

ЗМІСТ

1	ПОСТАНОВКА ЗАДАЧІ.....	6
1.1	Функціональні вимоги до програмного продукту	6
1.2	Нефункціональні вимоги до програмного продукту.....	7
1.3	Огляд та аналіз аналогів розроблюваного продукту	8
2	ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ВИКОРИСТОВУВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ	12
2.1	Цільова платформа.....	12
2.2	Мова програмування.....	14
2.3	Система управління базами даних	15
3.1	Методологія IDEF0	18
3.2	IDEF0-діаграма для продукту, що розробляється	19
3.2.1	Запит до системи	21
3.2.2	Обробка дій користувача.....	23
3.2.3	Зміна бази даних.....	23
4	ОПИС АЛГОРИТМІВ.....	25
4.1	Основний цикл програми	25
4.2	Створення рахунків.....	26
4.3	Створення категорій	27
4.4	Створення рахунку.....	28

Змін.	Арк.	№ Докум.	Підпис	Дата				
Розроб.					<div> <div>Літ.</div> <div>Аркуш</div> <div>Аркушів</div> </div> <div> <div>1</div> <div>51</div> </div> <div> <div>НТУУ «КПІ»</div> <div>ФІОТ гр. ІТ-51</div> </div>			
Переірв.								
Н.контр.								
Затв.								

5	СТРУКТУРА ДАНИХ І РЕСУРСІВ ПРОГРАМИ.....	30
5.1	Клас MainActivity	30
5.2	Клас UsingDataBaseListFragment	31
5.3	Клас FragmentTransactions.....	32
5.4	Клас UsingDataBaseActivity.....	33
5.5	Клас SingleEntityActivity	34
5.6	Клас AccountsActivity	34
5.7	Клас SingleAccountActivity.....	35
5.8	Клас CategoriesActivity	36
5.9	Клас FragmentCategories	36
5.10	Клас SingleCategoryActivity.....	37
5.11	Клас SingleTransactionActivity	38
5.12	Клас SQLiteHandler	39
5.13	Опис бази даних	40
6	КЕРІВНИЦТВО КОРИСТУВАЧА	42
6.1	Початок роботи з додатком.....	42
6.2	Вікно «Журнал»	42
6.3	Вікно «Платіж».....	45
6.4	Вікна «Рахунки» і «Рахунок».....	46
6.5	Вікно «Категорії»	48
6.6	Вікно «Категорія»	49
6.7	Вікно «Кругова діаграма» і «Стовпчаста діаграма».....	50
	ВИСНОВОК.....	52

						Арк.
						2
Змн.	Арк.	№ Докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53

ДОДАТОК А. Вихідний код програми 55

						Арк.
						3
Змн.	Арк.	№ Докум.	Підпис	Дата		

ВСТУП

Проблема управління особистим бюджетом є актуальною для кожної людини. Постійно здійснюються різні покупки, гроші беруться у кредит або отримуються з банківських відсотків. І бюджет стрімко падає, якщо його неефективно витратити. Людина, яка грамотно розпоряджається своїм бюджетом знає, скільки грошей є, розраховує, скільки грошей потрібно витратити, і приймає рішення, де можна заощадити і від чого можна взагалі відмовитися. Вищеописане можна значно спростити, розробивши автоматизовану систему управління особистим бюджетом. Саме для цієї мети і був придуманий проект «Домашня бухгалтерія», про який далі піде мова.

Мета «Домашньої бухгалтерії» – автоматизувати повсякденні фінансові розрахунки.

Кожен день проходить якнайменше декілька транзакцій (витрати та доход). І для найбільш ефективного розподілення бюджету необхідно контролювати будь-які транзакції, щоб потім проаналізувати їх і зробити певні висновки. Зазвичай, транзакції проходять через джерела, наприклад готівка або банківська карта. І дуже важко контролювати бюджет на кожному з джерел, а тим паче й причини витрат/надходжень на кожне з них.

Розроблювана система дозволить нам вести щоденник витрат і надходжень, розраховувати, як вони вплинуть на загальний бюджет, а також аналізувати, коли і через що було витрачено / отримано найбільшу кількість грошей. Також, буде можливість групувати витрати / надходження, тобто транзакції, за датою їх здійснення, за їх причиною. Це дозволить ефективно аналізувати сезонність транзакції, зробити висновки щодо того, на що витратилось більше, ніж планувалось та скорегувати майбутнє користування власним бюджетом. Система дозволить групувати транзакції за різними джерелами для більш точного аналізу звідки витрачається, або надходить найбільше грошей. Основним методом аналізу буде перегляд усієї статистики за

						Арк.
						4
Змн.	Арк.	№ Докум.	Підпис	Дата		

допомогою зручних діаграм та графіків, перегляд транзакцій з різними фільтрами та різною датою створення.

Система повинна бути ефективною, в плані обробки даних, виконання програмних функцій; повинна бути стійкою, тобто виконувати всі функції, які будуть розроблені, незалежно від зовнішніх факторів або ж реагувати на зміни шляхом виклику системної помилки (можливо, показувати на якому місці стався збій). Повинні бути створені діалогові вікна для інтерактивного режиму роботи з користувачем. Виконання вимог ергономіки інтерфейсу, створення комфортних умов роботи.

Ціль даної роботи є проектування та розробка мобільного додатка для ведення домашньої бухгалтерії. Користувачеві буде надаватися можливість внесення витрат / доходів, ведення обліку їх категорій, створення власних рахунків і робота з ними. Також, можливий перегляд історії для конкретного проміжку часу і перегляд статистики у вигляді кругової і стовпчатої діаграм.

У пояснювальній записці будуть описані основні етапи проектування та розробки програмного забезпечення, основні алгоритми його роботи, деталі реалізації, та, у кінцевому варіанті – керівництво користувача.

						Арк.
						5
Змн.	Арк.	№ Докум.	Підпис	Дата		

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Функціональні вимоги до програмного продукту

Функціональні вимоги визначають основний фронт робіт розробника, регламентують поведінку додатки в тій чи іншій ситуації. Вони ставлять завдання, які повинна виконувати система.

Необхідно створити програмне забезпечення для ведення домашньої бухгалтерії, що надає достатній функціонал для аналізу витрат / доходів і отримання наочної статистики.

З точки зору функціональних характеристик виділяються нижче перераховані вимоги.

Робота з платежами (транзакціями):

- вибір дати і часу;
- введення суми платежу;
- вибір категорії, для якої здійснюється платіж;
- вибір рахунку, для якого здійснюється платіж;
- створення нотатки;
- збереження нового платежу;
- редагування і видалення існуючих платежів;
- внесення платежу за допомогою аналізу вхідних смс на основі шаблонів.

Робота с категоріями:

- ведення назви категорії;
- вибір типу категорії: витрата, дохід;
- вибір логотипу категорії;
- збереження нової категорії;
- редагування і видалення існуючих категорій.

Робота із рахунками:

						Арк.
						6
Змн.	Арк.	№ Докум.	Підпис	Дата		

- введення назви рахунку;
- введення початкового балансу рахунку;
- збереження нового рахунку;
- редагування і видалення існуючого рахунку.

Перегляд статистики у вигляді діаграм:

- кругова діаграма;
- стовпчаста діаграма.

Основний діалог програми:

- перегляд журналу платежів окремо для витрат, поповнень;
- перегляд журналу для різних часових діапазонів: день, тиждень, місяць, рік;
- перегляд діаграм для різних часових діапазонів;
- навігаційне меню.

1.2 Нефункціональні вимоги до програмного продукту

Нефункціональні вимоги, визначають вимоги до системи в цілому, а не в окремих випадках використання, описують властивості і характеристики, які повинна демонструвати система, а також обмеження, які повинні бути дотримані в окремих випадках.

Виділимо нефункціональні вимоги для наступних груп: Зовнішні інтерфейси, атрибути якості, обмеження.

Зовнішні інтерфейси:

- інтерфейс програми повинен мати привабливий дизайн;
- інтерфейс повинен бути інтуїтивно зрозумілим;
- інтерфейс повинен бути естетичним;
- інтерфейс програми повинен відповідати основним правилам юзабіліті;
- дизайн додатків повинен бути в стилі Material design.

						Арк.
						7
Змн.	Арк.	№ Докум.	Підпис	Дата		

Атрибути якості:

- надійність – додаток повинен вірно реагувати на будь-які данні, введені користувачем, що не зависати під час роботи. Додаток має бути захищеним від помилок і гарантувати цілісність даних;
- продуктивність – додаток повинен виконувати всі передбачені операції без довгих очікувань і зависань. Виконання кожної операції має проходити в найкоротший термін;
- експлуатаційна придатність – додаток повинен виконувати всі необхідні вимоги і бути працездатним незалежно від часу його використання.

Обмеження:

- додаток виповнюється на мобільних пристроях під управлінням операційної системи Android;
- мінімальним API є 16.

1.3 Огляд та аналіз аналогів розроблюваного продукту

Розглянемо деякі раніше створені програмні продукти.

HomeBank – загальнодоступна програма для ведення домашньої бухгалтерії. Надається можливість контролювати свої доходи і витрати, аналізувати бюджет. Програма дозволяє впорядковувати платежі за категоріями, планувати транзакції для автоматичного вносити в базу даних (наприклад, щоденні рахунки). У програмі також є функція одночасного редагування кількох полів.

За допомогою HomeBank можна аналізувати фінансовий стан і складати звіти. Програма надає можливість генерувати звіти по поточному стану бюджету. Також, можливо налаштувати фільтрацію транзакцій і параметрів звіту. Присутній функціонал візуалізації даних у вигляді графіків і діаграм.

						Арк.
						8
Змн.	Арк.	№ Докум.	Підпис	Дата		

Можна зробити висновок, що програма HomeBank добре справляється з поставленими завданнями. Але, також, в ній присутні мінуси: програма поширюється виключно на комп'ютери, що не дає можливості оперативно занести платежі. Також в програмі HomeBank занадто захламичений інтерфейс, що видно на рисунку 1.1, і присутній надмірний функціонал.

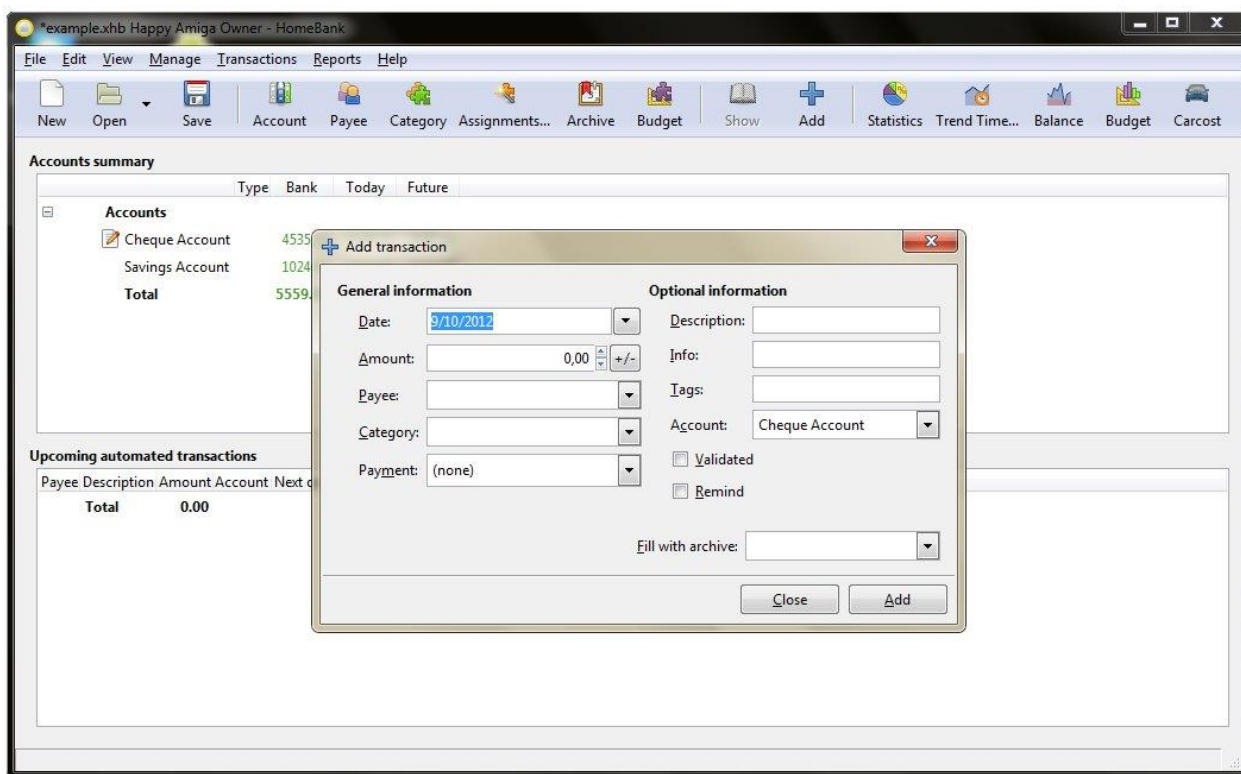


Рисунок 1.1 – Додаток HomeBank

Далі, розглянемо додаток «Кеерсофт домашня бухгалтерія». Цей додаток містить у собі наступний функціонал:

- облік витрат / доходів;
- можливість ведення рахунків;
- можливість розподілу витрат / доходів за категоріями;
- облік боргів;
- складання бюджету витрат і доходів;
- планування бюджету;
- параметризованих пошук;
- можливість застосування фільтрів;

						Арк.
						9
Змн.	Арк.	№ Докум.	Підпис	Дата		

- погашення кредитів і боргів по частинах.

Головним мінусом даної програми є те, що безкоштовного функціоналу мало для того, щоб зручно користуватися програмою. Також відсутня функція візуального представлення даних: у вигляді діаграм. Приклад її роботи можна побачити на рисунку 1.2






☰	Расходы	Рубли ▾	🔍
^	03.03.17	2 530,00р	
	Кошелек Продукты питания / Хл...	10,00р 1 батон	
	Кошелек Транспорт / Метро	20,00р 2 поездка	
	Кредитка Техника / Монитор	2 500,00р 1 шт.	
^	02.03.17	2 150,00р	
	Webmoney Услуги / Печать фотогр...	125,00р	
	Кошелек Транспорт / Автобус	25,00р 1 поездка	
Строк: 6		Общая сумма: 4 680,00р	

Рисунок 1.2 – Додаток «Keepsoft домашня бухгалтерія»

Наступна розглянута програма – Finance PM. Дана програма реалізує наступний функціонал:

- можливість управління гаманцями;
- можливість додавання таких операцій як доходи, витрати і перекази;
- можливість редагувати категорії;
- можливість роботи з декількома валютами;
- можливість регулярних платежів;

						Арк.
						10
Змн.	Арк.	№ Докум.	Підпис	Дата		

- можливість синхронізації даних;
- можливість створення шаблонних операцій.

В цілому, як і раніше розглянутий додаток «Keepsoft домашня бухгалтерія», даний варіант справляється з поставленими для нього завданнями, але має аналогічний мінус – необхідність придбання дорогої ліцензії. Приклад роботи програми можна побачити на рисунку 1.3



Рисунок 1.3 – Додаток «Finance PM»

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ВИКОРИСТОВУВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Цільова платформа

Мета даної роботи – створити додаток з ведення домашньої бухгалтерії для мобільного пристрою. Тому, необхідно вибрати операційну систему, на якій воно буде виконуватися.

Серед головних кандидатів можна виділити Android і iOS. Розглянемо дані операційні системи окремо.

Android – операційна система, розроблена для мобільних пристроїв, електронних книг, смарт-годин, телевізорів та інших пристроїв. Операційна система Android має ядро Linux і має власну реалізацію Java Virtual Machine – Dalvik.

Компанія Google надає можливість писати власні програми на мові програмування Java, використовуючи бібліотеки Android Native Development Kit. Також, надається можливість імпортувати бібліотеки і компоненти додатків мови C та інших.

Серед переваг Android можна виділити:

- Android має найкращі показники х вбудованих функцій, як інтернет-серфінг, взаємодія з додатками Google;
- Android є платформою з відкритим вихідним кодом, що надає широке коло можливостей;
- Android надає встановлювати програми без інтернету, а також підтримує налагодження додатків по USB, що дозволяє писати і тестувати власні програми на своєму Android-пристрої;
- Android доступний для різних апаратних платформ, таких як ARM, MIPS, x86.

						Арк.
						12
Змн.	Арк.	№ Докум.	Підпис	Дата		

Недоліками Android є:

- Android має ряд вразливостей в безпеці;
- у зв'язку з тим, що можна встановлювати додатки з «недовірених джерел», розвивається піратство;
- Android має надмірну фрагментацію, що тягне за собою чималі проблеми для розробників;

Далі, розглянемо операційну систему IOS.

IOS – Операційна система, що випускається компанією Apple. Призначена для смартфонів, планшетів і програвачів

Для створення програмного забезпечення під IOS використовується мова програмування Objective-C або Swift і середовище розробки Xcode. Також, можна розробляти на C і C ++.

Серед переваг IOS можна виділити

- інтерфейс, зрозумілий на інтуїтивному рівні;
- висока, в порівнянні з іншими ОС, швидкість роботи;
- висока практичність додатків;
- відсутність вразливостей безпеки;
- багатозадачність (мале споживання ресурсів згорнути Додатками).

