ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка
до бакалаврської кваліфікаційної роботи
на ступінь вищої освіти бакалавр

на тему: «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ KOTLIN НА БАЗІ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ ПОДОРОЖЕЙ ТА ПОЇЗДОК КОРИСТУВАЧІВ»

Виконав: студент 4 курсу, групи ПД-44
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Віштак А.Ю.
(прізвище та ініціали)
Керівник Коваленко Д.С.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтроль
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____О.В.Негоденко

«____»_____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Віштаку Андрію Юрійовичу_____

1.Тема роботи: « Розробка програмного забезпечення мовою Kotlin на базіопераційної системи Android для фінансового планування подорожей та поїздок користувачів»

Керівник роботи Коваленко Данило Сергійович

Затверджені наказом вищого навчального закладу від — «12» березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вхідні дані до роботи:

3.1. Предметна частина та специфікація вимог

3.2. Програмне забезпечення мовою Kotlin на базі Android

3.3. Розробка

3.4. Науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз предметної області та специфіка вимог до програмного забезпечення.

4.2.Проектування програмного забезпечення мовою Kotlin на базі операційної системи Android.

4.3. Реалізація програмного забезпечення фінансового планування.

4.4. Висновки

5. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21 – 22.04.21	
2	Аналіз існуючих прототипів	22.04.21 – 23.04.21	
3	Дослідження програмних засобів	23.04.21 – 01.05.21	
4	Моделювання об’єкту проектування	01.05.21 – 05.05.21	
5	Розробка функціоналу бота	05.05.21 – 07.05.21	
6	Вступ, висновки, реферат	07.05.21 – 08.05.21	
7	Розробка презентації застосунку	08.05.21 – 24.05.21	
8	Попередній захист роботи	25.05.21	

Студент Віштак А.Ю.

Керівник роботи Коваленко Д.С.

РЕФЕРАТ

Текстова частина магістерської робот 56 с., 13 рис., 19 джерела.

Об'єкт дослідження – процес внесення інформації про планування коштів, тобто фінансового планування подорожей та поїздок користувачів.

Предмет дослідження – програмний продукт для фінансового планування подорожей та поїздок користувачів. Методи розробки базуються на технології Kotlin Android Studio.

Мета роботи – є розробка мобільного програмного забезпечення для фінансового планування подорожей та поїздок користувачів для мобільної платформи.

Метод дослідження – аналітичний з використанням комп'ютерних технологій.

В роботі приведено основні дані про характеристики програм, їх типи та види розгортання. Здійснено аналіз існуючих систем, вказані їх недоліки та запропоноване власне рішення.

Запропоноване програмне рішення, що складається із розроблених клієнтського додатку для ОС Android.

Ключові слова ВИТРАТИ, ФІНАНСОВЕ ПЛАНУВАННЯ. ПОЇЗДКИ, ANDROID, KOTLIN, АРХІТЕКТУРА, БАЗА ДАНИХ

ЗМІСТ

ВСТУП

Сучасній людині важко уявити життя без високотехнологічних пристроїв, таких як смартфон, планшет, тому, що ці науково - технічні новинки дозволяють не тільки підтримувати зв'язок з людьми із різних куточків світу, але і допомагають у формуванні та розвитку сучасного інформаційного простору. Через глибоку взаємодію з користувачем та можливістю покрити майже всі його інформаційні потреби виникла популярність мобільних додатків. За рахунок гнучкості, мобільні додатки замінили собою цілий ряд пристроїв, які були необхідними для користувача і колись здавалися незамінними, це навігатор, радіо приймач, фотокамера, календар, блокнот, відеокамера. На сьогодні за допомогою смартфону можна не тільки робити високо-якісні зображення, а й переглядати інформацію на сторінках у соц.мережах, блогах, інфо.порталах. З часом постає питання фінансового планування подорожей та поїздок, вдосконалити його і відмовитися від неконструктивних витрат. Планування персонального заощадження на подоржі та поїздки привчає до дисципліни у питанні фінансів, вказує на не завжди конструктивні пріоритети розподілення коштів, також може стати інструментом поліпшення фінансового становища сім'ї. Тому, перш ніж братися за планування персональних витрат на подорожі та поїздки, потрібно врахувати, що це також певна робота, вона потребує часу, уваги, концентрації і рішучості. Багато людей досить добре почуваються і без ведення обліку витрат і при цьому успішно справляються з розподілом коштів. І тому, щоб прийняти рішення в першу чергу необхідно, вияснити причини того, що підштовхує взяти під контроль фінансові витрати на подорожі та поїздки. Ще слід врахувати, що ведення фінансового обліку і планування подорожей та поїздок досить трудомістка справа і без конкретної цілі розпочинати роботу немає сенсу. Існує багато схем, порад, методів ведення і планування подорожей та поїздок.

Сам процес планування вбачає прийняття рішень і здійснення заходів, які на кожній стадії планування впливають на майбутнє користувача. Тобто потрібно постійно на кожному етапі фінансового планування враховувати нову інформацію. Засобом вирішення цих питань є формування системи інтегрованого фінансового планування з розробкою варіантних сценаріїв залежно від передбачуваних тенденцій внутрішнього розвитку і зміни зовнішнього середовища. В першу чергу має значення розробка методичного інструментарію ризико-орієнтованого фінансового планування для прийняття планових рішень щодо забезпечення фінансової стійкості в умовах невизначеності.

Надзвичайно важливим аспектом в створенні програмного забезпечення є архітектура. Вона впливає на швидкість розробки, можливість підтримки, тестування, гнучкість та стабільність програмного продукту. Коримтувачі мають змогу автоматизувати процес розподілу коштів на подорожі та поїздки, отримуючи з додатку всю необхідну для цього інформацію та набір інтерактивних інструментів. Отже, підсумовуючи наше дослідження, можна стверджувати, що на сьогодні фінансове планування є одним із альтернативних фінансових інструментів, яке використовується користувачами для налагодження фінансової діяльності та підвищення власної рентабельності й платоспроможності, як одних із найбільш важливих показників міцного фінансового стану користувача. Приділяючи більше уваги фінансовому плануванню, можна досягти зміцнення фінансової стабільності користувача.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СПЕЦИФІКАЦІЇ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ ФІНАНСОВОГО ПЛАНУВАННЯ ПОДОРОЖЕЙ ТА ПОЇЗДОК.

1.1. Поняття фінансового планування

Незалежно від часу постає питання фінансового планування подорожей та поїздок, вдосконалити його і відмовитися від неконструктивних витрат. Процес розробки системи фінансових планів - це є фінансове планування, і полягає він у визначенні фінансових цілей. Встановленні ступені відповідності цих цілей поточному фінансовому стані і формулюванні послідовності дій, спрямованих на досягнення поставлених цілей. В процесі планування розробляють узагальнені фінансові показники кількісного та якісного характеру і визначають найбільш ефективні шляхи їх досягнення. Процес фінансового планування обов'язково включає такі етапи:

- аналіз поточного фінансового стану;
- прогнозування майбутніх значень планових показників;
- безпосереднє складання планів та розробка процедури внесення коректив в процес реалізації планів.

Планування персональних відкладень на подоржі та поїздки дисциплінує у питанні фінансів, вказує на не завжди конструктивні пріоритети розподілу коштів, а також може стати інструментом поліпшення фінансового становища сім'ї. Але, перш ніж братися за планування персональних витрат на подорожі та поїздки, потрібно врахувати, теи що це також робота, і вона потребує часу, уваги, концентрації і рішучості. Багато людей досить добре почуваються і без ведення обліку витрат і при цьому успішно справляються з розподілом коштів. При прийнятті рішення в першу чергу необхідно, вияснити причини того, що спонукає взяти під контроль фінансові

витрати на подорожі та поїздки. Ще слід врахувати, що будь яке ведення фінансового обліку і планування подорожей та поїздок досить трудомістка справа і без конкретної цілі займатися нею немає сенсу. Існує безліч схем, порад, методів ведення і планування подорожей та поїздок. Проте пропонувати якийсь метод, як універсальний, недоречно. Мінливість життя і його мобільність навряд чи дозволять вам безклопітно вести контроль фінансового планування подорожей та поїздок по чітко обраній схемі. Зміна курсу валют, подорожчання путівок, незаплановані витрати на харчування та екскурсії під час подорожі та інші зваби і непередбачувані витрати здатні значно сколихнути ваш персональний бюджет. Тому логічно обрати найбільш підходящу схему і керуватися нею.

1.2. Специфікація вимог до програмного забезпечення та аналіз існуючих аналогів, що реалізують функції системи фінансового планування подорожей та поїздок.

Специфікація вимог до програмного забезпечення – це є структурований набір вимог до програмного забезпечення і її зовнішні інтерфейси призначені для створення основи для угоди між замовником і розробником про те, як працювати з програмним забезпеченням. Може включати в себе деякі користувальницькі скрипти англ. use cases, які описують параметри взаємодії між користувачами і програмним забезпеченням. Призначені для користувача сценарії є засобом для представлення функціональних вимог. На додаток до сценаріїв користувача, специфікація містить нефункціональні вимоги, які накладають обмеження на дизайн або реалізація. Функціональні вимоги визначають, що повинна робити програмна система. Він визначає функцію програмної системи або її модуля. Функціональність вимірюється як набір входів в систему, що перевіряється, на виході з системи. Реалізація функціональних вимог в системі планується на етапі проектування системи, тоді як, у випадку нефункціональних вимог, це планується в документі

Архітектура системи. Функціональна вимога підтримує генерування нефункціональних вимог. Функціональні вимоги можуть включати такі компоненти, які можна виміряти в рамках функціонального тестування:

1. Сумісність
2. Безпека
3. Точність
4. Відповідність

Нефункціональна вимога говорить про „якою повинна бути система", а не „що повинна робити система" (функціональна вимога). Вони здебільшого походять від функціональних вимог, заснованих на вкладі замовника та інших зацікавлених сторін. Деталі реалізації нефункціональних вимог задокументовані в документі Архітектура системи. Нефункціональні вимоги пояснюють якісні аспекти системи, що будується, а саме. продуктивність, портативність, зручність використання і т. д. Нефункціональні вимоги, на відміну від функціональних вимог, впроваджуються поступово в будь-якій системі.

До нефункціональних вимог належать наступні підтипи (невичерпні):

1. Продуктивнісь
2. Юзабіліті
3. Технічне обслуговування
4. Надійність
5. Переносимість
6. Підтримка
7. Адаптованість і інше.

Вимоги складають основний елемент для розробки будь-якої програмної системи. Можна побудувати систему з функціональними вимогами, але її можливості неможливо визначити та виміряти. Сказавши це, дуже важливо мати

якісні функціональні вимоги, що випливають із вимог бізнесу, щоб мати якісну працюючу систему програмного забезпечення.

Фінансовн планування полягає в контролі щоденних доходів і витрат, планування подорожей з метою досягнення довгострокових цілей, регулярного формування заощаджень з метою забезпечення фінансової безпеки на випадки непередбачених форс-мажорних обставин.

У процесі обліку особистих планувань виникають різні проблеми, деякі з них це:

1) планування фінансових заощаджень для майбутніх поорожей, яке вимагає деяких обчислень, використовуючи поточні дані фінансового стану та статистичні дані, засновані на попередніх доходах і витратах;

2) деякі сім'ї мають спільні бюджети, і було б зручно зберігати дані разом, використовуючи один бюджет, та забезпечити різнорівневий доступ до фінансів різним членам сім’ї.

