НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9:658.14/.16

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету	Завідувач кафедри комп'ютерних наук
інформаційних технологій	
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____ **2022 р.**	_____ **2022 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Аналіз фінансової діяльності підприємства на основі використання сучасних інформаційних технологій»

Спеціальність 121 – «Інженерія програмного забезпечення»

Освітня програма «Програмне забезпечення інформаційних систем»

Орієнтація освітньої програми освітньо-професійна

Гарант освітньої програми

к.т.н., доцент _____ Голуб Б.Л.

Керівник магістерської кваліфікаційної роботи

к.е.н., доцент _____ Густера О.М.

Виконав _____ Снігир Р.В.

КИЇВ-2022

НУБіП України

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"01" листопада 2021 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Снігиру Роману Володимировичу
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Програмне забезпечення інформаційних систем»
Орієнтація освітньої програми: освітньо-професійна
Тема магістерської кваліфікаційної роботи «Аналіз фінансової діяльності підприємства на основі використання сучасних інформаційних технологій»
затверджена наказом ректора НУБіП України від "01" листопада 2021 р. №1861 «С»

Термін подання завершеної роботи на кафедру 27 жовтня 2022 р.
Вихідні дані до магістерської кваліфікаційної роботи:
1. дані фінансового обліку підприємства;
2. існуючі рішення на ринку.

Перелік питань, що підлягають дослідженню:
1. Аналіз предметної області
2. Дослідження інструментів OLAP
3. Проектування системи
4. Дослідження інструментів Data Mining
5. Розробка алгоритмів аналізу даних
6. Дослідження отриманих результатів

Дата видачі завдання "01" листопада 2021 р.

Керівник магістерської кваліфікаційної роботи _____ Густера О.М.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Снігир Р.В.
(підпис) (прізвище та ініціали)

ЗМІСТ

НУБіП України

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AMQP – Advanced Message Queuing Protocol

API – Application Programming Interface

DRF – Django Rest Framework

DWH – Data Warehouse

MVC – Model View Controller

OLAP – online analytical processing (аналітична обробка у реальному часі).

SQL – Structured query language (мова структурних запитів).

STOMP – Streaming Text Oriented Messaging Protocol

UML – Unified Modeling Language

XMPP – Extensible Messaging and Presence Protocol

БД – база даних.

ПЗ – програмне забезпечення

СД – сховище даних.

СКБД – система керування базами даних.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

ВСТУП

Кожного дня в світі відбувається безліч грошових транзакцій при яких купляються чи продаються різноманітні послуги чи товари. Досить складно відслідкувати рух всіх грошей в світі, та кожне підприємство повинно вміти правильно розпоряджатися своїми коштами.

Раніше люди вручну вели всі записи, що займало дуже багато часу, а особливо коли це стосувалося підприємств де працювало велика кількість людей. Але в сучасному світі все більше починають використовуватись програмні засоби, які допомагають автоматизувати всі ці процеси.

Система обліку і аналізу фінансової діяльності підприємства допомагає автоматизувати ведення обліку на підприємствах будь-якого типу. Вона допомагає швидко та зручно вносити витрати та прибутки а також отримувати аналітичні звіти.

Тож актуальність обраної теми полягає в тому, що у сучасному світі є безліч підприємств, що досі проводять облік та аналіз своїх фінансових даних вручну. Це вимагає дуже багато часу та людських ресурсів. Виходом з такої ситуації є використання системи обліку та аналізу фінансових діяльності підприємства, що допоможе автоматизувати більшість процесів та зменшити використання часу на ці процеси.

Об’єкт дослідження: Дані обліку та аналізу фінансової діяльності підприємства.

Предмет дослідження: Процеси автоматизації обліку та аналізу фінансової діяльності підприємства.

Мета: Оцінити доцільність використання технологій Data mining та Machine learning для аналізу фінансової діяльності підприємства.

Завданням дослідження є створення системи обліку і аналізу фінансової діяльності підприємства, за допомогою якої можна буде проводити аналіз фінансового стану підприємства.

У процесі дослідження розроблене програмне забезпечення що допомагає вести облік фінансової діяльності підприємства, а також використано методи,

що дозволяють автоматично створювати аналітичну звітність користуючись різними алгоритмами машинного навчання, а також статистичними формулами.

Апробація результатів дослідження

Снігир Р.В., Густера О.М.: Аналіз фінансової діяльності підприємства на основі використання сучасних інформаційних технологій. Збірник матеріалів XIII Міжнародної науково-практичній конференції молодих вчених "Інформаційні технології: економіка, техніка, освіта". 26-27 жовтня 2022 року, НУБіП України, Київ. – С. ??. Режим доступу: ???????.

Магістерська робота має наступну структуру:

- перший розділ в якому описано предметну область;
- другий розділ в якому змодельована система;
- третій розділ в якому обрані технології розробки та реалізована система;
- четвертий розділ містити результати розробки та дослідження.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про фінансовий облік підприємства

Ефективне функціонування економіки можливе за умови фінансового здоров'я суб'єктів господарювання. Для цього керівництво підприємства повинно періодично оцінювати його фінансовий стан. Регулярний контроль фінансового стану забезпечить менеджерів цінною інформацією про події, що відбулися, відбуваються та відбуватимуться. Це дозволить виокремити в діяльності поточні та потенційні проблеми, а також вжити комплекс заходів для усунення проблемних моментів і навіть для їх попередження. Результати такого аналізу можуть бути використані рядом зовнішніх користувачів: постачальниками, клієнтами, конкурентами, державними бухгалтерами, банківськими установами, податковими органами, страховими компаніями, інвесторами (в т.ч. потенційними), органами статистики тощо [21].

Таким чином, фінансова звітність - це одна з найважливіших характеристик діяльності будь-якого суб'єкта господарювання[28]. Вона дозволяє визначити конкурентоспроможність підприємства, його можливості в діловому співробітництві, його поточну та потенційну стабільність і надійність, заздалегідь правильно вибрати стратегію розвитку. Але оцінка фінансового стану являє собою процес аналізу майна та ресурсів господарюючого суб'єкта за допомогою методики експрес-аналізу та поглибленого аналізу, що дозволяє виявити проблемні моменти в діяльності підприємства, встановити закономірності та причини їх виникнення, а також намітити шляхи усунення

Необхідність проведення аналізу фінансової звітності з'ясовується в ході аналізу наступних питань:

- як досягти поставленої мети без втрат у поточній діяльності?
- як управляти майном підприємства з метою підвищення ефективності діяльності в поточному періоді?
- які пріоритети джерел формування активів?
- наскільки стійким є розвиток підприємства?

НУБіП України

- наскільки високий рейтинг підприємства в ділових колах?
- чи є воно привабливим для інвесторів?

Основними завданнями аналізу фінансової звітності є:

- загальна оцінка пасивів і активів підприємства в цілому та за їх видами
- виявлення причин змін, що відбулися, оцінка наслідків змін;
- вивчення складу і структури власного і позикового капіталу;
- оцінка стану, структури і тенденцій зміни основного і оборотного капіталу;
- забезпечення відповідності розміру власного та позикового капіталу обсягу основного та оборотного капіталу плановим потребам підприємства;
- забезпечення оптимальної структури капіталу з позицій його ефективного функціонування;
- оцінка ефективності використання активів підприємства;
- оптимізація структури капіталу за його видами;
- пошук шляхів підвищення ефективності формування та використання капіталу підприємства, а також стратегії фінансової безпеки підприємства;
- об'єктивна оцінка динаміки та стану ліквідності підприємства, платоспроможності та фінансової стійкості;
- оцінка конкурентоспроможності підприємства.

Таким чином, в сучасних умовах господарювання, оцінка фінансового стану підприємства є одним з найважливіших факторів стабільного та ефективного функціонування будь-якої організації.

1.2 Огляд інформаційних джерел та існуючих рішень

Точність і ефективність надання економічної інформації є вирішальним чинником успішного економічного розвитку підприємств і цілих країн.

Автоматизація бухгалтерського обліку значно розширює його можливості і має вирішальне значення для вирішення цього завдання. З огляду на велику різноманітність бухгалтерських програм, представлених сьогодні на ринку, користувачі комп'ютерних систем повинні зробити правильний вибір на основі конкретних потреб, вимог і функціональності програмного забезпечення. При цьому необхідно враховувати відмінності вітчизняної бухгалтерської побудови від розвинених економік. Якщо в Україні донедавна облік обмежувався обліково-реєстраційними завданнями, то в розвинутих країнах ці питання не є основною частиною бухгалтерського обліку. Основна увага приділяється фінансовому аналізу та швидкій обробці інформації. Тому дуже важливо правильно оцінити завдання та мету автоматизації бухгалтерського обліку.

Завданням автоматизації є підвищення якості роботи бухгалтерів і бухгалтерської роботи в цілому. Комп'ютер – це той інструмент, який дозволяє максимально у повному обсязі використати кваліфікацію спеціаліста та спростити повсякденну роботу бухгалтера.

Для ефективного створення свого програмного забезпечення яке повинно виконувати якусь діяльність, потрібно проаналізувати аналоги, якщо вони є. Таких аналогів може не бути, але в такому випадку можна взяти максимально схожі системи, проаналізувати їх, та відібрати той функціонал який буде корисний для створюваної системи або програмного забезпечення.

Розглянемо програмні застосунки які використовують при бухгалтерському обліку та аналізі фінансової діяльності підприємства.

«Парус-Бухгалтерія» – повнофункціональний програмний продукт, який дозволяє автоматизувати бухгалтерський та податковий облік в організаціях і підприємствах малого та середнього бізнеса[29].

Основні функції які має це програмне забезпечення:

- облік банку та каси;
- облік основних засобів;
- облік господарських операцій;
- облік матеріальних цінностей;

- облік посвідчень про відрядження та авансових звітів;
- формування оборотних відомостей різних типів:
  - руху коштів за рахунками;
  - рух коштів за аналітичними рахунками;
  - рух коштів у розрізі проводок;
  - рух матеріальних цінностей;
  - розрахунки з дебіторами-кредиторами;
- облік вхідних і вихідних податкових накладних;
- реєстри отриманих, виданих податкових накладних;
- облік податкового кредиту та податкових зобов'язань, валових доходів і валових витрат;
- усі звіти, регламентовані поточним законодавством та аналітична звітність;
- подання звітності у форматі xml.

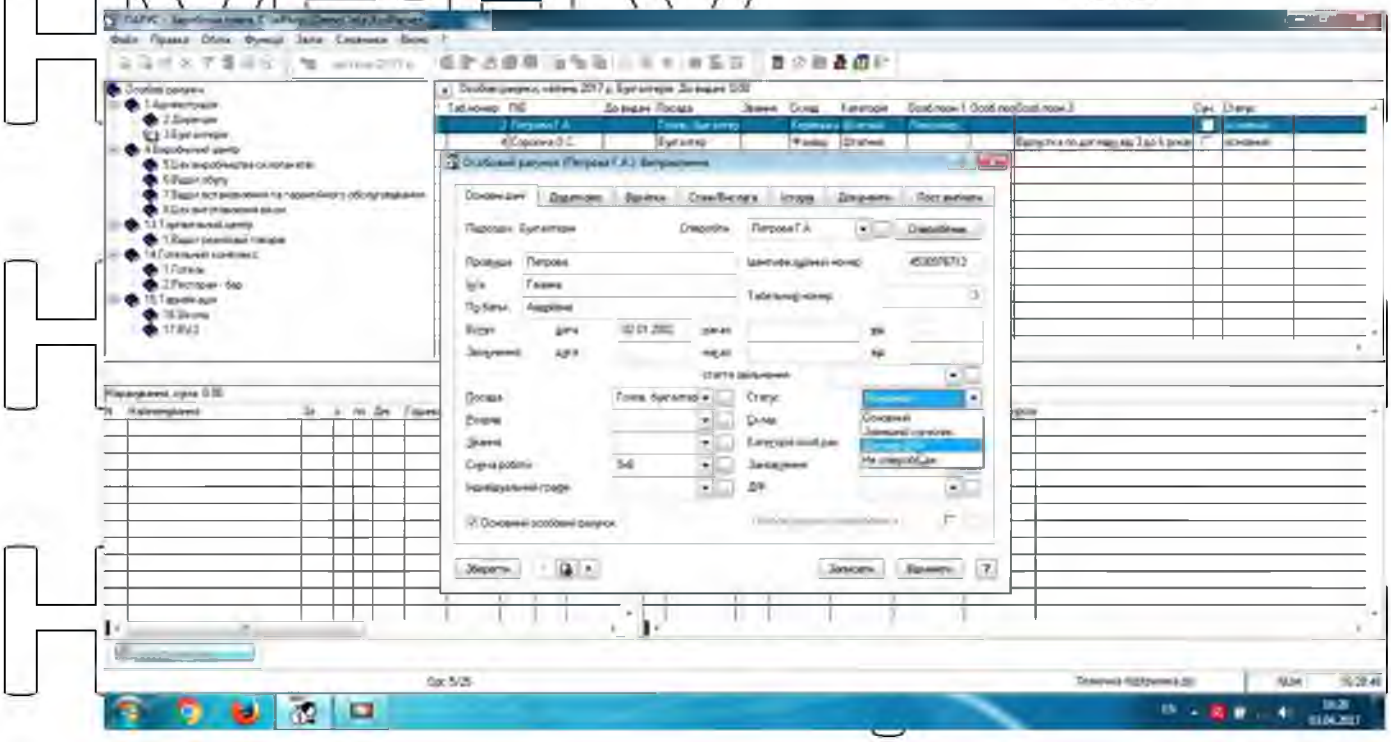


Рис. 1 Інтерфейс програмного забезпечення «Парус»

На рисунку 1 показаний інтерфейс програмного забезпечення «Парус».

Також є додаткова можливість з інтеграції програмного продукту «Парус-Бухгалтерія» в зв'язці з інформаційно-аналітичною системою по законодавству України «Парус-Консультант». У головному меню будь-якого розділу «Парус-Бухгалтерія» є можливість завантажити список нормативних і законодавчих актів, консультацій та роз'яснень спеціалістів, які регламентують дії користувача стосовно того облікового регістру, з якого було викликано допомогу.

«БюджетСофт» - це українське програмне забезпечення, яке дозволяє ефективно керувати підприємством, підвищувати рівень управління та спрощувати роботу співробітників. Це програмне забезпечення можна налаштовувати під бізнес-процеси компанії, що дозволяють контролювати великі масиви даних та інформаційні потоки.

Плюси цього програмного забезпечення:

- має модульну будову, завдяки якій кожне з рішень може працювати як у взаємодії з іншими рішеннями системи, так і автономно. Тому існує можливість індивідуального підбору необхідної функціональності;
- є можливість додавати процедури користувача та завдання користувача.
- є функція формування звітності, в тому числі з можливістю побудови звітів довільної форми на підставі будь-якої інформації;
- є функція контролю над діями користувача, що використовується для реєстрації змін даних, бізнес-процесів, початку/завершення сеансів роботи користувача.

«1С Підприємство» - універсальна бухгалтерська програма[30]. Вона надає можливість ведення обліку всіх складових бухгалтерії підприємства, незважаючи на галузь до якої відноситься підприємство.

Головним мінусом цієї програми є те що вона є дуже складною в опануванні користувачем, а також має проблеми зі звітністю при закритті звітного періоду.

Функціональні можливості «1С Підприємство»:

НУБіП України

- ведення обліку діяльності кількох організацій;
- управління продажами;
- планування продажів;
- управління замовленнями покупців і внутрішніми замовленнями;
- роздрібна торгівля;
- ціноутворення;
- управління закупівлями;
- управління відносинами з покупцями та постачальниками;
- складський облік;
- управління грошовими потоками підприємства;
- управління взаєморозрахунками;
- облік виробництва;
- управління персоналом і розрахунок зарплати;
- кадровий облік та аналіз кадрового складу;
- розрахунок заробітної плати;
- регламентована звітність, обчислення регламентованих податків;
- бухгалтерський і податковий облік;
- аналіз результатів діяльності підприємства;
- рапорт керівнику;
- монітор ефективності;
- структура оборотних коштів.

На рисунках 2-3 показаний інтерфейс ПЗ «1С Підприємство»

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

Рис. 2 Моніторинг ефективності «1С Підприємство»

Структура оборотных средств + финансовый анализ

Сформировать | Заголовок | Настройка ...

Период с: 04.05.2015 по: 24.05.2015 Периодичность: Неделя

Структура оборотных средств + финансовый анализ

04.05.2015 - 24.05.2015

Измерение	На начало периода	Неделя с 4 мая 2015	Неделя с 11 мая 2015	Неделя с 18 мая 2015
Показатели				
Денежные средства	1 007 271,78	996 943,51	656 240,89	648 772,42
Дебиторская задолженность	11 802,70	22 255,11	9 239,24	9 532,18
Дебиторская задолженность контрагентов	11 802,70	22 248,78	9 239,24	9 532,18
Дебиторская задолженность подотчетных лиц		6,33		
Кредиторская задолженность	90 139,79	90 502,71	91 170,45	94 979,62
Кредиторская задолженность контрагентов	90 139,79	90 502,71	91 170,45	94 979,62
Кредиторская задолженность подотчетных лиц				
Себестоимость товаров (за исключением принятых на комиссию)	56 489,30	49 946,01	58 934,30	58 371,00
Коэффициент абсолютной ликвидности	11,17455	11,01562	7,19796	6,83065
Коэффициент срочной ликвидности	11,30549	11,26153	7,29930	6,93101
Коэффициент текущей ликвидности	11,93218	11,81340	7,94572	7,54557
Чистые оборотные активы	985 423,99	978 641,92	633 243,98	621 695,98
Обеспеченность собственным оборотным капиталом	0,91619	0,91535	0,87415	0,86747
Коэффициент маневренности собственного оборотного капитала	0,05732	0,05104	0,09307	0,09389
Доля труднореализуемых активов в общей величине оборотных активов	0,05252	0,04672	0,08135	0,08145
Соотношение труднореализуемых и легкореализуемых активов	0,05543	0,04901	0,08856	0,08867

Рис. 3 Структура обігових коштів

Проаналізувавши наведені аналоги можна дійти висновку що в них є декілька основних мінусів:

- висока ціна за користування;
- висока складність використання без належного навчання;
- наявність підходящих технічних характеристик комп'ютера а іноді навіть окремого сервера або бази даних.

Тому при розробці програмного забезпечення яке буде вирішувати поставлені задачі потрібно враховувати, щоб це програмне забезпечення було доступне користувачу з будь-якого пристрою, який має доступ до інтернету та браузеру, просте у розумінні та використанні, щоб інтерфейс ПЗ був «дружнім» для користувача, а також по можливості був недорогим по ціні.

1.3 Постановка завдань дослідження

Порівнявши аналоги системи, а також проаналізувавши предметну область дослідження можна дійти висновку що розроблювана система обліку та аналізу фінансової діяльності підприємства повинна мати 2 модулі: ПЗ що буде вести облік фінансів, а також аналітичний модуль, який буде аналізувати внесені дані та формувати аналітичну звітність, якою потім зможуть користуватися аналітики або ж менеджери підприємства.

В цілому система має виконувати наступні функції:

- додавання нових підприємства;
- додавання нових користувачів;
- додавання нових категорій;
- робота з транзакціями (додавання, зміна, видалення, перегляд);
- імпорт та експорт записів в систему та з неї;
- відслідковування дій користувачі в системі;
- показ аналітичних графіків;
- формування аналітичної звітності відносно категорій, типу, дати та інших атрибутів переказів використовуючи наступні алгоритми:
  - лінійна регресія;
  - дерево рішень;
  - кластеризація;
  - статистичні функції;
  - вертикальний аналіз.

НУБіП України

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Основна інформація про об'єктно-орієнтоване моделювання

Мета об'єктно-орієнтованого проектування та моделювання – навчитися застосовувати об'єктно-орієнтовані концепції на всіх етапах життєвого циклу розробки програмного забезпечення. Об'єктно-орієнтоване моделювання та проектування – це спосіб який використовує моделі організовані навколо концепцій реального світу для осмислення проблем. Об'єкт є фундаментальною конструкцією, яка поєднує як структуру даних, так і поведінку.

Призначення моделей:

- тестування фізичного об'єкта перед створенням;
- спілкування з клієнтами;
- візуалізація;
- зниження складності.

В об'єктно-орієнтованому моделюванні та проектуванні існує 3 типи моделей: Модель класів, Модель станів та Модель взаємодії. Вони пояснюються в такий спосіб.

Модель класів - показує всі класи, що у системі. Модель класів показує атрибути та поведінку, пов'язані з об'єктами. Діаграма класів використовується для відображення моделі класу. Діаграма класу показує ім'я класу, атрибути, потім функції або методи, пов'язані з об'єктом класу. Мета побудови моделі класу – захопити ті концепції з реального світу, які важливі для застосування.

Модель станів - описує ті аспекти об'єктів, які пов'язані з часом та послідовністю операцій - події, що відзначають зміни, стани, що визначають контекст для подій, та організацію подій та станів. Дії та події на діаграмі станів стають операціями над об'єктами у моделі класів. Діаграма станів визначає модель станів.

НУБіП України

Модель взаємодії - використовується для того, щоб показати різні взаємодії між об'єктами, та як об'єкти взаємодіють задля досягнення поведінки системи загалом.

Для відображення моделі взаємодії використовуються такі діаграми:

- діаграма варіантів використання;
- діаграма послідовності;
- діаграма активності.

2.1 Діаграма прецедентів

Діаграма прецедентів або ж діаграма варіантів використання - це спосіб узагальнення інформації про систему та користувачів у цій системі. Зазвичай її використовують для графічного представлення взаємодії між різними елементами системи. Діаграми прецедентів визначають події у системі та те, як ці події протікають, проте ця діаграма не описує, як ці події реалізуються[25].

Варіант використання - це методологія, що використовується у системному аналізі для визначення, уточнення та організації вимог до системи. У цьому контексті термін "система" стосується того, що розробляється або експлуатується, наприклад, веб-сайт з продажу та обслуговування товарів поштою. Діаграми варіантів використання використовуються в UML стандартної нотації для моделювання об'єктів та систем реального світу. Діаграма варіантів використання має низку переваг перед аналогічними діаграмами, такими як блок-схеми.

Причини, з яких організація може захотіти використовувати діаграми варіантів використання, включають:

- подання цілей систем та користувачів;
- визначити контекст, у якому має розглядатися система;
- визначити вимоги до системи;
- забезпечити модель потоку подій під час взаємодії з користувачем;
- забезпечити зовнішній погляд на систему;

- показують зовнішній та внутрішній вплив на систему;
- як працюють діаграми варіантів використання.

Цілі системи можуть включати планування загальних вимог, перевірку дизайну апаратного забезпечення, тестування та налагодження програмного продукту, що розробляється, створення онлайн-каталогу довідки або виконання завдань, орієнтованих на обслуговування клієнтів. Наприклад, випадки використання в мерчандайзингу включають замовлення товарів, оновлення каталогів, обробку платежів і роботу з клієнтами. Діаграма варіантів використання складається з чотирьох компонентів.

- кордон, що визначає систему, що цікавить, стосовно навколишнього світу;
- дійові особи, зазвичай залучені до роботи системи, зумовлені відповідно до їх ролі;
- варіанти використання, які є конкретними ролями, що виконуються учасниками всередині та навколо системи;
- відносини між дійовими особами та варіантами використання.

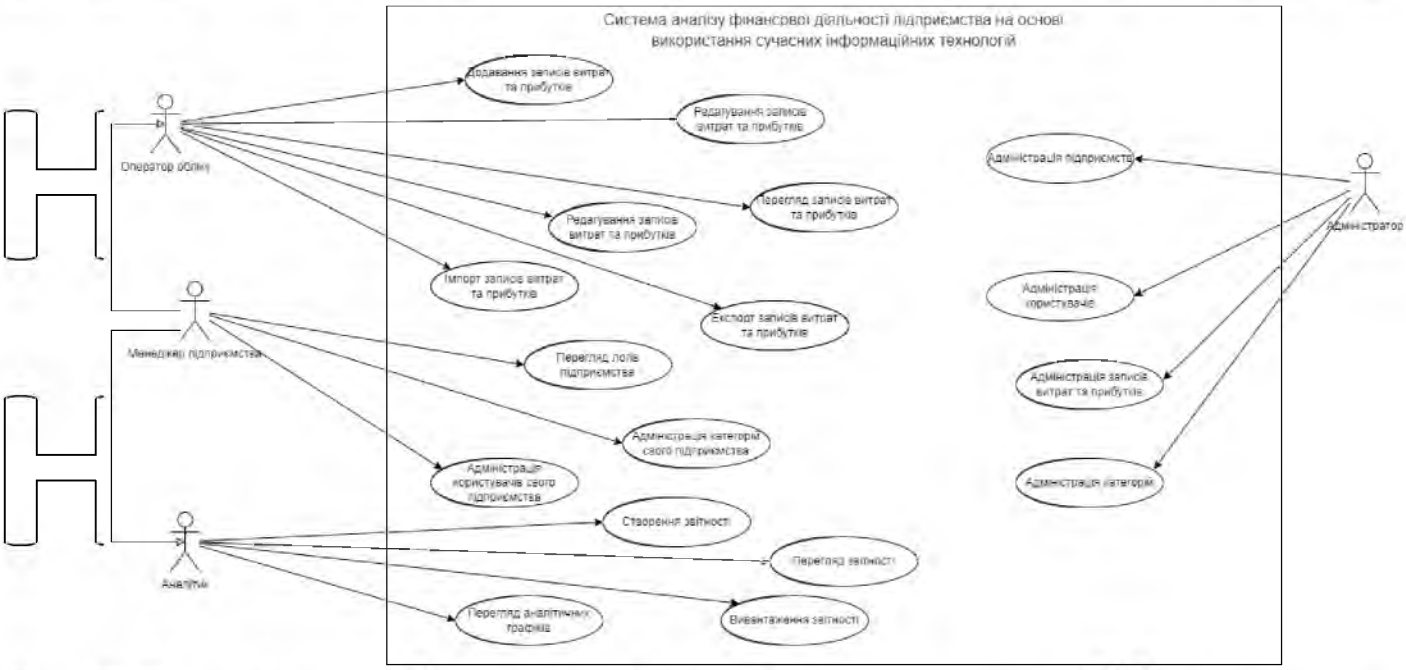


Рис. 4 Діаграма прецедентів

Основні актори діаграми прецедентів показаної на рисунку 4:

- оператор обліку – працівник підприємства, який займається обліком витрати та прибутків підприємства, а також має можливість імпортувати дані в систему та вивантажувати їх з системи;
- аналітик – працівник підприємства, що займається формуванням аналітичної звітності та аналізом фінансових прибутків і витрат, а також має можливість переглядати основні статистичні показники підприємства;
- менеджер підприємства – керівник підприємства, який займається прийняттям рішень щодо управління підприємством та переглядом аналітичної звітності;
- адміністратор – користувач, який займається додаванням нових компаній та призначенням менеджерів до цих компаній.