Серед недоліків iOS можна виділити:

- відсутня можливість прямого доступу до файлової системи;
- відсутність можливості писати програми без платного софту;
- відсутність по-своєму налаштувати пристрій;
- обмеження на встановлювані додатки.

На даний час близько 60% всіх мобільних пристроїв знаходяться під управлінням операційної системи Android, коли під керуванням IOS – менше 35%.

Проаналізувавши операційні системи Android і iOS, прийнято рішення, що цільова платформа для розробляється буде Android.

						Арк.
						13
Змн.	Арк.	№ Докум.	Підпис	Дата		

Дане рішення було прийнято за далі описаним причин. По-перше, Android – відкрита платформа, в зв'язку з чим, на ній можна реалізувати більше функцій. По-друге, Android надає широкий, у порівнянні з iOS, вибір API і різних бібліотек. По-третє, Android має більш широку аудиторію, ніж інші операційні системи.

2.2 Мова програмування

У попередньому пункті, цільової операційною системою був обраний Android. Для Android необхідно написати програму на мові програмування, компілятори яких здатні генерувати байт-код для Dalvik.

На даний час найпопулярнішим і раціональним способом є написання Android-додатки на строго типизованій об'єктно-орієнтованій мові програмування Java. Java є об'єктно-орієнтованою мовою програмування і має строгу типізацію.

Перелічимо основні особливості мови:

- програми, написані цією мовою, транслюються в спеціальний байт-код, який може бути розпізнаний на будь-якому пристрої, на якому встановлена віртуальна машина Java;
- програми на Java, під час виконання, повністю контролюються віртуальною машиною, яка підтримує систему безпеки. Будь-які дії, які перевищують повноваження програми викличуть переривання;
- використання віртуальної машини призводить до деякого зниження продуктивності.

Тому, було прийнято низку удосконалень:

- байт-код транслюється в машинний код безпосередньо під час роботи програми, дозволяючи зберігати версії класів в машинному коді;
- у стандартних бібліотеках використовується код, орієнтований на платформу;

						Арк.
						14
Змн.	Арк.	№ Докум.	Підпис	Дата		

- використовуються апаратні засоби, що прискорюють обробку байт-коду. Серед основних можливостей Java можна виділити: автоматизована робота з пам'яттю, система обробки винятків, велика стандартна бібліотека класів, можливість створення мережевих додатків, можливість створення багатопотокових додатків, система узагальнень, можливість виразів лямбда виразів та замикань.

Так як Java є об'єктно-орієнтованою мовою програмування, то тут реалізуються три основні парадигми.

- інкапсуляція – властивість системи, що дозволяє об'єднати дані і методи, що працюють з ними, в класі;
- спадкування – властивість системи, що дозволяє описати новий клас на основі вже існуючого, з частково або повністю запозичує функціональність. Клас, від якого виробляється спадкування, називається базовим, батьківським або суперкласом. Новий клас – нащадком, спадкоємцем, дочірнім або похідним класом;
- поліморфізм – властивість системи, що дозволяє використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта.

2.3 Система управління базами даних

В додатку, що розробляється необхідно постійно впорядковано зберігати набір даних, з якими працює користувач. Для цієї мети самим раціональним рішенням буде використання бази даних. Для управління створення бази даних буде використовуватися система управління базами даних.

Так як Android надає API-інтерфейси, необхідні для використання бази даних SQLite, то в додатку, що розробляється буде використовуватися саме ця система управління базами даних.

SQLite є вбудованої реляційної системою управління базами даних.

						Арк.
						15
Змн.	Арк.	№ Докум.	Підпис	Дата		

SQLite не використовує парадигму клієнт-сервер, тому що використовується движок є не окремо працюючим процесом, з яким взаємодіє додаток, а надає бібліотеку, з якої можливо компонувати програму. В такому випадку движок стає складовою частиною програми. Для обміну даними використовуються виклики функцій бібліотеки SQLite.

Використовуючи такий підхід, збільшується продуктивність, зменшуються витрати ресурсів, а також спрощується процес написання програми.

SQLite зберігає базу даних в єдиному файлі на тому комп'ютері, на якому виконується програма. Незважаючи на такий похід, дані з бази даних можна читати з різних процесів і потоків.

API-інтерфейси, необхідні для використання бази даних на платформі Android, доступні в складі пакету android.database.sqlite.

						Арк.
						16
Змн.	Арк.	№ Докум.	Підпис	Дата		

3 МОДЕЛЮВАННЯ ПРОЦЕСІВ

Для поліпшення якості розробки програмного забезпечення, необхідно провести ретельне моделювання процесів. До того, як писати код безпосередньо, потрібно описати процес через різні елементи: дії, дані, події. Моделювання процесів описує логічний взаємозв'язок елементів, властивих процесу.

Завдяки моделюванню процесів, можна зрозуміти роботу в цілому і провести її аналіз. Для великих проектів варто виконувати докладний і багатогранне моделювання. Моделювання влучно проводити ітеративно – тобто, після першого варіанту схеми побачити недоліки у самих коренях, та перебудувати, відповідним чином. Після декількох ітерацій – скласти кінцевий варіант та деком позувати його якнайбільш можливо. Декомпозовані блоки найкраще будуть показувати роботи системи як ззовні, так і всередині.

Серед основних цілей моделювання можна виділити:

- описати процеси. Моделюючи процеси, можна простежити, що виконується в процесах від їх початку до завершення. Це дозволяє отримати загальний погляд на процеси і допомагає збільшити їх ефективність;
- нормувати процеси. Моделювання процесів задає правила виконання процесів, тобто то, яким чином вони повинні бути виконані;
- встановлення взаємозв'язку процесів. Під час моделювання встановлюється зв'язок між процесами і вимогами до них.

У даній роботі буде розглянуто функціональне моделювання – вид моделювання, що має на увазі опис процесів у вигляді взаємопов'язаних, чітко структурованих функцій. У цьому виді моделювання реальну послідовність функцій дотримуватися не обов'язково.

						Арк.
						17
Змн.	Арк.	№ Докум.	Підпис	Дата		

Функціональне моделювання буде виконано за допомогою методології IDEF0. Такий тип діаграми дозволить з достатньою глибиною декомпозиції описати усі функції системи та описати ієрархію.

3.1 Методологія IDEF0

Методологія IDEF0 передбачає побудову ієрархічної системи діаграм - одиничних описів функцій в системі. Спочатку проводиться найбільш загальний опис системи, її зовнішня взаємодія (контекстна діаграма), після чого проводиться декомпозиція. Під час декомпозиції система розбивається на інші системи (підсистеми) і кожна підсистема описується окремо. Потім, для досягнення необхідного ступеня деталізації, кожна підсистема декомпонується.

Кожна IDEF0-діаграма містить блоки і дуги. Блоки зображують функції моделюваної системи. Дуги позначають взаємозв'язок і взаємодія між блоками.

Функціональні блоки (зображуються прямокутниками) означають функції, процеси або завдання, що виконуються в системі. Ім'я блоку має виражатися іменником, що позначає дію.

Блоки в IDEF0 розміщуються за принципом домінування: самий домінуючий блок розташовується зверху, менш домінуючі – нижче. Під домінуванням розуміється ступінь впливу, який один блок надає на інший.

Взаємодія блоків описується в вигляді стрілок, зображуваних одинарними лініями зі стрілками на кінцях. Стрілки являють собою якусь інформацію і іменуються іменниками.

В IDEF0 розрізняють кілька типів стрілок:

- вхід – входить в роботу зліва і показує інформаційні потоки, які перетворюються в процесі;
- управління – входить в роботу зверху і показує матеріальні і інформаційні потоки, які перетворюються на процесі, але потрібні для його виконання;

						Арк.
						18
Змн.	Арк.	№ Докум.	Підпис	Дата		

- механізм – входить в роботу знизу і показує людей, технічні засоби, інформаційні системи, за допомогою яких процес реалізується;
- результат – виходить з роботи праворуч і показує дані, матеріальні та інформаційні потоки, які повертаються блоком.

Структурну схему діаграми методології IDEF0 можна розглянути на рисунку 3.1.

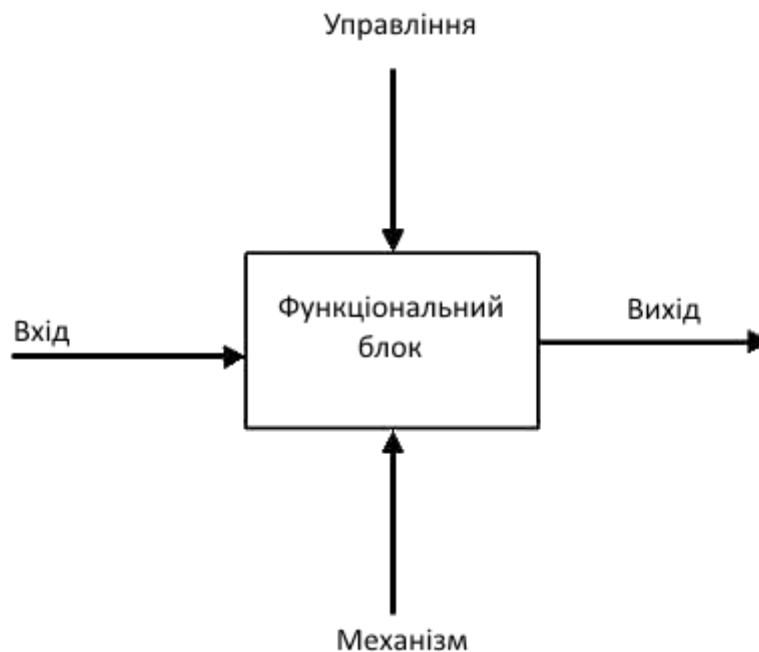


Рисунок 3.1 – Структурна схема діаграми методології IDEF0

3.2 IDEF0-діаграма для продукту, що розробляється

Почнемо з побудови контекстної IDEF0-діаграми. Основною функцією розробляється є обслуговування користувача, за допомогою обробки дій, здійснених нею. Тому, визначимо єдину роботу контекстної діаграми як «Обслуговування користувача додатки». Далі визначимо вхідні і вихідні дані, а так леї механізми і управління.

Для обслуговування користувача, необхідно вивести користувачеві необхідні дані, надати можливість сформулювати запити, відкрити доступ до бази даних і обробити його запит.

						Арк.
						19
Змн.	Арк.	№ Докум.	Підпис	Дата		

В якості вхідних даних буде використовуватися «Вихідна база даних» і «Запити користувача». Наслідком обробки запитів користувача є зміна бази даних, тому вихідними даними буде «змінена база даних». Процес буде виконуватися автоматизованими засобами програми, використовуючи методи збереження та обробки інформації.

Зважаючи на вищенаведений опис, побудуємо контекстну діаграму роботи «Домашньої бухгалтерії». Контекстну діаграму зображено на рисунку 3.2

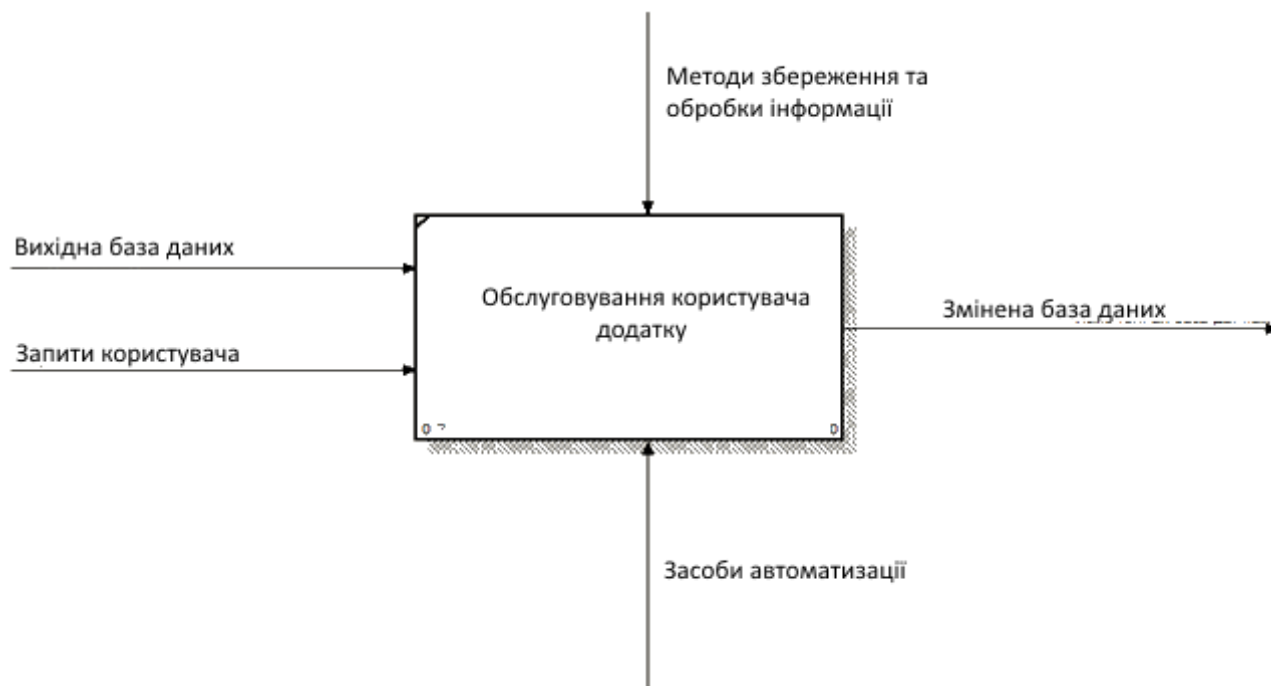


Рисунок 3.2 – Контекстна IDEF0-діаграма «Домашньої бухгалтерії»

Для уточнення всіх процесів зробимо декомпозицію контекстної діаграми. Маємо наступну послідовність обслуговування користувача програми:

1. Звернення до системи
2. Обробка запиту користувача
3. Зміна бази даних

Побудуємо декомпозицію контекстної діаграми та деком позиуємо усі наступні блоки. Результат декомпозиції контекстної діаграми зображено на рисунку 3.3.

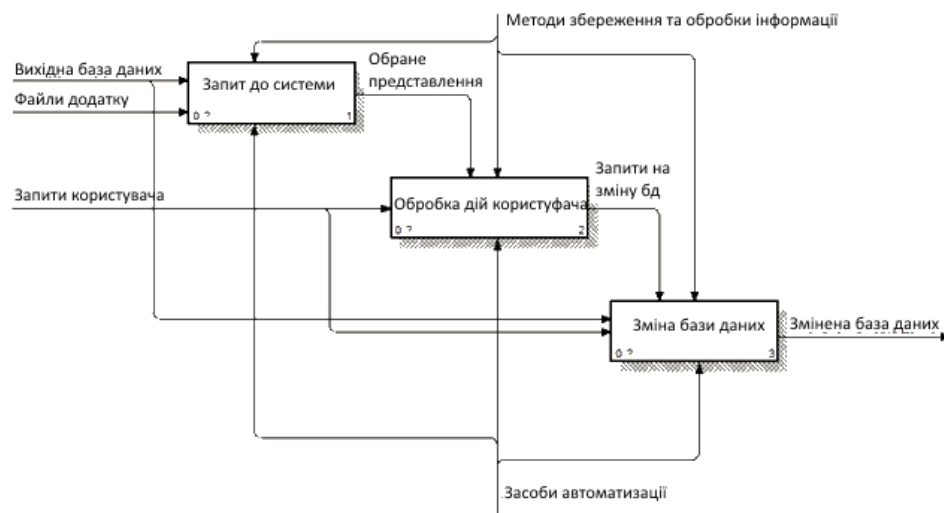


Рисунок 3.3 – Декомпозиція контекстної діаграми

3.2.1 Запит до системи

Коли відбувається запуск програми, виконується звернення до системи. Перш за все, аналізуються файли налаштувань, а зокрема файл AndroidManifest.xml. У ньому описуються основні компоненти програми: служби, діяльності, контент-провайдери, оголошується ім'я Java-пакета додатка, оголошуються дозволу, перераховуються сторонні бібліотеки. Також вказується мінімальний рівень API Android, необхідний для роботи програми.

Далі, на основі оброблених налаштувань, вибирається необхідна Activity - клас, який представляє окремий вікні керування або його візуальний інтерфейс.

Для обраного раніше Activity, завантажуються дані з бази даних, виводяться в необхідні компоненти і представляються користувачеві.

