

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему:

**«СИСТЕМНИЙ АНАЛІЗ ТЕХНОЛОГІЙ МОБІЛЬНОЇ РОЗРОБКИ ДЛЯ
ДОДАТКУ РОЗРАХУНКУ ФІНАНСОВИХ ВИТРАТ»**

Виконала: студентка групи 2147
ст.

(підпис) Озерянова О.В.

Керівник роботи:

(підпис) Гайдаєнко О.В.

м. Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студентці Озеряновій Ользі Василівні

1. Тема роботи: «Системний аналіз технологій мобільної розробки для додатку розрахунку фінансових витрат»

Керівник роботи: Гайдасенко О.В.

Затверджені наказом ректора № 269-уч від "18" травня 2022 року.

2. Термін подання роботи: 20 червня 2022 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів):

1) Дослідження предметної області і обґрунтування запропонованих рішень;

1.1. Загальні відомості та обґрунтування доцільності розробки

1.2. Особливості класифікації мобільних додатків

1.3. Етапи розробки мобільних додатків для iOS або Android

1.4. Ключові моменти при розробці мобільних додатків

1.5. Аналіз та порівняльна характеристика існуючих аналогів

1.6. Огляд технологій розробки мобільних додатків

1.7. Аналіз мов програмування

1.8. Порівняльний аналіз середовищ програмування

2) Розробка задачі алгоритму її розв'язування;

2.1. Постановка задачі

- 2.2. Аналіз принципів розробки мобільних додатків
- 2.3. Аналіз особливостей розробки бази даних
- 2.4. Розробка моделей мобільного додатку
 - 2.4.1. Розробка структури бази даних
 - 2.4.2. Розробка ієрархічної моделі мобільного додатку
 - 2.4.3. Розробка алгоритму роботи мобільного додатку

3) Реалізація інформаційного, організаційного, технічного та програмного забезпечення;

- 3.1. Розробка бази даних автоматизованої системи
- 3.2. Розробка мобільного додатку
 - 3.2.1. Реалізація логіки програмного продукту
 - 3.2.2. Реалізація розмітки Android-додатку
- 3.3. Розробка інтерфейсу програмного продукту
- 3.4. Тестування програмного забезпечення
- 3.5. Аналіз методів тестування
 - 3.5.1. Чорний ящик
 - 3.5.2. Білий ящик
 - 3.5.3. Тестування з використанням емулятора
- 3.6. Тестування автоматизованої системи

4) Охорона праці.

- 1.1. Охорона праці та довкілля
- 1.2. Охорона праці
- 1.3. Правові та організаційні питання охорони праці
- 1.4. Правила охорони праці під час експлуатації електронно-обчислювальних машин
- 1.5. Пожежна безпека

5. Перелік презентаційних матеріалів:

Графічні матеріали у вигляді презентації слайдів у кількості 11 слайдів для демонстрації при захисті дипломного проєкту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайдаєнко О.В.	14.02.2022	23.02.2022
2	Гайдаєнко О.В.	23.02.2022	27.03.2022
3	Гайдаєнко О.В.	27.03.2022	27.05.2022
4	Гайдаєнко О.В.	27.05.2022	01.06.2022

7. Дата видачі завдання 14 лютого 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	14.02.22-23.02.22	
2	Розробка задачі алгоритму та її розв'язування, розробка моделей мобільного додатку	23.02.22-05.03.22	
3	Розробка структури бази даних, ієрархічної моделі та алгоритму роботи .	05.03.22-27.03.22	
4	Реалізація інформаційного, організаційного, технічного та програмного забезпечення	27.03.22-25.04.22	
5	Розробка мобільного додатку	25.04.22-27.05.22	
6	Вирішення питань охорони праці	27.05.22-01.06.22	
7	Оформлення дипломної роботи	01.06.22-04.06.22	
8	Подання дипломної роботи на кафедральний захист		
9	Подання дипломної роботи рецензенту		
10	Подання дипломної роботи на державний захист		

Студент

(підпис)

Озерянова О.В

(ПІБ)

Керівник роботи

(підпис)

Гайдаєнко О.В.

(ПІБ)

Анотація

У дипломному проекті розроблено автоматизовану систему ведення сімейного бюджету як мобільний додаток. Продукт вирішує проблему ведення обліку сімейних і особистих витрат і доходів. Програма значно полегшує контроль за фінансовим станом користувача, так як за потребою надає доступ до інформації про поповнення бюджету, щоденні та щомісячні витрати, заплановані покупки.

Для реалізації мобільного додатку розроблено модель роботи додатку, структуру бази даних, ієрархічну модель автоматизованої системи. Також спроектовано інтуїтивно-зрозумілий інтерфейс.

Дипломний проект викладений на 96 сторінках друкованого тексту, містить 14 рисунків, 4 таблиці.

Проект виконаний українською мовою на сучасному рівні з застосуванням комп'ютерних технологій.

Аннотация

В дипломном проекте разработана автоматизированная система ведения семейного бюджета как мобильное приложение. Продукт решает проблему ведения учета семейных и личных расходов и доходов. Программа значительно облегчает контроль за финансовым состоянием пользователя, так как при необходимости предоставляет доступ к информации о пополнении бюджета, ежедневные и ежемесячные расходы, запланированные покупки.

Для реализации мобильного приложения разработана модель работы приложения, структуру базы данных, иерархическую модель автоматизированной системы. Также спроектировано интуитивно-понятный интерфейс.

Дипломный проект изложен на 96 страницах печатного текста, содержит 14 рисунков, 4 таблицы.

Проект выполнен на украинском языке на современном уровне с применением компьютерных технологий.

Abstract

In the thesis project developed an automated system for managing the family budget as a mobile application. The product solves the problem of keeping records of family and personal expenses and income. The program greatly facilitates control over the financial state of the user, since, if necessary, provides access to information on budget replenishment, daily and monthly expenses, and planned purchases.

For the implementation of a mobile application, a model of the application, a database structure, a hierarchical model of an automated system has been developed. Also designed an intuitive interface.

The diploma project is presented on 96 pages of printed text, contains 14 pictures, 4 tables.

The project is executed in the Ukrainian language at the present level with the use of computer technologies.

Зміст

Вступ.....	9
1. Дослідження предметної області і обґрунтування запропонованих рішень.....	11
1.1 Загальні відомості та обґрунтування доцільності розробки.....	11
1.2 Особливості класифікації мобільних додатків	13
1.3 Етапи розробки мобільних додатків для iOS або Android.....	15
1.4 Ключові моменти при розробці мобільних додатків.....	18
1.5 Аналіз та порівняльна характеристика існуючих аналогів.....	23
1.6 Огляд технологій розробки мобільних додатків.....	28
1.7 Аналіз мов програмування.....	32
1.8 Порівняльний аналіз середовищ програмування.....	34
2. Розробка задачі алгоритму її розв'язування.....	38
2.1 Постановка задачі.....	38
2.2 Аналіз принципів розробки мобільних додатків.....	39
2.3 Аналіз особливостей розробки бази даних	41
2.4 Розробка моделей мобільного додатку	44
2.4.1 Розробка структури бази даних.....	46
2.4.2 Розробка ієрархічної моделі мобільного додатку.....	48
2.4.3 Розробка алгоритму роботи мобільного додатку.....	50

3. Реалізація інформаційного, організаційного, технічного та програмного забезпечення.....	54
3.1 Розробка бази даних автоматизованої системи.....	54
3.2 Розробка мобільного додатку.....	57
3.2.1 Реалізація логіки програмного продукту.....	57
3.2.2 Реалізація розмітки Android-додатку	60
3.3 Розробка інтерфейсу програмного продукту	64
3.4 Тестування програмного забезпечення.....	71
3.5 Аналіз методів тестування.....	72
3.5.1 Чорний ящик.....	72
3.5.2 Білий ящик.....	73
3.5.3 Тестування з використанням емулятора.....	73
3.6 Тестування автоматизованої системи.....	77
 4. Охорона праці та довкілля.....	81
4.1 Охорона праці.....	81
4.2 Правові та організаційні питання охорони праці.....	81
4.3 Правила охорони праці під час експлуатації електронно- обчислювальних машин	82
4.4 Пожежна безпека	91
 Висновки та рекомендації.....	93
Список використаних джерел.....	94

Вступ

На сьогоднішній день ринок техніки все більше тяжіє до портативних пристроїв. Адже для сучасної людини важливо постійно мати доступ до Інтернету, електронної пошти, соціальних мереж. Через незручність використання стаціонарних комп'ютерів і ноутбуків все частіше на перший план виходять мобільні телефони та планшети.

Розробка мобільних додатків на сучасному етапі є вкрай популярною послугою. Кількість користувачів мобільних пристроїв на різних платформах зростає з кожним днем. Сьогодні користувач вимагає широкого спектру різноманітного програмного забезпечення для офісу та вирішення бізнес-задач. Тому розробка мобільних додатків є дуже актуальною.

Однією з рис успішних людей є вміння поводитися з грошима, а саме вести їх облік. Раніше підрахунок власних витрат виконувався за допомогою ручки, паперу і калькулятора, що не дуже зручно. Зараз же, зі швидким розвитком смартфонів, облік власних фінансів стає набагато легшим і зручнішим завдяки використанню спеціальних мобільних додатків. За їх допомогою можна не тільки підрахувати свої витрати і доходи, а й спланувати власний бюджет на тиждень, місяць, рік, що є дуже актуальним для тих, хто має намір контролювати свої фінанси.

Мета та задачі дослідження. Метою роботи є підвищення ефективності ведення сімейного бюджету за рахунок створення сучасних інструментів обліку та аналізу особистих і сімейних фінансів та їх реалізації у мобільному додатку, що дозволяє автоматизувати процес моніторингу фінансового стану сім'ї та керувати фінансовими операціями із забезпеченням матеріальної стабільності сімейного бюджету.

Основними задачами дослідження є:

- Аналіз і систематизація існуючих аналогів, а також особливостей реалізації мобільних додатків.
- Розробка моделей автоматизованої системи ведення сімейного бюджету.
- Розробка бази даних та алгоритму функціонування мобільного додатку.
- Програмна реалізація та тестування автоматизованої системи ведення сімейного бюджету.

Об'єктом дослідження постає процес розробки мобільних додатків.

Предмет дослідження – методи та засоби реалізації мобільних додатків, моделі та алгоритми автоматизованої системи ведення сімейного бюджету.

Методи дослідження. У процесі дослідження використовувалися: технології розробки мобільних додатків під операційну систему Android, методи роботи з базою даних SQLite, методи передачі даних по HTTP протоколах при синхронізації локальної і серверної баз даних

1. Дослідження предметної області і обґрунтування запропонованих рішень

1.1 Загальні відомості та обґрунтування доцільності розробки

Однією з рис успішних людей є вміння поводитися з грошима, а саме вести їх облік. Раніше підрахунок власних витрат виконувався за допомогою ручки, паперу і калькулятора, що не дуже зручно. Зараз же, зі швидким розвитком смартфонів, облік власних фінансів стає набагато легшим і зручнішим завдяки використанню спеціальних мобільних додатків. За їх допомогою можна не тільки підрахувати свої витрати і доходи, а й спланувати власний бюджет на тиждень, місяць, рік, що є дуже актуальним для тих, хто має намір контролювати свої фінанси.

Ведення обліку власних фінансових витрат – це важлива складова кожного дня. Приділяючи всього п'ять хвилин на день для ведення такої «фінансової звітності» можна сформувати фінансову відповідальність і в майбутньому уникнути багатьох проблем із сімейними та особистими фінансами.

Планування та облік бюджету полягає в контролі щоденних доходів і витрат, аналізі корисних і даремних покупок, планування бюджету з метою досягнення довгострокових цілей, регулярного формування заощаджень з метою забезпечення фінансової безпеки на випадки непередбачених форс мажорних обставин.

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

1) планування фінансових заощаджень для майбутніх покупок, 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.

По суті, згадані мобільні пристрої (смартфони, планшети тощо) є невеликими комп'ютерами. Як будь-який комп'ютер, вони не можуть працювати без спеціальних програм. Окрім основної програми — операційної системи, вони зазвичай містять і ряд дрібних програм, переважно — мобільних додатків.

Мобільний додаток (англ. «Mobile app») — програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних додатків встановлені на самому пристрої або можуть бути завантажені на нього з он-лайн магазинів мобільних додатків, таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за плату.

Вони використовуються для безпроводної передачі (отримання) інформації та/або проведення певних, у тому числі і платіжних, операцій. Тобто створення мобільного додатка фактично є створенням програмного продукту.

Використання в певних напрямках діяльності мобільних додатків у наш «інтернетно-гаджетний» час дуже допомагає у досягненні конкретних господарських цілей і результатів.

Мобільний додаток для ведення бюджету 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.

Будь який мобільний додаток є об'єктом авторського права. Належність майнових права на мобільний додаток залежатиме від способу його створення. У разі створення додатка підприємство або особа, котра його створила, може мати на нього виняткові майнові права, а може мати лише визначений договором обсяг прав. Всі моменти та особливості обов'язково повинні визначатися у відповідних документах. Умова про те, що замовникові передаються виняткові майнові права на створений об'єкт у повному обсязі дозволить йому практично без обмежень розпоряджатися результатами розробки.

1.2 Особливості класифікації мобільних додатків

Складно уявити сучасний бізнес-проект, який не використовує засіб просування мобільними додатками. Сфера розвивається стрімко швидко, більшість користувачів гаджетів, девайсів активніше проводять час за додатком.

Перед тим, як починати розробку мобільних додатків, слід чітко визначитися з головними функціями майбутньої розробки. Від того, які функції має виконувати мобільний додаток, залежить тривалість його розробки, рівень складності і, звичайно, ціна. Загалом визначають чотири основні типи мобільних додатків:

1. Інформаційні додатки. Основна мета – інформування користувача (потенційного клієнта). Розповідає про новинки компанії. Автономно від ресурсу допускається виключно новинне додаток, орієнтоване на певний сегмент новин або ж на все відразу. Використовується для реклами – інформаційні додатки одні з самих благодатних майданчиків розміщення реклами серед додатків.

2. Корпоративні. Їх мета – спрощення роботи компанії, швидка передача даних між працівниками або отримання корпоративної інформації. Цільовою аудиторією таких мобільних додатків є, насамперед, працівники компанії, а також

реальні та потенційні клієнти та партнери. Розробка мобільних додатків такого типу простіша, для їх дизайну використовуються кольори та елементи корпоративного стилю компанії.

3. Контентні. Це мобільні додатки, основна мета яких надавати різного роду інформацію (текстову або у відео чи аудіоформаті). Головним чином, така розробка мобільних додатків здійснюється для засобів масової інформації, радіо, телеканалів чи порталів.

4. Сервісні. Їх мета впливає з назви, а саме — надавати певні сервісні послуги, тобто виконувати завдання, які ставить користувач, у режимі реального часу. Розробка мобільних додатків такого типу є складною, адже вони повинні працювати, як годинник: починаючи від калькулятора або будильника і закінчуючи програмами для роботи з великими обсягами тексту або графіки.

5. Ігрові. Лідер додатків сьогодні. Підійдуть для продажу рекламних майданчиків, вірусна гра може захопити користувача надовго: поїздки в метро, нудні лекції — кожна вільна хвилина з грою. Заробляти можна платним контентом або рекламою. Створення мобільних додатків такого типу — привабливий дизайн, захоплюючий сюжет. Складний вид розробки додатків для iOS або Android. Основне завдання таких мобільних додатків — розважати, але це не означає, що воно єдине. Отже, розробка мобільних додатків такого типу ускладнюється необхідністю вдало розмістити контекстну або пряму рекламу.

6. Інтернет-магазин. Тепер онлайн торгівля стала доступнішою. Розробка програми для Android — це зростання продажів на третину (за статистикою більше половини користувачів СНД використовують продукт від Google на мобільних девайсах). Основна мета — можливість швидко зробити покупку, привабливий інтерфейс, оплата за допомогою тих же мобільних банківських служб.

Вищезазначені програми — це лише найпопулярніші види. Створення мобільних додатків обмежується тільки вашою фантазією і вимогами.

Для розробників мобільних додатків більш важливою є класифікація з точки зору їх структури:

1) веб-додатки можна назвати мобільною версією сайту з розширеним інтерактивом. Веб-додатки не розміщуються в спеціалізованих магазинах, а використовують для роботи браузер. Швидкість роботи таких додатків залежить від якості Інтернет-з'єднання, крім того вони характеризуються низькою вартістю і швидкими термінами реалізації, є кроссплатформенними;

2) гібридні додатки або генератори мобільних додатків – щось середнє між нативними і веб-додатками, встановлюються через офіційні магазини, мають обмежений доступ до апаратної частини пристрою, за функціоналом і якістю наближені більше до нативних додатків, проте дешевше їх, залежать від фреймворку, який використовувався розробником цього додатку;

3) нативні додатки пишуться на мовах програмування під конкретну платформу і вбудовуються в операційну систему, працюють швидко і коректно, володіють перевагою як по функціоналу, так і за швидкістю роботи інших мобільних додатків. Дані програми також мають доступ до апаратної частини пристроїв: камери, мікрофону, акселерометру, телефонною книги і т.п., економно витрачають ресурси, працюють повністю або частково при відключеному Інтернет-з'єднанні. Звичайно, мають високу вартість і великі витрати за часом написання програми, як наслідок того, що розробник повинен володіти спеціальними знаннями в середовищі розробки, а також з тієї причини, що кожній платформі відповідає своя мова програмування.

1.3 Етапи розробки мобільних додатків для iOS або Android

iOS - це операційна система, яка не обмежена жодним зональним чи мовним бар'єром. Це один з головних факторів цієї операційної системи, яка стає настільки відомою лише за 10 років свого розвитку. IOS підтримує 40 різних мов.

Як вже було сказано раніше, iOS обмежується лише обладнанням Apple або пристроями, виготовленими Apple. Однак існує безліч пристроїв Apple та їх версій, які підтримують iOS.

Отже, Apple має закриту систему, на відміну від Android, яка є відкритою.

Розробка андроїд додатків, також як і розробка додатків для iPhone між собою особливо не відрізняються. Хоч програмне забезпечення – головні конкуренти, алгоритми створення додатків не відрізняються.

Спочатку складається макет. Розробка додатків для Android буде якісною тільки у разі детальної промальовування макета за завданням замовника. Останній зможе наочно побачити передбачуваний результат, внести правки на папері – набагато простіше, ніж писати код заново.

Далі розробка додатків для iOS, розробка андроїд додатків повинна включати створення прототипу. Це спрощена версія програми, що дозволяє:

- Перевірити основи інтерфейсу.
- Визначити, підійде макет цільовим користувачам чи ні.
- Внести правки до початку верстки.