Мобільний додаток для фінансового планування вирішує проблеми користувачів таким чином:

- додаток має хмаре зберігання для даних бюджетів; усі користувачі мають обліковий запис для доступу до їх даних;
- користувачі отримують доступ до хмари зберігання з мобільного додатку;
- користувачі можуть бути згруповані в сімейства; бюджет може бути особистим чи спільним для сім'ї;
- якщо користувач хоче, щоб деякі витрати або доходи були приватними всередині сімейного бюджету, то він може помітити його спеціальним прапором, тоді і опис, і класифікатор будуть недоступні для інших членів сім'ї;
- кожен користувач може мати кілька бюджетів (особистий або сімейний).

З допомогою мобільного додатку користувач впорається з підрахунком власних коштів набагато якісніше і легше, за рахунок швидкого виконання арифметичних

операцій. Для фінансового планування подорожей існує безліч мобільних додатків, які не завжди повною мірою виконують функції, потрібні користувачеві. Для розробки додатку для фінансового планування та подорожей з оптимальним набором функцій проаналізовано переваги та недоліки найвідоміших аналогів.

1.2.1 Аналіз додатку « TravelSpend »

Основним конкурентом є додаток « TravelSpend ». За допомогою цього додатку можливе відстеження витрат під час подорожі світом. Це зручно, якщо користувач планує наступну поїздку або вже перебуває у відпустці. Цей додаток призначений для мандрівників, незалежно від того, чи є користувач індивідуальним мандрівником на відпочинок, парою, яка вирушає разом, або групою друзів на вихідних. Користувач відстежує свої витрати на подорожі. Додаток швидкий і простий у використанні та працює в автономному режимі. Користувач можете додавати фотографії та розподіляти витрати протягом декількох днів. Додаток допомагає користувачеві відстежувати бюджет подорожей та економити гроші. Додавати витрати можливо в будь-якій валюті. Вони автоматично конвертуються у нашу домашню валюту. Розділити рахунки, перевірити залишки та розрахуватись із боргами у рамках TravelSpend. Переглянути дані про наші витрати. Можна проаналізувати свої витрати, щоб уникнути перевитрат.

Вигляд діалогових вікон програми «TravelSpend» показано на рисунку 1.1

Рисунок 1.1 - Діалогові вікна програми.[1]

1.2.2 Аналіз додатку «Skyscanner»

З цією програмою ми можемо планувати свої подорожі і знаходити недорогі авіаквитки, номери в готелях, а також прокатні автомобілі. Програма Skyscanner шукає найдоступніші найкращі варіанти через своїх туристичних партнерів. Подібно до рейсів Google, ми можемо бачити найдешевші дати польоту, а також отримувати сповіщення про зміну цін. Тут можна знайти найвигідніші ціни на авіаквитки, а також можливо відслідковувати їх зміни і отримувати своєчасні повідомлення про зниження цін на обрані нами напрямки. Вибравши найбільш підходящий варіант перельоту, зможимо швидко забронювати квитки без додаткових комісій. Якщо ми не впевнені, куди саме хочемо поїхати, Skyscanner пропонує категорію, яка дозволяє ознайомитись із "Найвищими пропозиціями" із найближчого аеропорту за доступними цінами. Можемо відстежувати свої замовлення зі статусом на смартфоні та перевіряти оновлення в Подорожах та

переміщувати заброньовані рейси з однієї подорожі до іншої на вкладці Подорожі. А завдяки функції "Краще пропозиції", зможемо вибрати дні для авіаперельоту, коли ціни на обрані напрямки самі низькі. Для цього, програма дозволить нам відфільтрувати рейси за кількістю пересадок, часу в дорозі, авіакомпанії, класу обслуговування, а також часу вильоту і прибуття. Программа Skyscanner допомагає знайти кращі і недорогі готелі. Для цього, вона порівняє пропозиції сотень тисяч готелів через найбільш популярні сервіси бронювання і запропонує нам найоптимальніші варіанти. Приємною особливістю програми є збереження історії пошуку.

Вигляд діалогових вікон програми «Skyscanner» показано на рисунку 1.2

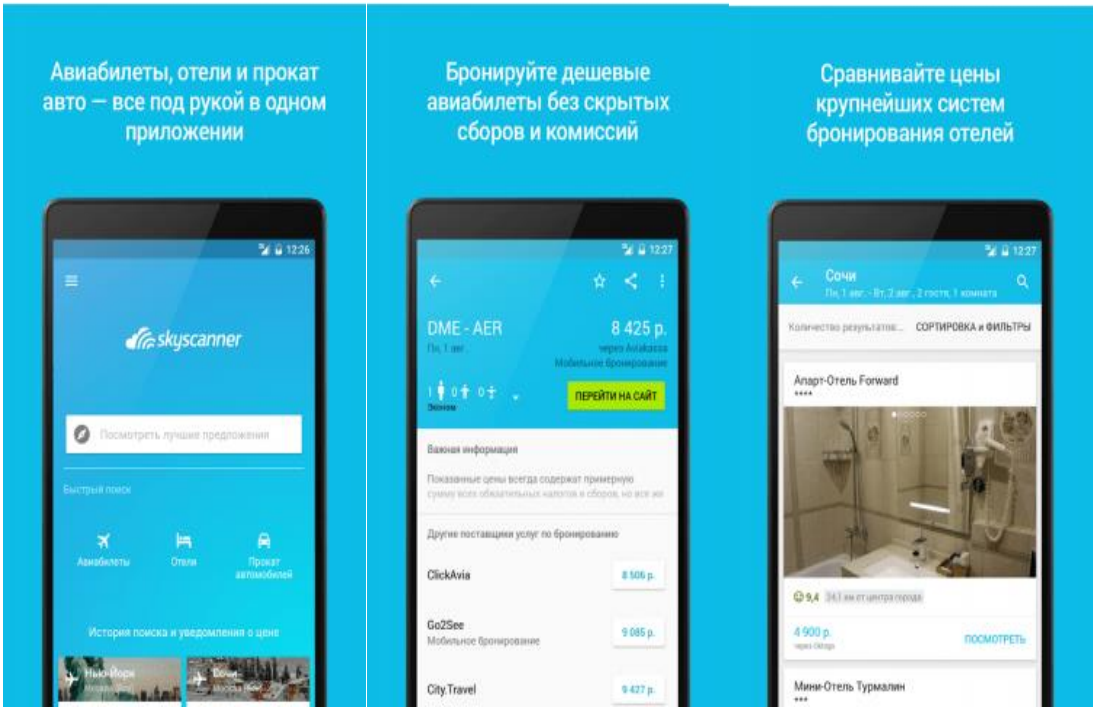

Рисунок 1.2 - Діалогові вікна програми «Skyscanner»[2]

1.3 Висновки розділу 1.

У першому розділі зроблено аналіз предметної області та специфікації вимог програмного забезпечення для контролю фінансового планування подорожей та поїздок, яка потребує розв'язання, а також наведено особливості класифікації мобільних додатків та визначено, до якого виду належить додаток, що розробляється, виявлено технічну проблему. Розглянуто існуючі додатки в області створення програмних засобів для ведення персонального планування. У них є свої мінуси та плюси. Тому є сенс у розробці власного рішення, і для цього було поставлено завдання, створити власний програмний продукт.

Проведено технічний аналіз основних аспектів, пов'язаних з розробкою мобільного додатку, що дозволяє зробити висновок, що розробка програмного продукту є доцільною на сучасному етапі.

Для створення власного продукту визначено варіанти використання програми та висунуто функціональні та нефункціональні вимоги. Обрано підходящу сжему для створення додатку.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ ПОДОРОЖЕЙ І ПОЇЗДОК ТА ОБГРУНТУВАННЯ ЙОГО АРХІТЕКТУРИ.

2.1 Визначення архітектури додатку

Набором внутрішніх структур програмного забезпечення є архітектура програмного забезпечення, складається вона із зв'язків, компонентів, доступних ззовні властивостей цих компонентів і можливих взаємодій між компонентами. Під компонентом в цьому визначенні мається на увазі довільний структурний елемент програмного забнзпечення, який можна виділити, визначивши інтерфейс взаємодії між компонентом і всім, що його оточує. Частіше при розробці програмного забезпечення термін "компонент" це одиниця розгортання, найменша частина системи, і її можна включити або не включити до складу.

Архітектура програмного забезпечення - набір структур або уявлень, що мають різні рівні абстракції і що показують різні аспекти, структуру відділів програмного забезпечення, структуру розгортання, або прив'язки компонентів програмного забезпечення до фізичних машин, можливі сценарії взаємодій компонентів, що об'єднуються зіставленням всіх представлених даних із структурними елементами програмного забезпечення. Характеристика якості програмного забезпечення в цілому визначає архітектура. Вона служить основним засобом спілкування між розробниками, і всіма іншими особами, зацікавленими в даному програмному забезпеченні. Вибір архітектури дає спосіб реалізації вимог на високому рівні абстракції. Саме архітектура майже повністю визначає характеристики програмного забезпечення, як надійність, переносимість і зручність супроводу. Впливає і на зручність використання, ефективність програмного забезпечення, які дуже залежать і від реалізації окремих компонентів.

Значно менший вплив архітектури на функціональність задану функціональність можна реалізувати, використавши абсолютно різні архітектури. Тому, вибір між однією або іншою архітектурою визначається більшою мірою, а саме нефункціональними вимогами і необхідними властивостями програмного забезпечення з точки зору зручності супроводу та переносимості. Для побудови гарної архітектури треба враховувати можливі недоліки між вимогами до різних характеристик і вміти вибирати компромісні рішення, що дають прийнятні значення за всіма показниками. Для підвищення ефективності в загальному випадку вигідніше використовувати монолітні архітектури, в яких виділено невелике число компонентів. Ним забезпечується економія пам'яті, так як кожний компонент зазвичай має свої дані, а тут число компонентів мінімальне, так і часу роботи, оскільки можливість оптимізувати роботу алгоритмів обробки даних є також тільки в рамках одного компонента. Дивлячись з іншого боку, для підвищення зручності супроводу, навпаки, краще розбити систему на велике число окремих маленьких компонентів, з тим щоб кожен з них вирішував свою невелику, але чітко певну частину загальної задачі.

Для того щоб підвищити надійність краще використати невеликий набір простих компонентів, або продублювати функції, тобто зробити декілька компонентів відповідальними за вирішення однієї підзадачі. Звернемо увагу, що помилки в програмному забезпеченні частіше носять невипадковий характер. Вони повторювані, на відміну від апаратного забезпечення, де помилки пов'язані з випадковими змінами характеристик середовища і можуть бути подолані простим дублюванням компонентів без зміни їх внутрішньої реалізації. Тоді при такому забезпеченні надійності портібно використовувати способи рішення однієї і тієї задачі в різних компонентах.

Іншим прикладом суперечливих вимог служать характеристики зручності використання і захищеності. Чим сильніша захищена система, тим більше

перевірок, процедур ідентифікації потрібно проходити користувачам. Відповідно, тим менш зручна для них робота з такою системою. При розробці реальних систем доводиться шукати деякий розумний компроміс, щоб зробити систему досить захищеною і здатною поставити відчутну перепону для несанкціонованого доступу до її даних і, в той же час, не відлякати користувачів складністю роботи з нею.

