Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 6.050103 – Програмна інженерія

**ЗАТВЕРДЖУЮ**

Завідувач кафедри СПЗ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (В.В. Любченко)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

|  |
| --- |
| Буслюку Петру Сергійовичу |
| (прізвище, ім’я, по-батькові) |

1. Тема роботи: \_Клієнт-серверна програма розрахунку перспективного прибутку від інвестиційної діяльності

Керівник роботи: \_\_Паулін Олег Миколайович, доктор технічних наук, доцент\_ (прізвище, ім’я, по-батькові, науковий ступінь, вчене звання)

затверджені наказом ректора ОНПУ від «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_ року № \_\_\_\_\_\_\_\_\_

2. Строк подання студентом роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_у відповідності до технічного завдання на розробку\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Вступ. Функціональні вимоги до системи. План виконання проекту. Проектування системи. Програмна реалізація. Тестування системи. Висновки. Список використаних джерел.\_Лістинги програми\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

У відповідності зі слайдами електронної презентації\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів проекту

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів дипломної  роботи | Строк виконання етапів роботи | Примітка |
| 1 | Специфікація вимог | 05.01.15 – 16.01.15 | Виконано |
| 2 | Планування проекту | 17.01.15 – 20.02.15 | Виконано |
| 3 | Проектування проекту | 21.0.15 – 07.04.15 | Виконано |
| 4 | Реалізація проекту | 08.04.15 – 28.05.15 | Виконано |
| 5 | Тестування програми | 29.05.15 – 18.06.15 | Виконано |
| 6 | Введення в експлуатацію | 19.06.15 – 30.06.15 | Виконано |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Буслюк П.С.\_\_\_\_

(підпис) (прізвище та ініціали)

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Паулін О.М.\_\_\_\_

(підпис ) (прізвище та ініціали)

Реферат

Пояснювальна записка до дипломної роботи: 59 с., 12 рис., 7 табл., 27 джерел.

Метою роботи є підвищення ефективності обчислення даних перспективного прибутку, щодо інвестованих коштів в різні сфери інвестованої діяльності, спрощення створення звітів щодо інвестованих компаній та створення системи віддаленого збереження та відновлення даних клієнта.

Методи розробки базуються на технології Java, із використанням компонентів Swing для GUI, H2 базами даних які підтримують SQL-подібні запити, Java Mail бібліотеки для інформування клієнтів на email адреси щодо нових цікавих проектів, Itext бібліотеки для створення звітів у pdf-форматі, SoketIO для передачі та відновлення даних та JDOM-бібліотеки для відновлення даних із xml-файлу локально.

Як результат роботи виконана реалізація клієнт-серверної програми для розрахунку перспективного прибутку від інвестиційної діяльності .

Ключові слова: клієнт-серверна GUI програма, JDOM, java xml, sql, h2 java, pdf, java mail, socket io java, itext, swing java .

ABSTRACT

The aim is to improve the efficiency of data calculations prospective profits on invested funds invested in various spheres of activity, simplify reporting on the investees and creating a system of remote conservation and restoration of customer data.

The methods of making are based on the technology Java, using Swing components for GUI, H2 databases that support SQL-like queries, Java Mail library to inform clients on the email address on new projects, Itext library for creating reports in pdf-format, SoketIO for transmission and data recovery and JDOM-library to recover data from xml-file locally.

As a result of the implementation done client-server application to calculate prospective profit from the investment.

Keywords: client-server GUI program, JDOM, java xml, sql, h2 java, pdf, java mail, socket io java, itext, swing java.

Зміст

[Вступ 7](#_Toc423196038)

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

[1.1.1 Опис предметної області і аналогів системи](#_Toc423196040)

[1.1.2 Варіанти використання системи 12](#_Toc423196041)

1.1 Нефункціональні вимоги до системи

[2 План виконання проекту 18](#_Toc423196042)

[2.1 Оцінка тривалості розробки 18](#_Toc423196043)

[2.2 Оцінка ризиків 22](#_Toc423196044)

[2.3 Розробка плану робіт проекту 26](#_Toc423196045)

[3 Проектування системи 29](#_Toc423196046)

[3.1 Концептуальне проектування 29](#_Toc423196047)

[3.2 Проектування алгоритмів 37](#_Toc423196048)

[3.3 Проектування інтерфейсу користувача 40](#_Toc423196049)

[4 Програмна реалізація 43](#_Toc423196050)

[4.1 Опис програмних технологій 43](#_Toc423196051)

[4.2 Опис програмних бібліотек 43](#_Toc423196052)

[4.3 Особливості створення структур даних та програмних модулів 44](#_Toc423196053)

[5 Тестування Системи 51](#_Toc423196054)

[Висновки 63](#_Toc423196055)

[Список використаних джерел 65](#_Toc423196056)

Додаток А. Інструкція з встановлення та використання

Додаток Б. Лістинг програми

**Вступ**

Основними проблемами у сучасному світі для інвесторів є пошук нових систем для інвестування та правильно розподілений бюджет (менеджмент коштів), вистроювання планів щодо інвестування та правильне розуміння ризиків щодо інвестованих інструментів.

Загалом, якщо мова іде про інвестування в шахрайські проекти (хайпи/фінансові піраміди), то як ви розумієте повинен бути максимальний прибуток в найкоротший час, а ризики при цьому можуть складати до 100 відсотків від суми інвестування. Це зокрема, як на мій погляд, най ризиковані проекти, однак є багато прикладів, коли люди, які взнали про це першими, стали дуже впливовими.

Якщо розглядається фондова біржа, то тут є зокрема декілька факторів щоб залишитись із своїми коштами та примножити їх. Це і математичне очікування, коли у виграшній операції ви маєте потрійну перевагу ніж у збитковій, тож на дистанції ви будете у профіті, і розподілення депозиту балансу щодо вашого інвестиційного портфелю на рівні не більше 10 відсотків від усіх коштів, і максимальне просідання балансу депозиту на рівні в 1 відсоток у збитковій угоді тощо. Однак, можуть бути фактори, які не можна спрогнозувати, наприклад банкротство компанії, неуважність при створенні нового ордеру, навіть різка зміна напряму тренду може призвести до повного банкротства, коли брокер не може закрити ваш ордер по необхідній ціні, і автоматично ваш ордер відхиляється, або ж просто відсутність мережі Інтернет може суттєво наблизити депозит до нуля тощо.

Тобто над усіма цими факторами рано чи пізно задумується кожний інвестор, а для цього треба бути готовим. Дана ж дипломна робота розроблена для того, щоб уникнути марної роботи із розрахунку прибутку, який можливо отримає клієнт в разі, коли все буде добре, якщо інвестує свої кошти в будь-який проект та над тим, щоб надати клієнту важливу інформацію щодо нових, мало знайомих, але перспективних інвестиційних фондів, щоб клієнт ввівши один раз дані міг з легкістю оновити їх, або відтворити заново. Це і являється ціллю даного проекту.

Для реалізації вказаної цілі в роботі було створено програмне середовище, за допомогою якого, навіть без спеціального навчання, будь-яка мало знайома з комп’ютером людина зможе з легкістю обчислити та зберегти дані щодо інвестованих фінансових фондів, відновити дані локально або віддалено та отримати новини не тільки щодо нових пропозицій із інвестування, але й оновити свої знання щодо новин бізнесу, інвестицій, фінансів через систему інформування на email адресу тощо.

Зокрема ця система може застосовуватись лише в окремих колах людей, для яких час – це гроші, а новини – це ідеї для втілення нових можливостей.

Також завдяки цій реалізації люди зможуть вирішити принаймні три найважливіших проблеми сучасності, про які я розповів вище – керувати своїми фінансами, продумувати плани щодо інвестування та отримувати інформацію про нові системи для примноження свого капіталу.

**1 ВИМОГИ ДО СИСТЕМИ**

* 1. **Функціональні вимоги до системи**

У функціональних вимогах можуть зустрічатись не зовсім відомі усім позначення, тож для того, щоб прояснити ситуацію можна навести приклад словника термінів (тезаріусу) для наступних слів:

* Архітектура клієнт-сервер – передбачає взаємодію та обмін даними між вузлами. Вона передбачає такі основні компоненти:

1. набір [серверів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80), які надають інформацію або інші послуги програмам, які звертаються до них;
2. набір [клієнтів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), які використовують сервіси, що надаються серверами;
3. [мережа](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%83%D0%BD%D1%96%D0%BA%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0), яка забезпечує взаємодію між клієнтами та серверами.