2.2 Діаграма класів

UML 2 розглядає структурні діаграми як категорію; самі діаграми не називаються "структурними діаграмами". Однак діаграма класів є яскравим прикладом структурної діаграми, яка надає початковий набір символічних елементів, що використовуються у всіх інших структурних діаграмах[23].

Призначення діаграми класів - показати типи, що моделюються в системі. У більшості UML-моделей до таких типів відносяться:

- клас;
- інтерфейс;
- тип даних;
- компонент.

Для цих типів в UML використовується спеціальна назва: "класифікатори". Взагалі, можна думати про класифікатор як про клас, але технічно класифікатор - це більш загальний термін, який відноситься і до інших трьох типів, наведених вище.

Клас - це план об'єкту. Об'єкти та класи йдуть рука об руку. І вся суть об'єктно-орієнтованого проектування полягає не в об'єктах, а в класах, тому що

класи використовуються для створення об'єктів. Отже, клас описує, яким буде об'єкт, але він не є самим об'єктом.

Насправді, класи описують тип об'єктів, в той час як об'єкти - це придатні для використання екземпляри класів. Кожен об'єкт був побудований з одного і того ж набору креслень і тому містить однакові компоненти (властивості і методи). Стандартне значення полягає в тому, що об'єкт - це екземпляр класу, а об'єкт - має стани і поведінку.

Основні класи діаграми класів розробленої системи представленої на рисунку 5:

- user – базовий клас користувача, котрий містить в собі атрибути Id, username, password, які допомагають ідентифікувати користувача та метод Login_to_account(), який дає можливість увійти в систему;
- accountant – клас користувача що наслідує атрибути базового користувача і має додаткові методи Create_record(), Change_record() та Delete_record() які дають користувачу можливість додавати, редагувати та видаляти записи з системи;
- analyst - клас користувача що наслідує атрибути базового користувача і має додатковий метою Create_report() що дозволяє користувачу створювати аналітичні звіти;
- manager – клас користувача який наслідує попередні два, та має їх всі атрибути та методи. Додатково цей клас може можливість створювати нових користувачів та категорії використовуючи методи Create_caregory() та Create_category();
- company – клас компанії, яка містить атрибути Id та Name де міститься назва компанії, а також методи List_companies() та Add_new_user(), що дають можливість переглядати список компаній та додавати користувачів до конкретної компанії;
- accounting report – клас аналітичного звіту де є такі атрибути, як: Id, date_start, date_end, status, message, file_path, created_at, updated_at. В цих атрибутах зберігається інформація про проміжок часу даних в звіті, статус

створення звіту, повідомлення про помилку якщо така виникла, де зберігається файл на файловій системі серверу, коли цей звіт був створений, та дата його зміни;

- category – клас категорії, що містить атрибути імені та номеру категорії Id та category_name. Також цей клас має методи для переліку категорій та додавання нових категорій List_categories() та Add_new_category() відповідно;
- accounting record – клас запису, який має такі атрибути, як: Id, amount_of_money, accounting_type, spent_date. Ці атрибути містять в собі інформацію про кількість грошей, тип транзакції та дату транзакції. Цей клас має схожі методи як і попередні що дають можливість переглядати записи та додавати нові: List_record(), Add_new_record().

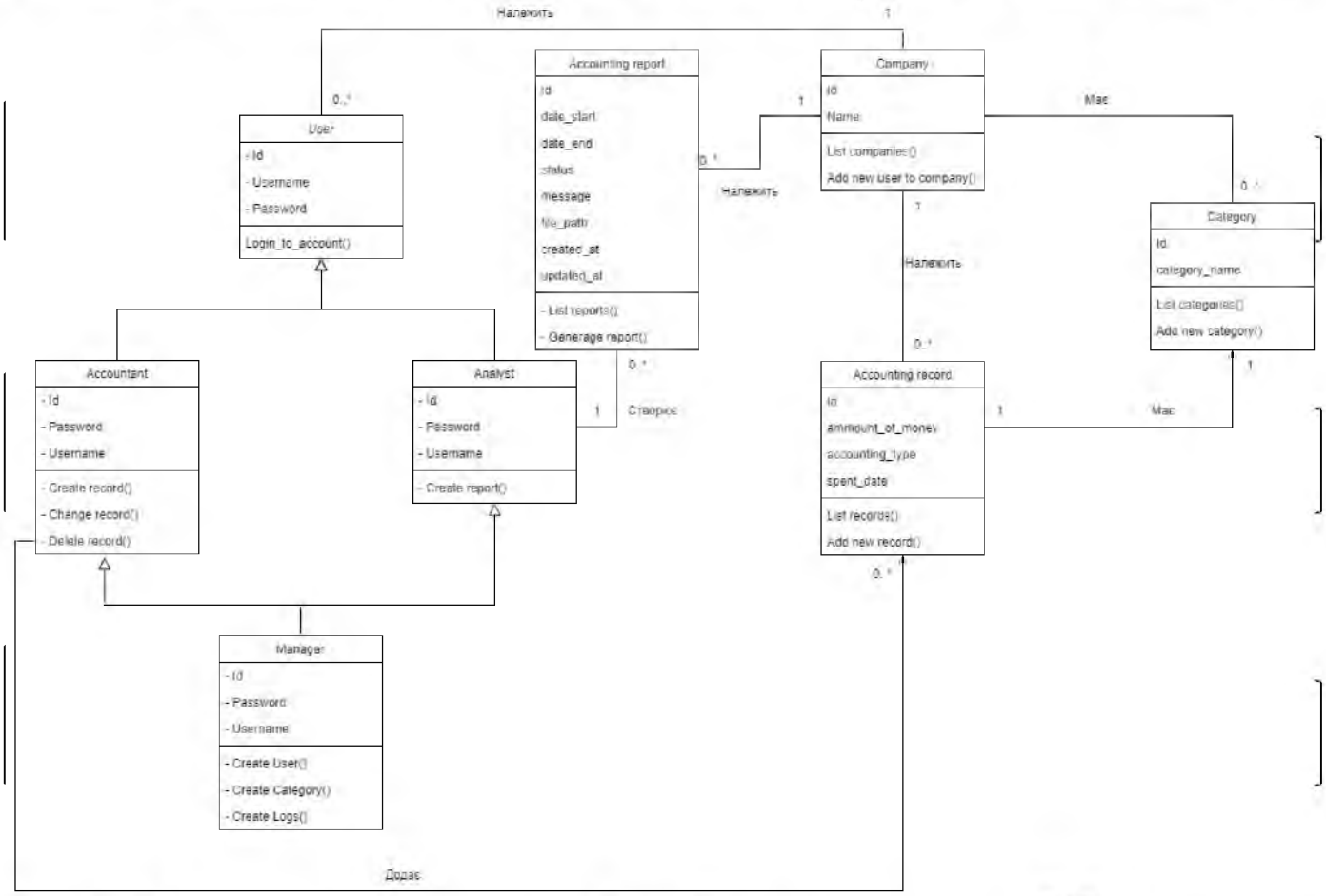


Рис. 5 Діаграма класів

Тож в цілому діаграма класів містить 8 об’єктів. Класи користувачів наслідують атрибути та функціонал один одного для розширення своїх можливостей.

2.3 Діаграма послідовності

Діаграма послідовності [7] використовується в першу чергу для відображення взаємодії між об'єктами в послідовному порядку, в якому ці взаємодії відбуваються. Подібно до діаграми класів, розробники зазвичай думають, що діаграми послідовності призначені виключно для них. Однак бізнес-персонал організації може знайти діаграми послідовності корисними для комунікації того, як бізнес працює в даний час, показуючи, як взаємодіють різні бізнес-об'єкти. Окрім документування поточних справ організації, діаграма послідовності на бізнес-рівні може бути використана як документ з вимогами для передачі вимог до майбутньої реалізації системи. Під час фази вимог проекту аналітики можуть підняти варіанти використання на наступний рівень, забезпечивши більш формальний рівень уточнення. Коли це відбувається, варіанти використання часто уточнюються в одну або декілька діаграм послідовності.

Технічний персонал організації може знайти діаграми послідовності корисними для документування того, як повинна поводити себе майбутня система. На етапі проектування архітектори та розробники можуть використовувати діаграму для визначення взаємодії об'єктів системи, таким чином конкретизуючи загальний дизайн системи.

Одне з основних застосувань діаграм послідовності полягає в переході від вимог, виражених у вигляді варіантів використання, до наступного і більш формального рівня уточнення. Варіанти використання часто уточнюються в одну або декілька діаграм послідовності. На додаток до їх використання при проектуванні нових систем, діаграми послідовності можуть бути використані для документування того, як об'єкти в існуючій системі в даний час

взаємодіють. Ця документація дуже корисна при передачі системи іншій особі або організації.

Основною метою діаграма послідовності є визначення послідовності подій, які призводять до певного бажаного результату. Основна увага приділяється не стільки самим повідомленням, скільки порядку, в якому вони відбуваються, тим не менш, більшість діаграм послідовності показують, які повідомлення надсилаються між об'єктами системи, а також порядок, в якому вони відбуваються. Діаграма передає цю інформацію вздовж горизонтального та вертикального вимірів: вертикальний вимір показує, зверху вниз, часову послідовність повідомлень/викликів у міру їх виникнення, а горизонтальний вимір показує, зліва направо, екземпляри об'єктів, до яких надсилаються повідомлення.

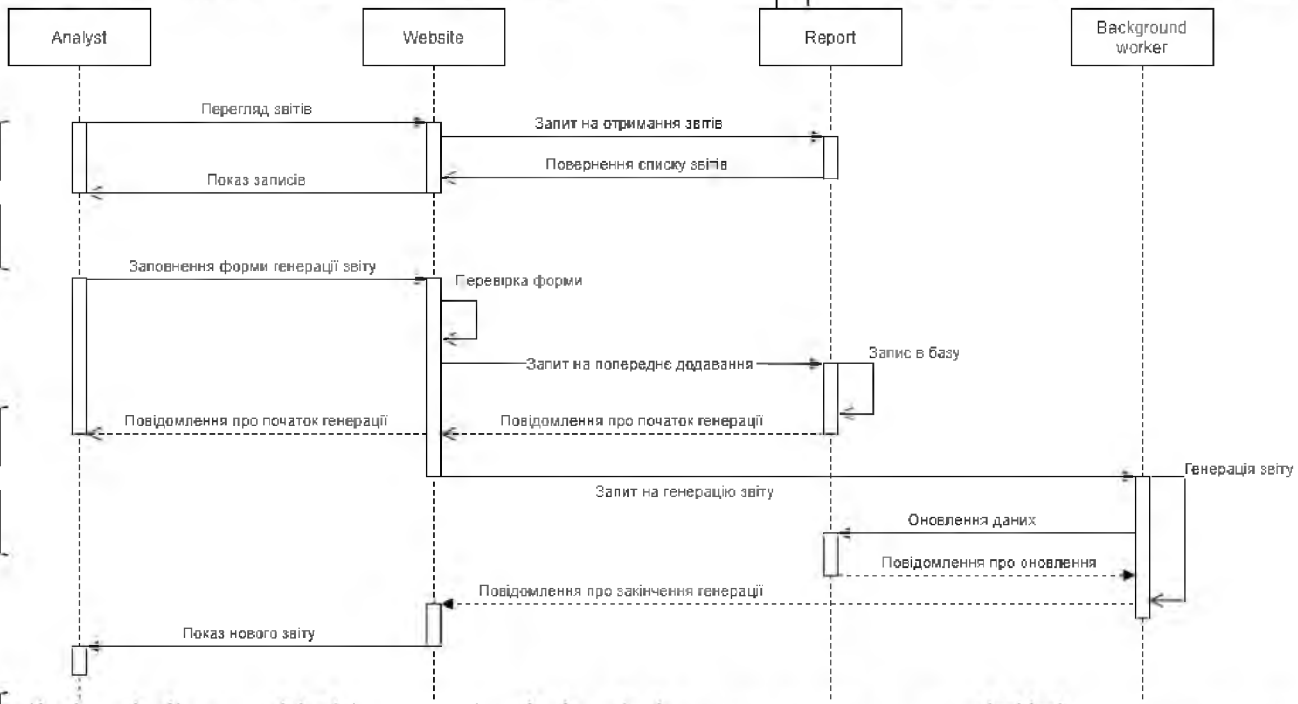


Рис 6 Діаграма послідовності генерації звіту

Об’єкти між якими йде обмін повідомленнями в діаграмі послідовності для генерації звіту показаної на рисунку 6:

- analyst – користувач, який має доступ до створення звіту та який заповнюючи форму починає процедуру його генерації;
- website – веб-застосунок, який є проміжною ланкою передачі

повідомлень та перевіряє данні що передаються;

- report – об’єкт в базі даних над яким відбуваються зміни;
- background worker – працівник веб-застосунку, який працює паралельно з основною програмою, та відповідає за генерацію самого звіту використовуючи дані передані від веб-застосунку.

Рис. 7 Діаграма послідовності додавання запису

Об’єкти між якими йде обмін повідомленнями в діаграмі послідовності для додавання запису показаної на рисунку 7:

- accountant – користувач, який має доступ до додавання запису та який заповнюючи форму відправляє її для внесення запису в базу даних;
- website – веб-застосунок, який є проміжною ланкою передачі повідомлень та перевіряє данні що передаються;
- record – об’єкт в базі даних над яким відбуваються зміни.

2.4 Діаграма розгортання

Діаграма розгортання показує, як система буде фізично розгорнута в апаратному середовиші. Її мета полягає в тому, щоб показати, де будуть фізично працювати різні компоненти системи і як вони будуть взаємодіяти

один з одним. Оскільки діаграма моделює фізичний час виконання, виробничий персонал системи буде широко використовувати цю діаграму.

Нотація на діаграмі розгортання включає елементи нотації, що використовуються на діаграмі компонентів, з деякими доповненнями, включаючи поняття вузла. Вузол представляє або фізичну машину, або вузол віртуальної машини (наприклад, вузол мейнфрейма).

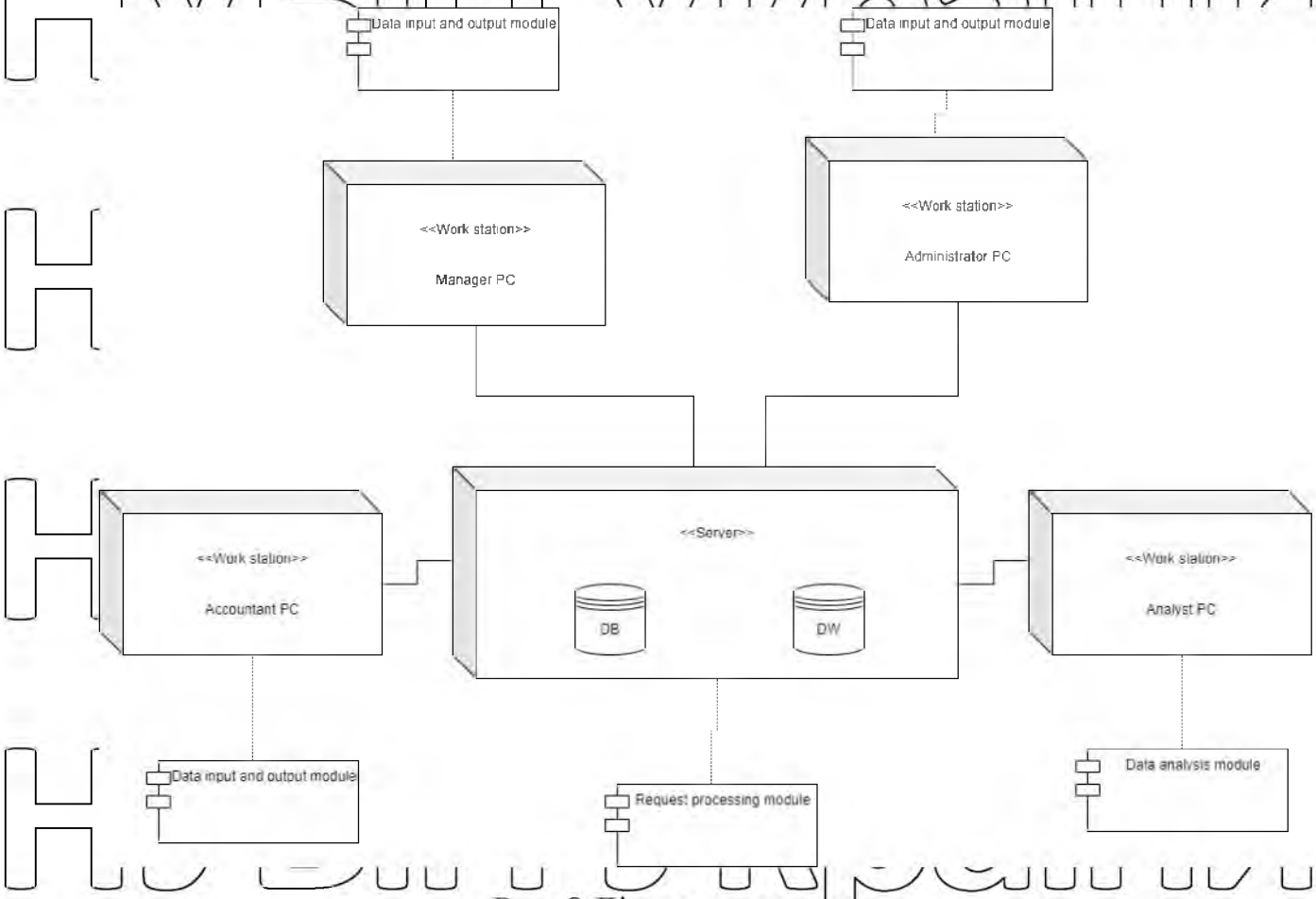


Рис. 8 Діаграма розгортання

Діаграма розгортання, що показана на рисунку 8 є досить простою. Вона містить всього 5 вузлів, 4 з яких займають робочі станції:

- manager pc – станція що має доступ до серверу використовуючи менеджерський доступ;
- administrator pc - станція що має доступ до серверу використовуючи адміністраторський доступ;
- accountant pc - станція що має доступ до серверу використовуючи

доступ оператора обліку;

- analyst pc - станція що має доступ до серверу використовуючи доступ аналітика;
- server – основний вузол діаграми в якому зберігаються дані системи та відбуваються обчислення. В цьому вузлі зберігається база даних та сховище даних.

2.5 Діаграма діяльності

Діаграми діяльності використовуються, щоб проілюструвати потік управління в системі і посилатися на кроки, пов'язані з виконанням варіанту використання. За допомогою діаграм діяльності моделюються послідовні та паралельні дії. Таким чином, в основному зображуються робочі процеси візуально за допомогою діаграми діяльності. Діаграма діяльності фокусується на стані потоку і послідовності, в якій він відбувається. Описується або зображується те, що викликає певну подію, використовуючи діаграму діяльності. UML моделює в основному три типи діаграм, а саме: діаграми структури, діаграми взаємодії та діаграми поведінки. Діаграма діяльності - це поведінкова діаграма, тобто вона відображає поведінку системи. Діаграма діяльності зображує потік управління від початкової до кінцевої точки, показуючи різні шляхи прийняття рішень, які існують під час виконання діяльності. За допомогою діаграми діяльності можна зобразити як послідовну обробку, так і паралельну обробку діяльності. Вони використовуються в бізнес-моделюванні та моделюванні процесів, де їх основне застосування полягає в зображенні динамічних аспектів системи. Діаграма робіт дуже схожа на блок-схему, але має свої відмінності.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

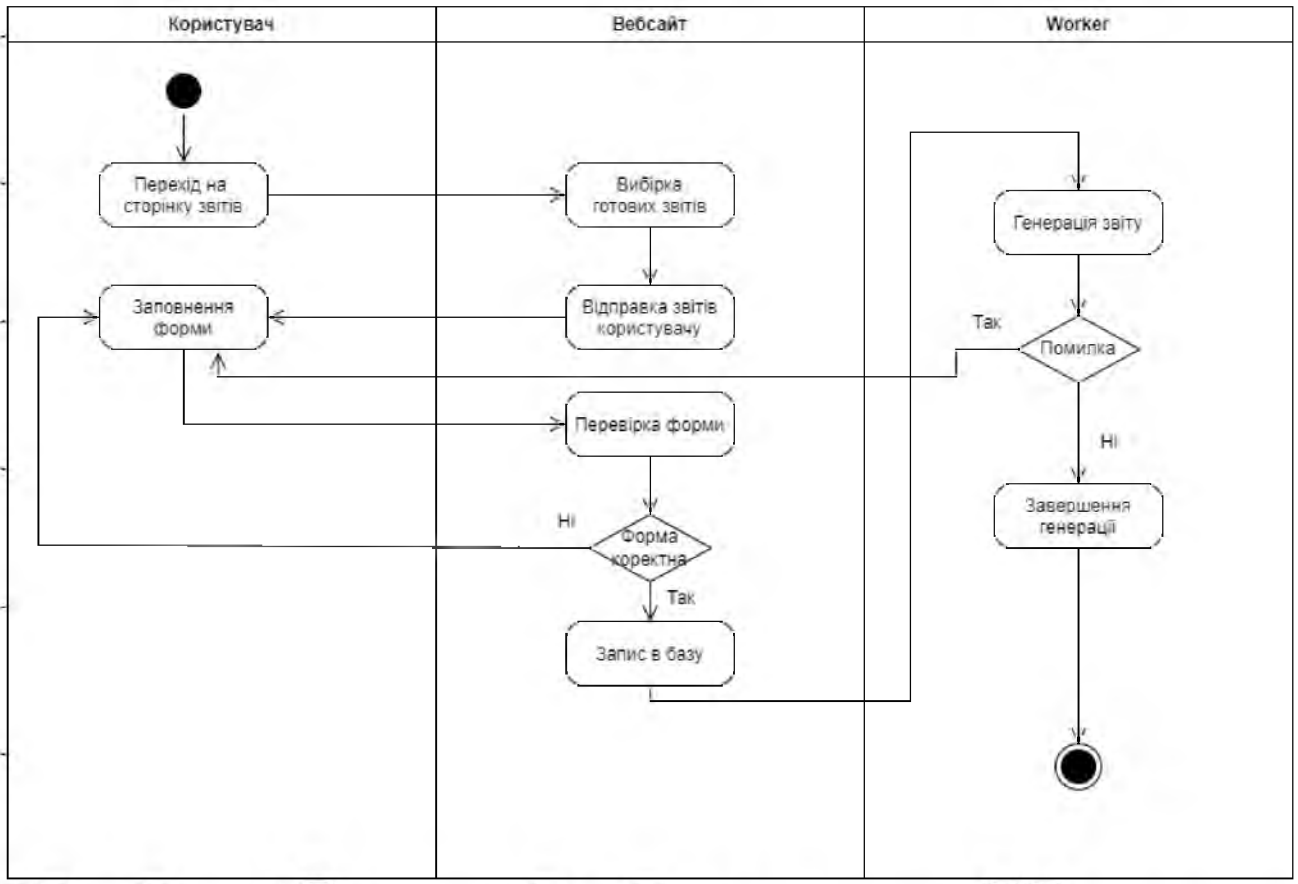


Рис. 9 Діаграма діяльності функції генерації звіту

В діаграми діяльності функції генераці звіту розроблюваної системи показаної на рисунку 9 демонструється послідовність виконання дій при генерації звіту та потік інформації між частинами. Початком цього походу є перехід до сторінки звітів користувачем. Після цього веб-застосунок фільтрує звіти які є в нього в базі даних та надсилає їх користувачу. Після цього користувач заповнює форму та відправляє ці дані до веб-застосунку. Наступним кроком веб-сервіс перевіряє надіслані дані та на основі перевірки вирішує як далі продовжувати роботу. Якщо перевірка успішна, дані направляються до фонового робітника, який використовуючи ці дані починає генерацію звіту. Якщо ж форма заповнена некоректно, ці дані повертаються до користувача з помилкою. Після завершення роботи фонового робітника йде перевірка на помилку створення звіту, після чого ці дані записуються до бази даних та результати надсилаються користувачу.

НУБіП України
НУБіП України
НУБіП України
НУБіП України

2.6 Діаграма кооперації

Діаграми кооперації UML [3], раніше відомі як діаграми колаборації, є типом поведінкових діаграм, які показують взаємодію, що відбувається між об'єктами в програмному забезпеченні або системі. Цей тип діаграм підкреслює повідомлення, якими обмінюються об'єкти. Діаграми кооперації найкраще використовувати коли один варіант використання має кілька сценаріїв, які потрібно зобразити.

Основні задачі діаграми кооперації:

- показати проходження повідомлень між об'єктами в системі або програмному забезпеченні;
- зображення взаємодії між об'єктами;
- візуалізація того, як відправляються і приймаються повідомлення між об'єктами, а також наслідки.

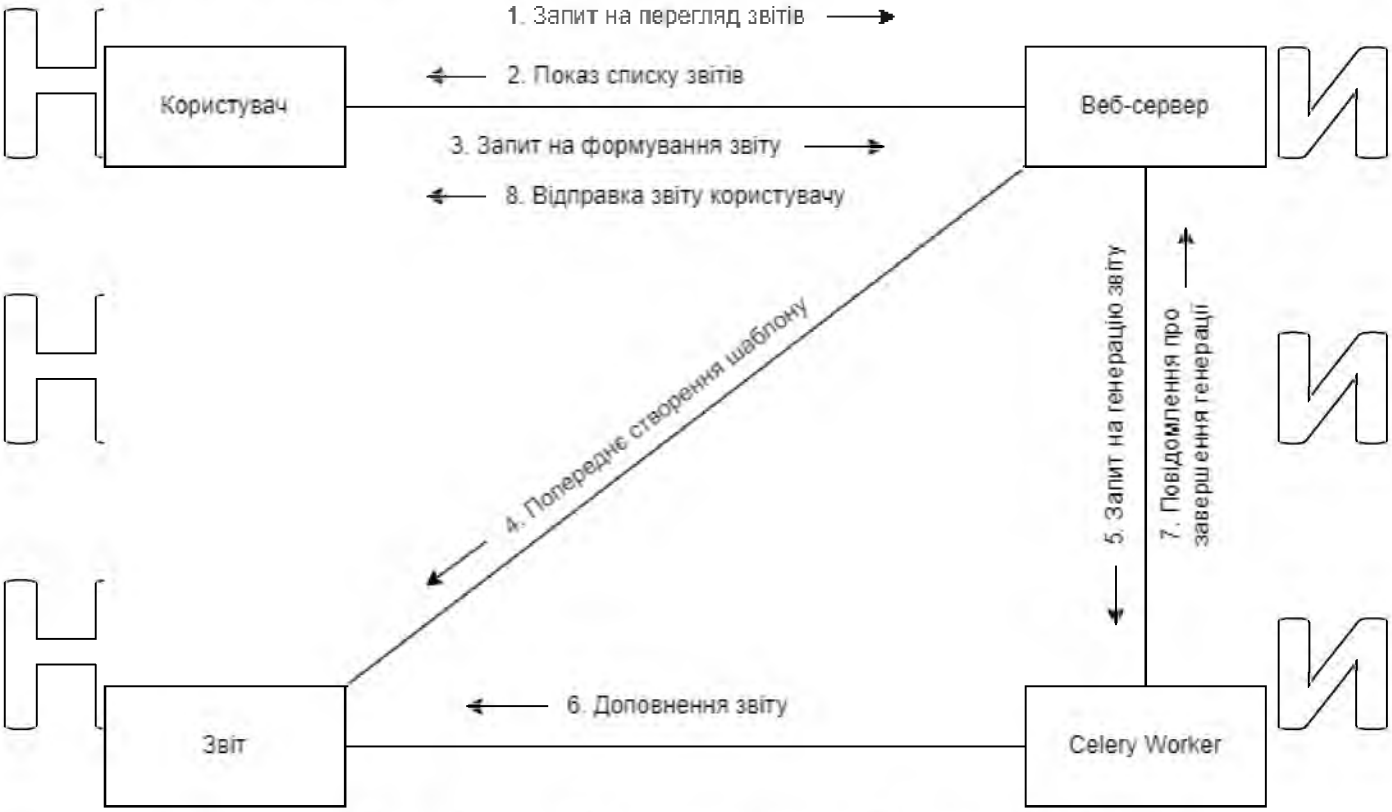


Рис. 10 Діаграма кооперації генерації звіту

На рисунку 10 показано діаграму кооперації для генерації звіту. Діаграма містить 4 об'єкти які обмінюються повідомленнями: користувач, веб-сервер,

celery worker, звіт. Нижче описані повідомлення якими об'єкти обмінюються між собою:

1. Користувач заходить на сторінку зі звітами чим автоматично посилає запит на отримання списку звітів.
2. Веб-сервер вибирає звіти, що відносяться до компанії користувача та надсилає їх користувачу.
3. Користувач заповнює форму для генерації звіту з часовими рамками записів які потрібно включити до звіту.
4. Веб-сервер створює запис звіту в базі даних та виділяє місце для файлу звіту в пам’яті.
5. Веб-сервер надсилає запит в чергу, після чого celery worker отримує запит з черги та починає генерацію звіту.
6. Після завершення генерації звіту, файл записується в пам’ять на диску та надсилається запит до бази даних, де будуть міститися додаткові дані про звіт.
7. Celery worker повідомляє веб-сервер про закінчення генерації.
8. Веб-сервер надає можливість користувачу для перегляду згенерованого звіту.