«Реалізація моделі предметної області являє собою архітектурне моделювання проектованого об'єкта. Загальний план реалізації, складений архітектором програмного забезпечення, регламентує способи отримання конкретних систем із загальної архітектури і компонентів». [7]

Архітектурна частина проекту складання моделі предметної області містить описи:

- інтерфейсу замовника для запуску конкретних підсистем;
- обробки запитів на зміни і розробку;
- процесів складання компонентів;
- вимірювань, супроводу та оптимізації бізнес-процесів.

Ця частина проекту описує збірку розробки об'єкта з імовірним використанням автоматизованих засобів для складання моделі. Рівень автоматизації збирання залежить від безліч факторів і пов'язаний з програмно-технічною оснащеністю проекту, і з наявністю достатнього рівня застосовуваних артефактів предметної області .

В загальному можливі наступні варіанти складання моделей проектів:

1. Збірка додатків з компонентів проводиться вручну.
2. Для складання компонентів застосовуються різноманітні інструментальні засоби. Документація формується автоматично.
3. Автоматичне складання моделі об'єкта із застосуванням інструментальних засобів для замовника (коштів породжує програмування), за допомогою яких формується запит на необхідне програмне забезпечення.

Виробництво програми може бути досягнутою одним запитом, якщо в ньому не втримується частини, створюваних традиційними методами розробки.
Вся документація також формується автоматично.

2.2 Обгрунтування вибору технологій

Розробка дизайну і інтерфейсу мобільного додатку – є найважливішим етапом у створенні програмного продукту. Від цієї роботи залежить те, як користувач сприйме додаток, чи сподобається він йому, чи буде додаток зручним в експлуатації, і чи стане він популярним.

Інтерфейс – це зовнішня оболонка додатка разом із програмами керування доступом і іншими схованими від користувача механізмами керування, що дає можливість працювати з документами, даними й іншою інформацією, що зберігається в комп’ютері чи за його межами. Головна мета будь-якого додатка – забезпечити максимальну зручність і ефективність роботи з інформацією: документами, базами даних, графікою, зображеннями.

Дизайн мобільного додатку має бути найпростіший і зрозумілий. З погляду на те, що мобільні пристрої мають маленький за розмірами екран, то при проектуванні додатків для них не можна керуватися тими ж правилами, що і для ПК. Основними вимогами до дизайну інтерфейсу мобільних додатків є:

- мінімум елементів. Непотрібно перевантажувати маленький простір дисплея мобільного телефону великою кількістю об'єктів. Ця вимога є головною для розробки будь-якого програмного забезпечення, а не тільки мобільного;
- управління телефонами з сенсорними екранами здійснюється за допомогою чуттєвого дотику. З погляду на те, що площа дотику пальця значно більше розмірів покажчика комп'ютерної миші, а також стілуса, інтерфейс не

повинен містити дрібних елементів. Вони погано помітні на невеликому екрані;

- розмір всіх написів повинен бути достатнім для того, щоб користувач міг прочитати їх з відстані не менше 30 см;
- важливі і часто використовувані елементи інтерфейсу повинні знаходитися в центрі екрану і мати достатній розмір для того, щоб виділятися серед інших;
- при перенесенні додатків з ПК на мобільну платформу можна обмежитися створенням зменшеної копії додатка.

Необхідно вдосконалити весь інтерфейс, прибрати зайві елементи, згрупувавши їх в схожі по функціоналу. При великій кількості об'єктів слід зробити додаткові «вікна», що змінюють один одного на дисплеї. На відміну від ПК, на мобільних платформах під вікнами розуміються елементи інтерфейсу, які займають весь простір екрану пристрою. Користувач здійснює переходи між такими вікнами за допомогою графічних елементів - навігаторів, або перетягуючи їх за допомогою пальця.

При проектуванні дизайну мобільного додатку може виникнути необхідність враховувати культурні особливості регіону, для якого воно призначене (читання справа наліво або правильний підбір колірної гами). У правильно спроектованому мобільному додатку повинні поєднуватися три основні властивості:

1. Зручність у використанні (інтуїтивний дизайн, об'єднання та використання всіх можливостей мобільного пристрою).

2. Мобільний додаток повинен бути цікавим користувачу. Саме тому найбільшою популярністю на сьогодні є мобільні ігри.

3. Корисність. Максимальний рейтинг мають ті додатки, які здатні бути дійсно потрібними.

Під час розробки мобільного додатку дотримано всіх вище наведені правила та спроектовано оптимальний на вигляд і для застосування інтуїтивно зрозумілий інтерфейс.

2.2.1 Платформа для розгортання додотку

Для розробки програмного продукту використовувалась мова Kotlin в Android Studio. Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS. Android Studio можна встановити на Windows, Mac і Linux. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного застосування в Google Play App Store. Переваги Kotlin полягає в простоті його мови, дозволяючи писати більш компактний код, але не втративши при цьому функціональності. При оголошенні змінних і параметрів типи даних вказуються після назви (роздільник - двокрапка). Крапка з комою як роздільник операторів так само необов'язкова, як в Scala і Groovy; в більшості випадків перекладу рядка досить, щоб компілятор зрозумів, що вираз закінчилося. Крім об'єктно-орієнтованого підходу, Kotlin також підтримує процедурний стиль з використанням функцій. Як і в мовах C / C ++ / D, точка входу в програму - функція "main".

Ключові можливості:

Повна сумісність з Java в обидві сторони (код на Java і Kotlin можна безболісно змішувати в одному проекті), автоматичний висновок типів змінних і функцій, анонімні функції дозволяють писати більш компактний код, зовнішні функції дозволяють розширювати інтерфейс існуючих класів, які не змінюючи їх, виразна система типів дозволяє виявляти багато помилок на етапі компіляції, а також

конструкції, що скорочують зайві повторення коду: властивості, повернуться до стандартних значень, мульти-привласнення, класи даних, автоматичне приведення типів і ін. Модуль додавання нових планувань і категорій призначений для тих випадків, якщо потрібно полегшити собі завдання - спочатку спокійно додати нові категорії і, створюючи новий список , вибирати з вже готові в базі даних категорії.

Для створення нової категорії пропонується вибрати зі списку існуючих - Spinner - або випадає виконує це завдання. Він надає швидкий спосіб вибору значення із запропонованого списку з варіантами. Оскільки список виводиться тільки при натисканні на спиннер, це економить місце на екрані вашого пристрою. У стані за замовчуванням спиннер відображає поточне значення. Якщо ж торкнутися компонента, то з'явиться меню, що випадає з усіма іншими доступними значеннями, з яких користувач може вибрати потрібну йому.

У Spinner завантажується набір даних, які виводяться на екран. Нам потрібно вивести набір даних складається з рядків, тому можна скористатися стандартним адаптером simple_spinner_item. Тому створимо стандартний адаптер наступною командою:

```
val adapter = ArrayAdapter<String> (this, android.R. layout. simple_spinner_item, l)
```

Оголошуємо змінну val, оскільки в процесі вона не буде змінюватися. Як dataset береться список рядків l - в ньому зберігаються всі імена категорій.

Методом add (index, name) створюємо в списку новий елемент з індексом, що збігається з номером категорії, що приймає значення імені категорії.

```
while (res5.moveToNext()) {
    l.add (res5.getString(0). toInt ()-1, res5.getString(1))
}
```

При натисканні на певну назву категорії, її назва повинна міститися в текстове поле. Для цього додамо обробник події на натискання елемента в Spinner.

```
val itemSelectedListener = object: OnItemSelectedListener {
        override fun onItemSelected (parent: AdapterView<*>?, view: View?, position: Int, id: Long) {
```

Обраний об'єкт знаходиться на позиції position. Тоді досить в нову змінну записати рядок з цієї позиції

```
            val item = parent?.getItemAtPosition(position) as String
```

NameCat -текстовий поле, в яке записується рядок з БД або введена з клавіатури. При додаванні з БД в тег запишемо відомості про позицію, таким чином збережемо дані про номер категорії.

```
            NameCat.setText(item)
            NameCat.tag = position
          }
        }
```

На сторінці додавання нового продукту є список, що розкривається SpinCat. Дамо його створений адаптер

```
SpinCat.adapter = adapter
```

Події на натискання об'єкта призначимо вище створений обробник події.

```
    SpinCat.onItemSelectedListener = itemSelectedListener
```

Призначимо підпис для списку:

```
     SpinCat.prompt = "Выбрать существующий"
```

Тепер з'являється проблема: потрібно відрізняти введену з клавіатури категорію від тієї, яка була введена з БД. Для цього при спробі додати продукт, шукаємо в БД категорію з тим же ім'ям, яка була записана в NameCat, якщо такий рядок знайдена, то змінно ро дамо номер цього рядка. Інакше створюємо нову категорію, і змінно ро дамо останній номер в таблиці. Таким чином ми не перевантажувати користувача купою вікон при додаванні нового продукту - все відбувається автоматично.

```
 var po = "error1"
```

```
        val lcp = db.allCat
        while (lcp.moveToNext()) {
            if (lcp.getString(1) == NameCat.text.toString() ) {
                po = lcp.getString(0)
                break
            }
        }
        if (po == "error1") {
            db.insertCat(NameCat.text.toString())
            val np = db.allCat
            np.moveToLast()
            po = np.getString(0)
        }
```

Створюється спеціальний адаптер для відображення списку:

В змінної myDataset зберігається список, який необхідно вивести в RecycleView.

```
class AdapterProd(var myDataset: ArrayList<BuyItem>) :
    RecyclerView.Adapter<AdapterProd.ViewHolder>()
```

Для переміщення даних в список використовується клас ViewHolder, в якому започатковано все текстові поля, які необхідно заповнити - відповідно для відображення категорії, потрібно використовувати їх назва, категорія і ціну.

```
  class ViewHolder(MyView: View?) : RecyclerView.ViewHolder(MyView)  {
      var NameProduct: TextView? = null
      var NameCategory: TextView? = null
      var NPrice: TextView? = null
      init {
```

Ініціалізувавши знаходимо текстові поля, в які потім запишуться дані і присвоюємо їх змінним.

```
            NameProduct = MyView?.findViewById(R.id.NameP)
            NameCategory = MyView?.findViewById(R.id.NameC)
            NPrice =  MyView?.findViewById(R.id.NamePr)
        }
    }
```

Для правильної роботи адаптера йому потрібно призначить дві обов'язкові функції - обробник події на створення і заповнення даних. Під час створення нового View, призначимо Holder відомості з раніше створеного XML-файла shop_prod. Таким чином він буде повторювати розташування об'єктів в ньому. Таким чином отримуємо «скелет» нового елемента списку -текстовий поля в тому положенні, де вони розписані в XML-файлі, але в масштабі однієї позиції

RecycleView

```
override fun onCreateViewHolder(parent: ViewGroup?,
                                viewType: Int): AdapterProd.ViewHolder {
        val textView = LayoutInflater.from(parent?.context)
                .inflate(R.layout.shop_prod, parent, false)
        return ViewHolder(textView)
    }
```

Для заповнення отримуємо елемент з нашого набору даних в позиції, на якому обробляється подія. І кожному об'єкту в Holder призначаємо значення з нашого набору даних.

```
    override fun onBindViewHolder(holder: ViewHolder?, position: Int) {
holder?.NameProduct?.text = myDataset[position].name
        holder?.NameCategory?.text = myDataset[position].kol
        holder?.NPrice?.text = myDataset[position].price
```

```
}
```

Призначимо створеному RecycleView з назвою ListProducts вид відображення і присвоюємо йому адаптер. kk - це список, який містить відомості про продукти

```
, kk.name, kk.cat, kk. price
ListProducts.layoutManager = LinearLayoutManager(this)
ListProducts.adapter = AdapterProd(kk)
```

