МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

"Інформаційна система для аналізу ринку та підрахунку депозитів в Україні"

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Берест О.Б.**

Студент гр. ІН–61 **Віршич В.А.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую _____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

Завдання

до випускної роботи

Студентки четвертого курсу, групи ІН-61 спеціальності “Інформатика” денної форми навчання Віршич Валерії Андріївни.

Тема: “Інформаційна система для аналізу ринку та підрахунку депозитів в Україні”

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів побудови мобільного застосунку; 2) постановка завдання й формування завдань дослідження; 3) огляд і опис засобів для розробки; 4) розробка інформаційної системи аналізу ринку депозитів; 5) аналіз результатів.

Дата видачі завдання “_____”_____2020 р.

Керівник випускної роботи _____ Берест О.Б.

Завдання прийняв до виконання _____ Віршич В.А.

РЕФЕРАТ

Записка: 60 стор., 41 рис., 7 додатків, 8 джерел.

Об'єкт дослідження — Інформаційна система перегляду ринку депозитів.

Мета роботи — розробити інформаційну систему для зручного перегляду у смартфоні, яка повинна бути швидкою при великій кількості інформації, зручна в дизайні і зрозуміла для звичайного користувача. Програма повинна працювати в усіх сучасних смартфонах з операційною системою Android.

Результати — виконано вибір методів вирішення поставленої задачі; на основі мови Java та бібліотек Selenium, GSON, JSOUP реалізовано клієнтську частину, а за допомогою AndroidStudio на базі Java, візуальну частину застосунка.

МОБІЛЬНИЙ ДОДАТОК, ДЕПОЗИТИ, JAVA, ANDROID, SELENIUM, GSON, JSOUP, JSON

ЗМІСТ

ВСТУП

Робота присвячена проблемі відсутності зручного моніторингу усіх найбільших банків з наданням депозитів.

Сьогодні власники грошових накопичень мають можливість покласти їх в банк і через деякий проміжок часу отримати за це гарантований приріст накопичень. Дуже і дуже заманливо, адже збільшення накопичень відбувається без участі третьої сторони. Однак в цій операції є свої нюанси, які не можна ігнорувати. Напевно кожен користувач в інтернеті, який прагне віднайти потрібний йому депозит, зіштовхується з проблемою пошуку вигідного та відповідного депозиту може поглинути людину на години. Задля того, щоб добитися гарантованого примноження капіталу, потрібно дослідити всі плюси і мінуси депозитів.

Економічне становище в кожній країні не завжди стабільне, а інфляція та економічна криза може поглинути практично весь прибуток. Клієнт банку, ризикує вибрати не дуже надійний банк, який обіцяє на початку співпраці великі доходи, але в наприкінці компанія стає банкрутом. Тому у багатьох інвесторів, які готові вкласти гроші в депозит, виникає питання, куди найвигідніше їх вкласти? Кожен день на ринку депозитів стаються зміни, отже дуже важко слідкувати за тим, де краще всього покласти власні кошти на збереження. В кожному банку свій сайт та додаток для огляду пропозицій. Але ж банків так багато, а пропозицій ще більше. Звісно існують веб-сайти, на яких можна відшукати депозит за нашим запитом, але ж ми зараз живемо в ері смартфонів.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Коротка інформація про мобільний додаток

Мобільний додаток - це спеціально розроблене під функціональні можл ивості гаджетів програмне забезпечення. Призначення ПО може бути найрізн оманітнішим: сервіси, магазини, розваги, онлайн-
помічники і інше. Ці додатки завантажуються і встановлюються самим корис тувачем через мобільні маркетплейси. Найбільші майданчики - AppStore, Goo gle Play. Технічно всі програми створюються під конкретну платформу мобіл ьного гаджета. Найбільш популярні операційні системи - iOS, Android, Windo ws Phone.

1.2 Зберігання об’єктів в форматі JSON

Під час роботи з великою кількістю об’єктів, виникає потреба в транспортуванні даних між сервером та клієнтом. Зазвичай це відбувається шляхом передачі інформації через API. Для зручної роботи з даною технологією, зручно надсилати дані в форматі JSON. JSON - це загальний формат для представлення значень і об'єктів. Спочатку він був створений для JavaScript, але багато інших мов також мають бібліотеки, які можуть працювати з ним. JSON - текстовий формат зберігання даних, повністю незалежний від мови програмування, але він використовує угоди, з якими працюють програмісти C-подібних мов. Ці властивості роблять дану технологію ідеальною для обміну даними. Об’єкти в JSON, зберігаються в парі ключ – значення.

1.3 Огляд відомих рішень

Одне з рішень даної проблеми було знайдено під час пошуку користуючись однією з найпопулярніших пошукових систем «google.com». Рішення представляє собою веб-сайт «http://investfunds.ua/banks/» [1]. Воно засноване на відтворенні таблиці з даними депозитів різних українських банків

(див. рис. 1). Є можливість зробити пошук депозиту з відповідними даними.

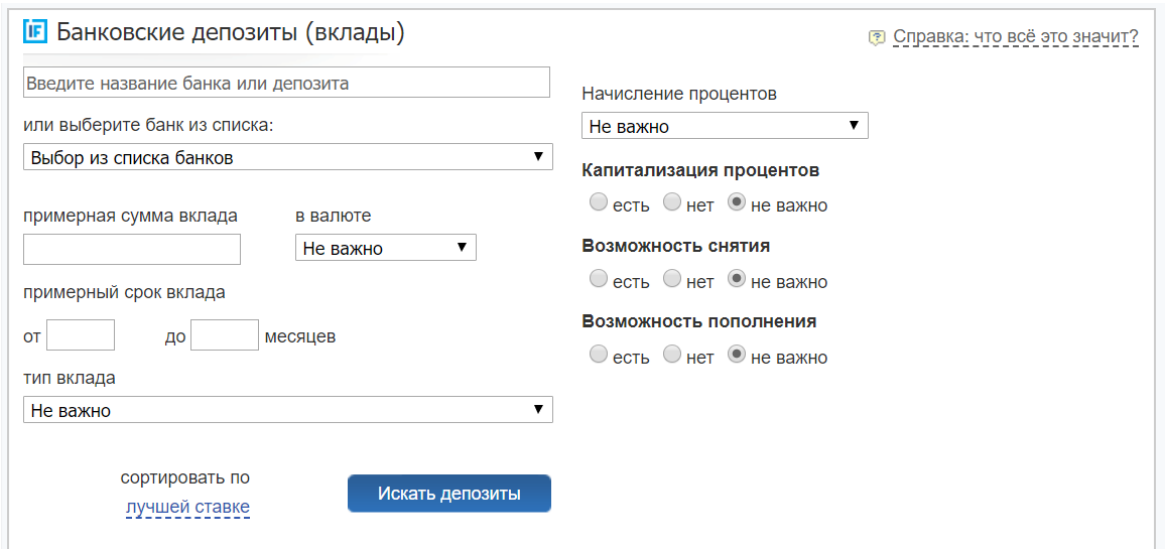

Рисунок 1 – Скріншот 1 сайту «http://investfunds.ua/banks/».

Але в даного рішення є недоліки. Дизайн сайту має досить скудний дизайн та досить малу інформацію про усі депозити банків (див. рис. 2). Це зовсім не дає повну картину усіх можливих депозитів.

Найдено 6 депозитов.

Название депозита	Название банка	Ваша ставка	Начисление процентов
Срочный (без возможности досрочного расторжения)	Альфа-Банк	16% UAH/12 мес./3 000	В конце срока
Срочный (без возможности досрочного расторжения)(ежемесячно)	Альфа-Банк	15% UAH/12 мес./3 000	Ежемесячно
Срочный (с возможностью досрочного расторжения)	Альфа-Банк	13% UAH/12 мес./3 000	В конце срока
Срочный (с возможностью досрочного расторжения)(ежемесячно)	Альфа-Банк	12% UAH/12 мес./3 000	Ежемесячно
Сберегательный	Альфа-Банк	10% UAH/0 мес./1 000	Ежемесячно
Депозит с выдачей именного сберегательного (депозитного) сертификата	Альфа-Банк	4.5% USD/12 мес./5 000	В конце срока

Рисунок 2 – Скріншот 2 сайту «http://investfunds.ua/banks/».

Веб-сторінка «https://minfin.com.ua/deposits/» [2] також має подібне рішення (див. рис. 3).

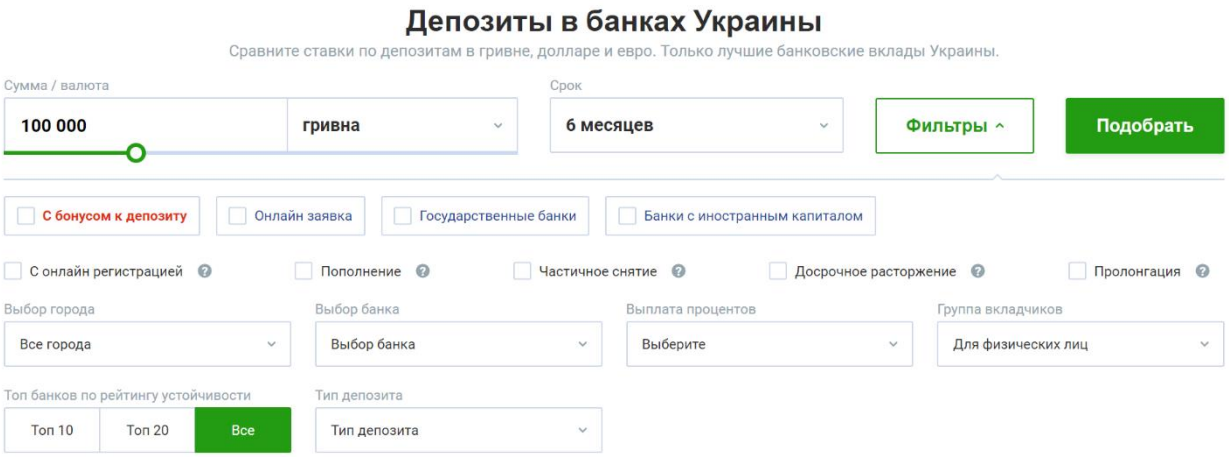

Рисунок 3 – Скріншот 1 сайту «https://minfin.com.ua/deposits/».

В даного рішення досить багато полів для пошуку, це робить інтерфейс більш громіздким та не зручним. Біля деяких полів є підказки, що означає той чи інший термін, це дуже зручно та корисно.

На рисунку №4 представлено результати пошуку. Сайт одразу дає інформацію про процентну ставку та дохід.

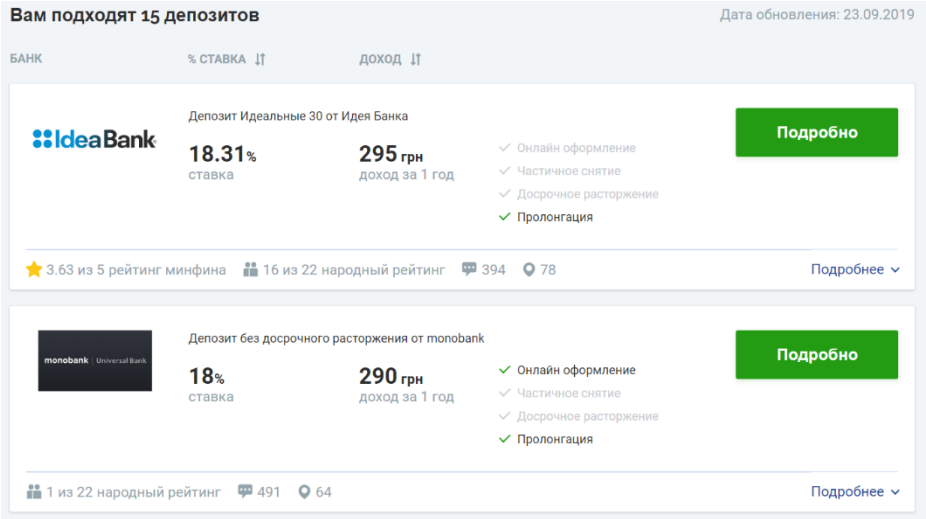

Рисунок 4 – Скріншот 2 сайту «https://minfin.com.ua/deposits/».