Тож за допомогою діаграми діаграми кооперації вдалося змоделювати «спілкування» між об’єктами під час генерації аналітичного звіту та показати, якими саме повідомленнями ці об’єкти обмінюються.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

3 РОЗРОБКА СИСТЕМИ

3.1 Інформаційне забезпечення системи

3.1.1 Вибір середовища проектування БД

В якості бази даних було прийнято рішення використовувати PostgreSQL.

PostgreSQL – це відома система управління базами даних з відкритим вихідним кодом. PostgreSQL написана на мові програмування C. На відміну від більшості СУБД, PostgreSQL є це не просто реляційна СУБД, а об'єктно-реляційна. Фундаментальною відмінністю об'єктно-реляційних БД є те, що вони підтримують об'єкти користувачів та їх поведінку, включаючи типи даних, домени, індекси, функції та операції. Все це робить PostgreSQL дуже гнучкою та надійною СУБД.

Переваги PostgreSQL [17]

- підтримує бази даних необмеженого розміру;
- надійні і потужні механізми реплікації і транзакцій;
- підтримка завантаження С-сумісних модулів і розширювана система вбудованих мов програмування;
- спадкування;
- легка розширюваність.

3.1.2 Реалізація фізичної моделі даних

Фізичне моделювання даних передбачає перетворення логічної моделі з суто бізнес-проекту в проект, оптимізований для роботи в конкретному середовищі. При фізичному моделюванні необхідно враховувати конкретну СУБД, апаратне середовище, частоту доступу до даних та шляхи доступу до даних. Фізичне моделювання даних передбачає додавання властивостей, таких як простір, вільний простір та індекси. Воно також може включати зміну логічних структур. Існує безліч фізичних змін, які адміністратор бази даних може вносити в логічні структури. Три основні причини відхилення від логічної моделі - це підвищення продуктивності, покращення зручності використання та економія DASD.

В базі даних показаній на рисунку 11 присутні такі сутності:

- AccountingRecord – сутність, що містить в собі дані про перекази:
  - AccountingRecordId – первинний ключ таблиці;
  - caregoryId – зовнішній ключ на категорію переказу;
  - userId – зовнішній ключ на користувача;
  - amount_of_money – кількість грощей;
  - accounting_type – тип переказу;
  - date – дата транзакції;
  - created_at – дата створення запису;
  - updated_at – дата оновлення запису.
- CategoryId – сутність, що містить в собі інформацію про категорію здійсненого переказу:
  - CategoryId – первинний ключ таблиці;
  - companyId – зовнішній ключ на таблицю компанії;
  - category_name – назва категорії;
  - created_at – дата створення категорії;
  - updated_at – дата оновлення категорії.
- AccountingUser - сутність, що містить в собі інформацію про працівника що заповнював інформацію про здійснений переказ:
  - AccountingUserId – первинний ключ таблиці;
  - companyId – зовнішній ключ на таблицю компанії;
  - username – логін користувача;
  - first_name – ім’я користувача;
  - last_name – прізвище користувача;
  - position – позиція користувача;
  - email – електронна адреса користувача;
  - password – пароль користувача;
  - is_staff – чи є користувач адміністратором системи;
  - is_active – чи є користувач активним в системі;

НУБіП України

- AccountingReport - сутність, що містить в собі інформацію про звіт що формується аналітиком:
  - AccountingReportId – первинний ключ таблиці;
  - userId – зовнішній ключ на користувача;
  - date_start – дата початку;
  - date_end – дата кінця;
  - status – статус генерації;
  - message – повідомлення про помилку якщо така була;
  - file_path – місце розміщення звіту на файловій системі;
  - created_at – дата створення звіту;
  - updated_at – дата оновлення звіту.

Є ще одна додаткова службова таблиця, яка використовується в системі для запису журналу подій, з якого буде братися інформація про дії користувачів в системі для показу їх менеджеру. Вона має наступні атрибути:

- LogId - первинний ключ таблиці;
- Content_type_id – зовнішній ключ та таблицю над якою виконують дії;
- User_id – зовнішній ключ на користувача який виконує дію;
- Action_time – час дії;
- Object_id – id об’єкта над яким виконали дію;
- Object_repr – текстове представлення об’єкта;
- Action_flag - тип дії;
- Change_message – текстове представлення змін.

НУБіП України

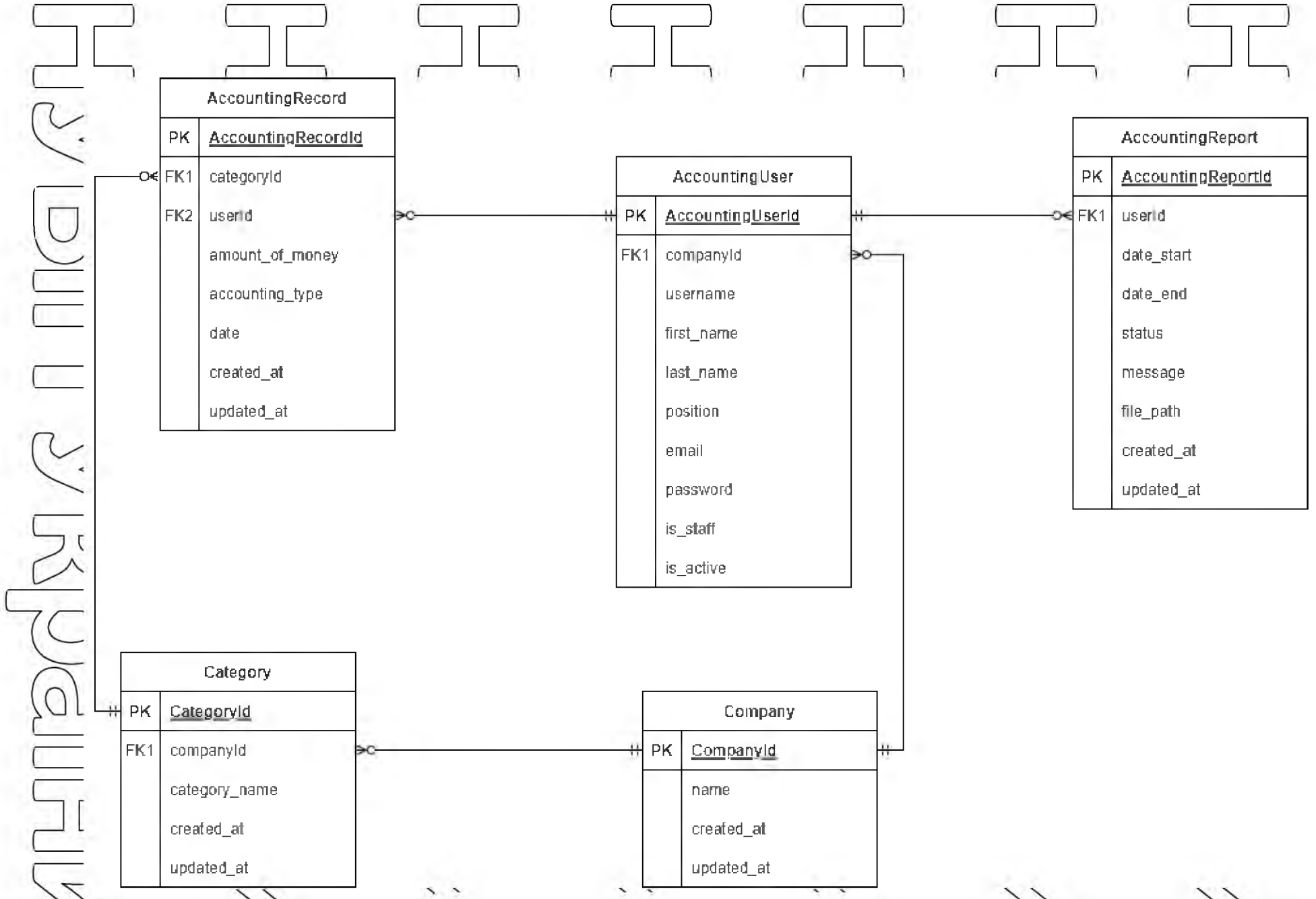


Рис. 11 Фізична модель даних

3.1.3 Проектування сховища даних

Поняття сховище даних визначене родоначальником цього напряму Уїльямом Інмоном як «предметно-орієнтоване, інтегроване, незмінне, таке що підтримує хронологію, набори даних, організоване з метою підтримання управління, покликане виступати в ролі єдиного джерела істини, що забезпечує менеджерів і аналітиків достовірною інформацією, необхідною для оперативного аналізу та прийняття рішень». Основне призначення сховища даних – надання точних даних та інформації з найменшими затратами часу і коштів.

Основний принцип його роботи був сформований автором: дані, одного разу занесені до сховища даних, у подальшому багаторазово витягуються з нього і використовуються для аналізу. Звідси випливає одна з основних переваг використання сховища даних в роботі підприємства – контроль за критично важливою інформацією, отриманою з різних джерел, як за виробничим ресурсом.

Дані в сховище надходять з баз даних, із зовнішніх джерел, наприклад статистичних звітів, «викачаних» з Інтернету прайслистів інших компаній і т.п. Наповнення інформаційних сховищ відбувається в декілька етапів:

1) екстракція (витяг) – імпорт даних у сховище з інформаційних підсистем, виробничих відділів та інших джерел, а також дані з різних зовнішніх джерел, де вони можуть мати різні імена, атрибути, одиниці виміру і способи кодування;

2) трансформація – консолідування, агрегування даних (тобто обчислюються сумарні або ін. показники), розбиття їх на фракції, коригування та трансформування у відповідні формати, „очищення” від індивідуальних ознак (тобто приведення до єдиного вигляду);

3) завантаження у сховище у вигляді „історичних пластів”, кожен з яких належить до конкретного періоду часу.

У результаті записи у таблицях сховищ даних ніколи не змінюються являючи собою «знімки даних», зроблені у визначені відрізки часу. Надмірність

даних є мінімальною, оскільки вони є відфільтрованими, сортованими і зведеними до єдиного формату.

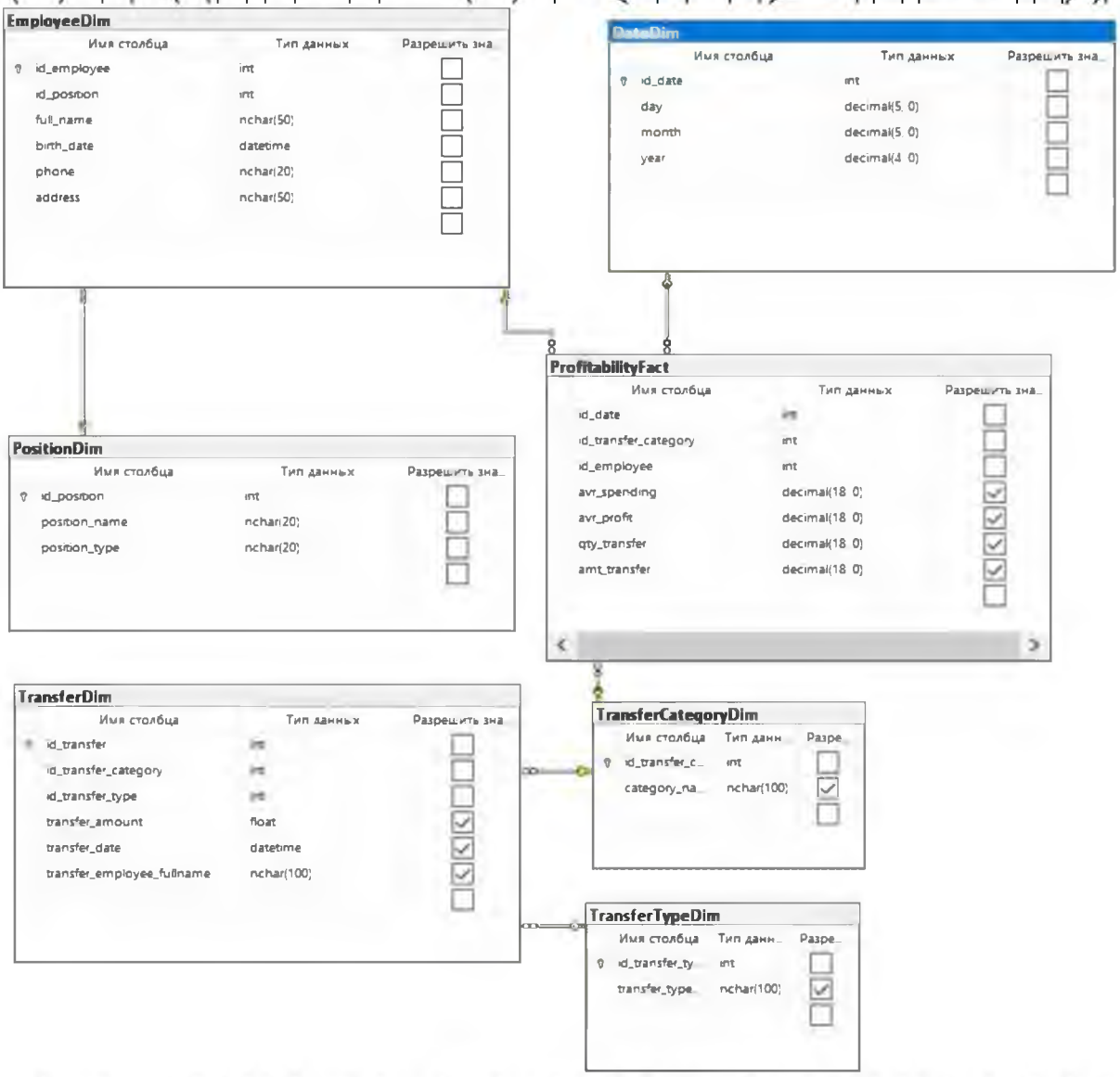


Рис. 12 Сховище даних

Вимір – це послідовність значень одного з параметрів, які аналізуються. В представленій фізичній моделі СД на рисунку 12 показані наступні таблиці вимірів:

- EmployeeDim – вимір, що містить в собі інформацію про працівника що заповнював інформацію, а саме його ім'я, позицію в підприємстві, адресу та телефон;
- PositionDim – вимір, що містить в собі інформацію про

позиції працівників в підприємстві, а саме назву та тип позиції;

- DateDim – часовий вимір, де зберігається рік, місяць та день;
- TransferDim – вимір, що містить в собі інформацію про здійснені перекази, а саме посилання на тип та категорію переказу, а також дату та суму переказу;
- TransferCategoryDim – вимір, що містить в собі інформацію про доступні категорії переказів, а саме назву категорії;
- TransferTypeDim - вимір, що містить в собі інформацію про доступні типи переказів, а саме назву типу.

Факт – використовується у СД та складається з вимірів, показників або фактів бізнес-процесу.

Таблиця ProfitabilityFact, містить інформацію у розрізі дати, категорії переказу, та працівника про:

- середні витрати;
- середні прибутки;
- кількість переказів;
- загальну суму переказів.

3.2 Програмне забезпечення системи

3.2.1 Вибір архітектури системи

Архітектура програмного забезпечення - це спосіб зображення структури програми чи комп'ютерної системи та поділу її на абстрактні системні елементи. Кожен такий елемент також може бути цілісною системою, а може бути розділений на елементи. Вивчення архітектури допоможе з'ясувати і визначити, на які частини розділити програмне забезпечення, і як ці компоненти будуть взаємодіяти між собою, як вони будуть працювати самостійно, і як вони будуть обмінюватися інформацією з іншими частинами ПЗ.

НУБіП України

На даний час є багато архітектурних шаблонів. Нижче наведені приклади основних шаблонів, які використовують для створення програмного забезпечення [19]:

- мікросервіси;
- модель-вигляд-контролер;
- клієнт-серверна архітектура;
- триярусна архітектура;
- сервісно-орієнтована архітектура;
- багаторівнева архітектура;
- керована подіями архітектура;
- канали й фільтри;

Архітектурний патерн мікросервісів набуває все більшої популярності в сучасному світі. За своєю суттю мікросервіси - це архітектурний стиль, в якому програмний додаток розбивається на невеликі сервіси, що взаємодіють між собою, але виконуються в окремих процесах. Такими сервісами також може бути програмне забезпечення, створене за різними архітектурними патернами, що дає можливість програмному забезпеченню стати більш гнучким з точки зору вибору інструментів і методів створення. Наприклад, можна використовувати різні мови програмування при написанні різних мікросервісів.

Такі мікросервіси розміщуються в різних так званих контейнерах і працюють незалежно один від одного, а з часом компілюватися в один програмний додаток. Відмінність мікросервісної архітектури від монолітної можна побачити на рисунку 13.

НУБіП України

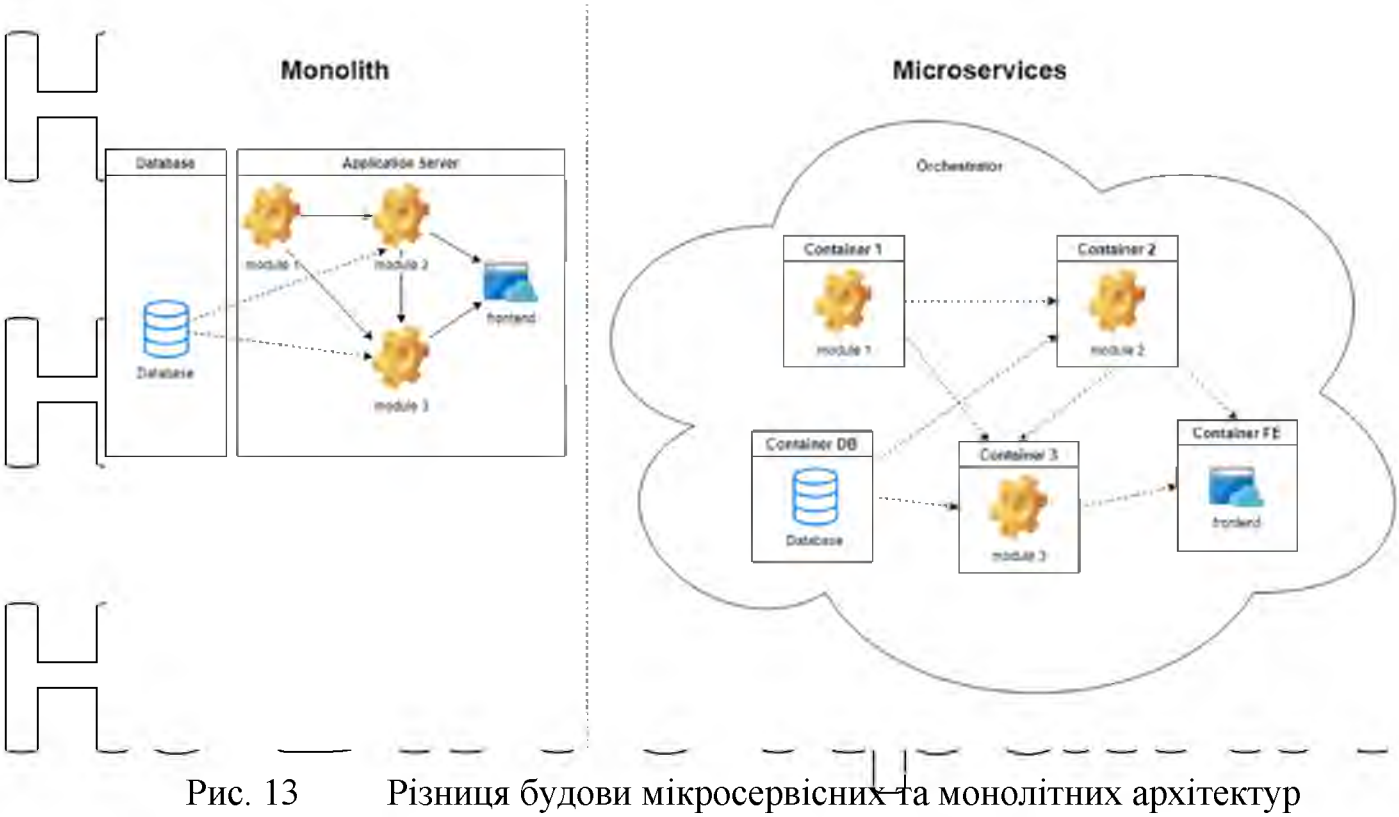


Рис. 13 Різниця будови мікросервісних та монолітних архітектур

Переваги та недоліки такої архітектури наведені у таблиці 1.

Таблиця 1 – Переваги та недоліки мікросервісної архітектури

Переваги	Недоліки
Високий рівень незалежності в розробці та впровадженні програмного забезпечення.	Складне розгортання та тестування мікросервісів.
Легко замінити одну послугу на іншу.	Використання великої кількості ресурсів для гарантування безпеки такої системи.
Ефективне використання системних ресурсів.	Дублювання програмного коду через незалежність сервісів.
Сервіси можуть бути реалізовані незалежно один від одного з використанням різних мов програмування.	Збільшення витрат на інфраструктуру, експлуатацію та моніторинг системи.

НУБіП України

В такій архітектурі можна поєднувати декілька мікросервісів з різними архітектурними патернами. Наприклад, використовувати архітектуру клієнт-сервер для передачі даних з телефону користувача на сервер. Крім того можна використовувати мікросервіс на основі архітектурного патерну модель-подання-контролер для реалізації додаткового функціоналу та розширення функціональності іншого сервісу.

Архітектуру клієнт-сервер можна представити як систему, більша частина ресурсів якої розташована на серверах, що приймають інформацію від користувачів [1] Такі системи використовують лише два компоненти:

- сервери, які зберігають та обробляють інформацію від клієнтів;
- клієнти, які використовують функції, надані сервери,

Також можна виділити мережу, як проміжний компонент, який забезпечує зв'язок між клієнтами та серверами. Структура архітектури клієнт-сервер показана на рисунку 14.

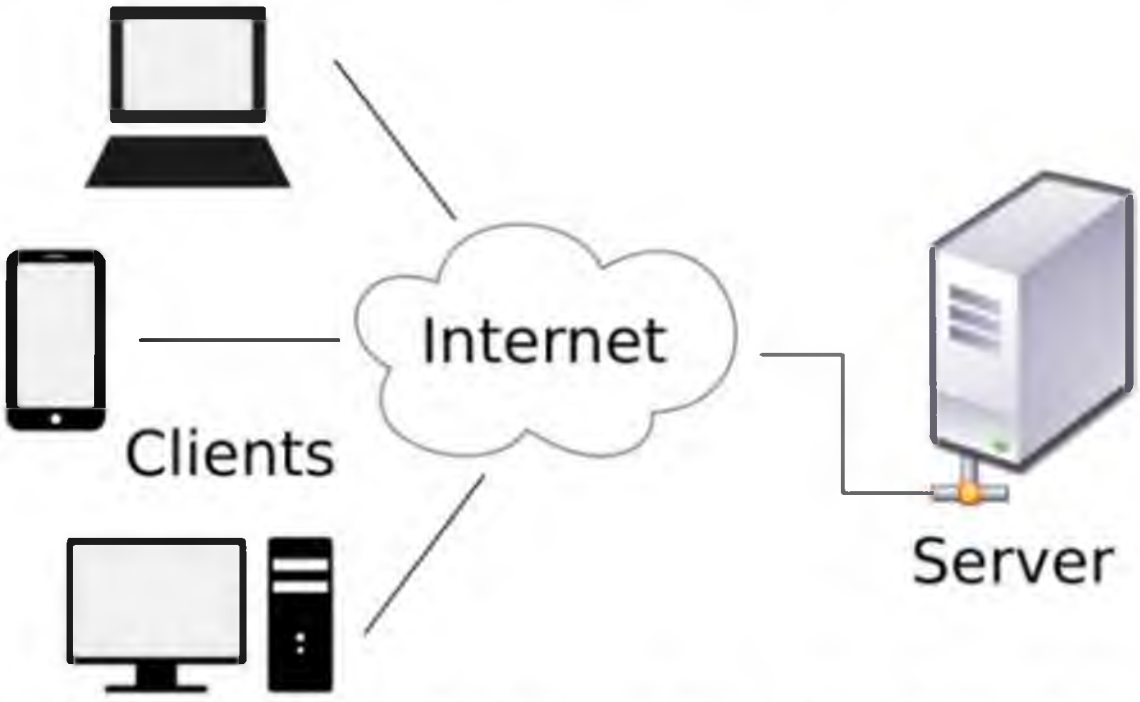


Рис. 14 Структура клієнт-серверної архітектури

Архітектура MVC дозволяє розділити код програми на 3 рівні відповідно:

- модель – рівень що взаємодіє з базою даних;
- вигляд – рівень що взаємодіє з користувачем;
- контролер - проміжний рівень, який обробляє дані, що надходять від бази даних або користувача і конвертує ці дані в команди, які надсилаються моделі або представленню.