Для більш точного аналізу ведеться облік часу створення списків і аналіз ведеться за обраним терміну. В Базі Даних відомості про час зберігаються в такий спосіб:

Створюємо нову статичну змінну dateFormat, в якій вказуємо, в якому вигляді буде зберігатися дата. Y - відображаються року, M- місяці, d- дні, H- годинник, m- хвилин, s - секунди. Решта символів поділяють дані про дати. У нових версіях Java необхідно надавати місцевий час. Так як додаток розраховане на місцевий час, встановимо змінну Local в дефолтний стан

```
val dateFormat = SimpleDateFormat("yyyy/MM/dd HH:mm:ss", Locale.getDefault())
```

В змінної date зберігається час в даний момент часу. Її відображення в строковому вигляді незручно для сприйняття, для зручності було створено dateFormat, який дану змінну представить в іншому вигляді.

```
val date = Date ()
```

Таким чином в стовпець dater поміщається відформатована в спеціальному форматі справжня дата, відображена в строковому вигляді. Для використання цього значення, потрібно звернутися до відповідної позиції в рядку

```
cv.put("dater", dateFormat.format(date).trim())
```

2.2.2 Мова програмування для розробки додатку

Мобільні пристрої (смартфони, планшети) стали найбільш перспективним каналом для спілкування та засобом оптимізації бізнес - процесів. Другими словами, приблизно дев'ять з десяти пристроїв використовують саме ОС Андроід,

тому актуальним для розробника програмного забезпечення для Android стає обрання інструментів. Ринок мобільних пристроїв це самий швидко - розвиваючий сегмент ринку. Смартфони виконують широкий спектр комп'ютерних завдань загального профілю. У нашому світі є велика кількість мов програмування, що ставить нас перед складною альтернативою - яку ж мову обрати? Можна скористатись різними рейтингами, статистикою, або не задумуватись над тим яку обрати та стати на бік більшості. Слід дивитись більш прагматично і звернути увагу на нові мови, які розвиваються, і що в свою чергу свідчить про те що вони враховують всі недоліки та переваги своїх попередників. Мій вибір зупиняється на такій мові як Kotlin. Головні особливості Kotlin - проста і повна сумісність з Java. Мова Kotlin створена компанією, яка робить дуже багато продуктів мовою Java і яка добре розбирається в сучасних інструментах розробки. Платформа Java - це перш за все екосистема: крім «офіційних» продуктів компанії Oracle, до неї входить безліч проектів з відкритим кодом: бібліотек та програмних каркасів різного профілю, на базі яких будується величезна кількість додатків. Тому для мови, дуже важлива сумісність з наявним кодом, який написаний мовою Java. При цьому необхідно, щоб наявні проекти могли переходити на нову мову поступово, тобто не тільки код мовою Kotlin повинен легко викликати код мовою Java, але і навпаки.

Мова Kotlin – сучасна за статистикою типова об’єктно-орієнтована мова програмування, що компілюється у Java і JavaScript. Kotlin можна створювати в JavaScript або в Native для запуску на iOS платформі. Перевагами мови Kotlin є її бурхливий розвиток, легкий перехід з Java на Kotlin – потрібно просто встановіти плагін Kotlin та їх сумісність, наявність extension functions для розробки чистих API, наявність null в системі типів. Kotlin стисла, що зменшує кількість помилок. Але існують і певні недоліки:

- Kotlin має меншу швидкість компілювання;

- середовища розробки (наприклад, Android Studio) працює повільніше із Kotlin. Мовою Kotlin писати простіше, ніж на Java, і при цьому можнавикористовувати код, написаний на Java і код на Kotlin в одному проекті, є можливість підключати бібліотеки, які написані на Java, з коду на Kotlin.
- При повній сумісності з Java, мова Kotlin надає додаткові можливості, що спрощують повсякденну роботу програміста і підвищують продуктивність. [3]

Перевагами Kotlin є:

- компактність коду. Швидко писати, легко читати, менше помилок;
- вбудований захист від помилок, насамперед – можливість створення порожніх об'єктів, вимога до програміста явно вказувати, що якась змінна може приймати значення NULL;
- легкість в освоєнні. Java-програмісту не складе труднощів перейти на Kotlin.

Для зручності написання коду під ОС Android, компанією JetBrains створено бібліотека "Kotlin Android Extensions". Що дозволяє використовувати в коді ідентифікатор, який декларується у XML файлі. Структура байт - коду Kotlin майже одинакова структурі Java, що робить додатки так же само швидкими. Але, Kotlin може підтримувати вбудовані функції, що дозволяють коду, який містить вирази, працювати швидше, ніж той, який написано на Java. Клас, що складається з 50 рядків коду при використанні Java, може бути написаний тільки одним рядком Kotlin. Великим плюсом є те, що Kotlin повністю підтримується IDE Android Studio, що є основною середою розробки під ОС Android. Тому перехід до розробки з використанням мови Kotlin позбавлений проблем. Як основну мову для написання Android проектів можна використовувати мову програмування Kotlin. Вона поєднує в собі лаконічність, виразність та продуктивність.

Вище наведені переваги та недоліки демонструють ефективність використання Kotlin, яка вимагає меншої кількості ліній коду, що робить її

використання безпечнішим в порівнянні із Java. Завдяки цьому покращується читабельність коду, ефективність його обробки, знижується ймовірність виникнення помилок при написанні коду ще на етапі компіляції.

Користувач додатку має отримати максимально зручне у повсякденному використанні мобільне розширення, яке було б водночас максимально розрахунковим та швидким у користуванні. Сервіс, що надаватиме інформацію стосовно фінансового планування, буде працювати під операційною системою Android, яка лишається найпоширенішою ОС для смартфонів.Мобільний додаток має надавати інформаційну базу, що відображатиметься у вигляді новин щодо організації фінансів на подорожі та поїздки.

1. Користувачам надаватиме змогу вчасно нагадати:
2. Про необхідність відкласти необхідну суму на подорож та поїздку;
3. Розподілити витрати ;

Можливість збереження інформації про подорожі та витрати на них.

Додаток "ФІНАНСОВЕ ПЛАНУВАННЯ" дозволяє по-новому поглянути на фінансові можливості. Оптимізувати багато елементів фінансування, що дає додаткові можливості для заощадження витрат і , як наслідок, накопичування коштів в кількісному сенсі, додаткове мотивування до заощадження. Ресурс об'єднує всі доходи в єдину інформаційно-аналітичну систему, яка дозволяє системно і централізовано керувати коштами.

На основі аналізу підходів які існують, до проектування агентно-орієнтованих програмних систем та специфікацій стандартів багатоагентних систем, виділені підходи для побудови архітектури агентної платформи та відповідні вимоги. Модель агентної платформи складається з наступних компонентів:

1. агент – це обчислювальний процес, який реалізує автономну комунікаційну функціональність програми;
2. фасилітатор каталогів, який надає по- слуги жовтих сторінок іншим агентам;

3. система управління агентом - обов’язковий компонент, який здійснює контроль за доступом до агентної платфор-ми та його використанням
4. 4) служба транспортування повідомлень - це метод зв’язку за замовчуванням між агентами на різних агентних платформах.

Функціонал програмного агенту показано на рисунку 2.1 у вигляді UML-діаграми прецедентів [7]. Для цієї системи рекомендовано класи користувача. Адміністратор агентної платформи – користувач, що відповідає за працездатність та налаштування системи. Розробник агентної платформи – відповідає за розробку компонентів для вирішення задач необхідного домену. Згідно із стандартом FIPA основна вимога агентної системи – це забезпечення здатності до перенесення програмного коду на різні платформи. Згідно із цією вимогою, було прийнято рішення використовувати JVM-середовище та мову програмування Kotlin. Здатність агентів реагувати динамічно на несприятливі ситуації і події робить простішим розробку стійких розподілених систем. Наприклад, якщо вузол закінчує роботу, то усі агенти, що виконуються на цій машині, попереджаються і їм надається час, щоб перейти на інший вузол і продовжити роботу зі збереженням попереднього стану. Підтримка роз’єднаного функціонування та розподілена парадигма виключають деякі сценарії відмови працездатності і дозволяють пропонувати відмовостійкі характеристики.

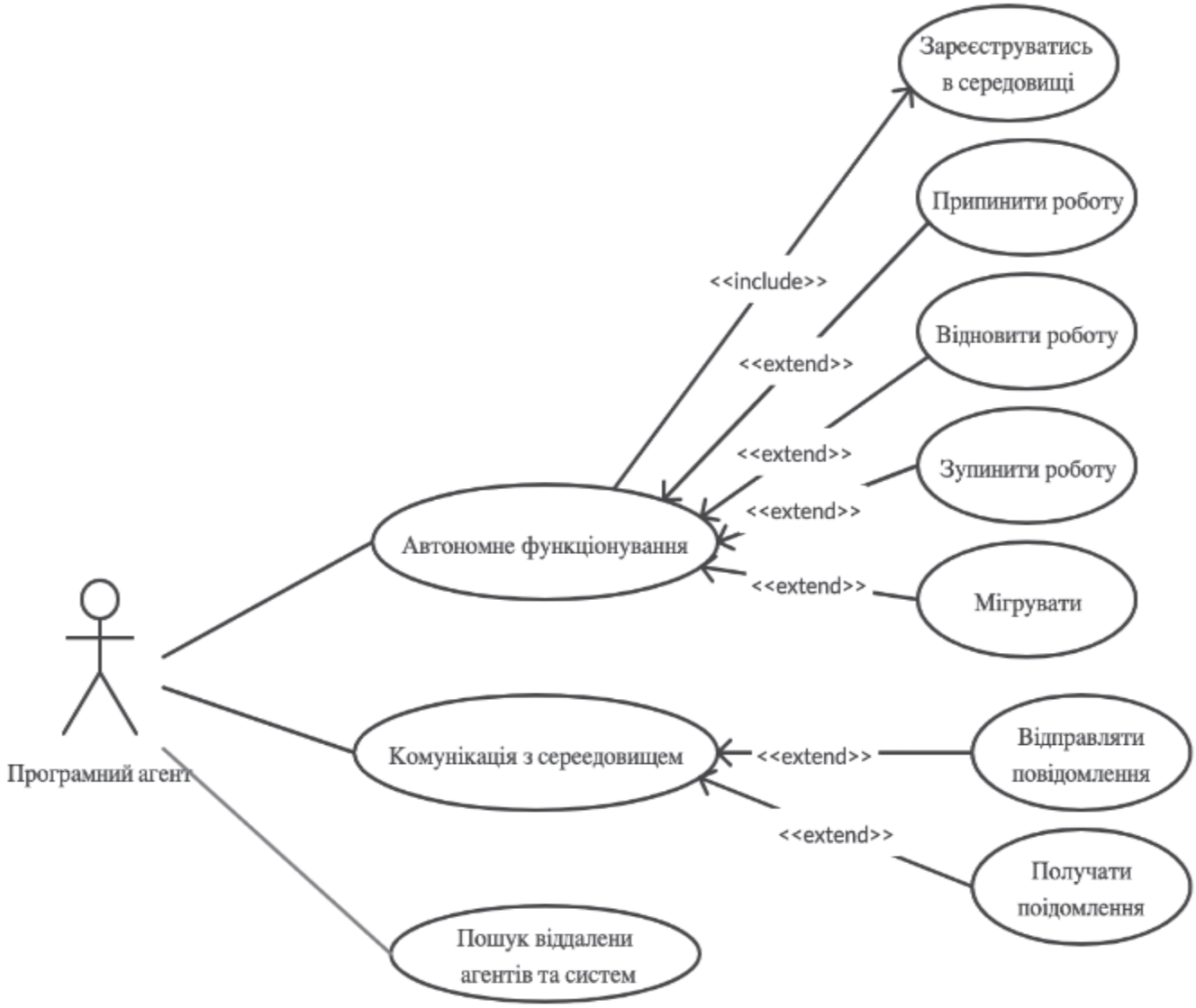

Рис.2.1. Діаграма прецедентів програмного агенту[7]

Для проектування фукнціональних та нефункціональних вимог, розглянуто аналоги та сформулювано наступні вимоги до прототипу багатоагентної системи:

1. Забезпечення здатності до перенесення коду на різні платформи.
2. Забезпечення роботи агентів у гетерогенному комп’ютерному середовищі.
3. Підтримка мережевої взаємодії.
4. Багатопоточна обробка.
5. Безпека.