Прототип при розробці андроїд додатків дозволяє окреслити майбутнє використання додатка, без деталей.

Якщо макет задовольнив, наступним етапом розробки програми iOS або android додатків буде верстка. Перед нею стоїть проаналізувати основні конкурентні додатки: їхні козири, інтерфейс, спеціальні можливості. Потрібно додавати щось своє, в крайньому випадку, бути не гірше інших.

Розробка додатків для Android буває:

- нативною;
- в якості веб-версії сайту.

Перша – краще. В цьому випадку пишеться з кодом, рекомендованому розробником самої мобільної платформи. Що iOS, Android краще реагують на такі додатки, їх можна розмістити на AppStore (доведеться платити третину Каліфорнійській компанії – тому і розробка додатків для iOS дорожче андроїда) або GooglePlay. Якщо створення мобільних додатків – спосіб просування ресурсу, то без он-лайн маркетів не обійтися. Звісно створення сайтів включає тест додатку, проте веб-версія сайту підійде в тому випадку, якщо трафік відвідувачів і так влаштовує, потрібно просто зробити зручнішим користування сайту на екранах мобільних.

Тестування додатків

Тестування інтерфейсу користувача iOS можна класифікувати на такі категорії:

- Вхідні дані: Тестування функціональності сенсорного екрану (наприклад, довгий / короткий дотик, 3D дотик, прокрутка), розміри кнопок, розташування кнопок, колір шрифтів та їх розмір тощо належать до цієї категорії.

- Жорсткі клавіші: власні програми функціонують безперебійно за допомогою вбудованих апаратних клавіш / жорстких клавіш, присутніх на пристрої, таких як клавіша додому, кнопки звуку тощо. Програма, що досліджується також має взаємодіяти з клавішами аналогічним чином.

- М'які клавіші / м'яка клавіатура. Необхідний зовнішній вигляд клавіатури, пристосування, коли це вам не потрібно, підтримка смайлів, символів, усіх символів / символів тощо.

- Екран: якщо додаток підтримується на кількох пристроях, слід перевірити його орієнтацію на всіх пристроях. Можуть відбуватися деякі зміни роздільної здатності залежно від пристрою, який обраний для процесу тестування. Одночасно слід проводити тестування для портретного/альбомного режимів та використання клавіатури в кожному з випадків.

Після того, як розробка додатків на Android або розробка додатків iOS завершена, настає етап тестування. Випускаючи у світ сирий продукт, можна наваритися на негатив користувачів, який зменшить кількість скачувань навіть після доопрацювання продукту.

Що перевіряється тестуванням? В першу чергу, юзабіліті програми. Потрібно пройти всі рівні, якщо це гра, скористатися усім переліком послуг, купити товар (інтернет-магазин), тестувати максимальні можливості графіки. Тестування проходить на декількох гаджетах. Якщо це розробка додатків для iOS, потрібно перевіряти на роботу iPhone, iPad відразу.

Перевіряється реакція, що викликається хаотичними натисканнями, регулюється енергоспоживання (чим менше – тим краще).

Після тестування, виявлені помилки усуваються. Фінальний етап – публікація програми. Переважно – в google play, AppStore (більше 70% користувачів завантажують програми звідти).

Додаток вимагає постійного оновлення програмного забезпечення, контенту (особливо у разі додатків інформаційного типу, якщо є карти, інтернет-магазин). Цим теж займаються розробники. Плюс, як би ретельно не проводилося тестування, після публікації знаходяться кілька нових проблем – чим швидше вони усуваються, тим менше користувачів йде остаточно.

Додаток необхідно просувати. Йому, як і сайту потрібна якісна реклама фейсбук. Практика показує, що просування пропозиції в два рази швидше приносить очікуваний результат, ніж просто якісно розроблений і викинутий в GooglePlay або Appstore і забутий.

1.4 Ключові моменти при розробці мобільних додатків

Ключові моменти, що роблять мобільний додаток дійсно якісним:

- Час запуску.

Важливою характеристикою мобільних додатків є час їх запуску. Оскільки мобільними пристроями зазвичай користуються часто і протягом нетривалих проміжків часу, здатність мобільного додатку до швидкого запуску є обов'язковою вимогою. Можливо, 6-секундне споглядання заставки текстового процесора, енциклопедії або середовища розробки в очікуванні появи основного вікна відповідної програми і приносить мало задоволення, однак, з урахуванням загальної тривалості типового сеансу роботи з такими додатками, цей фактор можна вважати малозначним.

У разі ж мобільного застосування, яке користувач збирається використовувати протягом 20 секунд для уточнення або поновлення невеликої порції інформації, втрата 6 секунд представляється недозволеною розкішшю. Розумно керуватися мнемонічним правилом, згідно з яким тривалість сеансу роботи користувача з додатком повинна набагато перевершувати тривалість

запуску цього додатка. У разі додатків для настільних комп'ютерів робочі сеанси тривають довше, і тому користувачі готові миритися з періодами очікування більшої тривалості. У разі ж мобільних додатків тривалість робочих сеансів менше, і тому висувається вимога, щоб тривалість періоду запуску програми була відповідно меншою. Тривалість періоду запуску додатка є важливим чинником і у випадку настільних комп'ютерів, але у випадку мобільних пристроїв, які характеризуються нерегулярністю і короткочасністю робочих сеансів, цей чинник набуває вирішальне значення.

- Відгук пристрою.

Сприймаючи мобільні пристрої як слухняні механічні іграшки, люди очікують від них відповідної поведінки. Коли користувач впливає на маніпулятор, натискає на кнопку або будь-яким іншим чином маніпулює елементами керування пристрою, він розраховує на фізичну реакцію з його боку. Не отримавши негайної реакції з боку пристрою, нетерплячий користувач починає нервувати і тут же повторює спробу виконання потрібної операції.

Якщо повторна спроба маніпуляції елементом управління, виконання клацання чи іншого аналогічного дії будуть оброблені вашим додатком або, помилково, іншим додатком, то в результаті цього можуть виникнути проблеми. Тому вкрай важливо дбати про те, щоб після виконання яких-небудь дій з пристроєм користувач отримував від нього негайне підтвердження реакції на ці дії в тій чи іншій формі.

Ідеальним варіантом такого підтвердження є виконання запитаної операції; краще цього нічого бути не може. На другому місці в цьому відношенні стоїть підтвердження того, що запит отримано і обробляється у фоновому режимі, в той час як додаток зберігає здатність сприймати інші запити. Третє місце належить відображенню курсору очікування, який повідомлятиме користувача про те, що запит обробляється; додаток ні на які подальші запити не реагує, але користувач знає, що робота з обслуговування його запиту розпочата і виконується. Найгірший з варіантів – це коли користувач не бачить ніякої відповідної реакції з боку пристрою і йому залишається лише здогадуватися про те, сприйнятий його

запит чи ні. Як і у випадку інших характеристик, які ми будемо вивчати, ці вимоги не є унікальними для мобільних пристроїв, але в даному випадку вони набувають особливого значення в силу специфіки роботи з цими пристроями і того, чого люди від них чекають.

- Фокусування уваги на окремих завданнях.

Ще однією характеристикою мобільного додатку, що визначає його успішність, є ступінь його спеціалізації. Для програми повинен бути чітко визначений набір завдань, ефективно вирішення яких воно має забезпечувати; здійснюються при цьому операції повинні виконуватися швидко і вимагати лише мінімальної кількості клацань, натискань або інших дій з боку користувача.

Саме з цієї причини в пристроях часто передбачаються спеціальні кнопки, що відповідають окремим завданням, які здатний вирішувати даний пристрій (наприклад, кнопка для переходу безпосередньо до бази даних, що зберігає інформацію про контакти, кнопка для перегляду календаря зустрічей або кнопка для зчитування штрих-коду та передачі його певного додатка).

Ви повинні докласти максимум зусиль до того, щоб якомога точніше визначити коло тих завдань, що часто зустрічаються та вирішення яких має вимагати мінімуму дій з боку користувача. Цією рекомендацією слід керуватися як щодо високорівневих функцій програми, так і щодо низькорівневих завдань, які зазвичай доводиться вирішувати користувачам в процесі роботи з даним додатком. Наприклад, якщо мобільний додаток має забезпечувати швидке введення дат, то ви повинні подбати про максимальне спрощення (і прискорення) зазначеного процесу і вимірюванні його ефективності. Якщо ваш мобільний додаток повинен забезпечувати можливість вказівки певного місця розташування за допомогою схеми міських вулиць, щоб користувач міг вибрати маршрут руху до кінцевого пункту призначення, то ви повинні подбати про те, щоб це могло бути зроблено якомога швидше.

Одна з поширених помилок, що допускаються розробниками при створенні додатків для мобільних пристроїв, полягає в тому, що програмний код намагаються зробити якомога більш коротким, щоб тим самим максимально

зменшити розмір програми. Хоча подібні цілі і є благородними, вони не повинні досягатися за рахунок зниження продуктивності праці користувачів. Не шкодуйте часу на написання додаткового коду, фокусую чого увагу користувачів на виконанні спеціалізованих завдань і що дозволяє скоротити час отримання кінцевого результату

- Налаштування взаємодії із зовнішніми джерелами інформації.

Дуже важливо розуміти, що створення відмінного мобільного додатку – це не лише написання коду, який правильно виконується на пристрої. Не слід забувати також про зовнішнє програмне забезпечення, з яким взаємодіє ваш мобільний додаток. Необхідно, щоб у додатку відповідним чином враховувалися особливості інформаційних джерел, що обслуговують мобільні пристрої, і здійснювалася перевірка того, що інформація надається в тій формі, яка відповідає даному пристрою. Як приклад можна привести служби електронної пошти для мобільних пристроїв. Багатофункціональні поштові додатки вимагають наявності відповідного програмного забезпечення як на стороні сервера, так і на стороні клієнта. Клієнт здійснює доступ до сервера для отримання інформації щодо повідомлень, що надійшли і завантажує відповідний вміст в локальний пристрій. Оскільки мережеві з'єднання, використовувані мобільними пристроями, відрізняються нерегулярністю доступу до них, зниженою пропускну здатністю, а в багатьох випадках і більш високою вартістю в порівнянні з тими сполуками, які використовуються персональними комп'ютерами, поштові служби сервера повинні бути пристосовані для задоволення цих вимог. Зазвичай це досягається за рахунок того, що на сервері надаються засоби, що дозволяють обмежувати обсяг отриманого вмісту, або фільтри, пропускають лише ту інформацію, яка дійсно може знадобитися користувачеві мобільного пристрою. Для забезпечення підтримки ефективних сценаріїв при обслуговуванні мобільних пристроїв може знадобитися розширення функціональних можливостей тих служб на стороні сервера, які спочатку призначалися для обслуговування персональних комп'ютерів. Крім того, повинні бути передбачені такі механізми змін параметрів конфігурації програми для його виконання на серверах, настільних комп'ютерах і

самих мобільних пристроях, за допомогою яких користувачі зможуть визначати свої інформаційні запити і налаштовувати інформаційні фільтри таким чином, щоб забезпечувалося виконання відповідних вимог. Проектування мобільних додатків далеко не обмежується фізичними межами самих мобільних пристроїв.

- Однаковості стилю інтерфейсу.

З урахуванням того, що будь-який мобільний пристрій відрізняється компактністю і самодостатністю, бажання користувачів працювати з інтерфейсом, який не залежить від характеру розв'язуваних за допомогою пристрою завдань, видаються цілком природними. Кожен мобільний пристрій має власний функціональний колорит. Типовий вдало спроектований додаток сприймається не як розрізнений набір засобів, а як гармонійне розширення можливостей самого пристрою.

З цієї причини при проектуванні додатків для мобільних пристроїв дуже важливо дотримуватися єдиного стилю. Способи запуску і зупинки додатків, переміщення в межах інтерфейсу, а також організації діалогів між користувачем і додатками повинні ретельно продумувати окремо для кожного типу пристроїв. Користувачі мобільних пристроїв підсвідомо підлаштовуються під метафори користувача інтерфейсу, і будь-які відхилення від звичних шаблонів доставляють їм відчутні незручності. Створення звичної робочого середовища багато значить і в разі додатків для настільних комп'ютерів, але в силу різноманітності можливостей, які пропонуються такими додатками, вони дозволяють добитися однієї і тієї ж мети декількома способами (наприклад, за допомогою сполучень клавіш, клацань мишею, меню і панелей інструментів). У разі ж мобільних пристроїв для вирішення даної задачі часто є тільки один шлях, і користувач мимоволі звикає до одного певного способу. Набагато краще мати чотири різних версії додатка, кожна з яких відповідає певній метафорі користувача інтерфейсу конкретного пристрою, ніж одне загальне додаток, який не може бути інтегроване належним чином ні в одне цільове пристрій.

- Відмінності в архітектурі комп'ютерів.

Якщо скористатися аналогією з області архітектури, то настільні та переносні комп'ютери можна уподібнити великим садибам у сільській місцевості. Тоді відносно мобільних пристроїв можна сказати, що вони схожі на окремі чи комунальні міські квартири. Кожен з цих типів жител призначений для задоволення певних потреб і висуває специфічні вимоги щодо найбільш ефективних способів їх використання.

Зазвичай садиби характеризуються великими розмірами, і в них є багато місць, придатних для зберігання різних речей. Одні з таких місць знаходяться неподалік, тоді як дістатися до інших дещо важче. У садибах рідко використовувані речі зазвичай зберігаються на горищах або в підвалах, звідки їх можна періодично витягувати по мірі того, як в них виникає необхідність.

З іншого боку, в умовах міських квартир місць, придатних для зберігання речей, значно менше. У квартирах зберігають речі, які повинні бути завжди під рукою. Ті ж речі, якими користуються рідко, в квартирі не тримають. Замість того щоб купувати рідко використовувані речі, їх часто вигідніше орендувати.

Те ж саме справедливо і щодо мобільних пристроїв. Як оперативна пам'ять, так і пам'ять для довготривалого зберігання даних орієнтовані на зберігання тієї інформації, яка використовується найчастіше. Рідко використовувані дані повинні виштовхувати на сервери, звідки їх можна буде запросити, коли вони будуть потрібні.

У настільних комп'ютерах ОЗП і довготривала пам'ять розділені. Водночас, в пристроях ОЗП часто використовуються і як робоча пам'ять додатків, і як проміжна або довгострокова пам'ять. В якості довгострокових сховищ даних все частіше використовується флеш-пам'ять, зазвичай у вигляді знімних карт пам'яті.

1.5 Аналіз та порівняльна характеристика існуючих аналогів

Для обліку фінансових витрат існує безліч мобільних додатків, які не завжди повною мірою виконують функції, потрібні користувачеві. Для розробки додатку

для ведення бюджету з оптимальним набором функцій проаналізовано переваги та недоліки найвідоміших аналогів.

Основним конкурентом є додаток Toshl Finance – найбільш просте і цікаве рішення для контролю особистих витрат. Щоб почати роботу, необхідно зареєструватися, вибрати основну валюту та синхронізуватися з сервером. Кожний календарний день в додатку являє собою закладку з фінансовими операціями. Додаток має тільки базові функції і включає такі розділи як витрати, дохід, статистика і бюджет. Замість звичних категорій витрат і доходів тут використовуються мітки, що істотно спрощує введення даних. З особливостей програми варто відзначити інтуїтивно зрозумілий інтерфейс (рис. 1.1), наочні інфографіки та автосинхронізація у фоновому режимі. Toshl Finance доступний безкоштовно, проте є і платна підписка, яка дозволяє додавати необмежену кількість доходів і бюджетів, експортувати звіти в PDF, Excel і Google Docs.

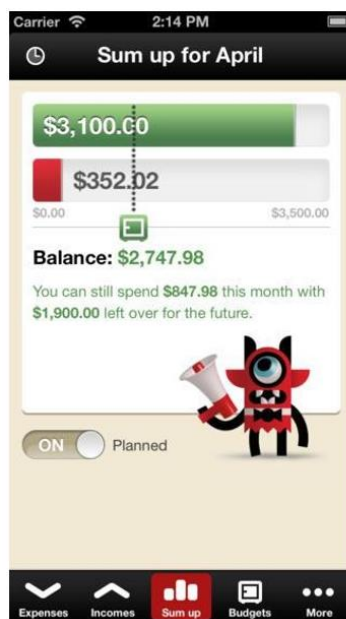


Рис.1.1 – Додаток Toshl Finance

Наступним аналогом розробки є додаток MoneyLover. Він має інтуїтивно простий дизайн і зрозумілий користувачеві набір функцій (рис. 1.2). Цей додаток буде зручним для тих, хто є фанатом яскравої графіки, і кому не потрібен поглиблений статистичний аналіз. Програма дозволяє створювати бюджети,

планувати особисті витрати, додавати власні категорії транзакцій, використовувати календар.

Також є функція створення плану заощаджень. Гроші, які планують заощадити, не показуються в загальному балансі, що дозволяє «забути» про них і не розраховувати при плануванні витрат на найближчий час. Крім того, можна встановлювати бюджет на окрему категорію чи на всі категорії в гаманці. Якщо ви перевищите бюджет, то декілька разів на день отримуватимете про це сповіщення. Вхід в програму можна захистити пін-кодом, а отже, зберегти витрати та доходи користувача від допитливих очей.

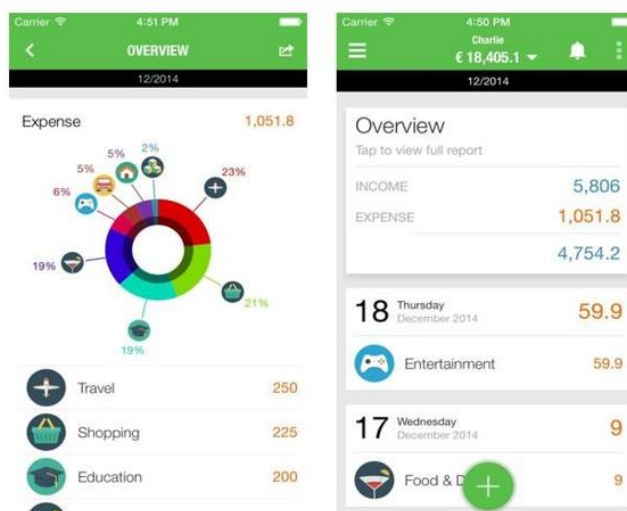


Рис.1.2 – Додаток MoneyLover

Monefy – додаток, який також тішить гарною графікою та простотою у використанні. Всі витрати та доходи подані у вигляді великої кольорової діаграми (рис.1.3). Розробники створили велику кількість категорій, за якими можна розділити витрати, наприклад, розваги, походи в кафе, шопінг, витрати на продукти, товари для дому, предмети гігієни, таксі та громадський транспорт та ін. Крім того, категорії можна змінювати на власний смак, видаляючи непотрібні та додаючи нові. Всі дані можна синхронізувати з Dropbox, тому їх можна переглядати на комп'ютері та інших пристроях.

