

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1.....	8
1.1.Основні поняття та визначення	8
1.2.Основні вимоги до системи	8
1.3. Аналіз відомих методів та рішень.....	9
Висновок до першого розділу.....	15
РОЗДІЛ 2.....	16
2.1.Дерево цілей	16
2.2.Конкретизація функціонування системи	20
2.3.Побудова ієрархії процесів.	26
Висновок до другого розділу	27
РОЗДІЛ 3.....	28
3.1.Вибір та обґрунтування засобів розв'язання задачі.....	28
3.2.Технічні характеристики обраних програмних засобів розроблення.....	34
Висновок до третього розділу	38
РОЗДІЛ 4.....	39
4.1.Опис створеного програмного засобу.....	39
4.2.Інструкція користувача	50
4.3.Аналіз контрольного прикладу.....	52
Висновок до четвертого розділу.....	56
РОЗДІЛ 5.....	57
5.1.Економічна характеристика програмного продукту	57

5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту	58
5.3.Оцінювання та аналізування факторів зовнішнього та внутрішнього середо - вищ	59
5.4. Формування стратегічних альтернатив	61
5.5. Бюджетування	64
Висновок до п'ятого розділу	69
ВИСНОВКИ	71
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	74
ANNOTATION	77
ДОДАТКИ	78

ВСТУП

Актуальність теми. Сьогодні, люди, чи то через високий темп життя, чи то через лінь, все менше хочуть приділяти час побутовим речам, які є необхідними для успішного проживання. Це твердження дуже актуальне, коли мова заходить до підрахунків місячних витрат та прогнозування особистого або навіть сімейного бюджету. Кожна повнолітня людина має банківську карточку. А так, як в наші дні все більше і більше речей переходить в цифровий вид, то ми все менше тримаємо в руках паперові гроші, надаючи перевагу оплаті різноманітних послуг більш зручному методу розрахунків. Саме тому тема онлайн-банкінгу стає близькою кожному. Думаю, що ніхто не стане сперечатися, що ПриватБанк, в межах України, найбільший банк, який є ще й державним. Використання послуг даного банку є знайомим кожному, а тому аналіз транзакцій, здійснених клієнтами даного банку, є актуальним сьогодні.

Мета і задачі дослідження. Полегшення проведення статистики власних витрат за рахунками ПриватБанку є ціллю дослідження даної інформаційної системи.

Для реалізації даної мети необхідно розв'язати такі задачі:

- здійснити аналіз предметної області шляхом дослідження та тестування існуючих рішень даної проблеми та визначити побажання майбутніх користувачів щодо функціоналу;
- провести системний аналіз об'єкта досліджень шляхом побудови дерева цілей, діаграми потоків даних та ієрархії задач;
- здійснити вибір засобів для вирішення даної проблеми шляхом дослідження та порівняння їх технічних характеристик;
- реалізувати інформаційну систему, провести ручне та автоматизоване тестування, описати основні модулі системи.

Об'єкт дослідження. Процес аналізу транзакцій банку.

Предмет дослідження. Методи та засоби для аналізу транзакцій банку.

Наукова новизна одержаних результатів. Зазнало подальшого розвитку проведення статистики по транзакціях клієнта банку. Підвищена простота подачі даної статистики. Покращено доступність самотійного аналізу власних витрат.

Практичне значення одержаних результатів. Даний продукт повністю готовий до використання. Маштаб використання обмежений кількістю клієнтів ПриватБанку. Практичне значення одержаних результатів принесе внесок в соціальне життя користувачів даного продукту, шляхом полегшення вираховування власних витрат та прогнозування майбутнього бюджету.

РОЗДІЛ 1

Аналітичний огляд літературних та інших джерел

1.1. Основні поняття та визначення

Перед тим, як почати розбирати вже існуючі шляхи вирішення задачі аналізу транзакцій клієнта банку, варто розібратися з важливими поняттями та термінами.

Банк – це, фактично, юридичне лице, яке, за наявності банківської ліцензії, має право і може надавати послуги банку, відомості про таку ліцензію внесені до Державного реєстру банків України. [1]

Банківська виписка – це звичайний реєстр транзакцій клієнта банку, які були здійснені протягом певного періоду часу, який визначається часовими термінами даної виписки. Може бути підставою для записів бухгалтерії. [2]

Банківська транзакція – це різні операції клієнта банку, які пов'язані з фінансами та рахунками (наприклад, нарахування зарплатні, поповнення мобільного рахунку, оплата податків, переведення коштів з однієї банківської карти на іншу). Варто зазначити, що банківські транзакції є двох видів: оффлайн та онлайн. Онлайн транзакції не можуть бути здійснені без з'єднання із банківським сервером, коли оффлайн транзакції – можуть. [3]

XML – це програмний засіб для забезпечення зберігання та транспортування даних. [4]

Аналіз даних – це дисципліна, метою якої є накопичення, обробка та використання для здійснення різного роду рішень, даних в великій кількості. Ці всі процеси полегшують процес знаходження закономірностей в цих даних, що за собою несе надзвичайний потенційний вплив на всі сфери нашого життя. [5]

1.2. Основні вимоги до системи

Визначивши основні поняття, можемо переходити до огляду проблеми та шляхів її вирішення. Отож, проблема полягає в тому, що нам потрібно визначити, куди перераховуються грошові потоки з нашого рахунку, взяти категорії, на які ми

витрачаємо найбільше коштів і конкретні суми. Єдине, що ми маємо на старті – це виписка транзакцій з банку в XML-форматі. Основними вимогами до системи є:

- доступність (кожен користувач має мати змогу використовувати дану систему без жодної підготовки);
- швидкість (отримання бажаного результату має відбуватися в лічені секунди);
- високий рівень автоматизації(користувач повинен зробити мінімальну кількість дій для досягнення бажаного ефекту);
- підтримка обробки даних з Приватбанку (напрямую або із допомогою виписки);
- кросплатформеність (система повинна працювати на різних операційних системах);
- безпека даних (відсутність ризику втрати даних, шахрайства та збору персональних даних користувача).

1.3. Аналіз відомих методів та рішень

ПриватБанк надає можливість переглянути виписку по витратах. Щоб отримати виписку по карті за допомогою Приват 24 досить на головній сторінці сервісу вибрати меню «виписки». При цьому в правій частині екрану з'явиться звіт про рух коштів за місяць по рахунку, що стоїть першим у списку. Для того, щоб отримати дані з якоїсь іншої карти, необхідно вибрати її зі списку і клацнути по ній. Виписка по картці ПриватБанку може бути отримана за будь-який період, для цього достатньо вручну встановити потрібні дати. Система дає можливість подивитися не тільки список операцій, а й діаграми надходження і витрати коштів, отримати інформацію про розподіл витрат за певний період. Тобто тут можна дізнатися скільки було витрачено коштів на покупки в магазинах, скільки знято в банкоматах, використано на оплату мобільного зв'язку та інших послуг, переведено на інші карти та інше. Внизу сторінки знаходиться розбивка по категоріях; при виборі цікавої категорії можна отримати список проведених операцій за місяць. Якщо користувач хоче використовувати

мобільний пристрій для цього, то у мобільному додатку Приват 24 виписка по карті доступна на домашній сторінці програми. Для цього досить вибрати меню «карти». У вікні можна побачити список доступних карт і залишки по кожній з них. Для перегляду операцій по карті необхідно підсвітити її. У вікні відобразиться рух коштів за останні дні. Але можливість перегляду діаграм, базованих на інформації по витратах, відсутня, тому використання мобільного додатку різко обмежує кількість отримуваної статистики по рахунках. Також, варто зазначити, що це не єдиний мінус даної системи – наведені діаграми зображені не найкращим чином, що робить важчим їхнє сприйняття та розуміння. Використовуючи сайт ПриватБанку ми можемо експортувати дану виписку в Excel форматі, в мобільному додатку дана функція відсутня. Інформацію взято з офіційного сайту ПриватБанку. [6]

Підібрано та проаналізовано вже готові шляхи вирішення даної проблеми. Ось опис деяких із них, також наведено плюси і мінуси даних систем:

1. Найпримітивніше рішення – це використання Microsoft Excel. Microsoft Excel – це програмний засіб, створений світовим гігантом в індустрії програмного забезпечення – Microsoft, який надає змогу працювати з електронними таблицями, стабільний випуск даного продукту було розпочато в 2016 році [7]. Отож, для того, щоб проаналізувати виписку за допомогою даної програми, нам потрібно завантажити виписку в програмне середовище і в ньому вже власноруч створювати макроси для підрахунку витрат за місяць, виділення категорій та сумування витрат по них. Після цього потрібно будувати діаграми та графіки по вже знайдених даних. Приклад інтерфейсу Microsoft Excel зображено на Рис.1.1.(зображення взято з офіційного сайту Microsoft [8]).

Мінусами даного підходу є те, що все потрібно робити власноруч. Це забирає багато часу, багато рутинної роботи. Люди, які не мають досвіду з даним програмним продуктом, не зможуть зробити всі потрібні дії для реалізації поставлених задач. Даний метод програє в усьому перед наступними рішеннями.

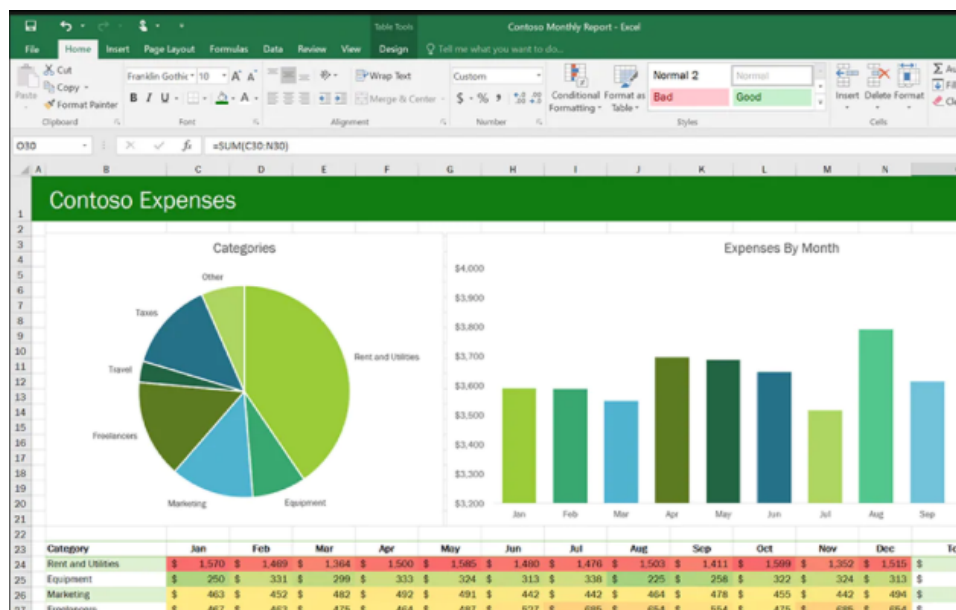


Рис.1.1.Інтерфес Microsoft Excel

2. Monefy – Money Manager - це мобільний додаток, який допомагає відслідковувати ваші витрати. Для того, щоб дана програма аналізувала ваші грошові потоки, вам потрібно лише додавати нові записи, коли ви здійснюєте покупку чаю чи викликаєте таксі і подібне. Вам не потрібно заповнювати нічого більше, окрім ціни. Вагомим плюсом є те, що ви можете синхронізувати дані з телефону та планшета. Можна створювати нові категорії, робити нові записи або, навпаки, видаляти старі – все буде синхронізовано. Перевагами даної програми є такі речі: зрозумілий та зручний інтерфейс, безпечна синхронізація між різними пристроями, підтримка різних валют, підтримка декількох акаунтів одночасно, вбудований калькулятор. Приклад інтерфейсу Monefy зображено на Рис.1.2. Зображення було взято з офіційного сайту мобільного додатка Monefy [9].

Основними мінусами даного додатка є те, що користувач повинен власноруч вводити всі транзакції, які він здійснює, також проблемою є те, що немає змоги проаналізувати дані за термін, до того моменту коли дану програму було скачено користувачем. Так, як користувач вводить власноруч всі записи, то це значить, що гарантії правдивих даних відсутні, що є вагомим мінусом для вирішення поставленої задачі.

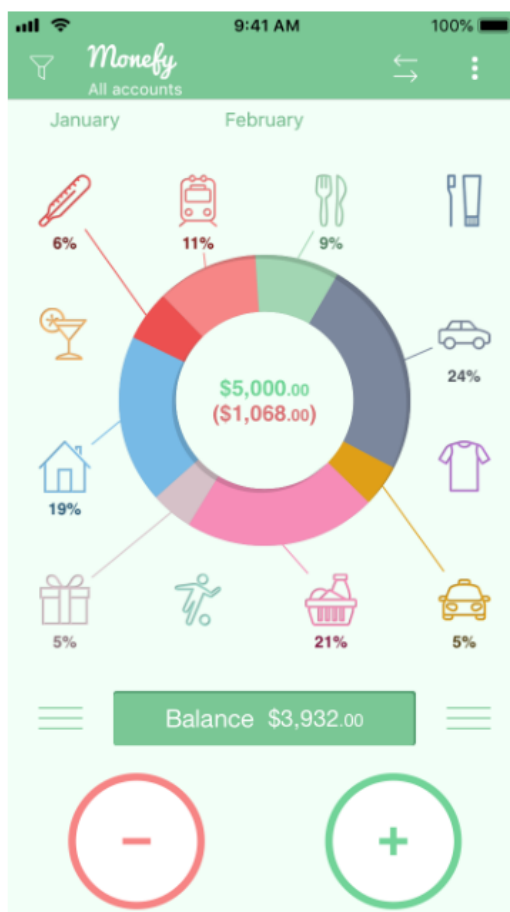


Рис.1.2.Інтерфейс Monefy

3. Дзен-мани: учет расходов и доходов – мобільний додаток, який підтримується операційними системи Android та IOS. Дана програма працює на основі синхронізації з банком. Підтримується синхронізація з такими банками : Приватбанк, Райффайзен Банк Аваль, УкрСибБанк, monobank та інші. За рахунок даного додатка користувач може економити на банківський SMS-повідомленнях, за рахунок того, що він синхронізується з банком та отримує повідомлення про ваші операції. Це все відбувається автоматично, додаток підгружає дані про операції та визначає її категорію. Інтерфейс Дзен-мани зображено на Рис.1.3., зображення взято з офіційного сайту [10]. Присутня синхронізація між різними пристроями, що надає змогу для планування коштів для сімей. Даний додаток був вперше випущений в 2015 році і підтримується досі (це вже 6 версія), більш ніж 500 тисяч разів було завантажено даний додаток, що говорить про те, що він має свою велику та стабільну аудиторію клієнтів.

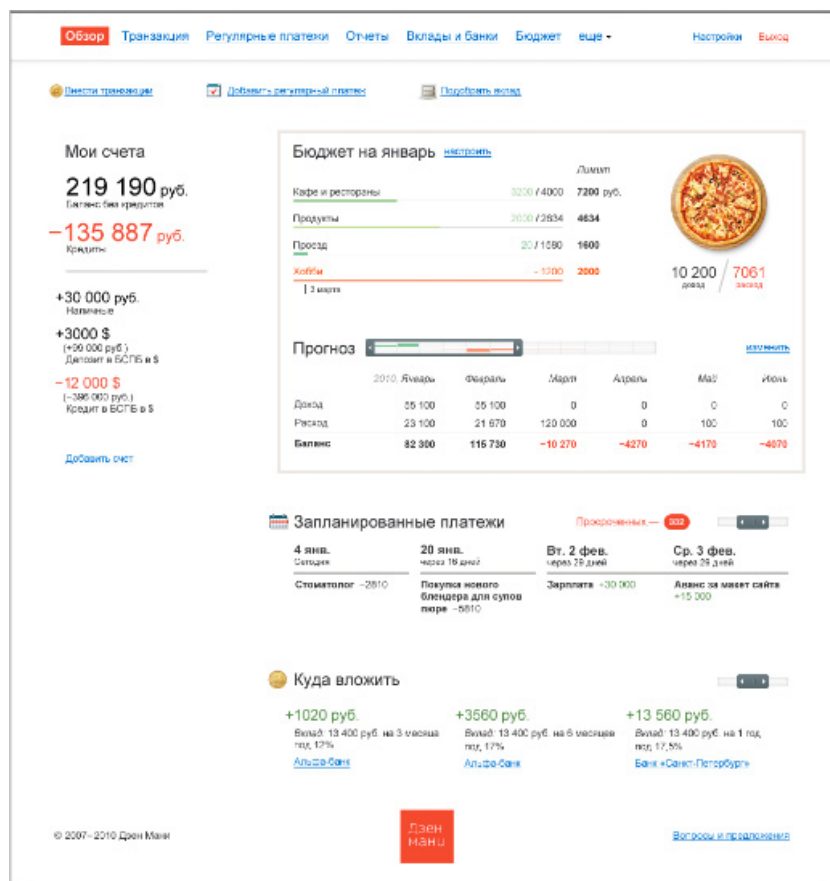


Рис.1.3.Интерфейс Дзен-мани

Основними мінусами даного додатку є те, що тут відсутня можливість перегляду статистики по транзакціях, які були здійснені до завантаження даної програми, що не може не засмучувати. Важливо зазначити, що синхронізація даної програми з банком несе за собою певні ризики пов'язані з небезпекою особистих даних, що лякає багатьох людей.

4. Monobank – мобільний додаток, розроблений банком з такою ж назвою. Даний банк є першим банком в Україні, який не має відділень. Найбільш новітній банк, який впроваджує нові ідеї та надає нові можливості своїм клієнтам. Говорячи про сам мобільний додаток, то він вперше був випущений в 2017 році і з того часу його було завантажено через Google Play Market більш, ніж один мільйон разів, що говорить про його популярність серед користувачів. Сам додаток надає можливість оплати будь-яких послуг за рахунок карточки або власного телефону. Можна оплачувати податкові рахунки, штрафи. Присутня функція кеш-беку. Але найважливішим серед цього всього для нас буде саме

можливість побачити статистику власних транзакцій. Дана функція доволі просто реалізована розробниками, немає зайвих непотрібних функцій, які б візуально заважили розгледіти потрібні дані, можна побачити статистику по витратах у вигляді "пайчарту" або у вигляді списку, що можна наявно побачити на Рис.1.4 та Рис.1.5, відповідно. Зображення зроблено на персональний мобільний пристрій. Інформацію взято з офіційного сайту. [11]

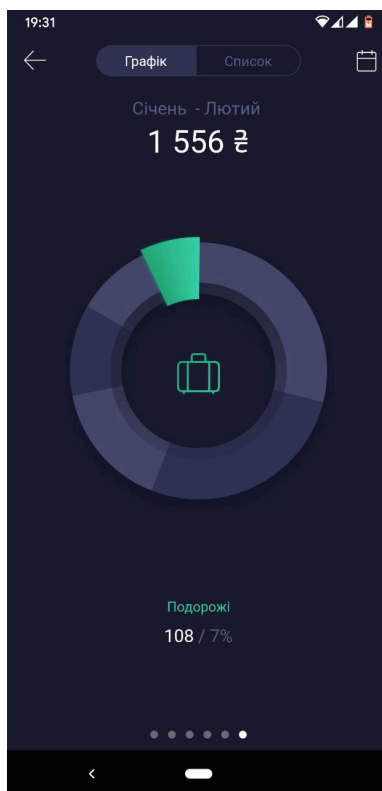


Рис.1.4.Приклад інтерфейсу monobank (кругова діаграма)

Основним мінусом даної функції, яку нам надає додаток monobank, є те, що вона працює лиш вбудовано в даний додаток, це значить, що ми ніяк не можемо використати її для аналізу транзакцій з ПриватБанку, також додатковим мінусом є й те, що немає можливості використати дану функцію на персональному комп'ютері.

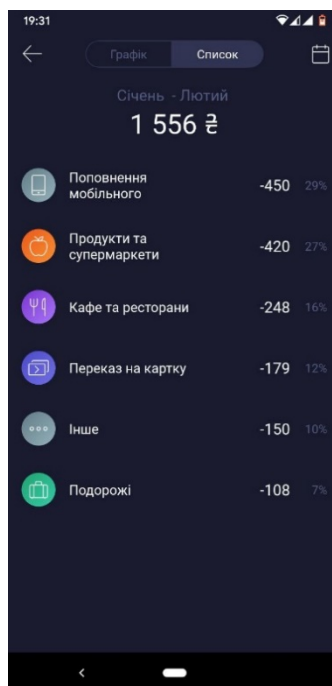


Рис.1.5.Інтерфейс monobank (у вигляді списку)

Висновок до першого розділу

Проаналізувавши декілька шляхів вирішення проблеми аналізу транзакцій клієнта банку, можна сказати, що кожна з наведених вище систем існує роками і має право на подальше існування. Вони мають велику кількість користувачів, але й мінуси присутні: у ПриватБанку – це відсутність можливості аналізу на мобільному телефоні і важко зрозумілі діаграми; Excel – це важкість виконання та затрати часу; Monefy – це низький рівень автоматизації; Дзен-мани – це відсутність веб-додатку та питання щодо безпеки власних даних; monobank – вбудованість в власну систему та нездатність аналізу даних інших банків (в нашому випадку ПриватБанку). Як результат дослідження, можна сказати, що створення нової системи буде видом покращення вже існуючих і є актуальним для реалізації.

РОЗДІЛ 2

Системний аналіз об'єкта дослідження

2.1.Дерево цілей

Для всебічного опису сутності системи, яка досліджується, в даній роботі побудовано дерево цілей [12] (Рис.2.1.)