На рисунку 2.2 зображені функціональні та нефункціональні вимоги до програмного забезпечення у вигляді діаграми [7] вимог в нотації UML. Дані вимоги певною мірою відображають вимоги стандарту FIPA. Вимоги до зовнішнього інтерфейсу включають: JVM-середовище та комплект розробника застосунків (JDK) на мові Java, який містить компілятор Java, стандартні бібліотеки класів Java не менш ніж 8 версії. Крім цього, необхідні бібліотеки розробника застосунків для Kotlin (Kotlin JDK) та середовищем виконання (Kotlin JRE). Для організації запуску декількох агентів одночасно, а також їх асинхронного керування використовується пакет розробки співпрограм Kotlin - coroutines.

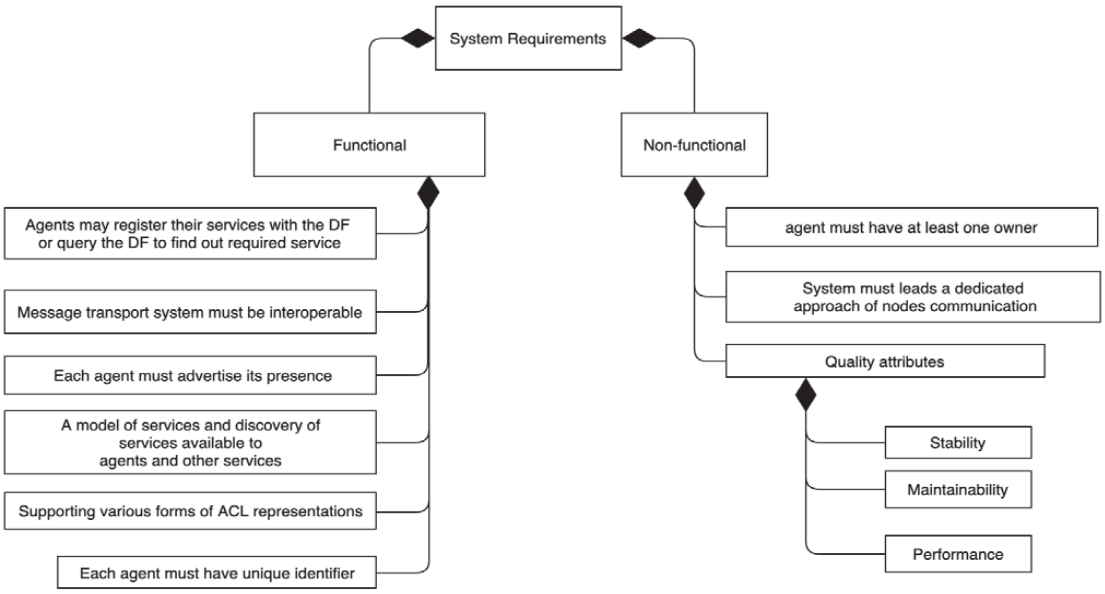

Рис.2. 2. Діаграма вимог ПС[7]

Основні компоненти та процес ініціалізації агентів зображено у вигляді UML-діаграми компонентів. Архітектура побудованої системи чітка і чиста. Отже, на основі проведеного аналізу існуючих підходів конструювання багатоагентних програмних систем і програмних компонентів агентної платформи розроблено прототип агентної платформи на основі використання формальних методів

проектування багатониткових систем з використанням мови програмування Kotlin. Відповідно до визначених вимог, для оцінки прототипу агентної платформи пропонується оцінити такі параметри, як обсяг виділеної оперативної пам’яті, обсяг навантаження процесора та кількість виділених потоків. Задля цього сформовано умовний приклад, який детально розглянуто у наступному розділі.

2.3 Проектування бази даних додатку

База даних – організована структура, призначена зберігати інформації даних та методів, за допомогою яких відбувається взаємодія з іншими програмно-апаратними…комплексами.

Системи управління базами даних (СУБД) – комплекс програмних засобів, призначених для створення структури, наповнення її змістом, редагування змісту та візуалізації інформації. Під візуалізацією інформації бази розуміється відбір даних, що відображаються відповідно до заданого критерію, їх упорядкування, оформлення і подальша видача на пристрій виведення або передавання по каналах зв’язку. На етапи поділяється проектування баз даних, кожен з яких відповідає за виконання певних дій.

Розробка інформаційно - логічної моделі даних предметної області, вона має базу на опису предметної області, одержаної в результаті її обстеження. Спершу визначають склад і структуру даних предметної області, що мають міститись в базі даних і забезпечувати виконання задач та застосувань користувача. Ці дані мають форму реквізитів, і містяться в різних документах - джерелах завантажених базах даних. Аналіз виявлених даних дозволяє визначити функціональні залежності реквізитів, які використовують для виділення інформаційних об'єктів, та відповідають вимогам нормалізації даних. Подальше визначення структурних зв'язків між об'єктами дозволяє побудувати інформаційно-логічну модель[10].

Розробка таблиць бази даних. Здійснюється засобами СУБД, та узгоджується із замовниками. Структура таблиць бази даних складається за допомогою засобів опису (конструювання) таблиць у СУБД із цілковитою відповідністю інформаційним об'єктам. Окрім таблиць, проектувальники розробляють і інші об'єкти бази даних, які призначені для автоматизації работи з базою, та обмеження функціональних можливостей работи з базою (безпека бази даних). Після формування структури таблиць база даних може наповнюватись даними з документів - джерел. Проектувальники, не наповнюють базу конкретними даними (оскільки замовник може вважати дані конфіденційними і не надавати стороннім особам). Винятком є експериментальне наповнення модельними даними на етапі відлагодження об'єктів бази.[10]

Після визначення основної частини даних, що замовник використовує, розпочинється розробка структури бази, структури її основних таблиць.

1. Починається робота з визначення генерального переліку полів, що може нараховувати десятки та сотні позицій.

2. До типу даних, що розміщуються в кожному полі, визначено тип кожного поля.

3. Розподілено поля генерального списку по базових таблицях. Розподіл здійснюють за функціональною ознакою. Метою є забезпечення одноразового введення даних в одну таблицю по можливості в рамках одного підрозділу, або на одному робочому місці. Потім розподіл полів здійснюють нормалізацію даних з метою видалення повторів даних у таблицях бази даних.

У кожної таблиці визначають ключове поле. Ним вибирають те поле, де дані повторюватись не можуть. Якщо у таблиці зовсім немає полів, які можна використовувати як ключові, то можна добавити додаткове поле на подобі лічильника, він не може містити дані, що повторяються.

Потім визначають зв'язки між таблицями (схеми даних). Оці зв'язки які є між таблицями організують на основі спільного поля, тоді в одній із таблиць воно обов'язково має бути ключовим. Отже на стороні "один" має бути ключове поле, яке не повторяється, значення на стороні "багато" мають можливість повторюватися.

"Паперовий" – це робота над технічними пропозиціями і вона закінчується розробкою схеми даних. Схеми потрібно узгоджувати із замовником, після чого розпочинати безпосереднє створення бази даних. Отже на цьому етапі завершується попереднє проектування бази даних, і на наступному етапі починається її безпосередня розробка (впровадження).

2.4 Висновки розділу 2.

В другому розділі було досліджено технологію архітектури програмного забезпечення на основі Android. Розглянуто процес проектування структури бази даних. Розглянуто етапи інформаційної моделі даних. Розібрали та вибрали мову для написання нашої програми. Навели приклад розробки програми для фінансового планування подорожей та поїздок для мобільної платформи. Зробивши аналіз програмного забезпечення, можемо приступити до опису та тестування продукту.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ТА ЗБЕРІГАННЯ ДАНИХ ДЛЯ ПЕРСОНАЛЬНОГО ВИКОРИСТАННЯ

3.1 Розробка програмного продукту.

Для розробки програмного продукту потрібно відкрити Android Studio.
У діалозі Welcome to Android Studio вибрано Start a new Android Studio project.
У діалозі New Project, дано ім'я нашому додатку. Ім'я домену прийнято за замовчуванням. Для підтримки мови Kotlin відзначено чекбокс Include Kotlin support. Рис.3.1

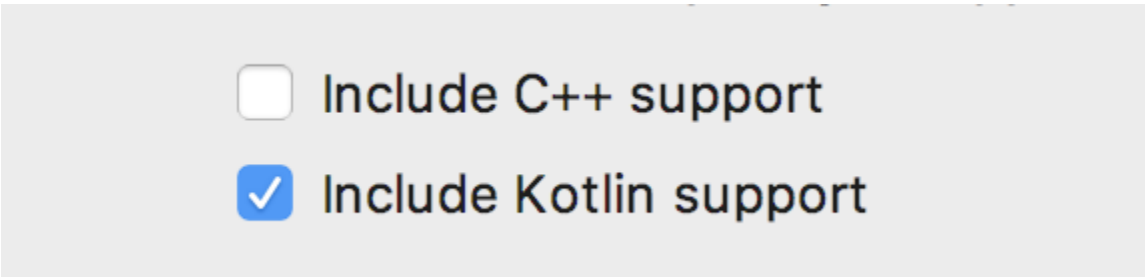

Рисунок 3.1 Чекбокс Include Kotlin support.

Місцезнаходження проекту залишено за замовчуванням. Зазвичай проекти зберігаються в папці з ім'ям AndroidStudioProjects в документах користувача. Тут можуть виникнути проблеми, якщо ім'я користувача вказано українськими буквами, тому що середовище розробки не сприймає кирилицю в шляху до файлів. В такому випадку побачемо попередження. Збережіть проект в іншому місці.
Натиснуто Next. У діалозі Target Android Devices вибрано тип платформи Phone and Tablet і мінімально підтримувану версію Android API 17 або вище. Підтримка більш старих версій нам поки не потрібна, оскільки відбивається на доступному функціоналі інструментів розробки. Натиснуто Next.
У діалозі додавання активують потрібно вибрати шаблон попередньо встановлених вікон програми і компонентів інтерфейсу. Вибераємо Empty Activity. Натискаємо Next. У діалозі Configure Activity залишено все за умовчанням. Натискаємо Finish.

Після цих кроків, Android Studio:

1. Створено папку для проекту Android Studio на диску, в місці розташування, вказаному при створенні проекту.
2. Далі відбувається побудова проекту (це може зайняти кілька хвилин). Android Studio використовує Gradle як системи побудови. Іноді система збирання видає повідомлення і попередження, наприклад зараз потрібно доустановити відсутній компонент SDK.
3. Відкрилося вікно редактора коду, що відображає поточний проект.