* Авторизація – керування рівнями та засобами [доступу](https://uk.wikipedia.org/w/index.php?title=%D0%94%D0%BE%D1%81%D1%82%D1%83%D0%BF&action=edit&redlink=1) до певного захищеного ресурсу, як в фізичному розумінні (доступ до кімнати готелю за карткою), так і в галузі цифрових технологій (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від [ідентифікатора](https://uk.wikipedia.org/wiki/%D0%86%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%82%D0%BE%D1%80) і [пароля](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BE%D0%BB%D1%8C) користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних.
* База даних ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) database) – це сукупність [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96_(%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%82%D0%B5%D1%85%D0%BD%D1%96%D0%BA%D0%B0)), організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96). Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.
* XML – розширювана мова розмітки ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Extensible Markup Language) запропонований консорціумом World Wide Web ([W3C](https://uk.wikipedia.org/wiki/W3C)) стандарт побудови [мов розмітки](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%82%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) ієрархічно структурованих [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96) для обміну між різними [застосунками](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), зокрема, через [Інтернет](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Є спрощеною підмножиною мови розмітки [SGML](https://uk.wikipedia.org/wiki/SGML). XML-документ складається із текстових знаків, і придатний до читання людиною.
* Інтернет (від [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Internet), міжмережжя – всесвітня система взаємосполучених [комп'ютерних мереж](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%83%D0%BD%D1%96%D0%BA%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0), що базуються на [комплекті Інтернет-протоколів](https://uk.wikipedia.org/wiki/TCP/IP). Інтернет також називають мережею мереж. Інтернет складається з мільйонів локальних і глобальних приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових,[оптичних](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%BE%D0%B2%D0%BE%D0%BB%D0%BE%D0%BA%D0%BD%D0%BE) і [бездротових технологій](https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%B4%D1%80%D0%BE%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0). Інтернет становить фізичну основу для розміщення величезної кількості інформаційних ресурсів і послуг, таких як взаємопов'язані гіпертекстові документи [Всесвітньої павутини](https://uk.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D1%81%D0%B2%D1%96%D1%82%D0%BD%D1%8F_%D0%BF%D0%B0%D0%B2%D1%83%D1%82%D0%B8%D0%BD%D0%B0) (World Wide Web – WWW) та [електронна пошта](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BF%D0%BE%D1%88%D1%82%D0%B0).
* Електронна пошта ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) e-mail, або email, скорочення від electronic mail) – популярний [сервіс](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D1%81%D0%B5%D1%80%D0%B2%D1%96%D1%81) в [інтернеті](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), що робить можливим обмін даними будь-якого змісту ([текстові документи](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), [аудіо](https://uk.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D1%96%D0%BE%D1%84%D0%B0%D0%B9%D0%BB)-[відеофайли](https://uk.wikipedia.org/w/index.php?title=%D0%92%D1%96%D0%B4%D0%B5%D0%BE%D1%84%D0%B0%D0%B9%D0%BB&action=edit&redlink=1), [архіви](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D0%B2_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [програми](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0)).

Тож, ознайомившись із даними словами наведемо приклад багаторівневої ієрархії системи функціональних вимог, зображеної на рис. 1.1 де показано WBS структуру даних функціональних вимог.

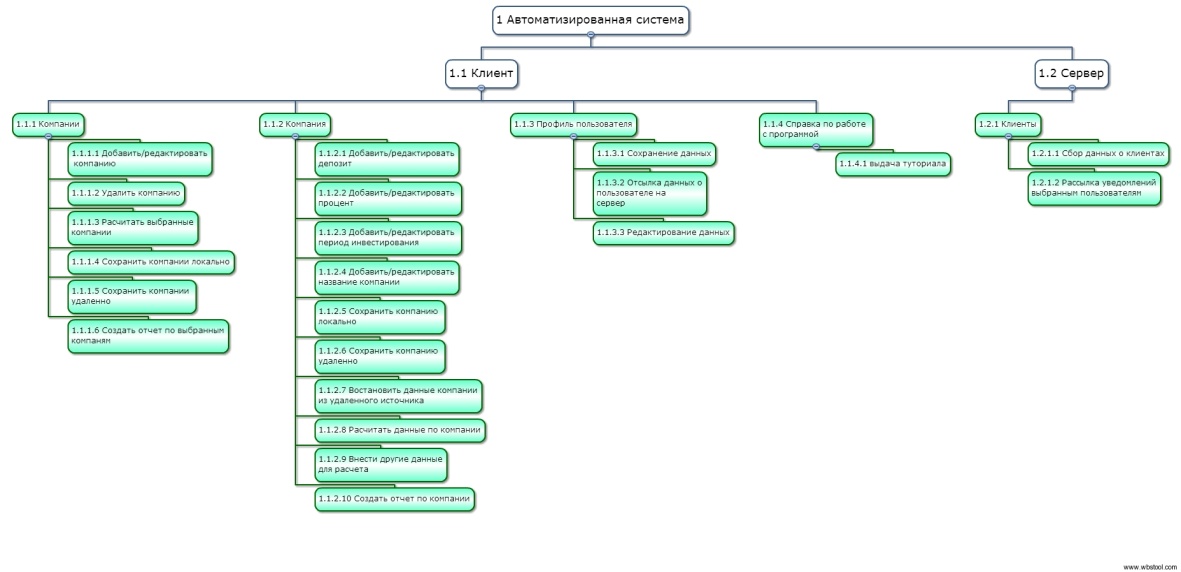


Рисунок 1.1 – Багаторівнева ієрархія функціональних вимог

Можна виключити такі вимоги, адже вони зберігаються у вимогах «Компаній»:

1. 1.1.2.5 – зберегти компанію локально;
2. 1.1.2.6 – зберегти компанію віддалено;
3. 1.1.2.7 – відновити данні з віддаленого джерела.

Визначення критеріїв за типом MoSCoW: усі вимоги мають тип M, однак можна додати ще пару вимог типу W – додати оновлення програми та ввести оплату за ПЗ, де тип М – повинні це зробити, W – не буде достатньо часу на розробку, але в майбутньому можливо буде розроблено.

* + 1. **Опис предметної області і аналогів системи.** Перш ніж приступати до розробки треба проаналізувати вже існуючі продукти на наявність функціональності яка в них присутня, і якщо необхідна функціональність відсутня, то має сенс на розробку власного продукту.

Серед знайдених в просторах Інтернету знайшлось лише декілька прикладів схожого функціоналу. Тож, можна виявити більш-менш підходящі до нашої теми дипломної роботи вже існуючі програмні продукти. Лише одна із наведеного нижче списку система являється для користування на комп’ютері (desktop), інші ж являються онлайн програмами, які розраховують дані на стороні клієнта в браузері.

Існуючі програмні продукти:

1. <http://invite-invest.ru/calculator-investitsii.html> (invest-invest.ru);
2. <http://soft.delovar.info/> (delovar.info);
3. [http://www.denega.ru/portcalc162.zip](http://mmgp.ru/redirect/away.php?url=http%3A%2F%2Fwww.denega.ru%2Fportcalc162.zip) (denega.ru);
4. [http://www.firmasoft.ru/hide/finans/clear.htm (firmasoft.ru](http://www.firmasoft.ru/hide/finans/clear.htm%20-%20firmasoft.ru)).

Можна побачити, що даних програмних продуктів не так вже й багато, сюди можна було також включити Microsoft Excel, але в ньому необхідно знати основи програмування, а це не дуже підходить під сучасні умови, коли більшість людей неспроможні навіть видалити вірус чи опановувати нові, незнайомі для них, технології, втрачаючи на це купу часу та нервів.

Тож, наведемо список проблем які зустрічаються в даних програмних продуктах та проаналізуємо, чи варто проектувати та розробляти наше програмне забезпечення (табл. 1.1).

Таблиця 1.1 - Таблиця рішень проблеми:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Продукти | Збереження даних віддалено | Розрахунок більше 1 компанії | Можливість довкладення/знімання суми | Відновлення даних |
| investinvest.ru | **–** | **–** | + | **–** |
| delovar.info | **–** | **–** | + | **–** |
| denega.ru | **–** | **–** | **–** | + |
| firmasoft.ru | **–** | **–** | **–** | **–** |

Як можна побачити, лише в декількох із них присутні лише часткові функції які необхідні для повноцінного функціоналу, із цього можна зробити висновок, що систему необхідно спроектувати та розробити із встановленими для цього функціональними вимогами.

* + 1. **Варіанти використання.** Проектування системи починається із початкового осмислення важливих функціональних елементів, які будуть в подальшому присутні в системі.