Monefy також можна захистити пін-кодом для особистого користування, або ж, навпаки, створити декілька акаунтів, тоді додатком зможуть користуватись декілька людей з одного пристрою.



Рис.1.3 – Додаток Monefy

Додаток FinancePM розроблений для Android та Windows Phone. Однією з переваг цього додатку є можливість ведення обліку фінансів одразу у кількох валютах. Це досить зручно для тих, хто багато подорожує за кордоном, адже не потрібно кожного разу конвертувати витрати в національну валюту. Крім того, є можливість створювати багато гаманців та підлаштовувати категорії витрат та доходів під себе (рис. 1.4).

Програма може встановлювати обмеження на бюджетні витрати і попереджувати, коли користувач наблизився до критичного рівня витрат. Також присутня функція налаштування постійних виплат (мобільний зв'язок, комунальні послуги, спортзал) і створення шаблонів платежів з повторюваною сумою.



Рис.1.4 – Додаток FinancePM

Переваги і недоліки аналогів розробки наведено в таблиці 1.1.

Табл.1.1

Порівняння аналогів

Назва	Переваги	Недоліки
Toshl Finance	— особистий додаток фінансів для мобільних телефонів; — автоматично синхронізується з веб- додатком Toshl.com.	— промоверсія занадто дорога; — немає можливості купувати тільки необхідну функціональність.
Money Lover	— зручний та інтуїтивно зрозумілий інтерфейс; — є функція планування заощаджень.	— не містить поглибленого статистичного аналізу; — немає ПК версії програми.
Monefy	— доходи і витрати - у вигляді діаграми; — існують категорії для витрат; — дані синхронізуються з Dropbox.	— Відсутня можливість планування покупок; — відсутність бюджету для групи користувачів.
Finance PM	— облік фінансів одразу у декількох валютах; — налаштування постійних витрат; — ліміт на витрати.	— вимагає додаткових зусиль і часу для введення даних; — немає ПК версії.

1.6 Огляд технологій розробки мобільних додатків

Розробка додатків для мобільних пристроїв – це процес, при якому додатки розробляються для невеликих портативних пристроїв таких як КПК, смартфони, стільникові телефони. Ці додатки можуть бути попередньо установлені на пристрій в процесі виробництва, завантажені користувачем за допомогою різних платформ для розповсюдження ПЗ або являтися веб-додатками, які обробляються на стороні клієнта або сервера.

Сьогодні існує великий вибір мов програмування для розробки мобільних додатків. Це пов'язано з тим, що для різних мобільних пристроїв доводиться використовувати різні мови програмування, що обумовлене тим, що мобільні пристрої мають різні операційні системи. Цільова платформа (або платформи) – iOS, Android, Windows Phone, BlackBerry – буде мати значний вплив на мову розробки, яка буде використовуватися. Наприклад, можна розробляти рідні додатки для кожної платформи або використовувати сторонній інструмент для оптимізації своїх додатків на різних платформах. Другий підхід може заощадити час і зусилля, хоча це може вплинути на зручність використання. Сучасні мобільні пристрої пропонують широкий спектр варіантів розробки.

Проаналізуємо основні технології, які використовуються для розробки додатків для мобільних телефонів. Перша технологія – це Java 2 Micro Edition (J2ME). Це набір специфікацій і технологій, призначених для різних типів портативних пристроїв. Існують два основні напрями: Connected Device Configuration (CDC) і Connected Limited Device Configuration (CLDC). Напрямок визначає тип конфігурації центральних бібліотек Java, а так само параметрів віртуальної машини Java (в якій будуть використовуватися додатки). Пристрої, які використовують технологію CDC будуть більш розвиненими, в якості прикладу можна навести комунікатори. До пристроїв CLDC відносяться звичайні мобільні телефони, які апаратно володіють більш скромними можливостями (ресурсами). Спеціальні режими дозволяють визначати функціональність конфігурацій для різних типів пристроїв. Режим Mobile Information Device Profile (MIDP)

призначений для CLDC портативних пристроїв з можливістю спілкування. CLDC і MIDP закладають основу реалізації J2ME. Java-код інтерпретується безпосередньо самим пристроєм за допомогою так званої Java Virtual Machine. Цей механізм робить можливим вільне розповсюдження Java-додатків, так як вони працюють на всіх пристроях з аналогічною Java-платформою. Програмування Java-додатків і на сьогоднішній день займає більшу частину, так як більшість мобільних пристроїв (в основному мобільні телефони) в світі мають вже встановлену Java-систему.

Наступна популярна сьогодні технологія – це технологія Qt. Вона в основному використовується в якості крос-платформного середовища, яке дозволяє використовувати написані з її допомогою додатки на різних пристроях і операційних системах, у тому числі Windows, Mac OS X, Linux, Symbian, Android та інших. Із зростаючою користувальницькою базою Qt, зростає і потреба у вбудованих, мобільних додатках і UI-розробників. Qt є однією з найбільш вдалих бібліотек для C++. Налаштування додатків, розроблених для мобільних пристроїв, відбувається за допомогою емулятора, який міститься в середовищі розробки. Таким чином, можна писати складні програми для мобільних пристроїв з використанням бібліотек C++ і підтримкою платформ. Для роботи Qt на мобільних пристроях необхідна установка певного фреймворку.

Ще однією технологією є технологія розробки Windows Phone SDK. Код додатка, що розроблюється, описується на мові XAML. Хоча насправді – це просто XML файли з мовою розмітки XAML. Платформа Windows Phone не просто чергова платформа для мобільних пристроїв. Вона містить у собі не тільки технологічну складову, але і повністю опрацьовану концепцію дизайну інтерфейсу і взаємодії з користувачем під назвою Metro-дизайн або стиль Metro. Вся розробка під Windows Phone ведеться в середовищі Visual Studio. Для мобільних додатків під Windows Phone відладка та тестування відбувається за допомогою емулятора Windows Phone у середовищі розробки Windows Phone.

Наступна технологія розробки – це iPhone SDK. Розробка під операційну систему iOS можлива тільки з використанням Mac OS X. Але в Інтернеті можна

знайти статті, як можна проводити розробку безпосередньо на Macintosh і навіть на VM. Варто зауважити, що Apple надає інструменти безкоштовно, але платити доведеться за підписку розробника. Для написання програм під iPhone пропонується використовувати Objective-C. При цьому є можливість писати так само і на C, і C++ (для цього необхідно змінювати розширення файлів з .M на .Mm). Правда при цьому повністю звільнитись від Obj-C не вдасться, майже весь API розрахований саме на Obj-C, виключення складають наприклад OpenGL (хоча для його ініціалізації доведеться використовувати кілька рядків коду на Obj-C), так само повністю доступні стандартні бібліотеки C/C++. Налаштування додатка відбувається за допомогою середовища XCode і емулятора iPhone встановленого в ній.

Для написання додатків під Symbian можна використовувати мову програмування C++. Розробка для Symbian OS (якщо говорити про C++) зазвичай ведеться на ПК. Серед розробки – звична для багатьох програмістів Visual Studio, це також можуть бути IDE Metrowerks CodeWarrior Development Studio, Borland C++ BuilderX Mobile Edition, Carbide. C++ (відносно нова IDE, створена компанією Nokia на базі Eclipse), забезпечена додатковими інструментальними пакетами (SDK). Розробнику доступні практично всі звичні можливості, як створення програмне забезпечення (ПЗ), так і налаштування (трасування, перегляд змінних, стека викликів, структур класів). Відлаштування програми запускається в емуляторі Symbian OS. Цю підсистему правильніше було б назвати симулятором, оскільки імітуються не апаратні засоби, а лише програмне оточення (відповідні API операційної системи, реалізовані поверх API Win32). При цьому програмні модулі, які завантажуються в емулятор, являють собою виконані файли для архітектури x86, відповідне ПЗ для цільової платформи формується після підсумкової компіляції. Це передбачає певну специфіку, але сьогодні емулятор забезпечує досить високу ступінь подібності та проблеми виникають лише при створенні програм, які нестандартно використовують API.

Для розробки під Android можна використовувати середовище Eclipse з встановленим плагіном ADT. Розробка ведеться на мові програмування Java. Є

можливість налагодження з використанням емулятора вбудованого в ADT або безпосередньо на мобільному пристрої з ОС Android. Існують різні версії SDK, які використовуються для написання коду для різних версій Android. Підтримується майже повна зворотна сумісність версій. Крім розробки на мові Java підтримується можливість більш низькорівневої розробки з використанням Android NDK (Native Development Kit) на мові C/C++. На частку мобільних пристроїв на базі Android припадає 50 відсотків всіх користувачів. Створення додатків для цієї платформи відрізняється від розробки сайту. В даному випадку також приділяється увага інтерфейсу – дружньому та інтуїтивно зрозумілому. Щоб не зникало бажання використовувати додаток кожен день, важливо опрацювати прототипи і дизайн програми. Розробка додатків для Android вимагає від розробників глибоких знань і розуміння принципів ефективної взаємодії користувача з додатком, законів дизайну.

Платформа Android надає розробникам найбільшу свободу вибору ОС, на якій розробляти додатки. Android розробники можуть використовувати Windows, Mac або Linux. BlackBerry користувачі можуть вибирати між Mac, Windows і Linux. Якщо ви хочете розробити для iOS і пристроїв Windows, єдиним варіантом є Mac і Windows відповідно.

Основні принципи проектування програмного забезпечення для мобільних пристроїв незмінні незалежно від платформи та операційної системи, встановленої на ній. Головними відмінностями є технічні особливості, такі як інструментарій, API і SDK, парадигми проектування інтерфейсу.

Для розробки автоматизованої системи ведення обліку фінансових витрат обрано платформу Android, так як на сьогоднішній день вона є найбільш популярною серед користувачів мобільних додатків.

1.7 Аналіз мов програмування

На сучасному етапі розвитку комп'ютерних технологій існує велика кількість мов програмування для розробки мобільних додатків. Для розробки додатків під платформу Android найчастіше використовують мову Java, C# та C++ (при розробці у Xamarin). Кожна з цих мов програмування має свої особливості і створена для вирішення задач певного типу.

Java значно спрощує життя розробників, встановлюючи жорсткі обмеження на способи передачі параметрів. Це особливо помітно при порівнянні з C++, де параметри можна передавати: 1) за значенням, 2) за посиланням, а враховуючи наявність покажчиків, код програми може стати надзвичайно заплутаним.

Мова Java виграє у C++ у багатьох аспектах і насамперед спрощує написання коду за рахунок відсутності:

- a) покажчиків;
- b) заголовних файлів;
- c) множинного наслідування.

Мова C# розроблялася з урахуванням досвіду багатьох більш ранніх мов програмування, в першу чергу C++ і Java [18]. Java і C# є подібними мовами.

Обидві ці мови можна розглядати як спробу вдосконалити C++, і в обох випадках це вдалося. Властивості, характерні як для C#, так і для Java:

1) вихідний текст програми компілюється в проміжний код, що не залежить від мови і платформи; цей код надалі виконується в спеціальному керованому середовищі;

2) автоматичний збір сміття (Garbage Collection) і заборона на використання покажчиків. У C# допускається обмежене використання покажчиків в блоках коду, позначених як «ненадійні»;

3) об'єкти створюються за допомогою ключового слова `new`, виділення пам'яті здійснюється з «купи» (heap), що знаходиться в розпорядженні середовища виконання;

4) багатопотоковість підтримується шляхом блокування об'єктів;

- 5) інтерфейси, множинна реалізація інтерфейсів класом, одноразове наслідування базового класу похідним;
- 6) внутрішні класи;
- 7) відсутність глобальних функцій і констант, всі елементи повинні належати класам;
- 8) масиви і рядки з вбудованою довжиною і перевіркою кордонів;
- 9) не застосовуються оператори «->», «::». У всіх випадках використовується оператор «.»;
- 10) null і boolean / bool є ключовими словами;
- 11) будь-яка величина повинна бути проініціалізована до того, як буде використана;
- 12) не можна використовувати цілі числа (integers) для управління операторами if.

Хоч у мов C# і Java досить багато спільного, але все одно не можна їх ототожнювати. C# теоретично є таким же портативним, як Java, але сьогодні він підтримується на набагато меншій кількості платформ. Java дозволяє створювати незалежні від платформи програми шляхом компіляції в проміжне представлення, яке називається байткодом.

Переваги мови Java:

1. Одна з основних переваг мови Java – незалежність від платформи, на якій виконуються програми: один і той же код можна запускати під управлінням операційних систем Windows, Solaris, Linux, Macintosh та ін. Це дійсно необхідно, коли програми завантажуються через Інтернет для подальшого виконання під управлінням різних операційних систем.
2. Синтаксис мови Java схожий на синтаксис мови C++, і програмістам, що знають мови C і C++, його вивчення не складає труднощів.
3. Крім того, Java – повністю об'єктно-орієнтована мова, навіть більшою мірою, ніж C++. Всі сутності в мові Java є об'єктами, за винятком небагатьох основних типів (primitive types), наприклад чисел.

4. Висока надійність. Розробники забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найбільш поширені помилки. Для цього в мові Java зроблено наступне:

- виключена можливість явного виділення і звільнення пам'яті. Пам'ять у мові Java звільняється автоматично за допомогою механізму збірки сміття. Програміст застрахований від помилок, пов'язаних з неправильним використанням пам'яті;
- введені істинні масиви і заборонена арифметика покажчиків. Тепер програмісти в принципі не можуть стерти дані з пам'яті внаслідок неправильного використання покажчиків;
- виключена можливість переплутати оператор присвоювання з оператором порівняння на рівність. Тепер не можна навіть скомпілювати вираз `if (entries = 3);`
- виключено множинне наслідування. Воно замінене новим поняттям – інтерфейсом. Інтерфейс дає програмісту майже все, що той може отримати від множинного наслідування, уникаючи при цьому складнощів, що виникають при управлінні ієрархіями класів.

Зважаючи на всі вище перераховані переваги мови Java, вирішено здійснювати розробку мобільного додатку саме з її використанням.

1.8 Порівняльний аналіз середовищ програмування

Для розробки додатків на мові програмування Java найпопулярнішими є Eclipse і Android Studio.

Eclipse – вільномодульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для розробників на мові Java, засоби для управління сирцевими кодами, візуальні побудовники GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах

програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

Eclipse являє собою фреймворк для розробки модульних кросплатформових застосунків із низкою особливостей:

- можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);
- крос-платформова;
- модульна, призначена для подальшого розширення незалежним розробниками;
- з відкритим сирцевим кодом;
- розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland.

Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує підтримка VSS та інших). З огляду на безкоштовність, у багатьох організаціях Eclipse – корпоративний стандарт для розробки ПЗ на Java.

Друге призначення Eclipse – служити платформою для нових розширень. Такими стали C/C++ Development Tools (CDT), розроблювані інженерами QNX разом із IBM, засоби для підтримки інших мов різних розробників. Безліч розширень доповнює Eclipse менеджерами для роботи з базами даних, серверами застосунків та інших.

Eclipse – це універсальне середовище розробки, що підтримує багато мов програмування. Є й інші середовища розробки з підтримкою Android (Net Beans, наприклад), але для зручності використано середовище Eclipse з встановленим плагіном ADT. Розробка ведеться на мові програмування Java. Є можливість налагодження з використанням емулятора вбудованого в ADT або безпосередньо на мобільному пристрої з ОС Android. Для опису інтерфейсу додатків

використовується ще одна мова – XML (розширювана мова розмітки). ADT плагін дозволяє конструювати інтерфейс Android-додатків без написання XML коду вручну. Цей же плагін пов'язує середу Eclipse з емулятором Android-пристрою. Емулятор – це віртуальний аналог реального телефону з обмеженою функціональністю. Він необхідний для тестування написаних додатків.

Android Studio – це відносно нове середовище розробки Андроїд додатків, що базується на платформі IntelliJ IDEA компанії JetBrains. Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки додатків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання додатків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Для прискорення розробки додатків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу додатку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів.

Android Studio має деякі недоліки, а саме пов'язані зі швидкодією:

- 1) процеси виконуються не виправдано довго, оскільки постійно синхронізація з Gradle значно уповільнює компіляцію проектів, в порівнянні з Eclipse;
- 2) часті підвисання вікон програми, дуже повільно робота з SpellChecker.

Взявши до уваги недоліки Android Studio для розробки автоматизованої системи ведення сімейного бюджету обрано середовище програмування Eclipse з встановленим плагіном для розробки мобільних додатків під Android.

2. Розробка задачі алгоритму її розв'язування

2.1 Постановка задачі

В сучасному середовищі вагомим спрямуванням інформатизації регіонів є піднесення інформаційного ринку. Технічний та технологічний поступ вимагає від суспільства покращення комунікативних властивостей мобільних пристроїв. Це визначає появу інформаційних продуктів, що впливають на поліпшення якості життя, можуть вирішувати різноманітні завдання в певних сферах життєдіяльності. Самим найперспективнішим серед цих продуктів є мобільні додатки (mobile apps), які є окремим програмним забезпеченням для відповідної операційної системи мобільного пристрою.

Метою дипломного проекту є підвищення ефективності ведення та розрахунку фінансових витрат за рахунок створення сучасних інструментів обліку та аналізу особистих і сімейних фінансів та їх реалізації у мобільному додатку, що дозволяє автоматизувати процес моніторингу фінансового стану сім'ї та керувати фінансовими операціями із забезпеченням матеріальної стабільності сімейного бюджету.

Задля досягнення поставленої мети роботи необхідно вирішити такі задачі:

- проаналізувати особливості розробки бази даних для мобільних додатків;
- розробити структуру бази даних;
- розробити моделі мобільного додатку;
- розробити структуру мобільного додатку;
- розробити базу даних автоматизованої системи розрахунку фінансових витрат;
- розробити алгоритм функціонування мобільного додатку;
- розробити інтерфейс автоматизованої системи;
- здійснити програмну реалізацію мобільного додатку;

- протестувати розроблений програмний продукт.

Проект призначений для спрощення процесу ведення особистого та сімейного бюджету, внесення доходів та витрат. Додаток дозволяє керувати щоденними та місячними витратами, планувати бажані покупки з виконанням принципів збереження стабільності фінансового стану сім'ї з урахуванням стратегії ведення особистих та сімейних видатків, що допомагає користувачу не перевищувати щомісячний бюджет. Також присутня можливість внесення коштів на загальний чи персональний рахунок, створення груп користувачів, які можуть вносити кошти на загальний рахунок.