Якщо ж натиснути на кнопку «Подробно», то відкриється більш детальна інформація про саме цей депозит (див. рис. 5):

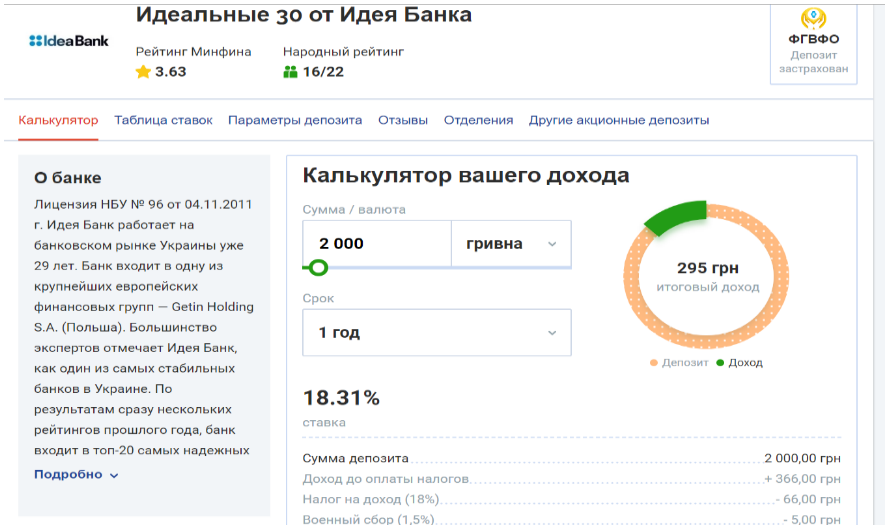

Рисунок 5 – Детальна інформація.

Є можливість змінити дані, та калькулятор самостійно порахує дохід, що є досить корисною функцією.

Але даний сайт має як багато переваг так і багато недоліків. Інтерфейс сайту є досить громіздким. Лише декілька результатів пошуку поміщаються на екрані. Занадто багато додаткової та непотрібної інформації. Не має потреби перераховувати таку велику кількість інформації, адже це може злякати користувача.

1.4 Постановка задачі

У даній роботі необхідно розробити інформаційну систему для зручного моніторингу ринку та розрахунків депозитів в Україні.

Вимоги до продукту:

- Актуальні дані
- Зручний інтерфейс
- Можливість зробити розрахунок доходу
- Можливість пошуку по введеним даним
- Можливість розрахувати власний депозит по введеним даним
- Рейтингова таблиця усіх найпопулярніших банків
- Діаграма депозитів

Результатом роботи має стати мобільний застосунок, який повинен бути швидким при великій кількості інформації, простий в дизайні і зрозумілий для користувача. Застосунок повинен бути адаптивним до більшості пристроїв на операційній системі Android, також повинен працювати у більшості сучасних смартфонах.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Вибір платформи

Наразі існує декілька методів створення доданків. Десктопний додаток, мобільний додаток та звісно веб-сайт. Десктопний додаток наразі досить мало популярний у наш час. Люди звісно користуються додатками на своєму ПК, але це дуже вузьке коло. Більш за все це іграшки або ж програми для роботи.

Рисунок 6 – Дослідження тенденції пошукових запитів [7].

Без смартфона або планшета важко уявити сучасну людину. Доступ до потрібної інформації в будь-якому кутку світу докорінно впливає на швидкість прийняття рішення. З кожним роком кількість власників мобільних портативних девайсів мобільного світу неухильно зростає, а з 2015 року частка пошукових запитів в Google [8], зроблених з мобільного, вперше за всю історію Інтернету перевищила кількість пошукових запитів з робочого столу (див. рис. 6 та рис. 7). І ця тенденція буде все більше збільшуватися.

Рисунок 7 – Динаміка приросту користувачів в Інтернеті.

Популярність мобільних додатків: що говорять дослідження?

- 90% часу користувачі мобільних пристроїв проводять в додатках і тільки 10% займає серфінг в інтернеті. (Дослідження Flurry [8])
- 58% власників смартфонів в США використовують свої гаджети для онлайн покупок, 66% з них роблять це щотижня. (Звіт Google «How people shop on their phones»).
- В середньому 4 додатки встановлено на смартфонах американців і 29% з них щотижня використовуються для онлайн покупок. (Звіт Google «How people shop on their phones»)
- 85% користувачів смартфонів віддають перевагу мобільному додатку ніж сайту. (Дослідження Compuware).

Найголовніший вибір, при розробці мобільних пристроїв – це вибір платформи мобільного пристрою. Для цього потрібно слідкувати за тенденціями ринку. Як видно на прогнозуванні (див. рис. 8), абсолютним лідером залишається Android.

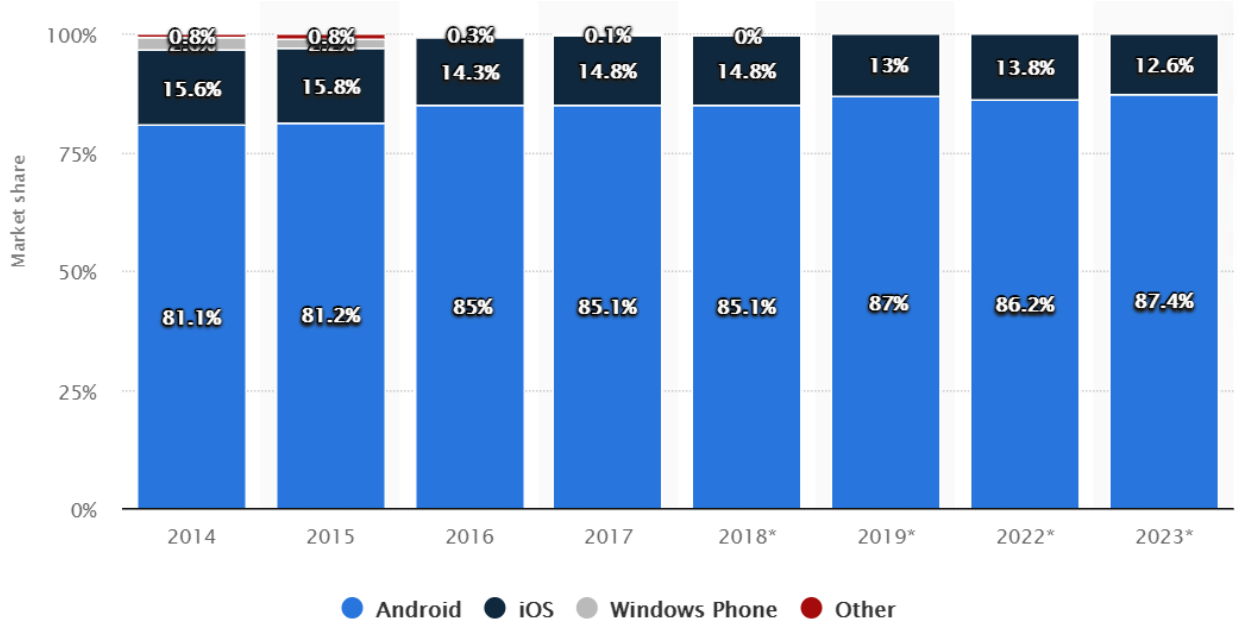

Рисунок 8 – Дослідження ринку операційних систем смартфонів.

4 переваги мобільного застосунку:

1. **Мобільні додатки зручніше і швидше мобільних сайтів**.

Високий темп життя робить користувачів дуже вимогливими до швидкості завантаження сайтів та інтернет-магазинів на екранах гаджетів. Згідно з дослідженням Kissmetrics, 46% користувачів смартфонів залишають сторінку якщо вона не відкривається протягом перших 10 секунд.

В цьому відношенні додатки поза конкуренцією. Вони завантажуються швидше мобільних сайтів, так як зберігають призначені для користувача налаштування і не вимагають завантаження всього контенту в браузері; Їхня робота практично не залежить від швидкості і якості інтернет-з'єднання.

Крім швидкості, додатки перевершують сайти в таких аспектах:

- доступ до нативним функцій смартфона (відеокамері, GPS-навігатора, функції розпізнавання голосу),
- інтерактивність (інтеграцією з соціальними мережами) і персоналізація контенту.
- зручність у використанні та найкраща адаптація під мобільні пристрої.

Загальновідомо, користувачі вибирають те, що швидше і зручніше, отже, ваші клієнти напевно віддадуть перевагу додатку ніж сайту.

2. **Мобільні додатки підвищують лояльність клієнтів**.

Згідно з даними Criteo, коефіцієнт утримання клієнтів (Customer Retention Rate, CRR) у додатків на 50% вище, ніж у мобільних сайтів. Це означає, що додаток у порівнянні з браузерної версією інтернет-магазину в два рази збільшує ймовірність повторних покупок. Причини високого CRR додатків пов'язані з тим, що телефон завжди під рукою, а іконка програми, немов яскравий банер, постійно нагадує користувачеві про можливість зробити покупку в мережі.

3. **Додатки краще конвертують відвідувачів в покупців і підвищують обсяг продажів магазину**.

Дослідження Criteo наочно демонструють, що мобільні додатки збільшують продажі і дохід.

Коефіцієнт конверсії у додатків на 200% вище, ніж у мобільних сайтів; Середній чек додатків на 140% перевищує аналогічні показники мобільної версії веб-ресурса. Коеффіцієнт відмов на стадії оформлення замовлення в додатках становить тільки 20%, в той час як в мобільних версіях сайту - 97%, а в десктопних - 68%.

Додатки підштовхують користувачів до покупок, тим більше, що їх функціонал дозволяє оплатити замовлення в один клік.

4. **Додатки забезпечують більш високий рівень взаємодії з клієнтом**.

Додатки дозволяють ефективно впливати на користувачів смартфонів за допомогою розсилки push-повідомлень з рекламними повідомленнями і акціями (якщо користувачі не відключили цю функцію на своїх гаджетах).

Згідно з дослідженнями SocialMediaToday, push-повідомлення більш ефективні в порівнянні з розсилкою e-mail і SMS-повідомлень з наступних причин:

- Миттєва доставка адресату і велике охоплення аудиторії;
- Середній відсоток відкриття 90%, при 23% для e-mail повідомлень;
- Низька вартість рекламної кампанії;
- Відсутність спаму та шкідливих програм;
- Низький відсоток відмов;
- Можливість використання інструменту Deep Linking.

- Грамотна розсилка push-повідомлень здатна конвертувати користувачів в покупців і повернути інтерес до додатка клієнтів які «дрімають».

Отже можемо дійти до такого висновку, що останні кілька років електронна комерція семимильними кроками рухається в мобільну площину. З огляду на сучасні тенденції, розробка додатків для мобільних пристроїв забезпечить інтернет-магазинах конкурентну перевагу, збільшить продажі і лояльність клієнтів бренду. Без сумніву, майбутнє за мобільними додатками.

