.02.2021).

17 SQL – что это такое простым языком URL: <https://www.zeluslugi.ru/info-czentr/it-glossary/term-sql> (дата звернения: 29.02.2021).

18 Что такое Firebase (realtime database) URL: <https://webkyrs.info/page/chto-takoe-firebase-realtime-databas> (дата звернения: 29.02.2021).

19 Fragment (Фрагменты) URL: <http://developer.alexanderklimov.ru/android/fragment.php> (дата звернения: 01.03.2021).

20 Аватар: что это такое URL: <https://promo.ingate.ru/seo-wikipedia/avatar/> (дата звернения: 05.03.2021).

21 Android Lollipop URL: https://ru.wikipedia.org/wiki/Android_Lollipop (дата звернения: 10.03.2021).

22 Что такое архитектура процессоров и как ее определить URL: <https://tvoysmartphone.ru/uroki/171-chto-takoe-arhitektura-processorov-i-kak-ee-opredelit.html> (дата звернения: 10.03.2021).

23 Инфраструктура URL: <https://ru.wikipedia.org/wiki/Инфраструктура> (дата звернения: 10.03.2021).

24 Гугл игры URL: https://ru.wikipedia.org/wiki/Google_Play (дата звернения: 10.03.2021).

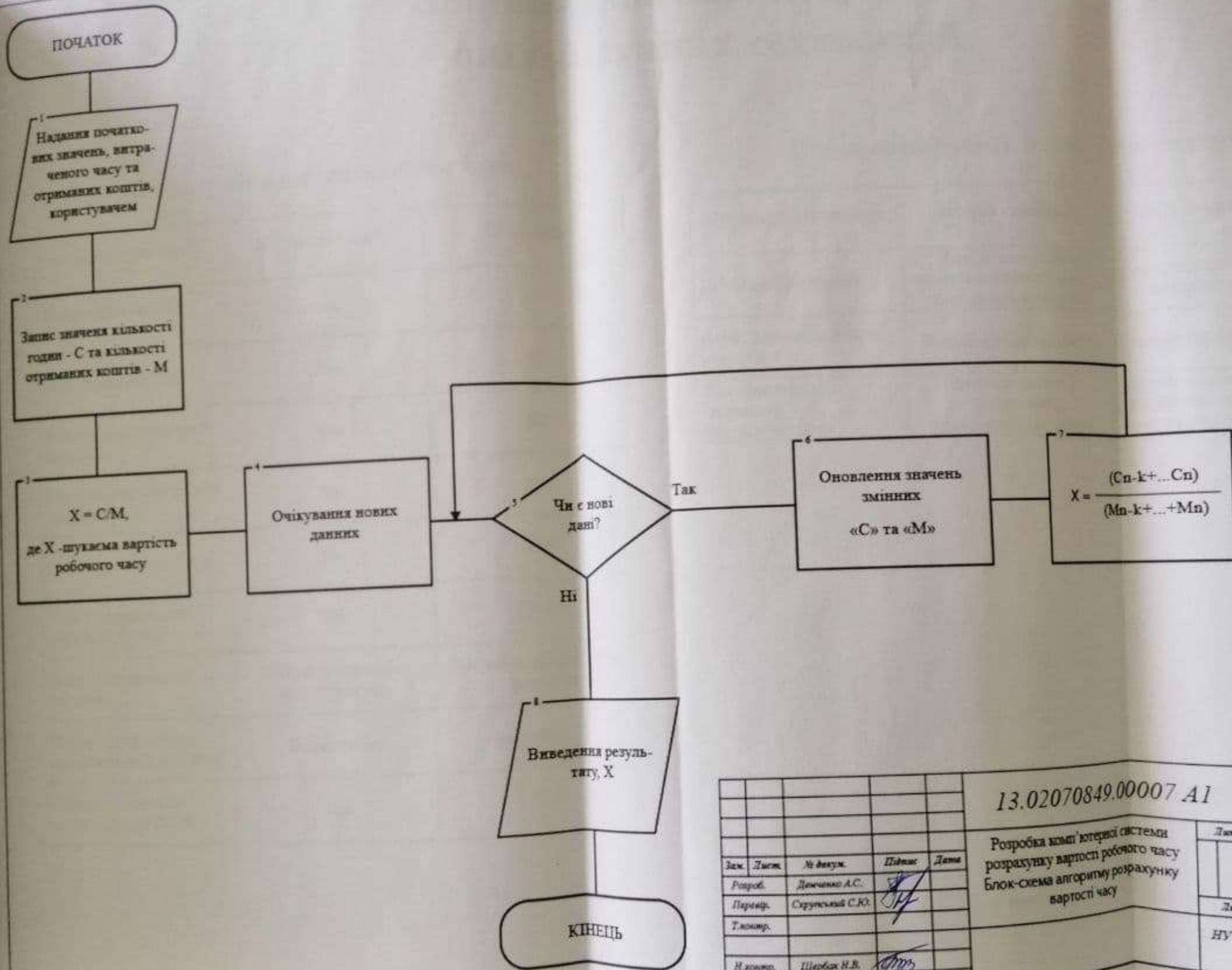
25 Что такое мобильное SDK URL: <https://www.carrotquest.io/blog/что-такое-mobilnyj-sdk/> (дата звернения: 05.03.2021).