До бурхливого розвитку мережі Інтернет ця архітектура використовувалася для звичайних програмних застосунків з графічним інтерфейсом, і лише згодом почала широко застосовуватися для створення веб-додатків. Структуру архітектурної схеми "модель-представлення-контролер" показано на рисунку 15.

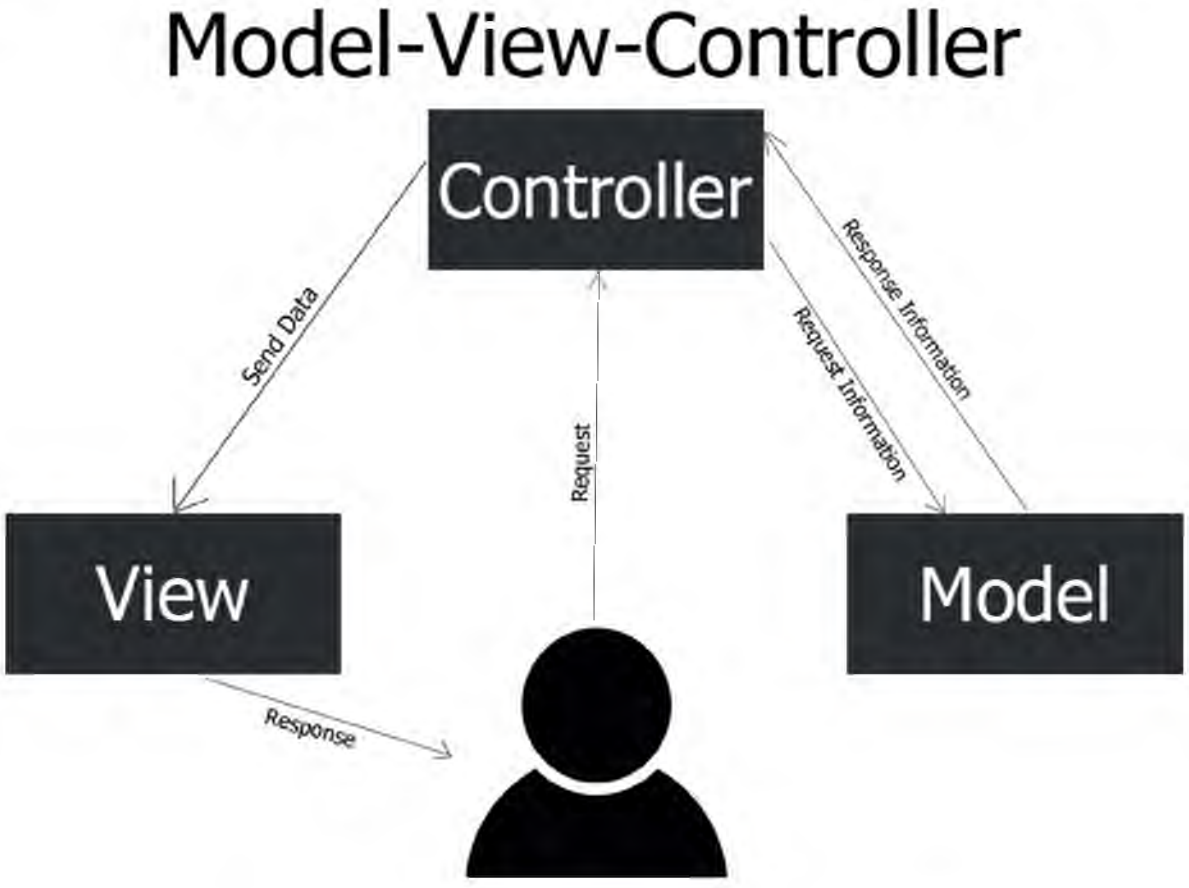


Рис. 15 Структура архітектури модель-вигляд-контролер

НУБіП України

Зрештою, було прийнято рішення об'єднати декілька типів архітектур, в основі яких лежить мікросервісна архітектура. Цей вибір надасть системі великої гнучкості та ширших можливостей у реалізації.

Тому можна зробити висновок, що ця архітектура найкраще підходить для впровадження розроблюваної системи.

3.2.2 Вибір технологій розробки системи

Коли мікросервіси обирають архітектурою системи, перш за все, потрібно переконатися, що її елементи можуть запускатися і працювати незалежно один від одного. Для цих випадків використовуються інструменти для контейнеризації, які дозволяють керувати ізольованими контейнерами. Один з найпопулярніших таких інструментів є Docker.

Він дозволяє інкапсулювати і копіювати додаток в зручні стандартизовані пакети. Це зменшує складність і невизначеність середовища. Це також значно спрощує перехід від розробки до виробництва програмних застосунків, а також скорочує час використання апаратного забезпечення. В одному фізичному боксі або віртуальній машині можна розмістити кілька контейнерів в різних середовищах (різних операційних системах).

Docker має два основних терміни, які використовуються для визначення функціональності[26]:

- контейнер - це одиниця програмного забезпечення, яка упаковує код і всі його залежності таким чином, щоб програма швидко і надійно працювала з одного обчислювального середовища в інше.
- образ контейнера Docker - це легкий, самодостатній, виконуваний програмний пакет, який включає в себе все, що необхідно для запуску програми: код, середовище виконання, системні інструменти, системні налаштування та бібліотеки.

Образи стають контейнерами під час їх виконання, а у випадку з контейнерами Docker, вони стають контейнерами, коли вони працюють на Docker Engine. Контейнерне програмне забезпечення, буде доступне як для

Windows, так і для Linux, завжди буде працювати однаково, незалежно від інфраструктури. Контейнери ізолюють ПЗ від навколишнього середовища і гарантують, що воно працює однаково, незважаючи на відмінності, наприклад, між розробкою і установкою.

Python - об'єктно-орієнтована інтерпретована мова програмування високого рівня з динамічною типізацією [18]. Ця мова програмування широко використовується практично у всіх сферах розробки програмного забезпечення. Вона досить проста у розумінні та вивченні . Має великий інструментарій та багато сторонніх модулів, які збільшують функціональність мови. Основними сферами використання мови Python є наступні галузі розробки програмного забезпечення:

- веброзробка;
- бази даних;
- системне програмування;
- машинне навчання;
- автоматизація процесів;
- ігрова індустрія.

Таким чином, можна зробити висновок, що Python є універсальною мовою програмування, яка підходить для реалізації практично будь-якого програмного забезпечення.

Для реалізації архітектури патерну модель-подання-контролер також буде використовуватися мова програмування Python у поєднанні з деякими фреймворками та модулями. Основним фреймворком для роботи буде Django.

Django - це високорівневий фреймворк з відкритим вихідним кодом на мові Python, який використовується для розробки веб-додатків[4]. Архітектура систем, побудованих на його основі, дещо відрізняється від звичної модель-вид-контролер. Розробники Django називають її model-view-template. Основна відмінність полягає в тому, що в Django "шаблоном" є "представлення", а "контролером"- "представлення". Веб-додатки на основі Django будуються з

однієї або декількох частин, які є модульними. Це ще одна суттєва відмінність цього фреймворку від інших.

Ще одним серверним компонентом є Django Rest Framework. DRF - це бібліотека, яка працює зі стандартними моделями Django і дозволяє створювати потужний і гнучкий API для комп'ютерних систем [5]

Інтерфейс прикладного програмування (API) - це набір функцій для взаємодії між різними компонентами. В основному такі інтерфейси використовуються для взаємодії програмних продуктів між собою. В даному випадку ці функції будуть використовуватися для з'єднання деяких front-end та back-end компонентів.

Для виконання фонових завдань, які можуть займати багато часу, знадобиться інструмент управління чергами завдань. Одним з найпопулярніших таких інструментів є Celery.

Celery - це проста, гнучка і надійна розподілена система для обробки величезних обсягів повідомлень, яка надає операціям інструменти, необхідні для підтримки такої системи. Це черга завдань, орієнтована на обробку в реальному часі, яка також підтримує планування завдань [2] Такий інструмент дозволяє виконувати будь-які фонові завдання, наприклад:

- формувати звіти;
- робити планові зміни в записах БД;

При використанні Celery безпосередньо з БД може виникнути проблема з великою кількість запитів до бази від фонових завдань може впливати на швидкість роботи БД, що може вплинути на швидкість роботи всього веб-додатки. У такому випадку рекомендується використовувати брокери повідоомлень.

AMQP - протокол прикладного рівня, який фокусується на обробці повідомлень. Брокер AMQP розміщує чергу в пам'яті, що знижує навантаження на базу даних, оскільки фоновим процесам Celery не обов'язково так часто відправляти запити на отримання інформації, адже така черга має механізм доставки нових задач фоновим процесам. Навіть якщо брокер з якихось причин

буде перевантажений, це не вплине на роботу веб-додатку, з яким взаємодіє користувач. В якості AMQP-брокера було вирішено використовувати платформу RabbitMQ [19], яка дозволяє компонентам програмного комплексу обмінюватися повідомленнями. Вона має підтримку таких протоколів: HTTP, STOMP, XMPP.

Для розробки клієнтської частини веб-додатку, так званого фронт-енду, було вирішено використати наступні технології:

- JavaScript;
- Бібліотека jQuery;
- Компоненти Bootstrap.

JavaScript - це об'єктно-орієнтована скриптова мова програмування. Призначена для написання сценаріїв веб-сторінок, веб-додатків, розширень браузерів, серверних додатків, десктопних та мобільних додатків[12].

jQuery - бібліотека мови програмування JavaScript, яка дає розробнику широкий спектр інструментів для кросбраузерної адресації об'єктів DOM [10] Однією зі складових цієї бібліотеки є інтерфейси для Ajax-додатків. Технологія Ajax дозволяє завантажувати дані у фоновому режимі шляхом надсилання запитів до сервера.

3.2.3 Діаграма компонентів

Діаграми компонентів ілюструють програмне забезпечення, вбудовані контролери тощо, з яких складається система. Діаграма компонентів має вищий рівень абстракції, ніж діаграма класів - зазвичай компонент реалізується одним або декількома класами (або об'єктами) під час виконання. Вони є будівельними блоками, тому компонент може в кінцевому підсумку охоплювати велику частину системи.

НУБіП України

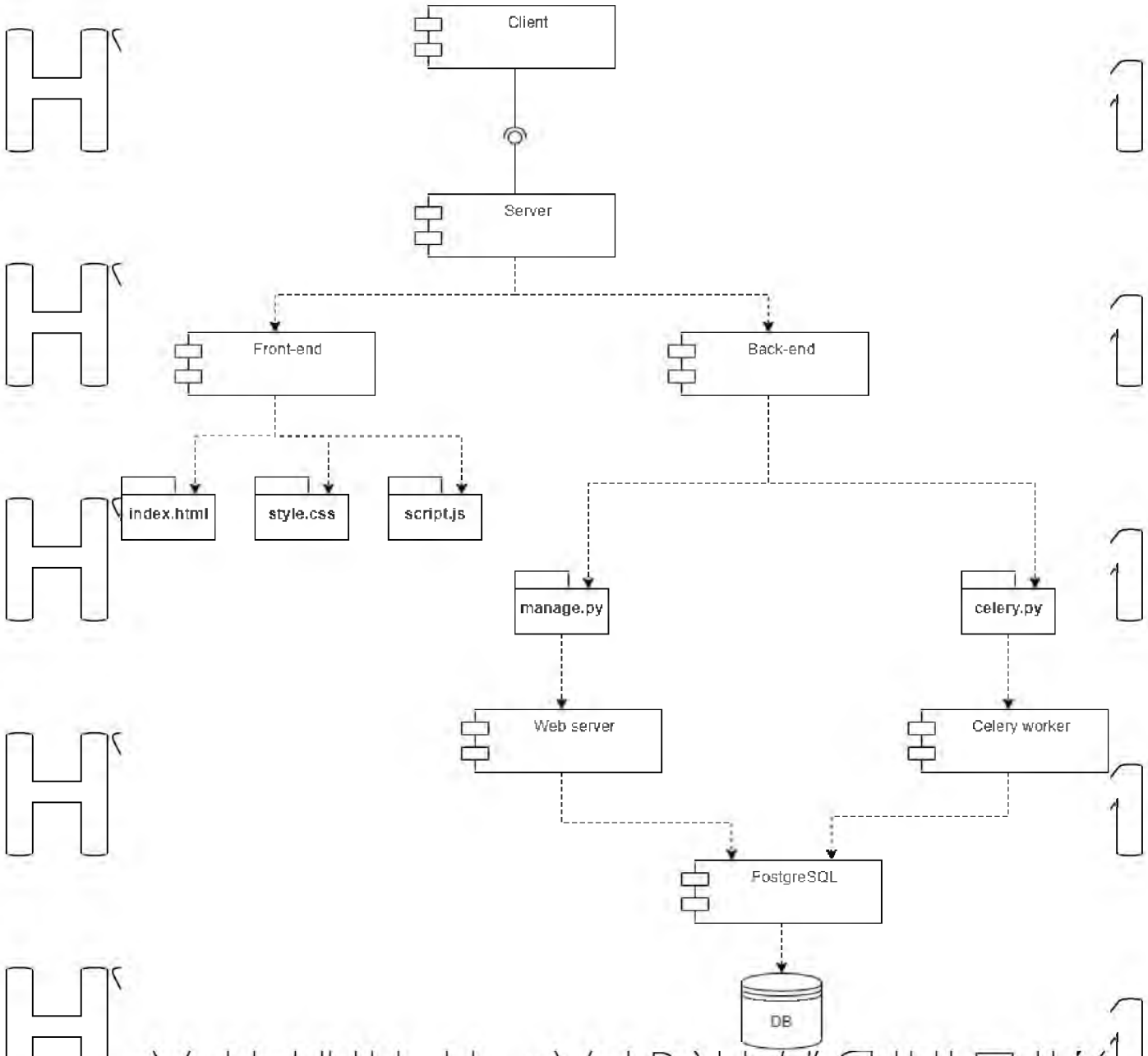


Рис. 16 Діаграма компонентів

На рисунку 16 зображена діаграма компонентів де є такі основні об’єкти:

- Client – компонент клієнта;
- Server – компонент серверу, що включає в себе компоненти Front-end та Back-end;
- Front-end – компонент що включає в себе файли, які використовуються для показу інтерфейсу веб-сайту для користувача;
- Index.html – елемент, що містить розмітку сторінки;
- Style.css – елемент, що містить стилі сторінки;

- Script.js – елемент, що містить в собі скрипти для роботи інтерфейсу;
- Manage.py – елемент, що запускає роботу веб-серверу та контролює його роботу;
- Celery.py – елемент, що запускає роботу фонового працівника та контролює його роботу;
- Web server – компонент, що відповідає за роботу веб-серверу;
- Celery worker – компонент, що відповідає за роботу фонового працівника;
- PostgreSQL – компонент, що відповідає за роботу з базою даних;
- DB – компонент бази даних.

3.2.4 Діаграма пакетів

У мові UML діаграма пакетів - це шаблон для групування елементів та визначення їх взаємозалежностей (пакетів). Основною метою діаграм пакетів є спрощення складних діаграм класів, які можуть бути використані для групування класів у пакети. Ці групи допомагають визначити ієрархію. Варто зазначити, що UML елементи в пакеті базуються на логічних зв'язках.

Діаграма пакетів - це набір визначених елементів, які семантично пов'язані і потенційно можуть змінюватися разом. Вона являє собою групування елементів моделі та визначення їх взаємозв'язків. Певною мірою, діаграма пакетів є контейнером для елементів моделі. Однак, пакет може містити всю діаграму, окреме ім'я компонента або навіть не містити жодного компонента.

Пакетна діаграма використовується для впорядкування компонентів/елементів системи високого рівня, тому пакети можуть використовуватися для великих системних організацій, які містять діаграми, документи та інші цілі проекту. Ідея полягає в тому, щоб створити систематизоване представлення набору інструкцій або процедур. Тому ви можете використовувати діаграму пакетів для того, щоб:

- спростити складні діаграми класів і організувати класи в пакети;
- визначити пакети як папки з файлами і використовувати їх на всіх діаграмах UML;
- визначати ієрархічні відносини (угрупування) між пакетами, а також іншими пакетами або об'єктами;
- створювати структуру та візуалізувати складні процеси в спрощені пакети в техніці, освіті та інших галузях, з метою наочного зображення нелінійних процесів.

На рисунку 17 зображення діаграму пакетів де є 3 основних пакети:

- User;
- Server;
- Db.

В кожному з цих пакетів є підпакети. Наприклад в пакеті «User» є такі підпакети:

- відкритий пакет «UI» – пакет, що відповідає за відображеиня інтерфейеу користувачу;
- закритий пакет «User Network» - пакет, що відповідає за інтернет з’єднання користувача.

В пакеті «Server» є такі підпакети:

- закритий пакет «Server Network» - пакет, що відповідає за інтернет з’єднання сервера;
- відкритий пакет «API access» - пакет, що відповідає за доступ до функцій системи;
- відкритий пакет «Server api» - пакет, що відповідає функції системи доступні користувачу;
- закритий пакет «Server worker» - пакет, що відповідає функцію генерації звіту;
- закритий пакет «File system» - пакет, що містить файлову систему.

В пакеті «DB» є такі підпакети:

- закритий пакет «DB access» - пакет, що відповідає за доступ до бази;
- відкритий пакет «DB GUI» - пакет, що відповідає за функції бази;
- відкритий пакет «DB Requests» - пакет, що відповідає за запити до бази;

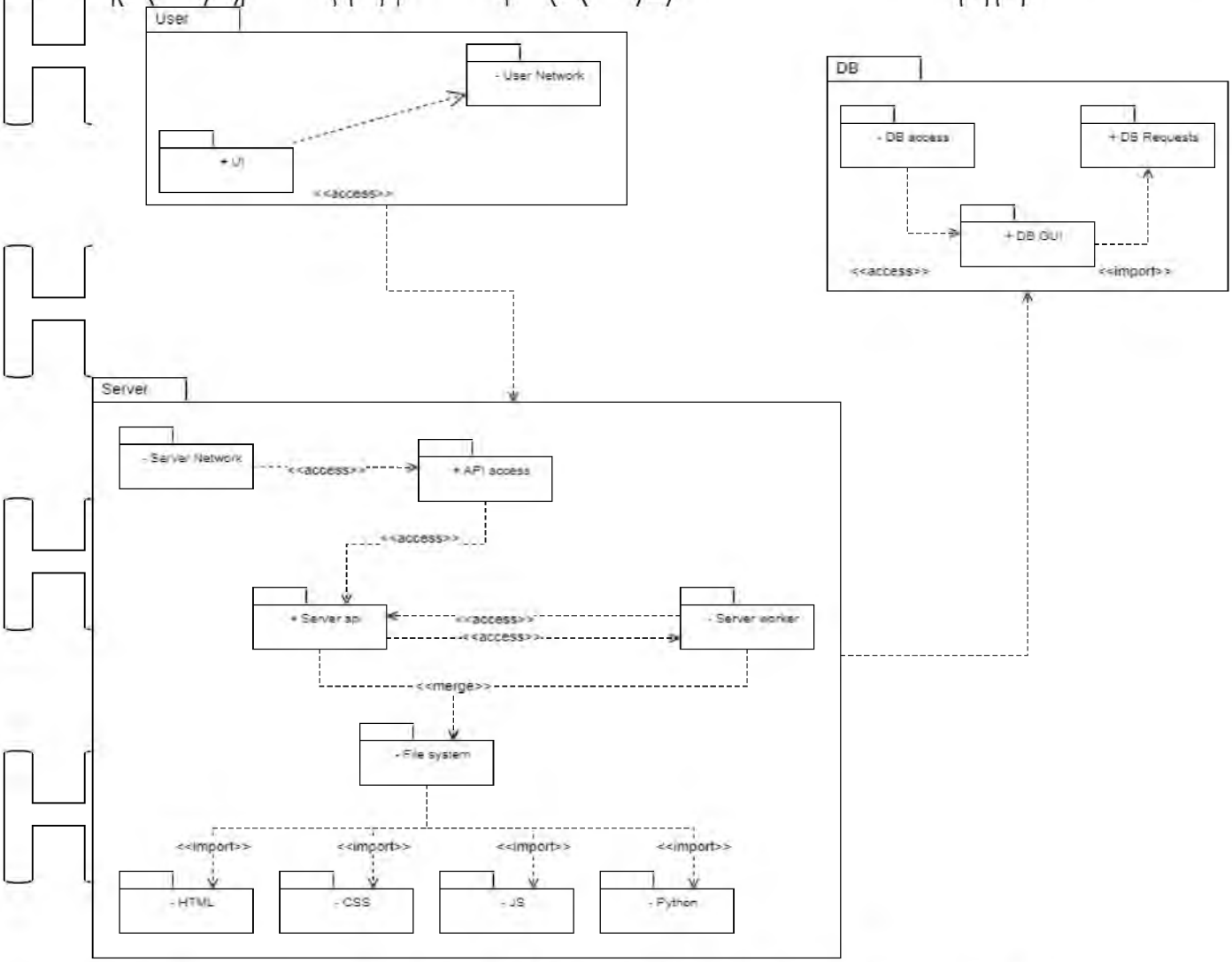


Рис. 17 Діаграма пакетів

3.2.5 Реалізація системи обраними технологіями

3.2.5.1 Створення середовища розгортання

Як зазначалося в попередніх розділах, програмна платформа Docker використовується для розгортання програмної складової комп'ютерної системи. Docker дозволяє упаковувати програмне забезпечення в контейнери, які містять

все необхідне для запуску програми, а також розгортати і масштабувати його в будь-якому середовищі.

Чому потрібно використовувати Docker:

- швидка доставка програмного забезпечення;
- стандартизація операцій;
- ефективне переміщення з локального середовища розробки на робоче;
- економія ресурсів.

У яких випадках потрібно використовувати Docker:

- для розробки мікросервісів;
- для неперервної доставки й інтеграції;
- для обробки великих даних як сервіс;
- для створення контейнерів як сервіс.

Програмне забезпечення працює всередині ізольованих контейнерів під управлінням інструментарію Docker. Кожен контейнер має власний образ, на основі якого він працює.

Образ Docker - це прототип контейнера, який містить конфігураційні файли, набори інструкцій, операційну систему та всі необхідні залежності. Образи створюються шляхом накладання нових шарів. Наприклад, береться образ операційної системи, а наступним шаром на нього встановлюється програмне забезпечення, наприклад, веб-сервер, після чого його можна записати як новий образ і зберегти.

Docker-контейнер - це запущений образ, до якого можна підключатися і виконувати різні дії. Такі контейнери також можна зупиняти і видаляти.

Комп'ютерна система буде використовувати кілька образів і контейнерів. Основний образ побудований на основі існуючого дистрибутива операційної системи з встановленою за замовчуванням мовою програмування Python, інші образи використовуються без змін. Для створення нового образу використовується Dockerfile

НУБіП України

Dockerfile - це текстовий документ, який містить інструкції та команди для створення нових образів. Створення образу для комп'ютерної системи показано у лістингу 1.

Лістинг 1 – Створення нового образу Docker

```
1   FROM python:3.9.7
2   ENV PYTHONUNBUFFERED=1
3   WORKDIR /code
4   RUN apt-get update && apt-get install -y systemd
5   COPY requirements.txt /code/
6   RUN pip install -r requirements.txt
7   COPY . /code/
8   COPY start_flower.sh /start_flower.sh
```

У лістингу 1 використовуються наступні інструкції:

- FROM python:3.9.7 – інструкція, яка показує який образ буде використаний як основа для нового, в даному випадку це дистрибутив операційної системи Linux з встановленою мовою програмування Python версії 3.9.7;
- ENV PYTHONUNBUFFERED=1 – інструкція, яка визначає, що логи системи повинні виводитися відразу ж в консоль розробника;
- WORKDIR /code – робочу папку проекту;
- RUN apt-get update && apt-get install -y system – інструкція, що запускає оновлення системи;
- COPY requirements.txt /code/ - копіює файл з залежностями, які потрібно встановити для роботи системи;
- COPY . /code/ - копіює файли проекту для роботи системи;
- COPY start_flower.sh /start_flower.sh – копіює файл, необхідний для роботи системи.

Після того, як всі інструкції внесені у файл, образ можна збирати. Для цього використовується команда «`docker build .`», яка запускає процес побудови нового образу.

Тепер образ готовий до використання і запуску в контейнері, а для того, щоб контейнери коректно запускалися і знали, які образи використовувати для запуску, потрібно описати файл Docker-compose.

Файл Docker-compose - це текстовий файл, який містить опис сервісів (контейнерів) таким чином, щоб їх можна було запускати разом в ізольованому середовищі.

У лістингу 2 наведено Docker-compose файл для комп'ютеризованої системи обліку витрат.

Лістинг 2 – Опис сервісів для комп'ютерної системи обліку витрат

```yaml
version: "3.9"
services:
  db:
    container_name: db
    image: postgres
    restart: on-failure
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - "5432:5432"
  redis:
    image: redis:latest
    container_name: redis
  web:
    build: .
    container_name: tg_money_accounting-web-1
    command: python manage.py runserver 0.0.0.0:8000
    restart: on-failure
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    volumes:
      - .:/code
    environment:
      - TZ=Europe/Kiev
      - RABBITMQ_DEFAULT_USER=admin
      - RABBITMQ_DEFAULT_PASS=admin
    ports:
      - 5672:5672
      - 15672:15672
  worker:
    build: .
    container_name: celery_worker
    restart: on-failure
    command: celery -A diplom worker -E -B -c 1 -l info
    environment:
      DJANGO_SETTINGS_MODULE: diplom.settings
    volumes:
      - .:/code
    depends_on:
      - db
      - rabbitmq
      - redis
    links:
      - rabbitmq
      - redis
  flower:
    image: mher/flower
    restart: on-failure
    environment:
      - CELERY_BROKER_URL=amqp://admin:admin@rabbitmq//
      - FLOWER_PORT=5556
    ports:
      - 5556:5556
    depends_on:
      - worker
      - rabbitmq
      - redis
```

У лістингу 2 наведено опис основних сервісів комп'ютеризованої системи обліку витрат. Він використовує наступні параметри:

- container_name - назва контейнера;
- image – базове зображення контейнера;
- restart - перезапуск контейнера у разі виникнення помилок тощо;
- environment - змінні, які будуть записані в оточенні операційної

системи;

- ports - порти, які будуть використовуватись контейнерами;
- build - налаштування, яке вказує місце розташування Docker-файлу для збірки образу контейнера;
- volumes - налаштування, що вказує розташування томів даних, які використовуються Docker-контейнерами;
- depends_on - залежності від інших контейнерів;
- command - параметр, який запускає команду після запуску контейнера.

Для запуску контейнерів в Docker використовується команда "docker-compose up". Для зручності розробки було створено Makefile, який містить в собі основні команди для розробки ПЗ. На лістингу 3 показані команди, які використовувались для розробки.