2.2 Вибір методів розроблення

Програмний продукт зручно розділити на 2 основні структури: сервер та додаток з інтерфейсом.

Серверна частина – слугує для отримання, конвертування, зберігання та надсилання актуальних даних.

Для автоматичного зчитування, конвертування та зберігання актуальної інформації застосуємо такі бібліотеки:

Selenium WebDriver - це інструмент для автоматизації дій веб-браузера. У більшості випадків використовується для тестування Web-додатків, але цим не обмежується. Зокрема, він може бути використаний для виконання рутинних завдань адміністрування сайту або регулярного отримання даних з різних джерел (сайтів).

Jsoup - це бібліотека з відкритим кодом, призначена для аналізу, вилучення та маніпулювання даними, що зберігаються в документах HTML.

Gson (також відомий як Google Gson) - це бібліотека з відкритим кодом для серіалізації та десеріалізації об'єктів Java до (і від) JSON.

2.3 Вибір засобів програмування

Серед усіх мов програмування досить яскраво виділяється Java. Якщо нам необхідно розширити наш програмний продукт, то Java ідеальний варіант. У цій мові, існують інструменти для створення серверу, веб-сайтів, мобільних

застосунків та налаштована робота з усіма найпопулярнішими базами даних.

Нижче перерахування найбільш вагомі переваги:

1. **Гнучкість.** Java довела, що C - процедурний, керований вручну і залежить від платформи код - це не межа досконалості.
2. **Об'єктно-орієнтоване програмування.** При ООП можна повторно використовувати об'єкти в інших програмах. Запобігає помилки, оскільки об'єкти приховують інформацію, до якої не повинно бути доступу [3]. Більш ефективно організовує структуру програм, в тому числі великих. Спрощує обслуговування і модернізацію старого коду.
3. **Java - це мова високого рівня,** тобто він схожий на людську мову. На відміну від мов низького рівня, які нагадують машинний код [4]. Мови високого рівня перетворюється за допомогою компіляторів або інтерпретаторів. Це спрощує розробку, роблячи мову більш легким для написання, читання і обслуговування.
4. **Синтаксис** Java заснований на C ++, тому Java схожа на C. Проте, синтаксис Java простіше, що дозволяє новачкам швидше вчитися і ефективніше використовувати код для досягнення конкретних результатів.
5. **Безпека.** Існує думка, що Java - безпечний мова, проте це не зовсім так. Сама мова не захищає вас від вразливостей, але деякі його функції усувають поширені уразливості. По-перше, на відміну від C, в Java немає покажчиків. Покажчик - це об'єкт, який зберігає адресу осередки пам'яті іншого значення, що може викликати несанкціонований доступ до пам'яті [5]. По-друге, в Java є Security Manager, створена для кожної програми політика безпеки, в якій можна вказати правила доступу. Це дозволяє запускати додатки Java в «пісочниці» і усувати таким чином уразливості.
6. **Незалежність від платформи** («Написати один раз і використовувати скрізь»). Можна створити Java-додаток на Windows, скомпілювати його в байт-код і запустити його на будь-який інший платформі, яка підтримує

віртуальну машину Java (JVM). Таким чином, JVM служить рівнем абстракції між кодом і обладнанням.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Проектування

Для того, щоб описати функціональність та поведінку додатку, розробимо модель та діаграму варіантів використання. Для того щоб розробити правильно діаграму прецедентів, системний аналітик повинен створити такі умови:

- чітко відокремити систему від її середовища;
- визначити дійових осіб (акторів) в системі, їх взаємодія з системою і на очікуваний результат функціональності системи (див. табл. 1);
- визначити поняття предметної області, які відносяться до детального опису функціональності системи.

Таблиця 1 – Актори системи

Назва актора	**Короткий опис**
актори-користувачі:	
Користувач	Людина, яка здійснює переглядає інформацію про депозити банків

Робота над діаграмою може початися з текстового опису (див. табл. 2). Так як існують різні способи роботи з продуктом, потрібно описати усі доступні варіанти сценарію (див. табл. 3).

Таблиця 2 – Варіанти використання системи

Назва ВВ	**Короткий опис**
1 Запуск програми	**Для користувача:** *1.1 Передумови:* Користувач запустив програму *1.2 Процес:* Користувач чекає пока програма запуститься. *1.3 Післяумови :* Користувач потрапляє на головне вікно програми.
2 Обрати потрібну валюту	**Для користувача:** *2.1.1 Передумови:* Користувач потрапляє на головне вікно програми. *2.1.2 Процес:* Користувач обирає валюту для відображення існуючих депозитів у цій валюті. *2.1.3 Післяумови:* Дані в таблиці відповідають обраній валюті.
3 Обрати депозит	**Для користувача:** *3.1.1 Передумови:* Користувач потрапляє на головне вікно програми. *3.1.2 Процес:* Користувач обирає депозит та натискає на кнопку «Подробнее» *3.1.3 Післяумови:* Перехід на сторінку з усією інформацією про депозит.
4 Розрахувати дохід	**Для користувача:** *4.1.1 Передумови:* Користувач потрапляє на головне вікно програми. Відкриває меню. Обирає вкладку «Калькулятор». *4.1.2 Процес:* Користувач вводить дані задля розрахунку потрібного депозиту. *4.1.3 Післяумови:* Користувач отримує число придутку.

Таблиця 3 – Сценарій ВВ

Назва ВВ	**Короткий опис**
1 Запуск програми	*1.1 Передумова:* Користувач запустив додаток. *1.2 Процес:* Програма відображає таблицю з даними.
2 Вказати дані для пошуку	*2.1 Передумова:* Користувач запустив додаток та перейшов у меню. З меню він потравляє до вкладки «Поиск». *2.2 Процес:* Користувач обирає потрібні дані для пошуку. *2.3 Післяумова:* Програма відображає таблицю даними що відповідають пошуку.
4 Розрахувати дохід	**Для користувача:** *4.1.1 Передумови:* Користувач потрапляє на головне вікно програми. Відкриває меню. Обирає вкладку «Калькулятор». *4.1.2 Процес:* Користувач вводить дані задля розрахунку потрібного депозиту. *4.1.3 Післяумови:* Користувач отримує число придутку.

Після текстового опису усіх можливих сценаріїв, будуємо діаграму варіантів використання (див. рис. 9) відповідну до опису.

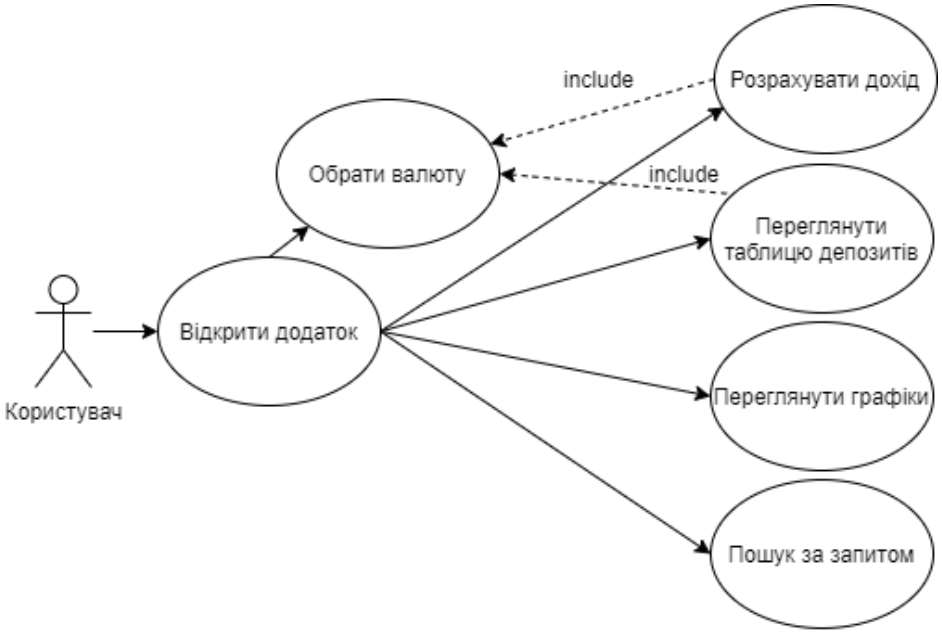

Рисунок 9 – Діаграма варіантів використання.

Наступним кроком, буде створення діаграми діяльності. Ця UML-діаграма, показує дії, стани яких описані на діаграмі станів. Під діяльністю розуміється специфікація виконуваного поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів - вкладених видів діяльності і окремих дій, з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого.

Діаграма діяльності добре відображає:

- послідовність дій;
- події, які розпочинають дії або є кінцевим результатом;
- умови збільшення сценарію;

На рисунку №10 зображено діаграму діяльності на якій спрощено показаний процес запиту пошукової інформації задля отримання вузького спектру потрібних депозитів.

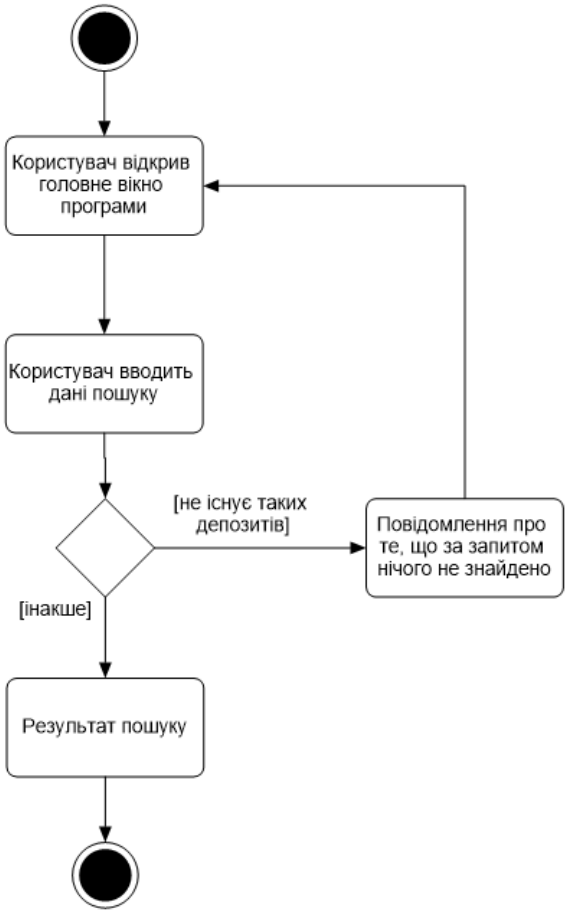

Рисунок 10 – Діаграма діяльності «Вказати дані для пошуку».

На рисунку №11 зображено діаграму діяльності на якій спрощено показаний процес дій депозиту, обраного (або ні) з таблиці пропонованих.