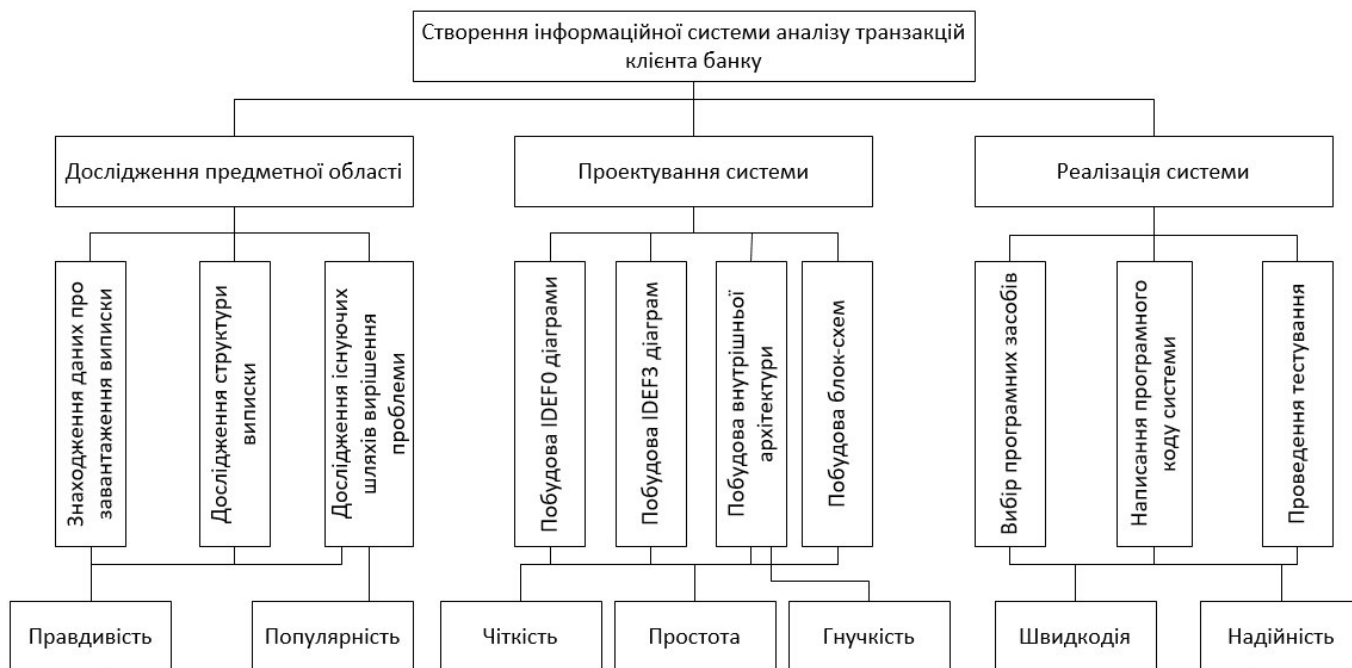


Рис.2.1.Дерево цілей інформаційної системи

Як можна побачити з Рис.2.1. дерево цілей представлено чотирма ієрархічними рівнями, а саме:

- *Перший рівень* – це основна мета, яка представлена самим створенням інформаційної системи аналізу транзакцій клієнта банку, тобто основна ціль нашої роботи.
- *Другий рівень* – це аспекти основної мети, які вказують, які основні задачі мають бути виконані для досягнення бажаного.
- *Третій рівень* – це підаспекти нашої основної мети, даний рівень рахується проміжним, але є не менш важливим, за інші, тому що надає більш детальну інформацію.
- *Четвертий рівень* – це критерії якості функціонування системи.

Пропоную розглянути більш детально дане дерево цілей.

Першим аспектом вказано *"Дослідження предметної області"* - даний етап є надзвичайно важливим для створення будь-якої інформаційної системи, адже саме він вибудовує основні поняття та надає фундамент для подальшого просування до наступних етапів, які не можуть бути реалізовані без основи. Він представлений такими підаспектами:

- *Знаходження даних про завантаження виписки* – так, як проєктована система передбачає аналіз транзакцій саме з виписки, то нам потрібна інформація про те, як і де можна взяти даний документ.
- *Дослідження структури виписки* – для того, щоб аналізувати дані, потрібно спочатку взнати, який тип і структуру вони мають.
- *Дослідження існуючих шляхів вирішення проблеми* - аналіз готових рішень є надзвичайно корисним процесом, який надає розуміння, що вже зроблено і можна запозичити, а що потрібно удосконалювати для усунення недоліків.

Основними критеріями якості для даного аспекту, було обрано *"Правдивість"* – тобто, досліджувана інформація, про банківську виписку та існуючі шляхи рішення, повинна бути достовірною та актуальною; *"Популярність"* – якщо система популярна, то значить вона подобається користувачам, що говорить про високу якість та інтерес до неї.

Наступним аспектом є *"Проектування системи"* – під час даного етапу відбувається народження бачення та каркасу інформаційної системи, описуються основні процеси та їх деталізація. Тут в нас є наступні підаспекти:

- *Побудова IDEF0 діаграми* – дана діаграма описує основні бізнес-процеси, що полегшує розуміння предметної області.
- *Побудова IDEF3 діаграм* – діаграма надає більш деталізований опис функціонування системи та послідовності процесів в ній.
- *Побудова внутрішньої архітектури* – даний етап базується на побудові архітектури програмних модулів системи.

- *Побудова блок-схем* – потрібна для кращого розуміння, що саме і як потрібно робити розробнику програмного продукту.

Критеріями якості обрано "Чіткість" і "Простоту", адже всі діаграми та блок-схеми повинні бути зрозумілі та не мати нічого зайвого, окремо варто виділити "Гнучкість", тобто архітектура системи повинна бути гнучкою до можливих змін.

Третім аспектом є *"Реалізація системи"* – на даному етапі розробники, за допомогою інформації, яка була набута на попередніх етапах розробки, створюють програмний продукт. Підаспектами є:

- *Вибір програмних засобів* – потрібно обрати найбільш зручні та ефективні програмні інструменти.
- *Написання програмного коду системи* – без даного процесу є неможливим існування системи.
- *Проведення тестування* – це потрібно для того, щоб бути впевненим в коректності роботи програми.

Для останнього, але не по значущості, аспекту було обрано такі критерії якості системи: *"Швидкодія", "Надійність"*.

Для критеріїв, вищенаведеного дерева цілей, було зроблено оцінку за важливістю внеску в отримання бажаного результату. Сума критеріїв повинна бути рівною одиниці, результати оцінки наведено на Табл.2.1.

Порядкові номери критеріїв:

1. Правдивість.
2. Популярність.
3. Чіткість.
4. Простота.
5. Гнучкість.
6. Швидкодія.
7. Надійність.

Оцінка важливості критеріїв генеральної мети

Критерій	1	2	3	4	5	6	7
Оцінка	0.2	0.07	0.12	0.16	0.13	0.16	0.16

Після аналізу дерева цілей та максимально схожих систем до проектованої, які вже існують, було обрано три шляхи впровадження системи:

1. Інформаційно-аналітична система – базована на тому, що користувач завантажує виписку витрат з ПриватБанку та завантажує її в систему, які на основі транзакцій дає візуалізовану та текстову статистику.

2. Базована на власному списку витрат система – передбачає ручне введення кожної витрати та її ціни, після чого створюється новий запис до списку грошових витрат, на основі якого буде проведено статистику.

3. Базована на синхронізації з банком система – передбачає синхронізацію з банківською карткою, коли проводиться оплата даною картою, то система автоматично підвантажує дану транзакцію в свою базу даних.

Для вибору одного з трьох альтернативних шляхів реалізації, було застосовано МАІ (метод аналітичної ієрархії) [13]. Кожному варіанту реалізації виставлено якісну оцінку відповідності критерію за шкалою 1 – 5 (де 5 – це найбільш відповідне рішення, 1 – це найменш відповідне рішення). Висновок щодо кожної альтернативи сформовано за допомогою формули (2.1). Отримано Табл.2.2.

$$\Sigma = O_1 * p_1 + \dots + O_n * p_n, \quad (2.1)$$

де O_n - оцінка, p_n - важливість критерію.

Дивлячись на Табл.2.2., бачимо, що перша альтернатива набрала 4.50 балів, друга - 3.22, третя - 4.43. Тобто, ручний ввід власних транзакцій виявився найменш підходящим методом. Перший і третій підхід опинилися близько по балах один до одного, але все ж різниця є. Враховуючи, що перша альтернатива набрала найбільше балів і має найбезпечніший підхід, з точки зору особистих даних, вибір падає саме на неї.

Порівняння альтернатив за критерієм генеральної мети

Критерій	1	2	3	4	5	6	7	Результат
Альтернатива 1	5	2	5	5	4	4	5	4.50
Альтернатива 2	3	3	4	3	5	2	3	3.22
Альтернатива 3	5	5	4	5	4	4	4	4.43

2.2.Конкретизація функціонування системи

Для деталізації основного процесу інформаційної системи, побудовано контекстну діаграму IDEF0 та IDEF0(декомпонована), вони зображені на Рис.2.2. та Рис.2.3., відповідно.

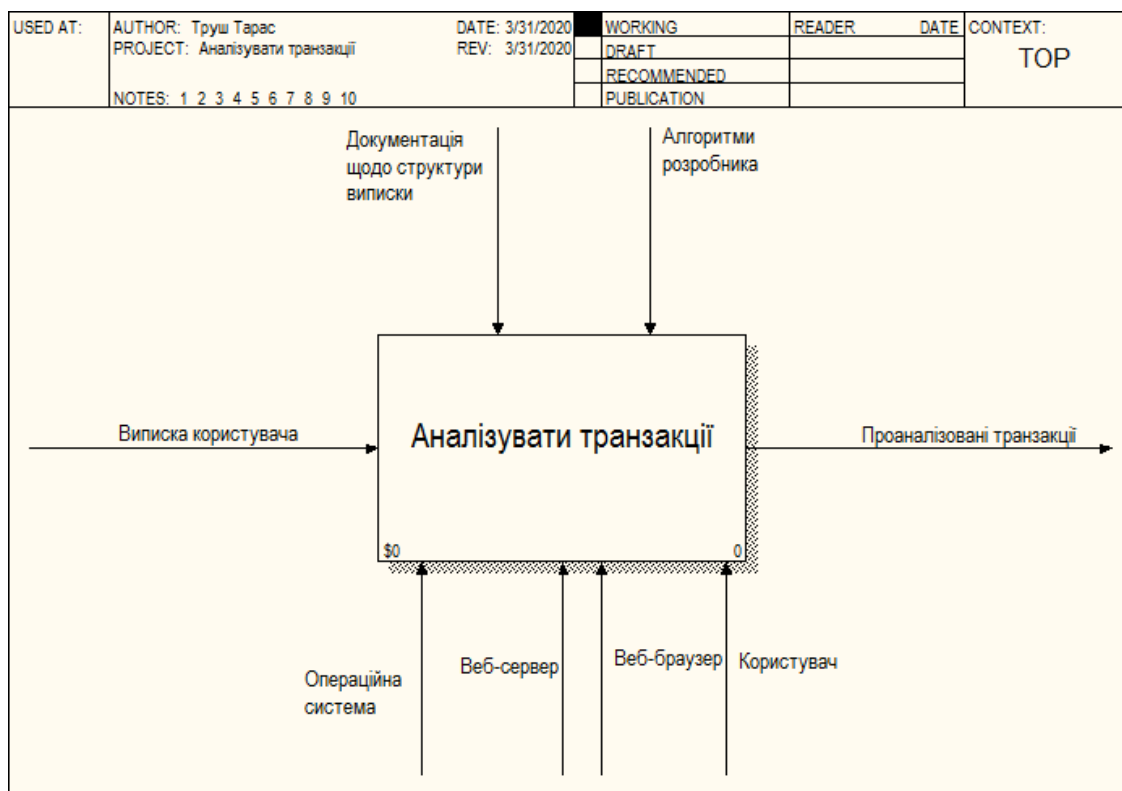


Рис.2.2. Контекстна діаграма

На Рис.2.2. ми можемо побачити, що нашим основним процесом є процес "Аналізувати транзакції". Для цього на вхід подається виписка по витратах (тобто по транзакціях) користувача, яка і буде джерелом даних, які будуть аналізуватися даною системою. Як механізм, будуть використовуватись операційна система

персонального засобу користувача, сам користувач, веб-браузер та веб-сервер. Це все буде керуватися на основі документації щодо структури даної виписки та алгоритмів, які розроблені розробником спеціально для вирішення конкретних завдань, поставлених перед ним.

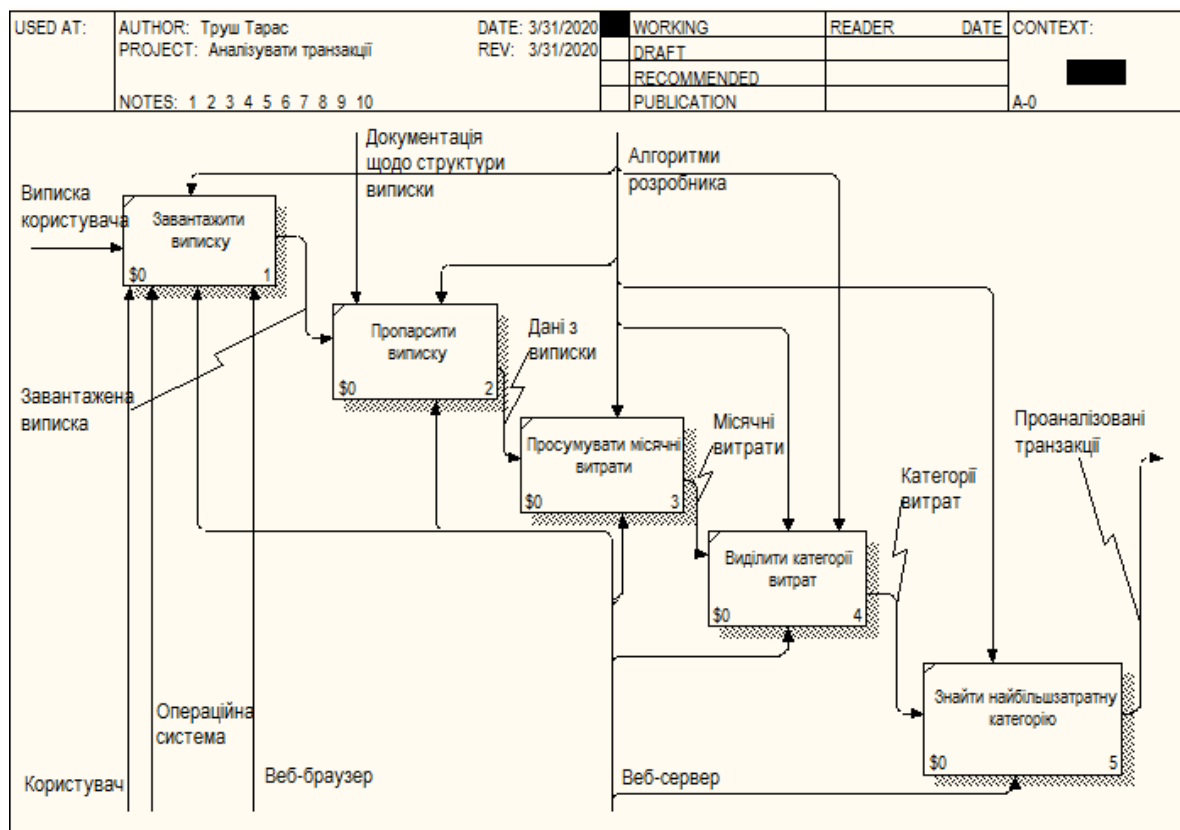


Рис.2.3. IDEF0 діаграма (декомпонована)

На Рис.2.3. ми бачимо декомпоновану IDEF0 діаграма, яка більш докладно деталізує ієрархію процесів. На зображенні ми бачимо, що все починається з того, що на вхід подається виписка користувача, в даний процес задіяні *Користувач*, який *вибирає виписку, яку завантажити*; *Операційна система*, яка надає доступ до файлової системи; *Веб-браузер*, який слугує провідником між сервером та користувачем; *Веб-сервер*, на якому й збережеться та буде оброблено обраний файл. На виході маємо виписку, яка збережена на сервері. Далі виписка парситься, веб-сервер бере участь в цьому, адже саме він обробляє ці всі дані. Після того, як ми витягнули дані з виписки ми сумуємо місячні витрати та виділяємо категорії, за рахунок чого находимо найбільш затратну категорію витрат користувача. На виході маємо проаналізовані транзакції користувача.

Для того, щоб показати детальніший перебіг подій, побудовано діаграми перебігу робіт (потоків даних), іншими словами IDEF3. На Рис.2.4. зображено деталізацію процесу "Завантажити виписку".

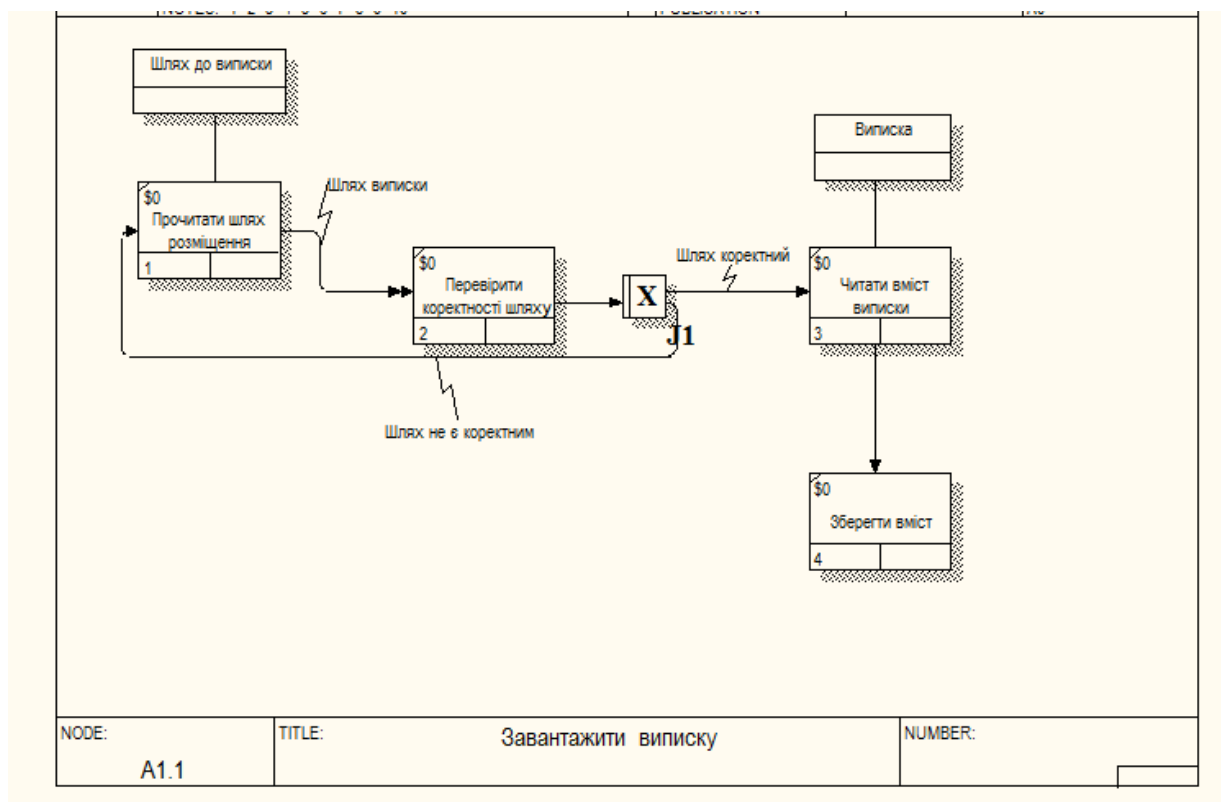


Рис.2.4.Конкретизація процесу "Завантажити виписку"

Отож, на даному рисунку ми бачимо, що процес починається з того, що у нас є шлях до файлу-виписки, який нам вказав користувач. Система зчитує даний шлях розміщення файлу, після чого, перевіряє чи цей шлях є коректним, тобто, чи він правильно введений і чи існує такий файл взагалі. Якщо шлях некоректний, то система прочитає новий шлях, який вкаже користувач і почне заново попередні дії. Якщо ж шлях коректний, то ми отримуємо доступ до виписки, після чого система зчитує вміст даного файлу і зберігає його на сервері.

На Рис.2.5. конкретизовано процес "Пропарсити виписку". З Рис.2.5. бачимо, що процес починається з того, що у нас є вміст виписки, який до того було збережено на сервері системи. Для того, щоб витягнути коректно самі дані з цього вмісту, потрібно спочатку перевірити, чи структура вмісту є коректною, інакше витягування даних дасть помилку. Якщо структура некоректна, то потрібно

завантажити нову виписку. Якщо все коректно, то вміст виписки витягується (парситься) та перетворюється в програмний об'єкт. Це все робиться через те, що дані виписки є в XML або JSON форматі. Після чого система зберігає дані, які було витягнуто, тобто транзакції користувача.