Тож, для цього існують так звані Use Case діаграми (діаграми варіантів використання/діаграми прецедентів), за допомогою яких на мові UML (уніфікованій мові моделювання) зображуються актори та взаємозв’язки між прецедентами (основними функціями системи).

Зокрема на діаграмах варіантів використання зображують систему яку розроблюють у вигляді прямокутника, де зверху пишуть назву системи, акторів чи сервісів (у вигляді людини), безпосередньо прецедентів та взаємозв’язків між акторами чи сервісами та прецедентами.

Структуру змодельованої системи можна переглянути нижче на рис. 1.2 – діаграмі прецедентів для системи розрахунку перспективного прибутку.

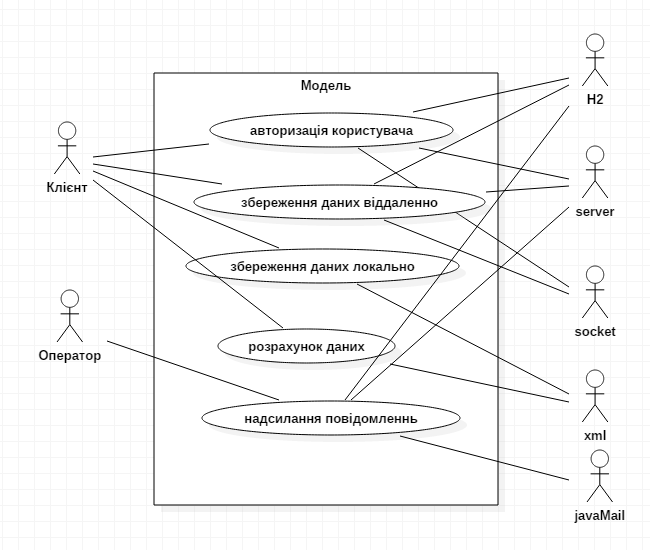


Рисунок 1.2 – Діаграма прецедентів системи

Згідно рис. 1.2 можна виділити такі типи акторів та сервіси в системі:

1. Клієнт – кінцевий користувач;
2. Оператор ­­– кінцевий користувач;
3. SocketIO – система взаємодії по протоколу TCP / IP;
4. H2 – БД, взаємодіюча по протоколу TCP / IP;
5. JavaMail ­– бібліотека, взаємодіюча по протоколу TCP / IP;
6. Xml-сховище локальних даних.

Для того щоб зробити детальніший план для розробки, необхідно описати яка послідовність дій буде в кожному прецеденті, тож наведемо це в табл. 1.2 – із сценаріями використання для кожного із них.

Таблиця 1.2 – Сценарії використання

|  |  |  |
| --- | --- | --- |
| Варіант використання №\_\_ : «Назва прецеденту» | Дії та суб’єкти | Опис дій |
| Варіант використання №1:  «Авторизація користувача» | Передумова | К заповнив усі необхідні поля профілю |
| Післяумова | Сервер дав згоду на користування системою |
| Суб'єкт | Користувач(К), Сервер(С), Система Користувача (СК) |
| Основний потік | 1. К входить до програми та заповнює профіль. СК пропонує збереження даних профілю. 2. СК відправляє дані авторизації на С. 3. С шукає в БД дані К та відсилає відповідь. |
| Альтернативний потік | 1а.1 К не згоджується із збереженням даних. СК не зберігає дані. Вихід з програми.  1б.1 К вже заповнив дані. Перехід до 2 пункту.  2а.1 С не знайшла К в БД. С створює К в БД та надає доступ. |
| Варіант використання №2:  «Збереження локальних даних локально» | Передумова | К авторизований у СК; К вказав персональні дані. |
| Післяумова | Дані збережені |
| Суб'єкт | Користувач(К), Система Клієнта(СК) |
| Основний потік | 1. К обирає збереження даних. СК перевіряє дані та зберігає їх у XML файл |
| Альтернативний потік | 1а.1 Дані не коректні. СК видає повідомлення щодо невірних даних |
| Варіант використання №3:  «Збереження даних віддалено» | Передумова | К авторизований у СК; у К є доступ до Інтернет мережі; К вказав у персональні дані. |
| Післяумова | Обліковий запис збережений. |
| Суб'єкт | Користувач(К), Система Клієнта(СК), Сервер (С) |
| Основний потік | 1. К обирає збереження даних віддалено. СК перевіряє дані та відсилає їх на С. 2. С отримує їх та зберігає в БД. |
| Альтернативний потік | 1а.1 К вказав невірні дані. СК пропонує ввести дані ще раз. |
| Варіант використання №4:  «Розрахунок даних» | Передумова | К авторизований у СК; Дані заповнені; Дані вірні; |
| Післяумова | Дані розраховані |
| Суб'єкт | Користувач(К), Система Клієнта(СК) |
| Основний потік | 1. К обирає компанії для розрахунку. СК розраховує дані та видає результат. |
| Альтернативний потік |  |
| Варіант використання №5:  «Надіслання повідомлення» | Передумова | Є доступ до БД та мережі Iнтернет. |
| Післяумова | Повідомлення відіслані вказаним адресатам. |
| Користувачі | Сервер(С), БД, Оператор(О), Клієнт(К) |
| Основний потік | 1. О обирає К для оповіщення та вводить саме повідомлення та назву заголовку. С встановлює email адресу кожного К та надсилає повідомлення. |
| Альтернативний потік | 1а.1 Невірна адреса. С пропускає К. |

* 1. **Нефункціональні вимоги до системи**

Нефункціональні вимоги характеризують властивості реалізації, які потрібно використовувати при безпосередній реалізації системи. Тож, можна описати спершу зовнішні інтерфейси.

Зовнішні інтерфейси системи: інтерфейс повинен бути інтуїтивно зрозумілим, із однаковим оформленням на усіх формах, тож було розроблено однотипний інтерфейс для усіх форм. Також використовуються компоненти меню та таблиць для надання повноцінної інформації, розбитої на певні блоки із мінімальною затратою місця на формах для кращого зовнішнього вигляду.

Одночасно із цим, були внесені зміни в наступні стилі оформлення: шрифт – Segoe UI, назви кнопок та елементів на формах – російською мовою, задній фон: сірий (RGB(240,240,240)).

За управління відповідають стандартні елементи swing бібліотеки (для графічного зображення елементів на моніторі), яка надається разом із jdk (набором компонентів для розробки). Також налаштовані гарячі клавіші:

1. F1 – довідка;
2. Ctrl+R- Формування звіту;
3. Ctrl+A – вибір усіх компаній;
4. Alt+C – розрахунок даних обраних компаній;
5. Ctrl+N – створення нової компанії;
6. Ctrl+D – видалення обраної компанії;
7. Ctrl+Alt+S – збереження компаній віддалено;
8. Ctrl+Alt+R – відновлення даних з серверу;
9. Alt+N – редагування назви компанії;
10. Alt+D – редагування депозиту;
11. Alt+P – редагування відсотку;
12. Alt+I – редагування тривалості інвестування.

Відображення повідомлень відбувається в стандартних MessageBox операційної системи.

При користуванні системою необхідно, щоб версія операційної системи була від Windows – XP SP1 та вище.

Також, для повноцінної роботи програми необхідна обов’язкова наявність JDK версії 8u77 та вище.

Інтерфейси передачі даних між клієнтом та сервером: Wifi або Ethernet.

Зокрема можна виділити основні нефункціональні вимоги, а саме:

Продуктивність:

* максимальний час відгуку програми на запит до 2 секунд;
* максимальна кількість обслуговуваних клієнтів одночасно до 2000.

Надійність:

* мінімальний час безвідмовної роботи 10 хвилин;
* також передбачено відновлення даних після збоїв у системі.

Технології розробки:

* ОС: Windows 7, Windows 10;
* СУБД: H2;
* програмні технології: desktop;
* структури даних: xml, files, db.

Інструменти розробки:

* IDE(платформа для розробки): NetBeans;
* збирач програмних модулів: JDK версія 8u77;
* VCS (система контролю версій): Git;
* бібліотеки модульного тестування: jUnit4;
* автоматизоване тестування: Jubula.

1. **ПЛАН ВИКОНАННЯ ПРОЕКТУ**
   1. **Оцінка тривалості розробки**

Серед методологій оцінок тривалості проекту є SLOC (кількість строк коду) не підходяща за декількома факторами та FP (функціональні точки), за допомогою яких ми спроектували нашу систему.

UCP метод оцінки об’єму виконуваних робіт входить до методології функціональних точок, тож, виділимо наступні показники для оцінки тривалості нашого продукту. В табл. 2.1 приведено визначення типів акторів і підрахована кількісна вага акторів (UAW).