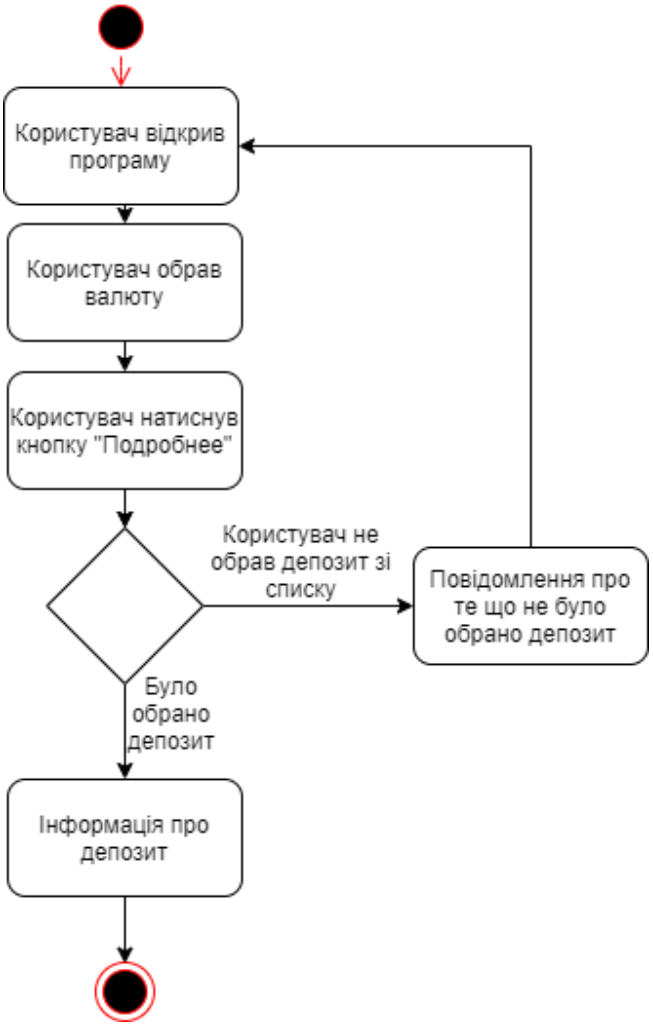

Рисунок 11 – Діаграма діяльності «Обрати депозит».

На рисунку №12 зображено діаграму діяльності на якій спрощено показаний процес дій розрахунку прибутку заданого депозиту.

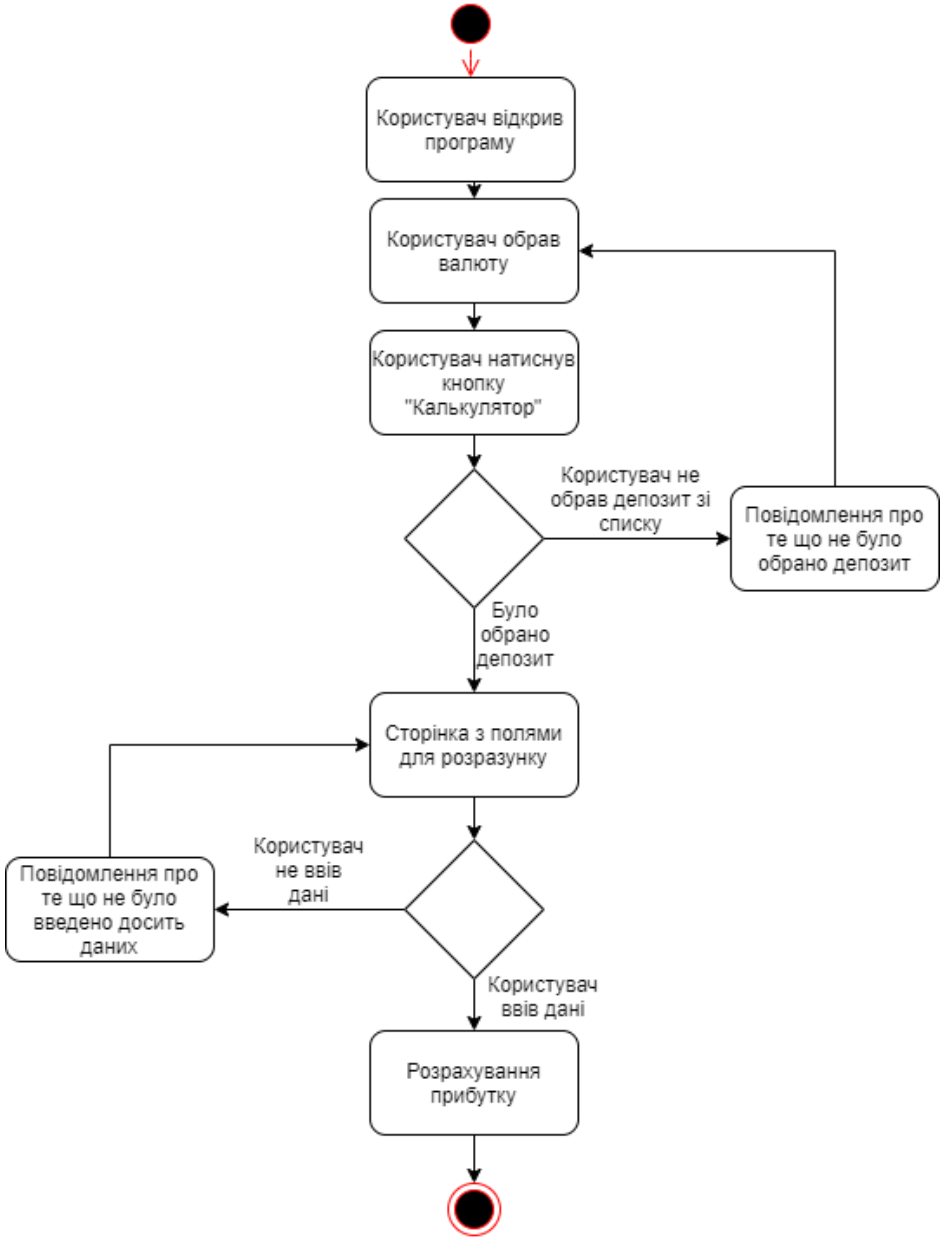

Рисунок 12 – Діаграма діяльності «Ввести дані для розрахунку».

3.2 Реалізації основних компонентів

1. Сервер

Найголовнішою частиною програми, є його сервер, так як наш інтерфейс без даних нагадує лише калькулятор. Ядро серверу складає програма, яка шляхом парсингу, зчитує інформацію з джерела. Для реалізації даної задачі, необхідно було використовувати 2 бібліотеки (Selenium та JSOUP) та спеціальний веб-драйвер для браузера. Selenium допоможе нам відтворити «клік» миші, а JSOUP вже зчитає дані з відповідного елементу. Клас «Browser» відповідає за дії з браузером.

```
public class Browser {
    private WebDriver driver;

//Задаємо драйвер для роботи з «Google Chrome»
    public void setDriver() {
        System.setProperty("webdriver.chrome.driver", "chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }

//Метод для того щоб відобразити веб-сторінку у браузері
    public void openPage(String page) {
        driver.navigate().to(page);
        WebElement element = driver.findElement(
                By.xpath("//*[@class='sc-282uzu-1 fOORvs']"));
        do {
            try {
                element.click();
            } catch (StaleElementReferenceException | NoSuchElementException e)
{
                element = null;
            }
        }
        while (element != null);
    }

//Зчитування дати зі сторінки
```

```
            public String getDate(String page) {
                driver.get(page);
             return driver.findElement(By.xpath(
"/html/body/main/div/div/section/div/div/div[2]/div/div[1]/div[2]/span[2]/time"))
                        .getText();
          }

        //Зчитування депозитів в список
            public ArrayList<Deposit> getDeposits(String currency) throws Exception {
                List<WebElement> table = driver.findElements(By.xpath(
              "//*[@id=\"root\"]/div/section/div/div/div[2]/div/div[2]/table/tbody"));
                String tableString = table.get(0).getText();
             String[] substring = tableString.split("\n");
                ArrayList<Deposit> deposits = new ArrayList();
                for (int i = 0; i + 3 < substring.length; i += 4) {
                    Pattern pattern = Pattern.compile(
                            "\\b(?<!\\.)(?!0+(?:\\.0+)?%)(?:\\d|[1-
   9]\\d|100)(?:(?<!100)\\.\\d+)?%");
                    Matcher matcher = pattern.matcher(substring[i + 1]);
                    String percent = "";
                    String depositName = "";
                    if (matcher.find()) {
                        depositName = substring[i + 1].substring(0, matcher.start());
                        percent = matcher.group();
                    }
                    deposits.add(new Deposit(substring[i], depositName, currency,
                            Float.parseFloat(percent.replaceAll("%", "")),
                            substring[i + 2], substring[i + 3]));
                }
                return deposits;
            }
        }
```

При запуску програми, вона перевіряє чи існують вже актуальні дані. Перевіряє дату, та наявність усіх 3 валют. Якщо дані не актуальні, або ж бракує файлу, сервер повністю зчитає нові дані з джерела. Створює для кожного депозиту, відповідний унікальний об’єкт депозиту, та конвертує усі дані в формат JSON (див. рис. 13).

```
public class Deposit {
        private String bankName;
        private String depositName;
        private String currency;
        private float percent;
        private String duration;
        private String payout;

        public Deposit(String bankName, String depositName, String currency,
                       float percent, String duration, String payout) {
            this.bankName = bankName;
            this.depositName = depositName;
            this.currency = currency;
            this.percent = percent;
            this.duration = duration;
            this.payout = payout;
        }
    }
```

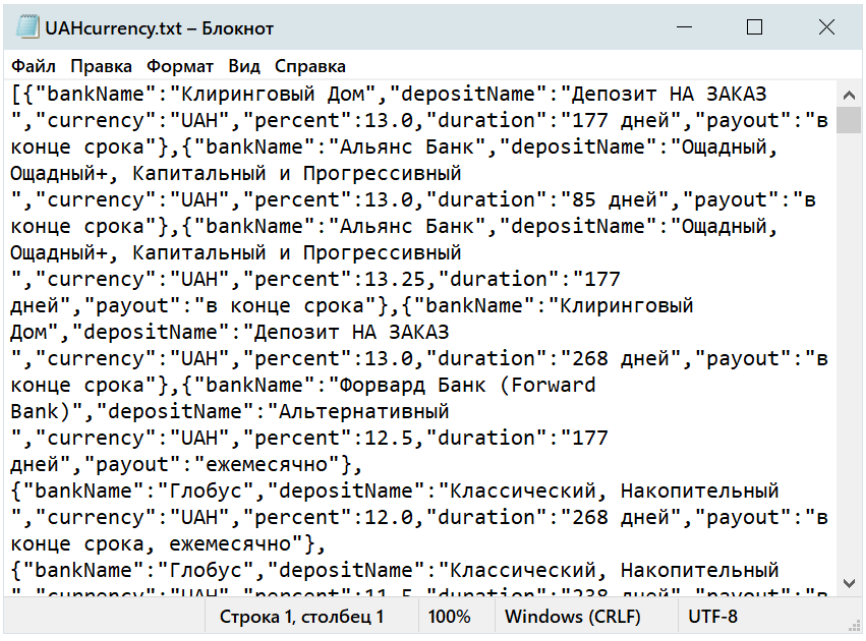

Рисунок 13 – Депозити в форматі JSON.

Завершальним кроком записує дані до файлу:

```
public void saveFile(ArrayList<Deposit> deposits) {
        try(FileWriter fileWriter = new FileWriter(file, false)) {
                gson.toJson(deposits, fileWriter);
        } catch (IOException e) {
            System.out.println("Something was wrong with uah currency file");
```

```
        }
    }
```

2. Клієнт

Для демонстрації загальної структури ієрархії класів системи мобільного застосунку, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними використовується діаграма класів (див. рис. 13). Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування.

Рисунок 14 – Діаграма класів.

Перед початком роботи з програмою, її слід встановити на сучасний смартфон з операційною системою Android. Після встановлення на робочому столі з’явиться іконка застосунку (див. рис. 15).