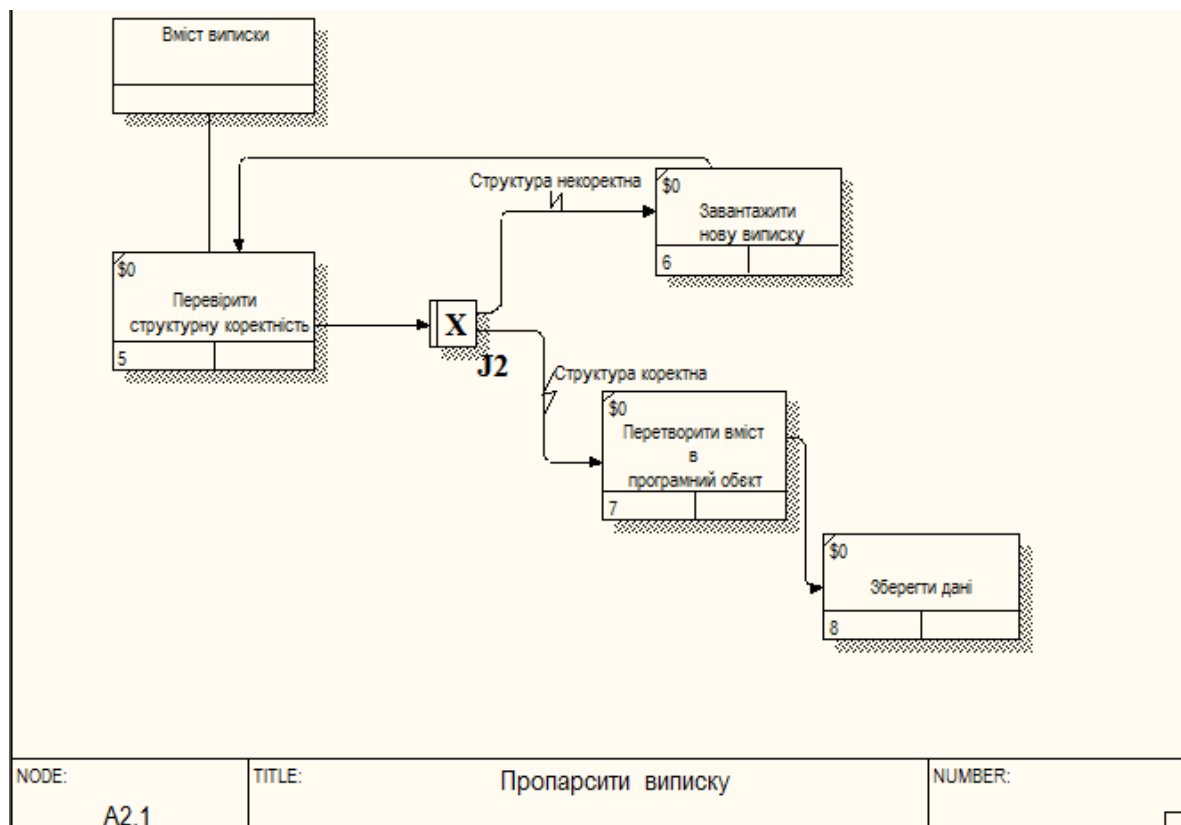


Рис.2.5.Конкретизація процесу "Пропарсити виписку"

На Рис.2.6. ілюстровано конкретизацію процесу "Просумувати місячні витрати".

На Рис.2.6. видно, що процес сумування місячних витрат починається з того, що система має дані отримані з виписки, які були до того збережені на сервері. Далі система витягує дані щодо дати транзакції та її ціну, наприклад, 12.02.2020 - дата і "234.25 грн" – це ціна. Ці дві речі не обов'язково одночасно починаються, але для продовження вони обидві мають бути закінчені. Після цього ці дані додаються по датах автоматично зберігаючись. В результаті ми маємо дані про те, скільки було витрачено в конкретний день.

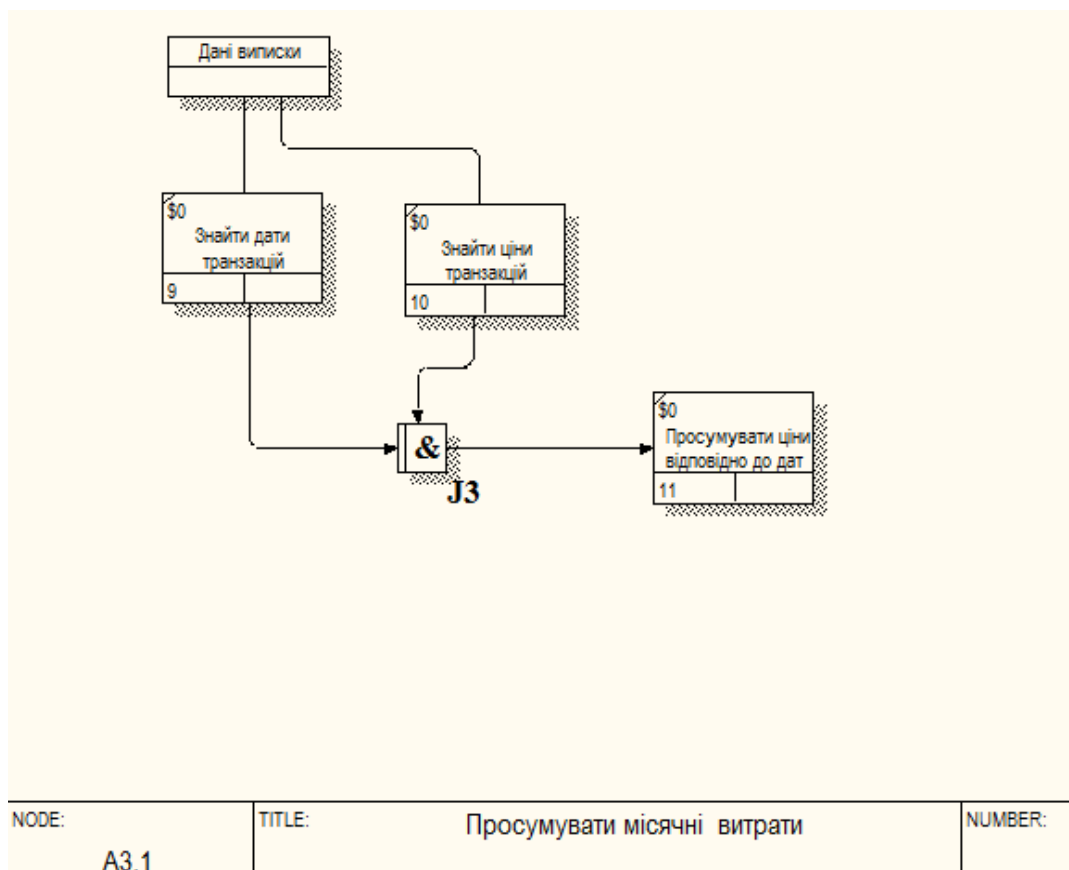


Рис.2.6.Конкретизація процесу "Просумувати місячні витрати"

На Рис.2.7. зображено деталізацію процесу "Виділити категорії витрат".

З рисунку видно, що процес починається з того, що є транзакції за місяць. Отож, спочатку йде перевірка, чи є ще записи про транзакції. Якщо нема, то нема що аналізувати, значить завершуємо процес. Якщо ще є записи, то йде перехід на наступний запис про транзакції та перевірка категорії транзакції. В випадку, коли дана категорія вже є в базі даних, то просто додається ціна транзакції до загальної ціни категорії, після чого йде перехід на перевірку, чи залишились ще записи і так по колу, поки не закінчуться записи, тоді процес завершується. Якщо ж категорія така, якої ще не було, то спочатку вона добавляється в базу даних і тоді сумується і повторюються всі ті самі дії, що були описані.

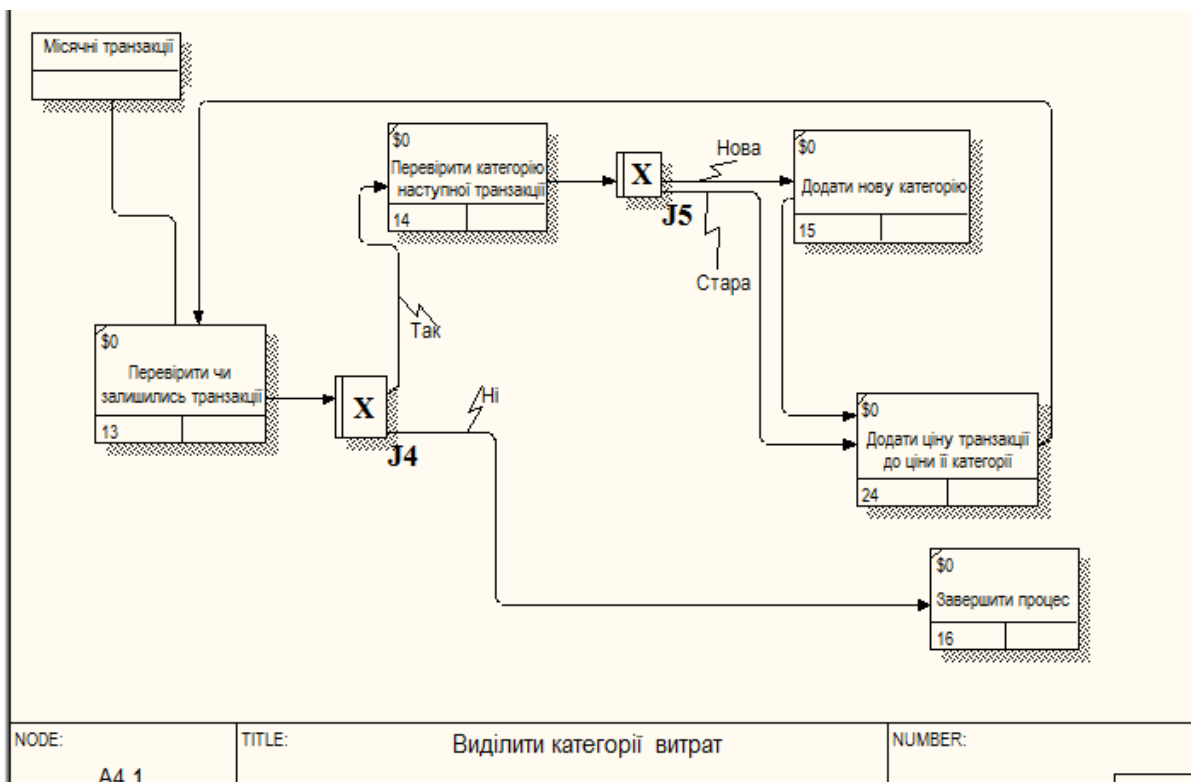


Рис.2.7.Деталізація процесу "Виділити категорії витрат"

На Рис.2.8. деталізовано процес "Знайти найбільш затратну категорію".

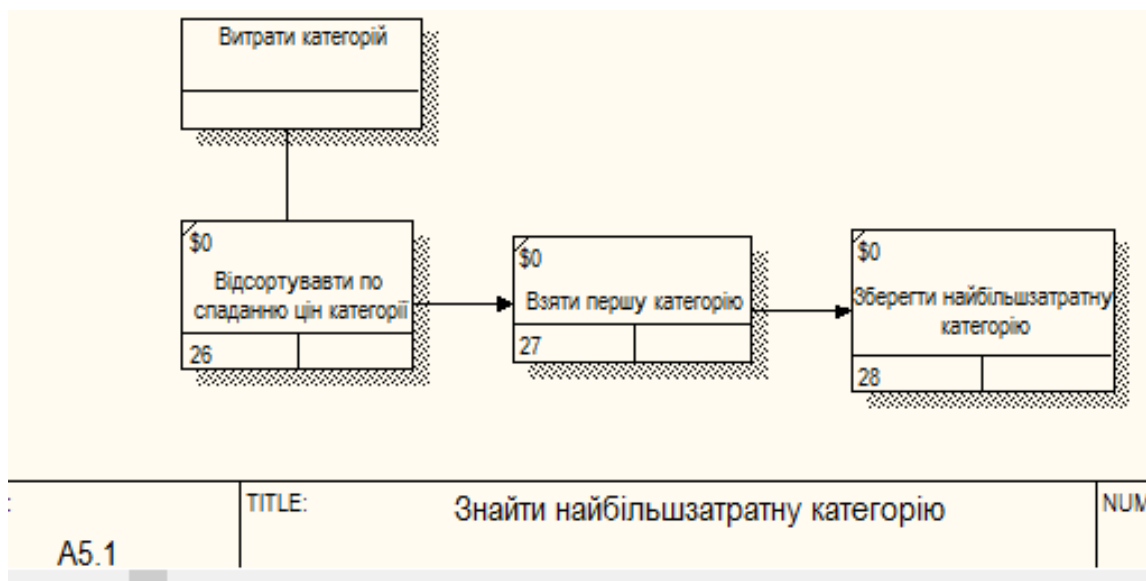


Рис.2.8.Деталізація процесу "Знайти найбільш затратну категорію"

На Рис.2.8. видно, що процес починається з того, що є витрати категорій, які були отримані в попередньому процесі. Дані записи з витратами відсортовуються за спаданням величини цін даних категорій і після цього вибирається перша, тобто найбільш затратна.

2.3. Побудова ієрархії процесів.

Подано структуру досліджуваної системи у вигляді ієрархії процесів різних рівнів, а саме: у вигляді деревовидної структури яку було згенерована в All fusion process modeler Рис.2.9.

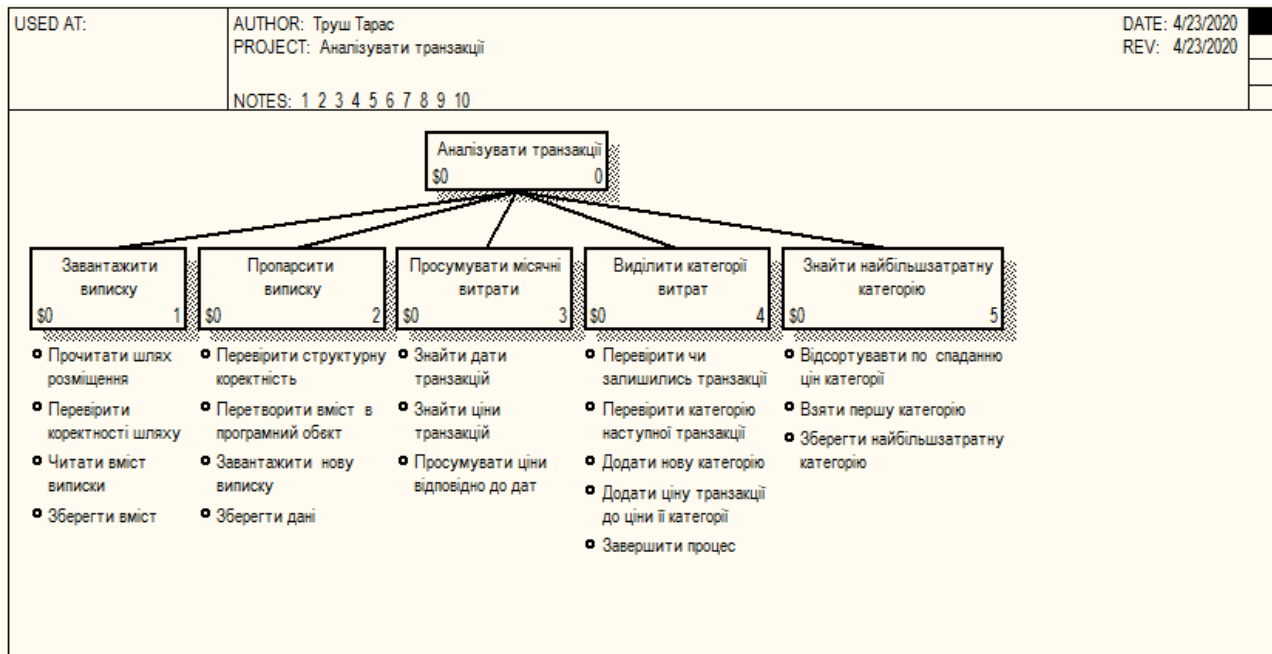


Рис.2.9.Ієрархія процесів досліджуваної системи

Як можна побачити на Рис.2.9., головною задачею є "Створити інформаційну систему аналізу транзакцій клієнта банку". Бачимо, що далі йдуть чотири підзадачі, а саме: "Завантажити виписку", "Пропарсити виписку", "Просумувати місячні витрати", "Знайти категорії та їх ціну". Кожна підзадача представлена набором певних функцій, а саме:

- "Завантажити виписку" - представляє собою набір функцій для читання та перевірки коректності шляху до файлу, та завантаження його.
- "Пропарсити виписку" – дана підзадача відповідає за перевірку коректності структури та витягування (читання) вмісту файлу.
- "Просумувати місячні витрати" – знаходження та додавання даних по місяцях року, для кращої структуризації.
- "Знайти категорії та їх ціну" – виділення нових категорій витрат, підрахунок втрачених коштів для кожної категорії.

Висновок до другого розділу

Протягом виконання даного розділу, було побудовано дерево цілей, для усестороннього окреслення сутності системи, що досліджується; побудовано функціональну діаграму IDEF0, для відображення основного процесу, та діаграми IDEF3 для подальшої конкретизації роботи; побудовано ієрархію процесів у дерево-подібнову представлені, що описує основні складові (модулі) інформаційної системи аналізу транзакцій клієнта банку.

РОЗДІЛ 3

Програмні засоби розв'язання задачі

3.1. Вибір та обґрунтування засобів розв'язання задачі

Для безпечної та комфортної розробки програмного продукту потрібно якимось чином забезпечити контроль за кодом, в випадку, якщо щось піде не за планом, мати змогу вернутися до попереднього стану свого коду. З даною проблемою вже роками борються за допомогою систем контролю версій (далі vcs). VCS – це те, що не дасть розпастись вашому проекту, коли над ним працює багато програмістів, це зможе вернути програмний код до попереднього стану. Дані засоби надають такі можливості:

- Збереження даних.
- Синхронізація.
- Скасування дій короткого терміну.
- Скасування дій довгого терміну.
- Відслідковування змін.
- Відслідковування власників змін.
- Розгалуження та злиття збережень.

Як альтернативи можна розглядати такі системи контролю версій, як Git [14] та Mercurial [15]. Ці дві системи контролю версій були практично одночасно випущені у використання. Вони обидві працюють на основі ядра Linux, але Git все ж є більш популярною системою.

Переваги Git:

- Висока швидкість виконання всіх операцій.
- Дешевезна операцій з гілками коду.
- Повна історія розробки доступна в оффлайн режимі.

Недоліки Git:

- Важкий старт, для тих, хто користувався старішими системами.

Переваги Mercurial:

- Легше почати використовувати.
- Хороша документація.

Недоліки:

- Нема можливості злиття двох батьківських гілок.
- Використання плагінів, а не скриптів.
- Менша воля дій.

Так, як **Git** надає більшу волю дій, можливість злиття батьківських гілок та має вищу швидкодію, що допоможе швидше та комфортніше працювати з ним, то йому надається пріоритет. Враховуючи, що був досвід користування саме даною системою, то вибір впав саме на неї.

Так як **Git** представлений свого роду командною строкою, що трохи ускладнює його використання для недосвідчених користувачів, прийнято рішення використовувати допоміжну програму **Sourcetree** [16]. Даний програмний засіб – це візуальний клієнт системи контролю версій, як **Git** так і **Mercurial**, варто зазначити, що він є безплатним, що є вагомим плюсом для звичайного користувача.

Розібравшись з системами контролю версій, потрібно визначитись з мовою програмування, на якій буде реалізована проектована система. Вибір впав на дві дуже популярні мови: **Java** [17] та **C#** [18]. Дані мови по своїй суті є дуже подібним, адже вони обидві входять в нішу Сі-подібних мов програмування. Багато бібліотек **C#** написані на **Java**. Синтаксис надзвичайно схожий. Але, хоч і компанія **Oracle**, яка і є розробником **Java**, продовжує підтримувати та розвивати своє дітище, ця мова має певні обмеження та не дає такої волі дій, як це робить її противник. Так, як **C#** набагато новіша мова і надає більшу волю дій розробнику, саме її було обрано для використання.

Для написання веб-додатків, використовуючи **C#**, можна застосовувати два програмні інструменти: **Microsoft .Net Core** [19] або **Microsoft .Net Framework** [20].

.Net Core варто використовувати, коли:

- Для створення кросплатформених додатки.
- При орієнтації на мікросервіси.

- При використанні контейнерів Docker [21].
- Якщо потрібні великі додатки з високою швидкістю.
- Для створення систем з підтримкою різних версій .Net.

.Net Framework:

- Додаток на даний момент використовує вже .Net Framework.
- Додаток використовує бібліотеки, які не підтримуються .Net Core.

Варто зазначити, що .Net Core набагато новіший продукт компанії Microsoft, саме на нього компанія зараз і робить свій акцент, а старіший продукт (.Net Framework) все більше переходить в звичайну підтримку. Враховуючи те, що **.Net Core** є новішою технологією, яка надає більше можливостей та зручностей, то саме вона буде використана для створення нашої систем.

Визначившись з мовою програмування, можна обрати програмне середовище, в якому буде писатися код. Компанія-розробник мови пропонує два шляхи – це Microsoft Visual Studio [22] або Microsoft Visual Studio Code [23]. Visual Studio Code – це легкий але потужний редактор коду, який працює на вашому комп'ютері, він підтримується Windows, MacOS, Linux. Має вбудовану підтримку JavaScript, TypeScript, Node.js. Якщо в Вас операційна система MacOS, то будьте певні в виборі саме даного редактора коду, адже його альтернатива не підтримується даною системою. Microsoft Visual Studio – це повноцінне IDE, яке є набагато потужнішим за Visual Studio Code, але він не є кросплатформним і підтримується лиш операційною системою Windows. Враховуючи те, що система буде проектуватися на персональному комп'ютері, який працює на операційній системі Windows, то немає причин використовувати більш легшу та обмежену альтернативу, саме тому вибір падає на **Microsoft Visual Studio**, так як дана IDE дає безліч функцій, які можуть бути застосовані в розробці.

Для збереження постійних даних потрібно застосовувати бази даних [24]. Для вирішення даного завдання можна використовувати PostgreSQL [25] – це безплатна база даних, яка вигідно відрізняється від інших комерційних та з відкритим кодом баз даних.

Дана база даних має такі плюси:

- Транзакції.
- Вкладені запити.
- Представлення.
- Цілісність посилань та зовнішні ключі.
- Важкі блокування.
- Типи, які визначаються користувачем.
- Наслідування.
- Правила.
- Перевірка сумісності версій.