Лістинг 3 – Makefile з командами для розробки

```
run:
    sudo docker-compose up

stop:
    sudo docker-compose stop

build:
    sudo docker-compose up --build

connect:
    sudo docker exec -it tg_money_accounting-web-1 bash

shell:
    sudo docker exec -it tg_money_accounting-web-1 python manage.py shell

migrate_db:
    sudo docker exec -it tg_money_accounting-web-1 python manage.py makemigrations
    sudo docker exec -it tg_money_accounting-web-1 python manage.py migrate
```

Далі перелічено команди які використовуються та їх призначення:

- run – запуск контейнерів з ПЗ;
- stop – зупинка контейнерів;
- build – створення контейнерів;
- connect – підключення до контейнеру веб-сервера;

- shell – підключення до середовища Python в контейнері веб-сервера;
- migrate_db – створення міграційних файлів та проведення міграцій бази даних.

НУБіП України

3.2.5.2 Створення веб-серверу системи

Як вже зазначалося у виборі технологій розробки, основною мовою програмування цієї системи є мова програмування Python, а також фреймворки для створення серверного програмного забезпечення Django та Django Rest Framework. Для роботи з базою даних в Django використовується вбудована система ORM.

Об'єктно-реляційне відображення (ORM) - це технологія, яка дозволяє запитувати та маніпулювати даними з бази даних, використовуючи об'єктно-орієнтовану парадигму. Говорячи про ORM, більшість людей мають на увазі бібліотеку, яка реалізує техніку об'єктно-реляційного відображення, звідси і словосполучення "ORM".

Лістинг 4 - Модель AccountingRecord для роботи з таблицею в базі даних

```
class AccountingRecord(models.Model):
    PROFIT = 'Прибуток'
    EXPENSE = 'Витрата'
    ACCOUNTING_TYPE = ((PROFIT, 'Прибуток'), (EXPENSE, 'Витрата'))

    amount_of_money = models.FloatField()
    accounting_type = models.CharField(
        max_length=20, choices=ACCOUNTING_TYPE, default=PROFIT)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    spent_date = models.DateTimeField(default=timezone.now())
    user_tg = models.ForeignKey(
        AccountingUser, on_delete=models.CASCADE)
    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    class Meta:
        app_label = 'accounting'
        db_table = 'acc_record'
        verbose_name = 'Запис'
        verbose_name_plural = 'Записи'

    def __str__(self):
        return f'Запис:\n категорія - {self.category}\n тип запису - {self.accounting_type}'
```

В лістингу 4 показано ORM модель для роботи з таблицею AccountingRecord в базі даних. Така модель автоматично створює первинний ключ. Перелік атрибутів моделі:

- amount_of_money – кількість грошей в транзакції. Визначається полем з плаваючою точкою;
- accounting_type – тип транзакції. Визначається вибором між значеннями «Прибуток» або «Витрата»;
- category – категорія транзакції, є зовнішнім ключем до таблиці «Category»;
- spent_date – дата транзакції. Визначається типом «DateTime»;
- user_tg – користувач що вносить дані в БД, є зовнішнім ключем до таблиці AccountingUser;
- company – компанія якій належить цей запис, є зовнішнім ключем до таблиці Company.

Також ця модель має додатковий клас Meta, який визначає такі налаштування, як назву моделі в контексті ПЗ, назву таблиці в базі даних та відображення записів в однині та множині. А також має додатковий метод «__str__» який описує відображення об’єктів з бази в системі.

На лістингу 5 показаний програмний код, який використовується для імпорту та експорту транзакцій в та з бази даних. Нижче описані атрибути та методи які описують поведінку веб-серверу:

- template_name – html файл, який використовується для відображення даних на інтерфейсі користувача;
- success_url – посилання, на яке переходить користувач після успішного використання імпорта чи експорта даних;
- permission_required – атрибут, який містить список дозволів, які повинен мати користувач для доступу до сторінки;
- get_context_data – метод, який визначає початкові дані для відображення користувачу. В даному випадку він визначає 2 форми які будуть передаватись на інтерфейс користувача;

НУБіП України

• post – метод, що визначає поведінку веб-сервера, коли користувач надсилає заповнену форму. Спочатку програма визначає тип заповненої форми, після чого визначає подальші дії. У випадку експорту програма перевіряє правильність заповнення форми, після чого формує CSV файл з записами з бази даних. У випадку імпорту програма також перевіряє форму та відправляє завантажений файл до фонового працівника, який на фоні завантажує записи з файла до БД.

Лістинг 5 – Програмний код що відповідає за імпорт та експорт транзакцій

```
class AccountingExportImportView(LoginRequiredMixin, PermissionRequiredMixin, TemplateView):
    template_name = 'main_page/accounting_export_import.html'
    success_url = '/accounting-export-import/'
    permission_required = ('accounting.add_accountingrecord',
                           'accounting.view_accountingrecord')

    def get_context_data(self, *, object_list=None, **kwargs):
        context = {}
        context['form_export'] = AccountingExportForm()
        context['form_import'] = AccountingImportForm()
        return context

    def post(self, request, *args, **kwargs):
        request_type = request.POST['request_type']
        if request_type == "export":
            form_export = AccountingExportForm(request.POST)
            if form_export.is_valid():
                csv_df = form_export_file(request.user, form_export.cleaned_data)
                response = HttpResponse(content_type='text/csv')
                response['Content-Disposition'] = 'attachment; filename=export.csv'
                response.write(u'\ufeff'.encode('utf8'))
                csv_df.to_csv(path_or_buf=response, index=False)
                return response
        else:
            form_import = AccountingImportForm(request.POST, request.FILES)
            if form_import.is_valid():
                temp_file = str(tempfile.NamedTemporaryFile(delete=False, suffix='.csv'))
                with open(temp_file, 'wb') as file:
                    file.write(request.FILES['csv_file'].read())
                import_accounting_from_csv.delay(temp_file, request.user.username)
                return redirect('accounting_export_import')
        return render(request, self.template_name, context={'form_export': AccountingExportForm})
```

Лістинг решти моделей для зв'язку з базою та функцій, що відповідають за обробку запитів на кінцеві точки (endpoint) веб-серверу знаходяться в додатку А.

3.2.5.3 Створення аналітичного блоку

Для формування звітності також використовується мова Python в зв'язці з декількома бібліотеками для обробки даних. Нижче описаний перелік використовуваних бібліотек:

- pandas - програмна бібліотека, написана для мови програмування Python, призначена для маніпулювання та аналізу даних. Зокрема, вона пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами[16];
- sklearn - це вільно поширювана бібліотека машинного навчання для мови програмування Python. Вона містить різноманітні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, градієнтний бустинг, k-середні та DBSCAN, і призначена для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy[20];
- numpy - бібліотека для мови програмування Python, що додає підтримку великих багатовимірних масивів і матриць, а також велику колекцію високорівневих математичних функцій для роботи з цими масивами[14];
- matplotlib - бібліотека побудови графіків для мови програмування Python та її розширення для чисельної математики NumPy[13];
- statistics – модуль, що надає функції для обчислення математичної статистики числових[12].

Аналітичний модуль системи використовує моделі машинного навчання для аналізу даних, побудови графіків та загалом у формуванні звіту. В аналітичному модулі використовуються такі алгоритми для аналізу даних, як лінійна регресія, дерево прийняття рішень та кластеризація.

В лістингу 6 показано використання бібліотеки машинного навчання «skelearn» для тренування моделі «DecisionTree», що використовує алгоритм дерева рішень.

Лістинг 6 – Реалізація алгоритму дерева рішень

НУБіП України

```
def train_tree_model(records: list[AccountingRecord]):
    X_ = np.array([[datetime.datetime.toordinal(record.spent_date),
                    record.amount_of_money] for record in records])
    y_ = np.array([record.category.category_name for record in records])
    X_train, X_test, y_train, y_test = train_test_split(
        X_, y_, test_size=0.10, random_state=42)
    clf = tree.DecisionTreeClassifier(max_depth=3)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    return clf, X_test, y_test, y_pred
```

У функцію передаються записи з бази даних, після чого йде поділ даних на цільові дані та дані, на базі який відбувається класифікація. Після чого береться вибірка даних для тренування моделі та саме тренування моделі. Далі відбувається передбачення після чого ці дані повертаються до головного потоку роботи програми.

Лістинг 7 – Програмний код відображення результатів алгоритму дерева рішень

```
clf, X_test, y_test, y_pred = train_tree_model(records_profit)
pdf.add_page(orientation='L')
fig = plt.figure()
_ = tree.plot_tree(clf,
                   feature_names=["spent_date", "amount_of_money"],
                   class_names=category_list,
                   filled=True)
fig.suptitle('Дерево рішень прибутків', fontsize=12)
plt.savefig(earn_tree_plot_name, dpi=200, transparent=True)
pdf.image(earn_tree_plot_name, x=25, w=230, h=155)
pdf.set_font('FreeSans', '', 11)
pdf.cell(0, th, 'Метрики моделі:', 0, 2, align='L')

category_number_list = {category: i for i,
                        category in enumerate(category_list)}
y_test = [category_number_list[category] for category in y_test]
y_pred = [category_number_list[category] for category in y_pred]
tree_metrics = get_model_metrics(y_test, y_pred)
for metric in tree_metrics.keys():
    pdf.cell(
        0, th, f'  {metric} = {tree_metrics[metric]}', 0, 2, align='L')

number_category_list = {i: category for i,
                        category in enumerate(category_list)}
```

В лістингу 7 показаний приклад того, як програмним кодом відображаються результати роботи алгоритму дерева рішень. Після отримання

результатів, вони відображаються на графіку за допомогою метода «plot_tree», після чого графік зберігається та прикріплюється до звіту після назви. Далі йде відображення основних метрик моделі:

- mean_squared_error – середньоквадратична похибка;
- r2_score – коефіцієнт детермінації.

Решта програмного коду аналітичного модуля показана в додатку Б.

3.2.6 Поглиблене дослідження даних за допомогою інструментів OLAP-технологій

3.2.6.1 OLAP технології

Онлайн-аналітична обробка (OLAP) - це технологія, яка організовує великі бізнес-бази даних і підтримує складний аналіз. Вона може використовуватися для виконання складних аналітичних запитів без негативного впливу на транзакційні системи[15].

Бази даних, які бізнес використовує для зберігання всіх своїх транзакцій і записів, називаються базами даних онлайнової обробки транзакцій (OLTP). Ці бази даних зазвичай мають записи, які вводяться по одному. Часто вони містять велику кількість інформації, яка є цінною для організації. Однак бази даних, які використовуються для OLTP, не були розроблені для аналізу. Тому отримання відповідей з цих баз даних є дорогим з точки зору часу та зусиль. OLAP-системи були розроблені для того, щоб допомогти витягти цю бізнес-аналітичну інформацію з даних у високопродуктивний спосіб. Це пов'язано з тим, що бази даних OLAP оптимізовані для великих навантажень на читання і малих навантажень на запис.

Розглянемо OLAP в наступних сценаріях:

- Вам необхідно швидко виконувати складні аналітичні та спеціальні запити, без негативного впливу на ваші OLTP-системи.
- Ви хочете надати бізнес-користувачам простий спосіб формування звітів з ваших даних
- Ви хочете забезпечити ряд агрегацій, які дозволять користувачам

отримувати швидкі, узгоджені результати.

OLAP особливо корисний для застосування агрегованих обчислень над великими обсягами даних. OLAP-системи оптимізовані для сценаріїв, що вимагають великих обсягів читання, таких як аналітика і бізнес-аналітика. OLAP дозволяє користувачам сегментувати багатовимірні дані на зрізи, які можна переглядати у двох вимірах (наприклад, зведена таблиця) або фільтрувати дані за певними значеннями. Цей процес іноді називають "нарізкою і нарізкою" даних, і він може бути виконаний незалежно від того, чи розділені дані між кількома джерелами даних. Це допомагає користувачам знаходити тенденції, виявляти закономірності та досліджувати дані без необхідності знати деталі традиційного аналізу даних.

Семантичні моделі можуть допомогти бізнес-користувачам абстрагуватися від складних взаємозв'язків і полегшити швидкий аналіз даних.

3.2.6.2 Створення та розгортання кубу

Як механізм вилучення, обробки та передачі даних буде використовуватись програмний засіб Visual Studio з розширеннями Analysis Services та Integration Services.

Розширення Analysis Services використовується для створення та розгортання кубу. Приклад розгортання наведений далі.

Першим кроком для початку роботи з розширенням Analysis Services є встановлення цього розширення в середовищі Visual Studio.

Після встановлення можна розпочинати роботу з розширенням, створивши проект використовуючи новий шаблон. Далі потрібно підключити нове джерело даних. В даному випадку це сховище даних створене використовуючи SQL Management Studio.

Після підключення джерела даних потрібно обрати які таблиці будуть використовуватись у побудові куба. Цей процес показаний на рисунках 18.

НУБіП України

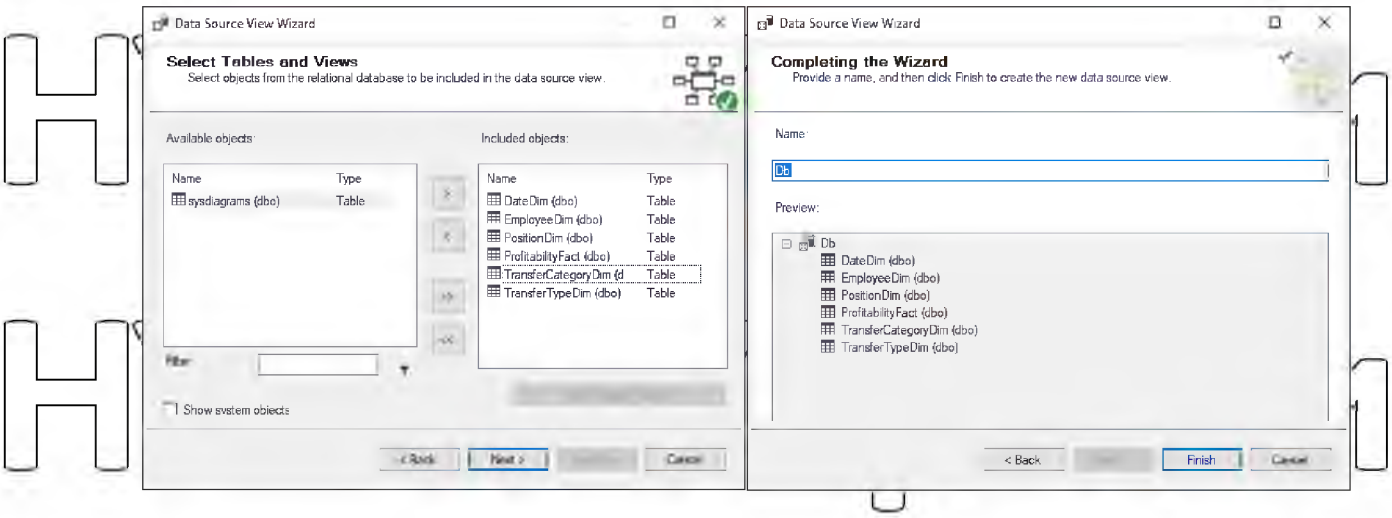


Рис. 18 Вибір потрібних таблиць

Далі потрібно створити виміри використовуючи існуючі таблиці. Створення вимірів у розширенні Analysis Services показано на рисунку 19.

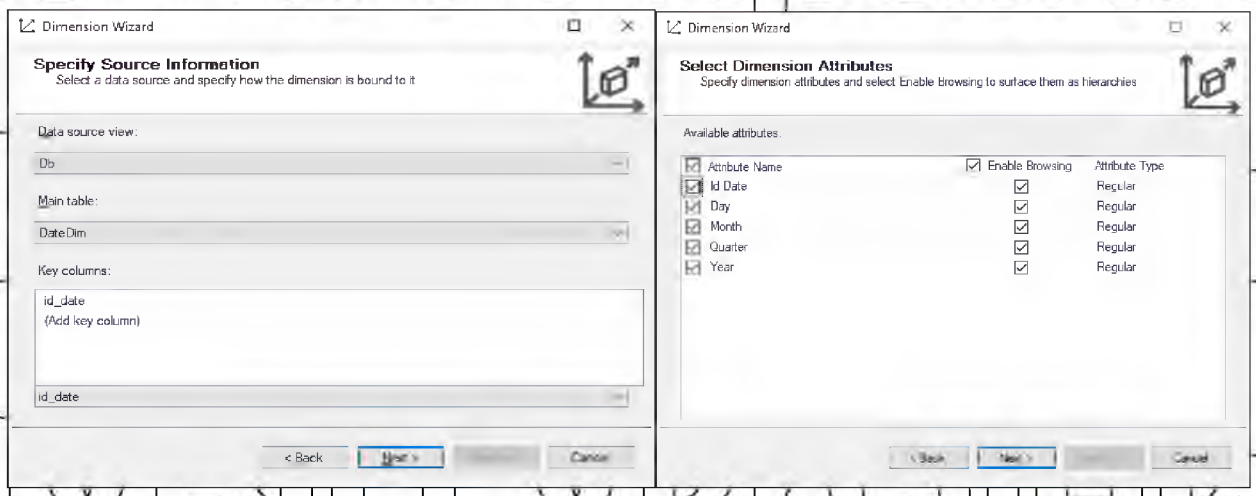


Рис. 19 Створення нового виміру

Після створення всіх вимірів можна починати створювати куб. Для цього потрібно обрати створені виміри після чого середовище самостійно організує зв'язки між ними та відобразить вигляд кубу. Вигляд створеного кубу показаний на рисунку 20.

НУБіП України

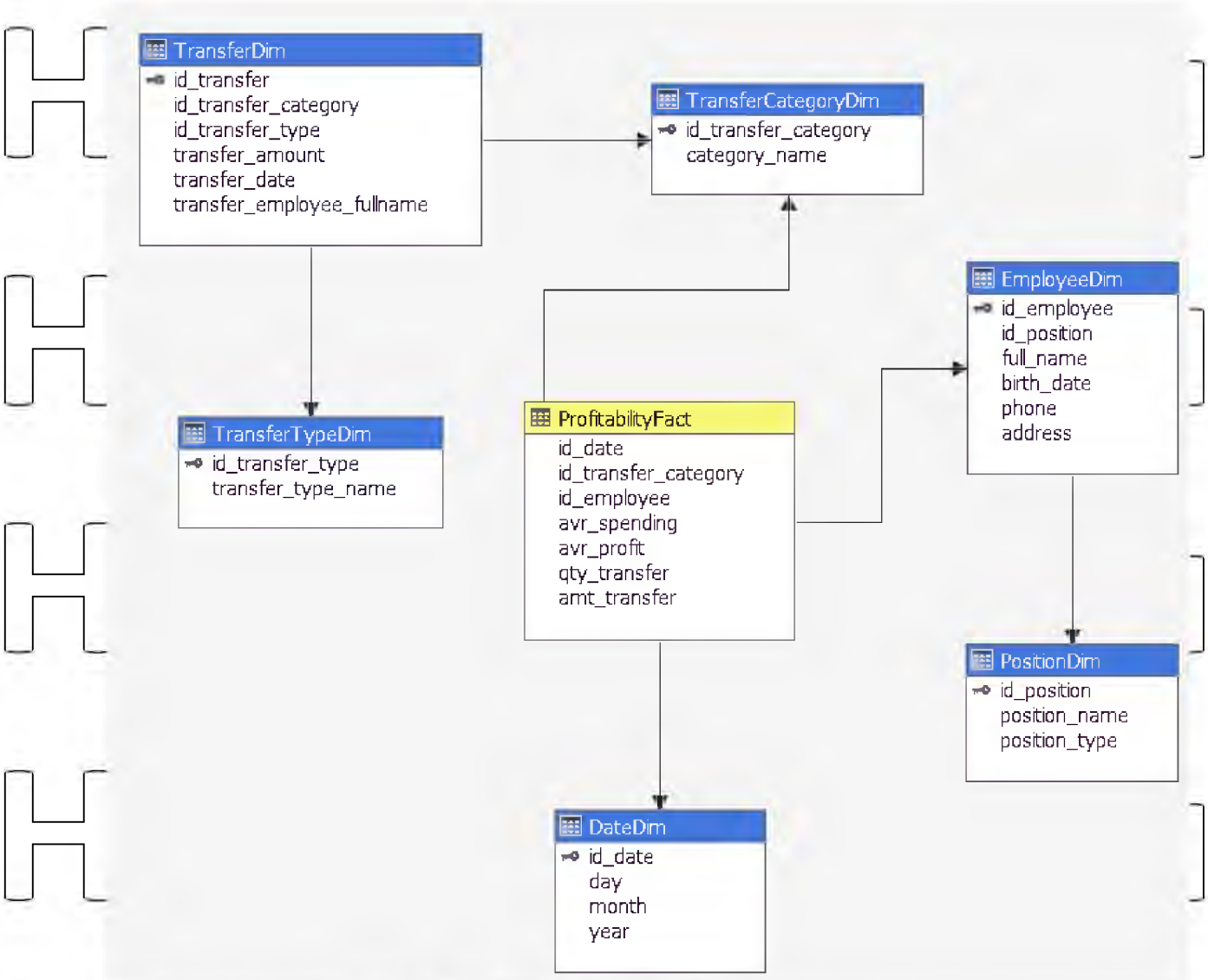


Рис. 20 Вигляд створеного кубу

Для заповнення таблиці фактів використовується розширення Integration Services для побудови потоків обробки даних Data Flow. Для цього потрібно створити новий проект в Visual Studio використовуючи новий шаблон.

Після створення проекту можна починати створення потоків даних. Для цього в проект потрібно додавати потоки, після чого можна перейти до налаштування потоків та визначення даних, які вони будуть обробляти. В даному випадку потоки даних будуть виглядати як представлено на рисунку 21.

НУБіП України
НУБіП України
НУБіП України

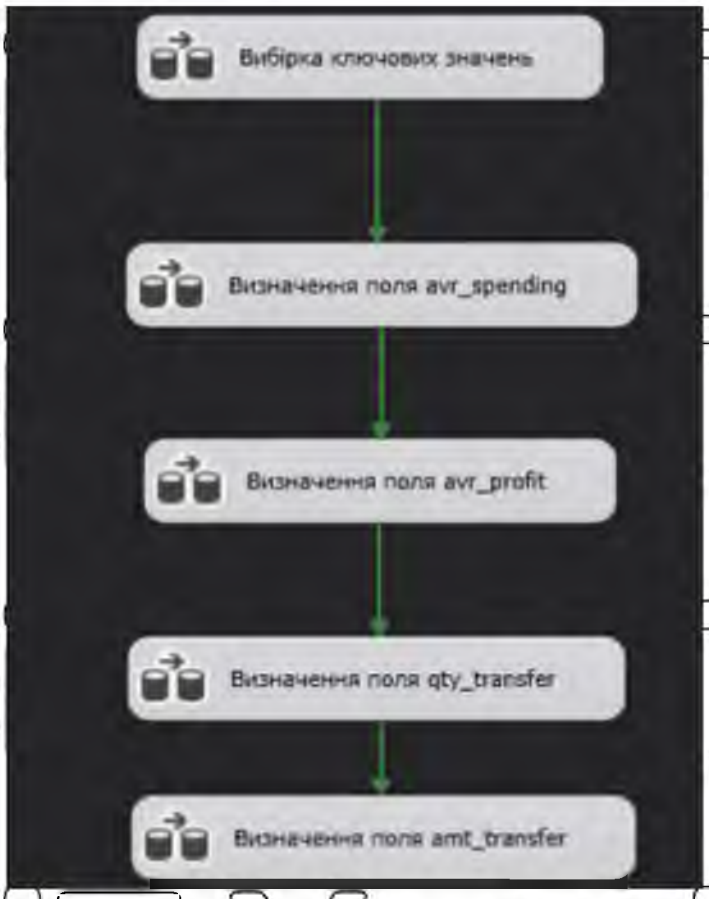


Рис. 21 Представлення Data Flow

Для налаштування потоків використовуються блоки, що обчислюють дані. Обробка даних показана на рисунку 22:

- Вибірка потрібних значень для розрахунків – блок де за допомогою SQL запитів з таблиць вибираються дані для обробки. Приклад такого запиту показаний на рисунку 23;
- Обнулення непортібних значень – непотрібні при обчисленні дані обнуляються, щоб не впливати на результати обчислень;
- Функція агрегування – блок де виконуються основні обчислення та агрегування даних.
- Обробка нульових значень потрібна для того щоб в СД не зберігалися значення NULL;
- Обробка нових значень та завантаження потрібні щоб співставити отримані результати з таблицею фактів.

НУБіП України

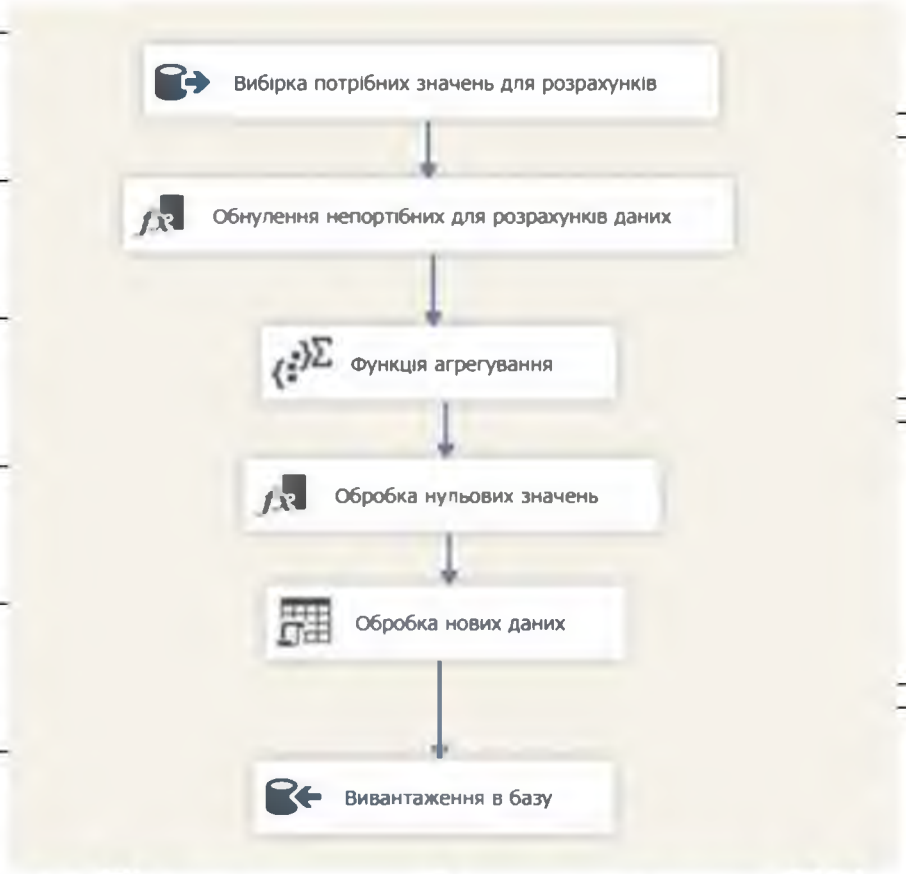


Рис. 22 Блокове представлення потоку даних

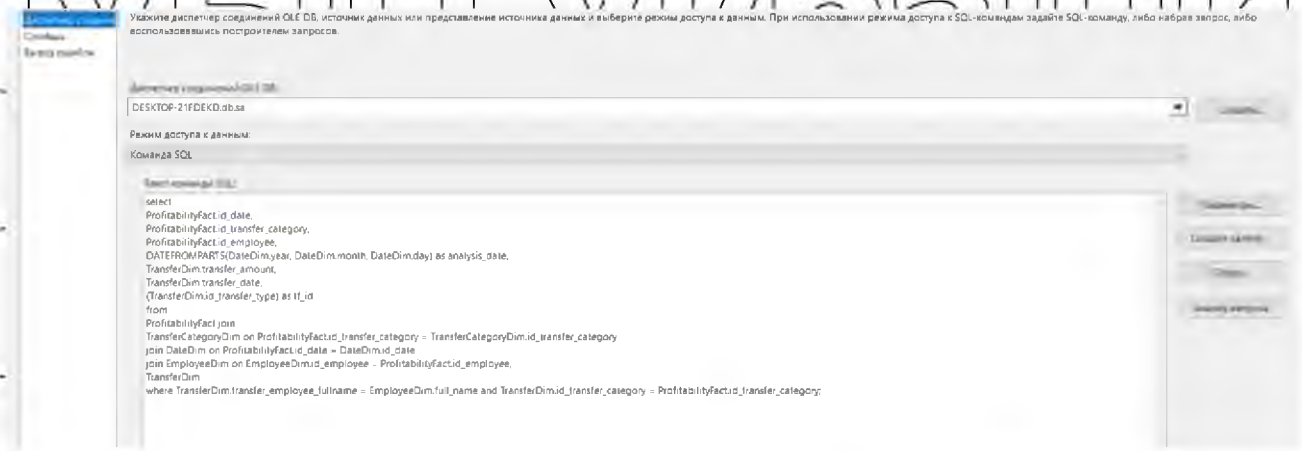


Рис. 23 Вибірка даних за допомогою SQL запитів

Результатом цих обчислень є заповнена таблиця фактів в СД. Приклад заповнення показаний на рисунку 24.