26 Что такое Google реклама в приложении URL: <https://admob.google.com/intl/ru/home/resources/what-is-admob/> (дата звернения: 05.03.2021).

27 OneSignal – обзор сервиса URL: <https://startpack.ru/application/onesignal> (дата звернения: 05.03.2021).

28 Проверка подлинности Firebase URL: <https://firebase.google.com/docs/auth> (дата звернения: 05.03.2021).

29 SHA - 1 URL: <https://en.wikipedia.org/wiki/SHA-1> (дата звернения: 05.03.2021).



13.02070849.00007 A1				
Зам.	Лист	№ докум.	Підпис	Дата
Розроб.		Домченко А.С.	<i>[Signature]</i>	
Перевір.		Скученко С.Ю.	<i>[Signature]</i>	
Т.контр.				
Н.контр.		Щербак Н.В.	<i>[Signature]</i>	
Затверд.		Бударинська Р.К.	<i>[Signature]</i>	
Розробка комп'ютерної системи розрахунку вартості робочого часу				
Блок-схема алгоритму розрахунку вартості часу				
Лист 1		Листів 1		
НУ "Запорізька політехніка" КНТ-517				

Порівняльна характеристика використаних технологій

IDE розробки клієнтської частини

Характеристики	IDE Android Studio	IDE Eclipse
Підтримка компанії Google	Так	Ні
Наявність емулятора	Так	Ні
Вузько спеціалізований напрям	Так	Ні
Підтримка декількох мов розробки	Так	Так
Підтримка мови C/C++	Так	Так
Наявність російської мови інтерфейсу?	Так	Так
Потреби потужності системи	Потребує великого об'єму ресурсів	Потребує малого об'єму ресурсів
Платне чи безплатне програмне забезпечення?	Безкоштовне	Безкоштовне
Наявність системи контролю версій GIT	Так	Ні

Платформа роботи клієнтської частини

Порівняльні параметри	Android - пристрій	IOS - пристрій
Де цільова аудиторія	Більш масовий та орієнтований на середній або бюджетний ринок	Орієнтована на преміум користувачів
Який додаток дешевше розробити	Розробка більш дешева	Розробка дорожча
Тип монетизації: покупки у додатку/реклама	Знижка покупна спроможність, монетизація рекламою	Покупна здатність у 5 разів вища, монетизація покупками у додатку

13.02070849.00007 ПЛІ				
Зам.	Лист	№ докум.	Підпис	Дата
Розроб.	Довгаченко А.С.			
Перевір.	Скороходський С.Ю.			
Т.ліст.				
Н.ліст.	Щербак Н.В.			
Затверд.	Кудрявцев Р.К.			
Розробка комп'ютерної системи розрахунку вартості робочого часу				
Порівняльна характеристика використаних технологій				
			Лист 1	Листів 1
НТУ "Запорізька політехніка" КНТ-317				

Макети сторінок комп'ютерної системи

Макет сторінки історій

The diagram illustrates a mobile application interface layout. It consists of three main sections:

- Top Navigation Bar:** A horizontal bar at the top containing a single button labeled "Відображення навігації" (Navigation Display).
- Central Content Area:** A large rectangular area below the top bar, labeled "Область відображення історії танцю та надходження коштів" (Area for displaying dance history and income).
- Bottom Navigation Bar:** A horizontal bar at the bottom containing five buttons labeled "Меню", "Навігація", "Історія танцю", "Навігація", and "Відомості" (Menu, Navigation, Dance History, Navigation, Information).