Вікно Android Studio має такий вигляд. Рис.3.2

Вікно Android Studio

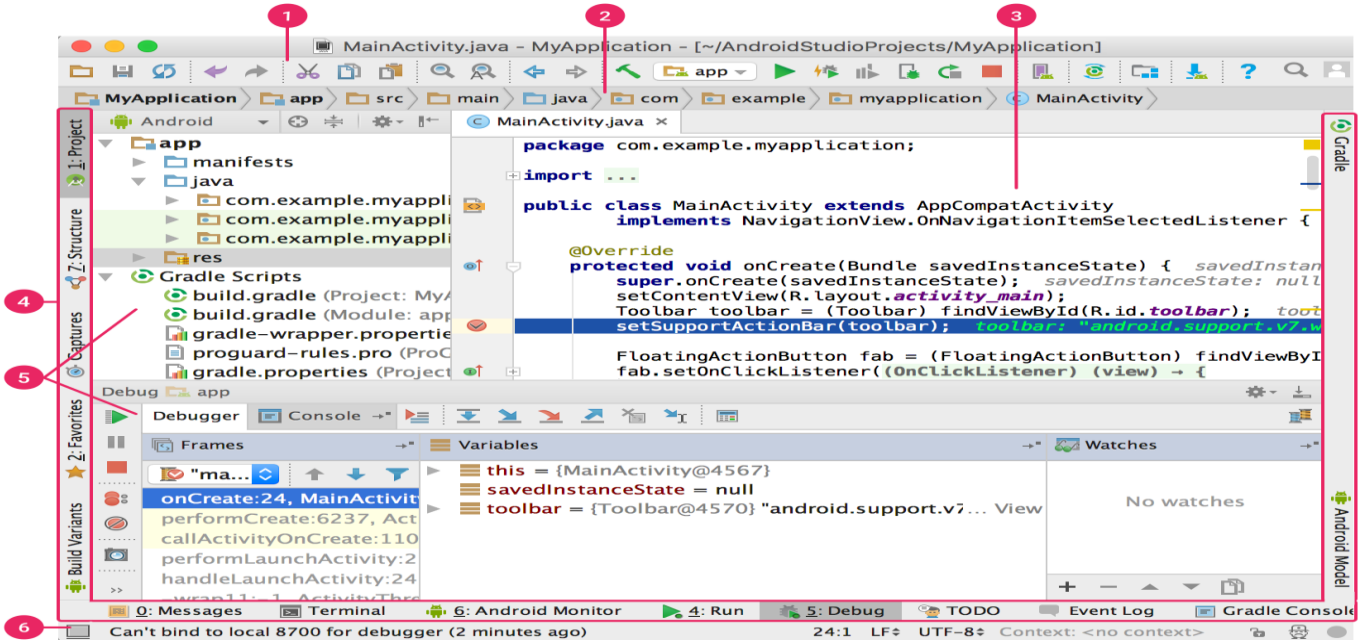

Рис. 3.2 Інтерфейс Android Studio

Розглянено інтерфейс Android Studio:

1. Панель Toolbar надає швидкий доступ до самих затребуваних командам, як запуск програми або відкриття SDK менеджера.

2. Панель Navigation bar допомагає переміщатися по проекту і відкривати файли для редагування. Він забезпечує більш компактний вигляд структури, видимої у вікні Project.
3. Вікно редактора дозволяє створювати і редагувати код. Залежно від типу відкритого файлу вид редактора може змінитися. Наприклад, при перегляді файлу макета редактор відображає редактор макета.
4. Панель Tool buttons розташована по периметру вікна IDE і містить кнопки, що дозволяють розгортати або згортати окремі вікна інструментів.
5. Вікна інструментів надають доступ до певних завдань, таким як управління проектами, пошук, управління версіями і багато іншого. Їх можна розгорнути і згорнути. Далі ми розглянемо їх більш детально.
6. Status bar відображає стану проекту і самої IDE, а також різні попередження або повідомлення.

Головне вікно можна впорядкувати, щоб надати більше місця на екрані, приховуючи або переміщаючи панелі інструментів і вікна інструментів. Для доступу до більшості функцій IDE також можна використовувати поєднання клавіш. У будь-який момент можна виконати пошук по вихідного коду, баз даних, командам, елементам користувальницького інтерфейсу і т. Д., Двічі натиснувши клавішу Shift або клацнувши збільшувальне скло у верхньому правому куті вікна Android Studio. Це може бути дуже корисно, якщо, наприклад, намагаємось знайти конкретну команду IDE, яку забули, як запустити.Рис.3.3

Вікно структури проекту.

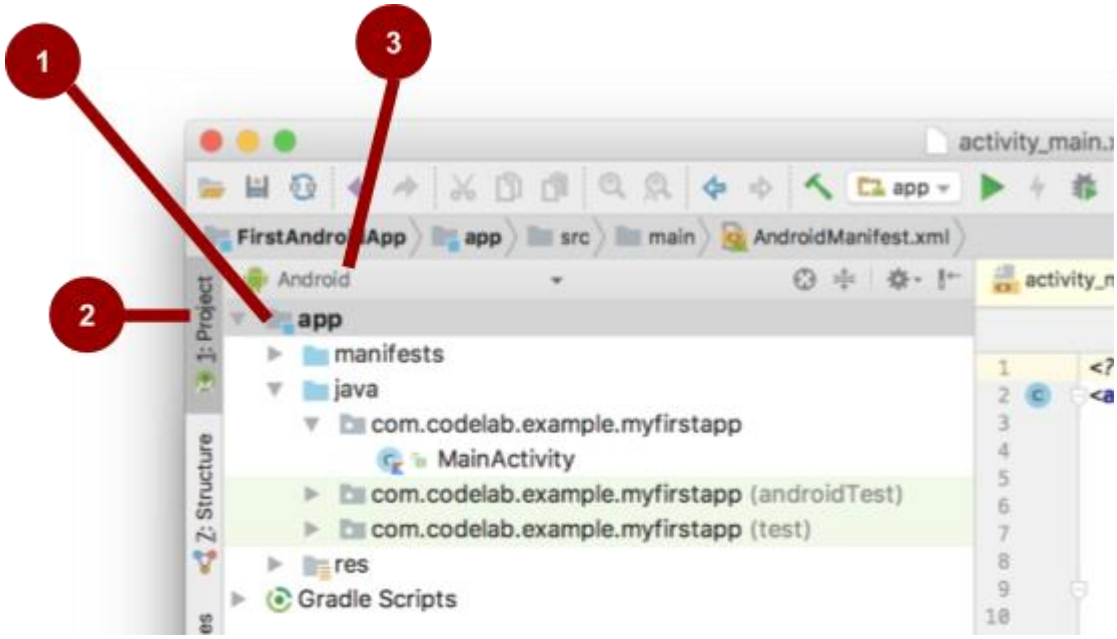

Рис. 3.3 Вікно структури проекту Android Studio

Можна подивитися ієрархію файлів програми декількома способами.

1. Вікно структури проекту (1) з відображенням дерева файлів. (Див. 1 на Рис.3.3)
2. Кнопка Project (2) приховує і відображає вікно структури проекту. Вмикаємо пункт меню View> Tool Buttons щоб побачити цю кнопку.
3. Поточний вибір виду проекту Project> Android. Натискаємо випадає Android (3) для перегляду інших доступних видів проекту.

У вигляді відображення Project> Project бачимо повну структуру папок проекту, як вона виглядає на диску. Рис. 3.4.Тут багато файлів і папок і новачкові тут можна заплутатися. Головне, на що слід звернути увагу: проект має модульну структуру і основний модуль нашого застосування – app.

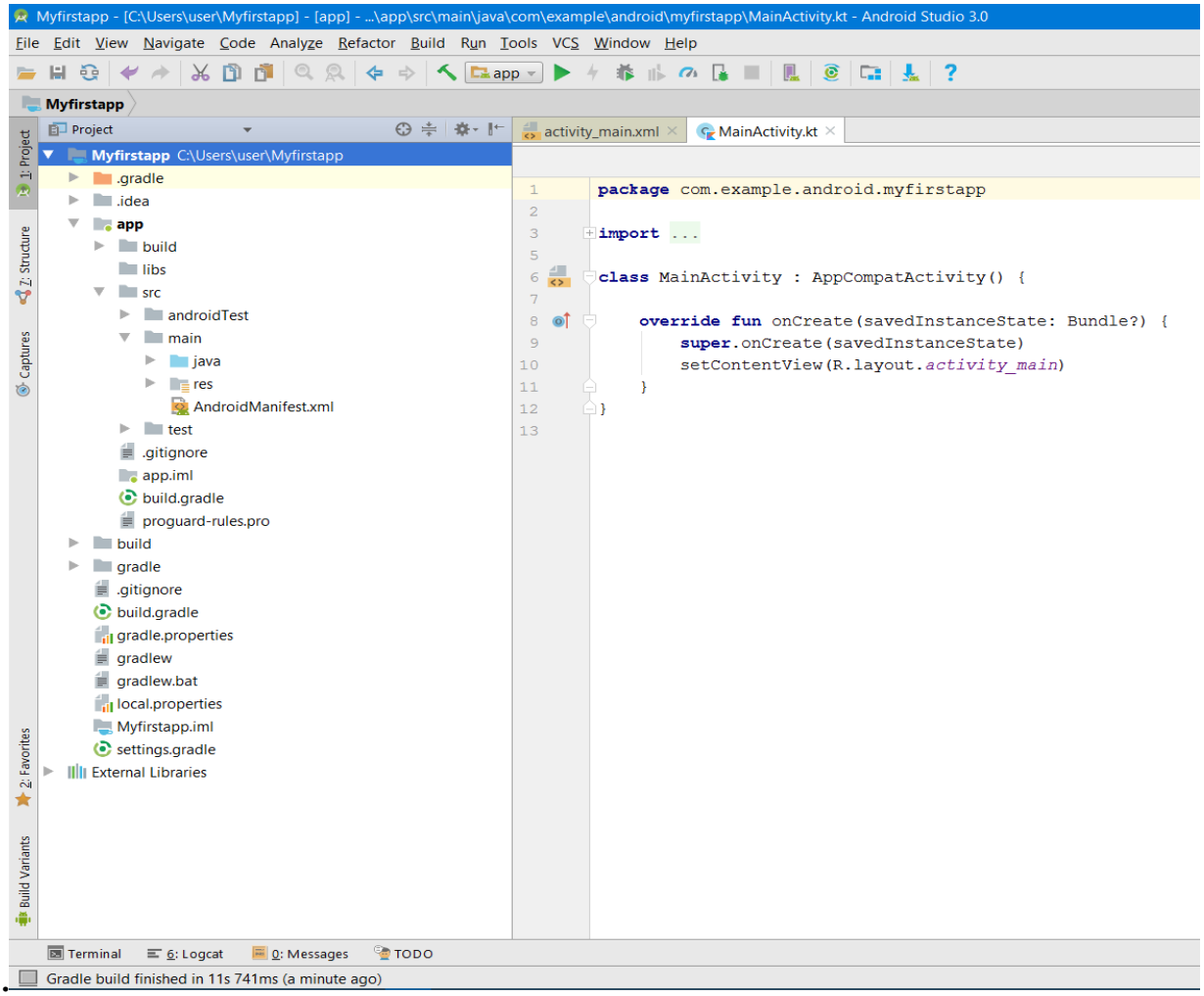

Рис.3.4 Вікно структури проекту Android Studio в вигляді Project> Project

Якщо розкрити папку app, можна побачити такі папки:

- build - тут файли, створювані системою в процесі компіляції, краще нічого там не змінювати
- libs - папка для сторонніх бібліотек, що підключаються до проекту
- src - папка для вихідного коду і ресурсів
- всередині src знаходиться папка main - це основна робоча папка, з якої ми будемо мати справу

- всередині main знаходиться папки java і res - це папки для коду та ресурсів, розглянемо їх пізніше

Також в папці модуля app є файл системи збирання build.gradle, розглянено його пізніше. Якщо переключитися на вигляд проекту Project> Android, побачите тільки папку модуля app зі спрощеною структурою проекту. Це уявлення зручніше, оскільки тут відфільтровані тільки ті файли, з якими ми безпосередньо будемо працювати. Але слід пам'ятати, що в такому вигляді структура папок проекту не відповідає тій, що зберігається на диску. Рис.3.5.Наприклад, ви не знайдете на диску в папці проекту папку manifests. А в дереві папок в такому режимі не відображаються папки src і main.