Результат декомпозиції роботи «Звернення до системи» зображено на рисунку 3.4

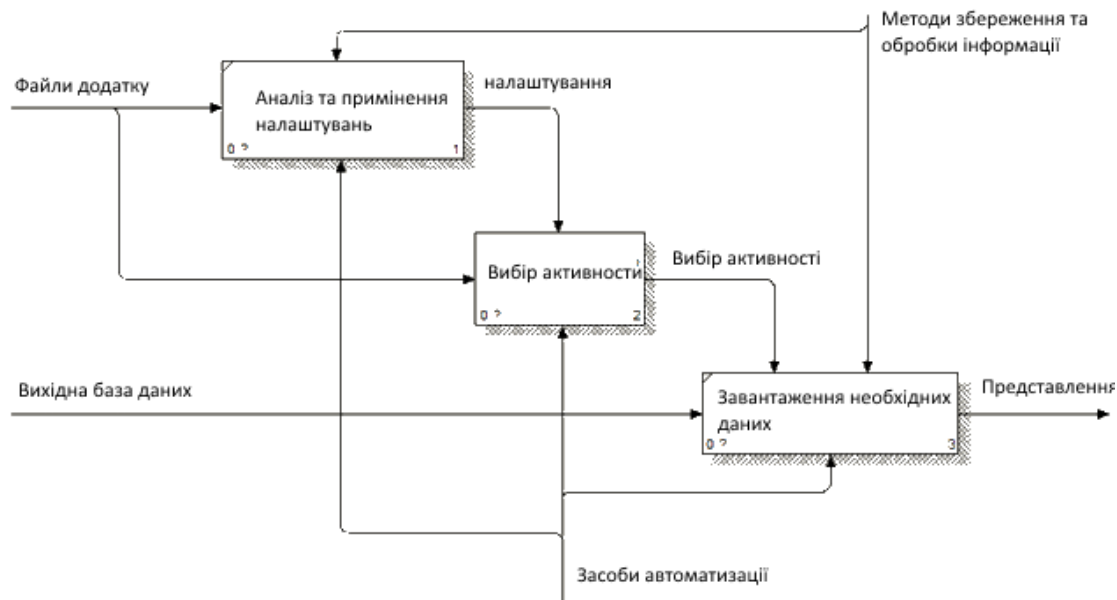


Рисунок 3.4 – Декомпозиція роботи «Звернення до системи»

Детальніше опишемо процес вибору Activity.

Визначення стартового Activity знаходиться в файлі AndroidManifest.xml. При його аналізі необхідно знайти Activity, яке містить атрибути, що вказують на те, що він стартовий. Далі в пакеті програми знаходиться клас, який йому відповідає. Далі управління передається обраному класу.

Результат декомпозиції роботи «Вибір необхідного Activity» зображено на рисунку 3.5.



Рисунок 3.5 – Декомпозиція роботи «Вибір необхідного Activity»

3.2.2 Обробка дій користувача

Після того, як користувачу представилася необхідна інформація, він отримує можливість маніпулювати цими даними. Необхідно коректно обробляти будь-які його дії.

Перш за все, система очікує введення користувачем деяких даних. Після того, як він вводить дані, і дає команду на їх обробку (наприклад, кнопка «зберегти»), система отримує дані.

Далі, відбувається обробка даних, а спочатку – перевірка їх коректності. Система проводить перевірку даних і в разі некоректного введення інформує про це користувача. Якщо система отримує коректні дані – відбувається їх обробка, і формуються запити на зміну бази даних.

Результат декомпозиції роботи «Обробка дій користувача» зображено на рисунку 3.6.

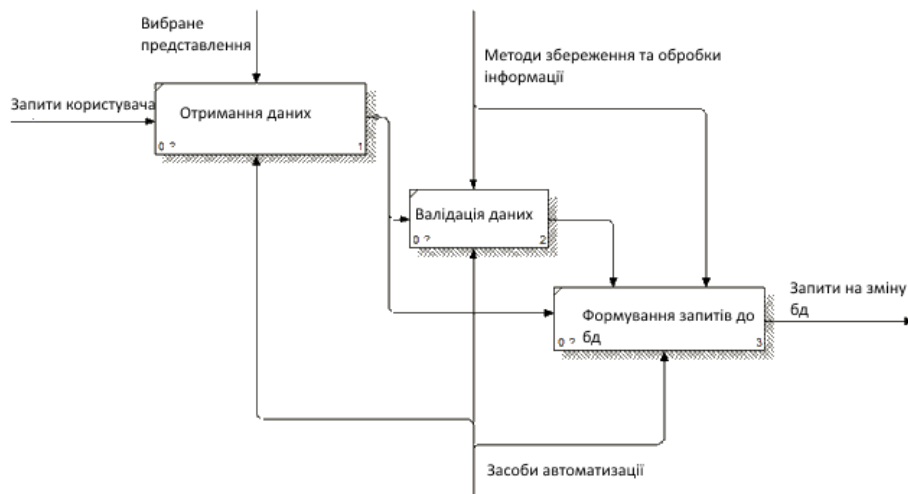


Рисунок 3.6 – Декомпозиція роботи «Обробка дій користувача»

3.2.3 Зміна бази даних

Після дій користувача, були сформовані запити на зміну бази даних. Запити необхідно обробити і внести відповідні зміни в базу даних.

Спочатку, в залежності від того на якому з Activity проводилися дії користувача, вибирається необхідна таблиця. Далі, необхідно сформувати пул даних. Після формування пулу, необхідно внести його в базу даних.

Після виконання даних операцій, користувачеві знову представляється Activity з можливістю повторного маніпулювання даними.

Результат декомпозиції роботи «Зміна бази даних» зображено на рисунку 3.7.

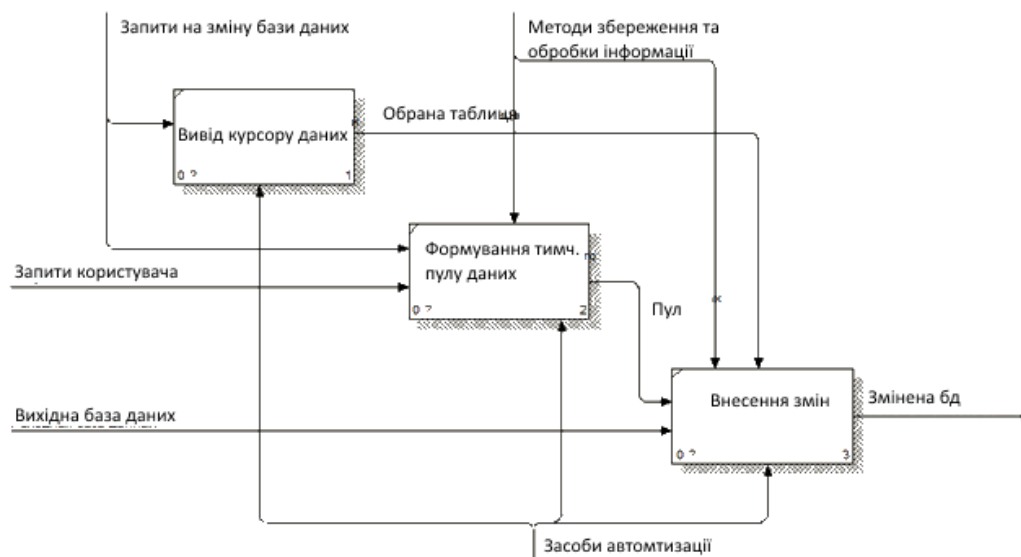


Рисунок 3.7 – Декомпозиція роботи «Зміна бази даних»

4 ОПИС АЛГОРИТМІВ

Наведемо основні алгоритми, які використовуються в додатку, що розробляється. Проведемо графічну візуалізацію за допомогою блок-схем.

4.1 Основний цикл програми

Запускаючи додаток, користувач отримує переглянути дані про платежі за певний період часу, а також окремо для категорій витрат і поповнень. Також, надається можливість переглянути існуючі рахунки та категорії.

Далі, користувач може прийняти рішення про створення рахунку, або категорії. При наявності рахунку і категорій, надається можливість створити новий платіж. Після створення нового платежу, відбувається перегляд діаграм платежів.

Після цього, основний цикл програми або закінчується, або циклічно повторюється, починаючи з першого етапу.

Основні процеси детальніше будуть описані в наступних підрозділах. Блок-схему основного циклу програми зображено на рисунку 4.1.

						Арк.
						25
Змн.	Арк.	№ Докум.	Підпис	Дата		



Рисунок 4.1 – Блок-схема основного циклу програми

4.2 Створення рахунків

Натискаючи відповідну кнопку в вікні рахунків, користувач переходить до вікна створення рахунку.

На рисунку 4.2 зображено блок-схему алгоритму створення рахунку. Спочатку, користувачем вводиться назва рахунку, а далі – початкова сума на рахунку. При натисканні кнопки для збереження, відбувається перевірка введених даних. При успішній перевірці – дані зберігаються в базу даних, інакше – виводиться повідомлення про помилку, і вимоги до параметру, який введено неправильно.

						Арк.
						26
Змн.	Арк.	№ Докум.	Підпис	Дата		

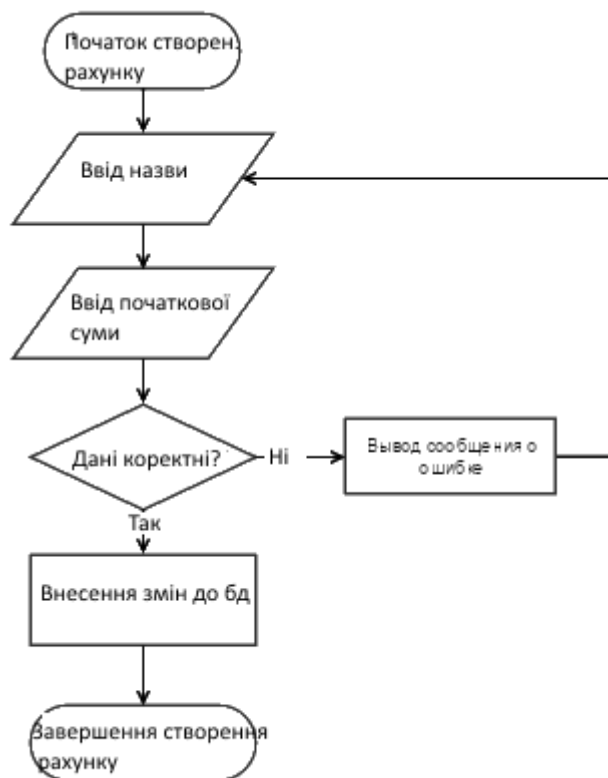


Рисунок 4.2 – Блок-схема алгоритму створення рахунку

4.3 Створення категорій

Натискаючи відповідну кнопку в вікні категорій, користувач переходить до вікна створення категорії.

Для створення нової категорії, користувач вводить назву нової категорії, вибирає тип категорії (трата або поповнення), а також вибирає іконку для нової категорії.

Перш ніж зберегти нову категорію, проходить перевірка введених даних, і, в разі знаходження помилки – користувачеві виводиться відповідне повідомлення.



Рисунок 4.3 – Блок-схема алгоритму створення категорії

4.4 Створення рахунку

Для створення платежу, користувач вводить дату і час здійснення платежу. Далі, відбувається введення суми платежу, а також вибір категорії платежу і рахунку для якого дана операція проводиться.

Перед збереженням платежу в базі, дані введені користувачем підлягають перевірці. При знаходженні невірних даних виводиться повідомлення про помилку, і вимоги до параметру, який введено неправильно.

Також, якщо користувач вибере опцію зберегти поточний платіж і створити новий, то алгоритм повториться з самого початку, інакше – завершити алгоритм.

Блок-схема вищеописаної послідовності створення категорії зображена на рисунку 4.4.

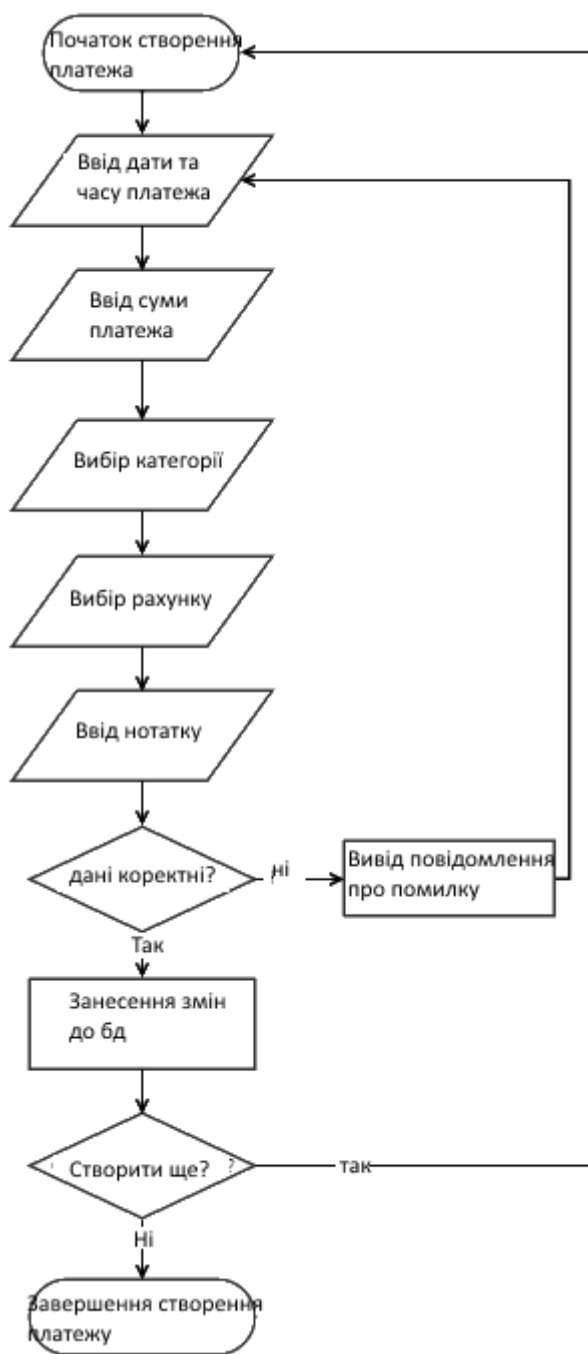


Рисунок 4.4 – Блок-схема алгоритму створення рахунку

5 СТРУКТУРА ДАНИХ І РЕСУРСІВ ПРОГРАМИ

У цьому розділі наводяться результати розробки структур і ресурсів даних, з якими працює програма і їх характеристики. Вихідний код програми наведено у додатку А.

Для класів будуть описані:

- Опис і призначення класу;
- Опис і призначення змінних класу;
- Опис і призначення методів класу;
- Схема родинних класів, якщо така є.

Також, приводиться детальний опис бази даних.

5.1 Клас MainActivity

В даному класі описана логіка стартового вікна програми. Реалізується логіка бокового меню, а також логіка перегляду журналу всіх платежів.

Базовий клас: AppCompatActivity

Реалізовані інтерфейси :

NavigationView.OnNavigationItemSelectedListener

Поля класа:

- DateSelector dateSelector – об'єкт, що генерує різні відрізки часу: день, тиждень, місяць, рік і показує їх користувачеві.

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів;
- void onBackPressed() – обробка натискання кнопки «назад» на пристрої;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання елементів в бічному меню. Викликає нове Activity;

						Арк.
						30
Змн.	Арк.	№ Докум.	Підпис	Дата		

- DateSelector getDateSelector() – геттер поля dateSelector, призначений для фрагментів, що викликаються даними Activity.

Владені класи:

- SectionsPagerAdapter extends FragmentStatePagerAdapter – відповідає за посторінковий перегляд журналу. Надає можливість перегляду журналу за категоріями: «Все», «Витрати», «Поповнення». використовує фрагмент FragmentTransactions.

5.2 Клас UsingDataBaseListFragment

Даний клас є абстрактним і надає можливість виведення списку елементів

Базовий клас: ListFragment

Реалізовані інтерфейси: Відсутні

Поля класу:

- SQLiteHandler handler – об'єкт для управління створенням бази даних і управлінням версіями;
- SQLiteDatabase db – об'єкт, який має методи для створення, видалення, виконання команд SQL і виконання інших загальних завдань управління базами даних;
- Cursor cursor – об'єкт, що забезпечує довільний доступ на читання і запис до набору результатів, що повертається запитом бази даних.

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації полів класу. Також відкриває доступ до бази даних;
- void onDestroy() – закриває використовувані ресурси.

						Арк.
						31
Змн.	Арк.	№ Докум.	Підпис	Дата		

5.3 Клас FragmentTransactions

В даному класі описаний Fragment, який реалізує висновок платежів. Тут генерується контекстне меню при довгому натисненні на елемент списку. Для маніпуляції платежами викликає SingleTransactionActivity.

Базовий клас: UsingDataBaseListFragment

Реалізовані інтерфейси: Відсутні

Поля класу:

- String query – запит до бази даних для отримання платежів;
- SimpleCursorAdapter transactionsCursorAdapter – адаптер для виведення платежів;
- DateSelector dateSelector – об'єкт, що генерує різні відрізки часу: день, тиждень, місяць, рік і показує їх користувачеві.

Методи класу:

- static FragmentTransactions FragmentTransactionsFactory(String query) – фабричний метод для генерації об'єктів класу, ініціалізувавши поле query переданим значенням;
- void onActivityCreated(@Nullable Bundle savedInstanceState) – ініціалізація полів і обробників подій;
- void onResume() – обробка повернення користувача до Fragment;
- void onCreateOptionsMenu(Menu menu, MenuInflater inflater) – генерація меню;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання на елементі меню;
- void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) – генерація контекстного меню
- boolean onContextItemSelected(MenuItem item) – обробка натискання на елементі в контекстному меню;
- setQuery(String query) – ініціалізація поля query переданим значенням;

						Арк.
						32
Змн.	Арк.	№ Докум.	Підпис	Дата		

- `String getBaseQuery()` – метод для отримання запиту до бази даних, що повертає всі платежі;
- `void initializeAdapter()` – ініціалізація адаптера для виведення елементів списку;
- `void requeryCursor()` – перезавантаження даних для курсора;
- `String initializeConditions(String query)` – застосування фільтра відрізка часу для отримання платежів;
- `decreaseDate(View view)` – зменшити на одиницю відрізок часу; `increaseDate(View view)` – збільшити на одиницю відрізок часу;
- `updateCursors()` – оновити курсори всіх фрагментів, що зберігаються у `FragmentManager`.