Рисунок 15 – Іконка програми.

Після запуску програма зчитує з файлів об’єкти та створює на їх основі об’єкти депозитів для подальшої роботи з ними.

```
public List<Deposit> readGSON(Activity activity, int f) {
        StringBuilder sb = new StringBuilder();
        try (InputStream is = activity.getResources().openRawResource(f);
             BufferedReader br = new BufferedReader(new InputStreamReader(is))) {
            String data = "";
            try {
                while ((data = br.readLine()) != null) {
                    sb.append(data);
                }
            } catch (IOException e) {
                System.out.println(e);
        }
        } catch (IOException e) {
            e.printStackTrace();
        }
        List<Deposit> deposits = gson.fromJson(sb.toString(), new
TypeToken<List<Deposit>>() {
        }.getType());
        return deposits;
    }
```

1. Обрати валюту

Після запуску з’являється головне вікно програми (див. рис. 16-17) в якій користувач має можливість обрати потрібну йому валюту.

Рисунок 16 – Вікно з вибором валюти.

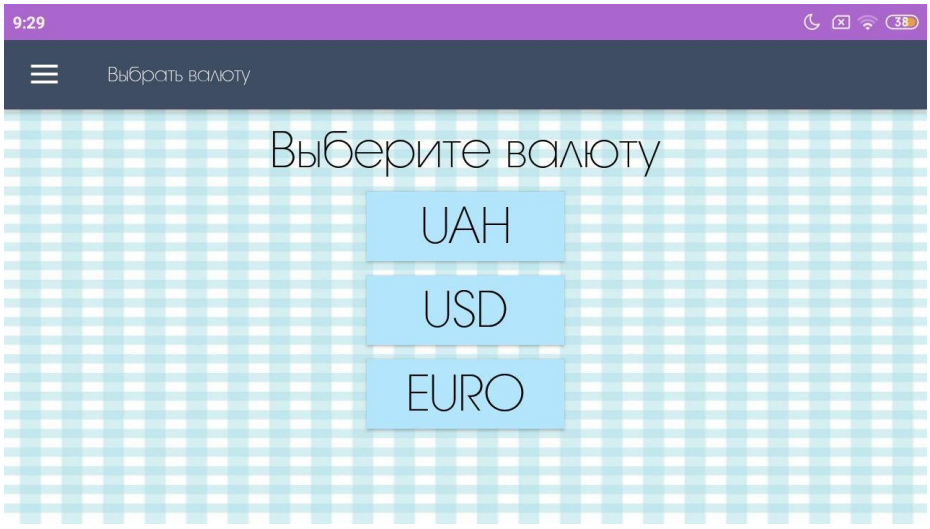

Рисунок 17 – Вікно з вибором валюти (горизонтальне положення).

За дії кожної кнопки, відповідає відповідний метод. Нижче показано метод, що відповідає за натискання кнопки з валютою у євро:

```
public void onClickEURO(View view) {

    GalleryFragment.currency = "EURO";
```

```
        setDefaultFields();
        move(GalleryFragment.getInstance());
    }
```

2. Навігаційне меню

У верхньому лівому кутку знаходиться кнопка виклику бокового меню (див. рис. 18), це навігація застосунку. З нього можна потрапити у будь-яке вікно програми.

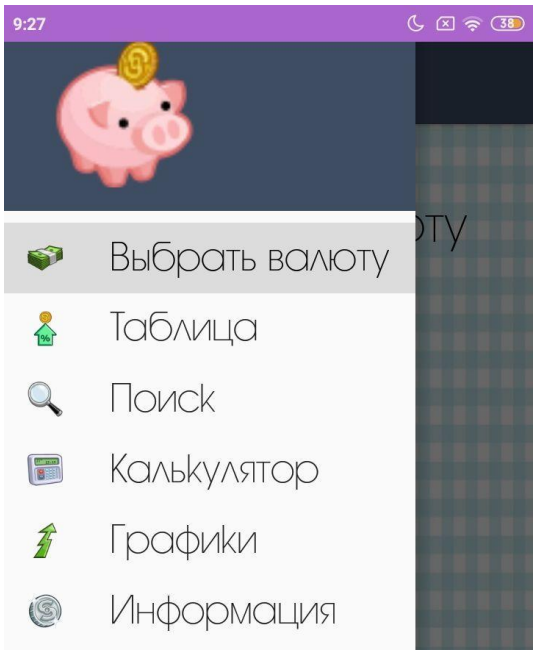

Рисунок 18 – Бокове меню.

3. Таблиця

Після того, як користувач обрав валюту, його переадресує на вкладку з таблицею лише з даною валютою (див. рис. 19-20). На дане вікно можна також потрапити через навігаційне меню, обрав пункт «Таблица». Якщо користувач попередньо не обирав валюту, за замовчуванням в таблиці будуть знаходитись лише депозиті в українській валюті.

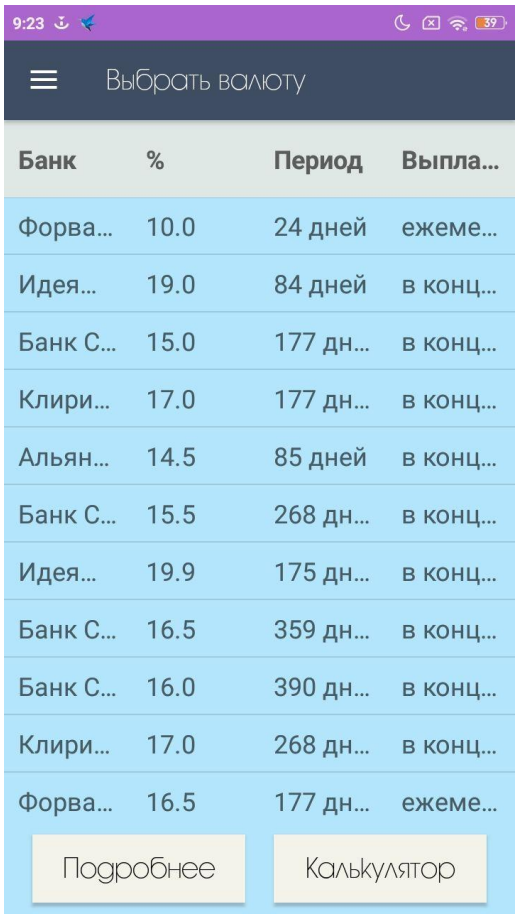

Рисунок 19 – Таблиця депозитів в українській валюті.

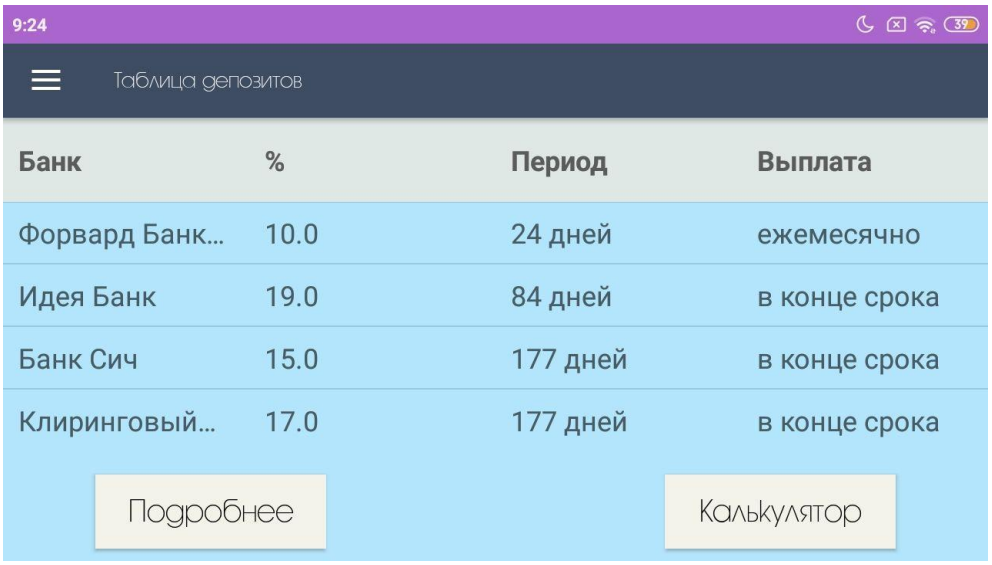

Рисунок 20 – Горизонтальне положення таблиці.

За відображення таблиці з відповідними стовпчиками, відповідає метод printTable:

```
public void printTable(View root) {
            TableView<String[]> tb = root.findViewById(R.id.tableView);
            tb.setColumnCount(4);
            tb.setHeaderBackgroundColor(Color.parseColor("#e0e8e5"));
            tb.setHeaderAdapter(new
SimpleTableHeaderAdapter(root.getContext(), getHeaders()));
            tb.setDataAdapter(new SimpleTableDataAdapter(root.getContext(),
fillTable()));
```

Було додано до методу «прослуховувач», для того щоб можна було обрати будь-який депозит зі списку, для подальшої роботи з ним.

```
            tb.addDataClickListener((TableDataClickListener) (rowIndex,
clickedData) -> {
                ShareViewModel.index = rowIndex;
                GalleryFragment.clicked = true;
            });
```

За наповнення таблиці змістом, відповідає метод fillTable:

```
    public String[][] fillTable() {
        List<Deposit> deposits;
        if (fields == null)
            deposits = manager.getDeposits();
        else
            deposits = manager.searchDeposit(fields);
        Deposit d;
        column = new String[deposits.size()][4];
        for (int i = 0; i < deposits.size(); i++) {
            d = deposits.get(i);
            column[i][0] = d.getBankName();
```

```
            column[i][1] = String.valueOf(d.getPercent());
            column[i][2] = d.getDuration();
            column[i][3] = d.getPayout();
        }
        return column;
    }
```

Під таблицею знаходиться 2 кнопки. Якщо користувач спробує натиснути на будь-яку кнопку, без попередньо обраного депозиту, з’явиться попереджувальне повідомлення про те, що користувач не обрав депозит зі списку, і неможливо виконати переадресацію на наступне вікно програми (див. рис. 21).

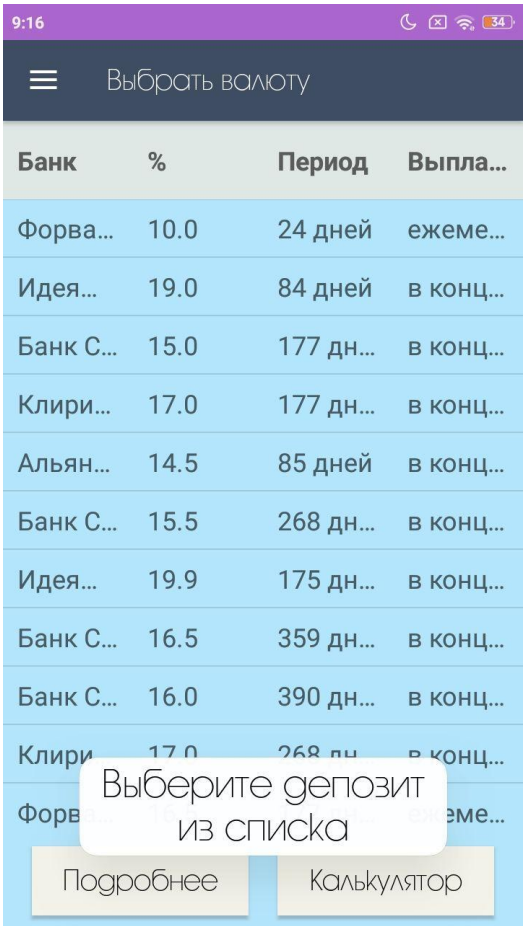

Рисунок 21 – Горизонтальне положення таблиці.