Плюсами є й те, що кожна версія проходить процедуру регресивного тестування, яке забезпечує стабільність. За рахунок того, що дана система має відкритий код, то всі помилки вирішуються дуже швидко. Швидкодія даної системи також вражає, останні тестування показали, що вона не поступається в даному аспекті навіть платним комерційним системам. Через те, що код є відкритим, користувачі пропонують нові можливості, що допомагає нарощувати функціонал системи. Доступ додатка до бази даних відбувається за рахунок процесу бази даних. Користувацькі програми не можуть отримати доступ до даних самостійно, навіть якщо вони працюють на том ж комп'ютері, на якому виконується серверний процес. Це забезпечує високий рівень безпеки даних, які зберігаються в даній базі даних.

Таке розділення клієнта та сервера дозволяє побудувати розподілену систему. Можна відокремити клієнта від сервера шляхом мережі та розробляти клієнтський додаток в середовищі, яке зручне для користувача даної системи.

Також можна використати MySQL[26] – це безплатна база даних з відкритим кодом, яка надає ефективне керування базами даних шляхом з'єднання їх з програмними засобами. Вона стабільна та надійна і доволі потужна, що веде за собою результат у вигляді першості за популярністю у світі. Плюси:

- Дешевизна – можна економити шляхом використання даної бази даних, адже вона є безкоштовна.

- Гнучкість відкритого коду.
- Повний контроль робочого процесу.
- Відтримка різноманітних транзакцій.
- Масштабованість при потребі.
- Безпека даних.

В зв'язку з тим, що **MySQL** потребує менше місця на жорсткому диску та вона більш "легша" за альтернативний варіант, було вибрано саме її.

Для зручнішої та продуктивнішої роботи з обраною базою даних підібрано **MySQL Workbench**[27] – являється графічним інструментом для роботи із **MySQL**. Забезпечує легкість виконання різноманітних задач, пов'язаних із базами даних. В ньому вбудовано SQL розробка, адміністрування, дизайн та створення баз даних.

- Тут вбудовано візуальні інструменти для виконання таких функцій:
- Створення та перегляд баз даних.
- Створення та перегляд інших об'єктів баз даних, а саме: моделі, тригери, індекси і т.д.
- Створення, виконання та оптимізація запитів.
- Налаштування серверів.
- Створення копій та відновлення.
- Перегляд статусу сервера.

Docker – це платформа для контейнеризації з відкритим кодом. Даний програмний засіб надає можливість розробникам упаковувати додатки в контейнери, що допомагає використовувати первинний код зі всіма операційними системами, що грає надзвичайно важливу роль при розробці, адже це знімає рамки, щодо операційної системи. Звісно, можна обійтися без даного засобу, але він є набором інструментів, який дає змогу розробнику будувати, розміщувати, запускати, оновлювати та зупиняти контейнери, використовуючи прості команди. Чому

використовуються саме контейнери? Тому що вони включають в себе лише операційні процеси та необхідні залежності, необхідні для виконання коду, але не завантажують з собою екземпляр операційної системи, що робить їх "легкими"; використовуючи контейнери, можна запускати декілька разів стільки копій додатка, скільки б це можна було з робити з використанням віртуальних машин; порівнюючи з віртуальними машинами, контейнери є швидшими і легшими для розміщення їх в інтернеті та перезапуску, що робить їх хорошим вибором для команд розробників, які працюють за методологіями Agile. Плюсами використання Docker є:

- Покращена переносимість – докер-контейнери запускаються без модифікацій щодних ком'ютерів, інформаційних центрів та хмарних середовищ.
- Легкість – декілька процесів можуть бути комбіновані в один єдиний контейнер, це робить можливим побудову додатка, який продовжуватиме працювати навіть тоді, коли одна його частина оновлюється або відлагоджується.
- Автоматичне створення контейнера – Docker може автоматично створити контейнер на основі коду додатка.
- Контроль версій контейнера – є можливість відслідковувати версії контейнера, в разі потреби повертати їх до попередньої версії, бачити, хто створив дану версію, надає змогу завантажувати лише різницю між існуючою версією та новою, що схоже з принципом роботи систем контролю версій.
- Відкриті бібліотеки контейнерів – розробники можуть отримати доступ до відкритого реєстру, в якому зберігаються тисячі контейнерів користувачів.

По цих причинах було обрано саме даний програмний засіб.

Є декілька інструментів, які допомагають працювати з Docker, а саме: DockStation [28], Kitematic [29] та Portainer [30].

DockStation – це потужний та функціональний засіб для роботи з Docker. Суттю є те, що використовуючи даний засіб, використовується графічний інтерфейс, а не

рядкові команди, що полегшує сприйняття того, що відбувається. Перевагами є : робота з Docker Compose, нативна підтримка даної конфігурації, за рахунок чого отримується повноцінна підтримка всього функціоналу Docker. Можливість роботи над віддаленими записами, моніторинг ресурсів та швидка і дружелюбна підтримка. Головним недоліком є закритий код даної системи.

Kitematic – аналогічно попередній альтернативі, надає графічний інтерфейс для роботи з Docker. Являється офіційною власністю компанії Docker, яка викупила її в 2015 році. Основною задачею є управління та конфігурування окремих контейнерів. Недоліками даної системи є те, що вона є незручною при використанні її для роботи з більш ніж одним проектом. Основною перевагою є те, що тут відкритий код та дана система є офіційним продуктом компанії Docker.

Portainer – навідмінну від своїх попередників, дана альтернатива є веб-додатком, який дозволяє проводити налагодження та різні маніпуляції з контейнерами. Перевагами є: можливість повноцінного налаштування проекту, можливість підключення до віддалених записів, перегляд статистики контейнерів. Недоліки: непродуманий та перевантажений інтерфейс, який ускладнює роботу з даною системою, для багатьох робота з веб-додатком може бути менш зручною, ніж десктопні альтернативи, незручний при роботі більш, ніж з одним проектом.

Враховуючи те, що розробка проекрованої інформаційної системи буде відбуватися в одному проекті, це анулює мінуси Kitematic, також, даний засіб є дуже простий та має зрозумілий інтерфейс, який не заважає, навідмінну від Portainer. Відкритий код також є плюсом в сторону даної систему на противагу DockStation. Враховуючи все вище сказане, було обрано **Kitematic**.

3.2. Технічні характеристики обраних програмних засобів розроблення

Для представлення технічних характеристик вибраних програм для розроблення інформаційної систем, було обрано табличний вид.

На Табл.3.1. вказано технічні характеристики програми Git.

Таблиця 3.1.

Технічні характеристики Git

Дата випуску	7 квітня 2005 року
Розробник	Software Freedom Conservancy
Мова на якій написано	Ci, UNIX, Perl, Tcl, Python, C++
Мова інтерфейсу	Англійська
Підтримувані протоколи	Git-протокол, SSH, HTTP, HTTPS
Підтримувані операційні системи	Windows, MacOS, Linux
Підтримувані бітності	32, 64
Місце на жорсткому диску	44.5 МБ

На Табл.3.2. можна переглянути технічні характеристики SourceTree.

Таблиця 3.2.

Технічні характеристики SourceTree

Дата випуску	13 травня 2013 року
Розробник	Atlassian
Мова інтерфейсу	Англійська, російська, німецька, французька, китайська, Іспанська, Японська
Підтримувані протоколи	HTTPS, SSH
Підтримувані операційні системи	Windows 7, 8, 8.1, 10, MacOS
Підтримувані бітності	32, 64
Місце на жорсткому диску	24.3 МБ
Ліцензія	Freeware

На Табл.3.3. вказані технічні характеристики Visual Studio.

Таблиця 3.3.

Технічні характеристики Visual Studio

Дата випуску	2 квітня 2019 року
Розробник	Microsoft
Мова на якій написано	C++, C#

Мова інтерфейсу	Англійська, російська, німецька, французька, китайська, іспанська, японська, чеська, польська, португальська
Підтримувані протоколи	HTTPS, SSH
Підтримувані операційні системи	Windows 7, 8, 8.1, 10
Підтримувані бітності	32, 64
Місце на жорсткому диску	800 МБ, залежить від встановлених компонентів (зазвичай, потрібно від 20 ГБ до 50 ГБ)
Процесор	Тактова частота не нижче 1.8 ГГц, мінімум двоядерний
Оперативна пам'ять	2 ГБ; рекомендується 8 ГБ (мінімум 2.5 ГБ при виконанні на віртуальній машині)
Відеокарта	Мінімальне розширення 720p (1280 на 720 пікселів), для оптимальної роботи потрібно 1366 на 768 пікселів і вище
Додаткові вимоги для встановлення	Права адміністратора. .NET Framework 4.5.2.
Ліцензія	Microsoft EULA

На Табл.3.4. вказано технічні характеристики Docker.

Таблиця 3.4.

Технічні характеристики Docker

Дата випуску	13 березня 2013 року, стабільний випуск з 19 січня 2017 року
Розробник	Solomon Hykes, Docker Inc.
Мова на якій написано	Go
Мова інтерфейсу	Англійська

Підтримувані операційні системи	Windows 10 Pro, Enterprise, Educational, Linux
Підтримувані бітності	64, 32
Місце на жорсткому диску	3 ГБ
Репозиторій	github.com/docker/docker-ce
Оперативна пам'ять	2 ГБ
Інтеграція	Ansible, Chef, Jenkins, Kubernetes, Microsoft Azure, OpenStack Nova, OpenSVC, Puppet, Salt, Travis CI, Vagrant, VMware, Vsphere
Додаткові вимоги для встановлення	Статичний IP-адрес, CS Docker Engine 1.10 або вище, Linux kernel 3.10 або вище
Ліцензія	Apache License 2.0

На Табл.3.5. можна переглянути технічні характеристики Kitematic.

Таблиця 3.5.

Технічні характеристики Kitematic

Дата випуску	2013 рік
Розробник	Docker Inc.
Мова інтерфейсу	Англійська
Підтримувані операційні системи	Windows 7 і вище, MacOS 10.8 і вище
Підтримувані бітності	64
Місце на жорсткому диску	1.3 ГБ
Репозиторій	github.com/docker/docker-ce
Оперативна пам'ять	3 ГБ
Ліцензія	Apache License 2.0

На Табл.3.6. наведені технічні характеристики MySQL Workbench

Технічні характеристики MySQL Workbench

Дата випуску	12 серпня 2013 рік
Розробник	Oracle Corporation
Мова інтерфейсу	Англійська
Підтримувані операційні системи	Кросплатформленість
Написано на	C++, C#, Objective-C
Місце на жорсткому диску	125 МБ
Репозиторій	github.com/mysql/mysql-workbench
Оперативна пам'ять	4 ГБ (рекомендовано 6 ГБ)
Ліцензія	GNU Public License
Процесор	Intel Core, Xeon 3GHz (або Dual Core 2HGz) або еквівалентної потужності процесор
Ядра	Single(Dual/Quad Core рекомендовано)
Відео-карта	nVidia або ATI з підтримкою OpenGL 1.5 і вище
Розширення екрану	1920x1200 рекомендовано

Висновок до третього розділу

Під час виконання третього розділу, було обрано та описано основні програмні засоби для реалізації інформаційної системи аналізу транзакцій клієнта банку, а саме: Git, SourceTree, C#, Microsoft Visual Studio 2019, Microsoft .Net Core 3.0, Docker, MySQL, MySQL Workbench та Kitematic. Також було вказано їхні технічні характеристики за допомогою таблиць.

РОЗДІЛ 4

Практична реалізація

4.1.Опис створеного програмного засобу

Створений програмний засіб використовує базу даних на основі MySQL. Як результат, вона являється реляційною базою даних, тобто, основаною на таблицях та їхніх зв'язках. Вся потрібна інформація зберігається у трьох таблицях, які пов'язані між собою зв'язками, а саме: **expensesfiles**, **transactions** та **users**.

Розглянемо кожну таблицю:

users – дана таблиця зберігає інформацію про всіх користувачів, які були зареєстровані в системі. Вона має такі атрибути:

1. *userid* – це є первинний ключ та унікальний ідентифікатор, за допомогою якого і буде здійснюватися розділення різних записів, його тип INT, тобто, цілочисельне число, також вказано, що це число не може бути від'ємним та автоінкрементується для того, щоб не було можливості генерації однакових ключів;

2. *email* – дане поле призначене для ідентифікації користувача в системі, при авторизації. Тип VARCHAR(45), тобто рядок довжиною не більше сорока п'яти символів, чого достатньо. Також, в разі потреби, можна за допомогою даного атрибута надсилати якісь повідомлення про оновлення на електронну пошту;

3. *password* - даний атрибут потрібен для реєстрації користувача та в подальшому для здійснення автентифікації його в системі. Так само, як і його попередник, має тип VARCHAR(45).

Модель редактора таблиці зображено на Рис.4.1.

expensesfiles – дана таблиця представляє собою дані про файл-виписку про транзакції клієнта. Потрібна для того, щоб користувач мав можливість завантажити декілька виписок і аналізувати їх окремо, а не змішувати чи переписувати поверх транзакції одного файлу на іншого.




Table Name:




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 userid	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рис.4.1. Таблиця users

expensesfiles має три атрибути:

1. *fileid* – первинний ключ даної таблиці, використовується для ідентифікації запису. Тип INT – цілочисельне число, вказано, що лише додатні числа будуть використовуватися для автоінкрементації даного значення;

2. *filepath* – так як в шлях файлу входить і його назва, то користувачу буде легше розрізнити, який саме файл обрано, також, це може слугувати певною підказкою, де шукати первинний файл-джерело аналізу, звісно, якщо він все ще є на тому місці. Тип VARCHAR(45), не може бути значення NULL;

3. *userid* – зовнішній ключ, який має на меті, визначення, який саме користувач завантажив даний файл, і кому він належить. Це потрібно для того, щоб користувачу аналізувались лише його дані, лише з конкретного файлу, а не усі. Тип INT, не може бути NULL, повинен бути лише додатній.

Зображення даної таблиці можна побачити на Рис.4.2.




Table Name:




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 fileid	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 filepath	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 userid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рис.4.2. Таблиця expensesfiles

transactions – в даній таблиці зберігаються усі записи, тобто транзакції, з усіх файлів користувачів. Це і буде основне сховище інформації для аналізу. Тут є 8 атрибутів:

1. *transactionid* – первинний ключ. Тип INT – ціле число, не може бути від'ємним та приймати значення NULL, значення автоінкрементуються;

2. *day* – відображає номер дня місяця, коли була здійснена транзакція. Тип INT – ціле, не від'ємне, не NULL;

3. *month* – номер місяця в який було здійснено транзакцію користувача. Тип INT – ціле, не від'ємне, не NULL;

4. *year* - номер року в який було здійснено транзакцію користувача. Тип INT – ціле, не від'ємне, не NULL;

5. *description* – опис, або певна додаткова інформація про здійснену транзакцію. Тип VARCHAR(100), не може бути NULL;

6. *category* – даний атрибут використовується для зберігання інформації про категорію конкретної транзакції. Тип VARCHAR(45), не NULL;

7. *price* – поле в яке зберігається сума, іншими словами, ціна транзакції. Тип DOUBLE – додатні числа з комою;

8. *fileid* – зовнішній ключ, який використовується для визначення, якому саме документу належить дана транзакція. Тип INT – ціле число, додатнє, не може бути NULL.

Дану таблицю зображено на Рис.4.3..




Table Name:

transactions









Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
 transactionid	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
 day	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 month	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 year	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 description	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 category	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 price	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 fileid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рис.4.3.Таблиця transactions

Модель бази даних (діаграму зв'язків) зображено на Рис.4.4.

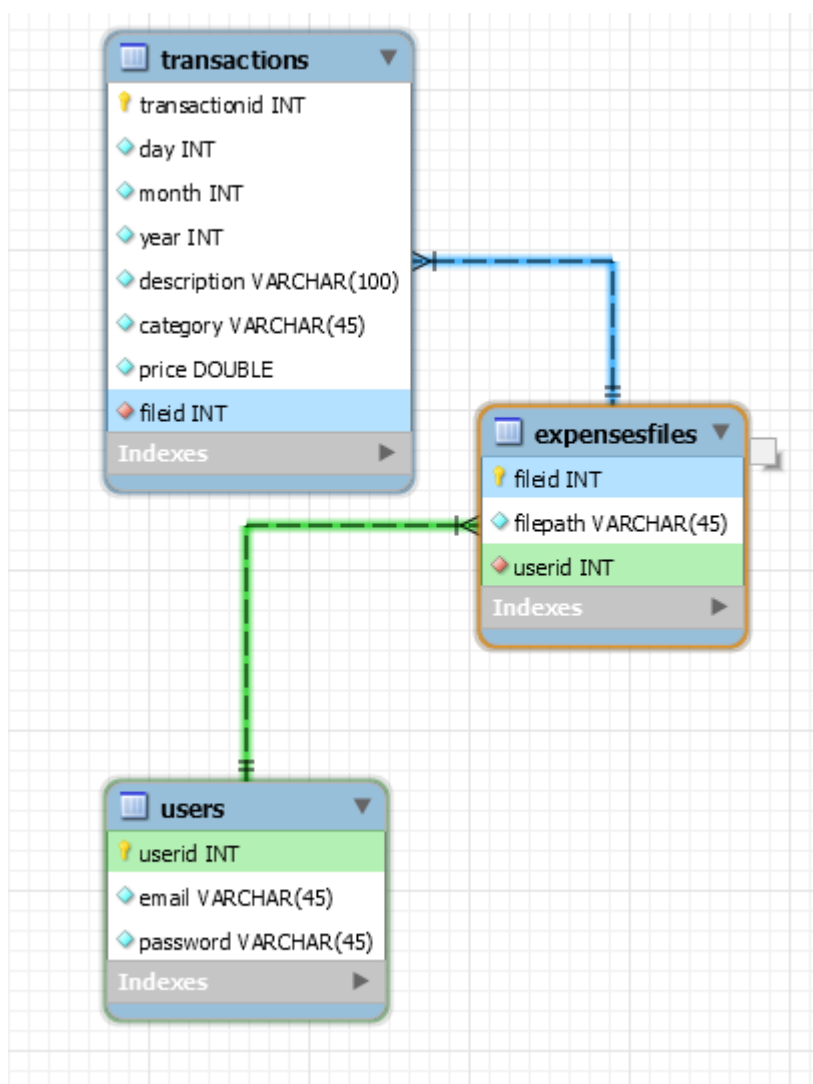


Рис.4.4.Діаграма зв'язків бази даних

Як бачимо таблиці expensesfiles і transactions мають відношення 1 : N, відповідно. А users та expensesfiles – 1 : N, аналогічно.

Котрольний варіант програмного засобу аналізу транзакцій клієнта банку представлений у вигляді проекту, який в свою чергу поділяється на п'ять під-проектів. Їх можна умовно розділити на дві категорії: сервер та клієнт, адже дана програма по своїй суті є кросплатформленою клієнт-серверною системою, що забезпечує можливість використовувати її на різних операційних системах, а саме: Windows, Android, IOS, MacOS, Linux.

На Рис.4.5. зображено загальну структуру проекту.

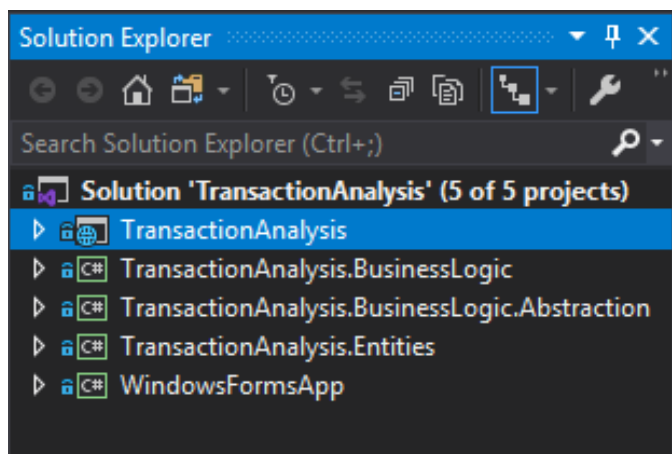


Рис.4.5. Структура проекту

До серверної частини програми відносяться:

TransactionAnalysis - ключовий проект серверної частини (Рис.4.6.), адже саме тут розміщено контроллер який має HTTP запити, а саме:

- **GetDiagramInfo** – реалізація Get запиту, який повертає список об'єктів, які вміщують в собі інформацію про місяць та кількість витрат.
- **GetCategoriesInfo** – Get запит, який вертає список об'єктів із інформацією про категорію та її ціну.
- **GetCategoryTransactions** – Get запит, який список транзакцій, які були зроблені в конкретний місяць і відносяться до конкретної категорії, яка була передана, як параметр.

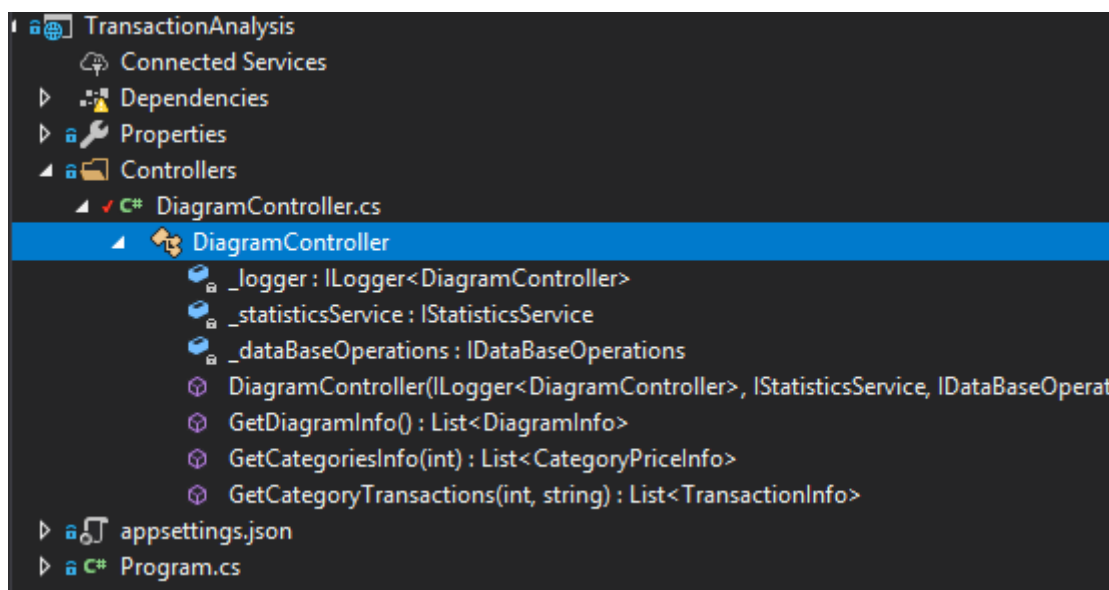


Рис.4.6. Структура TransactionAnalysis

Також, в даному проекті знаходяться важливі файли налаштування серверу:

- Appsettings.json – в даному файлі вказані основні конфігурації проекту.
- Program.cs – тут розміщена вхідна точка (Main функція) з якої все починається.