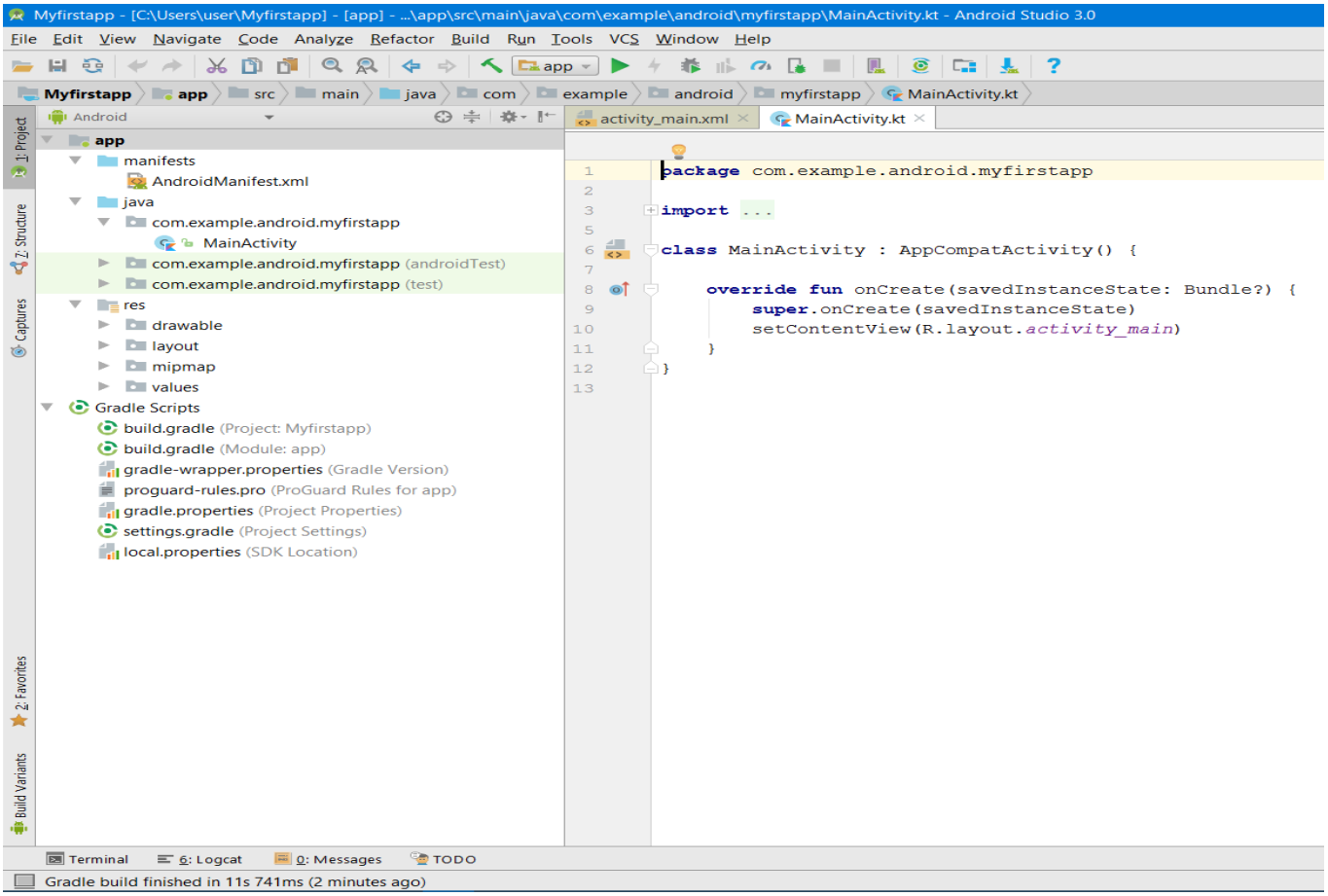

Рис.3.5 Вікно структури проекту Android Studio в вигляді Project> Android

У вигляді Project> Android бачимо в папці app такі папки: manifests, java, і res.

1. Розкриваємо папку manifests. Ця папка містить файл AndroidManifest.xml. Цей файл описує всі компоненти програми Android і зчитується системою середовища виконання Android під час запуску програми.
2. Розкриваємо папку java. Всі файли коду на мові Котлін і Java організовані тут. Папка java містить три папки: com.example.android.myfirstapp (або вказане ім'я домену): Ця папка містить файли вихідного коду Котлін і Java для нашого застосування. З цією папкою будемо працювати більшу частину часу, оскільки це головний пакет проекту.

com.example.android.myfirstapp (androidTest): ця папка для розміщення інструментальних тестів.

com.example.android.myfirstapp (test): ця папка для розміщення модульних тестів

3. Розкриваємо папку res. Ця папка містить всі ресурси для нашого застосування, включаючи зображення, файли макетів екранів, строкові ресурси, значки та іконки, кольори і стилі оформлення. Вона включає такі папки: drawable: Всі зображення для вашої програми будуть збережені в цій папке.layout: Ця папка містить файли макета для активують - екранів додатка. В даний час додаток має одне активують з файлом макета activity_main.xml.mipmap: Ця папка містить значки запуску додатка - це ті значки, які відображаються на андроїд-пристрої після установки пріложенія.values: Містить ресурси, такі як рядки і кольору, що використовуються в додатку.

Також у вікні структури проекту є папка Gradle Scripts. Розкриємо її. У верхній частині списку є два файли з однаковими іменами build.gradle. Це файли збірки для системи Gradle, яка використовується для компіляції, побудови і упаковки додатків і бібліотек. У дужках після імені файлів вказана приналежність файлу модулю App або всього проекту. Файл зборки рівня проекту містить настройки для всього проекту, а файл збірки рівня модуля містить настройки для модуля. Найчастіше

будемо працювати саме з файлом збірки рівня модуля. Він містить такі основні секції, як:

- android {...}, де вказані версії інструментів розробки, мінімальна підтримувана версія API, ідентифікатор і версія додатка для Google Play, і інші параметри
- dependencies {...}, яка містить список бібліотек, що підключаються до проекту. Можуть бути підключені як локальні (поміщені в папку libs), так і зберігаються віддалено бібліотеки.

3.2 Тестування ефективності роботи.

Найбільш очевидний спосіб перевірити роботу програми в процесі розробки - встановити і запустити його на андроїд-пристрої.

Для запуску програми з середовища розробки Android Studio можемо використовувати 2 способи:

- підключений до ПК смартфон або планшет під управлінням системи Android
- емулятор Android

Обидва ці способи мають свої переваги і недоліки. Для початківця розробника підійде будь-який спосіб. Що таке андроїд-смартфон, знаємо , раз зацікавилися темою розробки додатків. А що таке емулятор Android? Це запуск системи Android на ПК в окремій віртуальній машині.

Існує багато емуляторів Android, наприклад BlueStacks, Droid4X, Nox APP Player, Genymotion, і інші. Більшість з них розроблені для ігор і не дуже підходять для тестування додатків, крім Genymotion. В Android Studio є власний емулятор, який може запускати образи віртуальних пристроїв з різними характеристиками, таких як смартфони Nexus і Pixel, а також планшети найбільш поширених типів.

В Android Studio є утиліта Android Virtual Device (AVD) manager для створення віртуального пристрою (також відомого як емулятор), що моделює конфігурацію певного типу андроїд-пристрої.

Першим кроком є створення конфігурації, яка описує віртуальний пристрій.

1. У головному меню Android Studio, вибераємо Tools> Android> AVD Manager, або натискаємо іконку AVD Manager в тулбаре.Рис.3.6.
2. Натискаємо кнопку + Create Virtual Device. (Якщо віртуальний пристрій створено раніше, у вікні відображаються всі існуючі пристрої і кнопка + Create Virtual Device знаходиться внизу.) У діалоговому вікні Select Hardware відображається список попередньо налаштованих типів апаратних пристроїв.
3. Вибераємо пристрій, наприклад Nexus 5, і натисніть Next.
4. У вікні System Image, на вкладці Recommended, вибераємо потрібну версію.
5. Якщо посилання Download відображається поруч з версією, значить вона ще не встановлена і потрібно завантажити її. При необхідності натискаємо посилання, щоб почати завантаження, і натискаємо кнопку Next після закінчення завантаження.
6. У наступному діалоговому вікні приймаємо значення за замовчуванням і натиснуто кнопку Finish.
7. Якщо вікно Your Virtual Devices AVD Manager ще відкрито, закриваємо його.

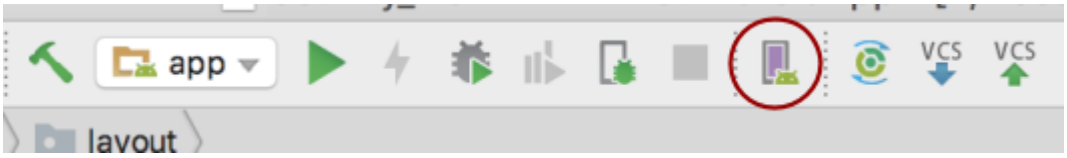

Рис.3.6

На комп'ютері встановлена операційна система Windows, а процесор AMD, то бачимо попередження в вікні вибору обладнання для віртуального пристрою. Справа в тому, що образи віртуальних пристроїв з типом x86 не працюють на зв'язці

Windows + AMD. Причому на зв'язці Linux + AMD таких проблем немає. Процесори Intel підтримують образи x86 на будь-якій операційній системі. Для Windows + AMD ми можемо використовувати образи типу arm, але вони набагато повільніші. Виходом буде використання реального пристрою - андроїд-смартфона або планшета, підключеного до комп'ютера.

Іншою поширеною проблемою є відключена віртуалізація процесора. Включити віртуалізацію можна в биосі, в інтернеті можна знайти багато інструкцій, як це робиться.Також потрібно в SDK Manager на вкладці SDK Tools встановити Intel x86 Emulator Accelerator для кращої продуктивності і швидкості роботи. Тільки для процесорів Intel. Якщо у вікні вибору системного способу немає активних образів для установки, завантажуємо образ необхідної версії Android в SDK менеджері. Відкриваємо вікно SDK Manager і на вкладці SDK Platforms справа внизу встановлюємо прапорець Show Package Details для відображення всіх компонентів платформи (див. Рис.3.7). Відзначаємо для завантаження потрібний файл System Image. Після завантаження обраний образ буде доступний у вікні AVD Manager.