Таблиця 2.1 – Визначення типів акторів

|  |  |  |  |
| --- | --- | --- | --- |
| Тип актора | Вага(*Кi*) | Примітка | Актори(*Vi*) |
| Простий | 1 | Представляє іншу систему, взаємодіючу по зумовленій API (REST, SOAP, dll). | 1 |
| Середній | 2 | Представляє іншу систему, що взаємодіє по протоколу типу TCP / IP. | 3 |
| Складний | 3 | Кінцевий користувач, що взаємодіє через GUI або Webpage | 2 |
| UAW = | | | 13 |

Кількісна вага акторів (UAW) визначається за формулою 2.1:

, (2.1)

де *Кi* – вага актора,

*Vi* – кількість акторів.

Далі необхідно визначити кількісну складність варіантів використання (UUCW) за формулою 2.2 із даних, представлених в табл. 2.2, де визначено вагу акторів та кількість варіантів використання (див. рис. 1.2).

Таблиця 2.2 – Визначення типів варіантів використання за кількістю транзакцій

|  |  |  |  |
| --- | --- | --- | --- |
| Тип варіанту використання | Вага(*VAi*) | Кількість транзакцій | Кількість варіантів використання(*KAi*) |
| Простий | 5 | До 4 | 2 |
| Середній | 10 | Від 4 до 7 | 2 |
| Складний | 15 | Від 7 | 1 |
| Розрахунок кількісної складності варіантів використання (UUCW) | | | 45 |

Розрахована UUCW за формулою 2.2 складає 45.

, (2.2)

де *KAi* – кількість варіантів використання,

*VAi* – вага кожного прецеденту.

Наступним кроком для визначення *UCP* значення являється сумування отриманих раніше значень *UUCW* та *UAW.* Тож, *UCP* становить 58 одиниць.

Однак, необхідно встановити ще часову характеристику значень варіантів використання (Hours Of Effort Per Use Case Point). А для цього необхідно визначити спочатку технічні фактори системи. Представимо дані в табл. 2.3, де приведемо технічні фактори системи.

Таблиця 2.3 – Технічні фактори

| Фактор | Опис | Вага(*VSi*) | Пояснення | Оцінка(KSi) |
| --- | --- | --- | --- | --- |
| T1 | Розподіленість системи | 2 | Інформує про потребу системи в розподілених обчисленнях | 1 |
| T2 | Час відгуку | 1 | Визначає ефективність системи з точки зору часу відгуку, потоку робіт тощо | 4 |
| T3 | Ефективність кінцевого користувача | 1 | Визначає ефективність користувача з точки зору його (її) сприйняття | 5 |
| T4 | Складність обробки | 1 | Визначає, чи будуть застосовуватись складні алгоритми для обробки даних | 5 |
| T5 | Фокус на повторному використанні коду | 1 | Визначає, чи будуть елементи коду системи використовуватись знову | 2 |
| T6 | Простота інсталяції | 0,5 | Визначає метод інсталяції та простоту інсталяції для кінцевого користувача, чи буде потреба в спеціалісті для інсталяції системи | 3 |
| T7 | Простота використання | 0,5 | Визначає узгодженість інтерфейсу користувача з його потребами | 4 |
| T8 | Портативність | 2 | Визначає, чи має застосування працювати в різних середовищах | 3 |
| T9 | Простота змін | 1 | Визначає, чи будуватиметься система в такий спосіб, щоб спростити її модифікації в майбутньому | 3 |
| T10 | Паралельні обчислення | 1 | Інформує, чи будуть мати в системі місце паралельні обчислення | 0 |
| T11 | Засоби захисту | 1 | Визначає чи потребуватиме система спеціальні засоби захисту даних чи системи | 2 |
| T12 | Доступ до третьої сторони | 1 | Визначає ступень використання системи зовнішніми системами або акторами | 2 |
| T13 | Потреби в спеціальному навчанні | 1 | Визначає, чи потрібно організувати тренінги для користувачів | 1 |
| Технічна факторна складність (TCF) | | | | 0,955 |

Технічна факторна складність являється одним із основних елементів оцінки тривалості проекту, тому вона розраховується за формулою 2.3 і складає 0,955. Це означає, система не дуже легка в розробці.

, (2.3)

де *KSi* – оцінка фактору,

*VSi* – вага фактору.

Після визначення технічних факторів обчислимо зовнішні фактори за формулою 2.4, дані для якої представлені в табл. 2.4.

Таблиця 2.4 – Оцінка зовнішніх факторів

|  |  |  |  |
| --- | --- | --- | --- |
| Фактор | Опис | Коефіцієнт(KEi) | Значення (VEi) |
| T1 | Знайомство з процесом  Розробки | 1,5 | 4 |
| T2 | Досвід подібних проектів | 0,5 | 5 |
| T3 | Досвід об’єктно-орієнтованої розробки | 1 | 5 |
| T4 | Досвідченість провідного  Аналітика | 0,5 | 4 |
| T5 | Мотивація | 1 | 5 |
| T6 | Стабільність вимог | 2 | 5 |
| T7 | Часткова зайнятість  Працівників | -1 | 3 |
| T8 | Складність мови програмування | -1 | 2 |
| Зовнішній фактор (EF) | | | 0,635 |

Сама ж формула для розрахунку зовнішніх факторів представлена нижче:

, (2.4)

де *KEi* – оцінка зовнішнього фактору,

*VEi* – вага зовнішнього фактору.

Зовнішні фактори показують, наскільки сприятливі умови для розробки системи, в нашому випадку зовнішній фактор EF складає 0,635.

Далі необхідно визначити скоректовану UCP (AUCP), для цього необхідно перемножити між собою раніше знайдені значення UCP, TCF та EF. Тож, AUCP, яка показує кількісну вагу із урахуванням різних факторів складає 35,17.

Зараз визначимо загальну тривалість проекту за HOEPUCP, спершу встановивши, що більшість значень T1-T8 перевищують 3, тому одному АUCP відповідає 35 робочих годин, тож помноживши раніше знайдене значення скоректованої UCP на кількість робочих годин ми отримаємо 1230,95 робочих годин.

Однак, для більш наглядної оцінки тривалості операцій можна скористатись діаграмою Гантта, розробленою Генрі Л. Ганттом в 1990 році, яка ілюструє план/графік з тривалості будь-якого проекту. Тривалість проекту зображено на рис. 2.1.

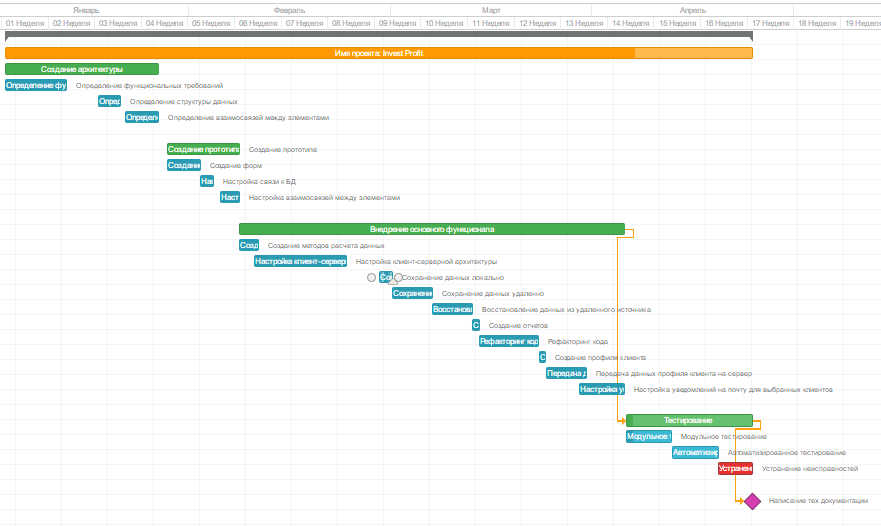


Рисунок 2.1 – Діаграма Гантта

Із рис.2.1 випливає, що на розробку та впровадження системи необхідно приблизно 16 тижнів роботи по 8-10 годин на добу, що складає приблизно 807 годин роботи без урахування написання технічної документації. Із урахуванням документації до проекту знадобиться приблизно 830 годин.

* 1. **Оцінка ризиків**
  2. **Розробка плану робіт проекту**

Ієрархічна структура робіт (ІСР) (англ. Work Breakdown Structure, WBS; іноді Структура декомпозиції робіт, СДР) - це ієрархічне розбиття всієї роботи, яку необхідно виконати для досягнення цілей проекту, на більш дрібні операції і дії до такого рівня, на якому способи виконання цих дій цілком зрозумілі і відповідні роботи можуть бути оцінені і сплановані. Вона включає також визначення проміжних результатів всіх складових структури робіт.