5.4 Клас `UsingDataBaseActivity`

Даний клас є абстрактним і надає можливість `Activity` працювати з базами даних. Виробляє відкриття і закриття використовуваних ресурсів.

Базовий клас: `AppCompatActivity`

Реалізовані інтерфейси: Відсутні

Поля класу:

- `SQLiteHandler handler` – об'єкт для управління створенням бази даних і управлінням версіями;
- `SQLiteDatabase db` – об'єкт, який має методи для створення, видалення, виконання команд SQL і виконання інших загальних завдань управління базами даних;
- `Cursor cursor` – об'єкт, що забезпечує довільний доступ на читання і запис до набору результатів, що повертається запитом бази даних.

Методи класу:

- `void onCreate(Bundle savedInstanceState)` – виконання ініціалізації полів класу. Також відкриває доступ до бази даних;

						Арк.
						33
Змн.	Арк.	№ Докум.	Підпис	Дата		

- void onDestroy() – закриває використовувані ресурси.

5.5 Клас SingleEntityActivity

Даний клас є абстрактним і надає можливість Activity працювати з конкретними сутностями: рахунок, платіж, категорія.

Базовий клас: UsingDataBaseActivity

Реалізовані інтерфейси: Відсутні

Поля класу:

- long entityId – Id записи елемента в базі даних;

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів. Також отримує entityId з переданих значень;
- boolean onCreateOptionsMenu – генерирує меню Activity;
- abstract boolean onOptionsItemSelected(MenuItem item) – обробка натискання елемента в меню.

5.6 Клас AccountsActivity

В даному класі описано Activity, що надає інтерфейс доступу до рахунків. Тут виводиться список рахунків, генерується контекстне меню при довгому натисненні на елемент. Для маніпуляції рахунками викликає SingleAccountActivity.

Базовий клас: UsingDataBaseActivity

Реалізовані інтерфейси: Відсутні

Поля класу:

- ListView accountsListView – список для відображення рахунків;
- SimpleCursorAdapter accountsCursorAdapter – адаптер для відображення елементів списку;

						Арк.
						34
Змн.	Арк.	№ Докум.	Підпис	Дата		

Методи класу:

- void onCreate(Bundle savedInstanceState) виконання ініціалізації всіх фрагментів і завантажувачів;
- void onResume() – виконує завантаження даних, коли користувач повертається до Activity;
- void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) – генерує контекстне меню при довгому натисненні на елементі списку;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання елемента в контекстном меню;
- void initializeAdapter() – ініціалізація адаптера для виведення елементів списку;
- void requeryCursor() – перезавантаження даних для курсора.

5.7 Клас SingleAccountActivity

В даному класі описано взаємодію з конкретним рахунком користувача: створення, зміна та видалення.

Базовий клас: SingleEntityActivity

Реалізовані інтерфейси: Відсутні

Поля класу:

- TextView titleTextView, startBalanceTextView – поля редагування назви рахунку і початкової балансу відповідно;

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання елемента в меню;
- void initializeViews() – ініціалізація елементів уявлення;

						Арк.
						35
Змн.	Арк.	№ Докум.	Підпис	Дата		

- void saveAccount(View view) – валідація даних і збереження рахунку;

5.8 Клас CategoriesActivity

В даному класі описано Activity, що надає інтерфейс доступу до категорій. Реалізується логіка відображення категорій і поділу категорій на групи.

Базовий клас: AppCompatActivity

Реалізовані інтерфейси: Відсутні

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів;

Вкладені класи:

- SectionsPagerAdapter extends FragmentStatePagerAdapter – відповідає за посторінковий перегляд категорій. Надає можливість перегляду категорій по групах: «Витрати», «Поповнення». використовує клас FragmentCategories.

5.9 Клас FragmentCategories

В даному класі описаний Fragment, який реалізує висновок категорій. Тут генерується контекстне меню при довгому натисненні на елемент списку. Для маніпуляції категоріями викликає SingleCategoryActivity.

Базовий клас: UsingDataBaseListFragment

Реалізовані інтерфейси: Відсутні

Поля класу:

- String query – запит до бази даних для отримання платежів;
- SimpleCursorAdapter categoriesCursorAdapter – адаптер для виведення категорій;

Методи класу:

						Арк.
						36
Змн.	Арк.	№ Докум.	Підпис	Дата		

- static FragmentCategories FragmentCategoriesFactory (String query) – фабричний метод для генерації об'єктів класу, ініціалізувавши поле query переданим значенням;
- void onActivityCreated(@Nullable Bundle savedInstanceState) – ініціалізація полів і обробників подій;
- void onResume() – обробка повернення користувача до Fragment;
- void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) – генерація контекстного меню;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання на елементі в контекстному меню;
- setQuery(String query) – ініціалізація поля query переданим значенням;
- String getBaseQuery() – метод для отримання запиту до бази даних, що повертає всі платежі;
- void initializeAdapter() – ініціалізація адаптера для виведення елементів списку;
- void requeryCursor() – перезавантаження даних для курсора.

5.10 Клас SingleCategoryActivity

В даному класі описано взаємодію з конкретною категорією платежу користувача: створення, зміна та видалення.

Базовий клас: SingleEntityActivity

Реалізовані інтерфейси: Відсутні

Поля класу:

- TextView titleTextView – поле редагування назви категорії;
- RadioButton incomeCategoryRadioButton, outgoCategoryRadioButton – радіо-кнопки вибору типу категорії (трата / поповнення);
- Spinner logoSpinner – випадає для вибору логотипу категорії;

						Арк.
						37
Змн.	Арк.	№ Докум.	Підпис	Дата		

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання елемента в меню;
- void initializeViews() – ініціалізація елементів уявлення;
- void initializeSpinners() – ініціалізація списку;
- void saveCategory (View view) – валідація даних і збереження категорії.

5.11 Клас SingleTransactionActivity

В даному класі описано взаємодію з конкретним платежем користувача: створення, зміна та видалення.

Базовий клас: SingleEntityActivity

Реалізовані інтерфейси: Відсутні

Поля класу:

- TextView currentDateTextView, currentTimeTextView – поля виведення дати і часу відповідно;
- Spinner categoriesSpinner, accountsSpinner – випадають списки категорій і рахунків;
- EditText transactionAmountEditText, noteEditTextEditText – поля введення суми платежу і замітки відповідно;
- Button signButton – кнопка управління знаком суми платежу;

Методи класу:

- void onCreate(Bundle savedInstanceState) – виконання ініціалізації всіх фрагментів і завантажувачів;
- void onResume() – обробка повернення користувача до Activity;
- boolean onOptionsItemSelected(MenuItem item) – обробка натискання елемента в меню;

						Арк.
						38
Змн.	Арк.	№ Докум.	Підпис	Дата		

- void initializeViews() – ініціалізація елементів уявлення;
- void initializeSpinners() – ініціалізація випадючих списків;
- void initializeListeners() – ініціалізація обробників подій;
- void setInitialDateTime() – установка поточного часу і його відображення;
- void setDate(View view) – збереження дати, обраної користувачем;
- void setTime(View view) – збереження часу, обраного користувачем;
- void saveTransactionContinue(View view) – обробник натискання кнопки «Зберегти і новий»;
- void saveTransactionClose(View view) – обробник натискання кнопки «Зберегти»;
- boolean executeSaving() – збереження даних;
- boolean executeDataBaseSaving() – збереження даних в базу даних;
- boolean validateData() – валідація даних;
- void addNewCategory (View view) – обробка натискання кнопки створення нової категорії;
- void addNewAccount(View view) – обробка натискання кнопки створення нового рахунку;

5.12 Клас SQLiteHandler

Клас, який реалізує управління створенням бази даних і управлінням її версіями.

Базовий клас: SQLiteOpenHelper

Реалізовані інтерфейси: Відсутні

Поля класу:

- static final int DATABASE_VERSION – версія бази даних;
- static final String DATABASE_NAME – назва бази даних;

						Арк.
						39
Змн.	Арк.	№ Докум.	Підпис	Дата		

- Context context – контекст, в якому використовуються об'єкти даного класу.

Методи класу

- SQLiteHandler (Context context) – конструктор класу. Проводиться ініціалізація змінних;
- void onCreate(SQLiteDatabase db) – первинне створення бази даних;
- void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) – оновлення бази даних;
- void onConfigure(SQLiteDatabase db) – конфігурація підключення до бази даних;
- createCategories(SQLiteDatabase db) – первинне заповнення бази даних категоріями;
- createAccounts(SQLiteDatabase db) – первинне заповнення бази даних рахунками.

5.13 Опис бази даних

Схема бази даних, використовуваної в додатку, що розробляється, має такий вигляд, зображений на рисунку 5.1.

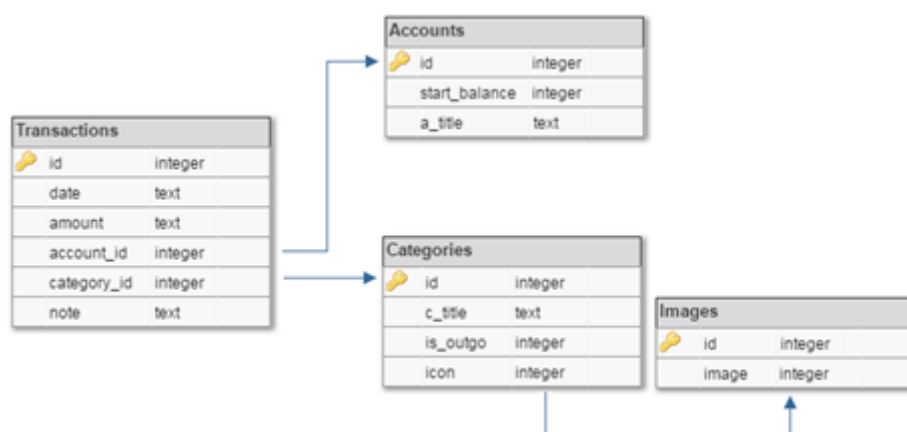


Рисунок 5.1 – Схема бази даних

Таблиця Transactions – кожен запис таблиці являє собою платіж в додатку і складається з наступних полів:

- date – дата платежу;
- amount – сума платежу;
- account_id – id рахунку, для якого здійснюється платіж;
- category_id – id категорії, для якої здійснюється платіж;
- note – замітка.

Таблиця Accounts – кожен запис таблиці являє собою рахунок в додатку, і складається з наступних полів:

- start_balance – баланс рахунку на момент його створення;
- a_title – назва рахунку.

Таблиця Categories – кожен запис таблиці являє собою категорію в додатку, і складається з наступних полів:

- c_title – назва рахунку
- is_outgo – поле, визначальне трата це або поповнення;
- icon – іконка категорії;
- Таблиця Images – кожен запис таблиці являє собою зображення в додатку;
- image – назва зображення з файлів ресурсів.

Кожна таблиця містить поле id, яке унікально ідентифікує кожну запис.

Між такими полями, як id рахунку у таблиці транзакцій та id безпосередньо рахунку встановлено обов’язковий зв’язок, та каскадне видалення даних. Тобто, про видаленні одного рахунку, всі його транзакції будуть видалені.

Між такими полями, як id каетгорії у таблиці транзакцій та id безпосередньо категорії також встановлено обов’язковий зв’язок, та каскадне видалення даних.

						Арк.
						41
Змн.	Арк.	№ Докум.	Підпис	Дата		

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Початок роботи з додатком

Використовуючи цю програму, Ви завжди маєте можливість виконувати наступні дії:

- проводити операції над платежами;
- додавати / змінювати / видаляти категорії;
- додавати / змінювати / видаляти рахунку;
- перегляд статистики у вигляді діаграм;
- перегляд журналу за певний період часу;
- перегляд журналу окремо для витрат і доходів;
- перегляд категорій окремо для витрат і доходів;
- перегляд рахунків.

6.2 Вікно «Журнал»

При першому запуску програми користувач побачить наступне вікно, зображене на рисунку 6.1:

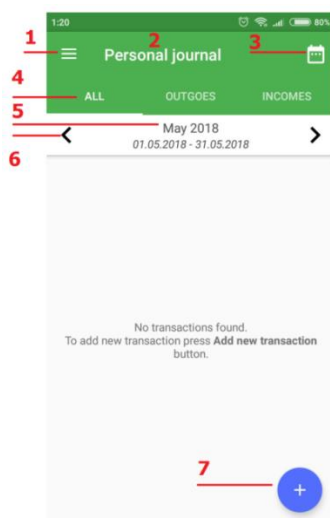


Рисунок 6.1 – Вікно додатки «Журнал»

						Арк.
						42
Змн.	Арк.	№ Докум.	Підпис	Дата		

Опишемо детально кожен з замаркованих елементів.

1. Кнопка відкриття меню додатку. При її натисканні буде відкрито бічне меню. Також, його можна відкрити за допомогою слайда з краю екрану зліва направо. Детальніше меню буде описано в одному з наступних підрозділів.
2. Тема вікна.
3. Меню вибору кроку періоду для відображення журналу. Доступні опції: день, тиждень, місяць, рік. Після зміни кроку тимчасового періоду, дані будуть оновлені автоматично і будуть відображені для періоду, що містить поточну дату.
4. Тема посторінкового перегляду. Журнал можна переглядати для всіх платежів, а також окремо для витрат і поповнень. Змінити сторінку перегляду можна або натиснувши на один із заголовків або слайдом по екрану в одну зі сторін..
5. Період, для якого відображається журнал. Значення показані в двох форматах: дд.мм.rrrr – дд.мм.rrrr і в зручному зручно читається форматі.
6. Кнопка переходу до попереднього періоду часу (аналогічна і для переходу до наступного періоду часу). Натиснувши на неї, буде оновлено період часу і відповідно оновлені відображаються дані. Наприклад, якщо Ви вибрали «Місяць», то після натискання даної кнопки, будуть відображені дані для «Березень 2017».
7. Кнопка додавання нового платежу. Нажав на неї, користувач перейде до нового вікна.

Після додавання деяких записів, вікно журнал прийме вигляд, зображений на рисунку 6.2 або альтернативний вигляд у іншій вкладці, зображений на рисунку 6.3 (процес додавання нових записів буде розглянуто далі).

						Арк.
						43
Змн.	Арк.	№ Докум.	Підпис	Дата		

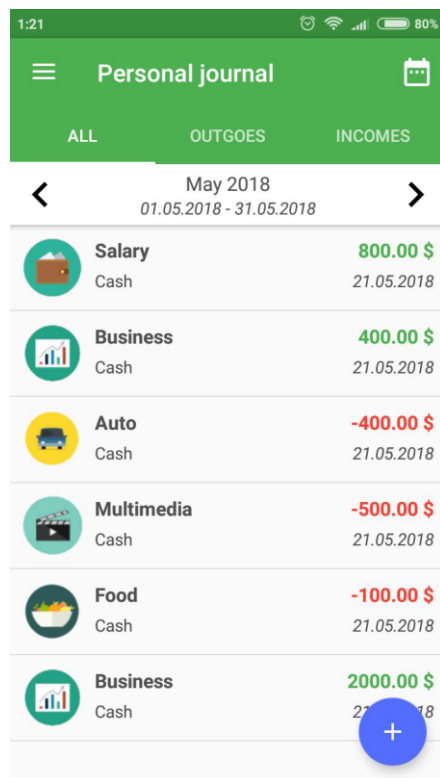


Рисунок 6.2 – Вікно додатки «Журнал». Група «Всі»

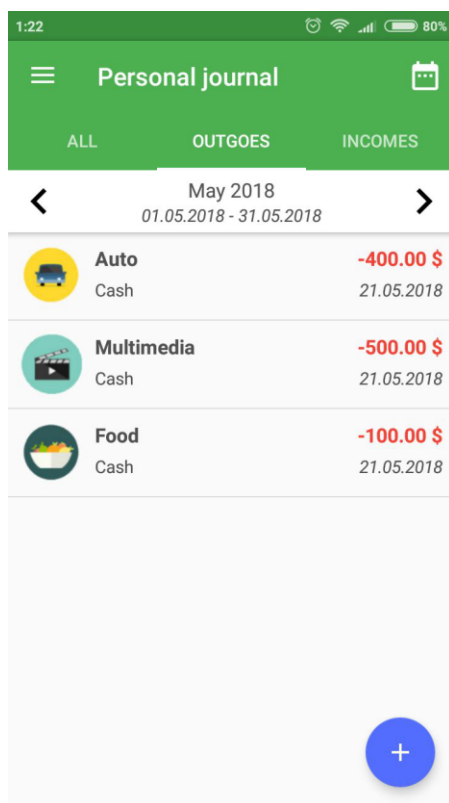


Рисунок 6.3 – Вікно додатки «Журнал». Група «Витрати»

						Арк.
						44
Змн.	Арк.	№ Докум.	Підпис	Дата		

Кожен запис в журналі має стандартну форму, зображену на рисунку 6.4.
Розглянемо її докладніше.

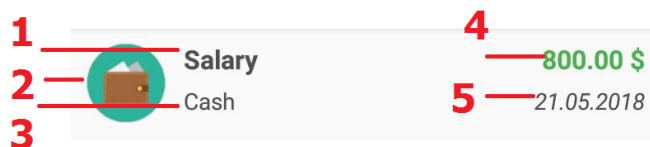


Рисунок 6.4 – Приклад журналу.