TransactionAnalysis.BusinessLogic – проект, який відповідає за основну бізнес логіку сервера (Рис.4.7.). Він представлений двома класами:

DataBaseOperations.cs – даний клас надає всі потрібні функції для роботи з базою даних, а саме:

- void LoadRecordToDB(int,int,int,string,double,string) – завантаження нового запису до бази даних;
- string LoadFileToDB(string) – завантаження файлу до бази даних, приймає шлях файлу, як параметр;
- List<TransactionInfo> getCategoryMonthRecords(int,string) – отримання списку транзакцій, які були зроблені в конкретний місяць та відносяться до певної категорії.

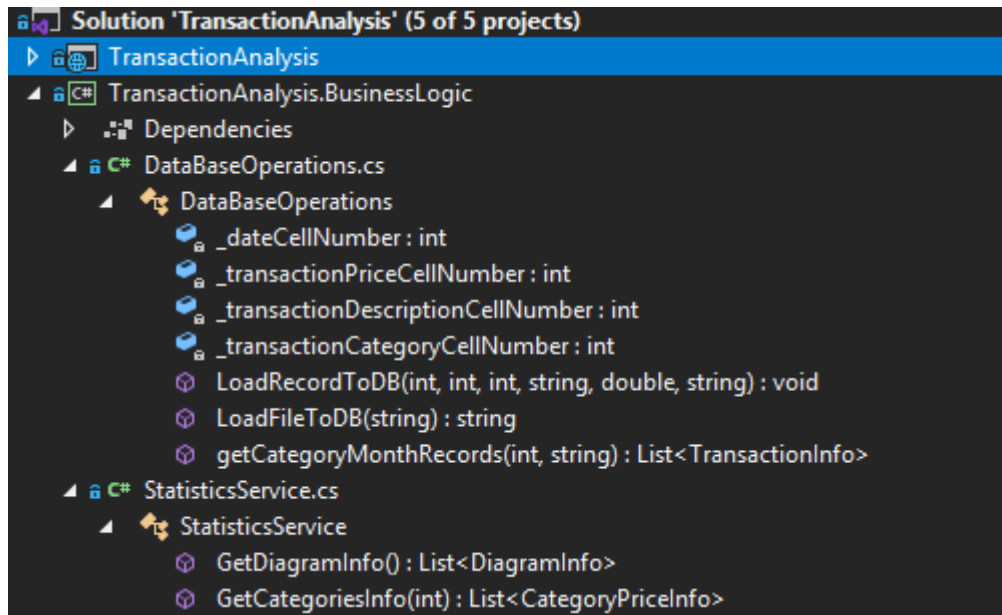


Рис.4.7. Структура TransactionAnalysis.BusinessLogic

Клас має такі поля:

- `_dateCellNumber : int;`
- `_transactionPriceCellNumber : int;`

- `_transactionDescriptionCellNumber : int;`
- `_transactionCategoryCellNumber : int.`

StaticsService.cs – даний клас надає інструменти для отримання потрібної інформації для статистики. Тут є такі методи:

- `GetDiagramInfo() : List<DiagramInfo>` - даний метод відповідає за надання даних по витратах за кожен місяць року.
- `GetCategoriesInfo(int) : List<CategoryPriceInfo>` - надає інформацію про категорії, а саме: кількість грошей, які було витрачено на дану категорію.

TransactionAnalysis.BusinessLogic.Abstraction – даний модуль головного проекту представляє собою інтерфейси для бізнес логіки, що надає можливість, вразі потреби, замінити стару реалізацію на нову, і це не приведе до конфліктів класів. Прикладом реалізації даних інтерфейсів є попередній модуль. Структура даного під-проекту наведена на Рис.4.8.

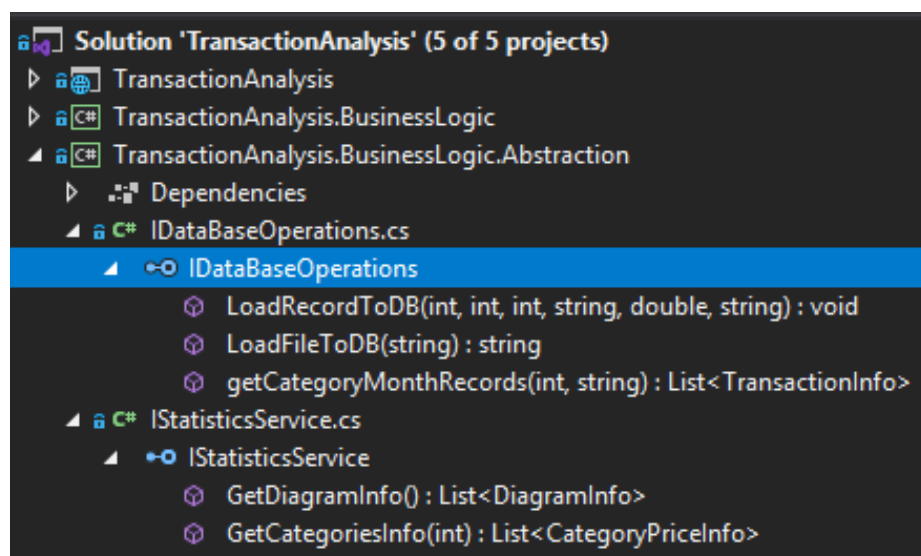


Рис.4.8. Структура *TransactionAnalysis.BusinessLogic.Abstraction*

Отже, даний проект має два класи-інтерфейси:

IDatabaseOperations.cs, який має такі методи:

- `LoadRecordToDB() : void;`
- `LoadFileToDB(string) : string;`
- `getCategoryMonthRecords() : List<TransactionInfo>.`

IStaticsService.cs – даний клас, аналогічно своїй реалізації, яку було описано вище, має два методи:

- `GetDiagramInfo() : List<DiagramInfo>;`
- `GetCategoriesInfo(int) : List <CategoryPriceInfo>.`

TransactionAnalysis.Entities – даний під-проект містить в собі класи, які відповідають за представлення основних сутностей проектованої системи. Структура даного проекту наведена на Рис.4.9.

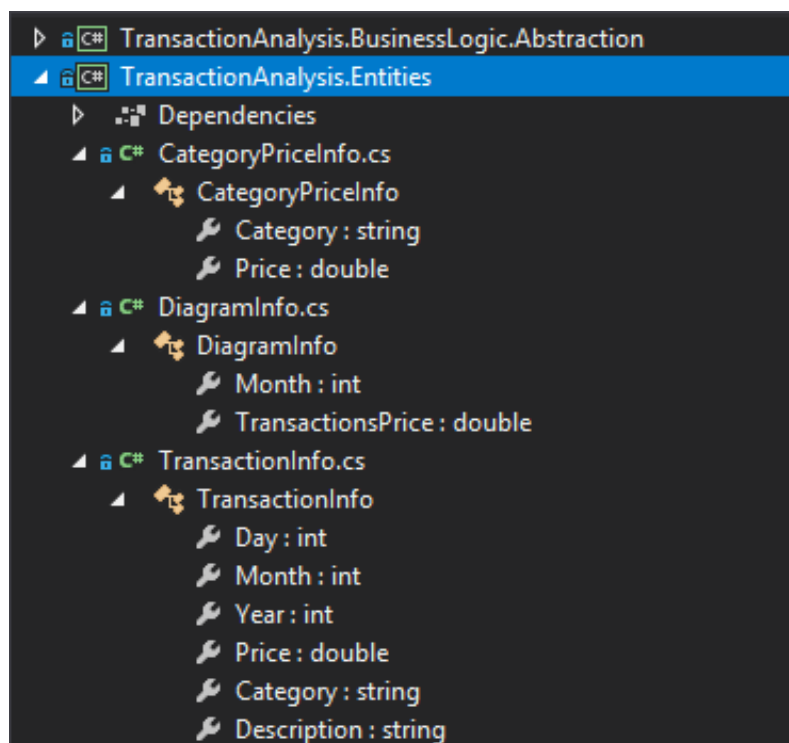


Рис.4.9. Структура *TransactionAnalysis.Entities*

В даному модулі є декілька класів:

CategoryPriceinfo.cs – зберігання інформації щодо назви категорії та її вартості.

Клас має такі поля:

- `Category: string;`
- `Price: double.`

DiagramInfo.cs – відповідає за зберігання даних для діаграми, а саме: номер місяці в році та кількість коштів, які було витрачено за даний місяць. Відповідно, є два поля:

1. `Month: int;`

2. TransactionsPrice: double.

Клієнтська частина представлена клієнтським додатком, який написаний за допомогою Windows форм. Даний проект має назву **WindowsFormsApp**. Структура даного проекту зображена на Рис.4.10.

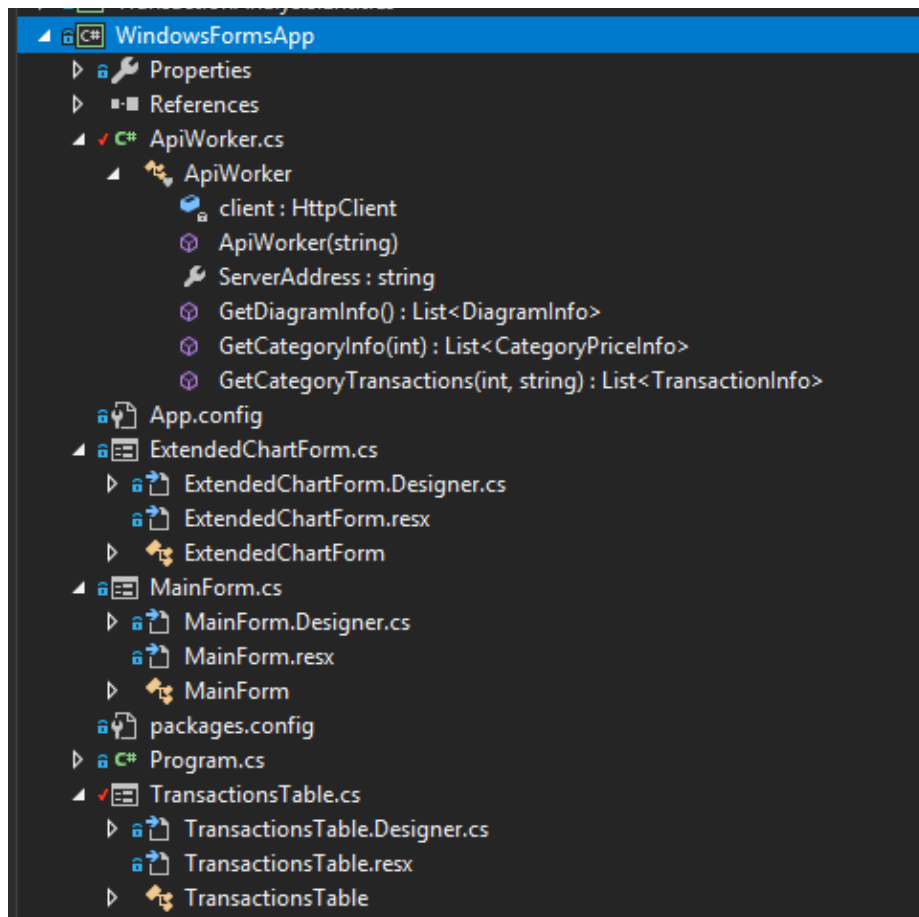


Рис.4.10. Структура WindowsFormsApp

Як можна побачити на рисунку, проект складається з декількох класів.

ApiWorker.cs – даний клас призначений для встановлення зв'язку клієнта із сервером та їхньої взаємодії. Для цього є два поля:

1. client : HttpClient;
2. ServerAdress : string.

Для різноманітної взаємодії із сервером, реалізовано такі методи:

- GetDiagramInfo() : List<DiagramInfo> - відповідає за виклик запиту, який надасть у вигляді відповіді список місяців та кількість витрат на них.

- `GetCategoryInfo(int) : List<CategoryPriceInfo>` - викликає `Get` запит, який повертає список категорій та їхню ціну за вказаний місяць.

- `GetCategoryTransactions(int, string) : List<TransactionList>` - за допомогою заданого методу, програма-клієнт отримує всі транзакції, які були здійснені в конкретний місяць і належать до вказаної категорії.

ExtendedChartForm.cs – даний клас відповідальний за ініціалізацію всіх потрібних елементів та обробку виконуваних дій користувача, які пов'язані із діаграмою витрат по категоріях за конкретний місяць. В класі є три поля:

1. `_monthNumber : int;`
2. `components : IContainer;`
3. `extendedChart : Chart.`

Для організації взаємодії є такі методи:

- `extendedChart_MouseDown(object, MouseEventArgs) : void` – дана функція відповідає за відслідковування координат миші користувача. При нажатті клавiшi миші відбувається перевірка, де саме цей клік був виконаний, в відповідності викликається потрібне вікно;

- `Dospose(bool) : void;`
- `InitializeComponent() : void` – ініціалізація компонентів форми.

MainForm.cs – даний клас відповідає за діаграму витрат по місяцях та взаємодію із нею. Використовуються такі поля:

- `_months : Dictionart<int, string>` - словник, який має в собі відповідності ключ – значення, де ключ – це номер місяця, а значення – його назва;
- `_apiWorker : ApiWorker;`
- `components : IContainer;`
- `openFileDialog : OpenFileDialog;`
- `tabPageHistogramms : TabPage;`
- `tableLayoutPanel : TableLayoutPanel;`
- `chart1-12 : Chart;`

- tabPageLoad : TabPage;
- btnUploadFile : Button;
- lblChoosenFilePath : Label;
- lblSelectFile : Label;
- btnSelectFile : Button;
- tabControl : TabControl.

Ці всі поля використовуються наступними методами:

- GetMonthNumber(string) : int – даний метод приймає назву діаграми в якій вказано номер місяця, який вона відображає, і витягує цей номер.
- GetMonthName(int) : string – функція приймає номер місяця і, за допомогою словника, визначає його назву.
- btnLoadFile_Click(object, EventArgs) : void – відповідає за відслідковування нажаття кнопки вибору завантажуваного файлу.
- btnUploadFile_Click(object, EventArgs) : void – відповідає за відслідковування нажаття кнопки завантаження обраного файлу.
- Form1_Load(object, EventArgs) : void – завантаження початкової форми.
- chart_Click(object, EventArgs) : void – відслідковування нажаття на елемент гістограми.

Program.cs – клас який містить в собі точку входу (метод Main).

TransactionsTable.cs – даний клас відповідає за роботу із таблицею транзакцій, які були здійснені в обраний місяць за певною категорією. Використовуються наступні поля:

- Components : IContainer;
- dataGridView1 : DataGridView;
- Day : DataGridViewTextBoxColumn;
- Month : DataGridViewTextBoxColumn;
- Year : DataGridViewTextBoxColumn;
- Category : DataGridViewTextBoxColumn;

- Price : DataGridViewTextBoxColumn;
- Description : DataGridViewTextBoxColumn.

В даному класі використовуються такі методи:

- dataGridView1_CellContentClick(object, DataGridViewCellEventArgs) : void – дана функція відповідає за відслідковування нажаття на вміст стовбця, що потенційно дає змогу відсортувати наявні записи в таблиці по величині ціни транзакцій;
- InitializeComponent() : void – ініціалізація компонентів таблиці.

Як можна було побачити, веб-сервер та клієнт-додаток спілкуються за допомогою Http запитів, програми побудовані на концепції ООП [31].

4.2.Інструкція користувача

4.2.1.Вступ

Розроблений програмний засіб має на меті полегшення аналізу транзакцій клієнта банку, що в свою чергу забезпечує економію часу користувачеві даної системи. Враховуючи те, що кожне юридичне лице може мати банківський рахунок, то дана система може бути використана не тільки в побуті людей, а й в бізнесі.

Користувачеві не потрібно нічого налаштовувати. Щодо вхідних даних, то ними є лише файл-виписки транзакцій з банку.

4.2.2.Загальні відомості про програму

Програма складається з двох частин, які комунікують з собою сервера та клієнта. Програма - сервер носить назву "TransactionsAnalysis", а клієнт – "FinAnalitics". Програму написано на мові C# з застосуванням фреймворку Microsoft .Net Core 3.1., Windows Forms та мови MySQL для здійснення запитів до бази даних. Програма може бути застосована як в побутовому житті так і в бізнесі.

4.2.3.Класи вирішуваних завдань

Даний програмний засіб націлений на такі завдання:

- Завантаження виписки з транзакціями.
- Візуалізація витрат за кожен місяць.
- Візуалізація місячних витрат з виділенням категорій та визначенням найбільш затратної категорії.
- Відображення транзакцій здійснених в конкретний місяць і які відповідають конкретній категорії.

4.2.4.Опис основних характеристик і особливостей програми

Даний програмний продукт розрахований сервісний режим роботи, тобто, немає потреби такої, щоб він працював цілодобово. Мета в тому, щоб користувач при потребі міг обрати файл-виписку та отримати аналіз по ньому. Щодо зменшення часу виконання задачі при використанні даної програми, то оцінити точно неможливо, адже кожна людина б робила аналіз власноруч різний період часу, який залежить від досвіду, навичок та об'єму аналізованої інформації, але можна точно сказати, що використання програми точно зменшить час, який потрібно витратити, у рази, а інколи і у десятки разів.

4.2.5.Відомості про функціональні обмеження на застосування

Основними умовами, які є обов'язковими до виконання, є наявність в користувача доступу до глобальної мережі Інтернет, так як взаємодіє клієнта із сервером здійснюється шляхом Http протоколу, то відсутність з'єднання унеможливило використання даного продукту. Також не менш важливим фактором є наявність виписки витрат із банку, а саме: ПриватБанку, адже функції аналізу заточені саме під структуру даної виписки. Щоб отримати таку виписку потрібно перейти по посиланню - <https://privat24.privatbank.ua/p24/web/#statements/cards/> та авторизуватися. Після цього відкриється таке вікно, як на Рис.4.11.

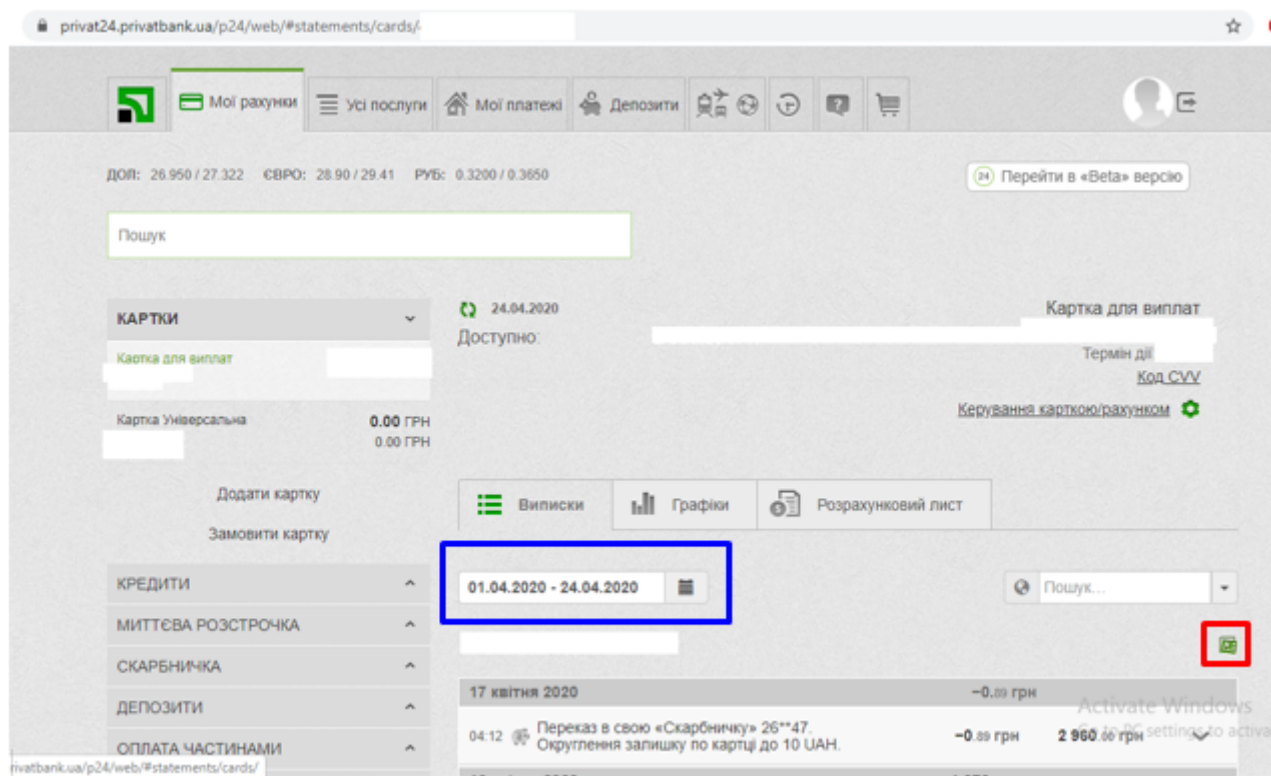


Рис.4.11.Інтерфейс сайту ПриватБанк

Для вибору часових рамок, за які було здійснено транзакції, потрібно обрати початкову та кінцеву дату в полі, яке виділене на даному рисунку синім прямокутником. Після вибору дат, потрібно натиснути кнопку, яка виділена на рисунку червоним прямокутником. Файл-виписки автоматично завантажиться на ваш персональний пристрій.