Структура декомпозицій робіт представлена на рис 2.2.

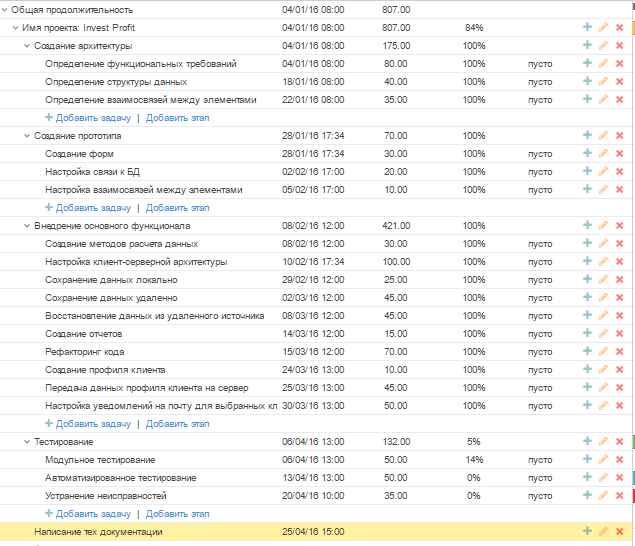


Рисунок 2.2 – WBS структура проекту

Тож, зважаючи на те, що було описано раніше, про тривалість проекту, загальна тривалість проекту в межах допустимих норм, і найбільш довші етапи це розробка безпосередньо самого функціоналу, розроблення архітектури та тестування.

1. **ПРОЕКТУВАННЯ СИСТЕМИ**
   1. **Концептуальне проектування**

В концептуальному проектуванні безпосередньо необхідно розробити логіку системи, необхідно продумати усі елементи системи та взаємозв’язок між ними.

Ґрунтуючись на досвіді, можу лише сказати, що розробляти правильну логіку може лише людина, яка має певний досвід у створенні подібних систем, включно із особливостями мови програмування та інших факторів, які існують.

Загалом, щоб показати логіку системи використовують UML Діаграми концептуальних класів, наприклад для нашої системи ця діаграма зображена на рис. 3.1.

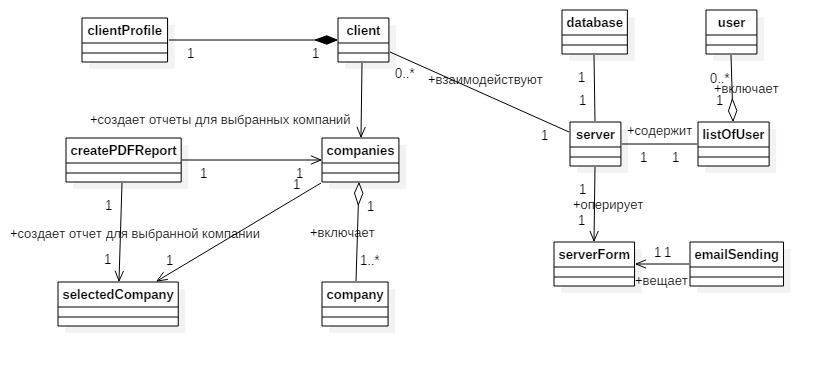


Рисунок 3.1 – Діаграма концептуальних класів

Із рис. 3.1 можна явно виявити, що структура розбивається на 2 пакети – клієнтську та серверну частину, в якій присутні необхідні класи як для однієї так і для іншої частини.

* 1. **Проектування алгоритмів**

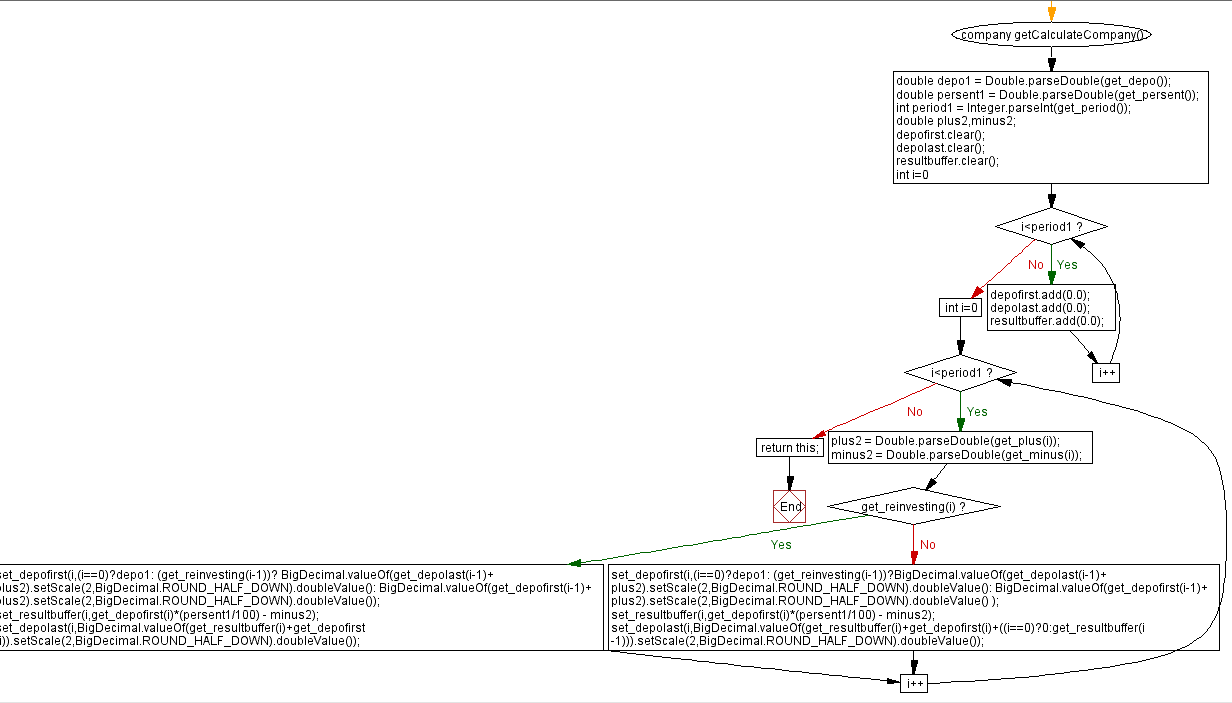
Діаграма програмних класів:

## C:\Diplom\report\диаграма програмных класов.jpg

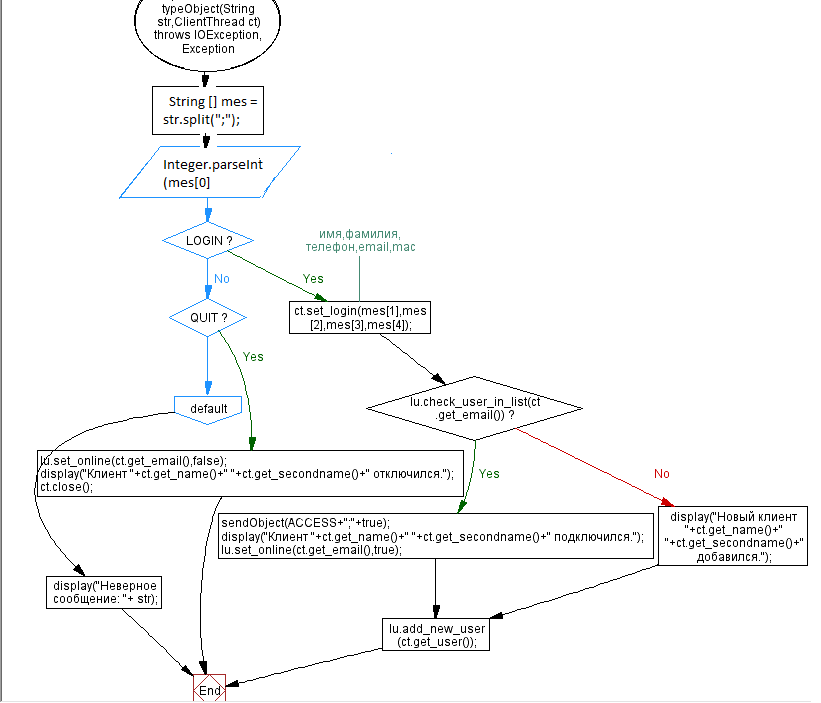
Рисунок 3.2 – Діаграма програмних класів

Блок-схема двох важливих для проекту функцій:

1 Блок-схема алгоритму розрахунку даних (блок-схема автоматично створена в програмі на основі коду програми, надалі блок-схема буде перерисована):



2 Блок-схема обміну даних між клієнтом та сервером. Серверна частина. (блок-схема автоматично створена в програмі, надалі блок-схема буде перерисована):



Оцінка може бути оцінена як часом (часова складність) так і об’ємом пам’яті (ємкісна складність).