1. Назва категорії, для якої відбувався платіж.
2. Логотип категорії.
3. Назва рахунку, для якого відбувався платіж.
4. Сума платежу. У разі поповнення колір буде зеленим, в зворотному випадку – червоним.
5. Дата здійснення платежу.

6.3 Вікно «Платіж»

Детально розглянемо кожен елемент вікна «Платіж», зображеного на рисунку 6.5.

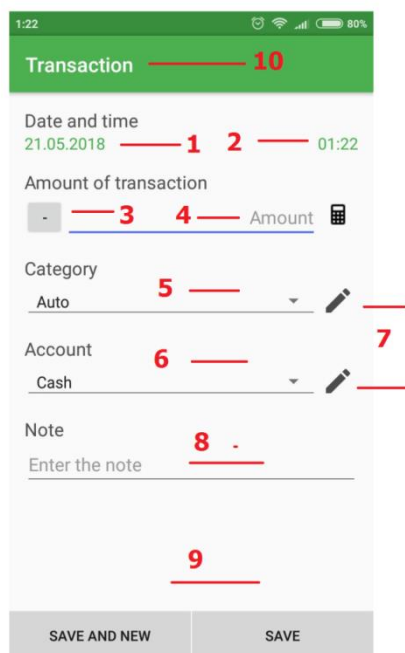


Рисунок 6.5 – Вікно додатки «Платіж»

						Арк.
						45
Змн.	Арк.	№ Докум.	Підпис	Дата		

1. Текстове поле із зазначенням дати платежу. При натисканні на кожне – з'явиться діалогове вікно з можливістю змінити цей параметр.
2. Текстове поле з зазначенням часу платежу. При натисканні результат аналогічний датою.
3. Поле для вибору знака суми (негативна або позитивна).
4. Поле для введення суми платежу.
5. Список, що випадає для вибору категорії платежу.
6. Список, що випадає для вибору рахунку, для якого здійснюється платіж.
7. 7. Кнопки для додавання нової категорії і рахунки.
8. Поле для введення замітки про платіж.
9. опка «Зберегти» зберігає введені дані і повертає користувача до перегляду платежів, а кнопка «Зберегти і новий» – зберігає платіж і надає можливість ввести новий.
10. Заголовок вікна.

6.4 Вікна «Рахунки» і «Рахунок»

Користувачеві надається можливість переглядати рахунки. Приклад вікна «Рахунки» зображено на рисунку 6.6.

						Арк.
						46
Змн.	Арк.	№ Докум.	Підпис	Дата		

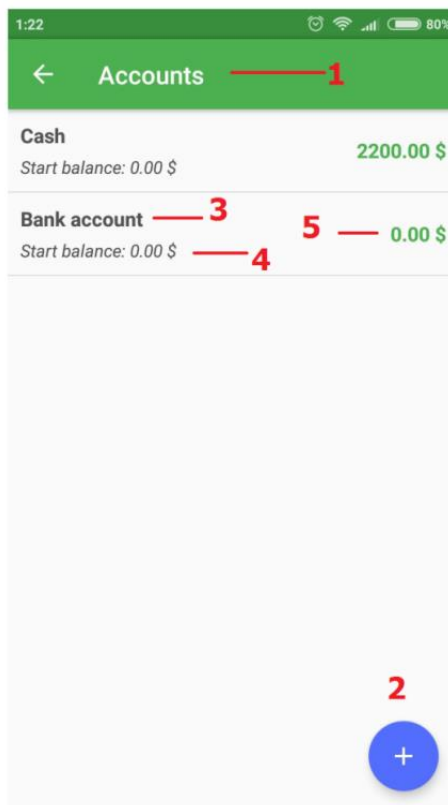


Рисунок 6.6 – Вікно додатки «Рахунки»

1. Заголовок вікна.
2. Кнопка для додавання нового рахунку.

Наведемо елементи списку рахунків:

3. Назва рахунку.
4. Початковий баланс рахунку.
5. Поточний баланс рахунку.

Вікно «Рахунок» має досить тривіальний вид: має поля введення назви платежу, його початкового балансу і кнопку збереження. Дане вікно має вигляд, зображений на рисунку 6.7.

						Арк.
						47
Змн.	Арк.	№ Докум.	Підпис	Дата		

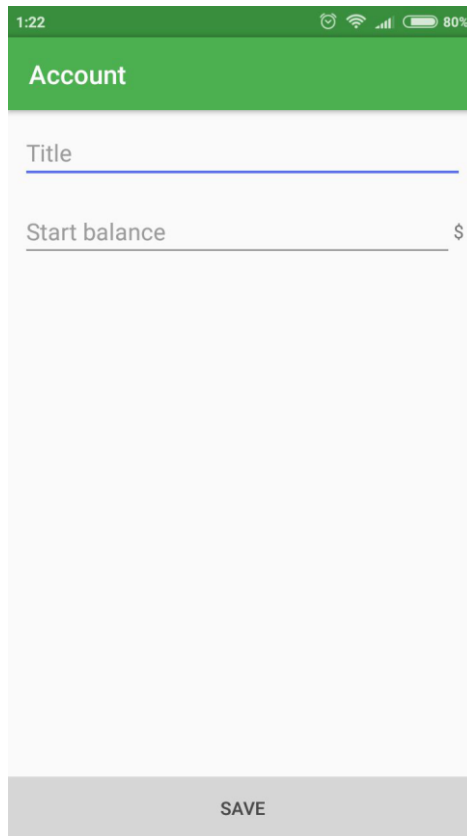


Рисунок 6.7 – Вікно додатки «Рахунок»

6.5 Вікно «Категорії»

Детально опишемо кожен елемент вікна категорії:

1. Заголовок вікна.
2. Тема посторінкового перегляду. Категорії можна переглядати для категорій витрат і поповнень окремо. Змінити сторінку перегляду можна або натиснувши на один із заголовків або слайдом по екрану в одну зі сторін.
3. Кнопка для додавання нової категорії.

Наведемо елементи списку категорій

4. Логотип категорії.
5. Назва категорії.

Вікно «Категорії» зображено на рисунку 6.8.

						Арк.
						48
Змн.	Арк.	№ Докум.	Підпис	Дата		

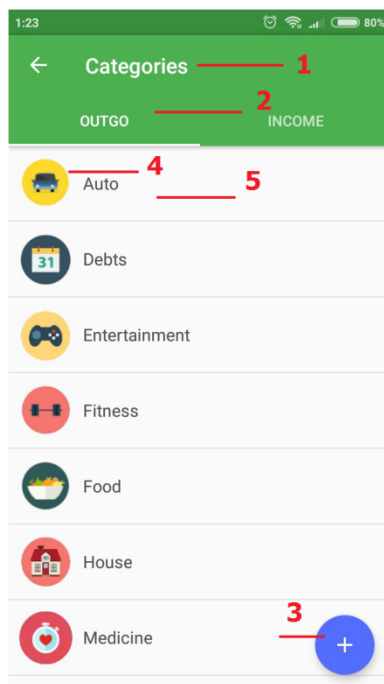


Рисунок 6.8 – вікно додатки «Категорії»

6.6 Вікно «Категорія»

Вікно категорії має наступні компоненти:

1. Заголовок вікна.
2. Список, що випадає для вибору логотипу категорії.
3. Поле для введення назви категорії.
4. Радіо-кнопки для вибору типу категорії (трата / поповнення).
5. Кнопка для збереження категорії.

Приклад роботи програми «Домашня бухгалтерія» у даному вікні можна побачити на рисунку 6.9.

						Арк.
						49
Змн.	Арк.	№ Докум.	Підпис	Дата		

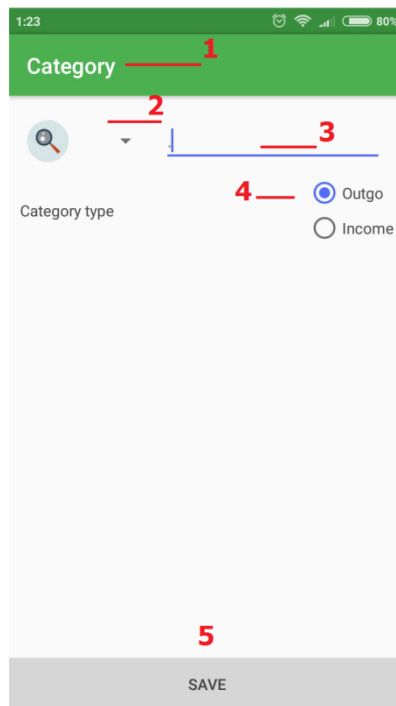


Рисунок 6.9 – Вікно додатки «Категорія»

6.7 Вікно «Кругова діаграма» і «Стовпчаста діаграма»

Дані вікна несуть в собі інформативний характер, тому головним їх елементом є сам графік. Також тут доступні функції для вибору періоду часу, аналогічно вікну «Журнал». Їх вигляд можна побачити на малюнках 6.10, 6.11.

						Арк.
						50
Змн.	Арк.	№ Докум.	Підпис	Дата		

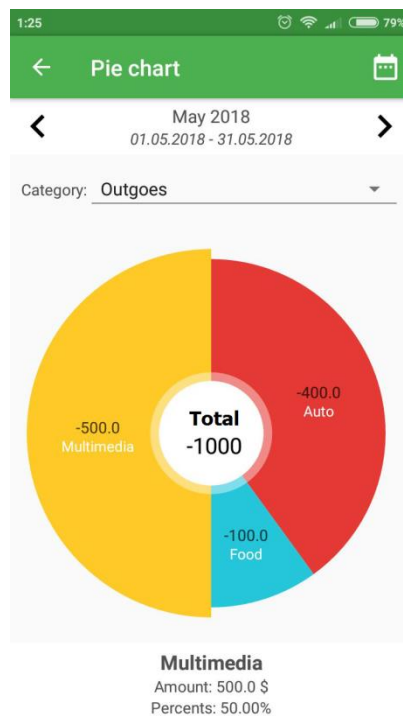


Рисунок 6.10 – Вікна додатку «Круговая діаграма»

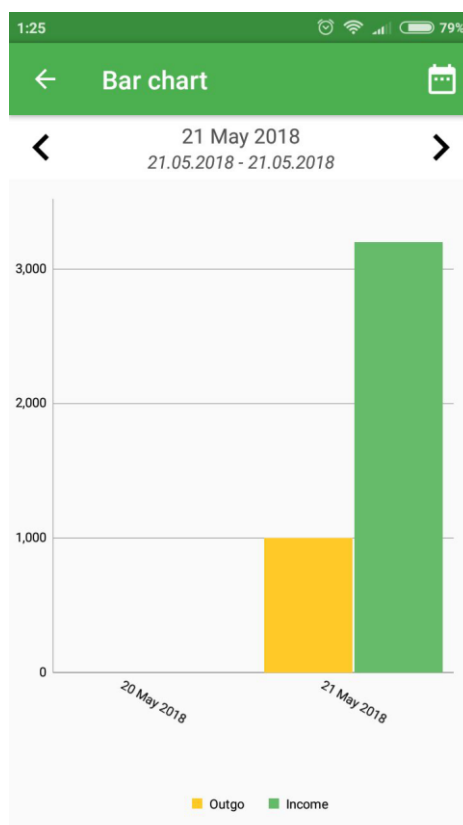


Рисунок 6.11 – Вікна додатку «Стовпчаста діаграма»

						Арк.
						51
Змн.	Арк.	№ Докум.	Підпис	Дата		

ВИСНОВОК

В ході виконання даної роботи було розроблено мобільний додаток, використовуючи яке користувач може вести власну домашню бухгалтерію. Для цього були реалізовані основні функції, такі як: робота з платежами (їх створення, зміна, видалення), робота з рахунками, робота з категоріям

Кожному ключовому процесу було виділено окреме вікно, що дозволяє користувачеві зосередитися на виконанні конкретного процесу, не відволікаючись на інші деталі.

Важливу в розробленому додатку роль, грає можливість перегляду графіку витрат і платежів. Представлені два види діаграм: кругова і стовпчата. Плюсом перегляду статистики в такому вигляді є те, що людині зручніше сприйняти візуальне представлення даних, а не сирі цифри.

Для реалізації цієї програми використовувалася мова програмування Java і система управління базами даних SQLite. Розмітка вікон програми проводилася за допомогою розширюваного мови розмітки XML. Використовувалася інтегроване середовище розробки Android Studio.

						Арк.
						52
Змн.	Арк.	№ Докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Эккель Б. Философия Java [Текст] / Б. Эккель. – СПб: Питер, 2009. – 640 с.
2. Шилдт Г. Java 8. Полное руководство [Текст] / Г. Шилдт. – Москва: Издательский дом «Вильямс», 2016. – 1376 с.
3. Дейтел П. Android для разработчиков [Текст] / П. Дейтел, Х. Дейтел. – СПб: Питер, 2016. – 512 с.
4. Харди Б. Программирование под Android [Текст] / Б. Харди, Б. Филлипс – СПб: Питер, 2014. – 592 с.
5. Черемных С.В Моделирование и анализ систем [Текст] / С.В. Черемных – Москва: Финансы и статистика, 2006. – 192 с.
6. Сайт Android [Электронный ресурс]. – Режим доступа: <https://www.android.com>
7. Сайт SQLite [Электронный ресурс]. – Режим доступа: <http://www.sqlite.org>
8. Сайт Android Developers [Электронный ресурс]. – Режим доступа: <https://developer.android.com>
9. Вендров А. М. CASE-технологии – современные методы и средства проектирования информационных систем / Вендров А. М. – М.: Финансы и статистика, 1998 – 171 с.
10. Кальянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Кальянов Г. Н. – М.: Лори, 1996. – 242 с.
11. Марка Д., Методология структурного анализа и проектирования / Д. Марка, К. МакГоуэн. – М.: МетаТехнология, 1993. – 240 с.
12. Черемных С. В. Структурный анализ систем: IDEF-технологии / Черемных С. В., Семенов И. О., Ручки В. С. – М.: Финансы и статистика, 2003. – 208 с.
13. Кватрани Терри. Визуальное моделирование с помощью Rational Rose 2002 и UML / Кватрани Т.; пер. с англ. – М.: Издательский дом „Вильямс”, 2003. – 192 с.

						Арк.
						53
Змн.	Арк.	№ Докум.	Підпис	Дата		

14. Чкалов А. П. Базы данных: от проектирования до разработки приложений / Чкалов А. П. – СПб.: БХВ-Петербург, 2003. – 384 с.
15. Кириллов В. В. Структуризованный язык запросов (SQL): учебн. пособ.: [Электронный ресурс] / В. В. Кириллов, Г. Ю. Громов. – СПб: Санкт-Петерб. госуд. техн. универ. , каф. обч. тех, 1998
16. Дейт К. Дж. Введение в системы баз данных / [Дейт К. Дж] 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с: ил.
17. Харрингтон Дж. Л. Проектирование реляционных баз данных/ [Харрингтон Дж. Л.] – М.: Издательство "Лори", 2006 – 232 с.
18. Сайт «Keepsoft HomeBand» [Электронный ресурс]. – Режим доступа: <https://www.keepsoft.ru/index.php>
19. Сайт «Личная бухгалтерия homebuh» [Электронный ресурс]. – Режим доступа: <http://homebuh.pro/>
20. Гутманс Э., Баккен С, Ретанс Д. РНР 5. Профессиональное программирование./ Пер. с англ. СПб: Символ- Плюс, 2006. 704 с., ил.
21. Матеріали офіційного сайту мови програмування Java [Електронний ресурс]. – Режим доступу: <https://www.java.com>.