Результаты | Сообщения

	id_date	id_transfer_category	id_employee	avr_spending	avr_profit	qty_transfer	amt_transfer
1	7	2	4	6000	3750	3	13500
2	2	2	3	2500	0	1	2500
3	1	2	4	6000	3750	3	13500
4	6	2	4	6000	3750	3	13500
5	2	1	3	2833	2500	4	11000
6	5	2	4	6000	3750	3	13500
7	9	1	4	4666	6000	5	26000
8	4	2	4	6000	3750	3	13500
9	8	1	4	4666	6000	5	26000
10	9	1	3	2500	2666	12	31000
11	7	1	4	4666	6000	5	26000

Рис. 24 Результат заповнення таблиці фактів

НУБіП України

3.2.6.3 Побудова звітності в середовищі BI

Для побудови звітності використовується середовище Visual Studio з розширенням Reporting Services. Як і у випадку з створенням кубу, для початку роботи потрібно створити проект використовуючи шаблон серверу звітів.

Після створення проект потрібно підключити джерело даних, з яких будуть формуватися звіти. Наступним після обрання джерела даних кроком буде вибірка даних з яких буде формуватися звіт. Це відбувається за допомогою SQL запитів як показано на рисунку 25.

НУБіП України

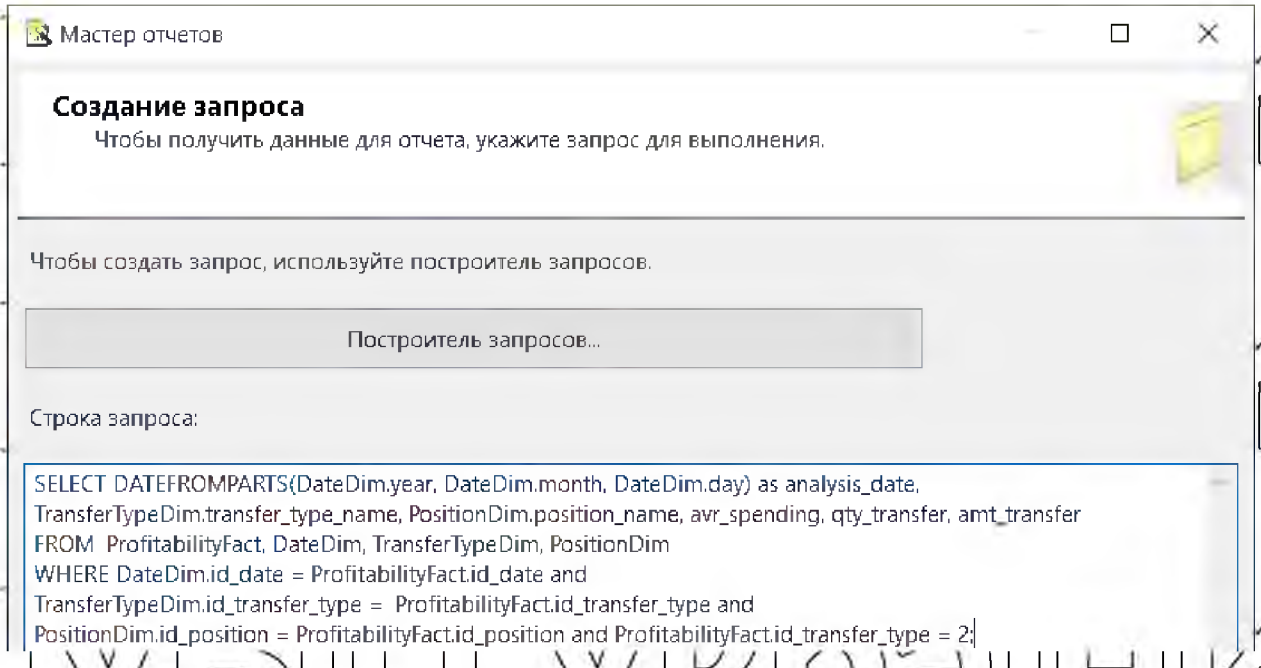


Рис. 25 SQL запит вибірки даних для формування звітів

НУБіП України

Наступним кроком йде оформлення шаблону звіту, де потрібно розташувати дані так, як вони будуть знаходитися у сформованому звіті. Приклад шаблону показаний на рисунку 26.

Рис. 26 Шаблон звіту

Результат формування звіту показаний на рисунку 27.

Рис 27 Результат формування звіту

3.2.6.4 Розрахунки KPI

KPI (Key Performance Indicator) - це показник досягнення успіху в певній діяльності або в досягненні певних цілей. Можна сказати, що KPI - це кількісно вимірний індикатор фактично досягнутих результатів. KPI дозволяють

проводити контроль ділової активності співробітників, підрозділів та компанії в цілому.

Для створення KPI буде використовуватися те ж середовище, що використовувалось при створенні кубу.

В даному випадку буде створено два ключових показники:

- показник витрат;
- показник прибутків;

Створення цих показників ефективності показане на рисунках 28 та 29.

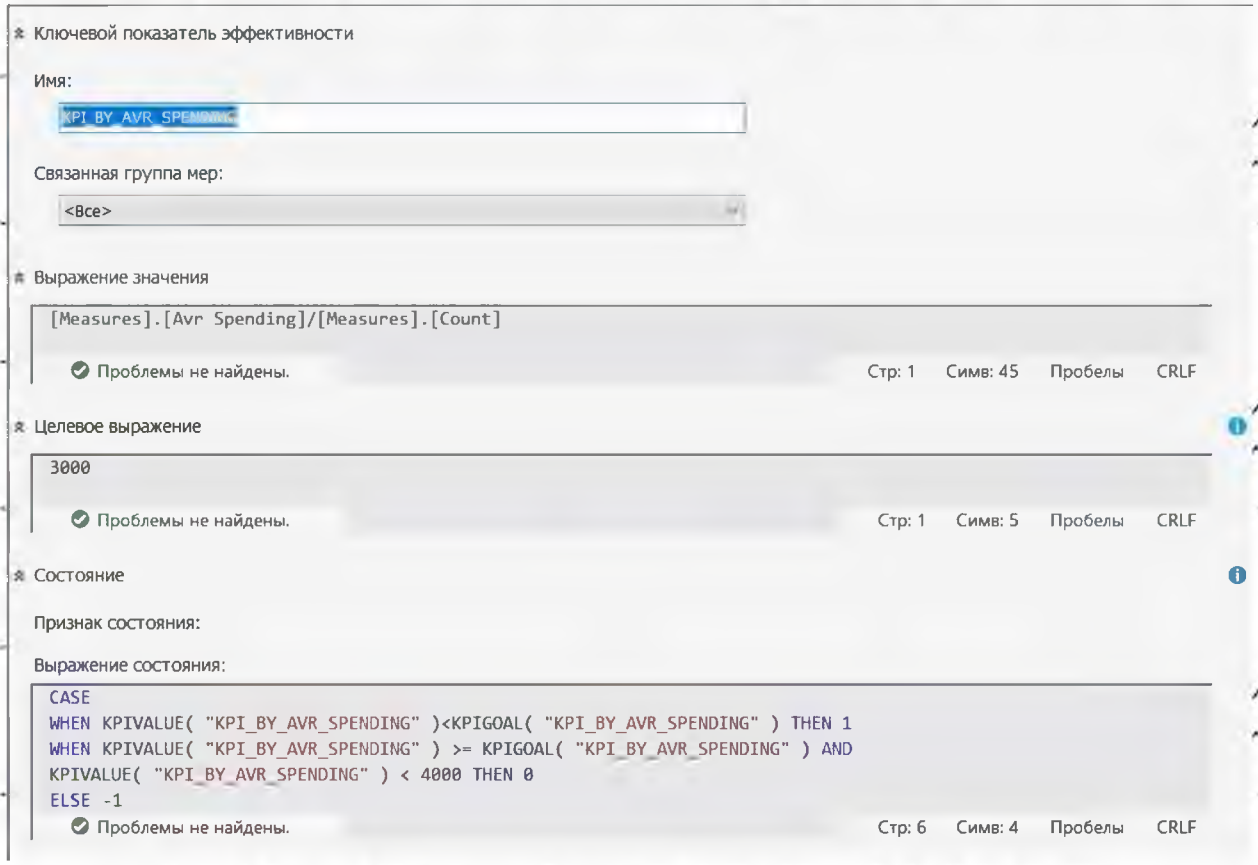


Рис. 28 Створення ключового показника KPI для витрат

Показник буде вираховуватися з середніх витрат та порівнюватися з цільовим показником, після чого поділятися на 3 категорії.

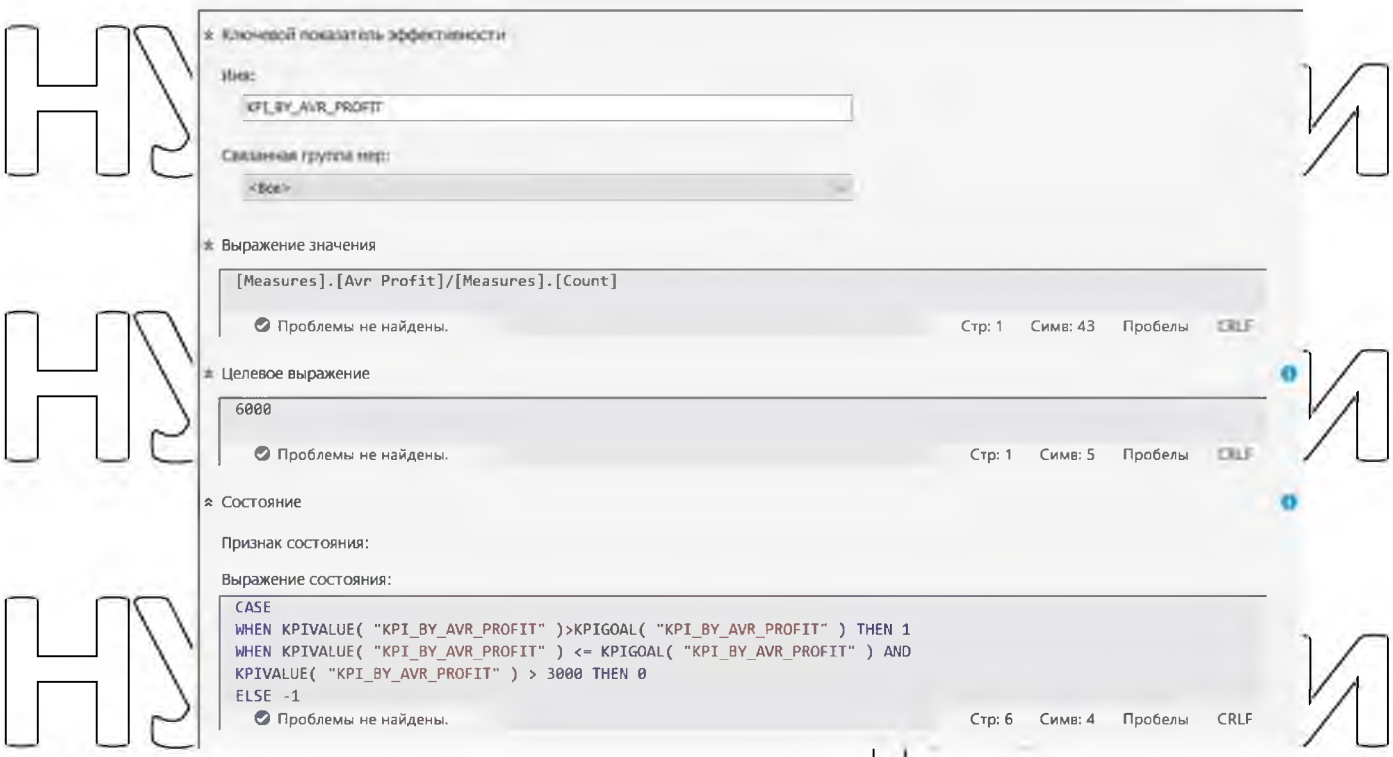


Рис. 29 Створення ключового показника KPI для прибутків

Показник буде вираховуватися з середнік прибутків та порівнюватися з цільовим показником, після чого поділятися на 3 категорії.

Результатом розрахунку KPI буде структура показана на рисунку 30.

Отобразить структуру	Значение	Цель	Состояние
KPI_BY_AVR_PROFIT	4149,03	6000	
KPI_BY_AVR_SPENDING	3822,56	3000	

Рис. 30 Результат розрахунку показників KPI

3.2.6.5 Реалізація алгоритму Decision Trees технологіями OLAP

Алгоритм дерева рішень належить до сімейства алгоритмів навчання з наглядом. На відміну від інших алгоритмів навчання з наглядом, алгоритм дерева рішень також можна використовувати для вирішення проблем регресії та класифікації.

Метою використання дерева рішень є створення навчальної моделі, яка може використовуватися для прогнозування класу або значення цільової

змінної шляхом вивчення простих правил прийняття рішень, виведених з попередніх даних (навчальних даних).

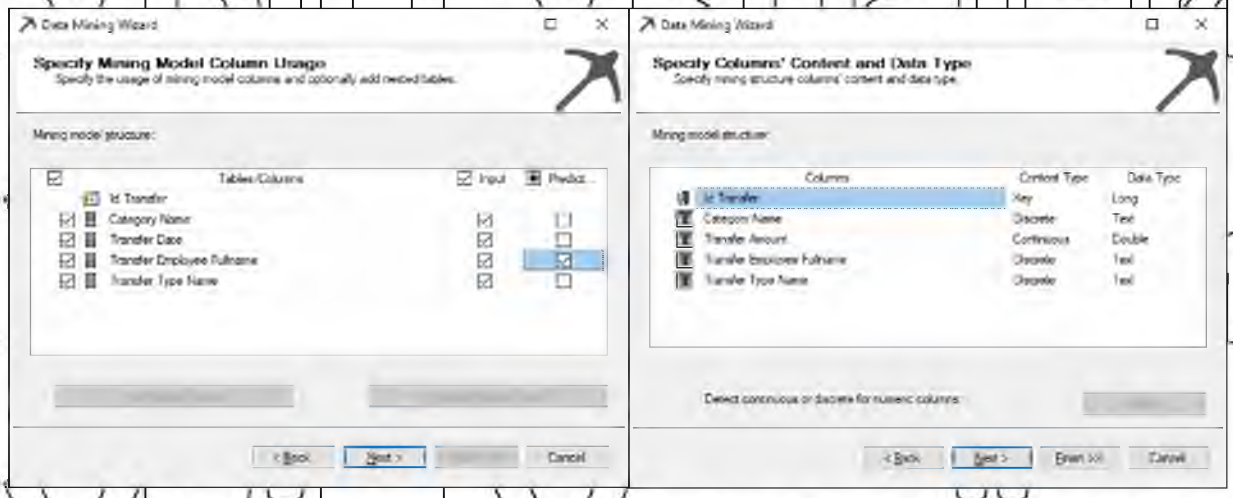


Рис. 31 Створення структури

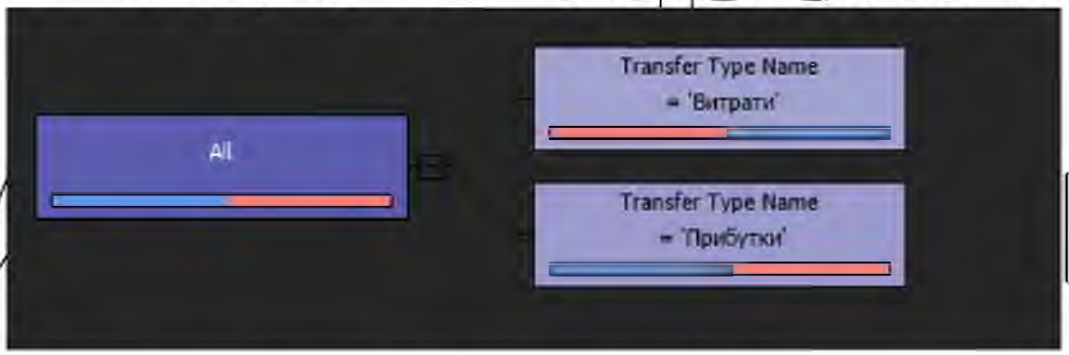


Рис. 32 Вигляд дерева рішень

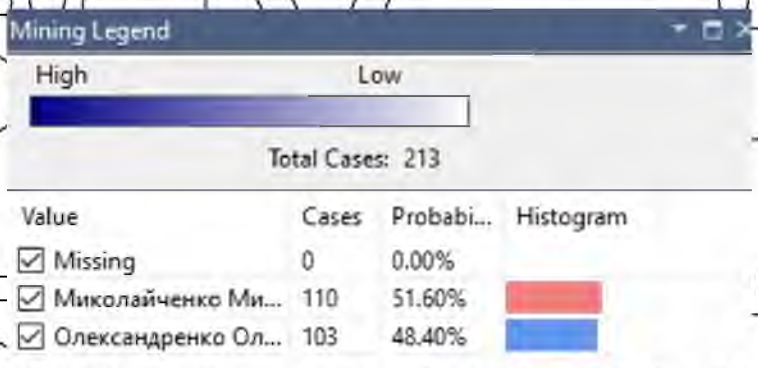


Рис. 33 Результати формування для витрат

На рисунку 33 можна побачити характеристики атрибутів для випадку коли тип транзакції «Витрати». В цьому вузлі представлено 213 випадків при тому що у базі представлено 450 записів, що означає ймовірність транзакції

потрапити в цей вузол дорівнює 47,3%. Також розраховані ймовірності для оформлення транзакцій конкретними користувачами. Так для транзакції бути оформленою першим користувачем ймовірність складає 51,6% а другим 48,4%.

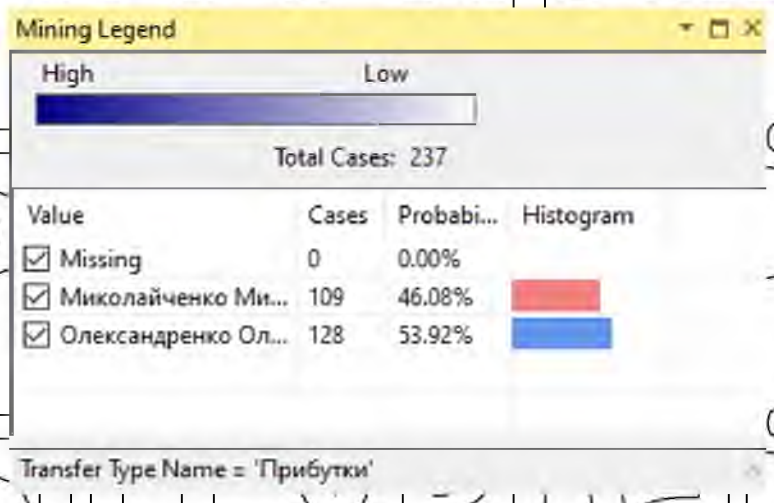


Рис 34 Результати формування для прибутків

На рисунку 34 можна побачити характеристики атрибутів для випадку коли тип транзакції «Витрати». В цьому вузлі представлено 237 випадків при тому що у базі представлено 450 записів, що означає ймовірність транзакції потрапити в цей вузол дорівнює 52,6%. Також розраховані ймовірності для оформлення транзакцій конкретними користувачами. Так для транзакції бути оформленою першим користувачем ймовірність складає 46,08% а другим 53,92%.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Тестування розробленої системи

4.1.1 Автоматичне тестування системи

Для перевірки ПЗ та його функцій, його необхідно тестувати. Тестування дозволяє розробникам побачити, чи коректно працюють всі модулі системи і взагалі проконтролювати працездатність всієї системи.

Тестування буде проводитись двома способами. Автоматичне тестування за допомогою Unit-тестів та мануальне тестування інтерфесу[9].

Для автоматичного тестування веб-серверу буде використаний вбудований у фреймворк Django функціонал, який надається модулем «test»[6].

Для запуску автоматичних тестів використовується команда «python manage.py test». Після запуску тестів створюється тестова база даних, для того щоб не втручатися в роботу основної БД. Після цього ПЗ збирає всі тести що є в проекті та по черзі запускає їх. В кінці тестування на екрані виводиться інформація про результати тестування. Приклад тестування показаний на рисунку 35.

```
# python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..........
----------------------------------------------------------------------
Ran 10 tests in 0.406s

OK
Destroying test database for alias 'default'...
```

Рис. 35 Автоматичне тестування

На лістингу 8 показано приклад програмного коду за допомогою якого тестуються функції системи.

Лістинг 8 – Тестування отримання списку транзакцій

```
class AccountingListViewTest(TestCase):
    def setUp(self):
        self.user_tg = AccountingUser.objects.create(username='test', password='test')

    def tearDown(self):
        self.user_tg.delete()

    def test_view_url_get_accounting_list_view(self):
        self.client.login(username='test', password='test')
        resp = self.client.get(reverse('accounting_list'))
        self.assertEqual(resp.status_code, 200)
```

У методі «setUp» створюється користувач до якого будуть відноситись транзакції. Після чого метод «test_view_url_get_accounting_list_view» авторизує користувача у системі та надсилає запит на отримання списку записів та перевіряє статус запиту.

4.1.2 Тестування користувацького інтерфейсу

Тестування користувацького інтерфейсу буде проводитись за допомогою Manual Testing. Таким чином можна перевірити чи зручним є використання системи та чи не буде виявлено помилок при користуванні. Перевірка інтерфейсу системи буде проведена зі сторони менеджера підприємства, якому доступні всі функції системи.

Першим розділом системи є розділ «Адміністрування», яким може користуватися тільки менеджер підприємства.

На рисунку 36 показано головну сторінку веб-сайту де відображено коротку звітну інформацію про стан підприємства. На ній ми можемо побачити такі дані:

- витрати та прибутки за місяць: 707 774 грн та 1 038 292 грн. Чистий прибуток за місяць вийшов 330 518 грн;
- витрати та прибутки за пів року: 5 277 017 грн та 8 226 750грн. Чистий прибуток за місяць вийшов 2 949 733 грн;
- витрати та прибутки за місяць: 10 154 670 грн та 17 434 261 грн. Чистий прибуток за місяць вийшов 7 279 591 грн;
- витрати та прибутки за місяць: 21 975 316 грн та 38 316 700 грн.

Чистий прибуток за місяць вийшов 16 341 384 грн.

Рис. 36 Головна сторінка системи

На рисунку 37 показано сторінку додавання нових користувачів підприємства та список існуючих.

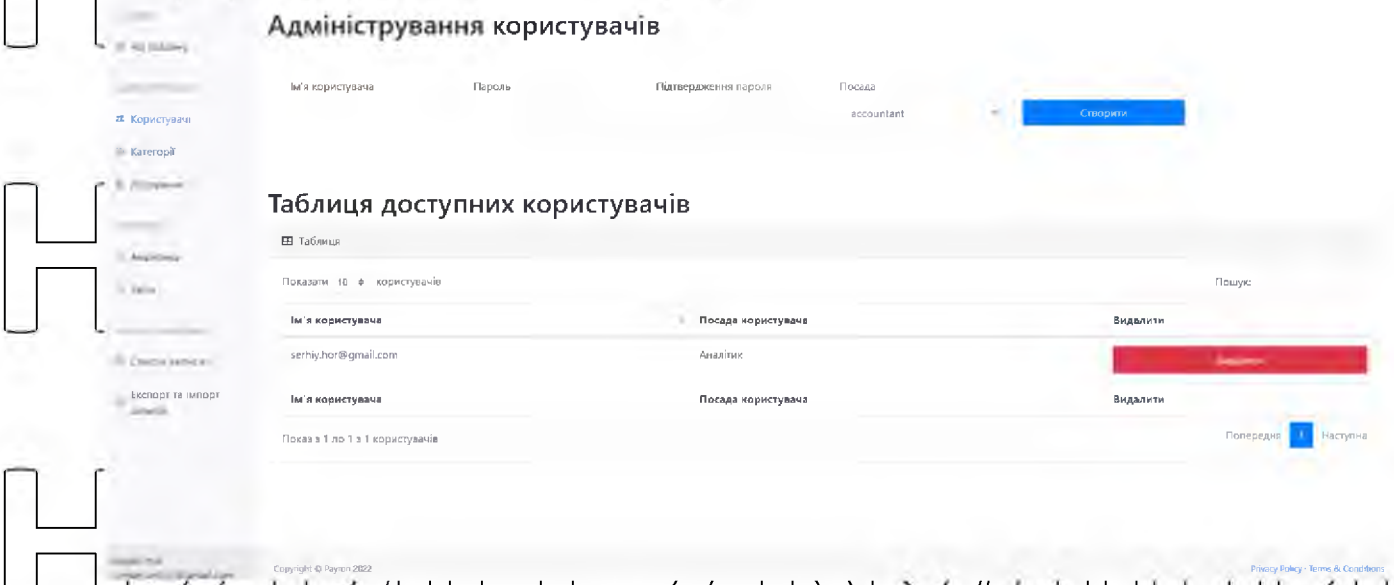


Рис. 37 Сторінка користувачів підприємства

На рисунку 38 показано сторінку додавання категорій перекказів та список існуючих.