Говорячи про веб-сервер, який аналізує дані, то він є кросплатформлений, тобто, потенційно даний програмний продукт може підтримувати сумісність із різними операційними системами. Якщо говорити про наведений приклад додатка-клієнта, то він підтримується операційною системою Windows 8 та вище. З Windows 7 можуть виникати проблеми через відсутність підтримки даної версії операційної системи компанією Microsoft.

4.3.Аналіз контрольного прикладу

Як доказ, того, що реалізований програмний продукт працює як слід, наведено результати виконання роботи.

Якщо користувач не зареєстрований, то спочатку він реєструється в системі, після чого проходить авторизацію в ній. Після чого появляється вікно Рис.4.12., на якому користувачу пропонується обрати файл, який слугуватиме початковим джерелом інформації, щодо транзакцій.

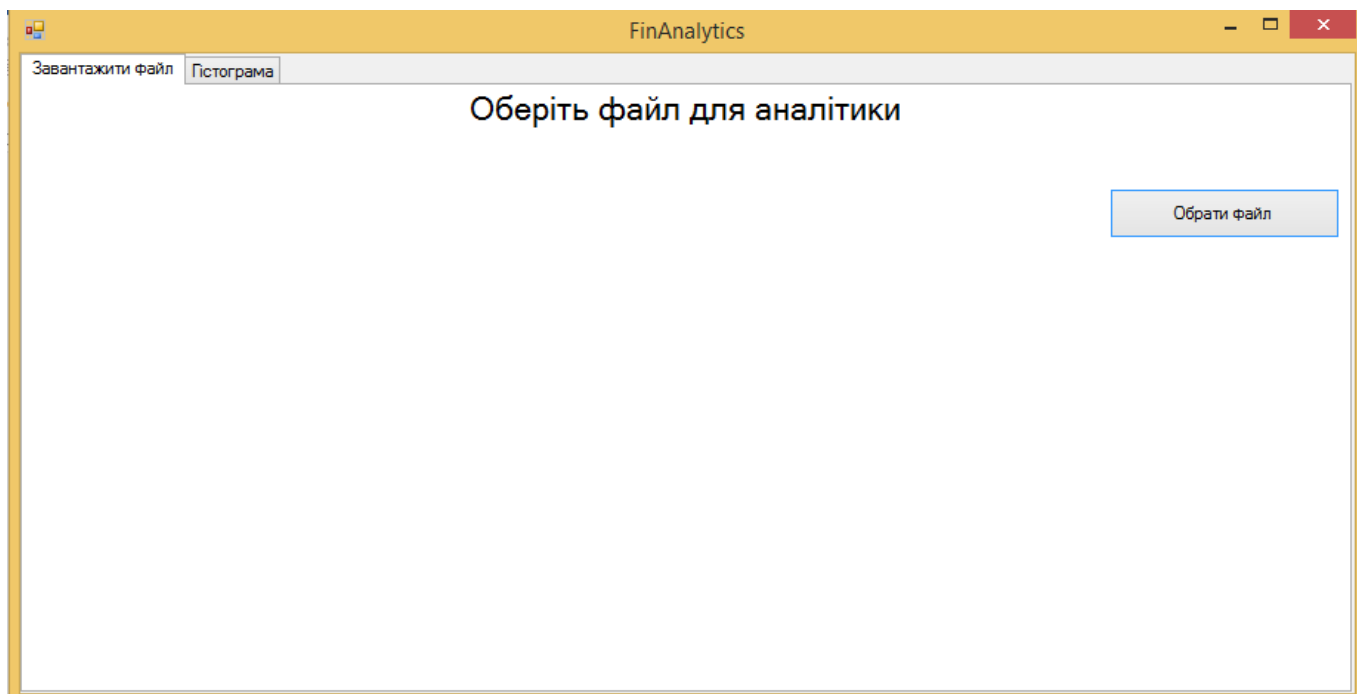


Рис.4.12.Вікно вибору файлу

Для того, щоб обрати файл, потрібно натиснути кнопку "Обрати файл", після чого появиться файловий менеджер операційної системи Windows Рис.4.13. Тут вибирається файл. Після обрання файлу появляється кнопка "Завантажити файл", нажаття якої, є підтвердженням завантаження файлу в базу даних системи. Після завантаження обраного файлу, потрібно перейти на вікно "Гістограма", яке виділене червоним прямокутником на Рис.4.14.

В зв'язку з тим, що виписка з банку несла інформацію, щодо витрат, в собі, лише за два місяці (квітень та березень), то й проведення статистики відбувається лише по цих місяцях, що видно на Рис.4.14.

Наведені гістограми показують назву місяця та суму, яку було витрачено за даний період. Також вони є клікабельні.

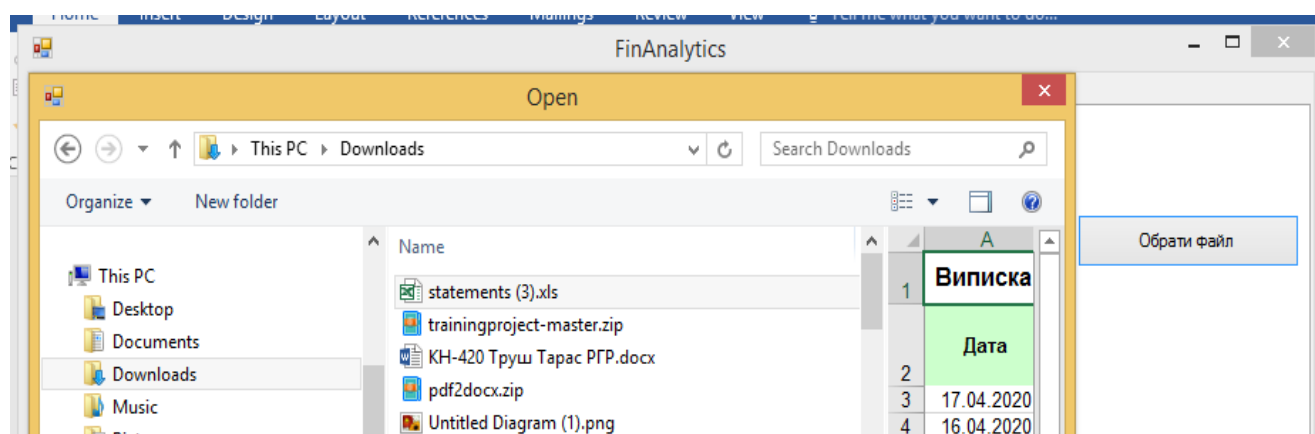


Рис.4.13.Вибір файлу-виписки

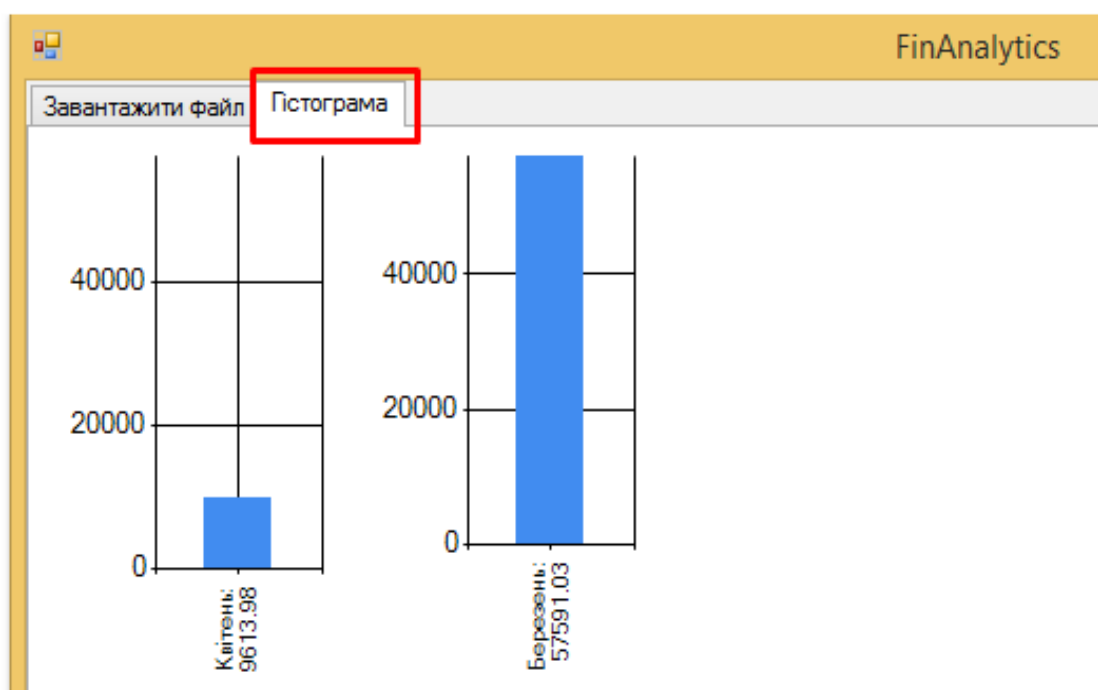


Рис.4.14.Гістограма місячних витрат

При натисканні на конкретний стовбець, відкривається вікно з більш детальною інформацією, що і зображено на Рис.4.15. Як ми можемо бачити, у вікні подані дані, щодо категорій витрат та конкретні суми, скільки було витрачено грошей на цю категорію за даний місяць (в наведеному випадку це березень). Ця інформація подана у вигляді піраміди, де кожен сегмент є відображенням категорії, чим вищий сегмент, тим більш затратна категорія (сегменти розміщені з низу до гори за спаданням затратності).

Також є можливість отримання списку всіх транзакцій по конкретній категорії. Для цього просто потрібно натиснути лівою кнопкою миші на одну з категорій витрат, які перелічено у правому верхньому куті Рис.4.15.

В наведеному прикладі було обрано категорію "Кафе, бари, ресторани", після чого відкрилося вікно з таблицею транзакцій по цій категорії, яке зображено на Рис.4.16.

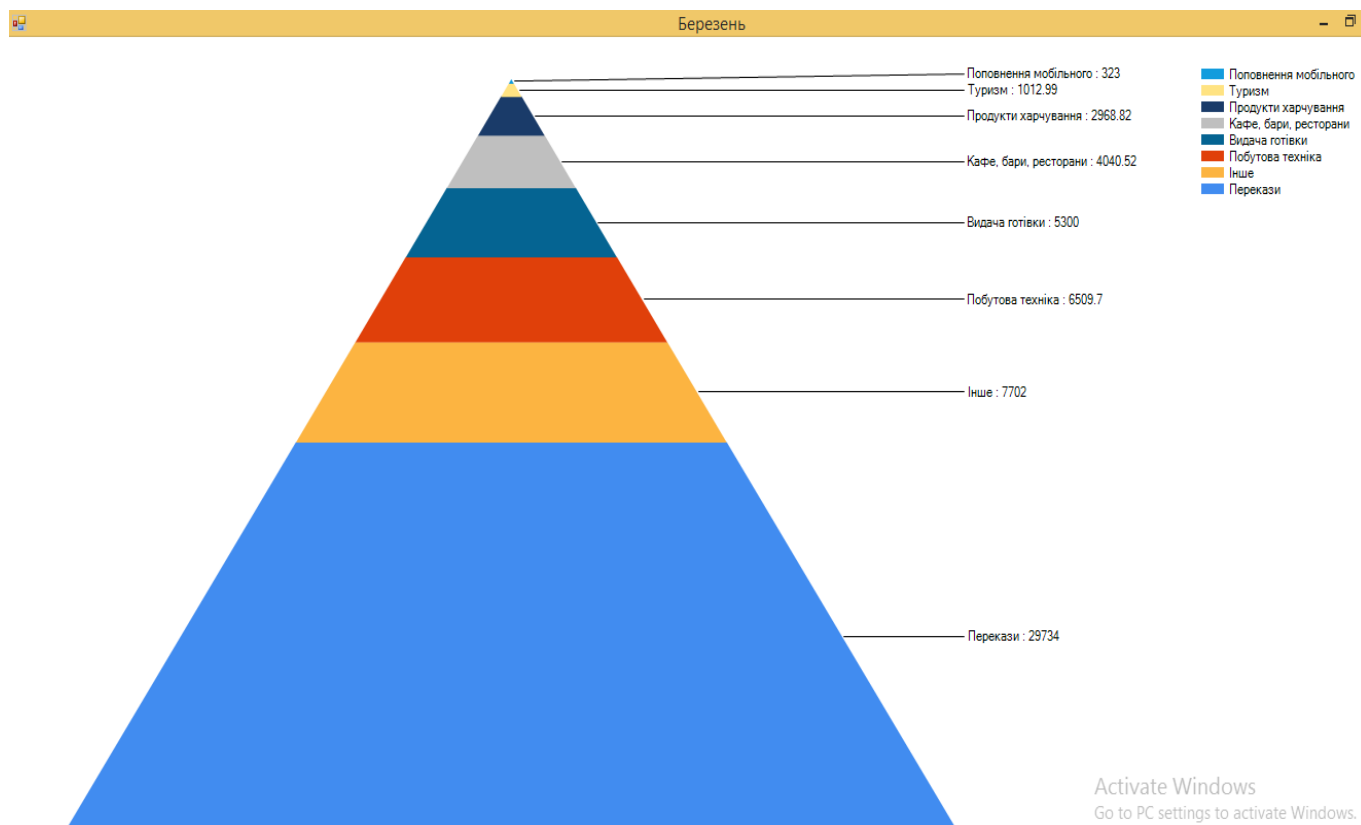
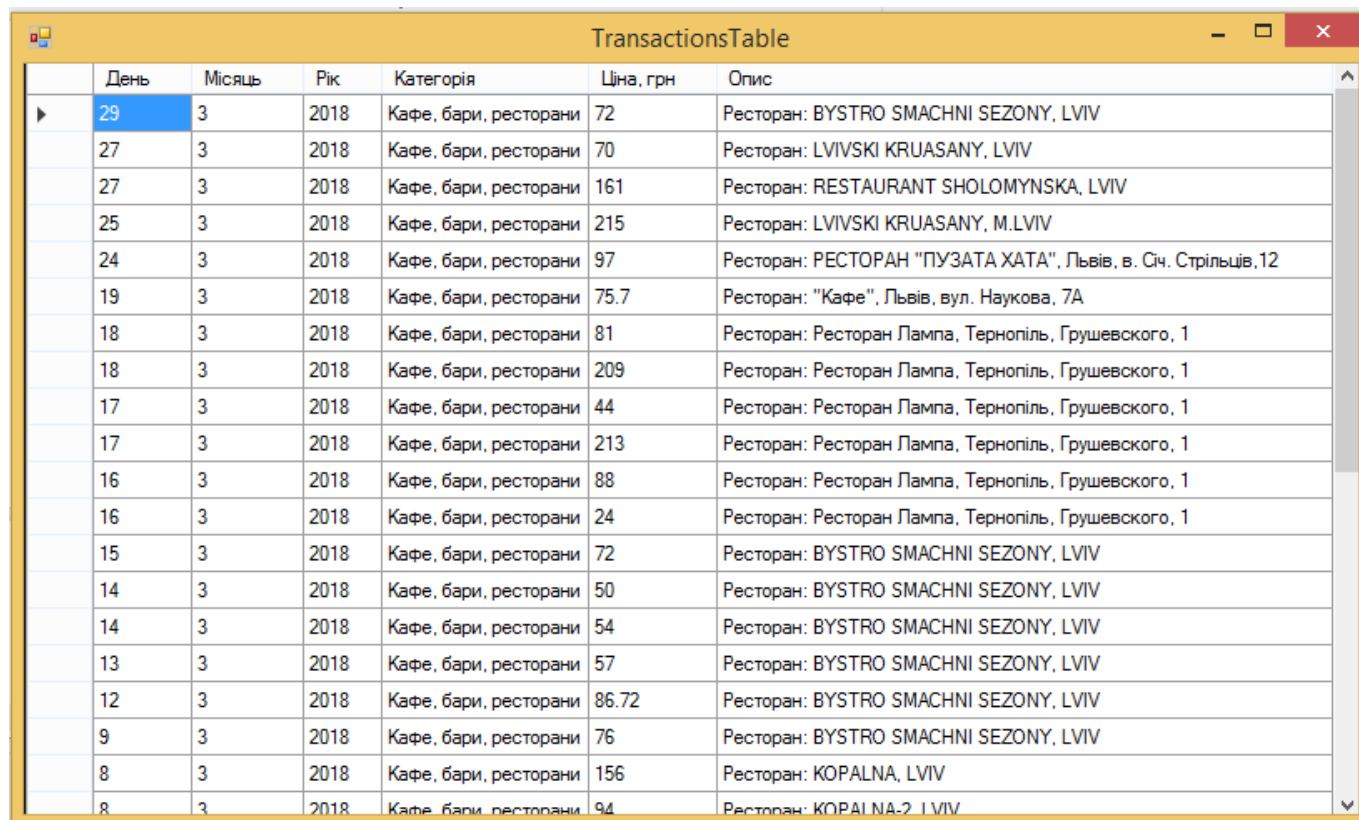


Рис.4.15.Витрати розділені по категоріях за конкретний місяць

Як видно на рисунку, ми отримуємо таблицю з записами, які є тими самими транзакціями по обраній категорії. Тут ми бачимо інформацію по таких полях:

- День;
- Місяць;
- Рік;
- Категорія;
- Ціна;
- Опис.

Є можливість відсортувати записи по деяких числових атрибутах, а саме: "День", "Місяць", "Рік" та "Ціна".



День	Місяць	Рік	Категорія	Ціна, грн	Опис
29	3	2018	Кафе, бари, ресторани	72	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
27	3	2018	Кафе, бари, ресторани	70	Ресторан: LVIVSKI KRUASANY, LVIV
27	3	2018	Кафе, бари, ресторани	161	Ресторан: RESTAURANT SHOLOMYNSKA, LVIV
25	3	2018	Кафе, бари, ресторани	215	Ресторан: LVIVSKI KRUASANY, M.LVIV
24	3	2018	Кафе, бари, ресторани	97	Ресторан: РЕСТОРАН "ПУЗАТА ХАТА", Львів, в. Січ. Стрільців, 12
19	3	2018	Кафе, бари, ресторани	75.7	Ресторан: "Кафе", Львів, вул. Наукова, 7А
18	3	2018	Кафе, бари, ресторани	81	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
18	3	2018	Кафе, бари, ресторани	209	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
17	3	2018	Кафе, бари, ресторани	44	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
17	3	2018	Кафе, бари, ресторани	213	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
16	3	2018	Кафе, бари, ресторани	88	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
16	3	2018	Кафе, бари, ресторани	24	Ресторан: Ресторан Лампа, Тернопіль, Грушевського, 1
15	3	2018	Кафе, бари, ресторани	72	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
14	3	2018	Кафе, бари, ресторани	50	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
14	3	2018	Кафе, бари, ресторани	54	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
13	3	2018	Кафе, бари, ресторани	57	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
12	3	2018	Кафе, бари, ресторани	86.72	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
9	3	2018	Кафе, бари, ресторани	76	Ресторан: BYSTRO SMACHNI SEZONY, LVIV
8	3	2018	Кафе, бари, ресторани	156	Ресторан: KOPALNA, LVIV
8	3	2018	Кафе, бари, ресторани	94	Ресторан: KOPALNA-2, LVIV

Рис.4.16.Список транзакцій

Висновок до четвертого розділу

В ході виконання даного розділу було описано створений програмний засіб для аналізу транзакцій клієнта банку, базу даних даного продукту та її таблиці, і зв'язки між ними. Також було наведено детальну інформацію про структуру програмного проекту, який складається з п'яти модулів, умовно поділених на серверну та клієнтську сторони. Описані їхній функції та їхнє призначення.

Написано інструкцію для користувача, для полегшеного використання розробленого програмного продукту. Інструкцію оформлено так, як того вимагає стандарт IEEE STD 1063-2001 "Standard for Software User Documentation".

Проаналізовано роботу розробленого програмного засобу з наведенням пояснень та знімків екрану виконуваної програми. Робота виконується коректно.

РОЗДІЛ 5

Економічне обґрунтування доцільності роботи

5.1. Економічна характеристика програмного продукту

Темою бакалаврської роботи було обрано інформаційну систему аналізу транзакцій клієнта банку. Дана система розрахована як для власного використання у вигляді відслідковування та аналізу власних витрат зроблених в межах ПриватБанку, так і для бізнесу, який користується послугами даного банку та має на меті отримувати статистику по чисельних транзакціях. Проаналізувавши процес ручного перегляду та виведення статистики по виписці з банку, було прийнято рішення щодо створення системи, яка значно спростить процес відслідковування власних витрат, що зекономить користувачам безліч часу, який вони можуть витратити на більш приємніші для них речі.

Дана інформаційна системи відповідає всім вимогам, які ставлять сучасні замовники, а саме:

- цільова аудиторія;
- економія часу користувача;
- просування продукту;
- управління даними.

На сьогоднішній день схожі системи є актуальними і набирають все більшого попиту серед людей, адже значка кількість послуг переходить в цифровий формат, люди все частіше використовують власні смартфони або банкові картки для оплати різноманітних послуг на противагу користування паперовими грошима, наприклад: оплата комунальних послуг, таксі, покупки товарів в інтернеті, поповнення мобільного рахунку і безліч інших речей. Існуючі системи, які працюють з потребою аналізу цих всіх транзакцій є не кросплатформленими або не можуть бути використані клієнтами ПриватБанку.