						Арк.
						54
Змн.	Арк.	№ Докум.	Підпис	Дата		

ДОДАТОК А

Вихідний код програми

Файл AccountsActivity.java.

```
package com.application.homeaccountancy.Activity;
package com.application.homeaccountancy.Activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.Toolbar;
import android.text.Html;
import android.view.ContextMenu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.CursorAdapter;
import android.widget.TextView;

import com.application.homeaccountancy.Data.AccountancyContract;
import com.application.homeaccountancy.Data.Adapter.AccountCursorAdapter;
import com.application.homeaccountancy.R;

// Класс описывающий Activity просмотра счетов
public class AccountsActivity extends UsingDataBaseActivity {
    // ListView для вывода счетов
    private ListView accountsListView;

    // Адаптер для вывода элементов списка
    private SimpleCursorAdapter accountsCursorAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Установка контента
        setContentView(R.layout.accounts_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Нахождение ListView с Id accounts
        accountsListView = (ListView) findViewById(R.id.accounts);
        // Зарегистрировать событие создания контекстного меню
        // при долгом нажатии на элемент списка
        registerForContextMenu(accountsListView);

        // Кнопка для создания нового счета
        FloatingActionButton addNewAccount = (FloatingActionButton) findViewById(R.id.fab);
        addNewAccount.setOnClickListener(new View.OnClickListener() {
            @Override
```

						Арк.
						55
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        public void onClick(View view) {
            Intent intent = new Intent(getApplicationContext(), SingleAccountActivity.class);
            startActivity(intent);
        }
    });

    // Инициализировать адаптер
    initializeAdapter();
}

@Override
protected void onResume() {
    super.onResume();
    requeryCursor();
    accountsCursorAdapter.changeCursor(cursor);
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Обработка нажатия на элементе из контекстного меню

    // Id нажатого элемента
    final long id = ((AdapterView.AdapterContextMenuInfo)item.getMenuInfo()).id;

    switch (item.getItemId()) {
        // Если нажата кнопка "Изменить"
        // Вызвать Activity изменения счета и передать Id нажатого элемента
        case R.id.change:
            Intent intent = new Intent(getApplicationContext(), SingleAccountActivity.class);
            intent.putExtra("id", id);
            startActivity(intent);
            return true;
        case R.id.delete:
            // Если нажата кнопка "Удалить"

            // Создать диалоговое окно для подтверждения действия
            // и удалить элемент в случае необходимости
            AlertDialog.Builder dialog = new AlertDialog.Builder(AccountsActivity.this);
            dialog
                .setTitle("Confirm the action")
                .setMessage("Do you really want to delete an account? All relative transactions will be deleted.")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        db.delete(AccountancyContract.Account.TABLE_NAME,
                            AccountancyContract.Account._ID + "=?",
                            new String[] {String.valueOf(id)}
                        );

                        requeryCursor();
                        accountsCursorAdapter.changeCursor(cursor);
                    }
                })
            });
    }
}

```

						Арк.
						56
Змн.	Арк.	№ Докум.	Підпис	Дата		


```

        .setNegativeButton("No", null)
        .create();
        dialog.show();
        return true;
    }
    return super.onContextItemSelected(item);
}

private void initializeAdapter() {
    // Поля для вывода данных
    String[] from = new String[] {
        AccountancyContract.Account.A_TITLE,
        AccountancyContract.Account.START_BALANCE,
    };

    // Элементы, куда будут выводиться данные
    int[] to = new int[] {
        R.id.account_list_item_title,
        R.id.account_list_item_start,
    };

    // Обновление данных в курсоре
    requeryCursor();

    // Инициализация адаптера
    accountsCursorAdapter = new AccountCursorAdapter(getApplicationContext(),
        R.layout.account_list_item, cursor, from, to, 0);
    accountsListView.setAdapter(accountsCursorAdapter);
}

private void requeryCursor() {
    // Запрос на получение всех счетов
    String query = "SELECT " +
        AccountancyContract.Account.TABLE_NAME + "." + AccountancyContract.Account._ID +
        AccountancyContract.COMMA +
        AccountancyContract.Account.A_TITLE + AccountancyContract.COMMA +
        AccountancyContract.Account.START_BALANCE + AccountancyContract.COMMA +
        "SUM(" + AccountancyContract.Transaction.AMOUNT + ") as " +
        AccountancyContract.Transaction.AMOUNT + " FROM " +
        AccountancyContract.Account.TABLE_NAME +
        " LEFT JOIN " + AccountancyContract.Transaction.TABLE_NAME + " ON " +
        AccountancyContract.Account.TABLE_NAME + "." + AccountancyContract.Account._ID + "=" +
        AccountancyContract.Transaction.ACCOUNT_ID +
        " GROUP BY " +
        AccountancyContract.Account.TABLE_NAME + "." + AccountancyContract.Account._ID;

    cursor = db.rawQuery(query, null);
}
}

```

Файл BarChartActivity.java

```

package com.application.homeaccountancy.Activity;

import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import com.application.homeaccountancy.Data.AccountancyContract;

```

						Арк.
						57
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

import com.application.homeaccountancy.DateSelector;
import com.application.homeaccountancy.R;
import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.components.AxisBase;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.formatter.IAxisValueFormatter;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

// Класс Activity, содержащего столбчатую диаграмму
public class BarChartActivity extends UsingDataBaseActivity {
    private String fromJoinSelector;

    // Поле календаря, по которому производятся итерации
    private int changeFieldIteration;

    // Объект для генерации временных периодов
    private DateSelector dateSelector;

    // Элемент столбчатой диаграммы
    private BarChart barChart;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Инициализация контента
        setContentView(R.layout.bar_chart_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Инициализация объекта для генерации временных периодов
        dateSelector = new DateSelector(this);
        dateSelector.setOnDecreaseClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                decreaseDate(v);
            }
        });
        dateSelector.setOnIncreaseClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                increaseDate(v);
            }
        });

        changeFieldIteration = Calendar.DAY_OF_YEAR;

        // Инициализация представления
        initializeViews();

        // Инициализация графика
        initializeGraphics();
    }

```

						Арк.
						58
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Меню для выбора шага временного периода
        getMenuInflater().inflate(R.menu.date_selector_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Обработка нажатия по элементу в меню

        int id = item.getItemId();
        if(!item.isChecked())
            item.setChecked(true);

        boolean isChangeState = true;
        // В зависимости от нажатого элемента - изменить переменную
        // для итерации
        // Изменить временной диапазон
        switch(id){
            case R.id.range_day:
                dateSelector.setChangeFieldInterval(Calendar.DAY_OF_YEAR);
                changeFieldIteration = Calendar.DAY_OF_YEAR;
                break;
            case R.id.range_week:
                dateSelector.setChangeFieldInterval(Calendar.WEEK_OF_YEAR);
                changeFieldIteration = Calendar.DAY_OF_YEAR;
                break;
            case R.id.range_month:
                dateSelector.setChangeFieldInterval(Calendar.MONTH);
                changeFieldIteration = Calendar.DAY_OF_YEAR;
                break;
            case R.id.range_year:
                dateSelector.setChangeFieldInterval(Calendar.YEAR);
                changeFieldIteration = Calendar.MONTH;
                break;
            default:
                isChangeState = false;
        }
        if (isChangeState) {
            dateSelector.resetState();
            initializeGraphics();
        }

        return super.onOptionsItemSelected(item);
    }

    private void initializeViews() {
        // Инициализация представления

        // Инициализация элемента столбчатой диаграммы
        barChart = (BarChart) findViewById(R.id.chart);

        // Убрать описание
        barChart.getDescription().setEnabled(false);

        // Установка параметров отображения пустых данных
        barChart.setNoDataText("No data found to be displayed.");
        barChart.setNoDataTextColor(Color.BLACK);
    }

```

						Арк.
						59
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

// Установить минимум для оси y
barChart.getAxisLeft().setAxisMinimum(0);

// Убрать правую ось
barChart.getAxisRight().setEnabled(false);

// Настройка вида графика
barChart.getXAxis().setAxisMinimum(0);
barChart.getXAxis().setDrawGridLines(false);
barChart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);
barChart.getXAxis().setCenterAxisLabels(true);
barChart.getXAxis().setLabelRotationAngle(30);
barChart.getXAxis().setGranularityEnabled(true);

// Убрать подсвечивание при нажатии на элементе графика
barChart.setHighlightPerDragEnabled(false);
barChart.setHighlightPerTapEnabled(false);

// Настройка легенды
barChart.getLegend().setXEntrySpace(20);
barChart.getLegend().setHorizontalAlignment(Legend.LegendHorizontalAlignment.CENTER);

// Убрать масштабирование по y
barChart.getViewPortHandler().setMaximumScaleY(1);
}
private void initializeStrings() {
    // Инициализация строк запроса

    // Часть запроса, содержащую блок выбора таблицы
    // данных и соединения
    fromJoinSelector = " FROM " + AccountancyContract.Transaction.TABLE_NAME + " INNER JOIN " +
        AccountancyContract.Category.TABLE_NAME + " ON " + AccountancyContract.Category.TABLE_NAME +
        "." + AccountancyContract.Category._ID + " = " + AccountancyContract.Transaction.TABLE_NAME + "." +
        AccountancyContract.Transaction.CATEGORY_ID;
}
private void initializeGraphics() {
    // Инициализация строк запроса
    initializeStrings();

    // Данные трат и пополнения соответственно
    List<BarEntry> outgoing = new ArrayList<>();
    List<BarEntry> incoming = new ArrayList<>();

    // Инициализация начальной и конечной даты итераций
    Calendar currentCalendarFrom = (Calendar)dateSelector.getCalendarFrom().clone();
    Calendar currentCalendarTill = (Calendar)dateSelector.getCalendarFrom().clone();
    if (changeFieldIteration == Calendar.MONTH) {
        currentCalendarTill.set(Calendar.MONTH, currentCalendarFrom.get(Calendar.MONTH));
        currentCalendarTill.set(Calendar.DAY_OF_MONTH,
currentCalendarTill.getActualMaximum(Calendar.DAY_OF_MONTH));
    }

    int index = 0;
    while (currentCalendarTill.compareTo(dateSelector.getCalendarTill()) <= 0) {
        // Инициализация строк временного диапазона
        // для которого будут вычисляться данные на текущей итерации
        String fromDate = String.format("%tY-%tm-%td 00:00", currentCalendarFrom, currentCalendarFrom,
currentCalendarFrom);
        String tillDate = String.format("%tY-%tm-%td 23:59", currentCalendarTill, currentCalendarTill,
currentCalendarTill);

```

						Арк.
						60
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

// Запрос на получение суммы платежей за определенный период времени
// Для трат или пополнений
String query = "SELECT SUM (" +
    AccountancyContract.Transaction.AMOUNT + ") " +
    fromJoinSelector + " WHERE " + AccountancyContract.Transaction.DATE + " >= " +
    fromDate + " AND " + AccountancyContract.Transaction.DATE + " <= " +
    tillDate + "" + " AND " + AccountancyContract.Category.IS_OUTGO +
    " = %d" ;

// Нахождение суммы трат
cursor = db.rawQuery(String.format(query, 1), null);
cursor.moveToFirst();
outgoes.add(new BarEntry(index, Math.abs(cursor.getInt(0))));

// Нахождение суммы пополнений
cursor = db.rawQuery(String.format(query, 0), null);
cursor.moveToFirst();
incomes.add(new BarEntry(index, Math.abs(cursor.getInt(0))));

// Переход к следующему временному отрезку
index++;
if (changeFieldIteration == Calendar.DAY_OF_YEAR) {
    currentCalendarFrom.set(Calendar.DAY_OF_YEAR, currentCalendarFrom.get(changeFieldIteration) + 1);
    currentCalendarTill.set(Calendar.DAY_OF_YEAR, currentCalendarTill.get(changeFieldIteration) + 1);
}
else {
    currentCalendarFrom.set(Calendar.DAY_OF_MONTH,
currentCalendarFrom.getActualMinimum(Calendar.DAY_OF_MONTH));
    currentCalendarFrom.set(Calendar.MONTH, currentCalendarFrom.get(Calendar.MONTH) + 1);

    currentCalendarTill.set(Calendar.DAY_OF_MONTH,
currentCalendarTill.getActualMinimum(Calendar.DAY_OF_MONTH));
    currentCalendarTill.set(Calendar.MONTH, currentCalendarTill.get(Calendar.MONTH) + 1);
    currentCalendarTill.set(Calendar.DAY_OF_MONTH,
currentCalendarTill.getActualMaximum(Calendar.DAY_OF_MONTH));
}

    cursor.close();
}

// Установка данных графика
BarDataSet barDataSetOutgoes = new BarDataSet(outgoes, "Outgo");
barDataSetOutgoes.setColor(Color.parseColor("#FFCA28"));
BarDataSet barDataSetIncomes = new BarDataSet(incomes, "Income");
barDataSetIncomes.setColor(Color.parseColor("#66BB6A"));

float groupSpace = 0.1f;
float barSpace = 0.01f;
float barWidth = 0.44f;

BarData data = new BarData(barDataSetOutgoes, barDataSetIncomes);
data.setDrawValues(false);
data.setBarWidth(barWidth);

barChart.setData(data);
barChart.groupBars(0, groupSpace, barSpace);
barChart.getXAxis().setAxisMaximum(index);

// Формат подписи осей графика

```

						Арк.
						61
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        if (changeFieldIteration == Calendar.DAY_OF_YEAR)
            barChart.getXAxis().setValueFormatter(new DaysValueFormatter(dateSelector.getCalendarFrom()));
        else
            barChart.getXAxis().setValueFormatter(new MonthValueFormatter(dateSelector.getCalendarFrom()));

        barChart.invalidate();
    }

    private void decreaseDate(View view) {
        // Уменьшение временного периода
        dateSelector.dateChange(-1);
        initializeGraphics();
    }
    private void increaseDate(View view) {
        // Увеличение временного периода
        dateSelector.dateChange(1);
        initializeGraphics();
    }

    // Форматирование дней месяца
    private class DaysValueFormatter implements IAxisValueFormatter {
        String[] month = new String[] { "Jan.", "Feb.", "Mar.", "Apr.", "May",
            "Jun.", "Jul.", "Aug.", "Sep.", "Oct.", "Nov.", "Dec." };

        Calendar startDate;
        DaysValueFormatter(Calendar calendar) {
            startDate = (Calendar)calendar.clone();
        }

        @Override
        public String getFormattedValue(float value, AxisBase axis) {
            Calendar currentCalendar = (Calendar) startDate.clone();
            currentCalendar.set(Calendar.DAY_OF_YEAR, currentCalendar.get(Calendar.DAY_OF_YEAR) + (int)value);

            return String.format("%d %s %d",
                currentCalendar.get(Calendar.DAY_OF_MONTH),
                month[currentCalendar.get(Calendar.MONTH)],
                currentCalendar.get(Calendar.YEAR));
        }
    }

    // Форматирование месяцев года
    private class MonthValueFormatter implements IAxisValueFormatter {
        String[] month = new String[] { "January", "February", "March", "April", "May",
            "June", "July", "August", "September", "October", "November", "December" };

        Calendar startDate;
        MonthValueFormatter(Calendar calendar) {
            startDate = (Calendar)calendar.clone();
        }

        @Override
        public String getFormattedValue(float value, AxisBase axis) {
            Calendar currentCalendar = (Calendar) startDate.clone();
            currentCalendar.set(Calendar.MONTH, currentCalendar.get(Calendar.MONTH) + (int)value);

            return String.format("%s %d",
                month[currentCalendar.get(Calendar.MONTH)],
                currentCalendar.get(Calendar.YEAR));
        }
    }
}

```

```
}
```

Файл CategoriesActivity.java

```
package com.application.homeaccountancy.Activity;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;

import com.application.homeaccountancy.Data.AccountancyContract;
import com.application.homeaccountancy.Fragment.FragmentCategories;
import com.application.homeaccountancy.R;

// Класс описывающий Activity просмотра ктегорий
public class CategoriesActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Инициализация контента
        setContentView(R.layout.categories_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Инициализация адаптера постраничного просмотра
        SectionsPagerAdapter mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());

        // Кнопка добавления новой категории
        FloatingActionButton addNewCategory = (FloatingActionButton) findViewById(R.id.add_new_category);
        addNewCategory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), SingleCategoryActivity.class);
                startActivity(intent);
            }
        });

        // Элемент постраничного просмотра
        ViewPager mViewPager = (ViewPager) findViewById(R.id.container);
        mViewPager.setAdapter(mSectionsPagerAdapter);

        TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(mViewPager);
    }

    // Адаптер для получения фрагментов страниц
    private class SectionsPagerAdapter extends FragmentPagerAdapter {

        SectionsPagerAdapter(FragmentManager fm) {
```

						Арк.
						63
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        // Генерация фрагментов, в котором определены данные
        // отдельно для категорий трат и пополнений
        FragmentCategories fragmentCategories = null;

        switch (position) {
            case 0:
                fragmentCategories = FragmentCategories.FragmentCategoriesFactory(
                    FragmentCategories.getBaseQuery() +
                        " WHERE " + AccountancyContract.Category.IS_OUTGO + " = 1"
                );
                break;
            case 1:
                fragmentCategories = FragmentCategories.FragmentCategoriesFactory(
                    FragmentCategories.getBaseQuery() +
                        " WHERE " + AccountancyContract.Category.IS_OUTGO + " = 0"
                );
                break;
            default:
                fragmentCategories = FragmentCategories.FragmentCategoriesFactory(
                    FragmentCategories.getBaseQuery()
                );
        }
        return fragmentCategories;
    }

    @Override
    public int getCount() {
        return 2;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        switch (position) {
            case 0:
                return "Outgo";
            case 1:
                return "Income";
        }
        return null;
    }
}

```

Файл MainActivity.java

```

package com.application.homeaccountancy.Activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.NavigationView;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;

```

						Арк.
						64
Змн.	Арк.	№ Докум.	Підпис	Дата		


```

import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.GravityCompat;
import android.support.v4.view.ViewPager;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.application.homeaccountancy.DateSelector;
import com.application.homeaccountancy.Fragment.FragmentTransactions;
import com.application.homeaccountancy.R;
import com.application.homeaccountancy.SMSParser.SMSParser;
import com.application.homeaccountancy.Utilities;

public class MainActivity extends UsingDataBaseActivity
    implements NavigationView.OnNavigationItemSelectedListener {

    // Объект для генерации временных периодов
    DateSelector dateSelector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Инициализация контента
        setContentView(R.layout.main_activity);

        // Инициализация тулраба
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        setTitle("Personal journal");

        // Инициализация объекта для генерации временных периодов
        dateSelector = new DateSelector(this);

        // Чтение данных из смс
        //initializeDataFromSMS();

        // Инициализация бокового меню
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        // Инициализация адаптера страничного просмотра
        SectionsPagerAdapter mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());

        // Элемент страничного просмотра
        final ViewPager mViewPager = (ViewPager) findViewById(R.id.container);
        mViewPager.setAdapter(mSectionsPagerAdapter);
        mViewPager.setOffscreenPageLimit(2);