Макет сторінки статистики

Головний елемент відображення
поточної вартості часу

Відображення навігації

Область відображення графіків та "Progress Bar" з
даними статистики

Логотип
Тайм-спу

Макет сторінки рейтингу

Карта району

Карта користувачів

Фільтри для відображення рейтингу

Відображення навігації

Полі з рейтингом користувачів

Зем.	Лист	№ докум.	Подпись	Дата	
Разработ.		Демченко А.С.			
Провер.		Степановский С.Ю.			
Т.эксперт.					
Н.эксперт.		Щербак Н.В.			
Технадзор		Кудрявцева Р.К.			

13.02070849.00007 ПЛ2

Розробка комп'ютерної системи
розрахунку вартості робочого часу
Макети сторінок
комп'ютерної системи

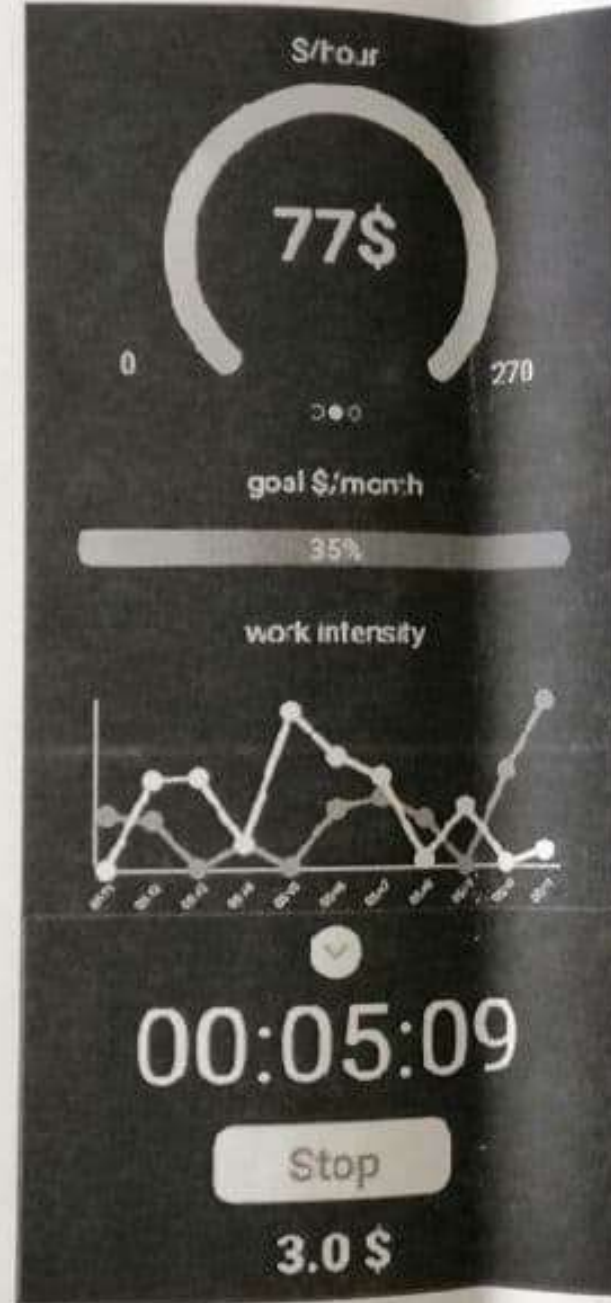
Лист	Маса	Матриця
Лист 1	Листів 1	
НУ "Запорізька політехніка" КНТ-517		

Демонстрація роботи комп'ютерної системи

Сторінка авторизації користувача



Сторінка статистики та таймеру



Сторінка історії роботи таймеру



Сторінка рейтингу користувачів



Зем.	Лист.	№ докум.	Підпис	Дата
Розроб.		Демонстрація А.С.		
Парент.		Сторінка С.Ю.		
Технік.				
Н.контр.		Щербак Н.В.		
Заступник.		Кудрявцев Р.В.		

13.02070849.00007 ПЛІЗ

Розробка комп'ютерної системи
розрахунку вартості робочого часу
Демонстрація роботи
комп'ютерної системи

Лист.	Маса	Масштаб
Лист 1	Лист 1	
НУ "Запорізька політехніка"		
КНТ-317		