2.2 Аналіз принципів розробки мобільних додатків

Android використовує спеціальні механізми опису дій засновані на Intent. При необхідності виконати дію (виклик, відправка пошти, SMS) викликається Intent. Для обміну даними між додатками використовується Content Providers. Зазвичай для створення проекту використовуються Android SDK, Java і IDE.

Кожна мова програмування – унікальна і копіює методи і класи. У парадигмі програмування, кожна з мов пов'язана одна з однією концепціями, принципами і абстракціями, що визначають фундаментальний стиль програмування. Рідними мовами для Android є мови розмітки XML, C ++ і C #.

Ефективні методи програмування під Android:

- програмування під Android – це Java, додатки якої транслуються в байт-код, що виконується віртуальною Java-машиною JVM;
- використання розмітки RelativeLayout і властивість «fill_parent» для цієї розмітки;
- використання порожніх елементів розмітки TextField з нульовою висотою і шириною з параметром «centerInParent» рівним «true» для вирівнювання елементів по центру екрану;
- використання елегантних способів доступу до даних, таких як values [SensorManager.DATA_X];

- використання методів `onPause()/onResume()` для збереження або закриття всього того, що цього вимагає;
- використання кольорової розмітки для об'єктів. Встановлення кольору фону для деяких об'єктів. Це дозволяє виділити помилки і знаходити їх швидше;
- використання IDE, яка відповідає особистим уподобанням і вимогам розробника. Eclipse – IDE класу «все в одному» для якої існує безліч доповнень. Eclipse має розширений візуальний редактор інтерфейсу, є можливість редагування XML-файлів в GUI за принципом конструктора, просто збираючи Layout-и із запропонованих «деталей»; плагін Eclipse MAT допоможе знайти витoki в пам'яті проекту; IntelliJ IDEA одна з найбільш швидких і зручних IDE;
- використання кількох робочих моніторів для програмування. Додавання вікна додатків на кількох екранах (вікно IDE, емулятор, прев'ю і документація);
- використання команди Source - Format для форматування XML-файлів для приведення їх у читабельний вигляд;
- використання плагіну XML Tools для Notepad++ для швидкого редагування XML-файлів і розуміння структури коду;
- використання Intents за допомогою окремого методу;
- уникнення створення безлічі об'єктів. Використання існуючих об'єктів;
- використання пулів потоків (thread pools), вбудованих класів AsyncTask;
- зберігання даних в SQLite, якщо вони займають великий об'єм;
- використання шаблонів (заздалегідь заготовлені шматочки коду), якими можна швидко скористатися для вставки в проект при натисканні на Ctrl + Space.

Існують принципи розробки продуктивних додатків під Android. Основні з них такі: необхідно економити апаратні ресурси, ефективно працювати з

виділенням пам'яті, протоколювати і аналізувати хід виконання додатку, уникати зайвих об'єктів і створювати методи статичними; для констант, класів необхідно використовувати модифікатор `static final` і не використовувати `enum` там, де модифікатор не вписується.

2.3 Аналіз особливостей розробки бази даних

База даних із правильною структурою надає актуальні й точні відомості. Оскільки правильна структура вкрай важлива для досягнення цілей під час роботи з базою даних, варто присвятити трохи часу вивченню принципів правильної структури. Зрештою, це збільшить шанси отримати базу даних, яка відповідатиме вашим потребам і легко пристосовуватиметься до змін.

Сучасні інформаційні технології побудовані на принципах взаємодії з базами даних. Бази даних грають велику роль у житті більшості людей. Вони поширені всюди, починаючи від невеликих підприємств, які утримують власні бази товарів, працівників, партнерів, постачальників і закінчуючи базами даних глобального масштабу – держави чи кількох держав, наприклад, реєстраційні дані про людину чи автомобіль. Бази даних дозволяють ефективно розмістити та упорядкувати дані, що дозволить отримати ефективний пошук та взаємодію. Тому вони займають провідне місце на ринку програмних продуктів.

В розробленій автоматизованій системі необхідно ефективно зберігати інформації про бухгалтерський облік. Для вирішення цієї задачі у мобільному додатку необхідно використати концепцію баз даних.

База даних – сукупність даних, що зберігаються у відповідності зі схемою даних і відображають стан об'єктів та їх взаємозв'язків, маніпулювання якими виконуються відповідно до правил засобів моделювання даних.

Бази даних є потужним і зручним способом зберігання і користування інформацією. Вони життєво необхідні у величезній кількості учбових закладів, установ, організацій, підприємств. Важливість баз даних важко переоцінити.

Реляційна база даних — колекція пов'язаних таблиць. Різниця між базою даних і реляційною базою даних знаходиться в будові таблиць. Якщо вести облік компанії на серії таблиць Excel, то робота йде тільки з базою даних. Можна отримувати інформацію з кожної таблиці, але тоді не буде можливості використовувати інформацію з однієї таблиці в якості основи для запиту інформації, що міститься на іншому аркуші. У реляційній базі даних, таблиці побудовані так, що є логічний зв'язок між ними. На підставі інформації, яку знайдено в одній таблиці, можна отримати відповідну інформацію з іншої таблиці.

В Android використовується база даних SQLite — це досить популярна, легка і швидка реляційна база даних. Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремим процесом, з яким взаємодіє додаток, а надає бібліотеку, з якою програма компілюється і движок стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому пристрої, на якому виконується додаток. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. У комплекті постачання йде також функціональна клієнтська частина у вигляді

виконуваного файлу `sqlite3`, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі движка можливо використовувати SQLite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite:

- транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення;
- встановлення без конфігурації – не потребує ані установки, ані адміністрування;
- база даних зберігається в одному кросплатформовому файлі на диску;
- підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів (Binary Large Object);
- малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями;
- швидший за популярні движки клієнт-серверних баз даних для найпоширеніших операцій;
- простий, легкий у використанні API;
- написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов та гілок;
- добре прокоментований сирцевий код зі 100% тестовим покриттям доступний як єдиний файл сирцевого коду на ANSI C, який можна легко вставити в інший проект;
- автономність (немає зовнішніх залежностей);
- кросплатформеність: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE). Легко переноситься на інші системи;
- поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

Думаючи про побудову різних таблиць, які потрібні в нашій базі даних, необхідно розглянути, як організувати ці таблиці і зв'язати одна з одною. Здебільшого, це буде зроблено за допомогою ключів. Є 2 типи ключів в таблиці:

- первинний ключ – значення, що є унікальним для кожного запису;
- зовнішній ключ – первинний ключ однієї таблиці знаходиться в іншій таблиці.

Використання SQLite дає розробнику і користувачеві кілька незаперечних переваг:

- не потрібно встановлювати додаткове програмне забезпечення;
- дані зберігаються в локальному файлі, що створює прозорість їх контролю, тобто їх можна переглядати і редагувати незалежно від вашого додатку;
- можливість імпорту або експорту таблиць в інші СУБД;
- в коді використовуються звичні SQL запити, що дозволяє в будь-який момент перевести додаток на роботу з іншими СУБД.

2.4 Розробка моделей мобільного додатку

Мобільні розробники – це ті люди, які вдихають життя в бездушні мобільні коробочки, якісь шамани-заклинателі, які говорять іноземною мовою і можуть практично все, а іноді це дійсно так. Але також це ті люди, які рухають прогрес вперед, створюючи все більш досконалі та складні програми. Хороший програміст – це не тільки знання мов і технологій, це ще і навички системного мислення, так як йому доводиться з нічого створювати складні системи, це також вміння працювати в команді, оскільки серйозний проект складно написати поодиночку, це і вміння спілкуватися і розуміти замовника, щоб створити саме той продукт, який він хоче.

Для роботи мобільного додатку необхідна локальна база даних для зберігання інформації про бюджети користувачів, яка синхронізується з серверною базою в хмарному середовищі. Синхронізація здійснюється через web-сервіс за допомогою HTTP-запитів при наявності підключення до інтернету. За допомогою

мобільного додатку відображується інформація користувача, і він може маніпулювати своїми даними.

Створення мобільних додатків може бути такого типу:

Інформаційні додатки. Основна мета — інформування користувача (потенційного клієнта). Розповідає про новинки компанії. Автономно від ресурсу допускається виключно новинне додаток, орієнтоване на певний сегмент новин або ж на все відразу. Використовується для реклами — інформаційні додатки одні з самих благодатних майданчиків розміщення реклами серед додатків.

Служби. Додатки, які переносять онлайн-ресурс на мобільний телефон. Підійде банкам, компаніям, що надають послуги онлайн. Не підійде таке створення мобільних додатків для продажу рекламних майданчиків, основна мета — зручність послуг клієнтам. Підвищує репутацію компанії.

Ігри. Лідер додатків сьогодні. Підійдуть для продажу рекламних майданчиків, вірусна гра може захопити користувача надовго: поїздки в метро, нудні лекції — кожна вільна хвилина з грою. Заробляти можна платним контентом або рекламою. Створення мобільних додатків такого типу — привабливий дизайн, захоплюючий сюжет. Складний вид розробки додатків для iOS або Android.

Додаток. Використовується супутнім елементом до техніки. Може бути пультом від нового телевізора, помічником управління автомобілем, стереосистемою будинку, іншої побутової технікою. Тип — схожий на службові, але носить більш розважальний характер.

Інтернет-магазин. Тепер онлайн торгівля стала доступнішою. Розробка програми для Android — це зростання продажів на третину (за статистикою більше половини користувачів СНД використовують продукт від Google на мобільних девайсах). Основна мета — можливість швидко зробити покупку, привабливий інтерфейс, оплата за допомогою тих же мобільних банківських служб.

Вищезазначені програми — це лише найпопулярніші види. Створення мобільних додатків обмежується тільки вашою фантазією і вимогами.

На рисунку 2.1 наведено загальну модель роботи мобільного додатку.

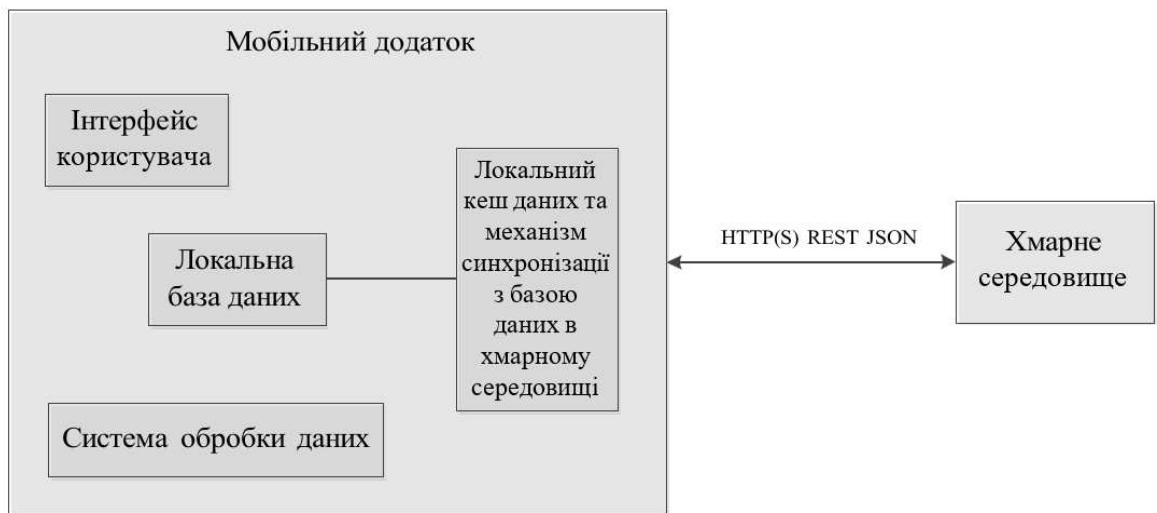


Рис. 2.1 – Модель роботи мобільного додатку

2.4.1 Розробка структури бази даних

Однією з обов'язкових умов при розробці баз даних є її нормалізація. Нормалізація схеми бази даних – покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Нормальна форма – властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми n , та всім попереднім нормальним формам, тоді вважається, що це відношення знаходиться у нормальній формі рівня n .

У роботі базу даних приведено до третьої нормальної форми, тобто, тоді вона відповідає вимогам першої та другої нормальних форм, а саме:

- перша нормальна форма:

1) кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис;

2) уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів у різних записах), правильно визначаючи не-ключові атрибути;

3) атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень;

– друга нормальна форма:

1) схема бази даних повинна відповідати вимогам першої нормальної форми;

2) дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці;

– третя нормальна форма:

1) схема бази даних повинна відповідати всім вимогам другої нормальної форми;

2) будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

На рисунку 2.2 зображено структуру бази даних автоматизованої системи розрахунку фінансових витрат.

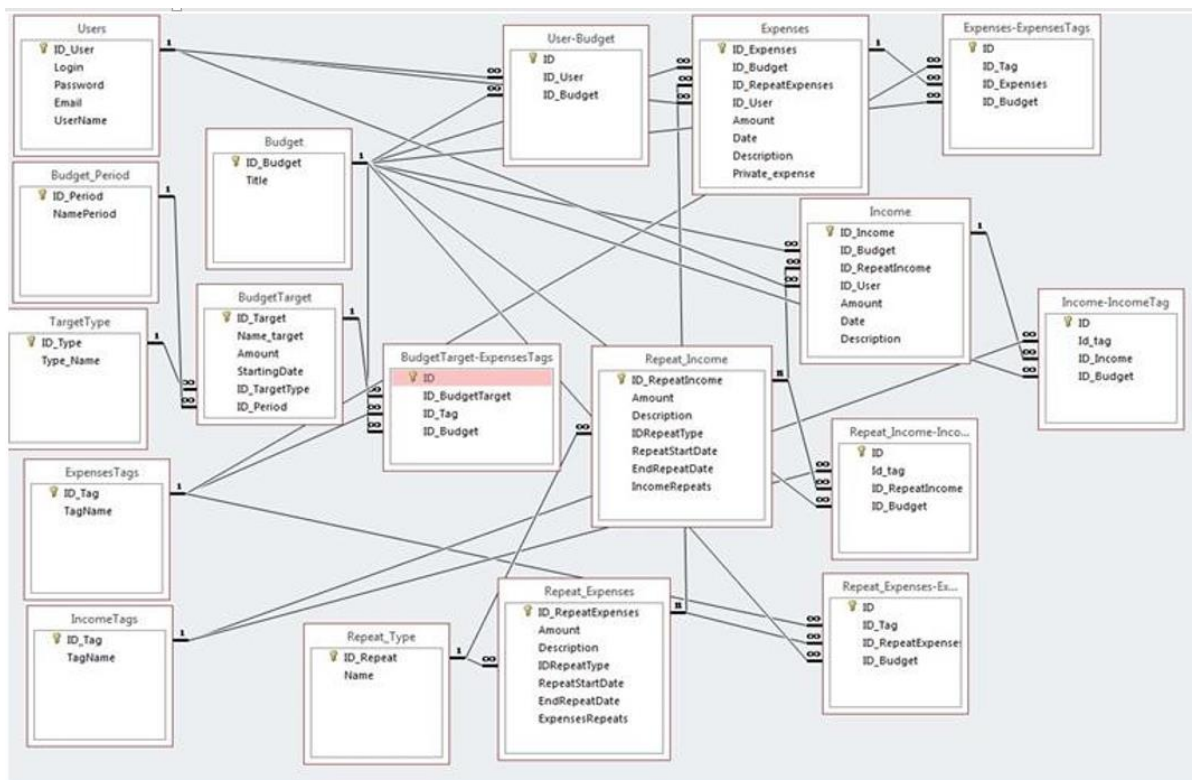


Рис. 2.2 – Структура бази даних

2.4.2 Розробка ієрархічної моделі мобільного додатку

У плані обробки взаємодії між інтерфейсом користувача і його логікою Android слідує архітектурному шаблону «Model-View-ViewModel» (MVVM).

Model-View-ViewModel – це шаблон проектування додатків для розділення коду інтерфейсу користувача і іншого коду. За допомогою MVVM декларативно визначається інтерфейс користувача (у нашому випадку, використовуючи XML) і використовується розмітка прив'язки даних, щоб пов'язати його з іншими рівнями, що містять дані і команди користувача.

Шаблон MVVM організовує код так, що можна змінювати окремі його частини, не впливаючи на інші. Це дає багато переваг, серед яких:

- можливість використання ітеративного, довільного стилю написання коду;
- спрощене тестування модулів;
- більш ефективне використання інструментів проектування;

- підтримка взаємодії в команді.

При використанні шаблону MVVM Android-додаток ділиться на такі частини:

1. Інтерфейс, який розробляється за допомогою технології XML.
2. Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel.
3. Функціональні зв'язки між інтерфейсом користувача і ViewModel реалізуються через Біндинг (bindings). Біндинги можуть бути написані в коді або визначені декларативним шляхом (Android використовує обидва типи).

View – базовий клас для всіх віджетів користувацького інтерфейсу. Інтерфейс Android-додатку являє собою дерево екземплярів нащадків цього класу.

Клас Activity і його підкласи містять логіку, що реалізує інтерфейс користувача. Цей клас відповідає ViewModel в архітектурному шаблоні Model-View-ViewModel (MVVM). Відношення між підкласом Activity і інтерфейсом користувача – це відношення один до одного; зазвичай кожен підклас Activity має тільки один пов'язаний з ним користувацький інтерфейс, і навпаки.

У дипломному проекті мобільний додаток містить п'ять рівнів моделей представлення підкласів класу Activity (рисунки 2.3). Перший рівень – це вікно програми, де користувач повинен ввести свій логін і пароль. Якщо ж він не зареєстрований, то йому спочатку необхідно перейти на другий рівень – рівень реєстрації користувача, а потім повернутися на перший. Третій рівень додатку – рівень бюджетів користувача. Тут він може додавати і видаляти бюджети або перейти на вже існуючий бюджет. Наступний рівень – рівень обліку фінансів. Користувач може додавати здійснені ним витрати або отримані доходи. Також тут є можливість поставити ліміт бюджету на місяць. Обравши одну з цих функцій, користувач переходить до останнього п'ятого рівня, де він додає детальну інформацію про витрати, доходи чи ліміт бюджету.