        // Кнопка для добавления нового платежа
        FloatingActionButton floatingActionButton = (FloatingActionButton)findViewById(R.id.add_new_transaction);

```

						Арк.
						65
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

floatingActionButton.setOnClickListener(new View.OnClickListener() {
    // В зависимости от страницы просмотра журнала
    // При добавлении новой записи определить категорию платежа
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), SingleTransactionActivity.class);
        final int[] category = new int[1];
        switch (mViewPager.getCurrentItem()) {
            case 1:
                intent.putExtra("category", 1);
                startActivity(intent);
                break;
            case 2:
                intent.putExtra("category", 0);
                startActivity(intent);
                break;
            case 0:
            default:
                getCategoryTypeFromDialog(intent).show();
        }
    }
});

TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
tabLayout.setupWithViewPager(mViewPager);
}

@Override
public void onBackPressed() {
    // Обработка нажатия на кнопку "назад"
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();

    // Обработка нажатия на элемент бокового меню
    // Вызвать соответствующее Activity
    if (id == R.id.menu_categories) {
        Intent intent = new Intent(getApplicationContext(), CategoriesActivity.class);
        startActivity(intent);
    }
    else if (id == R.id.menu_pie_graphic) {
        Intent intent = new Intent(getApplicationContext(), PieCharActivity.class);
        startActivity(intent);
    }
    else if (id == R.id.menu_bar_graphic) {
        Intent intent = new Intent(getApplicationContext(), BarChartActivity.class);
        startActivity(intent);
    }
    else if (id == R.id.menu_accounts) {
        Intent intent = new Intent(getApplicationContext(), AccountsActivity.class);
        startActivity(intent);
    }
}

```

						Арк.
						66
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
drawer.closeDrawer(GravityCompat.START);
return true;
}

private void initializeDataFromSMS() {
    // Чтение всех смс
    Cursor cursor = getContentResolver().query(Uri.parse("content://sms/inbox"),
        null, null, null, null);

    SMSParser smsParser = new SMSParser(getApplicationContext(), db);
    int count = 0;
    if (cursor != null && cursor.moveToFirst()) {
        do {
            // Получение текста, даты и id смс
            String smsMessage = cursor.getString(cursor.getColumnIndex("body"));
            long time = cursor.getLong(cursor.getColumnIndex("date"));

            // Обработка смс
            if (smsParser.handleSMS(smsMessage.trim(), time))
                count++;
        } while (cursor.moveToNext());
        cursor.close();

        if (count > 0)
            Utilities.makeToast(getApplicationContext(), String.format("%d transactions were added from SMS", count));
    }
}

public DateSelector getDateSelector() {
    return dateSelector;
}

// Адаптер для получения фрагментов страниц
private class SectionsPagerAdapter extends FragmentStatePagerAdapter {

    SectionsPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        FragmentTransactions fragmentTransaction;

        // Генерация фрагментов, в котором определены данные
        // отдельно для всех платежей, трат и пополнений
        switch (position) {
            case 0:
                fragmentTransaction = FragmentTransactions.FragmentTransactionsFactory(
                    FragmentTransactions.getBaseQuery()
                );
                break;
            case 1:
                fragmentTransaction = FragmentTransactions.FragmentTransactionsFactory(
                    FragmentTransactions.getBaseQuery() + " WHERE is_outgo = 1"
                );
                break;
            case 2:
                fragmentTransaction = FragmentTransactions.FragmentTransactionsFactory(
                    FragmentTransactions.getBaseQuery() + " WHERE is_outgo = 0"
                );
        }
    }
}

```

						Арк.
						67
Змн.	Арк.	№ Докум.	Підпис	Дата		


```

import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;
import com.github.mikephil.charting.formatter.IValueFormatter;
import com.github.mikephil.charting.highlight.Highlight;
import com.github.mikephil.charting.listener.OnChartValueSelectedListener;
import com.github.mikephil.charting.utils.ViewPortHandler;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

// Класс Activity, содержащего круговую диаграмму
public class PieCharActivity extends UsingDataBaseActivity {
    // region final int[] colors
    static final int[] colors = new int[] { Color.parseColor("#e53935"), Color.parseColor("#26C6DA"),
    Color.parseColor("#FFCA28"),
    Color.parseColor("#EC407A"), Color.parseColor("#26A69A"), Color.parseColor("#FFA726"),
    Color.parseColor("#AB47BC"), Color.parseColor("#66BB6A"), Color.parseColor("#FF7043"),
    Color.parseColor("#7E57C2"), Color.parseColor("#9CCC65"), Color.parseColor("#8D6E63"),
    Color.parseColor("#5C6BC0"), Color.parseColor("#D4E157"), Color.parseColor("#BDBDBD"),
    Color.parseColor("#42A5F5"), Color.parseColor("#FFEE58"), Color.parseColor("#78909C"),
    Color.parseColor("#29B6F6")};
    // endregion

    // Строки запроса
    private String fromJoinSelector, whereSelector;

    // Переменные представления
    private TextView categoryTextView, totalTextView, percentTextView;
    private Spinner categoriesSpinner;

    // Объект для генерации временных периодов
    private DateSelector dateSelector;

    // Элемент круговой диаграммы
    private PieChart pieChart;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Инициализация контента
        setContentView(R.layout.pie_char_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Инициализация объекта для генерации временных периодов
        dateSelector = new DateSelector(this);
        dateSelector.setOnDecreaseClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                decreaseDate(v);
            }
        });
        dateSelector.setOnIncreaseClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

```

						Арк.
						69
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        increaseDate(v);
    }
});

// Инициализация переменных представления
initializeViews();

// Инициализация слушателей событий
initializeListeners();

// Инициализация диаграммы
initializeGraphics();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.date_selector_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Обработка нажатия по элементу в меню

    int id = item.getItemId();
    if(!item.isChecked())
        item.setChecked(true);

    boolean isChangeState = true;
    // В зависимости от нажатого элемента - изменить переменную
    // для итерации
    // Изменить временной диапазон
    switch(id){
        case R.id.range_day:
            dateSelector.setChangeFieldInterval(Calendar.DAY_OF_YEAR);
            break;
        case R.id.range_week:
            dateSelector.setChangeFieldInterval(Calendar.WEEK_OF_YEAR);
            break;
        case R.id.range_month:
            dateSelector.setChangeFieldInterval(Calendar.MONTH);
            break;
        case R.id.range_year:
            dateSelector.setChangeFieldInterval(Calendar.YEAR);
            break;
        default:
            isChangeState = false;
    }
    if (isChangeState) {
        dateSelector.resetState();
        initializeGraphics();
    }
    return super.onOptionsItemSelected(item);
}

private void initializeViews() {
    // Инициализация элемента диаграммы
    pieChart = (PieChart) findViewById(R.id.chart);
    pieChart.getLegend().setEnabled(false);;
    pieChart.setCenterTextSize(20);
    pieChart.setHoleRadius(30);
}

```

```

pieChart.getContentDescription();
pieChart.getDescription().setEnabled(false);
pieChart.setTransparentCircleRadius(35);
pieChart.setNoDataText("No data found to be displayed.");
pieChart.setNoDataTextColor(Color.BLACK);

// Инициализация переменных представления
categoryTextView = (TextView) findViewById(R.id.category);
totalTextView = (TextView) findViewById(R.id.total);
percentTextView = (TextView) findViewById(R.id.percent);
categoriesSpinner = (Spinner) findViewById(R.id.spinner_category_type);
}
private void initializeGraphics() {
    int totalSum;
    final int totalSumEx;

    // Инициализация строк запроса
    initializeStrings();

    // Лист для хранения данных
    List<PieEntry> entries = new ArrayList<>();

    // Запрос для получение суммы трат или пополнений
    // За определенный период времени
    String query = "SELECT SUM(" + AccountancyContract.Transaction.AMOUNT + ") " +
        fromJoinSelector + whereSelector;

    cursor = db.rawQuery(query, null);
    if (cursor.moveToFirst())
        totalSum = cursor.getInt(0);
    else return;

    // Запрос для получения суммы трат/пополнений
    // Разделенный по категориям
    query = "SELECT " + AccountancyContract.Category.C_TITLE +
        AccountancyContract.COMMA + " SUM (" +
        AccountancyContract.Transaction.AMOUNT + ") " +
        fromJoinSelector + whereSelector +
        " GROUP BY " + AccountancyContract.Transaction.CATEGORY_ID;

    cursor = db.rawQuery(query, null);

    // Считывание данных
    int currentSum;
    if (cursor.moveToFirst()) {
        do {
            currentSum = cursor.getInt(1);
            entries.add(new PieEntry(Math.abs(currentSum), cursor.getString(0)));
        }
        while (cursor.moveToNext());
    }
    cursor.close();

    // Инициализация графика
    PieDataSet pieDataSet = new PieDataSet(entries, "Entries");
    pieDataSet.setValueTextColor(Color.argb(180, 0, 0, 0));
    pieDataSet.setValueTextSize(14);
    pieDataSet.setColors(colors);

    if (totalSum < 0)
        pieDataSet.setValueFormatter(new NegativeValueFormatter());

```

						Арк.
						71
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

pieDataSet.setDrawValues(true);
pieDataSet.setHighlightEnabled(true);

PieData pieData = new PieData(pieDataSet);
pieChart.setData(pieData);
pieChart.setCenterText("Total:\n " + String.valueOf(totalSum));

totalSumEx = totalSum;
// Вывод подробной информации при нажатии на элементе диаграммы
pieChart.setOnChartValueSelectedListener(new OnChartValueSelectedListener() {
    @Override
    public void onValueSelected(Entry e, Highlight h) {
        categoryTextView.setText(((PieEntry)e).getLabel());
        totalTextView.setText("Amount: " + String.valueOf(e.getY()) + " $");
        percentTextView.setText(String.format("Percents: %.2f%%", Math.abs((e.getY() * 100) / totalSumEx)));
    }

    @Override
    public void onNothingSelected() {
        categoryTextView.setText("Nothing selected");
        totalTextView.setText("Amount: not defined");
        percentTextView.setText("Percents: not defined");
    }
});
pieChart.invalidate();
}
private void initializeStrings() {
    int isOutgo = categoriesSpinner.getSelectedItemPosition() == 0 ? 1 : 0;

    // Инициализация строк запроса
    String fromDate = dateSelector.getFromString();
    String tillDate = dateSelector.getTillString();

    fromJoinSelector = " FROM " + AccountancyContract.Transaction.TABLE_NAME + " INNER JOIN " +
        AccountancyContract.Category.TABLE_NAME + " ON " + AccountancyContract.Category.TABLE_NAME +
        "." + AccountancyContract.Category._ID + " = " + AccountancyContract.Transaction.TABLE_NAME + "." +
        AccountancyContract.Transaction.CATEGORY_ID;

    whereSelector = " WHERE " + AccountancyContract.Transaction.DATE + " >= " +
        fromDate + " AND " + AccountancyContract.Transaction.DATE + " <= " +
        tillDate + " AND " + AccountancyContract.Category.IS_OUTGO + " = " + isOutgo ;
}
private void initializeListeners() {
    categoriesSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            initializeGraphics();
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    });
}

public void decreaseDate(View view) {
    // Уменьшение временного периода
    dateSelector.dateChange(-1);
    initializeGraphics();
}
public void increaseDate(View view) {

```



```

        // Увеличение временного периода
        dateSelector.dateChange(1);
        initializeGraphics();
    }

    // Форматирование отрицательных значений
    private class NegativeValueFormatter implements IValueFormatter {
        @Override
        public String getFormattedValue(float value, Entry entry, int dataSetIndex, ViewPortHandler viewPortHandler) {
            return String.format("-%.1f", value);
        }
    }
}

```

Файл SingleAccountActivity.java

```
package com.application.homeaccountancy.Activity;
```

```

import android.content.ContentValues;
import android.content.DialogInterface;
import android.database.sqlite.SQLiteConstraintException;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

```

```

import com.application.homeaccountancy.Data.AccountancyContract;
import com.application.homeaccountancy.R;
import com.application.homeaccountancy.Utilities;

```

```

// Класс, представляющий Счет
public class SingleAccountActivity extends SingleEntityActivity {
    // Переменные представления
    TextView titleTextView, startBalanceTextView;

```

```

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```

        // Инициализация контента
        setContentView(R.layout.single_account_activity);

```

```

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

```

```
        initializeViews();
```

```

        // Если данное Activity было вызвано для изменения
        // созданного счёта, то заполнить поля данными

```

```

        if (isEntityId()) {
            cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Account.TABLE_NAME +
                " WHERE " + AccountancyContract.Account._ID + "=?", new String[] {String.valueOf(getEntityId())});

```

```

            cursor.moveToFirst();
            titleTextView.setText(cursor.getString(cursor.getColumnIndex(AccountancyContract.Account.A_TITLE)));

```

```

        startBalanceTextView.setText(cursor.getString(cursor.getColumnIndex(AccountancyContract.Account.START_BALANCE)));
    }
}

```

						Арк.
						73
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Обработка нажатия на элементе из меню
    int id = item.getItemId();
    if(!item.isChecked())
        item.setChecked(true);

    if (id == R.id.delete) {
        // Если нажат элемент "Удалить"

        // Создать диалоговое окно для подтверждения действия
        // и удалить элемент в случае необходимости
        AlertDialog.Builder dialog = new AlertDialog.Builder(SingleAccountActivity.this);
        dialog
            .setTitle("Confirm the action")
            .setMessage("Do you really want to delete account? All relative transaction will be deleted.")
            .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    db.delete(AccountancyContract.Account.TABLE_NAME,
                        AccountancyContract.Account._ID + "=?",
                        new String[] {String.valueOf(getEntityId())}
                    );
                    finish();
                }
            })
            .setNegativeButton("No", null)
            .create();
        dialog.show();
    }
    return true;
}

private void initializeViews() {
    // Инициализировать переменные представления
    titleTextView = (TextView) findViewById(R.id.account_title);
    startBalanceTextView = (TextView) findViewById(R.id.start_balance);
}

public void saveAccount(View view) {
    String title = titleTextView.getText().toString();
    String startBalanceText = startBalanceTextView.getText().toString();

    // Валидация данных
    if (title.isEmpty()){
        Utilities.makeToast(this, "Account name can not be empty");
        return;
    }

    if (startBalanceText.isEmpty()){
        Utilities.makeToast(this, "Start balance is required.");
        return;
    }

    // Если элемент изменялся - обновить его в базе данных
    // Иначе - создать новый
    double startBalance = Double.parseDouble(startBalanceText);
    try {
        ContentValues contentValues = new ContentValues();
        contentValues.put(AccountancyContract.Account.A_TITLE, title);

```

```

        contentValues.put(AccountancyContract.Account.START_BALANCE, startBalance);

        if (isEntityId()) {
            db.update(AccountancyContract.Account.TABLE_NAME, contentValues,
                AccountancyContract.Account._ID + "=?", new String[] {String.valueOf(getEntityId())});
            finish();
        }
        else {
            db.insertOrThrow(AccountancyContract.Account.TABLE_NAME, null, contentValues);
            finish();
        }
    }
    catch (SQLiteConstraintException exception) {
        Utilities.makeToast(this, "Account with the same name already exists.");
    }
}
}

```

Файл SingleCategoryActivity.java

```

package com.application.homeaccountancy.Activity;

import android.content.ContentValues;
import android.content.DialogInterface;
import android.database.sqlite.SQLiteConstraintException;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.SimpleCursorAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.application.homeaccountancy.Data.AccountancyContract;
import com.application.homeaccountancy.R;
import com.application.homeaccountancy.Utilities;

// Класс, представляющий Категория
public class SingleCategoryActivity extends SingleEntityActivity {
    // Переменные представления
    private TextView titleTextView;
    private RadioButton incomeCategoryRadioButton, outgoingCategoryRadioButton;
    private Spinner logoSpinner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Инициализация контента
        setContentView(R.layout.single_category_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        initializeViews();

        // Если данное Activity было вызвано для изменения

```

						Арк.
						75
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

// созданной категории, то заполнить поля данными
if (isEntityId()) {
    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Category.TABLE_NAME +
        " WHERE " + AccountancyContract.Category._ID + "=?", new String[] {String.valueOf(getEntityId())});

    cursor.moveToFirst();
    titleTextView.setText(cursor.getString(cursor.getColumnIndex(AccountancyContract.Category.C_TITLE)));

    if (cursor.getInt(cursor.getColumnIndex(AccountancyContract.Category.IS_OUTGO)) != 0)
        outgoCategoryRadioButton.setChecked(true);
    else
        incomeCategoryRadioButton.setChecked(true);
}
initializeSpinners();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Обработка нажатия на элементе из меню
    int id = item.getItemId();
    if(!item.isChecked())
        item.setChecked(true);

    if (id == R.id.delete) {
        // Если нажат элемент "Удалить"

        // Создать диалоговое окно для подтверждения действия
        // и удалить элемент в случае необходимости
        AlertDialog.Builder dialog = new AlertDialog.Builder(SingleCategoryActivity.this);
        dialog
            .setTitle("Подтверждение действия")
            .setMessage("Вы действительно хотите удалить категорию? Все записи данной " +
                "категории будут стерты.")
            .setPositiveButton("Да", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    db.delete(AccountancyContract.Category.TABLE_NAME,
                        AccountancyContract.Category._ID + "=?",
                        new String[] {String.valueOf(getEntityId())}
                    );
                    finish();
                }
            })
            .setNegativeButton("Нет", null)
            .create();
        dialog.show();
    }
    return true;
}

private void initializeViews() {
    // Инициализировать переменные представления
    titleTextView = (TextView) findViewById(R.id.category_title);
    incomeCategoryRadioButton = (RadioButton) findViewById(R.id.category_income_rb);
    outgoCategoryRadioButton = (RadioButton) findViewById(R.id.category_outgo_rb);
    logoSpinner = (Spinner) findViewById(R.id.logo_spinner);
}
private void initializeSpinners() {
    SimpleCursorAdapter logoAdapter;