Оцінка складності кожного елементарного блоку:

Оператори введення/виведення , проста умова - О (1)

Складна умова - O (m), m - кількість полів.

Оцінка алгоритмічної складності циклів:

Лінійний час - О (n), де n – кількість операторів .

Загальна оцінка алгоритмічної складності :

По першому алгоритму буде квадратична складність, так як в функції присутній один цикл, по якому розраховуються певні оператори, після виконання необхідних умов.

По другому – така сама складність і з тим же описом.

Оцінка алгоритмічної складності доступу до даних:

В даних алгоритмах не використовуються зчитування даних із xml-файлів та БД, бо це зроблено раніше і дані вже завантажені в оперативну пам'ять комп’ютера.

* 1. Проектування інтерфейсу користувача

Зовнішні інтерфейси: Інтерфейс інтуїтивно зрозумілим, із однаковим оформленням на усіх формах. Також використовувались компоненти меню та таблиць для надання повноцінної інформації, розбитої на певні блоки із мінімальною затратою місця на формах для кращого зовнішнього вигляду. Тож, будуть використовуватись наступні стилі оформлення: шрифт – Segoe UI, назви кнопок - російською мовою, задній фон: сірий (RGB(240,240,240)).

За управління відповідать наступні компоненти: стандартні елементи swing. Також налаштовані гарячі клавіші: F1 – довідка,Ctrl+R- Формування звіту , Ctrl+A – вибір усіх компаній,Alt+C – розрахунок даних обраних компаній, Ctrl+N – створення нової компанії,Ctrl+D – видалення обраної компанії,Ctrl+Alt+S – збереження компаній віддалено, Ctrl+Alt+R – відновлення даних з серверу, Alt+N – редагування назви компанії, Alt+D– редагування депозиту, Alt+P – редагування відсотку, Alt+I – редагування тривалості інвестування. Відображення повідомлень відбувається в стандартних MessageBox ОС.

Також був застосований золотий перетин – це співвідношення, найвідповідніше для

[естетичного](https://uk.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%82%D0%B8%D0%BA%D0%B0) сприйняття зображення, вперше запропоноване давньогрецьким математиком [Евклідом](https://uk.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D1%96%D0%B4).

Програмні інтерфейси: робота в OS: Windows - XP SP1 та вище.

Інтерфейси передачі даних між клієнтом та сервером: Wifi або Ethernet.

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ**
   1. **Опис програмних технологій**

Опис програмних технологій включає в себе опис усіх ПЗ, які використовувались або потрібні для програмної реалізації проекту.

Тож, зокрема, використовувались такі програмні технології:

NetBeans IDE – [вільне](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) [інтегроване середовище розробки](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) (IDE) для [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [Java](https://uk.wikipedia.org/wiki/Java), [JavaFX](https://uk.wikipedia.org/wiki/JavaFX), [C](https://uk.wikipedia.org/wiki/%D0%A1_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))/[C++](https://uk.wikipedia.org/wiki/C%2B%2B), [PHP](https://uk.wikipedia.org/wiki/PHP), [JavaScript](https://uk.wikipedia.org/wiki/JavaScript), [HTML5](https://uk.wikipedia.org/wiki/HTML5), [Python](https://uk.wikipedia.org/wiki/Python), [Groovy](https://uk.wikipedia.org/wiki/Groovy). Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за замовчуванням підтримує розробку для платформ [J2SE](https://uk.wikipedia.org/w/index.php?title=J2SE&action=edit&redlink=1) і [J2EE](https://uk.wikipedia.org/wiki/J2EE).

Поширюється у open source code під ліцензіями [GPLv2](https://uk.wikipedia.org/wiki/GPLv2) і [CDDL](https://uk.wikipedia.org/w/index.php?title=CDDL&action=edit&redlink=1). Проект NetBeans IDE підтримувався і спонсорувався фірмою [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems) і після придбання Sun – [Oracle](https://uk.wikipedia.org/wiki/Oracle_Corporation), проте розробка NetBeans ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

NetBeans IDE доступна для платформ [Microsoft Windows](https://uk.wikipedia.org/wiki/Microsoft_Windows), [GNU](https://uk.wikipedia.org/wiki/GNU)/[Linux](https://uk.wikipedia.org/wiki/Linux), [FreeBSD](https://uk.wikipedia.org/wiki/FreeBSD) і [Solaris](https://uk.wikipedia.org/wiki/Solaris_(%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)). Для інших платформ доступна можливість зібрати NetBeans самостійно із open source code.

За якістю і можливостям останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи [рефакторинг](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), [профілювання](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D1%96%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%96_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8)), виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду тощо.

H2 – відкрита кросплатформна СУБД, повністю написана на мові Java.

Незважаючи на малий розмір (трохи більше 1 МБ) H2 підтримує такі можливості:

* два режиму роботи (клієнт-сервер, вбудований);
* два режиму зберігання даних (файлова система, пам'ять);
* підтримка планів виконання запитів;
* підтримка кластеризації і реплікації;
* шифрування даних;
* зовнішні (пов'язані) таблиці;
* драйвер odbc;
* повнотекстовий пошук;
* визначення доменів;
* мультіверсійний конкурентний доступ;
* підтримка послідовностей;
* підтримка ключових слів limit і offset в запитах;
* тимчасові таблиці;
* обчислювані стовпці;
* призначені для користувача агрегатні функції;
* призначені для користувача процедури стиснення clob / blob об'єктів
* робота з csv файлами на читання і запис;
* браузерна консоль управління;
* запуск як сервіс windows.

Що ще не реалізовано:

* віконні функції (реалізована тільки функція row\_number() та over ());
* багатопоточна обробка запитів;
* повна підтримка стандарту sql 2003.

Застосовується H2, наприклад, в проекті Grails як базова СУБД для розробки.

Java Сокети ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Socket – заглиблення, гніздо, роз’їм) – назва [програмного інтерфейсу](https://uk.wikipedia.org/wiki/API) для забезпечення обміну даними між [процесами](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Процеси при такому обміні можуть виконуватися як на одній ЕОМ, так і на різних ЕОМ, пов'язаних між собою [мережею](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0). Сокет – абстрактний об'єкт, що представляє кінцеву точку з'єднання.

* 1. **Опис програмних бібліотек**

JavaMail – це Java API, призначене для отримання і відправки електронної пошти з використанням протоколів SMTP, POP3 та IMAP. JavaMail є частиною платформи Java EE, але також доступний в якості додаткового пакета для використання в додатках Java SE.

Існує також альтернативна реалізація JavaMail з відкритим вихідним кодом – GNU JavaMail – яка реалізує тільки специфікацію JavaMail версії 1.3; це єдина вільна реалізація, що підтримує протокол NNTP, дозволяючи читати і відсилати статті в новинні групи.

До складу JavaMail не входить поштовий сервер, проте для подібних завдань можуть використовуватися вільні Apache James і Java Email Server (POP3 і SMTP), або бібліотека SubEthaSMTP для створення SMTP сервера.

JDOM це вільна Java-реалізація DOM для XML, створена з урахуванням особливостей мови і платформи Java. JDOM інтегрується з Document Object Model (DOM) і Simple API for XML (SAX), підтримує XPath і XSLT. У JDOM використовуються зовнішні парсери для генерації документів.

**4.3 Особливості створення структур даних та програмних модулів**

В основі створення програмних модулів лежить ООП підхід. Java відноситься до мов об'єктно-орієнтованого типу, тому, повністю підходить для написання програмних модулів.

Об'єктно-орієнтоване програмування (ООП) – одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Одною з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами).

Для збереження і відновлення даних на стороні клієнту використовуються xml-файли.

Структура даних по компаніям (для розрахунку даних) повинна бути така:

<companies>

<company>

<name>Назва компанії</name>

<depo>Початковий депозит</depo>

<persent>Середній відсоток інвестування</persent>

<period>Період інвестування</period>

<intervals>

<i>

<plus>Сумма довкладень за i-й період</plus>

<minus>Сумма знімань коштів за i-й період </minus>

<reinvesting>Перенесення поточної сумми у наступний період інвестування</reinvesting>

</i>

</intervals>

</company>

</companies>

Усі маніпуляції виконуються із даним файлом через JDOM парсер.