У нижньому лівому кутку знаходиться кнопка «Подробнее», переадресує користувача до вікна з детальною інформацією про депозит, що він обрав попередньо. У нижньому правому «Калькулятор». Якщо користувач попередньо обрав депозит зі списку, його переадресує до вікна з заповненими полями для розрахунку обраного депозиту.

4. Пошук

Наступним пунктом в навігаційному меню знаходиться пошук (див. рис. 22).

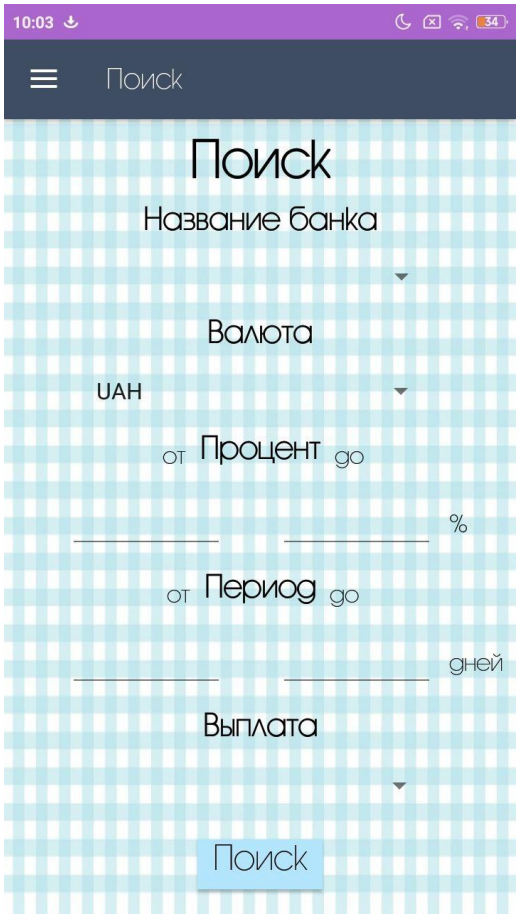

Рисунок 22 – Вікно з пошуком депозитів.

Дане вікно було створено для пошукових запитів користувачів. Можна обрати депозити потрібного банку. Вказати діапазон процентної ставки. Обрати потрібну валюту. Вказати діапазон терміну (адже усі банки мають різні термін виплатів). А також вподобання коли виплачується депозит. Користувач

має можливість залишити пустими майже усі поля, окрім валюти, та здійснити пошук. Програма самостійно проаналізує за якими параметрами відобразити депозити. Далі користувач автоматично переадресовується у вікно з результатами пошуку у вигляді таблиці.

5. Детальна інформація про депозит

Вікно слугує для відображення більш детальної інформації депозитів (див. рис. 23). Якщо одразу перейти у це вікно з бокового меню, користувачеві буде відображено за замовчуванням найперший у списку депозитів в українській валюті.

Рисунок 23 – Детальна інформація про депозит.

6. Калькулятор

Наступним кроком, розглянемо вікно з калькулятором депозитів (див. рис 24-25). В дане вікно можна потрапити з навігаційного меню, з вікна з таблицею після обрання депозиту зі списку та з детальної інформації депозиту.

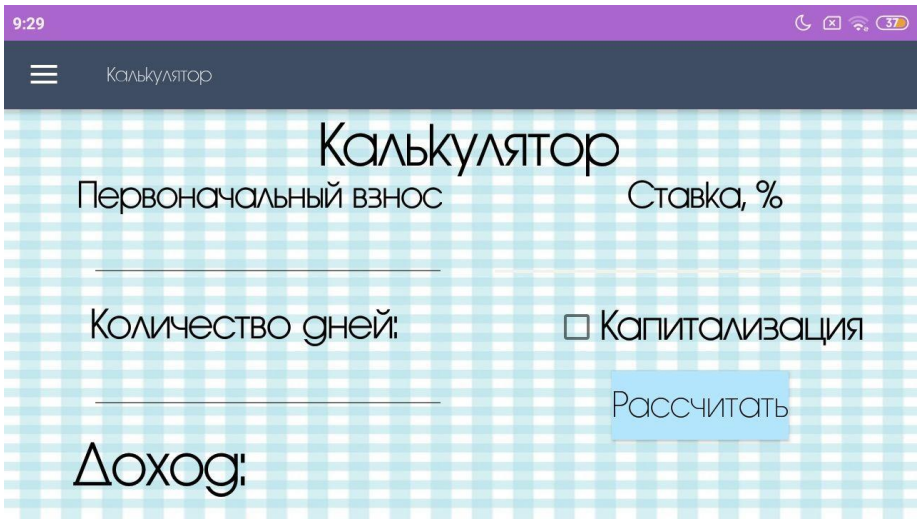

Рисунок 24 – Калькулятор депозитів (горизонтальне положення).

Рисунок 25 – Калькулятор.

Рисунок 26 – Раніше обраний депозит.

Якщо перейти з таблиці або ж детальної інформації, то поля, окрім суми,

будуть заповнені автоматично згідно тарифного плану депозиту (див. рис. 26).

В даному вікно усі поля є обов'язковими для заповнення. Користувач записує скільки він хоче покласти грошей на свій депозит. Яка процента ставка вказана у його депозиті, та кількість днів, за якими буде нараховуватися депозит. У разі потреби, можна обрати капіталізацію депозиту. Для розрахування депозиту використовується 2 формули: прості (див. рис. 27) та складні відсотки (див. рис. 28).

$$S_p = \frac{P \times I \times t}{K \times 100}$$

Рисунок 27 – Сума простих відсотків [6].

$$S_p = S - P = P \times \left[1 + \frac{I \times j}{100 \times K}\right]^n - P$$

Рисунок 28 – Сума складних відсотків [6].

Формула розрахунку складних відсотків застосовується для капіталізації депозиту. Після натиснення кнопки «Расчитать» - навпроти тексту «Доход» з'явиться дохід користувача.

Метод розрахунку депозиту:

```
public static void calculate(View view) throws NotEnoughDataException {
        EditText first = view.findViewById(R.id.first);
        int money;
        if (first.getText().toString().isEmpty())
            throw new NotEnoughDataException();
        else
            money = Integer.parseInt(first.getText().toString());
        float percent;
        EditText percentText = view.findViewById(R.id.percent);
        if (percentText.getText().toString().isEmpty())
            throw new NotEnoughDataException();
        else
            percent = Float.parseFloat(percentText.getText().toString());
```

```
        int days;
        EditText daysText = view.findViewById(R.id.days);
        if (daysText.getText().toString().isEmpty())
            throw new NotEnoughDataException();
        else
            days = Integer.parseInt(daysText.getText().toString());
        CheckBox checkBox = view.findViewById(R.id.checkBox);
        TextView textView = view.findViewById(R.id.sum);
        if (checkBox.isSelected())
            textView.setText(UtilPercents.getDifficultPercents(money, percent,
days));
        else
            textView.setText(UtilPercents.getSimplePercents(money, percent, days));
    }
```

Метод розрахунку простих процентів:

```
    public static String getSimplePercents(int money, float percent, int days){
        float result = (money * percent * days)/(365 * 100);
        return String.format("%.2f", result);
    }
```

Метод розрахунку складних процентів:

```
    public static String getDifficultPercents(int money, float percent, int
days){
        int numberOfOperation = days/30;
        int result = (int) (money * Math.pow((1 + (percent * days)/(100*365)),
numberOfOperation));
        return String.format("%.2f", result);
    }
```

7. Графік

Вікно з кольоровими, інтерактивними графіками (див. рис. 29). Можна переглянути як найперший депозит у рейтингу, так і останній. Є можливість приблизити або віддалити. Відображення рейтингу виконується за рахунок найвигіднішого депозиту в обраній валюті та терміну.

Рисунок 29 – Графік.

Рисунок 30 – Натиснення на шкалу.

При натисканні на будь-який депозит у верхньому графіку, з’явиться повідомлення з назвою банку та його процентною ставкою (див. рис. 30). Графік знизу використовується задля зручного перегляду рейтингу: збільшення, віддалення, переміщення.

3.3 Тестування

Протестуємо основний функціонал програми. Після запуску додатку, повинно з’явитися головне вікно програми з вибором валюти (див. рис. 31). Обираємо «USD». Програма автоматично переадресовує користувача за його запитом до потрібної таблиці. На рисунку №32 показано відповідні депозити, в раніше обраній валюті.

Рисунок 31 – Вибір валюти.

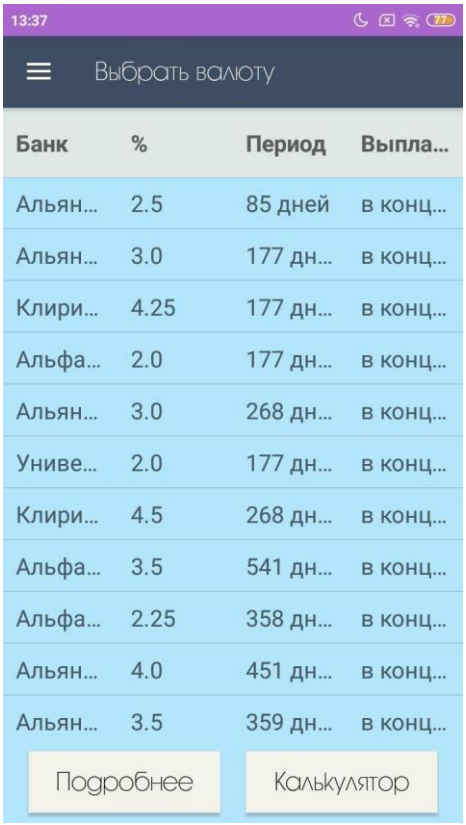

Рисунок 32 – «USD» валюта.

Наступним кроком, протестуємо пошукову систему. Задаємо параметри «Банк» - ПУМБ, валюта – EURO, процент від 0.5 до 0.6, решту залишимо порожніми (див. рис. 33) та натиснемо на кнопку «Поиск». Результат пошуку буде знаходитися в таблиці (див. рис. 34)

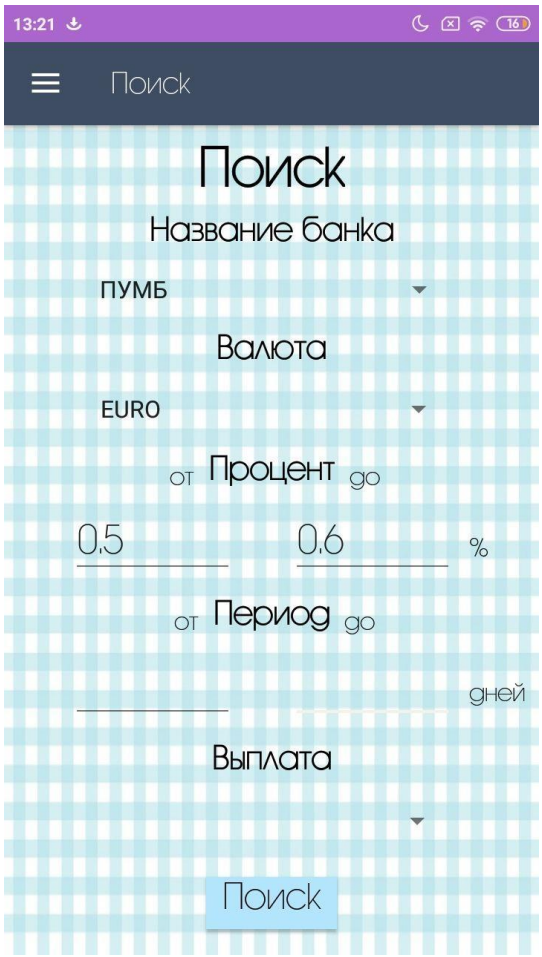

Рисунок 33 – Пошук.