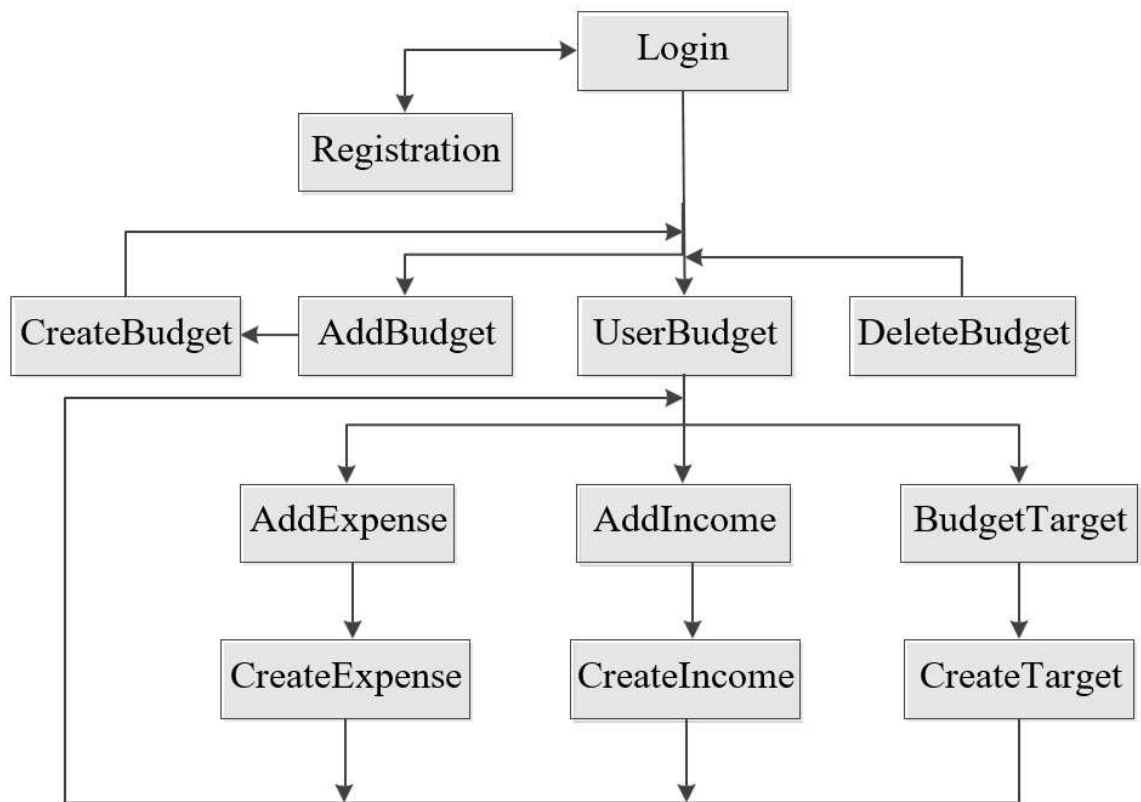


Рис. 2.3 – Ієрархічна модель мобільного додатку

2.4.3 Розробка алгоритму роботи мобільного додатку

Алгоритм – набір інструкцій, що описують порядок дій виконавця для досягнення результату рішення задачі за визначене число дій.

Для створення алгоритму необхідно знати:

- повний набір вихідних даних завдання (початковий стан об'єкта);
- мета створення алгоритму (кінцевий стан об'єкта);
- систему команд виконавця (тобто набір команд, які виконавець розуміє і може виконати).

Існує два методи розробки складних алгоритмів.

Метод послідовної деталізації завдання («зверху-вниз») полягає в тому, що вихідна складна задача розбивається на підзадачі. Кожна з підзадач розглядається і вирішується окремо. Якщо які-небудь з підзадач складні, вони також

розбиваються на підзадачі. Процес продовжується до тих пір, поки підзадачі не зведуться до елементарних. Вирішення окремих підзадач потім збираються в єдиний алгоритм розв'язання вихідної задачі. Метод широко використовується, тому що дозволяє вести розробку загального алгоритму одночасно кільком програмістам, які вирішують локальні підзадачі. Це необхідна умова швидкої розробки програмних продуктів.

Складальний метод («знизу-вверх») полягає у створенні безлічі програмних модулів, що реалізують рішення типових задач. При вирішенні складного завдання програміст може використовувати розроблені модулі в якості допоміжних алгоритмів (процедур).

Кожний алгоритм повинен відповідати ряду загальних вимог [24], а саме:

1) дискретність – алгоритм повинен представляти процес вирішення завдання як послідовне виконання деяких простих кроків. При цьому для виконання кожного кроку алгоритму потрібно кінцевий відрізок часу, тобто перетворення вихідних даних у результат здійснюється в часі дискретно;

2) детермінованість (визначеність) – у кожен момент часу наступний крок роботи однозначно визначається станом системи. Таким чином, алгоритм видає один і той же результат (відповідь) для одних і тих же початкових даних. У сучасному трактуванні у різних реалізацій одного і того ж алгоритму має бути ізоморфний граф. З іншого боку, існують імовірнісні алгоритми, в яких наступний крок роботи залежить від поточного стану системи і випадкового числа, що генерується. Однак при включенні методу генерації випадкових чисел в список «початкових даних», імовірнісний алгоритм стає підвидом звичайного;

3) зрозумілість – алгоритм повинен включати тільки ті команди, які доступні виконавцю і входять в його систему команд;

4) завершеність – при коректно заданих початкових даних алгоритм повинен завершувати роботу і видавати результат за кінцеве число кроків.

5) масовість (універсальність) – алгоритм повинен бути застосовний до різних наборі початкових даних;

6) результативність – завершення алгоритму певними результатами;

7) алгоритм містить помилки, якщо призводить до отримання неправильних результатів або не дає результатів зовсім;

8) алгоритм не містить помилок, якщо він дає правильні результати для будь-яких допустимих початкових даних.

Алгоритми в залежності від мети, початкових умов завдання, шляхів її вирішення, визначення дій виконавця поділяються наступним чином:

- механічні алгоритми – задають певні дії, позначаючи їх в єдиній і достовірній послідовності, забезпечуючи тим самим однозначний необхідний або шуканий результат, якщо виконуються ті умови процесу, завдання, для яких розроблений алгоритм;
- гнучкі алгоритми, наприклад стохастичні, тобто імовірнісні та евристичні;
- імовірнісний алгоритм дає програму рішення задачі кількома шляхами або способами, що призводять до ймовірного досягненню результату;
- евристичний алгоритм (від грецького слова «еврика») – алгоритм, що використовує різні розумні міркування без строгих обґрунтувань;
- лінійний алгоритм – набір команд (вказівок), виконуваних послідовно в часі один за одним;
- розгалужений алгоритм – алгоритм, який містить хоча б одну умову, в результаті перевірки якої може здійснюватися поділ на кілька паралельних гілок алгоритму;
- циклічний алгоритм – алгоритм, що передбачає багаторазове повторення однієї і тієї ж дії (одних і тих же операцій) над новими вихідними даними. Цикл програми – послідовність команд, яка може виконуватися багато разів (для нових початкових даних) до задоволення деякої умови;
- структурна блок-схема, граф-схема алгоритму – графічне зображення алгоритму у вигляді схеми пов'язаних між собою за допомогою стрілок (ліній переходу) блоків – графічних символів, кожен з яких відповідає одному кроку алгоритму. В середині блоку дається опис відповідної дії. Графічне зображення алгоритму широко використовується перед програмуванням завдання внаслідок його наочності.

Алгоритм розробленої автоматизованої системи є розгалуженим, так як містить декілька умов, в результаті перевірки яких здійснюється поділ на кілька паралельних гілок алгоритму. Блок-схема алгоритму блоку реєстрації та авторизації користувача наведена на рисунку 2.4.

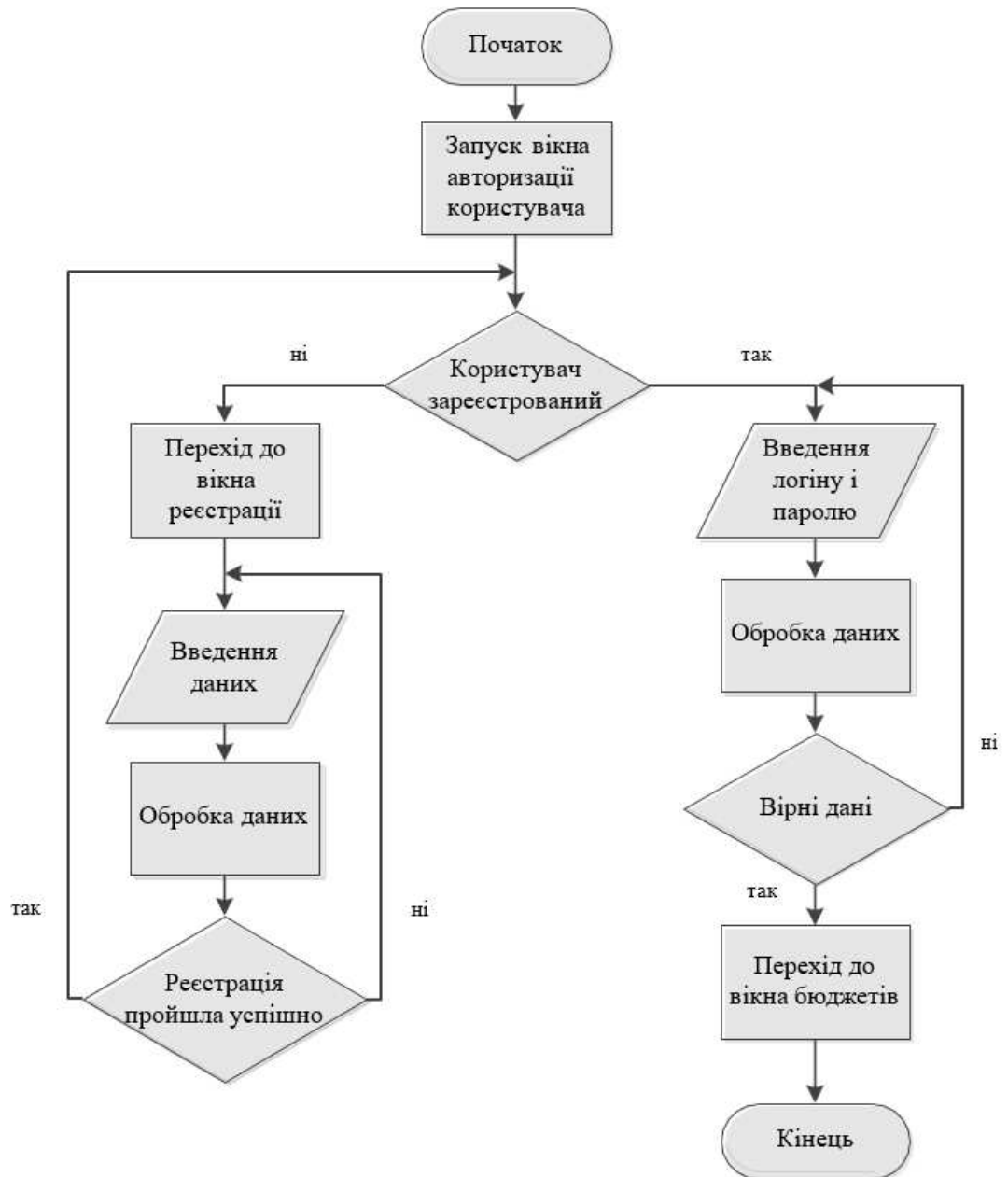


Рис. 2.4 – Блок-схема алгоритму авторизації користувача

Блок-схема алгоритму роботи блоку реєстрації та авторизації користувача, належить до розгалуженого типу.

3. Реалізація інформаційного, організаційного, технічного та програмного забезпечення

3.1 Розробка бази даних автоматизованої системи

В Android існує вбудована підтримка бази даних SQLite. Підтримуються всі функції SQLite, надається API оболонки із сумісним інтерфейсом. API Android SQLite є типовим; розробнику слід реалізувати всю обробку бази даних, включаючи створення, управління версіями, оновлення бази даних та інші настройки.

У додатку, при підключенні до бази даних необхідно вказати ім'я бази даних і її версію. При цьому можуть виникнути такі ситуації:

1) бази даних не існує. Це може бути наприклад у випадку первинної установки програми. У цьому випадку додаток повинно сам створити базу даних і всі таблиці в ній. І далі він вже працює з щойно створеною базою;

2) база даних існує, але її версія застаріла. Це може бути у випадку поновлення програми. Наприклад новій версії програми потрібні додаткові поля в старих таблицях або нові таблиці. У цьому випадку додаток повинен оновити існуючі таблиці та створити нові, якщо це необхідно;

3) база даних існує і її версія актуальна. У цьому випадку додаток успішно підключається до бази даних і працює.

Для обробки описаних вище ситуацій необхідно створити клас, який є нащадком для SQLiteOpenHelper (клас для створення бази даних). SQLiteOpenHelper містить два абстрактних методи [25]:

- onCreate – метод, який буде викликаний, якщо база даних, до якої треба підключитися, не існує;
- onUpgrade – буде викликаний при модифікації бази даних.

Створений клас-нащадок DBHelper повинен реалізувати перераховані вище методи, вписавши в них логіку створення і модифікації бази даних.

Клас DBHelper також неявно наслідує інтерфейс BaseColumns, в якому визначена строкова константа `_id`, що представляє ім'я поля для ідентифікаторів записів. У створюваних таблицях бази даних поле `_id` повинно мати тип `integer primary key autoincrement`. Описувач `autoincrement` є необов'язковим.

У методі зворотного виклику `onCreate()` необхідно реалізувати логіку створення таблиць. Метод `onUpdate()` викликається під час встановлення оновлень програми зі зміненою структурою таблиць.

Частина коду створення таблиць бази даних наведено нижче:

```
private static final String DATABASE_NAME = "FBdatabase.db";
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_TABLE_USERS = "Users";
private static final String DATABASE_TABLE_BPERIOD =
"Budget_Period"; private static final String DATABASE_TABLE_TTYPE =
"TargetType"; private static final String DATABASE_TABLE_BUDGET =
"Budget";

static final String LOGIN_COLUMN = "login"; static final String
PASSWORD_COLUMN = "password"; static final String EMAIL_COLUMN =
"email";

static final String USER_NAME_COLUMN = "user_name"; static final String
NAME_PERIOD_COLUMN = "name_period"; static final String
NAME_TYPE_COLUMN = "name_type"; static final String TITLE_COLUMN =
"title";

private static final String DATABASE_CREATE_SCRIPT_USERS = "create
table "
+ DATABASE_TABLE_USERS + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + LOGIN_COLUMN
+ " text not null, " + PASSWORD_COLUMN + " text not null, " +
EMAIL_COLUMN
+ " text not null," + USER_NAME_COLUMN + "text not null);";
```

```

        private static final String DATABASE_CREATE_SCRIPT_BPERIOD =
"create table "
        + DATABASE_TABLE_BPERIOD + " (" + BaseColumns._ID
        + " integer primary key autoincrement, " + NAME_PERIOD_COLUMN
        + " text not null);";

        private static final String DATABASE_CREATE_SCRIPT_TTYPE = "create
table "
        + DATABASE_TABLE_TTYPE + " (" + BaseColumns._ID
        + " integer primary key autoincrement, " + NAME_TYPE_COLUMN
        + " text not null);";

        private static final String DATABASE_CREATE_SCRIPT_BUDGET =
"create table "
        + DATABASE_TABLE_BUDGET + " (" + BaseColumns._ID
        + " integer primary key autoincrement, " + TITLE_COLUMN
        + " text not null);";

        @Override

        public void onCreate(SQLiteDatabase db) {
            db.execSQL(DATABASE_CREATE_SCRIPT_USERS);
            db.execSQL(DATABASE_CREATE_SCRIPT_BPERIOD);
            db.execSQL(DATABASE_CREATE_SCRIPT_TTYPE);
            db.execSQL(DATABASE_CREATE_SCRIPT_BUDGET);
        }

        @Override

        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_USERS);
            db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_BPERIOD);
            db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_TTYPE);
            db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_BUDGET);
            onCreate(db);
        }

```

3.2 Розробка мобільного додатку

3.2.1 Реалізація логіки програмного продукту

Технічні вимоги:

- операційна система – Android;
- мова програмування – Java;
- середовище розробки – Eclipse.

При розробці додатку під Android головним класом є клас Activity. Він представляє візуальну активність додатку і визначає дії, які може виконувати користувач. У процесі роботи класу Activity, спочатку створюється об'єкт класу Activity, потім він запускається, відпрацьовує та знищується, а користувач смартфона переходить до нового об'єкта.

Протягом життєвого циклу Activity може перебувати в одному з трьох станів:

1. Активне і виконується – цей користувацький інтерфейс знаходиться на передньому плані (на вершині стека Activity).
2. Призупинене – якщо даний інтерфейс користувача втратив фокус, але все ще видимий. У такому стані ніякий код не виконується.
3. Завершене – якщо інтерфейс користувача невидимий. У такому стані код не виконується.

Процес роботи класу Activity наведено на рисунку 3.1.

При створенні нової активності, наприклад, при запуску програми, Android викликає метод onCreate. У цьому методі проводиться початкове налаштування activity. Зокрема, створюються об'єкти візуального інтерфейсу. Цей метод отримує об'єкт Bundle, який містить колишній стан activity, якщо він був збережений. Якщо activity заново створюється, то даний об'єкт має значення null. Якщо ж activity вже раніше була створена, але перебувала у загальмованому стані, то bundle містить зв'язану з activity інформацію.

робота активності викликом методу `OnDestroy`, який виникає або, якщо система вирішить вбити `activity`, або при виклику методу `finish()`.

При проектуванні та розробці програмного забезпечення має відзначатися його швидкість та ефективність роботи. Дуже важливо постійно приділяти увагу оптимізації програмного забезпечення на всіх етапах його розробки.

Запустивши розроблений додаток, першою користувач бачить `LoginActivity` – вікно авторизації користувача, де йому необхідно ввести свій логін і пароль для подальшої роботи з додатком. Частина коду реалізації `LoginActivity` наведена нижче:

```
public void onClick(View view) {
    String login = inputLogin.getText().toString().trim(); String password =
inputPassword.getText().toString().trim();
    if (!login.isEmpty() && !password.isEmpty()) { checkLogin(login, password);
    } else {
        Toast.makeText(getApplicationContext(),
        "Please enter the credentials!", Toast.LENGTH_LONG)
        .show();
    }
}

private void checkLogin(final String login, final String password) { String
tag_string_req = "req_login";

    StringRequest strReq = new StringRequest(Method.POST,
AppConfig.URL_LOGIN, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Log.d(TAG, "Login Response: " + response.toString()); hideDialog();
            try {
                JSONObject jsonObj = new JSONObject(response);
                boolean error = jsonObj.getBoolean("error");
                if (!error) {
```

```

session.setLogin(true);
String uid = jsonObj.getString("uid");
JSONObject user = jsonObj.getJSONObject("user"); String name =
user.getString("name");
String email = user.getString("login"); String created_at = user
.getString("created_at"); db.addUser(name, login, uid, created_at);
Intent intent = new Intent(LoginActivity.this, BudgetActivity.class);
startActivity(intent); finish();
}

```

Спочатку відбувається перевірка на те, чи не пусті поля для вводу даних за допомогою конструкції if-else. Якщо поля заповнені, то викликається функція для перевірки логіна і пароля. Використовуючи екземпляр класу StringRequest, посилається запит в базу на перевірку правильності введених даних. Якщо дані вірні, то відкривається вікно бюджетів користувача.