Дані на стороні серверу зберігаються в БД. Доступ для елементів комірок доступний через активне підключення до БД із виконанням sql-подібних запитів.

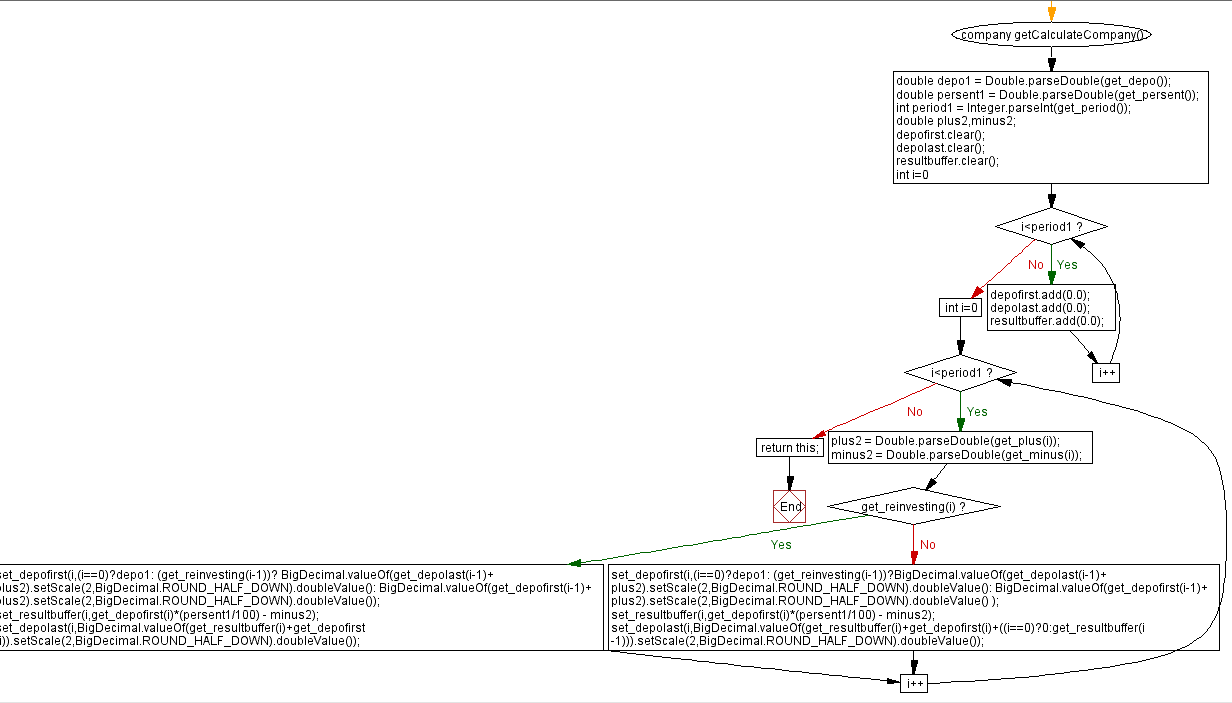
Приклад вибірки даних усіх користувачів із БД:

ResultSet result = st.executeQuery("SELECT \* FROM USER");  
while(result.next()){  
 // Виконуються необхідні маніпуляції

}

1. ТЕСТУВАННЯ СИСТЕМИ

1 Блок-схема алгоритму розрахунку даних (блок-схема автоматично створена в програмі на основі коду програми, надалі блок-схема буде перерисована):

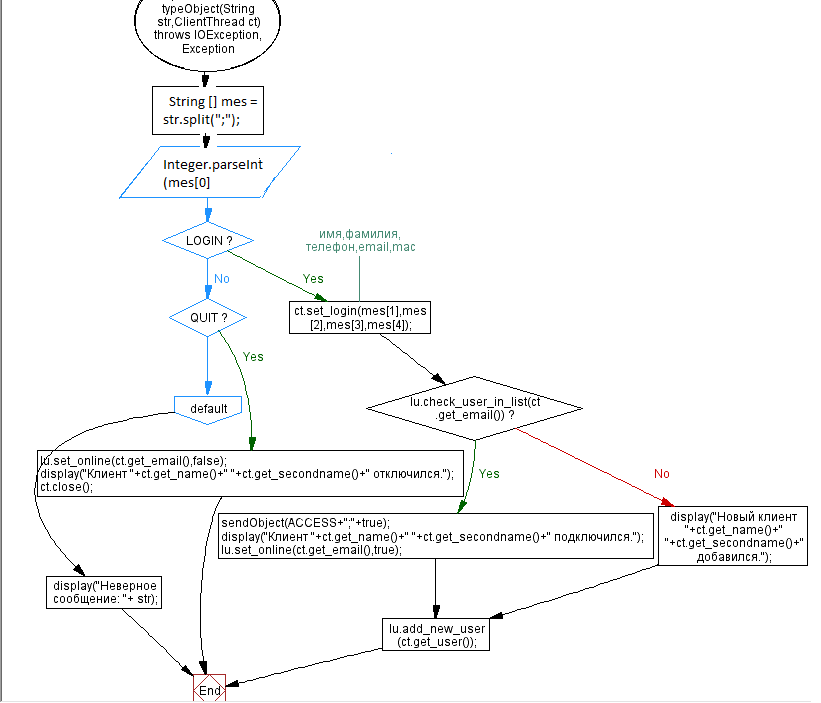


Метод чорного ящика:

Еквівалентне розбиття:

|  |  |  |
| --- | --- | --- |
| Вхідна умова | Правильний клас еквівалентності | Неправильний клас еквівалентності |
| Depo = 100; persent = 7; period = 10; reinvesting = true; | 191,72 – вірне округлення | 191,7151 – невірні дані без округлення чисел |
| Depo = 100; persent = 7; period = 10; reinvesting = true; plus[0]=100 | 420,97 – плюс додається до 0-го періоду | 380,56 – коли переноситься на наступну дату при реінвестуванні |
| Depo = 100; persent = 7; period = 10; reinvesting = true; plus[0]=100; minus[0]=100; | 191,72 – вірні дані зняття та до вкладення сум | 208,59 – невірна сума, бо дані до вкладені на початок, а зняті в кінці місяця, а потім сума перенесена на наступний період |

2 Блок-схема обміну даних між клієнтом та сервером. Серверна частина. (блок-схема автоматично створена в програмі, надалі блок-схема буде перерисована):



Метод білого ящика:

Покриття умов:

Подача сигналу на вхід:

Дані введення: 0;Peter;Buslyuk;34242423;skyliner270594@mail.ru;04-7D-7B-97-0C-9F

Перевірка коректності – усі дані вірні, обмежень немає

Дані введення: 0;Peter;Buslyuk;34242423;skyliner270594@mail.ru;04-7D-7B-97-0C-99F

Перевірка коректності – невірно, мак-адреса не вірна

Дані введення: 0;Peter;Buslyuk;34242423;skyliner270594mail.ru;04-7D-7B-97-0C-99F

Перевірка коректності – невірно, мак-адреса та е-мейл невірний

Дані введення: 3;Peter;Buslyuk;34242423;skyliner270594mail.ru;04-7D-7B-97-0C-99F

Перевірка коректності – невірно, такого типу вибору не існує

Перевірка на існування користувача:

Дані введення:skyliner270594@mail.ru – користувач існує

Skyliner11270594mail.ru – користувач не існує, створюється новий користувач

Таблиця - Функціональне тестування

|  |  |  |  |
| --- | --- | --- | --- |
| № прецеденту | Дія | Очікуваний результат | Результат тестування (успішний(passed)/ невдачний(failed) |
| 1 | Заповнення профілю вірними даними. | Система клієнта зберігає дані клієнта. Дані збережені. | Passed |
| 1 | Система Клієнта відправляє дані авторизації на сервер. | Дані клієнта прийняті на сервері та оброблені. | Passed |
| 1 | Сервер шукає дані клієнта в БД та відсилає доступ. | Доступ наданий | Passed |
| 1 | Сервер не знайшов клієнта в БД. | Сервер створив нового клієнта та надіслав доступ. | Passed |
| 2 | Клієнт обирає збереження даних компаній. | Система Клієнта перевіряє дані та зберігає їх у XML файл | Passed |
| 2 | Перевірка даних не коректність при збереженні. | Система Клієнта видає повідомлення щодо невірних даних. | Failed |
| 3 | Збереження даних віддалено. | Система Клієнта перевіряє дані та відсилає їх на Сервер. | Failed |
| 3 | Сервер отримує дані та зберігає в БД. | Сервер зберіг дані. | Passed |
| 3 | Вказано невірні дані. | Система Клієнта пропонує ввести дані ще раз. | Passed |
| 4 | Розрахунок обраних компаній. | Система Клієнта розраховує дані та видає результат. | Passed |
| 5 | Оповіщення клієнтів на обрані емейл адреси. | Клієнти оповіщені. | Passed |

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

**Додаток А. ІНСТРУКЦІЯ З ВСТАНОВЛЕННЯ ТА ВИКОРИСТАННЯ**

Початкова загрузка програми відбувається із того, що клієнту пропонується ввести дані щодо його подальшої роботи, а потім зберегти дані локально. Профіль клієнта знаходиться в пункті меню «Профіль». Після чого з’являється форма, зображена на рис. А.1.