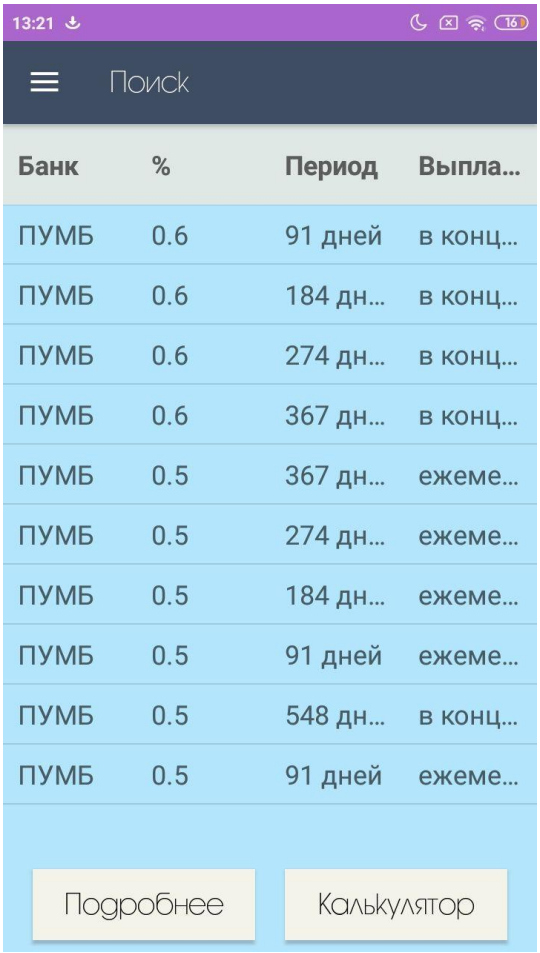

Рисунок 34 – Результат пошуку.

З таблиці необхідно обрати будь-який депозит з результатів пошуку або з будь-якої таблиці валют та натиснути на кнопку «Подробнее». Вікно з таблицею заміниться на вікно з детальною інформацією про даний депозит (див. рис. 36).

Після ознайомлення з детальними тарифами депозиту, наступним кроком є можливість порахувати прибуток від даного депозиту. Для цього потрібно лише натиснути на кнопку «Калькулятор». Потрібні дані з даного депозиту будуть вже заповнені у полях розрахунку. Користувачеві залишається лише вказати початковий внесок, у даний депозит (див. рис. 37).

Рисунок 36 – Депозит.

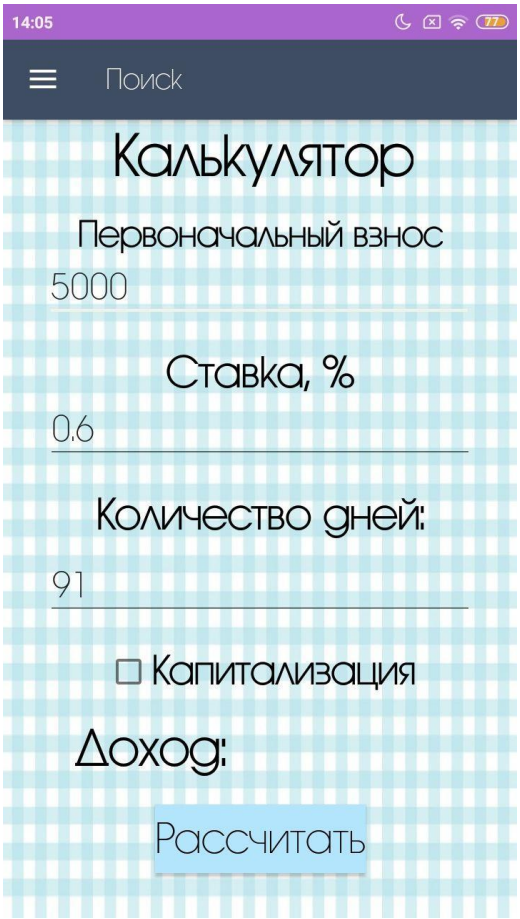

Рисунок 37 – Калькулятор депозиту.

Якщо користувач залишив хоча б одне поле порожнім, програма сповістить користувача що не всі поля заповнені (див. рис. 38). Заповнимо дані яких бракує, та натиснемо кнопку розрахувати (див. рис. 39).

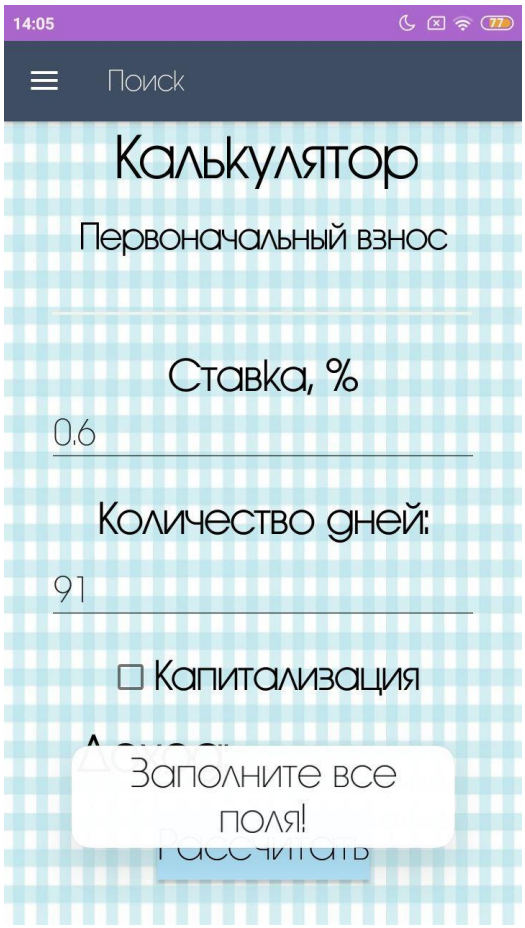

Рисунок 38 – Бракує даних.

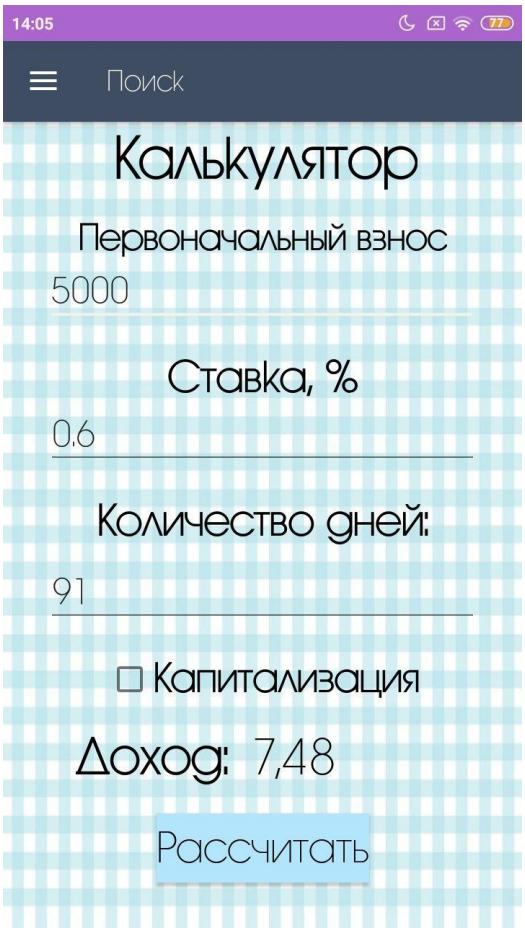

Рисунок 39 – Дохід.

Залишилось протестувати графіки додатку. Перейдемо до даного вікна з навігаційного меню. Та вкажемо іншу валюту та термін (див. рис. 40). Та на власний розсуд виберемо стовпчик з депозитом. З’явиться інформаційне повідомлення щодо даного депозиту (див. рис 41).

Рисунок 40 – Графік депозитів.

Рисунок 41 – Повідомлення.

ВИСНОВКИ

Робота присвячена проблемі відсутності зручного моніторингу усіх найбільших банків з наданням та розрахунку депозитів з будь-якими даними.

Рішення даної проблеми можна знайти на просторах інтернету, але вони існують незалежно одна від одної. Кожна виконує якусь малу частку та не є доскональним рішенням даної проблеми.

Рішення, що описується в даній роботі поєднує в собі все що потрібно для вирішення цієї проблеми:

- Актуальна база даних
- Зручний інтерфейс
- Можливість зробити розрахунок по обраному депозиту
- Можливість пошуку по введеним даним
- Можливість розрахувати власний депозит по введеним даним
- Рейтингова таблиця усіх найпопулярніших банків
- Діаграма депозитів для наглядного порівняння,

що полегшує пошук потрібного депозиту з величезної кількості доступних.

Для реалізації були використані бібліотеки: Selenium, GSON, JSOUP, HelloCharts. В результаті роботи було отримано інформаційну систему на смартфон, яка швидко працює при великій кількості інформації, проста в дизайні і зрозуміла для користувача. Застосунок повністю адаптивний до усіх сучасних мобільних пристроїв на базі Android. Додаток було успішно протестовано. Програма виконує усі перераховані функції справно та без помилок.

Список літератури:

1. Банковские депозиты (вклады) [Електронний ресурс]. – Режим доступу : http://investfunds.ua/banks/
2. Депозиты в банках Украины [Електронний ресурс]. – Режим доступу : https://minfin.com.ua/deposits/
3. Прохоренок Н. Основы Java /Н. Прохоренок – М. : БХВ-Петербург, 2016. с – 532.
4. Нимейер П. Программирование на Java П. Нимейер, Д. Леук – м.: Эксмо, 2015. С – 329.
5. Шилдт Г. Java 8. Полное руководство Шилдт Г., - м.: Диалектика-Вильямс, 2017. С – 972.
6. Формула расчета процентов по вкладам (депозитам) [Електронний ресурс]. – Режим доступу: https://bankirsha.com/formula-calculate-of-interest-on-deposit.html
7. Сила прикосновения, или Зачем мобильные приложения [Електронний ресурс]. – Режим доступу: http://agroportal.ua/views/blogs/sila-prikosnoveniya-ili-zachem-mobilnye-prilozheniya-agrariyam/
8. 5 преимуществ разработки мобильных приложений [Електронний ресурс]. – Режим доступу: https://spark.ru/startup/infoshell/blog/43721/5-preimuschestv-razrabotki-mobilnih-prilozhenij-dlya-biznesa