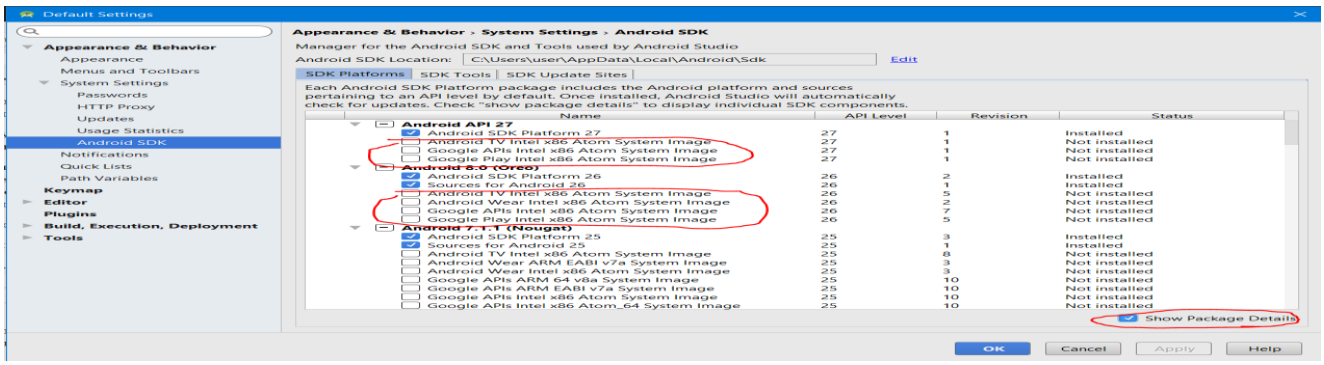

Рис. 3.7 Встановити прапорець Show Package Details для відображення всіх компонентів платформи

Запуск програми на емуляторі:

1. В Android Studio, вибераємо у головному меню команду Run> Run app або натискаємо іконку Run в панелі інструментів.

2. У вікні Select Deployment Target, під Available Virtual Devices, вибераємо щойно налаштований віртуальний пристрій і натискаємо OK.

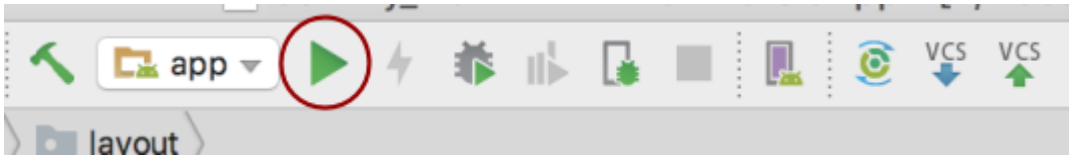

Рис. 3.8

Емулятор запускається і завантажується як фізичний пристрій. Залежно від швидкості комп'ютера це може зайняти деякий час. Можна подивитися в маленьку горизонтальну рядок стану в самому низу вікна Android Studio, щоб побачити хід виконання.

Після того як додаток буде побудовано і емулятор буде готовий, середа Android Studio встановить додаток в емулятор і запустить його. Побачимо наш додаток, як показано на наступному рисунку. Рис.3.9.

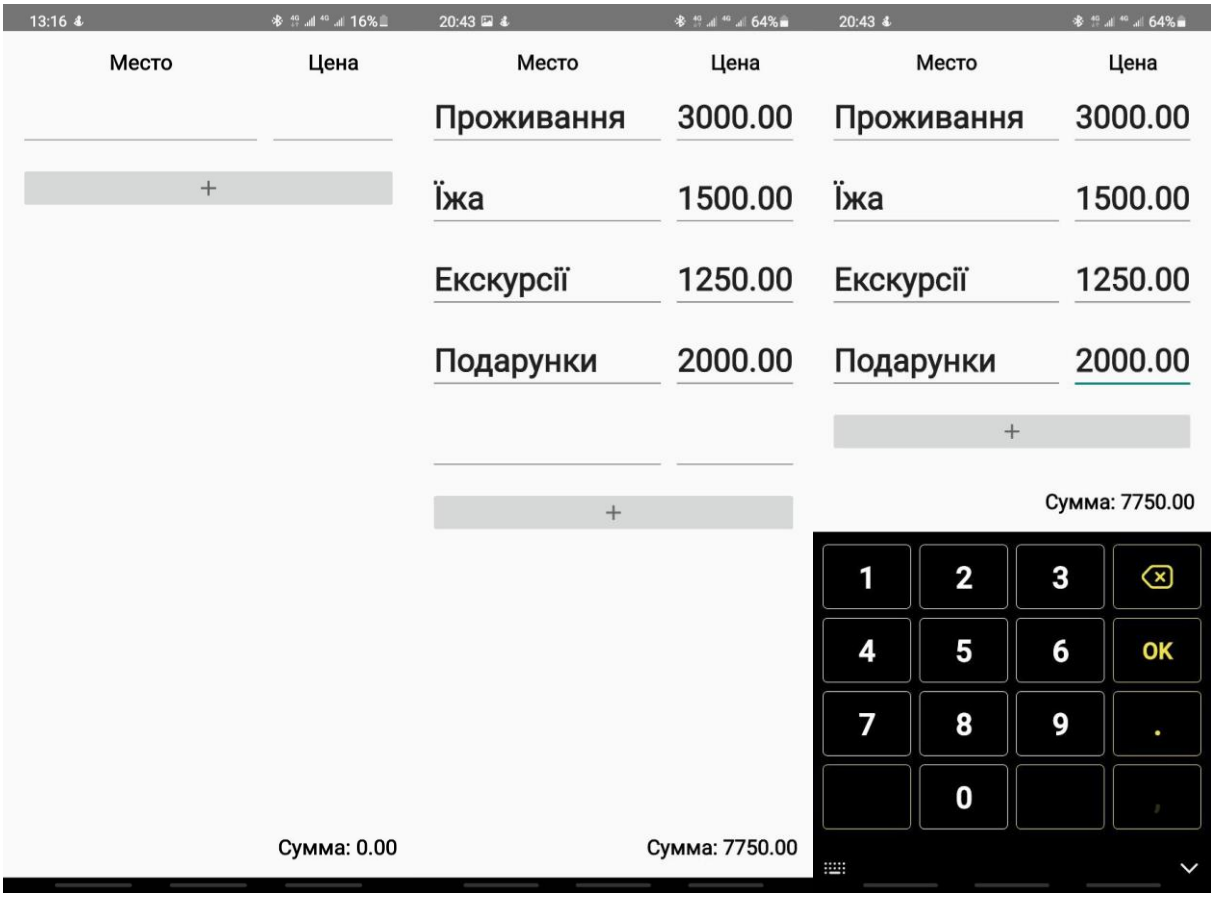

Рис. 3.9 Вікна додатку

3.3 Висновки розділу 3

В третьому розділі нашої роботи розглянуто послідовну розробку програмного продукту для Android на мові Kotlin. Розлянуто вікна Android Studio. Тестування підтвердило ефективність та правильність функціонування розробленого автоматизованої системи фінансового планування.Програмний продукт є легким у використанні, має привабливий та інтуїтивно зрозумілий інтерфейс. Додаток дозволяє заощаджувати час користувача за рахунок мінімального введення даних. Описано тестування та послідовний запуск програми яку створили. І вкінці відкрито вікна створеної програми. Отже, можемо зробити загальні висновки написаної роботи.

ВИСНОВКИ

В результаті проведеного вивчення питання стану програмного забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи була поставлена задача створити своє аналогічне програмне забезпечення з використанням технології. На мові Kotlin для операційної системи Android з використанням системи розробки Android Studio. Це дало можливість створити програмне забезпечення мобільного типу. Під час практики був проведений аналіз існуючих систем, вивчення їх функціональних можливостей та методів реалізації. В результаті цього аналізу було виявлено, що існуючі системи не відповідають вимогам, а також здебільшого мають складну структуру. В процесі проходження практики був спроектований та розроблений програмний продукт, що повністю задовольняє вимоги. Було досліджено методи і засоби програмної реалізації продукту. В результаті чого було обрано створення додатку для мобільної платформи “Android” на основі об’єктно орієнтованої парадигми програмування. Додаток розроблений на мові “Kotlin” з використанням платформи “Android”. Це дає змогу підвищити гнучкість та зручність системи в процесі розробки а також у процесі використання. Розроблена система було випробувана та протестована і задовольняє всім програмним вимогам. Додаток був розроблений для людей, які прагнуть мандрувати, та планувати свої витрати.

Отже, в результаті практики було розроблено мобільний додаток, що задовольняє всім вимогам, Також було отримано нові знання з розробки програмних продуктів для мобільних систем.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://travel-spend.com/
2. https://www.skyscanner.com.ua
3. Kotlin [електронний ресурс]. – 2018. – Режим доступу: https://kotlinlang.org/ Android Studio [електронний ресурс]. – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
4. REST [електронний ресурс]. – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/REST
5. Алістер Коберн. Сучасні методи опису функціональних вимог до систем
6. Б. Керниган, Р. Пайк. Практика программирования. – СПб.: «Невский диалект», 2001.
7. Біоніка інтелекта —2020—№1(94) —57-64с.(Розробка архітектуриагентної платформи для інтелектуального аналізу бізнес-процесів)
8. В. О. Грязнова, С. В. Єфіменко. Основи методології програмування. – К.: ВПЦ «Київський університет», 2005.
9. Дубова Н. SOA: подходы к реализации // Открытые системы. — 2004. — № 6. — С. 37–41.
10. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» Укл. В.В.Завгородній, К.М.Ялова.– Кам’янське: ДДТУ, 2019.– 144с.
11. Л.Новіков. Введення в Rational Unified Process. - Режим доступу:
12. Лешек А. Мацяшек Анализ требований и проектирование систем.
13. М.: Издательский дом "Вильямс". - 2002. - 432 с.
14. Майер Р. — Android 2. Програмування додатків для планшетних компютерів і смартфонів [Електронний ресурс]. — 2011. — Режим доступу: https://www.ozon.ru/context/detail/id/6752687/.

15. Методологія проектування та інструментарій для створення мобільних додатків. [Електронний ресурс]. –Режим доступу: http://www.kpi.kharkov.ua/archive/Наукова_періодика/vestnik/новые_решения_в_современных_технологиях/МЕТОДОЛОГІЯ_ПРОЕКТУВАННЯ_ТА_ІНСТРУМЕНТАРІЙ.pdf
16. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПЗ Программная инженерия. Программные требования. Режим доступу: http://www.sorlik.ru/swebok/3-1-software_engineering_requirements.pdf
17. Рик Роджерс — Android. Розробка додатків / Роджерс Рик. — М.: Эком, 2010. — 130 с.
18. Т.П. Караванова. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. – К.: Форум, 2002.
19. Э. Кингсли-Хьюджес, К. Кингсли-Хьюджес. C# 2005. Справочник программиста. – М.: ООО «ИД Вильямс», 2007.

Додаток А

Код програми

```
include ':app'
rootProject.name = "Imeldianne"
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "ru.russiantechapp.android.imeldianne"
        minSdkVersion 22
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
```

```
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.recyclerview:recyclerview:1.2.0'
}

// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    ext.kotlin_version = "1.4.32"
```

```
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.2"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```