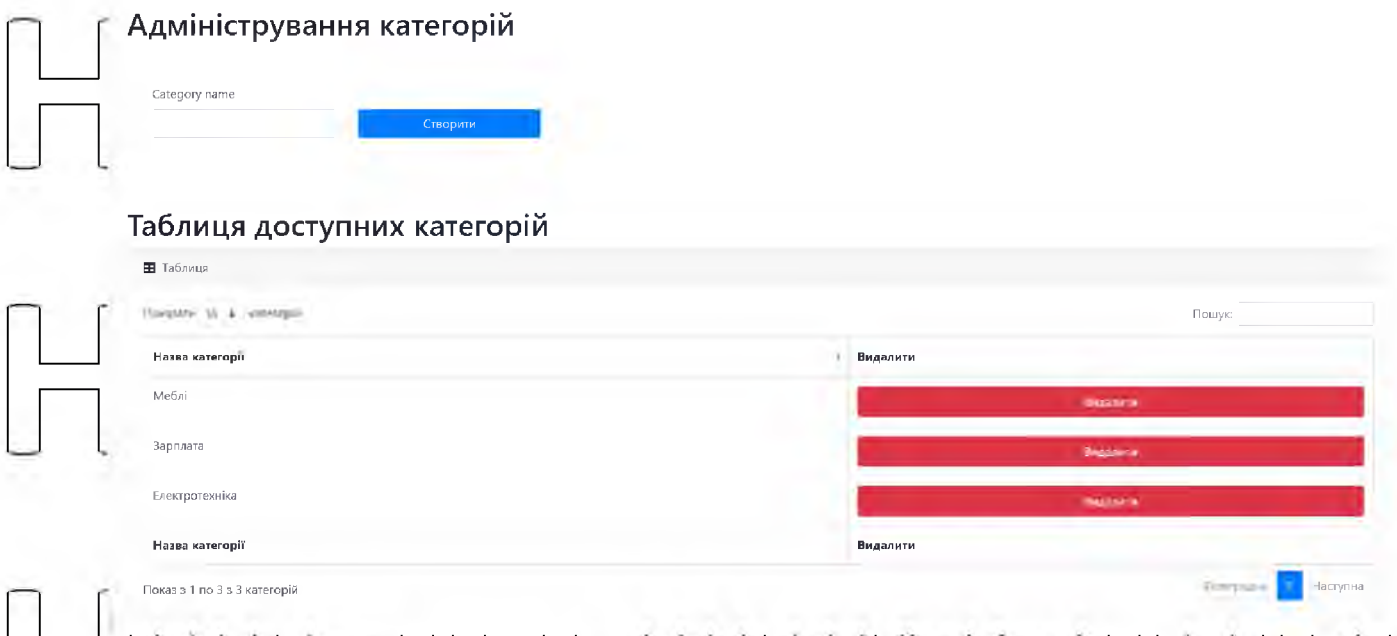


Рис. 38 Сторінка категорій підприємства

На рисунку 39 показано сторінку логування де показані дії користувачів підприємства.

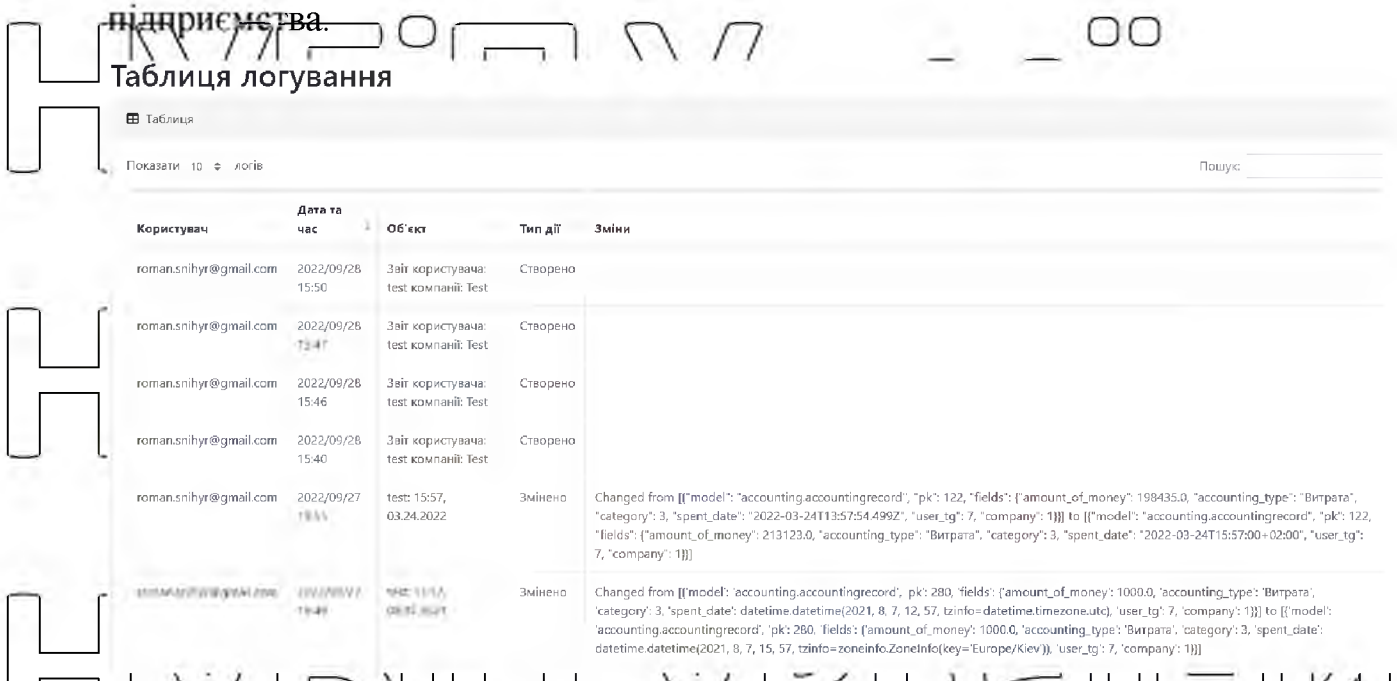


Рис. 39 Сторінка журналу дій користувачів

Наступним модулем є розділ «Аналітика» де присутні 2 підрозділи.

На рисунку 40 та 41 показана сторінка де за допомогою графіків зображено аналітику даних підприємства. Наприклад на графіку витрат за останні два тижня можна побачити що максимум витрат був 16.10.22 та складав

66 637 грн, а в основному витрати не перевищували 40 000 грн. А зі сторони прибутків максимум за цей період був 139 653 грн, а прибутки в основному не перевищували 100 000 грн.

Рис. 40 Інтерактивні графіки частина 1

На графіках витрат та прибутків за останні 6 місяців можна побачити максимуми та мінімуми. Для витрат це становить 995 331 грн та 707 774 грн відповідно. Для прибутків це 1 600 442 грн та 1 038 292 грн.

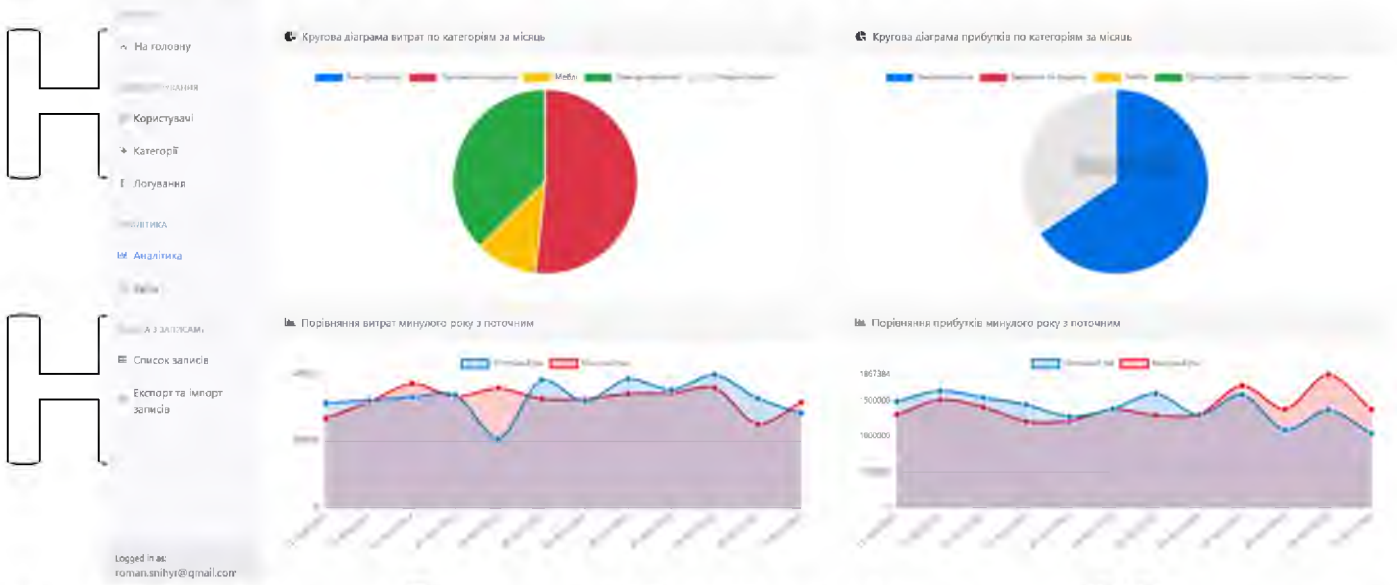


Рис. 41 Інтерактивні графіки частина 2

На рисунку 41 показано витрати та прибутки за місяць по категоріям а також порівняння прибутків та витрат з минулим роком. Так на круговій діаграмі витрат можна побачити що основний об'єм витрат йде на зарплати та податки працівників, що за останній місяць складало 365 855 грн. Наступною за об'ємом витрат йде категорія «Оренда території», яка складає 264 265 грн. На меблі підприємство витратило 77 654 грн. за останній місяць. Діаграма прибутків включає в себе дві категорії: Електротехніка – 680 300 грн та надані послуги – 357 992 грн. На графіках з порівняннями можна побачити що витрати тримаються приблизно однаково у порівнянні з минулим роком, а прибутки просідають під кінець року.

На рисунку 42 зображено сторінку формування звітів.

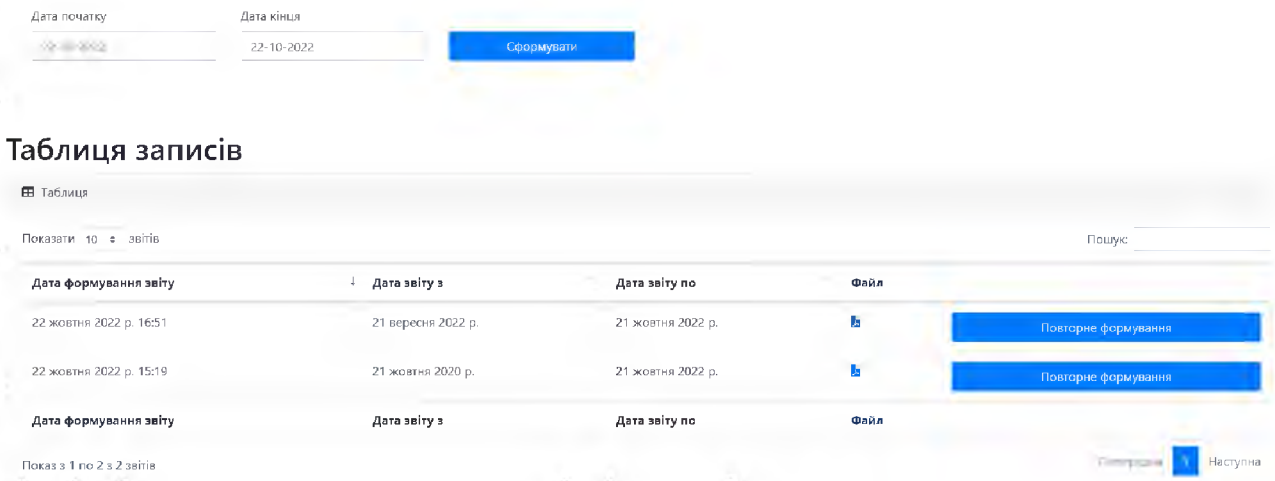


Рис. 42 Сторінка звітів

Останнім модулем є сторінки з додаванням записів.

На рисунку 43 зображено сторінку додавання записів та список існуючих.

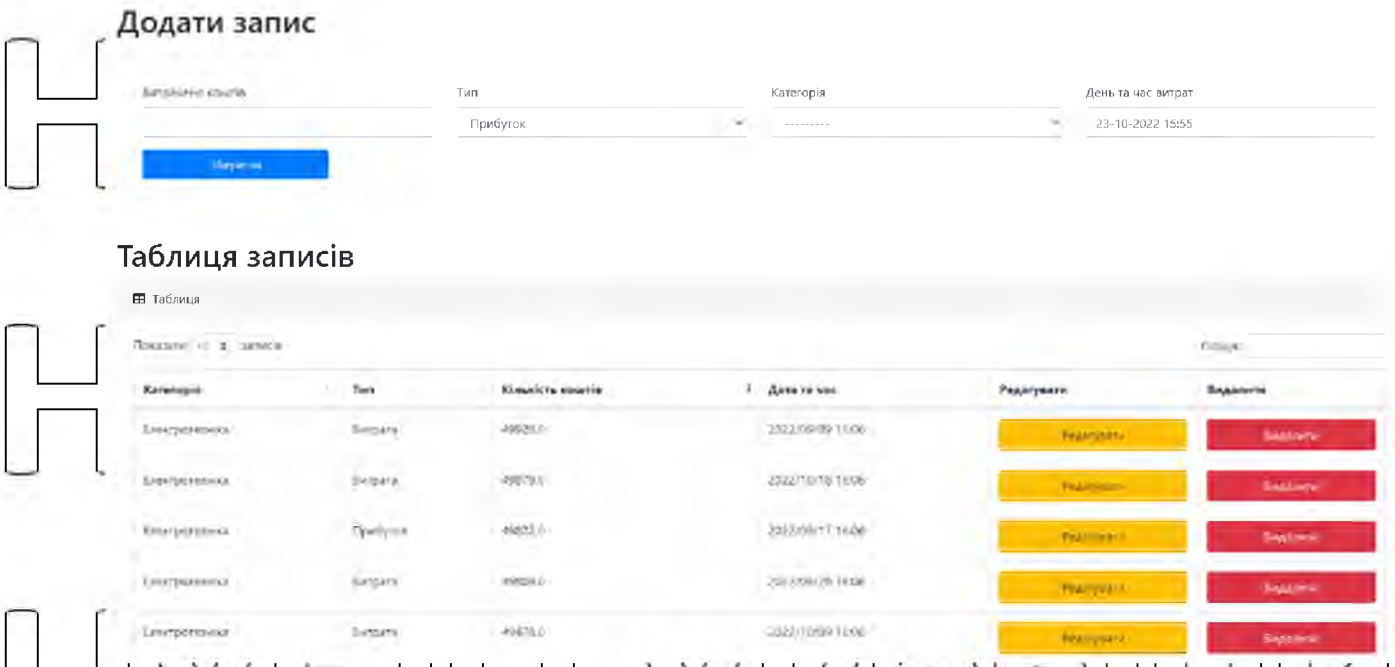


Рис. 43 Сторінка записів

На рисунку 44 зображено сторінку імпорту та експорту записів.

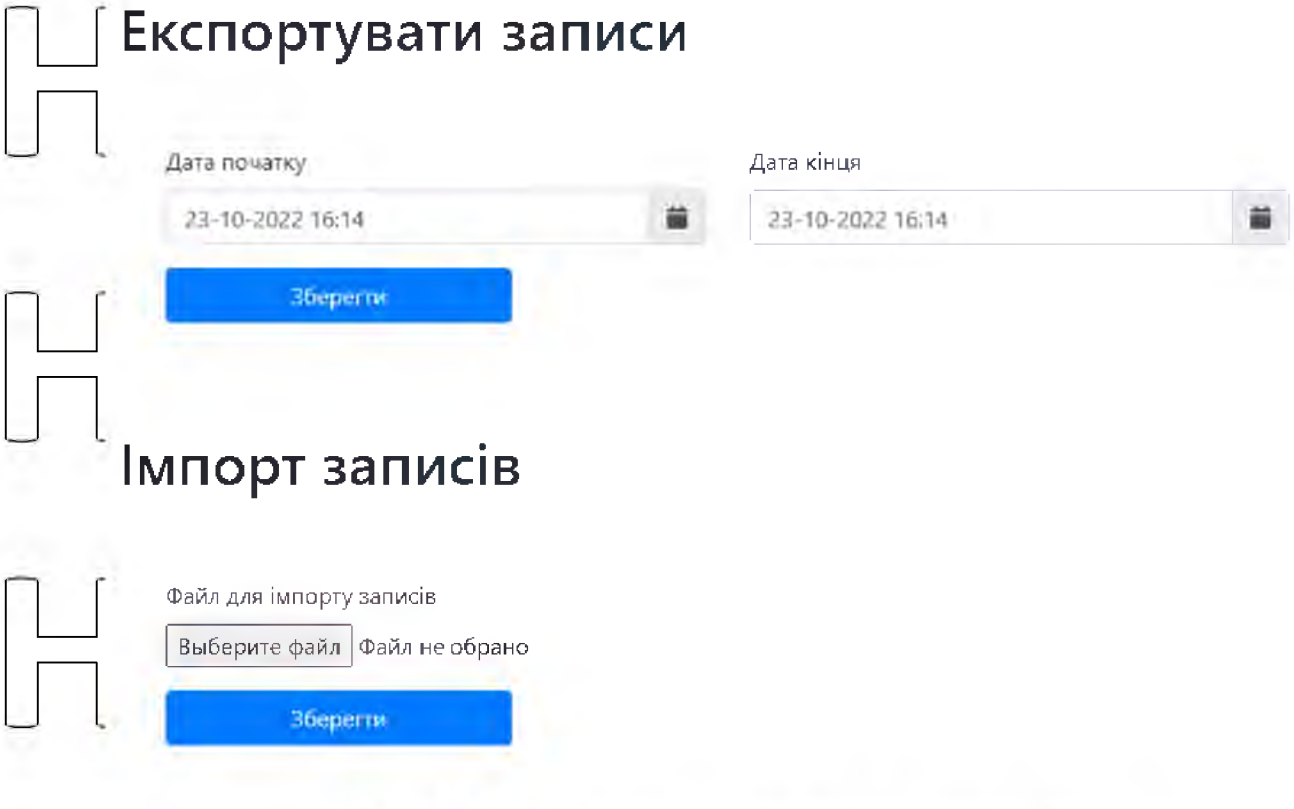


Рис. 44 Сторінка експорту та імпорту даних

Далі показано приклад сформованого звіту, де містяться основні дані підприємства в обраних часових рамках. Для формування звітності

використовуються кругові діаграми, гістаграми, лінійна регресія, дерева рішень та кластеризація. Також в кінці наведений вертикальний аналіз по категоріям, та статистичний аналіз витрат та прибутків.

На рисунку 45 показано коротка інформація по звіту. В заголовку написано хто формував звіт, та якому підприємству належить цей користувач. Далі йде інформація про часові рамки звіту. В даному випадку звіт має часові рамки від 21.09.2022 до 21.10.2022. Нижче йде коротка інформація про загальну суму витрат та прибутків та різниця між ними:

- Витрат – 821 162 грн;
- Прибутків – 1 261 230 грн;
- Різниця – 440 068 грн.

Нижче показані кругові діаграми витрат та прибутків. Для витрат є три категорії:

- Зарплата та подарки – 364 254 грн (44,4%);
- Оренда території – 351 017 грн (42,7%);
- Меблі – 105 891 грн (12,9%).

Для прибутків:

- Електротехніка – 850 398 грн (67,4%);
- Надані послуги – 410 832 грн (32,6%).

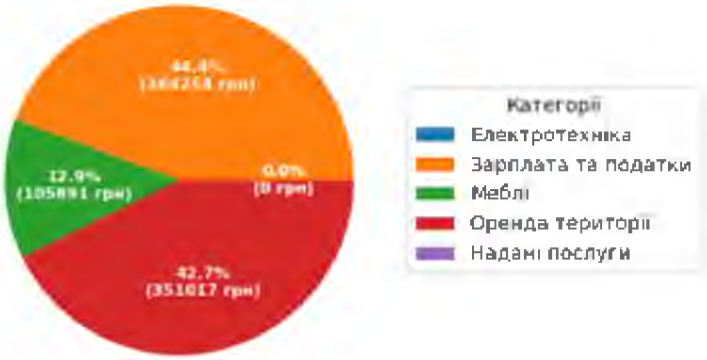
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

payren

Звіт для користувача "roman.snihyr@gmail.com" компанії "Kernel"
Часові рамки з 22.09.2022 по 21.10.2022

Всього витрат: 821162.0грн.
Всього прибутку: 1261230.0грн.
Різниця прибутку та витрат: 440068.0грн.

Кругова діаграма витрат



Кругова діаграма прибутку

Рис. 45 Результати сформованого аналітичного звіту частина 1

Як видно на гістограмі витрат на рисунку 46, можна дійти висновків що витрати вироловж цього місяця були стрибкоподібні. Тобто підприємство в цьому місяці витрачало кошти не кожного для.