3.2.2 Реалізація розмітки Android-додатку

Під час розробки Android-додатку, візуальна частина вікон реалізується окремо від логіки програми. Код розмітки екрану знаходиться у файлах з розширенням .xml у папці проекту, яка має назву «layout».

Розмітка для Android програми будується з використанням ієрархії View і ViewGroup об'єкти. View об'єкти це віджети для інтерфейсу користувача, такі як кнопки або текстові поля і ViewGroup це невидимий вид контейнерів, які визначають розташування дочірніх представлень, як наприклад, в сітці або вертикальному списку.

Існує кілька стандартних типів розміток:

FrameLayout – є найпростішим типом розмітки. Зазвичай це порожній простір на екрані, який можна заповнити тільки дочірніми об'єктами View або ViewGroup. Усі дочірні елементи FrameLayout прикріплюються до верхнього лівого кута екрана. У розмітці FrameLayout не можна визначити різне

розташування для дочірнього об'єкта. Наступні дочірні об'єкти View будуть просто малюватися поверх попередніх компонентів, частково або повністю затінюючи їх, якщо об'єкт, що знаходиться зверху, непрозорий, тому єдиний дочірній елемент для `FrameLayout` зазвичай розтягнутий до розмірів батьківського контейнера.

1. `LinearLayout` – вирівнює всі дочірні об'єкти в одному напрямку – вертикально або горизонтально. Напрямок задається за допомогою атрибута орієнтації `android: orientation`. Усі дочірні елементи поміщаються в стек один за іншим, так що вертикальний список представлень буде мати тільки один дочірній елемент в рядку незалежно від того, наскільки широким він є. Горизонтальне розташування списку буде розміщувати елементи в один рядок з висотою, рівній висоті найвищого дочірнього елемента списку.

2. `TableLayout` – позиціонує свої дочірні елементи в рядки і стовпці, як це звикли робити веб-майстри в тезі `table`. `TableLayout` не відображає лінії обрамлення для їх рядків, стовпців або комірок. `TableLayout` може мати рядки з різною кількістю комірок. При формуванні розмітки таблиці деякі комірки при необхідності можна залишати порожніми. При створенні розмітки для рядків використовуються об'єкти `TableRow`, які є дочірніми класами `TableLayout` (кожен `TableRow` визначає єдиний рядок в таблиці). Рядок може не мати комірок або мати одну і більше комірок, які є контейнерами для інших об'єктів. У комірку допускається вкладати інший `TableLayout` або `LinearLayout`.

3. `RelativeLayout` – дозволяє дочірнім компонентам визначати свою позицію щодо батьківського компонента або відносно сусідніх дочірніх елементів (ідентифікатора елемента). У `RelativeLayout` дочірні елементи розташовані так, що якщо перший елемент розташований по центру екрана, інші елементи, вирівняні відносно першого елемента, будуть вирівняні відносно центру екрана. При такому розташуванні, при оголошенні розмітки в XML- файлі, елемент, на який будуть посилатися для позиціонування інші об'єкти представлення, має бути оголошений раніше, ніж інші елементи, що звертаються до нього за його ідентифікатором.

`GridLayout` – На перший погляд він може здатися схожим на `TableLayout`. Але насправді він набагато зручніший і функціональніший. Розмітка відноситься до

класу `android.widget.GridLayout` і має колонки, рядки, клітинки як в `TableLayout`, але при цьому елементи можуть гнучко налаштовуватися. У `GridLayout` для будь-якого компонента можна вказати рядок і колонку, і в цьому місці таблиці він буде розміщений. Вказувати елемент для кожної комірки не знадобиться, це потрібно робити тільки для тих комірок, де дійсно повинні бути компоненти. Компоненти можуть розтягуватися на кілька елементів таблиці. Більш того, в одну комірку можна помістити кілька компонентів.

Всі описані розмітки є підкласами `ViewGroup` і успадковують властивості, визначені в класі `View`.

Для підключення створеної розмітки використовується код в методі `onCreate()`:

```
setContentView(R.layout.main),
```

де `main` – назва головної розмітки.

Код розмітки класу `LoginActivity` наведений нижче:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:background="@drawable/login"
tools:context="com.example.familybudget.LoginActivity" >

    <EditText

        android:id="@+id/editText2"                                android:layout_width="206dp"
        android:layout_height="wrap_content"                    android:layout_marginLeft="40dp"
        android:layout_marginTop="181dp"                        android:ems="10"                android:hint="Password"
        android:inputType="textPassword"                        android:textSize="15dp"
        android:visibility="visible" />