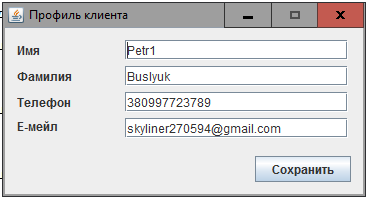


Рисунок А.1 – Форма клієнта «Профіль клієнта»

Клієнт повинен заповнити дані імені, фамілії, телефону та емейлу для інформування клієнта щодо інвестиційних новин серед нещодавно створених, або свої інвестиційних продуктів.

Наступним кроком необхідно додати нову компанію для розрахунку даних. Для цього обирається розділ меню «Компанії->Додати компанію», або натискається комбінацію клавіш Ctrl+N.

Далі необхідно ввести назву компанії, початковий депозит, середній відсоток та продовжність інвестування. Після цього створена компанія з’явиться в таблиці. На рис. А.2 зображено створені компанії, а виділеним рядком – щойно створена.

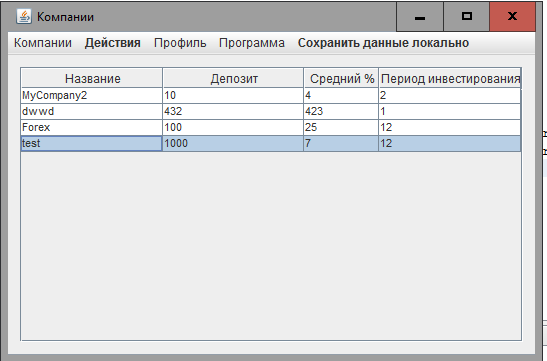


Рисунок А.2 – Форма «Компанії»

Перейдемо до щойно створеної компанії двійним натисненням лівої кнопки миші на неї.

Висвітиться вікно, показане на рис. А.3.

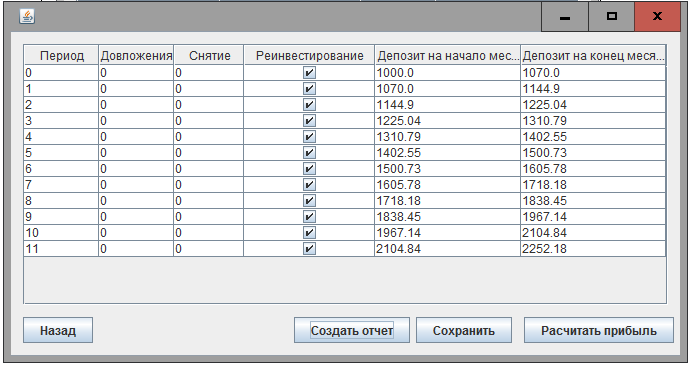


Рисунок А.3 – Форма обраної компанії

Тут ми можемо зберегти дані, створити звіт та розрахувати дані.

Розглянемо створення звіту. Звіт можна створити 2 шляхами – через вибір декількох компаній, і безпосередньо в обраній в 1 екземлярі компанії.

Для створення звіту необхідно буде ввести назву звіту, для того, щоб ви змогли в майбутньому його відшукати на комп’ютері. За умовчанням він зберігається в папці з програмою. Автоматично він зберігається в pdf форматі – рис. А.4.



Рисунок А.4 – Приклад тестового звіту

Приклад звіту всередині можна проглянути на рис. А.5.

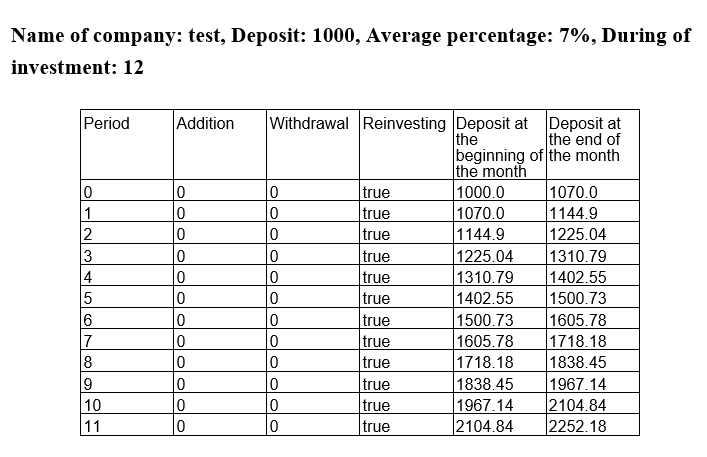


Рисунок А.5 – Приклад звіту в pdf-файлу

Інструкція оператора (сервер):

Коли оператор входить до серверної частини програми, то він попадає на головну форму, зображену на рис. А.6.

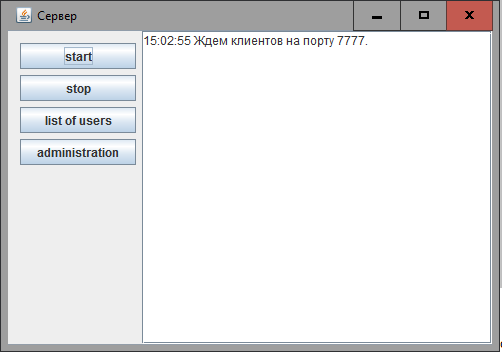


Рисунок А.6 – Головна форма серверу

Тут ми можемо бачити запуск та зупинення сервера, список клієнтів та адміністрування основних деталей.

При виборі списку клієнтів ми попадаємо на форму, де можемо проглянути список усіх користувачів, а також їх поточний статус, та персональні дані. Форма з клієнтами представлена на рис. А.7.

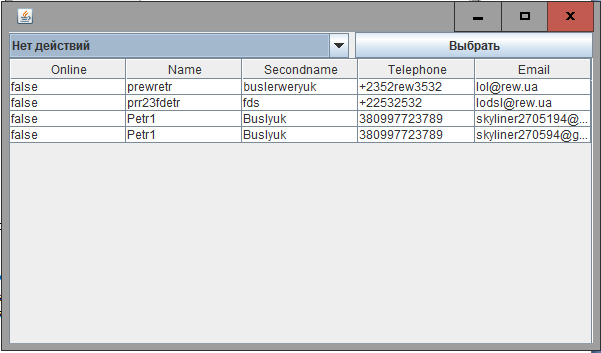


Рисунок А.7 – Форма клієнтів

Ми можемо відіслати заданим користувачам повідомлення на емейл адресу. Для цього необхідно обрати необхідних клієнтів на у випадаючому списку обрати «Вислати повідомлення» та натиснути на кнопку «Обрати».

Система запросить ввести Вас заголовок повідомлення та саме повідомлення. Після цього проінформує про відправлення.

Результат можна проглянути на рис. А.8.

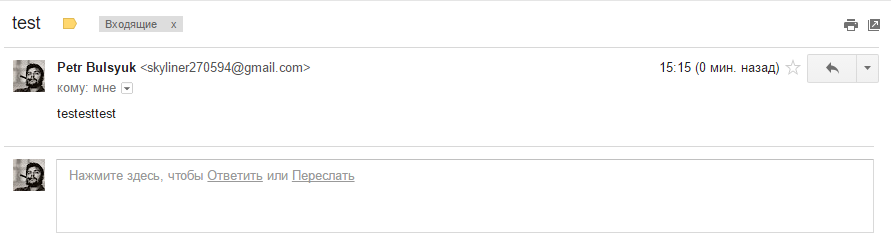


Рисунок А.8 – Поштовий ящик із результатом відправлених даних на вказаного клієнта

Для роботи з програмою перед її інсталяцією необхідно встановити:

1) Jdk 7 версії або вище;

2) БД H2 та зробити бекап із файлу дампу, котрий знаходиться на github репозиторії – для серверної частини.

3) Запустити jar-файл із програмою.

Додаток Б. ЛІСТИНГ ПРОГРАМИ