Основними факторами, що впливають на економічну успішність розробки і впровадження цього продукту є високий попит, а також зручність у використанні.

5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту

До шляхів вирішення поставленої задачі належать не лише інформаційні, а й інтелектуальні системи, які базовані на нейронних мережах та машинному навчанні. Це доволі сучасні та складні методології, які дають свої плоди для банків-гігантів та інших комерційних компаній, які працюють в галузях пов'язаних з аналізом даних про покупки клієнтів. Але такі системи є закритими і недоступні для звичайного користувача. По суті, звичайним користувачам є достатнім і функціонал, який надають простіші системи, які не використовують нейронних мереж. Саме тому, схожі шляхи вирішення і будуть аналізуватись.

Після дослідження таких платформ, як Google Play та App Store, яка базована для пристроїв, у яких операційною системою є IOS, а також глобальної мережі Інтернет, було сформовано вибірку популярних інформаційних систем, які використовуються для вирішення поставленої проблеми. Найбільш популярними та стабільними системами є : monobank, ZenMoney, Monefy. Говорячи про monobank, варто зазначити, що це мобільний додаток для банкінгу, але він також і надає можливість перегляду статистики по власних витратах, що підходить під вирішення поставленої мети. ZenMoney і Monefy - це повноцінно заточені на проведення статистики по транзакціях користувача. Отож, дані три системи було взято для подальшого аналізу поставленої проблеми. Після аналізу перелічених систем було наведено їхні переваги та недоліки. Враховуючи кількість користувачів даних систем, можна стверджувати, що створення інформаційної системи для аналізу транзакцій клієнта банку є актуальним на сьогодні.

Так як ПриватБанк є Державним банком України, банком, який має найбільшу кількість користувачів в нашій країні, можна точно сказати, що в розроблюваної системи є мільйони потенційних користувачів, що говорить про великий ринок збуту та економічну доцільність, та потенційний успіх розроблення системи аналізу транзакцій клієнтів банку.

5.3.Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ

Даний розділ передбачає оцінювання та аналіз факторів зовнішнього та внутрішнього середовищ групою експертів.

Фактори зовнішні оцінюються за шкалою $[-5;5]$, при цьому межі шкали відображають максимальний негативний та позитивний вплив факторів на організацію, 0 демонструє, що фактор впливає на організацію нейтрально.

Фактори внутрішні оцінюються за шкалою $[0;5]$, при цьому 0 демонструє нерозвинутість, відсутність чи катастрофічний стан фактора внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактора.

Сума вагомостей усіх факторів становить одиницю, тобто рівень вагомості для кожного фактора визначається за допомогою коефіцієнтів. Зважений рівень впливу факторів розраховується як добуток впливу фактора у балах та рівня вагомості. Результати експертних оцінок впливу факторів зовнішнього середовища наведено у табл. 5.1.

Таблиця 5.1

Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ

Фактори	Середня експертна оцінка, бали	Середня вагомість факторів	Зважений рівень впливу, бали
1	2	3	4
<i>Фактори зовнішнього середовища</i>			
Споживачі	5	0,11	0,55
Постачальники	0	0,10	0,00
Конкуренти	-3	0,10	-0,30
Державні органи влади	0	0,05	0,00
Інфраструктура	0	0,06	0,00
Законодавчі акти	0	0,10	0,00

Продовження табл.5.1

Профспілки, партії та інші громадські організації	0	0,05	0,00
Система економічних відносин в державі	0	0,06	0,00
Організації-сусіди	4	0,01	0,04
Міжнародні події	0	0,01	0,00
Міжнародне оточення	0	0,03	0,00
Науково-технічний прогрес	5	0,07	0,35
Політичні обставини	-2	0,06	-0,12
Соціально-культурні обставини	-2	0,05	-0,10
Рівень техніки та технологій	5	0,04	0,20
Особливості міжнародних економічних відносин	-1	0,02	-0,02
Стан економіки	-2	0,08	-0,16
Загальна сума		1	0,44
<i>Фактори внутрішнього середовища</i>			
Цілі	4	0,11	0,44
Структура	4	0,16	0,64
Завдання	4	0,07	0,28
Технологія	4	0,20	0,80
Працівники	3	0,21	0,63
Ресурси	4	0,25	1,00
Загальна сума		1	3,79

Отож, зважений рівень впливу зовнішнього середовища є позитивним та становить 0,44. Найбільш позитивно впливають на продукт споживачі, організації сусіди (розвиток яких призводить до збільшення користувачів даного продукту), науково технічний прогрес та рівень техніки та технологій (ці фактори напряду впливають на якість продукту та також їхній розвиток веде за собою переведення речей в цифровий вид). Найбільш негативними факторами є конкуренти, які

потенційно можуть забирати користувачів, стан економіки та політичні обставини в країні, адже низький рівень економіки веде за собою низький рівень заробітку, а це призводить до зменшення попиту на всі продукти. Сумарний зважений рівень впливу внутрішнього середовища становить 3,79.

З цього можна судити, що на ринку є потреба в створенні системи, яка б робила процес аналізу власних витрат максимально простим та швидким у використанні, як для звичайних користувачів так і для бізнесу.

5.4. Формування стратегічних альтернатив

У даному розділі необхідно здійснити вибір за першою та другою групою альтернативних стратегій розвитку (Рис.5.1 та Рис.5.2), та обґрунтувати їхню доцільність для даного програмного продукту (проектного рішення).

Перша група стратегічних альтернатив (Рис.5.1).



Рис.5.1. Стратегічні альтернативи першої групи

Критеріями поділу альтернативних стратегій розвитку є існуючий продукт (програмне забезпечення) та новий, а також супутні послуги.

Стратегія розроблення нового продукту (проектного рішення) характеризується створенням абсолютно нового програмного забезпечення, яке дає змогу вирішити новоутворені потреби людини, суспільства, економіки тощо.

Стратегія розвитку існуючого продукту (проектного рішення) означає модифікацію програмного забезпечення, його якісних характеристик.

Стратегія розвитку існуючого продукту (проектного рішення) з супутніми послугами означає пропонування на ринку модифікованого програмного забезпечення із додатковими послугами (встановлення, супроводження, коригування, адаптування до специфіки конкретного підприємства тощо).

Стратегія нового продукту (проектного рішення) з супутніми послугами означає розроблення нового програмного забезпечення та пропонування при його експлуатації додаткових послуг.

Друга група стратегічних альтернатив (Рис.5.2). Критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт.



Рис.5.2. Стратегія альтернативи другої групи

Глибше проникнення на ринок полягає в використанні існуючого продукту (проектного рішення) для збільшення частки на існуючому ринку. Якщо фірма

володіє достатніми ресурсами та потужностями для виготовлення існуючого продукту, то ця стратегія є найменш ризикованою. Однак, активно зростання на існуючому ринку призведе до зростання конкуренції. Стратегія буде успішною за умови обмежень у ресурсах та потужностях конкурентів або стрімкому розвитку самого ринку. Слід зазначити, що кожен ринок за обсягом має свій ліміт і якщо підприємство прагнучим розвиватись, то воно повинно використовувати інші запропоновані стратегії.

Стратегія розвитку ринку полягає в використанні існуючого продукту (програмного забезпечення) або незначній його модифікації для виходу на новий сегмент ринку, весь ринок або іноземний ринок. Ця стратегія є з вищим рівнем ризику, оскільки необхідно виходити на новий ринок, де можуть бути інші правила гри, вимоги та смаки споживачів тощо.

Стратегія розвитку продукту полягає у створенні нового продукту (програмного забезпечення) для існуючого сегменту ринку. Ця стратегія є досить ризиковою, оскільки вимагає створення нового продукту (програмного забезпечення) для існуючого сегменту споживачів. Однак, якщо ринок починає зменшувати обсяги та існуючий продукт є на етапі зрілості та падіння, тоді доцільно застосовувати стратегію розвитку продукту.

Стратегія диверсифікації реалізується шляхом виходу на нові сфери бізнесу. Тобто розширення номенклатури товарів, послуг тощо.

Переглянувши першу групу стратегічних альтернатив, я обираю *стратегію розроблення нового продукту з супутніми послугами*, тому що реалізую нову систему, яка полягає у розробленні методів та засобів аналізу даних по витратах клієнтів ПриватБанку.

З другої групи стратегічних альтернатив, я обираю *стратегію розвитку продукту*, з огляду на те що створюю нову систему для існуючого кола споживачів.

5.5. Бюджетування

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов'язаних з виготовленням та реалізацією продукту, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій. Бюджет матеріальних витрат наведено в табл.5.2.

Таблиця 5.2

Бюджет витрат матеріалів та комплектуючих виробів

Назва матеріалів та комплектуючих	Марка, тип, модель	Фактична кількість, шт.	Ціна за одиницю, грн.	Разом, грн.
Носії даних	Kingstone DTSE9 32GB	2	188,0	376,0
Периферійні пристрої	Mouse Logitech M185 WL Swift Grey	3	499,0	1497,0
Маршрутизатор	TP-LINK TL-WR841N	1	700,16	700,06
Гарнітура	Logitech Stereo Headset H110	3	499,0	1497,0
Разом:				4070,06

Для виконання розробки потрібні троє працівників:

- Старший програміст (СП);
- Молодший програміст (МП);
- Дизайнер (Д).

Витрати на оплату праці:

- Місячний оклад старшого програміста складає 77990 грн/міс;
- Місячний оклад молодшого програміста складає 13002 грн/міс;
- Місячний оклад дизайнера проекту складає 21450 грн/міс.

Отримаємо наступні значення денної заробітної плати працівників

(з розрахунку на 22 робочих днів у місяці):

- $C_{СП} = 77990 / 22 = 3545$ грн/день;
- $C_{МП} = 13002 / 22 = 591$ грн/день;
- $C_{Д} = 21450 / 22 = 975$ грн/день.

Розрахунок витрат на оплату праці працівника і-тої спеціальності обчислюємо за наступною формулою:

$$З_i = n * t * C, \quad (5.1)$$

де n – кількість працівників і-тої спеціальності, люд;

t – час, що витрачений працівником і-тої спеціальності, дні;

C – денна зарплата працівника і-тої спеціальності, грн.

За допомогою формули (5.1), обчислюємо витрати для оплати праці:

- $З_{СП} = 1 * 14 * 3545 = 49630$ грн.
- $З_{МП} = 1 * 17 * 591 = 10047$ грн.
- $З_{Д} = 1 * 12 * 975 = 11700$ грн.

Сумарні витрати на оплату праці обчислюються за допомогою наступної формули:

$$З = \sum_i^n З_i, \quad (5.2)$$

де $З_i$ – витрати на оплату праці працівника і-тої спеціальності.

За допомогою формули (5.2), обрахуємо сумарні витрати на оплату праці:

$$З = 49630 + 10047 + 11700 = 71377 \text{ грн.}$$

Розрахунок витрат на оплату праці показано у табл.5.3.

Таблиця 5.3

Бюджет витрат на оплату праці

Посада, спеціальність	Кількість працівників, осіб	Час роботи, дні	Денна заробітна плата працівників, грн.	Сума витрат на оплату праці, грн.
<i>Основна заробітна плата</i>				
Старший програміст	1	14	3545,0	49630,0

Продовження табл. 5.3

Молодший програміст	1	17	591,0	10047,0
Дизайнер	1	12	975,0	11700,0
Разом:				71377,0

Визначаємо суму єдиного соціального внеску для кожної категорії працівників:

$$\text{ЄСВ}_{\text{СП}} = 49630 * 0,22 = 10918,6 \text{ (грн)}$$

$$\text{ЄСВ}_{\text{МП}} = 10047 * 0,22 = 2210,34 \text{ (грн)}$$

$$\text{ЄСВ}_{\text{Д}} = 11700 * 0,22 = 2574,0 \text{ (грн)}$$

Бюджет обов'язкових відрахувань та податків наведений в табл. 5.4.

Таблиця 5.4

Бюджет обов'язкових відрахувань та податків

Посада, спеціальність	Сума основної заробітної плати	Сума додаткової заробітної плати	Разом витрат на оплату праці	Сума єдиного внеску на соціальне страхування*, грн.
Старший програміст	17725,0	-	49630,0	10918,6
Молодший програміст	10047,0	-	10047,0	2210,34
Дизайнер	11700,0	-	11700,0	2574,00
Разом:	71377,0	-	71377,0	15702,94

*нарахування єдиного внеску на соціальне страхування відповідає діючій ставці на момент виконання бакалаврської роботи 22%;

Розрахунок загальновиробничих витрат наведено у табл. 5.5.

Таблиця 5.5

Бюджет загальновиробничих витрат

Статті витрат	Сума, грн.
<i>Змінні загальновиробничі витрати, у т.ч.</i>	
- заробітна плата допоміжного персоналу;	14000,0
- витрати на МШП;	1400,0
- витрати на електроенергію та технологічні цілі;	920,0
Разом змінних витрат:	16320,0

<i>Постійні загальновиробничі витрати, у т.ч.</i>	
- комунальні послуги;	1250,0
- витрати на оренду;	12000,0
- інші постійні витрати;	19800,0
Разом постійних витрат:	33050,0
<i>Разом загальновиробничих витрат:</i>	49370,0

Розрахунок адміністративних витрат та витрат на збут відображено у табл.5.6.

Таблиця 5.6

Бюджет адміністративних витрат та витрат на збут

Статті витрат	Сума, грн.
1	2
<i>Адміністративні витрати, у т.ч.</i>	
- витрати на відрядження;	850,0
- витрати на МШП;	800,0
- витрати на сплату податків і зборів;	1700,0
-інші адміністративні витрати;	600,0
Разом адміністративних витрат:	3950,0
<i>Витрати на збут, у т.ч.</i>	
- витрати на рекламу;	1000,0
Разом витрат на збут:	1000,0

Всі отримані вище дані зафіксовано у табл.5.7 зведеного кошторису на розробку проектного рішення.

Таблиця 5.7

Зведений кошторис витрат на розробку проектного рішення

Статті витрат	Разом, грн.
Матеріали та комплектуючі вироби	4070,06
Основна заробітна плата	71377,0

Додаткова заробітна плата	-
Відрахування на соціальне страхування	15702,94
Загальновиробничі витрати, у т.ч.	
- змінні;	16320,0
- постійні;	33050,0
<i>Разом виробничих витрат:</i>	140520,0
Адміністративні витрати	3950,0
Витрати на збут	1000,0
Інші операційні витрати	-
<i>Разом виробничих і операційних витрат:</i>	145470,0

Для визначення фінансових результатів, необхідно розрахувати вартість (ціну) програмного продукту, який розробляється. Ціна визначається на основі суми виробничих і операційних витрат з врахуванням рентабельності виробництва.

$$Ц = СБ * Р + СБ, \quad (5.3)$$

де $Ц$ – ціна одиниці продукту, грн.

$СБ$ – собівартість продукту, грн.

$Р$ – рентабельність виробництва, %.

При рентабельності 33% вартість програмного продукту буде становити:

$$Ц = 140520 * 0,33 + 140520 = 186892 \text{ грн.}$$

$$\text{Дохід від реалізації продукції: } 186892 * 1 = 186892 \text{ грн.}$$

$$\text{Податок на додану вартість: } 186892 * 0.2 = 37379 \text{ грн.}$$

$$\text{Чистий дохід від реалізації продукції: } 186892 - 37379 = 149513 \text{ грн.}$$

$$\text{Валовий прибуток: } 149513 - 140520,0 = 8993 \text{ грн.}$$

$$\text{Фінансовий результат від операційної діяльності: } 8993 - 4950 = 4043 \text{ грн.}$$

$$\text{Податок на прибуток: } 4043 * 0,18 = 728 \text{ грн.}$$

$$\text{Чистий прибуток: } 4043 - 728 = 3315 \text{ грн.}$$

Всі обрахунки, які були проведені вище, можна побачити у табл.5.8.

Бюджет фінансових результатів

Показники	Сума, грн.
1	2
Дохід від реалізації продукції (1 шт)	186892,0
Податок на додану вартість (20%)	37379,0
Чистий дохід від реалізації продукції	149513,0
Собівартість реалізованої продукції	140520,0
Валовий прибуток	8993,0
Операційні витрати:	
- адміністративні витрати:	3950,0
- витрати на збут:	1000,0
- інші операційні витрати:	-
Фінансовий результат від операційної діяльності	4043
Податок на прибуток (18%)	728
Чистий прибуток (збиток)	3315

Як свідчать дані таблиці, чистий прибуток підприємства для розробки кроссплатформленої системи аналізу транзакцій клієнта банку становить 3315 грн. Даний прибуток можна пояснити тим, що проект є короткотривалим (лише один місяць), а одже не є витратним і, відповідно, недорогим. Варто зазначити й те, що за рахунок цього, замовник витратить набагато менше коштів.

Висновок до п'ятого розділу

Завершивши експертне оцінювання факторів зовнішнього та внутрішнього середовищ, ми визначили, що зважений рівень впливу факторів зовнішнього виробництва складає 0.44, а внутрішнього виробництва – 3.79. Отже, розробка кроссплатформленої системи аналізу транзакцій клієнта банку буде мати потенціал на ринку мобільних додатків та сайтів.

Під час етапу формування стратегічних альтернатив, для першої групи ми обрали стратегію розробки нового продукту з супутніми послугами через те, що ПЗ

буде створюватись з «нуля». Щодо другої групи, то було прийнято стратегію розвитку продукту, що являє розробку нового програмного продукту для існуючого кола користувачів.

У роботі ми рахували кількість витрат на виготовлення та реалізацію задуму. Загальна сума виробничих витрат склала 140520 грн. Окрім цього, також присутні адміністративні витрати (3950 грн.) та витрати на збут (1000 грн.).

На основі проведених обчислень та рентабельності на рівні 33%, було визначено вартість програмного продукту, що складає 186892 грн. Чистий прибуток від реалізації проекту складатиме 3315 грн.

Беручи до уваги все вищеописане, можна зробити висновок, що розробка даного програмного продукту є доцільною і економічно виправданою.

ВИСНОВКИ

Можна точно сказати, що кожна повнолітня людина має банківську карточку. А так, як в наші дні все більше і більше речей переходить в цифровий вид, то ми все менше тримаємо в руках паперові гроші, надаючи перевагу оплаті різноманітних послуг більш зручному методу розрахунків. Саме тому тема онлайн-банкінгу стає надзвичайно актуальною кожній дорослій людині. Говорячи про ПриватБанк, то він є найбільшим банком в Україні, що забезпечує велику кількість потенційних користувачів. Варто розуміти, що ніхто не бажає витратити час на підрахунки власних витрат – це нудно та довго. З огляду на це, актуальною є задача створення інформаційної системи аналізу транзакцій клієнта банку.

В результаті написання бакалаврської кваліфікаційної роботи було реалізовано інформаційну систему моніторингу терміну придатності продуктів харчування в цифровому домі. Реалізація складалась з п'яти етапів: аналітичний огляд літературних та інших джерел, системний аналіз об'єкта дослідження, вибір та обґрунтування засобів розв'язання задачі, практична реалізація та економічне обґрунтування доцільності роботи.

В результаті першого етапу були проаналізовані найбільш ефективні та популярні шляхи вирішення поставленої задачі, а саме: сайт ПриватБанку, Excle, Monefy, Дзен-мані та monobank. Як висновок, можна стверджувати, що кожна з перелічених систем має свої недоліки, з якими користувачі не хотіли б стикатися під час їхнього використання.

На другому етапі було здійснено проектування інформаційної системи, шляхом побудови дерева цілей, яке відображає поетапне досягнення головної мети. Була побудована функціональна діаграма IDEF0, яка відображає основний процес та діаграми IDEF3 для подальшої конкретизації роботи. В результаті було сформовано дерево задач, яке описує основні модулі та функції, які мають бути у інформаційній системі аналізу транзакцій клієнта банку.

Під час третього етапу була проаналізована множина технічних засобів для розробки кроссплатформених додатків під різні операційні системи. В результаті

аналізу цієї множини було обрано необхідні засоби для реалізації проектованої системи, а саме: система контролю версій Git, Sourcetree, мову програмування C#, середовище для розробки Visual Studio 2019, фреймворк для розробки кросплатформлених програм .Net Core 3.0, базу даних MySQL та MySql Workbench для комфортнішої роботи з нею.

В результаті четвертого етапу було проведено аналіз контрольного прикладу та визначено, що програмний продукт є робочим та відповідає поставленій задачі, аналізуючи транзакції клієнта банку. Було здійснено опис створеного програмного продукту відповідно до стандарту ГОСТ 19.402-78 “Описание программы”, згідно з яким в результаті було описано загальні відомості про програмний продукт, його функціональне призначення, логічну структуру, використовувані технічні засоби, виклик та завантаження, вхідні та вихідні дані. Було написано інструкцію користувача у відповідності до стандарту IEEE STD 1063-2001 “Standard for Software User Documentation”, згідно з яким було описано класи вирішуваних завдань, основні характеристики та особливості програми та відомості про її функціональні обмеження.