    <EditText
```

```

        android:id="@+id/editText1"                android:layout_width="206dp"
        android:layout_height="wrap_content"        android:layout_marginLeft="40dp"
        android:layout_marginTop="148dp"            android:ems="10"            android:hint="Login"
        android:textSize="15dp"
        android:visibility="visible" />
        <ImageButton
            android:id="@+id/imageButton2"            android:layout_width="wrap_content"
            android:layout_height="wrap_content"      android:layout_alignRight="@+id/editText2"
            android:layout_alignTop="@+id/imageButton1" android:layout_marginRight="32dp"
            android:onClick="goToRegistration" android:background="@drawable/button2" />
        <ImageButton
            android:id="@+id/imageButton1"            android:layout_width="wrap_content"
            android:layout_height="wrap_content"      android:layout_below="@+id/editText2"
            android:layout_marginRight="26dp"          android:layout_marginTop="18dp"
            android:layout_toLeftOf="@+id/imageButton2"
            android:background="@drawable/button1" android:onClick="Logging" />
    </RelativeLayout>

```

LoginActivity реалізовано за допомогою розмітки RelativeLayout, так як у нашому випадку її найзручніше використовувати.

Оголошення інтерфейсу користувача в файлах XML дозволяє відокремити інтерфейс додатку від коду, що означає, що можна змінювати визначення інтерфейсу без зміни і перекомпіляції коду. Наприклад, у додатку можуть бути визначені розмітки у файлах XML для різних орієнтацій екрану, різних розмірів пристроїв, різних мов і т.д. Крім того, оголошення розмітки в XML дозволяє легше візуалізувати структуру інтерфейсу і полегшує налагодження.

3.3 Розробка інтерфейсу програмного продукту

Розробка дизайну і інтерфейсу мобільного додатку – найважливіший етап створення програмного продукту. Від даної роботи залежить те, як користувач сприйматиме 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 повинен бути максимально простий і зрозумілий. З огляду на те, що мобільні пристрої мають невеликий за розмірами екран, при проектуванні додатків для них не можна керуватися тими ж правилами, що і для ПК. Основні вимоги до дизайну інтерфейсу мобільних додатків:

- мінімум елементів. Не слід перевантажувати невеликий простір дисплея мобільного пристрою великою кількістю об'єктів. Необхідно намагатися вмістити максимум функціоналу в лаконічний і дружній інтерфейс. Ця вимога є головною для розробки будь-якого ПЗ, а не тільки мобільного;
- управління сучасними телефонами з сенсорними екранами здійснюється за допомогою чуттєвого дотику. З огляду на те, що площа дотику пальця значно більше розмірів покажчика комп'ютерної миші, а також стилуса, інтерфейс не повинен містити дрібних елементів. По-перше, вони погано помітні на невеликому екрані. По-друге, торкаючись деякого ділянки дисплея, користувач може натиснути не той елемент, який йому потрібен, що в кращому випадку може призвести до зниження зручності використання програми, а в гіршому – до небажаних наслідків;
- розмір всіх написів повинен бути достатнім для того, щоб користувач міг

прочитати їх з відстані не менше 30 см;

- найбільш 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 мобільний платформу можна обмежитися створенням зменшеної копії додатка. Необхідно оптимізувати весь інтерфейс, прибрати всі зайві елементи, згрупувавши 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, що займають весь простір екрану пристрою. Користувач здійснює переходи між такими вікнами за допомогою графічних елементів- навігаторів, або перетягуючи їх за допомогою пальця (в залежності від тієї або іншої платформи і переваг творців додатка).

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). Найбільш популярними мобільними платформами є iPhone і Android. Вони мають багато спільного. Грамотно спроектований додаток буде включати в себе той функціонал, який використовує особливості кожної з них.

2. Мобільний додаток повинен бути цікавим користувачу. Саме тому найбільшою популярністю на сьогодні користуються мобільні ігри. Кращий засіб змусити людину використовувати додаток з великою кількістю функціоналу – внести елемент розваги.

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

Під час розроблення мобільного додатку було дотримано всіх вище

наведених правил та спроектовано оптимальний на вигляд і для застосування інтуїтивно зрозумілий інтерфейс. На рисунку 3.2 наведено вікно додатку

«Авторизація користувача». На рисунку 3.3 зображено сторінку витрат мобільного додатку, де користувач може додавати витрати, які він здійснив.

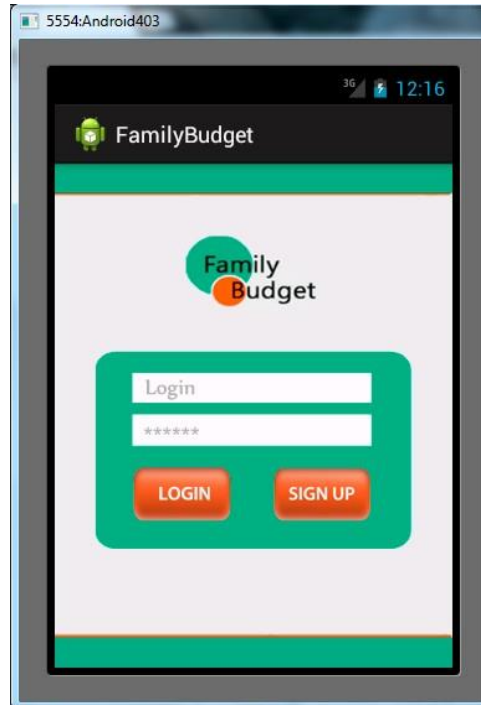


Рис.3.2 – Авторизація користувача



Рис. 3.3 – Додавання витрат

Під час проектування інтерфейсу користувача необхідно визначити структуру діалогу, який буде відбуватися між користувачем і мобільним пристроєм. Розглянуті нижче чотири варіанти структури діалогу є різновидами структури типу «питання – відповідь», проте кожна з них має свої особливості і найбільш зручна для певного класу задач.

Діалог типу «питання – відповідь». Структура діалогу типу «питання – відповідь» (Q&A) заснована на аналогії із звичайним інтерв'ю. Система бере на себе роль інтерв'юера і отримує інформацію від користувача у вигляді відповідей на запитання. Це найбільш відома структура діалогу; всі діалоги, керовані пристроєм, в тій чи іншій мірі складаються з питань, на які користувач відповідає. Однак у структурі Q&A цей процес виражений явно. У кожній точці діалогу система виводить як підказку одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке наступне питання задавати. Структура Q&A не гарантує мінімального обсягу введення, оцінюваного за кількістю натискань клавіш, однак при відповідному підборі скорочень можна зменшити будь-яку надмірність. Разом з тим, структура Q&A володіє одним істотним недоліком. Навіть якщо введення відбувається досить швидко, для людини, яка вже знає, які питання задає система, і які відповіді потрібно на них давати, відповідати на всю серію питань досить утомливо.

Діалоги на основі меню. Меню є найбільш популярним варіантом організації запитів на введення даних під час діалогу, керованого комп'ютером. Існує кілька основних форматів представлення меню на екрані:

- список об'єктів, обраних прямою вказівкою, або вказівкою номера (або мнемонічного коду);
- меню у вигляді блоку даних;
- меню у вигляді рядка даних;
- меню у вигляді піктограм.

Меню у вигляді рядка даних може з'являтися вгорі чи внизу екрану і часто залишається в цій позиції протягом всього діалогу. Таким чином, за допомогою

меню зручно відображати можливі варіанти даних для введення, доступних в будь-який час роботи з системою. Додаткові меню у вигляді блоків даних «спливають» на екрані в позиції, яка визначається поточним положенням покажчика, або «випадають» безпосередньо з рядка меню верхнього рівня. Ці меню зникають після вибору варіанта. Меню у вигляді піктограм являє собою безліч об'єктів вибору, розкиданих по всьому екрану; часто об'єкти вибору містять графічне представлення варіантів роботи. Якщо діалог складається виключно з меню, можна реалізувати послідовний інтерфейс, при якому користувач застосовує тільки пристрої для вказівки; проте така постійність рідко досягається на практиці.

Меню — це найбільш зручна структура діалогу для непідготовлених користувачів; жорстка черговість відкриття і ієрархічна вкладеність меню може викликати роздратування професіонала, уповільнювати його роботу. Традиційна структура меню недостатньо гнучка й не повною мірою узгоджується з методами адаптації діалогу, такими, наприклад, як випереджаюче введення, за допомогою якого можна прискорити темп роботи підготовленого користувача.

Діалог на основі екранних форм. Як структура типу «питання — відповідь», так і структура типу меню припускають обробку єдиної відповіді на кожному кроці діалогу. Діалог на основі екранних форм допускає обробку на одному кроці діалогу декількох відповідей. На практиці форми використовуються в основному там, де облік якоїсь діяльності вимагає введення достатньо стандартного набору даних. Людина, що заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати деяке питання, повертатися назад для корекції попередньої відповіді і навіть «порвати бланк» і почати заповнювати новий. Вона працює з формою до тих пір, поки не заповнить її повністю і не передасть системі. Програмна система може перевіряти кожну відповідь безпосередньо після введення або почекати і вивести список помилок лише після заповнення форми повністю.

Структура діалогу на основі екранної форми забезпечує високий рівень підтримки користувача: для кожного питання форми можуть бути передбачені

повідомлення про помилки та довідкова інформація. Користувачеві можна також надати допомогу, включивши деякі елементи формату відповіді в питання або в поле відповіді. Ця структура дозволяє підвищити швидкість введення даних в порівнянні зі структурою типу «питання – відповідь» і маніпулювати більш широким діапазоном вхідних даних, ніж меню; крім того, з нею можуть працювати користувачі будь-якої кваліфікації.

Діалог на основі командної мови. Структура діалогу на основі командної мови настільки ж поширена, як і структура типу меню. Вона дуже часто використовується в операційних системах і розташовується на іншому кінці спектру структур діалогу стосовно структури типу меню. Історично це перша з реалізованих структур діалогу. При такій організації діалогу програмна система не виводить нічого, крім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисканням клавіші «Enter». Відповідальність за правильність задавання команд лягає на користувача. Система інформує про неможливість виконання невірної команди, як правило, не пояснюючи причин.

Подібно меню, діалог на базі команд зручний для введення керуючих повідомлень, проте він забезпечує більш широкі можливості вибору в будь-якій точці діалогу і не вимагає ієрархічної організації обслуговуючих його програм. Програмна система може підтримувати досить велику кількість команд, але на практиці слід обмежувати їх число, щоб не перевантажувати пам'ять користувача. Структура на базі командної мови не відрізняється гарною підтримкою користувача і придатна в основному для підготовлених фахівців.

Структура на основі мови команд за своїми можливостями найшвидша і найбільш гнучка з усіх структур діалогу. Більшість користувацьких інтерфейсів на базі «природної» мови реалізується за допомогою мов команд з дуже великим набором ключових слів. Підготовлений користувач відчуває задоволення від відчуття того, що він керує системою, а не навпаки. Однак ця структура не забезпечує користувача підтримкою, тому навіть підготовлені користувачі вважають, що дуже складно використовувати всі закладені в ній

можливості. Більшість же користувачів добре знайомі тільки з вельми обмеженим набором засобів, з яким вони працюють регулярно.

Порівняльний аналіз вище розглянутих типів діалогів наведено в таблиці 3.1

* – використання цього типу діалогу даною категорією користувачів вимагає наявності системи допомоги;

** – використання засобів системи можливо тільки в обмеженому обсязі. Інтерфейс розробленої навчальної програми поєднує в собі два типи діалогу: діалог типу меню і «питання – відповідь».

Табл. 3.1

Порівняльний аналіз діалогів

Критерії	Тип діалогу			
	Меню	Питання – відповідь	Мова команд	Заповнення екранних форм
Ціль: запит обчислення складний вибір	+	+	+	+
введення даних введення даних (великий об'єм)		+	+	+
Тип користувача: програміст непрограміст є досвід роботи немає досвіду роботи	+	+	+	+
Час навчання: дуже мало менше, ніж 1 день більше, ніж 1 день	+	+	**	**
	+	+	+	+

Інтерфейс розробленої автоматизованої системи ведення сімейного бюджету поєднує в собі два типи діалогу: діалог типу меню і «питання – відповідь».

3.4 Тестування програмного забезпечення

Тестування програмного забезпечення – це процес технічного дослідження, який виконується на вимогу замовників, і призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись; процес аналізу або експлуатації програмного забезпечення з метою виявлення дефектів.

Тестування передбачає «аналіз» або «експлуатацію» програмного продукту. Тестова діяльність, що пов'язана з аналізом результатів розробки програмного забезпечення, називається статичним тестуванням. Воно передбачає перевірку програмних кодів, контроль та перевірку програми без запуску на комп'ютері. Тестова діяльність, що передбачає експлуатацію програмного продукту, називається динамічним тестуванням. Динамічне та статичне тестування доповнюють одне одного.

Тестування програмного коду - процес виконання програмного коду, спрямований на виявлення існуючих в ньому дефектів. Під дефектом тут розуміється ділянка програмного коду, виконання якого за певних умов призводить до несподіваного поведінки системи (тобто поведінки, не відповідає вимогам). Несподівана поведінка системи може призводити до збоїв у її роботі і відмов, в цьому випадку говорять про суттєві дефекти програмного коду. Деякі дефекти викликають незначні проблеми, що не порушують процес функціонування системи, але трохи ускладнюють роботу з нею. В цьому випадку говорять про середні або малозначні дефекти.

Мета застосування процедури тестування програмного коду - мінімізація кількості дефектів (особливо істотних) в кінцевому продукті. Тестування само по собі не може гарантувати повну відсутність дефектів у програмному коді системи. Однак, в поєднанні з процесами верифікації та валідації, спрямованими на усунення суперечливості і неповноти проектної документації (зокрема - вимог на систему), грамотно організоване тестування дає гарантію того, що система

задовольняє вимогам і веде себе відповідно до них у всіх передбачених ситуаціях.

Програмний продукт є якісним, коли:

- під час роботи користувача з програмним продуктом виникає невелика кількість відмов;
- програмний продукт надійний, а це означає, що його використання рідко викликало аварійні відмови;
- програмний продукт задовольняє вимогам більшості користувачів.

3.5 Аналіз методів тестування

3.5.1 Чорний ящик

Основна ідея в тестуванні системи як чорного ящика полягає в тому, що всі матеріали, які доступні тестувальнику – вимоги на систему, що описують її поведінку, і сама система, працювати з якою він може, тільки подаючи на її входи деякі зовнішні впливи і спостерігаючи на виходах деякий результат. Всі внутрішні особливості реалізації системи приховані від тестувальника, таким чином, система являє собою «чорний ящик», правильність поведінки якого по відношенню до вимог і належить перевірити.

З точки зору програмного коду чорний ящик може представляти з собою набір класів (або модулів) з відомими зовнішніми інтерфейсами, але недоступними вихідними текстами.

Основне завдання тестувальника для даного методу тестування полягає в послідовній перевірці відповідності поведінки системи вимогам. Крім того, тестувальник повинен перевірити роботу системи в критичних ситуаціях: що відбувається в разі подання невірних вхідних значень. В ідеальній ситуації всі варіанти критичних ситуацій повинні бути описані у вимогах на систему, і тестувальнику залишається тільки придумувати конкретні перевірки цих вимог. Проте в реальності в результаті тестування зазвичай виявляється два типи проблем системи. Невідповідність поведінки системи вимогам. Неадекватна

поведінка системи в ситуаціях, не передбачених вимогами.

Звіти по обох типах проблем документуються і передаються розробникам. При цьому проблеми першого типу зазвичай викликають зміну програмного коду, набагато рідше – зміну вимог. Зміна вимог в даному випадку може знадобитися через їх суперечливості (кілька різних вимог описують різні моделі поведінки системи в одній і тій самій ситуації) або некоректності (вимоги не відповідають дійсності).

Проблеми другого типу однозначно вимагають зміни вимог через їх неповноту – у вимогах явно пропущена ситуація, яка веде до неадекватної поведінки системи. При цьому під неадекватною поведінкою може розумітися як повний крах системи, так і взагалі будь-яка поведінка, не описана в вимогах.

Тестування чорного ящика називають також тестуванням за вимогами, тому що це єдине джерело інформації для побудови тест-плану.

3.5.2 Білий ящик

При тестуванні системи як білого ящика тестувальник має доступ не тільки до вимог до системи, її входів і виходів, а й до її внутрішньої структури, тобто бачить її програмний код.

Доступність програмного коду розширює можливості тестувальника тим, що він може бачити відповідність вимог ділянок програмного коду і визначати тим самим, чи на весь програмний код існують вимоги. Програмний код, для якого відсутні вимоги, називають кодом, не покритим вимогами. Такий код є потенційним джерелом неадекватної поведінки системи. Крім того, прозорість системи дозволяє поглибити аналіз її ділянок, що викликають проблеми, часто одна проблема нейтралізує іншу, і вони ніколи не виникають одночасно.

3.5.3 Тестування з використанням емулятора

Тестування мобільних додатків відбувається в середовищі розробки з

використанням емулятора. Після цього додаток тестується на пристрої. Емулятори є простим способом тестувати додаток на мобільному телефоні, не використовуючи його фізично. Нижче представлений список доступних інструментів для тестування додатків серед найпопулярніших мобільних операційних систем:

1) Google Android Emulator. Android Емулятор запускається на Windows як окремий додаток без необхідності повністю завантажувати та встановлювати Android SDK.

2) Офіційний Android SDK Emulator. Включає в себе емулятор мобільного пристрою, який реалізує всі апаратні і програмні особливості типового пристрою.

3) MobiOne. MobiOne Developer це mobile Web IDE для Windows допомагає розробнику програмувати, тестувати, налагоджувати упаковувати і впроваджувати мобільні веб-додатки на пристрої, такі як iPhone, BlackBerry, пристрої на Android і Palm Pre.

4) TestiPhone. Заснований на веб-браузері симулятор для швидкого тестування веб-додатків для iPhone. Працює з використанням Internet Explorer 7, Firefox 2 і Safari 3.

5) iPhoneу. Надає точну середу веб-браузера, розроблена Safari. Може бути використана для розробки веб-сайтів для iPhone. Не є емулятором iPhone. iPhoneу запускається тільки на Mac OS X 10.4.7 і вище.

6) BlackBerry Simulator. Існує безліч офіційних емуляторів BlackBerry. З будь-яким з них можлива перевірка того, як ПО, екран, клавіатура пристрої будуть працювати з додатком.

Для платформи Android найпопулярнішими емуляторами є: Bluestacks, Android SDK Emulator, Andy, Youwave, Oracle VM VirtualBox, GenyMotion. Нижче наведено їх переваги та недоліки.

Bluestacks на даний момент є одним з найвідоміших емуляторів для ПК завдяки досить-таки простому інтерфейсу і непоганому арсеналу можливостей. З плюсів можна назвати наступні:

- можливість роботи в повноекранному режимі;
- установка необмеженої кількості андроїд-додатків в оболонку;
- отримання root-прав, що дозволить працювати з емулятором як з повністю розблокованим гаджетом;
- синхронізація з пристроєм android, що озволить завантажити і встановити ігри та програми з вашого смартфона в програму;
- установка 3d android ігор на комп'ютер;
- можливість управління android-додатками за допомогою клавіатури;
- сумісність з дров'яними контролерами;
- можливість грати в онлайн-ігри.

До явних мінусів емулятора можна віднести:

- некоректна робота в Windows x64;
- може зустрічатися проблема при запуску через конфлікти з антивірусом;
- складність у роботі з онлайн-додатками.

Andy головний конкурент Bluestacks на ринку емуляторів андроїд. І не даремно, оскільки програма має низку переваг:

- згідно порівняльному аналізу, Andy перевершує конкурента завдяки більш стабільній роботі ігор;
- повноцінна підтримка онлайн-додатків на ПК;
- можливість використання андроїд-гаджета, як ігровий контролер;
- швидкодія в роботі навіть на слабких ПК, яка знову ж направлено на ігрову складову програми;
- настройка віртуальної машини Andy: обсяг оперативної пам'яті, необхідний додатком, кількість ядер процесора і т.д.;
- коректна робота з Windows x64.

До мінусів програми можна віднести такі як:

- інтерфейс може здатися досить-таки складним зважаючи на обширний функціонал додатку;
- проблеми з використанням клавіатури ПК в додатках.

Одна з переваг AVD при тестуванні пристроїв – можливість задавати довільні значення для розширення і щільності пікселів екрана. Також Android SDK Emulator має й інші переваги:

- можливість протестувати додаток на різних версіях ОС Android, на пристроях з різними типами дисплея;
- різні налаштування, необхідні для тестування, наприклад, зміна орієнтації екрану;
- емуляція SD-карти.

З недоліків можна виділити такі:

- великий проміжок часу між натисканням кнопки «Run» і запуском програми на емуляторі;
- емулятор працює дуже повільно.

Youwave відмінний емулятор Android для Windows, який не так відомий, як емулятори Andy або Bluestacks, але також має низку переваг:

- малі системні вимоги ПК, що дозволить запустити емулятор навіть на слабкому комп'ютері;
- підтримка навіть старих версій Android, починаючи з 2.3;
- інтуїтивний інтерфейс.

З явних мінусів слід виділити відсутність підтримки української мови.

Oracle VM VirtualBox відрізняється від вищеназваних емуляторів завдяки величезному арсеналу інструментів. Також крім версії для ПК існує додаток для Windows-планшетів. Програма орієнтована більше на розробників додатків. З відмінних якостей можна виділити такі:

- можливість встановити відразу декілька віртуальних систем Android на ПК;
- великий функціонал;
- стабільність при роботі на різних версіях Windows. До мінусів слід віднести:
- велике споживання ресурсів комп'ютера;
- складність в роботі для звичайних користувачів.

GenyMotion – ще один з відомих емуляторів, який чудово підійде розробникам ігор і програм для Android на ПК. Програму варто виділити завдяки таким якостям:

- велика кількість інструментів для роботи;
- хороша продуктивність;
- підтримка великої кількості ігор, також присутня можливість грати в них он-лайн;
- стабільність в роботі на ПК.

Проте ця програма, як і деякі її аналоги, відмовляється працювати в середовищі Windows x64.

Проаналізувавши всі переваги та недоліки емуляторів, вирішено використовувати Android SDK Emulator, так як він найкраще підходить для розробки додатків на ОС Windows x64.

3.6 Тестування автоматизованої системи

Для тестування автоматизованої системи ведення сімейного бюджету застосовано функціональний вид тестування на Android SDK Emulator. При даному підході текст програми не доступний, і програма розглядається як «чорний ящик».

Метою тестування автоматизованої системи є визначення того, на скільки коректно вона виконує свої функції. У даному випадку додаток повинен надавати інформацію про бюджет зареєстрованого користувача. Якщо ж користувач не зареєстрований – надати йому можливість зареєструватися.

Після запуску мобільного додатку з'являється вікно авторизації, в якому користувачу необхідно ввести свій логін і пароль. Якщо ж користувач не зареєстрований, йому необхідно перейти на вікно реєстрації (рис. 3.4). Якщо реєстрація пройшла успішно, то на екрані повинно з'явитися повідомлення про це.



Рис.3.4 – Вікно реєстрації користувача

Як видно на рисунку 3.5, при реєстрації з'явилося модальне вікно з повідомленням про успішну реєстрацію.

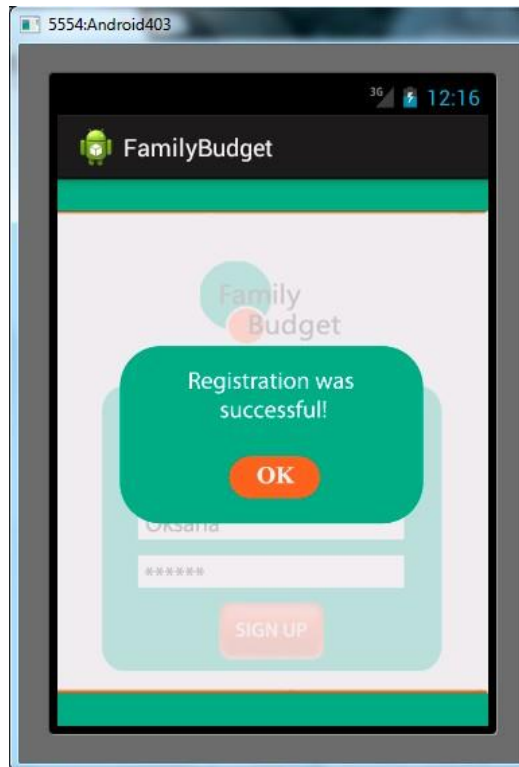


Рис.3.5 – Вікно повідомлення про успішну реєстрацію

Додаток видає правильний результат, отже, тестування функції реєстрації користувача пройшло успішно.

Наступною протестована функція додавання ліміту на витрати. Після заповнення необхідних полів і натиснення кнопки «Save», створено ліміт на бюджет на місяць у розмірі 500\$. Результат тестування наведено на рисунку 3.6.



Рис.3.6 – Встановлення ліміту на витрати

Отже, протестувавши основні функції мобільного додатку, збоїв і помилок у роботі не виявлено, програма працює коректно.

Тестування підтвердило ефективність та правильність функціонування розробленої автоматизованої системи ведення сімейного бюджету. Програмний продукт є легким у використанні, має привабливий та інтуїтивно зрозумілий інтерфейс. Додаток дозволяє заощаджувати час користувача за рахунок мінімального введення даних. Дає можливість організувати щоденні та щомісячні витрати, запланувати великі та малі покупки, внести їх до календаря, а також заощаджувати кошти для здійснення даних покупок.

Використовуючи програмний продукт, користувач не тільки ніколи не забуде про важливі дати, події на зразок днів народжень, річниць і т.п., а й зможе виділи необхідну кількість грошей на них.

4.Охорона праці та довкілля

4.1 Охорона праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності;

– діюча на підставі відповідних законодавчих та інших нормативних актів система соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, що забезпечують збереження здоров'я і працездатності людини в процесі праці.