ДОДАТОК

Додаток А. Клас для роботи з браузером

```
public class Browser {
    private WebDriver driver;

    public void setDriver() {
        System.setProperty("webdriver.chrome.driver", "chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }

    public void openPage(String page) {
        driver.navigate().to(page);
        WebElement element = driver.findElement(
                By.xpath("//*[@class='sc-282uzu-1 fOORvs']"));
        do {
            try {
                element.click();
            } catch (StaleElementReferenceException | NoSuchElementException e) {
                element = null;
            }
        }
        while (element != null);
    }

    public String getDate(String page) {
        driver.get(page);
        return driver.findElement(By.xpath(

"/html/body/main/div/div/section/div/div/div[2]/div/div[1]/div[2]/span[2]/time"))
                .getText();
    }
```

```
    public ArrayList<Deposit> getDeposits(String currency) throws Exception {
        List<WebElement> table = driver.findElements(By.xpath(
"//*[@id=\"root\"]/div/section/div/div/div[2]/div/div[2]/table/tbody"));
        String tableString = table.get(0).getText();
        String[] substring = tableString.split("\n");

        ArrayList<Deposit> deposits = new ArrayList();
        for (int i = 0; i + 3 < substring.length; i += 4) {
            Pattern pattern = Pattern.compile(
                    "\\b(?<!\\.)(?!0+(?:\\.0+)?%)(?:\\d|[1-
9]\\d|100)(?:(?<!100)\\.\\d+)?%");
            Matcher matcher = pattern.matcher(substring[i + 1]);
            String percent = "";
            String depositName = "";
            if (matcher.find()) {
                depositName = substring[i + 1].substring(0, matcher.start());
                percent = matcher.group();
            }
            deposits.add(new Deposit(substring[i], depositName, currency,
                    Float.parseFloat(percent.replaceAll("%", "")),
                    substring[i + 2], substring[i + 3]));
        }
        return deposits;
    }
}
```

Додаток Б. Клас для роботи з депозитами

```
public class DepositManager {
    private FileManager fileManager;
    public static List<Deposit> uahDeposits;
    public static List<Deposit> usdDeposits;
    public static List<Deposit> euroDeposits;
    public static List<Deposit> allDeposits;

    public DepositManager() {
        fileManager = new FileManager();
    }

    public List<Deposit> getDeposits() {
        switch (GalleryFragment.currency) {
            case "UAH":
                return uahDeposits;
            case "USD":
                return usdDeposits;
            case "EURO":
                return euroDeposits;
        }
        return null;
    }

    public void fillLists(Activity activity) {
        if (uahDeposits == null)
            uahDeposits = fileManager.readGSON(activity, R.raw.uah);
        if (usdDeposits == null)
            usdDeposits = fileManager.readGSON(activity, R.raw.usd);
        if (euroDeposits == null)
            euroDeposits = fileManager.readGSON(activity, R.raw.euro);
        allDeposits = new ArrayList();
        allDeposits.addAll(uahDeposits);
        allDeposits.addAll(usdDeposits);
        allDeposits.addAll(euroDeposits);
    }

    public List<String> getBankNames() {
```

```
        Set<String> names = new HashSet<>();
        names.add("");
        for (Deposit deposit : allDeposits) {
            names.add(deposit.getBankName());
        }
        return new ArrayList<>(names);
    }

    public List<Deposit> searchDeposit(final String[] fields) {
        List<Deposit> currentDeposits = allDeposits;
        switch (fields[1]) {
            case "UAH":
                currentDeposits = uahDeposits;
                break;
            case "USD":
                currentDeposits = usdDeposits;
                break;
            case "EURO":
                currentDeposits = euroDeposits;
                break;
        }
        Stream<Deposit> stream = currentDeposits.stream();
        if (!fields[0].isEmpty())
            stream = stream.filter(deposit -> deposit.getBankName()
                    .equals(fields[0]));
        if (!fields[2].isEmpty() && !fields[3].isEmpty())
            stream = stream.filter(deposit -> (deposit.getPercent() >=
Float.parseFloat(fields[2])
                    && deposit.getPercent() <= Float.parseFloat(fields[3])));
        if (!fields[4].isEmpty() && !fields[5].isEmpty())
            stream = stream.filter(deposit -> (deposit.getDurationInt() >=
Integer.parseInt(fields[4])
                    && deposit.getDurationInt() <= Integer.parseInt(fields[5])));
        if (!fields[6].isEmpty())
            stream = stream.filter(deposit ->
deposit.getPayout().contains(fields[6]));
        ShareViewModel.search = stream.collect(Collectors.toList());
        return ShareViewModel.search;
    }
```

```
    public static Set<Integer> getAllUAHPeriods(){
        SortedSet<Integer> periods = new TreeSet<>();
        for (Deposit deposit : uahDeposits) {
            periods.add(deposit.getDurationInt());
        }
        return periods;
    }
    public static Set<Integer> getAllUSDPeriods(){
        SortedSet<Integer> periods = new TreeSet<>();
        for (Deposit deposit : usdDeposits) {
            periods.add(deposit.getDurationInt());
        }
        return periods;
    }
    public static Set<Integer> getAllEUROPeriods(){
        SortedSet<Integer> periods = new TreeSet<>();
        for (Deposit deposit : euroDeposits) {
            periods.add(deposit.getDurationInt());
        }
        return periods;
    }
}
```

Додаток В. Каркас об’єкта депозита

```
public class Deposit {

    private String bankName;

    private String depositName;

    private String currency;

    private float percent;

    private String duration;

    private String payout;

    public Deposit(String bankName, String depositName, String currency,

                   float percent, String duration, String payout) {

        this.bankName = bankName;

        this.depositName = depositName;

        this.currency = currency;

        this.percent = percent;

        this.duration = duration;

        this.payout = payout;

    }



    public String getBankName() {

        return bankName;

    }

    public String getDepositName() {

        return depositName;
```

```
    }

    public String getCurrency() {

        return currency;

    }

    public float getPercent() {

        return percent;

    }

    public String getDuration() {

        return duration;

    }

    public String getPayout() {

        return payout;

    }

}
```

Додаток Г. Клас для роботи з файлами

```
public class FileManager {
    private static Gson gson;

    public FileManager() {
        gson = new Gson();
    }

    public List<Deposit> readGSON(Activity activity, int f) {
        StringBuilder sb = new StringBuilder();
        try (InputStream is = activity.getResources().openRawResource(f);
             BufferedReader br = new BufferedReader(new InputStreamReader(is))) {
            String data = "";
            try {
                while ((data = br.readLine()) != null) {
                    sb.append(data);
                }
            } catch (IOException e) {
                System.out.println(e);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        List<Deposit> deposits = gson.fromJson(sb.toString(), new
TypeToken<List<Deposit>>() {
        }.getType());
        return deposits;
    }
}
```

Додаток Д. Головний клас мобільного доданку

```
public class MainActivity extends AppCompatActivity {
    private AppBarConfiguration mAppBarConfiguration;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.setItemIconTintList(null);
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
                R.id.nav_tools, R.id.nav_share, R.id.nav_send)
                .setDrawerLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);

        new DepositManager().fillLists(this);
    }

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

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
```

```
    public void onClickUAH(View view) {
        GalleryFragment.currency = "UAH";
        setDefaultFields();
        move(GalleryFragment.getInstance());
    }

    public void onClickUSD(View view) {
        GalleryFragment.currency = "USD";
        setDefaultFields();
        move(GalleryFragment.getInstance());
    }

    public void onClickEURO(View view) {
        GalleryFragment.currency = "EURO";
        setDefaultFields();
        move(GalleryFragment.getInstance());
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void search(View view) {
        String[] fields = SlideshowViewModel.getFields(SlideshowFragment.root);
        GalleryViewModel.fields = fields;
        GalleryFragment.clicked = false;
        move(GalleryFragment.getInstance());
    }

    private void move(Fragment fragment) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction transaction = fragmentManager.beginTransaction();
        transaction.setCustomAnimations(android.R.animator.fade_in,
                android.R.animator.fade_out).replace(R.id.nav_host_fragment,
fragment);
        transaction.commit();
    }

    public void calculate(View view) {
        ToolsFragment.calculate();
    }
```

```
    public void getMoreInfo(View view) {
        if (!GalleryFragment.clicked) {
            Toast toast = Toast.makeText(GalleryFragment.root.getContext(),
                    "Выберите депозит из списка",
                    Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.BOTTOM, 0, 0);
            toast.show();
        } else {
            move(new ShareFragment());
        }
    }

    public void goToCalc(View view) {
        if (!GalleryFragment.clicked) {
            Toast toast = Toast.makeText(GalleryFragment.root.getContext(),
                    "Выберите депозит из списка",
                    Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.BOTTOM, 0, 0);
            toast.show();
        } else {
            move(new ToolsFragment());
        }
    }

    private void setDefaultFields() {
        GalleryViewModel.fields = null;
        GalleryFragment.clicked = false;
        ShareViewModel.search = null;
    }
}
```

Додаток Е. Клас для відображення таблиці

```
public class GalleryViewModel extends ViewModel {
    private DepositManager manager = new DepositManager();
    public static String[] fields;
    private String[] headers = {"Банк", "%", "Период", "Выплата"};
    private String[][] column;

    public String[] getHeaders() {
        return headers;
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public String[][] fillTable() {
        List<Deposit> deposits;
        if (fields == null)
            deposits = manager.getDeposits();
        else
            deposits = manager.searchDeposit(fields);
        Deposit d;
        column = new String[deposits.size()][4];
        for (int i = 0; i < deposits.size(); i++) {
            d = deposits.get(i);
            column[i][0] = d.getBankName();
            column[i][1] = String.valueOf(d.getPercent());
            column[i][2] = d.getDuration();
            column[i][3] = d.getPayout();
        }
        return column;
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void printTable(View root) {
        TableView<String[]> tb = root.findViewById(R.id.tableView);
        tb.setColumnCount(4);

        tb.setHeaderBackgroundColor(Color.parseColor("#e0e8e5"));
        tb.setHeaderAdapter(new SimpleTableHeaderAdapter(root.getContext(),
getHeaders()));
        tb.setDataAdapter(new SimpleTableDataAdapter(root.getContext(),
```

```
fillTable()));


        tb.addDataClickListener((TableDataClickListener) (rowIndex, clickedData) -> {
            ShareViewModel.index = rowIndex;
            GalleryFragment.clicked = true;
        });
    }
}
```

Додаток Е. Клас, для розрахунків

```
public class ToolsFragment extends Fragment {
    private ToolsViewModel toolsViewModel;
    private static View root;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        toolsViewModel =
                ViewModelProviders.of(this).get(ToolsViewModel.class);
        root = inflater.inflate(R.layout.fragment_tools, container, false);

        if (GalleryFragment.clicked)
            fillFields(root);
        return root;
    }

    public static void calculate() {
        try {
            ToolsViewModel.calculate(root);
        } catch (NotEnoughDataException e) {
            Toast toast = Toast.makeText(root.getContext(),
                    "Заполните все поля!",
                    Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.BOTTOM, 0, 0);
            toast.show();
        }
    }

    private void fillFields(View view) {
        Deposit deposit = null;
        if (ShareViewModel.search != null)
            deposit = ShareViewModel.search.get(index);
        else {

            switch (GalleryFragment.currency) {
                case "UAH":
                    deposit = uahDeposits.get(index);
                    break;
                case "USD":
```

```
                deposit = usdDeposits.get(index);
                break;
            case "EURO":
                deposit = euroDeposits.get(index);
        }
    }

    EditText percentText = view.findViewById(R.id.percent);
    percentText.setText(String.valueOf(deposit.getPercent()));
    EditText daysText = view.findViewById(R.id.days);
    daysText.setText(Integer.toString(deposit.getDurationInt()));
    CheckBox checkBox = view.findViewById(R.id.checkBox);
    if (deposit.getPayout().contains("капитализация"))
        checkBox.setSelected(true);
  }
}
```