В економічній частині бакалаврської кваліфікаційної роботи обґрунтовано доцільність розробки нового програмного продукту, розраховано всі його економічні характеристики. Період, протягом якого можливо здійснити розробку програмного продукту та запровадження його в дію, складає 22 дні. Для стратегічного планування з першої групи стратегічних альтернатив було обрано стратегію розроблення нового продукту з супутніми послугами, тому що реалізую нову систему, яка полягає у розробленні методів та засобів аналізу транзакцій клієнтів ПриватБанку. З другої групи стратегічних альтернатив, було обрано стратегію розвитку продукту, з огляду на те що створюю нову систему для існуючого кола споживачів. Проаналізувавши основні техніко-економічні характеристики нового ПП та аналогу, виявлено, що розроблений продукт враховує недоліки існуючого, і є більш оптимальним. Зробивши всі необхідні розрахунки, встановлено, що технологія розробки програмного продукту відповідає оптимальному рівню витрат, і, в кінцевому підсумку,

розроблений продукт є економічно доцільним та конкурентоспроможним для впровадження. На основі проведених розрахунків та рентабельності 33% визначено ціну 186892 грн. В результаті реалізації отримаємо чистий прибуток 3315 грн.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Банківські операції [текст]: навч. посіб. / Н.І.Демчук, О.В. Довгаль, Ю.П.Владика.– Дніпро: Пороги, 2017. – С. 7.
2. Виписки банку, їх перевірка та обробка [Електронний ресурс]. – Режим доступу до ресурсу: <https://studfile.net/preview/5079629/page:2>.
3. Транзакции: чтобы все прошло гладко [Електронний ресурс]. – Режим доступу до ресурсу: prostobank.ua/denezhnye_perevody/stati/tranzaktsii_chtoby_vse_proshlo_gladko.
4. Introduction to XML [Електронний ресурс]. – Режим доступу до ресурсу: https://www.w3schools.com/xml/xml_what.asp.
5. Аналіз даних [Електронний ресурс]. – Режим доступу до ресурсу: <https://prometheus.org.ua/dataanalysis/>.
6. Офіційний сайт ПриватБанку [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.privat24.ua/>.
7. Introduction to XML [Електронний ресурс]. – Режим доступу до ресурсу: <https://products.office.com/uk-ua/excel>.
8. Офіційний сайт Microsoft [Електронний ресурс]. – Режим доступу до ресурсу: <https://products.office.com/uk-ua/previous-versions/microsoft-excel-2007>.
9. Офіційний сайт Monefy [Електронний ресурс]. – Режим доступу до ресурсу: <http://www.monefy.me/>.
10. Офіційний сайт Дзен-мани [Електронний ресурс]. – Режим доступу до ресурсу: <https://zenmoney.ru/>.
11. Офіційний сайт monobank [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.monobank.ua/>.
12. Катренко А. В. Системний аналіз / Анатолій Васильович Катренко. – Львів: Новий Світ-2000, 2009. – 396 с.
13. Катренко А. В. Системний аналіз / А. В. Катренко, В. В. Пасічник. – Львів: Новий Світ-2000, 2011. – 396 с. – (Комп'ютинг).

14. Офіційний сайт Git [Електронний ресурс]. – Режим доступу до ресурсу: <https://git-scm.com/>.
15. Офіційний сайт Mercurial [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.mercurial-scm.org/>.
16. Офіційний сайт Sourcetree [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.sourcetreeapp.com/>.
17. Офіційний сайт Java [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.java.com/ru/>.
18. Офіційний сайт C# [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>.
19. Офіційний сайт Microsoft .Net Core [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.microsoft.com/ru-ru/dotnet/core/>.
20. Офіційний сайт Microsoft .Net Framework [Електронний ресурс]. – Режим доступу до ресурсу: <https://dotnet.microsoft.com/download/dotnet-framework>.
21. Офіційний сайт Docker [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.docker.com/>.
22. Офіційний сайт Microsoft Visual Studio [Електронний ресурс]. – Режим доступу до ресурсу: <https://visualstudio.microsoft.com/ru/>.
23. Офіційний сайт Microsoft Visual Studio Code [Електронний ресурс]. – Режим доступу до ресурсу: <https://code.visualstudio.com/>.
24. Верес О. М. Системи баз даних та знань / О. М. Верес, В. В. Пасічник, А. Ю. Берко. – Львів: Магнолія, 2006. – 584 с.
25. Офіційний сайт PostgreSQL [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.postgresql.org/>.
26. Офіційний сайт MySQL [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.mysql.com/>.
27. Офіційний сайт MySQL Workbench [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.mysql.com/products/workbench/>.

28. Офіційний сайт DockStation [Електронний ресурс]. – Режим доступу до ресурсу: <https://dockstation.io/>.
29. Офіційний сайт Kitematic [Електронний ресурс]. – Режим доступу до ресурсу: <https://kitematic.com/>.
30. Офіційний сайт Portainer [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.portainer.io/>.
31. Кравець П. О. Об'єктно-орієнтоване програмування / Петро Олексійович Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.

ANNOTATION

This thesis contains the research, development and implementation of the information system for the analysis of a customer bank transaction. Revealed main concepts of research branch. Analyzed the most popular ways of the problem solving. It is found significant disadvantages of these systems what were fixed in the own one.

With the help of methods of system analysis, the information system for the analysis of a customer bank transaction has been researched and designed. The tree of goals was built for presenting step by step the process of achieving of the main goal. Also, it was created the functional diagram IDEF0, what shows the main process in the system. IDEF3 diagrams were created for the workflow presenting. As a result, it was created a hierarchy of processes in the tree view what shows us the main functions of the system.

Before developing of the system, it was selected software tools, programs and programming languages for the best user experience during the developing and using of the system in the future. Its technical characteristics, advantages and disadvantages were compared with other ones.

During the implementation of the system, it was used C# programming language with its framework for the cross-platform programming called Microsoft .Net 3.0, it is very fast and at the same time easy to use. Also there were used another programs like MySQL WorkBench for comfort using of the MySQL what provides almost all operations with databases. The product that was created as a result is an information system for the analysis of a customer bank transaction in the form of a cross-platform application. For this system, a description was created that includes information about the functional purpose, the logical structure, the technical means that were used, the processes of call and download, input and output data, the interface, etc.

ДОДАТКИ

Додаток А

Коди файлів програмного забезпечення

Файл "DiagramController.cs"

```
using System;
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System.IO;
using NPOI.HSSF.UserModel;
using NPOI.SS.UserModel;
using System.Collections.Generic;
using TransactionAnalysis.BusinessLogic.Abstraction;
using MySql.Data.MySqlClient;
using TransactionAnalysis.Entities;

namespace TransactionAnalysis.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class DiagramController : ControllerBase
    {
        private readonly ILogger<DiagramController> _logger;
        private readonly IStatisticsService _statisticsService;
        private readonly IDatabaseOperations _databaseOperations;

        public DiagramController(ILogger<DiagramController> logger, IStatisticsService statisticsService,
            IDatabaseOperations databaseOperations)
        {
            _logger = logger;
            _statisticsService = statisticsService;
            _databaseOperations = databaseOperations;
        }

        [HttpGet("info")]
        public List<DiagramInfo> GetDiagramInfo()
        {
            return _statisticsService.GetDiagramInfo();
        }

        [HttpGet("categories/{monthNumber}")]
        public List<CategoryPriceInfo> GetCategoriesInfo(int monthNumber)
        {
            return _statisticsService.GetCategoriesInfo(monthNumber);
        }

        [HttpGet("categorytransactions/{monthNumber}/{categoryName}")]
    }
```

```

public List<TransactionInfo> GetCategoryTransactions(int monthNumber, string categoryName)
{
    var tmp = _dataBaseOperations.getCategoryMonthRecords(monthNumber, categoryName);
    return tmp;
}
}
}

```

ФайлD "DatabaseOperations.cs"

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Text;
using TransactionAnalysis.BusinessLogic.Abstraction;
using NPOI.HSSF.UserModel;
using NPOI.SS.UserModel;
using System.IO;
using TransactionAnalysis.Entities;

namespace TransactionAnalysis.BusinessLogic
{
    public class DataBaseOperations: IDatabaseOperations
    {
        private const int _dateCellNumber = 0;
        private const int _transactionPriceCellNumber = 7;
        private const int _transactionDescriptionCellNumber = 4;
        private const int _transactionCategoryCellNumber = 2;

        public void LoadRecordToDB(int day, int month, int year, string description, double price, string category)
        {
            string cs = @"server=localhost;userid=Taras;password=tbтарас16;database=transactionanalysis";
            string query = "insert into transactionanalysis.transaction(day,month,year,description,price,category) values('" + day + "','" + month + "','" + year + "','" + description + "','" + price + "','" + category + "')";

            MySqlConnection connection = new MySqlConnection(cs);
            MySqlCommand command = new MySqlCommand(query, connection);

            connection.Open();
            command.ExecuteReader();
            connection.Close();
        }

        public string LoadFileToDB(string filePath)
        {
            IWorkbook book;

            try
            {
                FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
            }
        }
    }
}

```

```

book = new HSSFWorkbook(fs);

var sheet = book.GetSheetAt(0);

for (int i = 2; i < sheet.LastRowNum; i++)
{
    var row = sheet.GetRow(i);
    var dateCell = row.GetCell(_dateCellNumber);
    var transactionPriceCell = row.GetCell(_transactionPriceCellNumber);
    var transactionCategoryCell = row.GetCell(_transactionCategoryCellNumber);
    var transactionDescriptionCell = row.GetCell(_transactionDescriptionCellNumber);

    DateTime transactionDate = DateTime.ParseExact(dateCell.ToString(), "dd.MM.yyyy", null);

    LoadRecordToDB(transactionDate.Day, transactionDate.Month, transactionDate.Year,
transactionDescriptionCell.StringCellValue.Replace("''", "`"), transactionPriceCell.NumericCellValue,
transactionCategoryCell.StringCellValue.Replace("''", "`"));
}
}
catch (Exception ex)
{
    return ex.ToString();
}

return null;
}

public List<TransactionInfo> getCategoryMonthRecords(int month, string category)
{
    List<TransactionInfo> result = new List<TransactionInfo> ();

    try
    {
        string cs =
@"server=localhost;userid=Taras;password=tbtaras16;database=transactionanalysis";
        string query = "SELECT * FROM transaction WHERE month = " + month + " AND category = " + category + """;

        MySqlConnection connection = new MySqlConnection(cs);
        MySqlCommand command = new MySqlCommand(query, connection);

        connection.Open();

        MySqlDataReader dataReader = command.ExecuteReader();

        while (dataReader.Read())
        {
            TransactionInfo tmpObject = new TransactionInfo();

            result.Add(new TransactionInfo()

```

```

        {
            Day = dataReader.GetInt32(1),
            Month = dataReader.GetInt32(2),
            Year = dataReader.GetInt32(3),
            Description = dataReader[4].ToString(),
            Price = dataReader.GetDouble(5),
            Category = dataReader[6].ToString()
        });
    }

    return result;
}
catch (Exception ex)
{
    return null;
}
} }

```

Файл "StatisticsService.cs"

```

using System;
using System.Collections.Generic;
using TransactionAnalysis.BusinessLogic.Abstraction;
using MySql.Data.MySqlClient;
using TransactionAnalysis.Entities;

namespace TransactionAnalysis.BusinessLogic
{
    public class StatisticsService : Abstraction.IStatisticsService
    {
        public List<DiagramInfo> GetDiagramInfo()
        {
            List<DiagramInfo> yearStatisticsInfo = new List<DiagramInfo>();

            try
            {
                string cs =
@"server=localhost;userid=Taras;password=tbтарас16;database=transactionanalysis";
                string query = "SELECT month, SUM(price) FROM transaction GROUP BY month";

                MySqlConnection connection = new MySqlConnection(cs);
                MySqlCommand command = new MySqlCommand(query, connection);

                connection.Open();
                MySqlDataReader dataReader = command.ExecuteReader();

                while (dataReader.Read())
                {
                    yearStatisticsInfo.Add(new DiagramInfo() { Month = dataReader.GetInt32(0),
TransactionsPrice = dataReader.GetDouble(1) });
                }
            }
            catch { }
        }
    }
}

```

```

        return yearStatisticsInfo;
    }
    catch (Exception ex)
    {
        return null;
    }
}

public List<CategoryPriceInfo> GetCategoriesInfo(int month)
{
    List<CategoryPriceInfo> monthCategoriesStatisticsInfo = new List<CategoryPriceInfo>();

    try
    {
        string cs =
@"server=localhost;userid=Taras;password=tbтарас16;database=transactionanalysis";
        string query = "SELECT category, SUM(price) FROM transaction WHERE month = " + month
+ " GROUP BY category";

        MySqlConnection connection = new MySqlConnection(cs);
        MySqlCommand command = new MySqlCommand(query, connection);

        connection.Open();
        MySqlDataReader dataReader = command.ExecuteReader();

        while (dataReader.Read())
        {
            monthCategoriesStatisticsInfo.Add(new CategoryPriceInfo() { Category =
dataReader[0].ToString(), Price = dataReader.GetDouble(1) });
        }

        return monthCategoriesStatisticsInfo;
    }
    catch (Exception ex)
    {
        return null;
    }
}
}

```

ФайлD "IDataBaseOperations.cs"

```

using System;
using System.Collections.Generic;
using System.Text;
using TransactionAnalysis.Entities;

namespace TransactionAnalysis.BusinessLogic.Abstraction
{
    public interface IDataBaseOperations

```

```

{
    void LoadRecordToDB(int day, int month, int year, string description, double price, string category);

    string LoadFileToDB(string filePath);

    List<TransactionInfo> getCategoryMonthRecords(int month, string category);

}
}

```

ФайлD "IStatisticsService.cs"

```

using System;
using System.Collections.Generic;
using TransactionAnalysis.Entities;

namespace TransactionAnalysis.BusinessLogic.Abstraction
{
    public interface IStatisticsService
    {
        List<DiagramInfo> GetDiagramInfo();

        List<CategoryPriceInfo> GetCategoriesInfo(int month);

    }
}

```

ФайлD "CategoryPriceInfo.cs"

```

using System;
using System.Collections.Generic;
using System.Text;

namespace TransactionAnalysis.Entities
{
    public class CategoryPriceInfo
    {
        public string Category { get; set; }

        public double Price { get; set; }
    }
}

```

ФайлD "DiagramInfo.cs"

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

```



```
namespace TransactionAnalysis
{
    public class DiagramInfo
    {
        public int Month { get; set; }

        public double TransactionsPrice { get; set; }
    }
}
```

ФайлD "DiagramInfo.cs"

```
using System;
using System.Collections.Generic;
using System.Text;

namespace TransactionAnalysis.Entities
{
    public class TransactionInfo
    {
        public int Day { get; set; }

        public int Month { get; set; }

        public int Year { get; set; }

        public double Price { get; set; }
        public string Category { get; set; }

        public string Description { get; set; }
    }
}
```

ФайлD "ApiWorker.cs"

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using TransactionAnalysis;
using TransactionAnalysis.Entities;

namespace WindowsFormsApp
{
    class ApiWorker
    {
        private static readonly HttpClient client = new HttpClient();
    }
}
```

```

//https://localhost:44378/
public ApiWorker(string serverAddress)
{
    ServerAddress = serverAddress;
}

public string ServerAddress { get; }

public List<DiagramInfo> GetDiagramInfo()
{
    var uri = $"{ServerAddress}/diagram/info";

    HttpResponseMessage response = client.GetAsync(uri).Result;
    response.EnsureSuccessStatusCode();
    string responseBody = response.Content.ReadAsStringAsync().Result;

    var infos = JsonConvert.DeserializeObject<List<DiagramInfo>>(responseBody);

    return infos;
}

public List<CategoryPriceInfo> GetCategoryInfo(int monthNumber)
{
    var uri = $"{ServerAddress}/diagram/categories/{monthNumber}";

    HttpResponseMessage response = client.GetAsync(uri).Result;
    response.EnsureSuccessStatusCode();
    string responseBody = response.Content.ReadAsStringAsync().Result;

    var infos = JsonConvert.DeserializeObject<List<CategoryPriceInfo>>(responseBody);

    return infos;
}

public List<TransactionInfo> GetCategoryTransactions(int monthNumber, string categoryName)
{
    var uri = $"{ServerAddress}/diagram/categorytransactions/{monthNumber}/{categoryName}";

    HttpResponseMessage response = client.GetAsync(uri).Result;
    response.EnsureSuccessStatusCode();
    string responseBody = response.Content.ReadAsStringAsync().Result;

    var infos = JsonConvert.DeserializeObject<List<TransactionInfo>>(responseBody);

    return infos;
}
}

```

Файл "MainForm.cs"

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace WindowsFormsApp
{
    public partial class MainForm : Form
    {
        private Dictionary<int, string> _months = new Dictionary<int, string>()
        {
            { 1, "Січень" },
            { 2, "Лютий" },
            { 3, "Березень" },
            { 4, "Квітень" },
            { 5, "Травень" },
            { 6, "Червень" },
            { 7, "Липень" },
            { 8, "Серпень" },
            { 9, "Вересень" },
            { 10, "Жовтень" },
            { 11, "Листопад" },
            { 12, "Грудень" },
        };

        private ApiWorker _apiWorker = new ApiWorker("https://localhost:44378");

        public MainForm()
        {
            InitializeComponent();
        }

        private int GetMonthNumber(string chartName)
        {
            return int.Parse(chartName.Substring("chart".Length));
        }

        private string GetMonthName(int monthNumber)
        {
            return _months[monthNumber];
        }
    }
}

```

```

private void btnLoadFile_Click(object sender, EventArgs e)
{
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        lblChosenFilePath.Text = $"Вибрано {openFileDialog.FileName}";

        btnUploadFile.Visible = lblChosenFilePath.Visible = true;
    }
}

private void btnUploadFile_Click(object sender, EventArgs e)
{
    tabControl.SelectTab(tabPageHistogramms);
}

private void Form1_Load(object sender, EventArgs e)
{
    var charts = new[]
    {
        chart5, chart6, chart4, chart1, chart2, chart3,
        chart7, chart9, chart8, chart10, chart12, chart11,
    };

    var diagramInfoList = _apiWorker.GetDiagramInfo();

    var dictionary = diagramInfoList.ToDictionary(x => x.Month, x => x.TransactionsPrice);
    var min = dictionary.Values.Min();
    var max = dictionary.Values.Max();

    foreach (var chart in charts)
    {
        chart.Series.Clear();
        chart.Legends.Clear();

        var series = new Series(chart.Name);

        int monthNumber = GetMonthNumber(chart.Name);

        if (dictionary.ContainsKey(monthNumber))
        {
            double monthPrice = dictionary[monthNumber];
            series.AxisLabel = $" {GetMonthName(monthNumber)}: {monthPrice}";
            series.Points.AddY(monthPrice);
        }
        else
        {
            chart.Visible = false;
        }
    }

    chart.ChartAreas[0].AxisY.Maximum = max;

```

```

        chart.Series.Add(series);
    }
}

private void chart_Click(object sender, EventArgs e)
{
    var actualSender = sender as Chart;

    int monthNumber = GetMonthNumber(actualSender.Name);

    var categoriesInfoList = _apiWorker.GetCategoryInfo(monthNumber);

    var extendedChart = new ExtendedChartForm(GetMonthName(monthNumber),
categoriesInfoList, monthNumber);
    extendedChart.ShowDialog();
}
}
}

```

Файл "TransactionsTable.cs"

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TransactionAnalysis.Entities;

namespace WindowsFormsApp
{
    public partial class TransactionsTable : Form
    {
        public TransactionsTable(List<TransactionInfo> transactionsList)
        {
            InitializeComponent();

            var row1 = new object[6];
            foreach (var listElement in transactionsList)
            {
                row1[0] = listElement.Day + "";
                row1[1] = listElement.Month + "";
                row1[2] = listElement.Year + "";
                row1[3] = listElement.Category;
                row1[4] = listElement.Price;
                row1[5] = listElement.Description;
            }
        }
    }
}

```

```

        dataGridView1.Rows.Add(row1);
    }
}

private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
}
}
}

```

Файл "Program.cs"

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}

```

Файл "ExtendedChartForm.cs"

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using TransactionAnalysis.Entities;

```

```

namespace WindowsFormsApp
{
    public partial class ExtendedChartForm : Form
    {
        private int _monthNumber;

        public ExtendedChartForm(string formCaption, List<CategoryPriceInfo> categoriesInfoList, int
monthNumber)
        {
            InitializeComponent();

            _monthNumber = monthNumber;

            Text = formCaption;

            extendedChart.Series.Clear();
            //extendedChart.Legends.Clear();

            var series = new Series()
            {
                Name = "text",
                ChartType = SeriesChartType.Pyramid,
                //Label = "label"
            };

            List<CategoryPriceInfo> sortedList = categoriesInfoList.OrderByDescending(x =>
x.Price).ToList();

            foreach (var info in sortedList)
            {
                series.Points.Add(new DataPoint
                {

                    //AxisLabel = info.Category,
                    Label = info.Category + " : " + info.Price,
                    LegendText = info.Category,
                    YValues = new[] { info.Price },
                });
            }

            extendedChart.Series.Add(series);
        }

        private void extendedChart_MouseDown(object sender, System.Windows.Forms.MouseEventArgs
e)
        {
            HitTestResult result = extendedChart.HitTest(e.X, e.Y);
            if (result != null && result.Object != null)
            {

```

```

// When user hits the LegendItem
if (result.Object is LegendItem)
{
    ApiWorker apiWorker = new ApiWorker("https://localhost:44378");
    // Legend item result
    LegendItem legendItem = (LegendItem)result.Object;
    List<TransactionInfo> transactionsList
apiWorker.GetCategoryTransactions(_monthNumber, legendItem.Name);

    var transactionsTable = new TransactionsTable(transactionsList);
    transactionsTable.ShowDialog();
    //MessageBox.Show(result2);
}
}
}
}
}
}

```

Файл "Startup.cs"

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using TransactionAnalysis.BusinessLogic;
using TransactionAnalysis.BusinessLogic.Abstraction;

namespace TransactionAnalysis
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

```



```

services.AddControllers();

services.AddTransient<IStatisticsService, StatisticsService>();

services.AddTransient<IDataBaseOperations, DataBaseOperations>();
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    //app.UseHttpsRedirection();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
}
}

```

Файл "Startup.cs"

```

{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft": "Warning",
            "Microsoft.Hosting.Lifetime": "Information"
        }
    },
    "AllowedHosts": "*",

    "EPPlus": {
        "ExcelPackage": {
            "LicenseContext": "NonCommercial"
        }
    }
}

```

Файл "Program.cs"

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace TransactionAnalysis
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```