– дозвіл на початок робіт підвищеної небезпеки, який необхідний організації чи підприємству, хто працює в будівництві.

4.2 Правові та організаційні питання охорони праці

За Законом України «Про охорону праці» в останній редакції 2004 року, охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних, лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі трудової діяльності. Дія цього Закону поширюється на всіх юридичних та фізичних осіб, які відповідно до законодавства використовують найману працю, та на всіх працюючих.

Для управління охороною праці створюються відповідні служби і призначаються компетентними органами посадові особи, які забезпечують вирішення конкретних питань охорони праці. На підприємстві з кількістю працюючих 50 і більше осіб роботодавець створює службу охорони праці відповідно до типового положення, що затверджується спеціально уповноваженим центральним органом виконавчої влади з питань нагляду за

охороною праці. На підприємстві з кількістю працюючих менше 50 осіб функції служби охорони праці можуть виконувати в порядку сумісництва особи, які мають відповідну підготовку. За порушення законодавства про охорону праці, невиконання розпоряджень посадових осіб органів державного нагляду за охороною праці юридичні та фізичні особи, які відповідно до законодавства використовують найману працю, притягаються органами державного нагляду за охороною праці до сплати штрафу у порядку, встановленому законом.

4.3 Правила охорони праці під час експлуатації електронно-обчислювальних машин

Правила охорони праці під час експлуатації електронно-обчислювальних машин поширюються на всі підприємства, установи, організації, юридичні особи, незалежно від форми власності, відомчої належності, видів діяльності та на фізичних осіб, які здійснюють розробку, виробництво та застосування електронно-обчислювальних машин і персональних комп'ютерів (далі – ЕОМ), у тому числі й на тих, які мають робочі місця, обладнані ЕОМ, або виконують обслуговування, ремонт та налагодження ЕОМ.

Загальні вимоги до виробничих приміщень.

Облаштування робочих місць, обладнаних відеотерміналами, повинно забезпечувати:

- належні умови освітлення приміщення і робочого місця, відсутність відблисків;
- оптимальні параметри мікроклімату;
- наявність шуму та вібрації;
- м'яке рентгенівське випромінювання;
- електромагнітне випромінювання;
- ультрафіолетове та інфрачервоне випромінювання;
- електростатичне поле між екраном і оператором;

Для всіх споруд і приміщень, в яких експлуатуються відеотермінали та ЕОМ,

повинна бути визначена категорія з вибухопожежної і пожежної безпеки відповідно до ОНТП 24-86 «Определение категорий помещений и зданий по взрывопожарной и пожарной опасности».

Виробничі приміщення, в яких розташовані ЕОМ, не повинні межувати з приміщеннями, де рівні шуму та вібрації перевищують норму (механічні цехи, майстерні тощо). Робочі місця з відеотерміналами або персональними ЕОМ у приміщеннях з джерелами шкідливих виробничих факторів повинні розміщуватися в ізольованих кабінах з обладнанням повітрообміном. Площу приміщень, в яких розташовують відеотермінали, визначають згідно з чинними нормативними документами з розрахунку на одне робоче місце, обладнане відеотерміналом: площа – не менше 6,0 м², обсяг – не менше 20,0 м³, з урахуванням максимальної кількості осіб, які одночасно працюють у зміні.

Стіни, стеля, підлога приміщень, де розміщені ЕОМ, повинні виготовлятися з матеріалів, дозволених для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду.

Обслуговування, ремонт та налагодження ЕОМ, вузлів та блоків ЕОМ слід виконувати в окремому приміщенні (майстерні), які можуть передбачити можливість вологого очищення поверхонь комунікацій та опалювальних приладів.

Підлога всієї зони обслуговування, ремонту та налагодження ЕОМ, вузлів та блоків ЕОМ має бути вкрита діелектричними килимками, термін використання яких після їх випробування на електричну міцність не закінчився, або викладена ізолювальними підстилками (шириною не менше ніж 0,75-0,8 м) для ніг.

Приміщення комп'ютерних класів (залів), в яких проводиться навчання на ЕОМ повинні мати суміжне приміщення (лаборантську) площею не менше 18 м² з двома входами: в учбове приміщення та в коридор (на сходову клітку).

Заземлені конструкції, що знаходяться в приміщеннях (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), мають бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

У приміщеннях з ЕОМ слід щоденно проводити вологе прибирання.

Приміщення, в яких розміщуються ЕОМ типу ЕС, СМ та інші великі ЕОМ загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежегасіння відповідно до вимог Переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежегасіння та пожежної сигналізації, СНиП 2.04.09-84, СН 512-78, Правил пожежної безпеки в Україні та вимог нормативно-технічної та експлуатаційної документації заводу-виробника. Підходи до засобів пожежегасіння повинні бути вільними.

Санітарно-гігієнічні вимоги.

Умови праці осіб, які працюють з ЕОМ, повинні відповідати I або II класу згідно з Гігієнічною класифікацією праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу № 4137-86, затвердженою МОЗ України 12.08.86.

Вимоги до освітлення.

Приміщення з ЕОМ повинні мати природне і штучне освітлення відповідно до СНиП СНиП II-4-79 «Природне та штучне освітлення». Природне світло повинно проникати через бічні світло прорізи, зорієнтовані, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5%. Розрахунки КПО проводяться відповідно до СНиП11-4-79. Вікна приміщень з відео терміналами повинні мати регульовальні пристрої для відкривання, а також жалюзі, штори, зовнішні козирки тощо.

Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути обладнане системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, де переважають роботи з документами, допускається вживати систему комбінованого освітлення (додатково до загального освітлення встановлюються світильники місцевого освітлення). Відношення яскравості екрану комп'ютера до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1.

При розташуванні відео терміналів ЕОМ за периметром приміщення лінії світильників штучного освітлення повинні розміщуватися локально над робочими місцями.

Для забезпечення нормованих значень освітлення в приміщеннях з відеотерміналами ЕОМ загального та персонального користування необхідно очищати віконне скло та світильники не рідше ніж 2 рази на рік, та своєчасно, проводити заміну ламп, що перегоріли (табл. 4.1).

Табл. 4.1

Норми освітленості в кабінетах і класах з ПК

Характеристика роботи	Робоча поверхня	Площина	Освітленість, лк	Примітка
Робота переважно з екранами дисплеїв ПК (50 % та більше робочого часу)	Екран	В	200	не вище
	Клавіатура	Г	400	не нижче
	Стіл	Г	400	не нижче
Робота переважно з документами (з екранами дисплеїв ПК менше 50 % робочого часу)	Екран	В	200	не вище
	Клавіатура	Г	400	не нижче
	Стіл	Г	500	не нижче
	Дошка	В	500	не нижче
Проходи основні	Підлога	Г	100	

Примітка: В - вертикальна площина, Г - горизонтальна площина.

Загальне освітлення має бути виконане у вигляді суцільних або переривчатих ліній світильників, що розміщуються збоку від робочих місць (переважно зліва) паралельно лінії зору працівників. Допускається застосовувати світильники таких класів світлорозподілу:

- світильники прямого світла — П;
- переважно прямого світла — Н;
- переважно відбитого світла — В.

Вимоги до рівнів шуму та вібрації.

У приміщеннях з ЕОМ рівні звукового тиску, рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам ГОСТ 12.1.003 ССБТ «Шум. Общие требования безопасности», СН 3223-85 «Санітарні норми

допустимих рівнів шуму на робочих місцях», затверджених Міністерством охорони здоров'я СРСР, ГР N 2411-81 «Гігієнічні рекомендації по встановленню рівнів шуму на робочих місцях з урахуванням напруженості та тяжкості праці», затверджених Міністерством охорони здоров'я України.

Вимоги до вентиляції, опалення та кондиціювання, мікроклімату.

Приміщення з ЕОМ повинні бути обладнані системами опалення, кондиціювання повітря або припливно-витяжною вентиляцією відповідно до СНиП 2.04.05–91 «Опалення, вентиляція, кондиціювання». Параметри мікроклімату, іонного складу повітря, вміст шкідливих речовин на робочих місцях, оснащених відеотерміналами, повинні відповідати вимогам пункту 2.4 СН 4088-86 «Санітарні норми мікроклімату виробничих приміщень», затверджених Міністерством охорони здоров'я СРСР, ГОСТ 12.1.005-88 «ССБТ Общие санитарно-гигиенические требования к воздуху рабочей зоны» (табл. 4.2), СН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» (табл. 4.3).

Табл. 4.2

Нормовані параметри мікроклімату для приміщень з ВДТ та ПЕОМ

Пора року	Категорія робіт згідно ГОСТ 12.1-005-88	Температура повітря, град.С,	Відносна вологість повітря, %	Шкідливість руху повітря, м/с
		оптимальна	оптимальна	оптимальна
Холодна	легка -1 а	22 – 24	40 – 60	0,1
	легка -1 б	21 – 23	40 – 60	0,1
Тепла	легка -1 а	23 – 25	40 – 60	0,1
	легка -1 б	22 – 24	40 – 60	0,2

Табл. 4.3

Рівні іонізації повітря приміщень при роботі на ВДТ та ПЕОМ (відповідно до СН 2152-80)

Рівні	Кількість іонів в 1 см куб. повітря	
	+ п	- п
Мінімально необхідне	400	600
Оптимальні	1500 – 3000	3000 – 5000

Максимально допустимі	50000	50000
-----------------------	-------	-------

Вимоги щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних та магнітних полів.

Рівні ультрафіолетового випромінювання не повинні перевищувати допустимих відповідно до СН № 4557-88 «Санітарні норми ультрафіолетового випромінювання у виробничих приміщеннях».

Гранично допустима напруженість електростатичного поля на робочих місцях не повинна перевищувати рівнів, наведених в ГОСТ 12.1.045 «ССБТ. Електромагнітні поля. Допустимі рівні на робочих місцях і вимоги до проведення контролю», СН № 1757–77 «Санітарно-гігієнічні норми допустимої напруги електростатичного поля» та ДСанПІН 3.3.2–007–98.

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05 м від екрана та корпусу відео термінала при будь-яких положеннях регулювальних пристроїв відповідно до Норм радіаційної безпеки України (НРБУ-97), затверджених постановою державного санітарного лікаря Міністерства охорони здоров'я України від 18.08.97 № 58, не повинна перевищувати $7,74 \times 10^{-12}$ А/кГ, що відповідає еквівалентній дозі 0,1 мбер/год. (100 мкР/год.).

Вимоги електробезпеки.

Під час проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ЕОМ необхідно дотримуватись вимог ПВЕ, ПТЕ, ПБЕ, СН 357–77 «Инструкция по проектированию силового осветительного оборудования промышленных предприятий», затверджених Держбудом, ГОСТ 12.1.006, ГОСТ 12.1.030 «ССБТ. Электробезопасность. Защитное заземление, зануление», ГОСТ 12.1.019 «ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты», ГОСТ 12.1.045, ВСН 59-88 Держкомархитектуры «Электрооборудование жилых и общественных зданий. Нормы проектирования», Правил пожежної безпеки в Україні, цих Правил, а також розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно–технічної і експлуатаційної документації заводу-виробника ЕОМ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережу штепсельних розеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

Вимоги до обладнання.

Відеотермінали, ЕОМ, ПЕОМ, спеціальні периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні відповідати вимогам чинних в Україні стандартів, нормативних актів з охорони праці та цих Правил. Відеотермінали, ЕОМ, ПЕОМ, спеціальні периферійні пристрої ЕОМ закордонного виробництва додатково повинні відповідати вимогам національних стандартів держав–виробників і мати відповідну позначку на корпусі, в паспорті або іншій експлуатаційній документації.

Вимоги до організації робочого місця користувача ЕОМ.

Організація робочого місця користувача відеотерміналу та ЕОМ повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ГОСТ 12.2.032 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования»; характеру та особливостям трудової діяльності.

Робочі місця з відеотерміналами відносно світлових прорізів повинні

розміщуватися так, щоб природне світло падало збоку, переважно зліва.

При розміщенні робочих місць з відеотерміналами та персональними ЕОМ необхідно дотримуватись таких вимог:

- робочі місця з відеотерміналами та персональними ЕОМ розміщуються на відстані не менше 1 м від стін зі світловими прорізами;
- відстань між бічними поверхнями відеотерміналів має бути не меншою за 1,2 м;
- відстань між тильною поверхнею одного відео термінала та екраном іншого не повинна бути меншою 2,5 м;
- прохід між рядами робочих місць має бути не меншим 1 м.

Конструкція робочого місця користувача відеотермінала (при роботі сидячи) має забезпечувати підтримання оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг – на підлозі або на підставці для ніг; стегна – в горизонтальній площині; передпліччя – вертикально; лікті – під кутом 70–90° до вертикальної площини; зап'ястя зігнуті під кутом не більше 20° відносно горизонтальної площини, нахил голови –15–20° відносно вертикальної площини.

Висота робочої поверхні столу для відеотермінала має бути в межах 680–800 мм, а ширина – забезпечувати можливість виконання операцій в зоні досяжності моторного поля. Рекомендовані розміри столу: висота –725 мм, ширина –600–1400 мм, глибина –800–1000 мм.

Для зниження статичного напруження м'язів рук необхідно застосовувати стаціонарні або знімні підлокітники довжиною не менше 250 мм, шириною – 50 – 70 мм, що регулюються по висоті над сидінням у межах 230 ± 30 мм та по відстані між підлокітниками в межах 350–500 мм.

Екран відеотермінала та клавіатура мають розташовуватися на оптимальній відстані від очей користувача, але не ближче 600 мм, з урахуванням розміру алфавітно–цифрових знаків та символів. Відстань від екрана до ока працівника повинна складати:

- при розмірі екрану по діагоналі 35/38 см (14" /15") – 600 – 700 мм;
- при розмірі екрану по діагоналі 43 см (17") – 700 – 800 мм;

- при розмірі екрану по діагоналі 48 см (19") – 800 – 900 мм;
- при розмірі екрану по діагоналі 53 см (21") – 900 – 1000 мм.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, регульованій за висотою, робочій поверхні окремо від столу на відстані 100–300 мм від краю, ближчого до працівника. Кут нахилу клавіатури має бути в межах 15'.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрану відеотермінала, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті 900 – 1300 мм, по глибині 400 – 500 мм.

При потребі високої концентрації уваги під час виконання робіт з високим рівнем напруженості суміжні робочі місця з відеотерміналами та персональними ЕОМ необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м.

Вимоги безпеки під час експлуатації ЕОМ.

Користувачі ЕОМ повинні слідкувати за тим, щоб відеотермінали, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ були справними і випробуваними відповідно до чинних нормативних документів. Щоденно перед початком роботи необхідно проводити очищення екрану відеотермінала від пилу та інших забруднень. Після закінчення роботи відеотермінал та персональна ЕОМ повинні бути відключені від електричної мережі. У разі виникнення аварійної ситуації необхідно негайно відключити відеотермінал та ЕОМ від електричної мережі. При використанні з ЕОМ та відеотерміналами лазерних принтерів потрібно дотримуватись вимог Санітарних норм та правил устрою та експлуатації лазерів № 5804–91, затверджених Міністерством охорони здоров'я в 1991 р.

Розрахунок штучного освітлення.

Рациональне освітлення виробничих ділянок є одним з найважливіших факторів попередження травматизму і професійних захворювань. Правильно організоване висвітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітленість на робочому місці повинна

бути такою, щоб працюючий міг без напруги зору виконувати свою роботу при припустимому з народногосподарської точки зору витратою засобів, матеріалів і електроенергії.

Приведемо розрахунок штучного освітлення для приміщення, розміри якого: довжина 5м, ширина 3м, висота 3м.

Розрахунок освітленості виконаємо методом коефіцієнта використання. Цей метод використовується для розрахунку загального рівномірного висвітлення горизонтальних поверхонь виробничих приміщень при відсутності затемнень.

4.4 Пожежна безпека

Пожежна безпека – це стан об’єкту, при якому виключається можливість пожежі, а у випадку її виникнення виключається дія на людей небезпечних факторів пожежі та забезпечується захист матеріальних цінностей.

Відповідно до Закону України «Про пожежну безпеку» пожежна безпека повинна забезпечуватися шляхом проведення організаційних, технічних та інших заходів, спрямованих на попередження пожеж, забезпечення безпеки людей, зниження можливих майнових втрат і зменшення негативних екологічних наслідків у разі їх виникнення, створення умов для швидкого виклику пожежних підрозділів та успішного гасіння пожеж.

Система активного пожежного захисту – це комплекс організаційних заходів і технічних засобів по боротьбі з пожежами і запобіганню дії на людей небезпечних чинників пожежі, а також обмеження матеріальних збитків від неї.

Основний напрямок діяльності пожежної охорони полягає у профілактиці пожеж і в обмеженні їх розмірів. Пожежі або вибухи на підприємствах завдають великих матеріальних збитків і майже завжди супроводжуються нещасними випадками з людьми (опіки, поранення) тощо.

Контроль за дотриманням затверджених відповідно до діючого законодавства норм і правил пожежної безпеки на об’єктах народного

господарства здійснюють органи Державного пожежного нагляду (ДНП) управління пожежної охорони Міністерства надзвичайних ситуацій України.

Відповідальність за прийняття протипожежних заходів в організаціях покладається персонально на їх керівників без права передовіряти цю відповідальність іншим, підлеглим їм особам, вони здійснюють загальне керівництво роботою в галузі пожежної безпеки підприємств і організацій.

На (підприємстві) встановлюються первинні засоби пожежегасіння відповідно до вимог Правил пожежної безпеки України. У приміщеннях з комп'ютерами рекомендується використовувати системи автоматичної пожежної сигналізації для виявлення пожежі та оповіщення працівників. Для приміщення, по якому проводиться розрахунок, потрібно влаштування місць утримання засобів пожежегасіння вогнегасників порошкових типу ОПУ–5 з вільним доступом у випадку необхідності їх використання.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

В результаті проведеної роботи було розроблено автоматизовану систему ведення сімейного бюджету як мобільного додатку. Сьогодні, коли людина постійно прагне бути на зв'язку, розробка мобільних додатків є дуже актуальною.

Для розробки мобільного додатку проаналізовано найбільш популярні існуючі аналоги, визначено їх переваги та недоліки. Також зроблено огляд особливостей класифікації мобільних додатків і визначено, що розроблений додаток належить до сервісних нативних додатків. Проведено варіантний аналіз засобів реалізації автоматизованої системи і обґрунтовано вибір середовища програмування.

Проаналізовано принципи розробки мобільних додатків і обрано платформу Android для розробки автоматизованої системи, а також здійснено аналіз особливостей розробки бази даних.

Для реалізації мобільного додатку розроблено модель роботи автоматизованої системи, структуру бази даних та ієрархічну модель мобільного додатку. Також наведено блок-схему алгоритму роботи блоку реєстрації та авторизації користувача, який належить до розгалуженого типу. Розроблено базу даних, логіку програмного продукту та розмітку вікон Android-додатку.

Розроблено інтуїтивно-зрозумілий інтерфейс користувача та визначено, що він належить до діалогу типу меню. Проведене тестування підтвердило ефективність і правильність функціонування розробленої автоматизованої системи ведення сімейного бюджету.

Програма значно полегшує контроль за фінансовим станом користувача, так як за потребою надає доступ до інформації про поповнення бюджету, щоденні та щомісячні витрати, заплановані покупки. Розроблений додаток сприятиме ефективному веденню бухгалтерського обліку сімейного бюджету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технології розробки мобільних додатків. [Електронний ресурс]. – Режим доступу: <http://lektsii.net/2-50017.html>
2. Розробка мобільних додатків. [Електронний ресурс]. – Режим доступу: <http://kiev.itstep.org/razrabotka-mobilnyh-prilozhenij-pod-android/>
3. Додатки для планування бюджету. [Електронний ресурс]. – Режим доступу: <http://terytoriya.com.ua/index.php/samovdoskonalennya/550-4-zruchni-dodatki-dlyaplanuvannya-vashogo-byudzhetu>
4. Мобільна бухгалтерія. [Електронний ресурс]. – Режим доступу: <http://itc.ua/articles/mobilnaya-buhgalteriya-android-prilozheniya-dlyaucheta-lichnyih-finansov/>
5. Білоконна К. В. Ринок мобільних гаджетів // К. В. Білоконна, О. Н. Романюк, О. О. Сівець. – Збірник матеріалів Всеукраїнської науковопрактичної Інтернет-конференції 2013 року «Електронні інформаційні ресурси в освіті і науці: створення, використання, доступ» – Вінниця 2013. – С. 14-16.
6. Bilokonna K.V. Market of video cards // K. V. Bilokonna, S. A. Romanyuk, O. A. Sivets. – Materiały IX międzynarodowej naukow-praktycznej konferencji «Dynamika naukowych badań – 2013» 07-15 lipca 2013 roku Volume 12. – Przemysł Nauka i studia 2013. – С. 53-56.
7. Бевз С. В. Модель обґрунтованого вибору методів тестового контролю знань // С. В. Бевз, В. В. Войтко, О. О. Сівець. – Вісник Хмельницького національного університету. Технічні науки. – Хмельницький 2013. – № 6. – С. 218-223.
8. Сівець О. О. Аналіз методів тестового контролю знань. // О. О. Сівець. – XLII регіональна науково-технічна конференція професорсько-викладацького складу, співробітників та студентів університету з участю працівників науководослідних організацій та інженерно-технічних працівників підприємств м. Вінниці та області. – Вінниця, 2013.

9. Сівець О. О. Обґрунтований вибір методів тестового контролю знань. // О. О. Сівець. – XLIII регіональна науково-технічна конференція професорсько-викладацького складу, співробітників та студентів університету з участю 87 працівників науково-дослідних організацій та інженерно-технічних працівників підприємств м. Вінниці та області. – Вінниця, 2014.

10. Сівець О. О. Розробка автоматизованої системи ведення сімейного бюджету як мобільного додатку. // О. О. Сівець. – Міжвузівський студентський вебінар «Інноваційні та інформаційні технології в бізнесі та освіті». – Вінниця, 2015.

11. Сівець О.О., Піддубецька М.П., Войтко В.В. Комп'ютерна програма ведення обліку фінансів сім'ї "Family Budget" // Свідоцтво про реєстрацію авторського права на твір №57994, від 05.01.2015. – 2015. – Свідоцтво про державну реєстрацію авторського права на твір.

12. Планування та облік сімейного бюджету. [Електронний ресурс]. – Режим доступу: <http://www.maryse.ru/page/planuvannja-ta-oblik-simejnogo-bjudzhetu>

13. Toshl Finance. [Електронний ресурс]. – Режим доступу: <http://itc.ua/articles/mobilnaya-buhgalteriya-android-prilozheniya-dlya-uchetalichnyih-finansov/>

14. MoneyLover. [Електронний ресурс]. – Режим доступу: <https://moneylover.me/>

15. Monefy. [Електронний ресурс]. – Режим доступу: <http://www.monefy.me/>

16. FinancePM. [Електронний ресурс]. – Режим доступу: <http://finance.uramaks.com/login.html>

17. Класифікація мобільних додатків [Електронний ресурс]. – Режим доступу: <http://voroninstudio.eu/uk/service/razrabotka-mobilnih-prilozheniy>

18. Мови програмування. [Електронний ресурс]. – Режим доступу: http://zei.narod.ru/Comparison_C_Java_Cpp_3.pdf

19. Eclipse. [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Eclipse>

20. Android Studio. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
21. Методологія розробки додатків. [Електронний ресурс]. – Режим доступу: <http://android.mobile-review.com/articles/22580/> 88
22. SQLite. [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/SQLite>
23. Архітектура Android-додатків. [Електронний ресурс]. – Режим доступу: <http://habrahabr.ru/post/140655/>
24. Алгоритми. [Електронний ресурс]. – Режим доступу: <http://ru.wikipedia.org/wiki/Алгоритм>
25. Клас SQLiteOpenHelper. [Електронний ресурс]. – Режим доступу: <http://androidfanclub.ru/programming/створення-бази-даних-клас-sqliteopenhelper>
26. Activity. [Електронний ресурс]. – Режим доступу: <http://metanit.com/java/android/2.1.php>
27. Layout. [Електронний ресурс]. – Режим доступу: <http://developer.alexanderklimov.ru/android/theory/layout.php>
28. 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
29. Тестування. [Електронний ресурс]. – Режим доступу: <http://arhivstatey.pp.ua/index.php?newsid=26947>
30. Емулятори Android. [Електронний ресурс]. – Режим доступу: <http://appstips.ru/articles/emulatory-android.html>
31. Грабовецький Б. Є. Економіка підприємства. / Б. Є. Грабовецький, Т.М. Пілявоз – Вінниця: ВНТУ, 2009 – 248 с.