    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Images.TABLE_NAME, null);

```

						Арк.
						76
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

logoAdapter = new SimpleCursorAdapter(this, R.layout.spinner_logo_item, cursor,
    new String[] { AccountancyContract.Images.COLUMN_NAME_IMAGE }, new int[] { R.id.logo }, 0);
logoSpinner.setAdapter(logoAdapter);

if (!isEntityId())
    return;

cursor = db.rawQuery("SELECT " + AccountancyContract.Images.TABLE_NAME + "." +
AccountancyContract.Images._ID + ", " +
    AccountancyContract.Category.ICON + " as 'icon_value'" + " FROM " +
    AccountancyContract.Images.TABLE_NAME + " LEFT JOIN " +
    AccountancyContract.Category.TABLE_NAME + " ON " +
    "icon_value=" +
    AccountancyContract.Images.COLUMN_NAME_IMAGE +
    " WHERE " + AccountancyContract.Category.TABLE_NAME + "." +
    AccountancyContract.Category._ID + "=" +
    String.valueOf(getEntityId()), null);

cursor.moveToFirst();
long iconId = cursor.getLong(
    cursor.getColumnIndex(AccountancyContract.Images.TABLE_NAME + "." +
AccountancyContract.Images._ID));

Utilities.selectSpinnerItem(logoAdapter, logoSpinner, iconId);
}

public void saveCategory(View view) {
    String title = titleTextView.getText().toString();

    // Валидация данных
    if (title.isEmpty()){
        Toast toast = Toast.makeText(this, "Название категории не может быть пустым", Toast.LENGTH_LONG);
        toast.show();
        return;
    }

    // Если элемент изменялся - обновить его в базе данных
    // Иначе - создать новый
    try {
        cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Images.TABLE_NAME +
            " WHERE " + AccountancyContract.Images._ID + "=?",
            new String[] { String.valueOf(logoSpinner.getSelectedItemId())});
        cursor.moveToFirst();

        ContentValues contentValues = new ContentValues();
        contentValues.put(AccountancyContract.Category.C_TITLE, title);
        contentValues.put(AccountancyContract.Category.IS_OUTGO,
            outgoCategoryRadioButton.isChecked() ? 1 : 0);
        contentValues.put(AccountancyContract.Category.ICON,
            cursor.getInt(cursor.getColumnIndex(AccountancyContract.Images.COLUMN_NAME_IMAGE)));

        if (isEntityId()) {
            db.update(AccountancyContract.Category.TABLE_NAME, contentValues,
                AccountancyContract.Category._ID + "=?", new String[] {String.valueOf(getEntityId())});
        }
        else {
            db.insertOrThrow(AccountancyContract.Category.TABLE_NAME, null, contentValues);
        }
        finish();
    }
}

```

						Арк.
						77
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        catch (SQLiteConstraintException excerion) {
            Toast toast = Toast.makeText(this, "Категория с таким именем уже существует", Toast.LENGTH_LONG);
            toast.show();
        }
        catch (Exception ignored) {}
    }
}

```

Файл SingleEntityActivity.java

```

package com.application.homeaccountancy.Activity;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import com.application.homeaccountancy.R;

// Класс, описывающий сущность (например: счет, категория)
public abstract class SingleEntityActivity extends UsingDataBaseActivity{
    // Id элемента в базе данных
    private long entityId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Попытка считать элемент Id
        // если сущность была ранее создана
        Bundle extras = getIntent().getExtras();
        if (extras != null && extras.containsKey("id")) {
            entityId = extras.getLong("id");
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Создать меню удаления, если сущность была ранее создана
        if (isEntityId())
            getMenuInflater().inflate(R.menu.dalete_menu, menu);

        return true;
    }

    @Override
    public abstract boolean onOptionsItemSelected(MenuItem item);

    public long getEntityId() {
        return entityId;
    }

    public boolean isEntityId() {
        return entityId > 0;
    }
}

```

Файл SingleTransactionActivity.java

```

package com.application.homeaccountancy.Activity;

```

						Арк.
						78
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.Toolbar;
import android.text.InputType;
import android.view.MenuItem;
import android.view.View;
import android.widget.SimpleCursorAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.TimePicker;

import com.application.homeaccountancy.Data.AccountancyContract;
import com.application.homeaccountancy.R;
import com.application.homeaccountancy.Utilities;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

// Класс, представляющий Платеж
public class SingleTransactionActivity extends SingleEntityActivity {
    // Переменные представления
    private TextView currentDateTextView, currentTimeTextView;
    private Spinner categoriesSpinner, accountsSpinner;
    private EditText transactionAmountEditText, noteEditTextEditText;
    private Button signButton;

    // Слушатели выбора даты и времени
    private DatePickerDialog.OnDateSetListener onDateSetListener;
    private TimePickerDialog.OnTimeSetListener onTimeSetListener;

    // Время платежа
    private Calendar dateTime;

    // Отрицательная сумма
    private boolean isNegativeAmount;

    // ID категории и счета платежа
    private long categoryID, accountID;

    // Категория платежа (трата/пополнение)
    private int categoryType;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Инициализация контента
        setContentView(R.layout.single_transaction_activity);

        // Инициализация тулбара
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

```

						Арк.
						79
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

// Инициализация элементов представлений и слушателей
initializeViews();
initializeListeners();

dateTime = Calendar.getInstance();
isNegativeAmount = true;

// Получение типа платежа
categoryType = getCategoryType();
if (categoryType == 0)
    signButton.callOnClick();

// Если данное Activity было вызвано для изменения
// созданной категории, то заполнить поля данными
if (isEntityId()) {
    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Transaction.TABLE_NAME +
        " WHERE " + AccountancyContract.Transaction._ID + "=?", new String[] {String.valueOf(getEntityId())});
    cursor.moveToFirst();

    String date = cursor.getString(cursor.getColumnIndex(AccountancyContract.Transaction.DATE));
    int amount = cursor.getInt(cursor.getColumnIndex(AccountancyContract.Transaction.AMOUNT));
    categoryID = cursor.getLong(cursor.getColumnIndex(AccountancyContract.Transaction.CATEGORY_ID));
    accountID = cursor.getLong(cursor.getColumnIndex(AccountancyContract.Transaction.ACCOUNT_ID));
    String note = cursor.getString(cursor.getColumnIndex(AccountancyContract.Transaction.NOTE));

    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd kk:mm:ss");

    Date time = null;
    try {
        time = formatter.parse(date);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    dateTime.setTime(time);

    transactionAmountEditText.setText(String.valueOf(Math.abs(amount)));
    noteEditText.setText(note);
}
setInitialDateTime();
}
@Override
protected void onResume() {
    super.onResume();

    // Инициализация выпадающих списков
    initializeSpinners();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Обработка нажатия на элементе из меню
    int id = item.getItemId();
    if (!item.isChecked())
        item.setChecked(true);

    if (id == R.id.delete) {
        // Если нажат элемент "Удалить"

        // Создать диалоговое окно для подтверждения действия
        // и удалить элемент в случае необходимости

```

						Арк.
						80
Змн.	Арк.	№ Докум.	Підпис	Дата		


```

AlertDialog.Builder dialog = new AlertDialog.Builder(SingleTransactionActivity.this);
dialog
    .setTitle("Подтверждение действия")
    .setMessage("Вы действительно хотите удалить данную запись?")
    .setPositiveButton("Да", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            db.delete(AccountancyContract.Transaction.TABLE_NAME,
                AccountancyContract.Transaction._ID + "=?",
                new String[] {String.valueOf(getEntityId())}
            );
            finish();
        }
    })
    .setNegativeButton("Нет", null)
    .create();
dialog.show();
}
return true;
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (data == null || !data.hasExtra("result"))
        return;

    double amount = data.getDoubleExtra("result", 0);
    transactionAmountEditText.setText(String.valueOf(Math.abs(amount)));
}

private void initializeViews() {
    // Инициализировать переменные представления
    categoriesSpinner = (Spinner)findViewById(R.id.transaction_category);
    accountsSpinner = (Spinner)findViewById(R.id.transaction_account);
    currentDateTextView = (TextView)findViewById(R.id.transaction_date);
    currentTimeTextView = (TextView)findViewById(R.id.transaction_time);
    transactionAmountEditText = (EditText)findViewById(R.id.transaction_sum);
    noteEditText = (EditText)findViewById(R.id.transaction_note);
    signButton = (Button) findViewById(R.id.button_sign);
}

private void initializeSpinners() {
    // Инициализировать выпадающие списки соответствующими данными
    SimpleCursorAdapter categoriesAdapter, accountsAdapter;

    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Category.TABLE_NAME +
        " WHERE " + AccountancyContract.Category.IS_OUTGO + "=" + categoryType +
        " ORDER BY " + AccountancyContract.Category.IS_OUTGO + " DESC" + AccountancyContract.COMMA +
        AccountancyContract.Category.C_TITLE + " ASC", null);
    categoriesAdapter = new SimpleCursorAdapter(this, android.R.layout.simple_spinner_item, cursor,
        new String[] { AccountancyContract.Category.C_TITLE }, new int[] { android.R.id.text1 }, 0);
    categoriesAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    categoriesSpinner.setAdapter(categoriesAdapter);

    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Account.TABLE_NAME, null);
    accountsAdapter = new SimpleCursorAdapter(this, android.R.layout.simple_spinner_item, cursor,
        new String[] { AccountancyContract.Account.A_TITLE }, new int[] { android.R.id.text1 }, 0);
    accountsAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    accountsSpinner.setAdapter(accountsAdapter);
}

```

						Арк.
						81
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

// Выбрать элементы, соответствующие платежу
// Если Activity было вызвано для изменения платежа
if (isEntityId()) {
    Utilities.selectSpinnerItem(categoriesAdapter, categoriesSpinner, categoryID);
    Utilities.selectSpinnerItem(accountsAdapter, accountsSpinner, accountID);
}
}
private void initializeListeners() {
    // Инициализация слушателей
    onTimeSetListener = new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            dateTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
            dateTime.set(Calendar.MINUTE, minute);
            setInitialDateTime();
        }
    };

    onDateSetListener = new DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
            dateTime.set(Calendar.YEAR, year);
            dateTime.set(Calendar.MONTH, monthOfYear);
            dateTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            setInitialDateTime();
        }
    };

    signButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            isNegativeAmount = !isNegativeAmount;

            if (isNegativeAmount)
                signButton.setText("-");
            else
                signButton.setText("+");
        }
    });
}

private void setInitialDateTime() {
    currentDateTextView.setText(String.format("%td.%tm.%tY", dateTime, dateTime, dateTime));
    currentTimeTextView.setText(String.format("%tH:%tM", dateTime, dateTime));
}

public void setDate(View view) {
    // Выбор даты
    new DatePickerDialog(this, onDateSetListener,
        dateTime.get(Calendar.YEAR),
        dateTime.get(Calendar.MONTH),
        dateTime.get(Calendar.DAY_OF_MONTH)).show();
}

public void setTime(View view) {
    // Выбор времени
    new TimePickerDialog(this, onTimeSetListener,
        dateTime.get(Calendar.HOUR_OF_DAY),
        dateTime.get(Calendar.MINUTE), true).show();
}

public void saveTransactionContinue(View view) {
    // Сохранить платеж и создать новый
    if (executeSaving()) {

```

```

        Intent intent = new Intent(getApplicationContext(), SingleTransactionActivity.class);
        startActivity(intent);
    }
}
public void saveTransactionClose(View view) {
    // Сохранить платеж и закрыть Activity
    executeSaving();
}

private boolean executeSaving() {
    if (executeDataBaseSaving()) {
        finish();
        return true;
    }
    return false;
}

public boolean executeDataBaseSaving() {
    // Сохранение платежа

    // Валидация данных
    if (!validateData())
        return false;

    double amount = Double.parseDouble(transactionAmountEditText.getText().toString());
    amount *= isNegativeAmount ? -1 : 1;

    ContentValues contentValues = new ContentValues();
    contentValues.put(AccountancyContract.Transaction.DATE, Utilities.getSQLiteTimeString(dateTime));
    contentValues.put(AccountancyContract.Transaction.AMOUNT, amount);

    contentValues.put(AccountancyContract.Transaction.ACCOUNT_ID, accountsSpinner.getSelectedItemId());
    contentValues.put(AccountancyContract.Transaction.CATEGORY_ID, categoriesSpinner.getSelectedItemId());
    contentValues.put(AccountancyContract.Transaction.NOTE, noteEditText.getText().toString());

    // Если элемент изменялся - обновить его в базе данных
    // Иначе - создать новый
    if (isEntityId()) {
        db.update(AccountancyContract.Transaction.TABLE_NAME, contentValues,
            AccountancyContract.Transaction._ID + "=?",
            new String[] {String.valueOf(getEntityId())});
    }
    else {
        db.insert(AccountancyContract.Transaction.TABLE_NAME, null, contentValues);
    }

    return true;
}

private boolean validateData() {
    // Валидация данных
    if (transactionAmountEditText.getText().toString().isEmpty()) {
        Utilities.makeToast(this, "Поле сумма обязательно для заполнения");
        return false;
    }

    if (accountsSpinner.getSelectedItemPosition() < 0) {
        Utilities.makeToast(this, "Необходимо выбрать для какого счёта производится операция");
        return false;
    }

    if (categoriesSpinner.getSelectedItemPosition() < 0) {
        Utilities.makeToast(this, "Необходимо выбрать категорию");
    }
}

```

						Арк.
						83
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

        return false;
    }

    cursor = db.rawQuery("SELECT * FROM " + AccountancyContract.Category.TABLE_NAME +
        " WHERE " + AccountancyContract.Category._ID + "=?",
        new String[] {String.valueOf(categoriesSpinner.getSelectedItemId())});

    if (cursor.moveToFirst()) {
        int isOutgo = cursor.getInt(cursor.getColumnIndex(AccountancyContract.Category.IS_OUTGO));
        double sum = Double.parseDouble(transactionAmountEditText.getText().toString());

        if (sum == 0) {
            Utilities.makeToast(this, "Сумма не может равняться нулю");
            return false;
        }
        else if (isOutgo > 0 && !isNegativeAmount) {
            Utilities.makeToast(this, "Недопустима положительная сумма для категории \"Траты\"");
            return false;
        }
        else if (isOutgo == 0 && isNegativeAmount) {
            Utilities.makeToast(this, "Недопустима отрицательная сумма для категории \"Пополнения\"");
            return false;
        }
    }

    return true;
}

public void addNewCategory(View view) {
    // Добавить новую категорию
    Intent intent = new Intent(getApplicationContext(), SingleCategoryActivity.class);
    startActivity(intent);
}

public void addNewAccount(View view) {
    // Добавить новый счет
    Intent intent = new Intent(getApplicationContext(), SingleAccountActivity.class);
    startActivity(intent);
}

private int getCategoryType() {
    if (isEntityId()) {
        cursor = db.rawQuery("SELECT " + AccountancyContract.Category.IS_OUTGO + " FROM " +
            AccountancyContract.Transaction.TABLE_NAME +
            " INNER JOIN " + AccountancyContract.Category.TABLE_NAME + " ON " +
            AccountancyContract.Transaction.CATEGORY_ID + " = " +
            AccountancyContract.Category.TABLE_NAME + "." + AccountancyContract.Category._ID +
            " WHERE " + AccountancyContract.Transaction.TABLE_NAME + "." +
            AccountancyContract.Transaction._ID + "=?",
            new String[] {String.valueOf(getEntityId())});
        cursor.moveToFirst();

        return cursor.getInt(cursor.getColumnIndex(AccountancyContract.Category.IS_OUTGO));
    }
    else {
        Bundle extras = getIntent().getExtras();
        if (extras != null && extras.containsKey("category")) {
            return extras.getInt("category");
        }
    }
    return -1;
}

```

						Арк.
						84
Змн.	Арк.	№ Докум.	Підпис	Дата		

```

    }

    public void callCalculatorForResult(View view) {
        Intent intent = new Intent(getApplicationContext(), CalculatorActivity.class);

        String amount = "";
        if (!transactionAmountEditText.getText().toString().isEmpty()) {
            amount = isNegativeAmount ? "-" : "";
            amount += transactionAmountEditText.getText().toString();
        }

        intent.putExtra("result", amount);
        startActivityForResult(intent, 0);
    }
}

```

Файл UsingDataBaseActivity.java

```

package com.application.homeaccountancy.Activity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import com.application.homeaccountancy.Data.SQLiteHandler;

// Класс, предоставляющий возможность Activity работать
// с базами данных
public abstract class UsingDataBaseActivity extends AppCompatActivity {
    // Вспомогательный класс для работы с БД
    protected SQLiteHandler handler;

    // База данных
    protected SQLiteDatabase db;

    // Курсор для доступа к данным
    protected Cursor cursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // инициализация переменных
        handler = new SQLiteHandler(getApplicationContext());
        db = handler.getReadableDatabase();
    }

    @Override
    public void onDestroy(){
        super.onDestroy();

        // При уничтожении Activity - освободить используемые ресурсы
        if (db != null)
            db.close();

        if (cursor != null)
            cursor.close();
    }
}

```

						Арк.
						85
Змн.	Арк.	№ Докум.	Підпис	Дата		