НУБіП України
НУБіП України

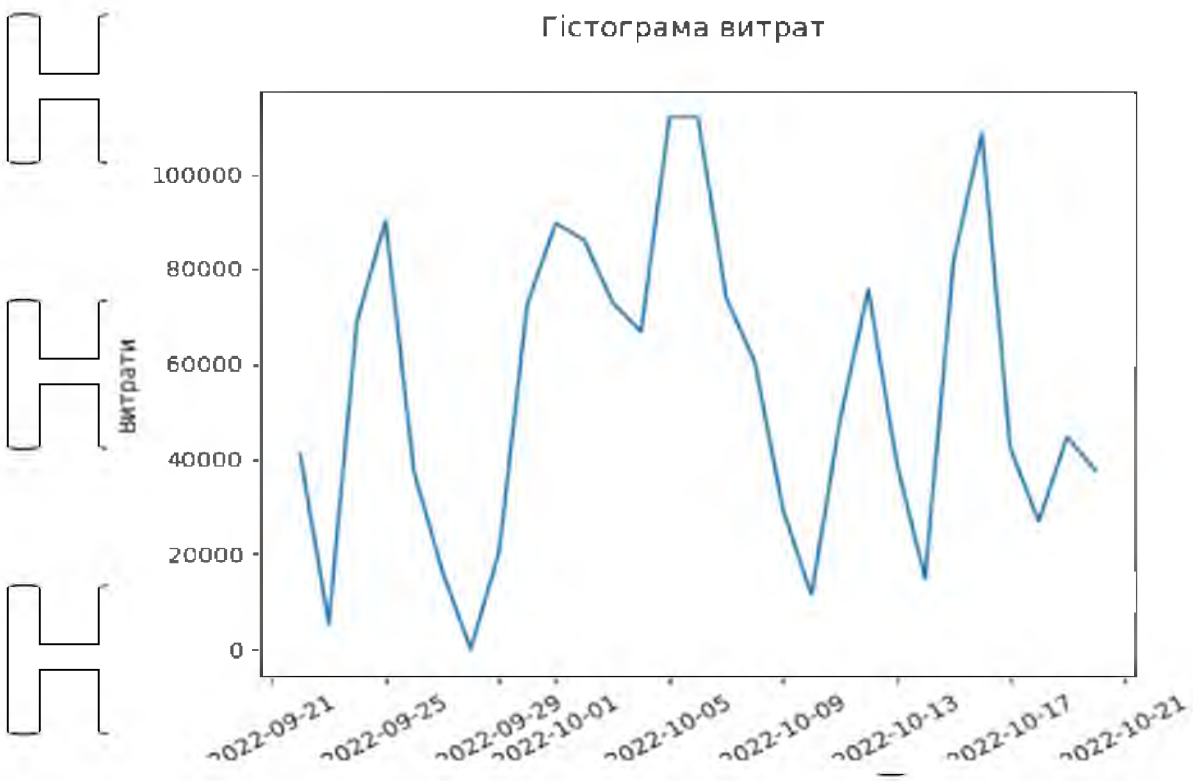


Рис. 46 Результати сформованого аналітичного звіту частина 2

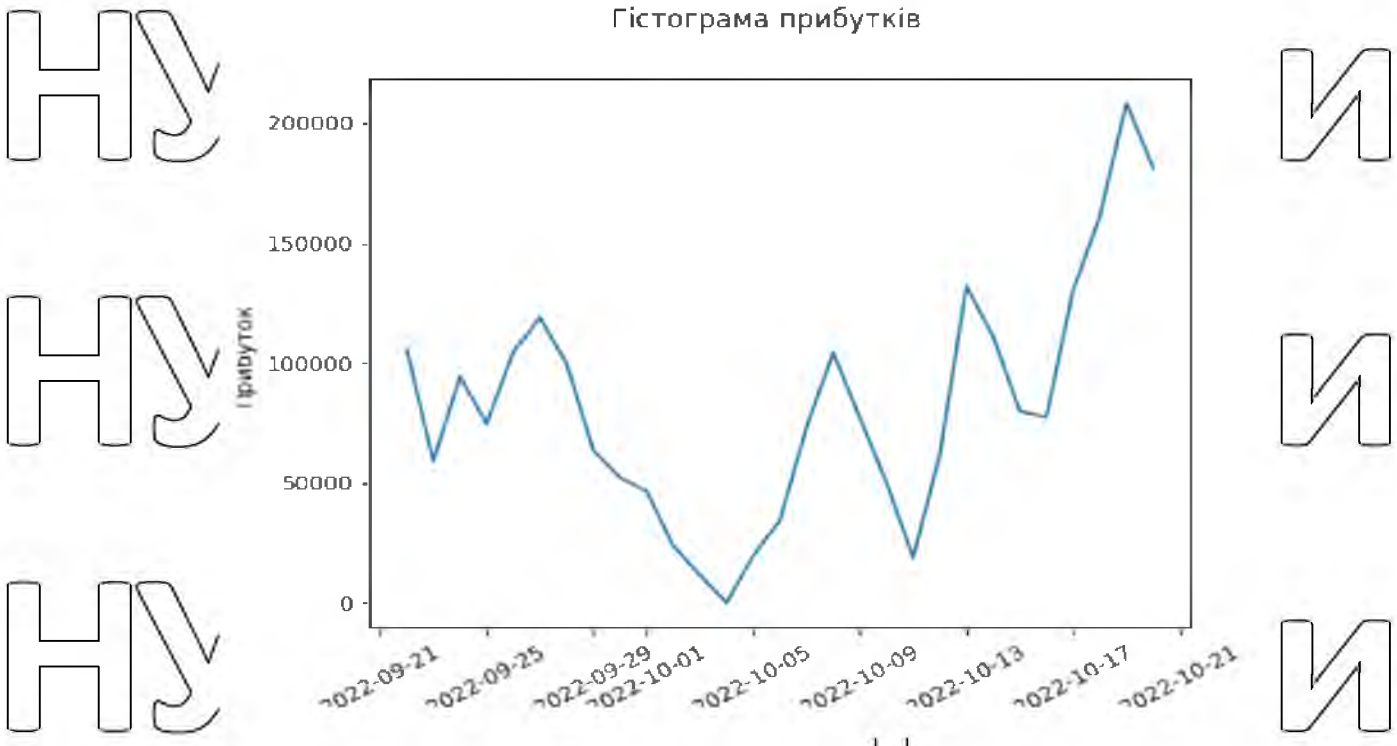
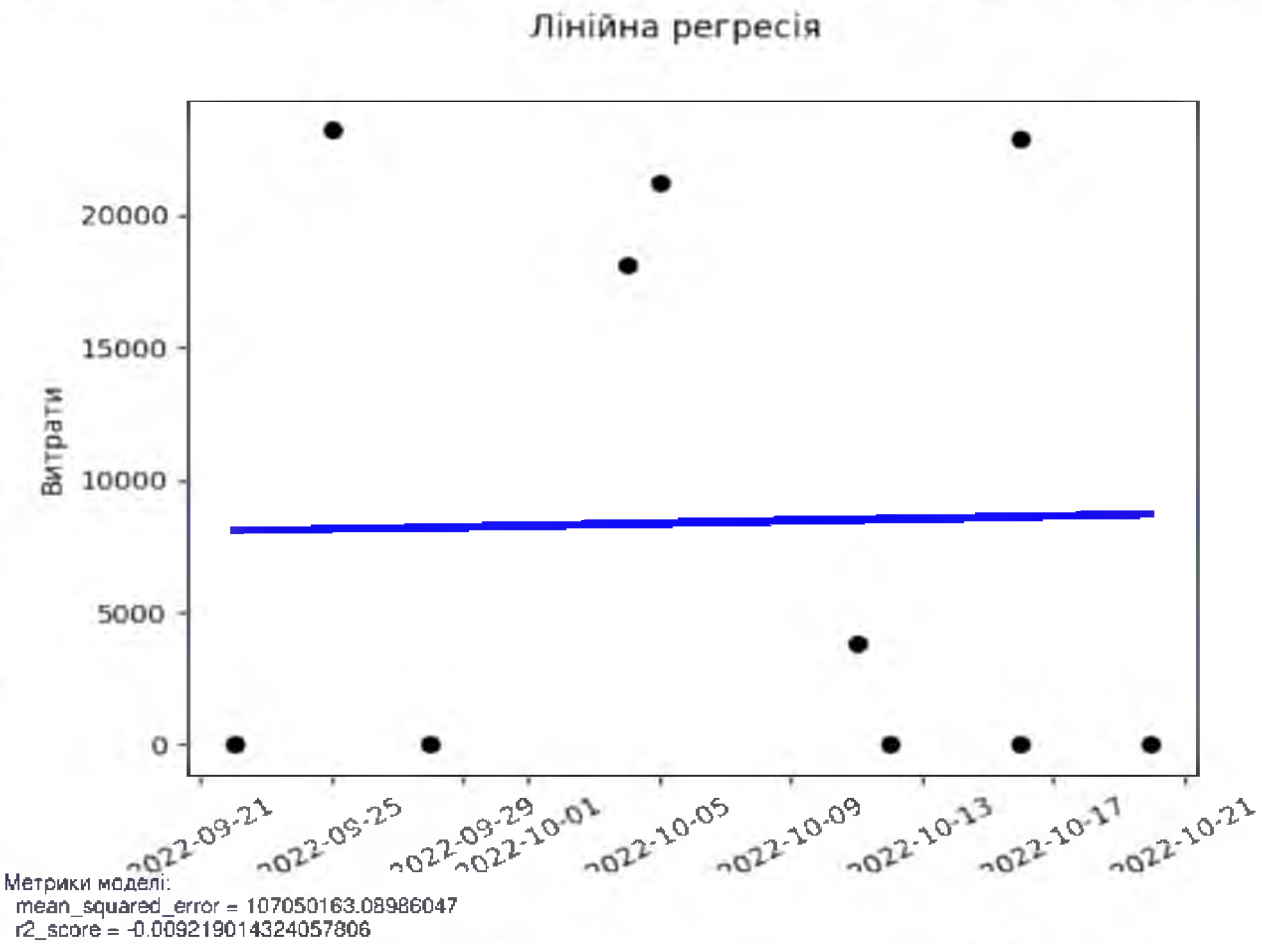


Рис. 47 Результати сформованого аналітичного звіту частина 3

На гістограмі прибутків також видно що не кожного дня підприємство отримувало прибутки, та можна побачити що під кінець прибутки почали збільшуватись.

На рисунку 48 показано лінійну регресію витрат, яку було натреновано на даних витрат за часовий проміжок. По ній можна побачити що витрати збільшуються, але не дуже чутливо.



Метрики моделі:
mean_squared_error = 107050163.08986047
r2_score = -0.009219014324057806

Рис. 48 Результати сформованого аналітичного звіту частина 4

На рисунку 49 показано лінійну регресію для прибутків, по ній можна побачити, що прибутки в цілому зростають до кінця часового проміжну.

Також ці моделі мають свої метрики моделі, по яких можна оцінювати чи правильно вони були натреновані, та наскільки можна довіряти цим даним.

НУБіП України

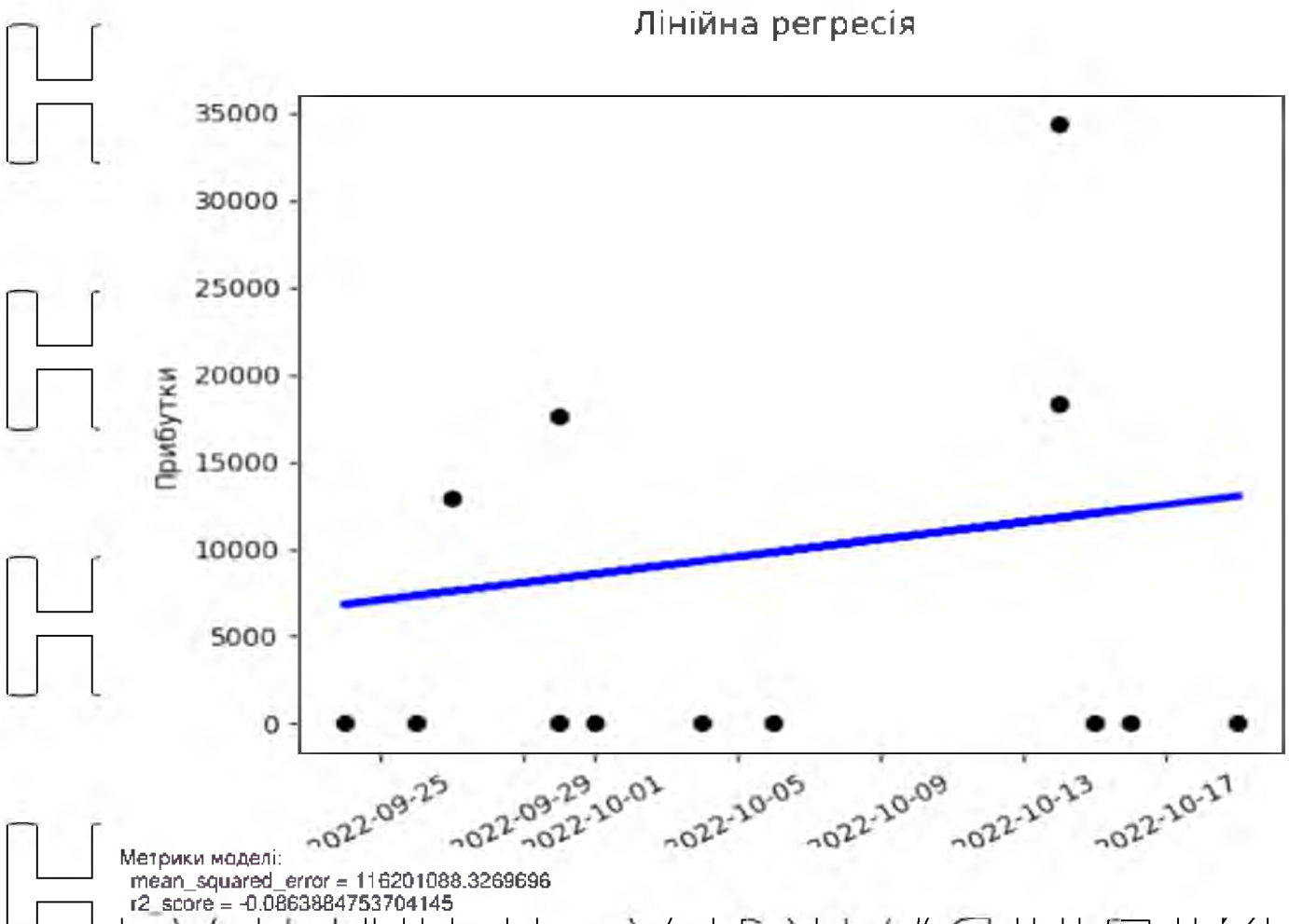


Рис. 49 Результати сформованого аналітичного звіту частина 5

На рисунках 50 та 51 показані представлення натренованих на відібраних даних моделі дерева рішень для витрат і прибутків. На вузлах дерев рішень показано характеристики за якими вони розподіляються. Наприклад дерево рішень витрат показує що категорія «Меблі» включає в себе всі витрати які менші ніж 9 402.5 грн. Категорія «Зарплата та податки» включає в себе всі витрати, що більше ніж 9 402,5 грн але менше ніж 19 811 грн. Всі інші витрати будуть відноситися до категорії «Оренда території». Для прибутків дерево рішень виглядає простіше, адже в даному випадку є тільки 2 категорії прибутків, тож всі прибутки, що менше 20 729,5 грн відносяться до категорії «Надані послуги», а всі інші до категорії «Електротехніка».

НУБіП України

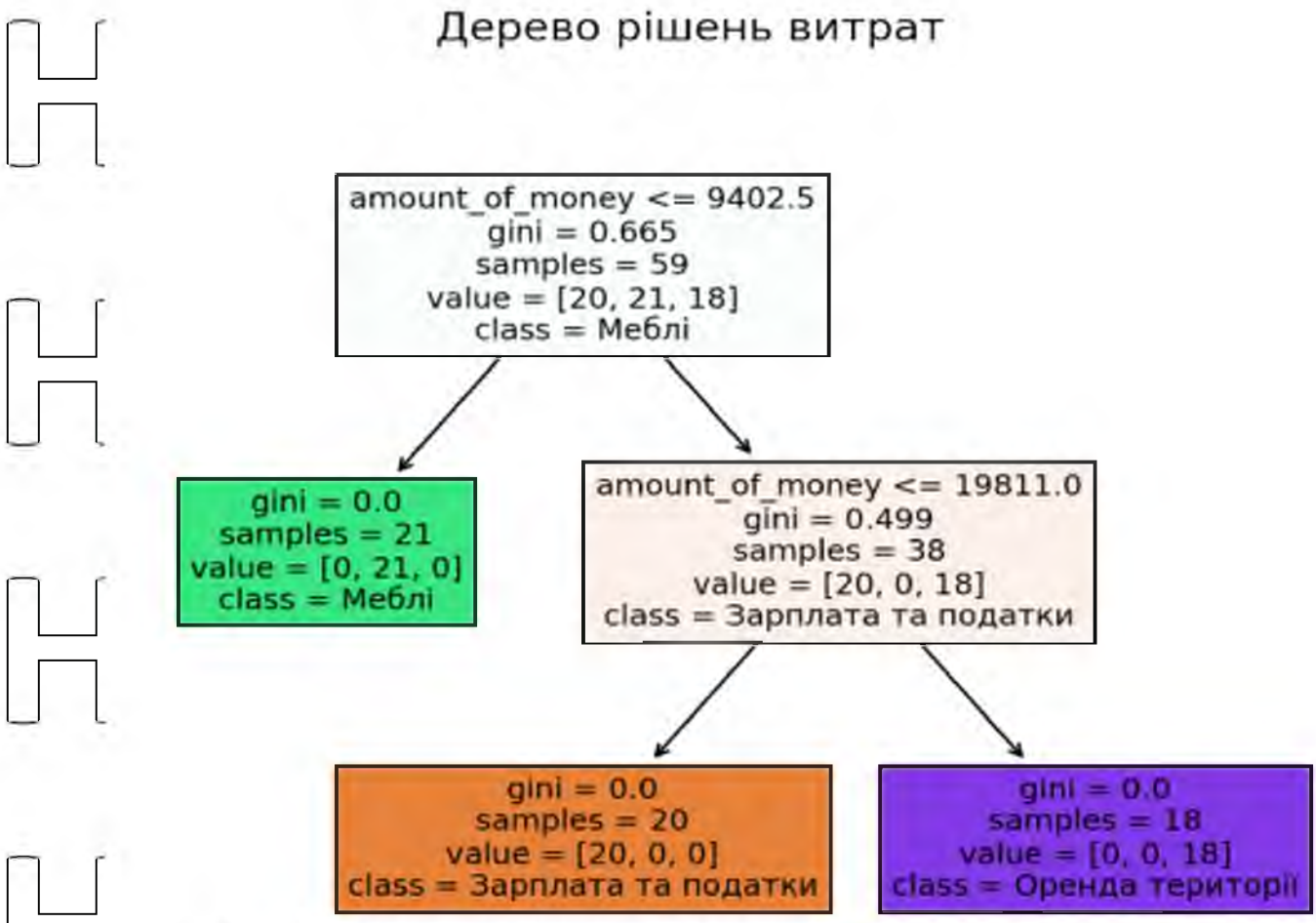


Рис. 50 Результати сформованого аналітичного звіту частина 6

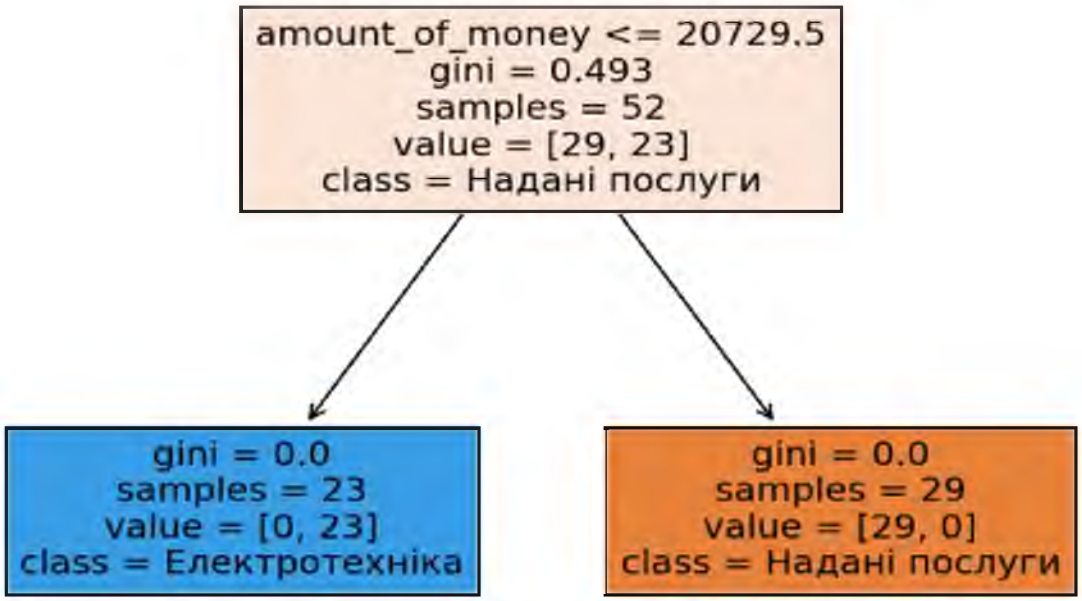


Рис. 51 Результати сформованого аналітичного звіту частина 7

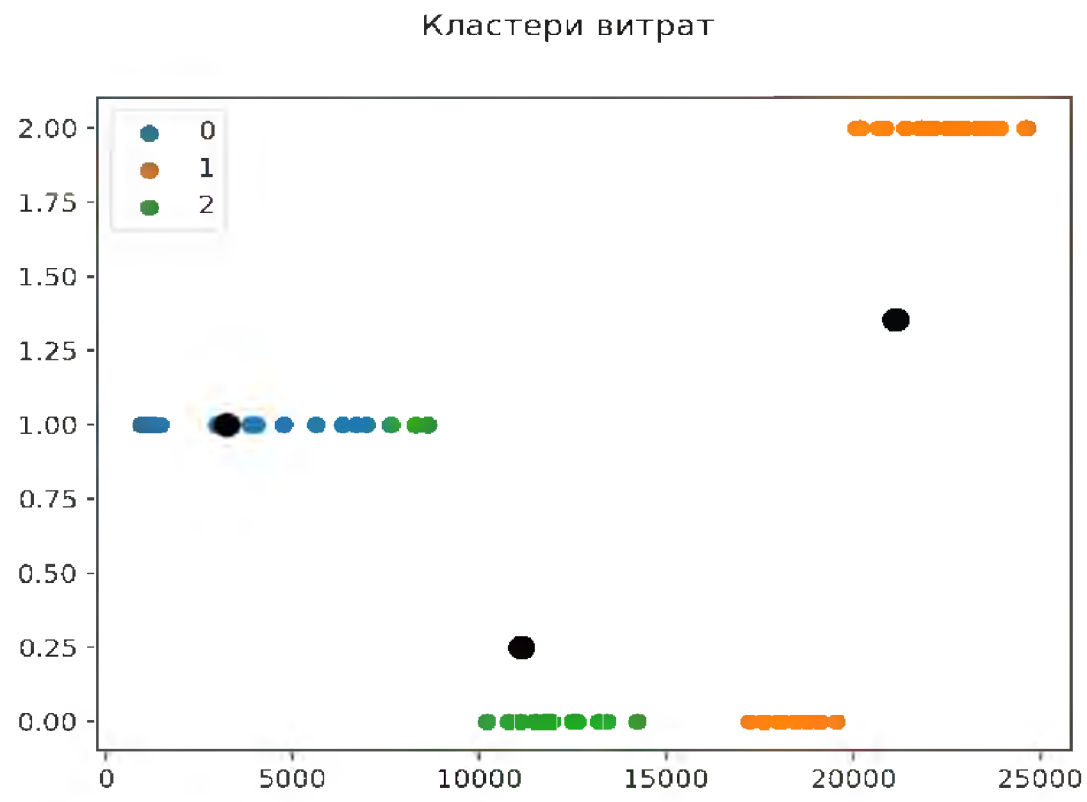


Рис. 52 Результати сформованого аналітичного звіту частина 8

Вертикальний аналіз

Категорія	Сума витрат	Сума прибутків	Відсоток витрат	Відсоток прибутків
Електротехніка	0	1139463.0	0.0 %	76.0 %
Зарплата та податки	328307.0	0	37.0 %	0.0 %
Меблі	94914.0	0	11.0 %	0.0 %
Оренда території	471424.0	0	53.0 %	0.0 %
Надані послуги	0	359737.0	0.0 %	24.0 %
Загалом	894645.0	1499200.0	100.0 %	100.0 %
Категорія	Сума витрат	Сума прибутків	Відсоток витрат	Відсоток прибутків

Статистичний аналіз витрат

Категорія	Середнє	Медіана	Ст. відхилення	Дисперсія
Зарплата та податки	14923.05	13810.5	3340.02	11155728.71
Меблі	4126.7	3961.0	2737.12	7491801.68
Оренда території	22448.76	22527.0	1509.86	2279663.49
Категорія	Середнє	Медіана	Ст. відхилення	Дисперсія

Статистичний аналіз прибутків

Категорія	Середнє	Медіана	Ст. відхилення	Дисперсія
Надані послуги	13836.04	13533.0	3144.5	9887869.96
Електротехніка	35608.22	36400.0	7583.69	57512318.76
Категорія	Середнє	Медіана	Ст. відхилення	Дисперсія

Рис. 53 Результати сформованого аналітичного звіту частина 9

На рисунку 53 можна побачити різні статистичні показники та вертикальний аналіз для витрат та прибутків. Статистичний аналіз дає можливість дізнатися середнє значення, медіану, стандартне відхилення та дисперсію. А у вертикальному аналізі у відсотках показано відсоток та суму яку займає категорія у кожному типі переказів.

НУБіП України

4.2 Вимоги до апаратного та програмного забезпечення

Апаратна складова грає важливу роль у функціонуванні всієї системи, адже від неї залежить працездатність системи. Також важливу роль відіграє операційна система, на якій буде запускатися програмне забезпечення, і в даному випадку різниця в операційній системі невелика, адже комп'ютерна система буде розгортатися за допомогою інструментарію для роботи з ізольованими контейнерами Docker, який дозволяє розгортання на будь-якій операційній системі, де він встановлений.

При цьому є декілька способів розгортання системи. Можна використати фізичний сервер, та використовувати його, або ж скористатися хмарними технологіями та орендувати сервер. Останній варіант є найбільш популярним і вигідним, адже користувачу залишається тільки завантажити свою систему на віддалений сервер, розгорнути її там і налаштувати для роботи. Багато сервісів з оренди віддалених серверів мають додаткові послуги з їх налаштування, підтримки та адміністрування. Також такий варіант вигідний тим, що багато сервісів дозволяють розширювати характеристики свого сервера простим переходом на інший тариф, що робить вдосконалення системи дуже зручним і простим.

У будь-якому з цих випадків апаратна складова повинна відповідати наступним мінімальним вимогам:

- процесор: 2-4 ядра сімейства процесорів серверного типу;
- оперативна пам'ять: 8 гігабайт;
- жорсткий диск: 500 гігабайт.

Якщо кількість користувачів буде швидко збільшуватись, можна також орендувати файловий сервер для збереження звітності.

Система повинна мати стабільне та постійне підключення до мережі Інтернет, щоб користувачі мали доступ до неї цілодобово.

Враховуючи, що компоненти системи будуть розгорнуті як окремі мікросервіси, вони будуть використовувати різні хост-порти для зв'язку між собою, тому комп'ютерна система повинна забезпечувати доступ до портів для взаємозв'язку між сервісними контейнерами.

Користувач повинен мати можливість використовувати систему на будь-якому пристрої, який має доступ до мережі Інтернет і на якому встановлено браузер. Це можуть бути такі пристрої, як мобільні телефони, планшети, ноутбуки, персональні комп'ютери тощо.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

ВИСНОВКИ

Під час виконання магістерської роботи була спроектована та реалізована система що допомагає проводити аналіз фінансової діяльності підприємства.

В ході виконання роботи було описано предметну область, проаналізовані аналогічні, або ж схожі системи, які допомагають підприємствам аналізувати свій фінансовий стан, визначено основні вимоги до систему по функціям, які вона повинна виконувати.

Наступним кроком виконання магістерської роботи є моделювання системи використовуючи UML діаграми. Для цього було побудовано і описано наступні діаграми:

- діаграма прецедентів;
- діаграма класів;
- діаграма послідовності;
- діаграма розгортання;
- діаграма діяльності;
- діаграма кооперації.

Після моделювання системи, наступним кроком йде розробка програмного забезпечення. Як основну архітектуру системи було обрано мікросервісну архітектуру, а також описано переваги цієї архітектури в даному випадку. Для розробки програмного забезпечення було використано такі технології: Python. Django. Django Rest Framework, Docker, JavaScript, HTML, CSS. Також для додаткового дослідження було використано OLAP технології, які також мають алгоритми, які допомагають аналізувати дані та будувати звітності.

Останнім кроком є тестування розробленого програмного забезпечення для того, щоб дізнатися чи виконує воно всі основні функції та чи відповідає всім поставленим вимогам. Для тестування ПЗ використовувалися два методи: автоматичне тестування за допомогою unit-тестів та ручне тестування користувацького інтерфейсу.

НУБіП України

Розроблене програмне забезпечення допомагає автоматично створювати аналітичну звітність фінансового стану підприємства та значно полегшує облік фінансів підприємства.

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Distributed Application Architecture". 2009. [Електронний ресурс] URL: https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf

2. Celery documentation. [Електронний ресурс] URL: https://docs.celeryproject.org/en/master/index.html

3. Communication Diagram [Електронний ресурс] URL: https://www.edrawmax.com/article/communication-diagram-uml.html

4. Django documentation. [Електронний ресурс] URL: https://docs.djangoproject.com/en/3.0/

5. Django Rest Framework. [Електронний ресурс] URL: https://www.django-rest-framework.org/

6. Evans Ehiorobo. How To Add Unit Testing to Your Django Project. 2020. [Електронний ресурс] URL: https://www.digitalocean.com/community/tutorials/how-to-add-unit-testing-to-your-django-project

7. Explore the UML sequence diagram [Електронний ресурс] URL: https://developer.ibm.com/articles/the-sequence-diagram/

8. G. Everest. BASIC DATA STRUCTURE MODELS EXPLAINED WITH A COMMON EXAMPLE. 1976. 39-46 ст.

9. IEEE Standards Board. IEEE Standard for Software Unit Testing: An American National Standard, ANSI/IEEE Std 1008-1987. 1999. 8 ст.

10. JQuery documentation. [Електронний ресурс] URL: https://jquery.com/

11. JavaScript [Електронний ресурс] URL: https://developer.mozilla.org/ru/docs/Web/JavaScript

12. Mathematical statistics functions [Електронний ресурс] URL: https://docs.python.org/3/library/statistics.html

НУБіП України

13. Matplotlib: Visualization with Python [Електронний ресурс] URL: https://matplotlib.org/

14. NumPy [Електронний ресурс] URL: https://numpy.org/

15. Online analytical processing (OLAP) [Електронний ресурс] URL: https://learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/online-analytical-processing

16. Pandas [Електронний ресурс] URL: https://pandas.pydata.org/

17. Postgresql. [Електронний ресурс] URL: https://www.postgresql.org/about/

18. Python documentation. [Електронний ресурс] URL: https://docs.python.org/3/

19. RabbitMQ documentation. [Електронний ресурс] URL: https://www.rabbitmq.com/documentation.html

20. Scikit-learn [Електронний ресурс] URL: https://scikit-learn.org/stable/

21. The analysis of enterprise's financial statement according to the balance data [Електронний ресурс] URL: https://www.academia.edu/7545249/The_analysis_of_enterprise_s_financial_statement_according_to_the_balance_data

22. Trung Anh Dang. Software Architecture: The Most Important Architectural Patterns You Need to Know. 2020. [Електронний ресурс] URL: https://levelup.gitconnected.com/software-architecture-the-important-architectural-patterns-you-need-to-know-a1f5ea7e4e3d

23. UML Class Diagram [Електронний ресурс] URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

24. UNIFIED MODELING LANGUAGE. 2005. [Електронний ресурс] URL: https://www.uml.org/what-is-uml.htm

25. Use case diagram [Електронний ресурс] URL: https://www.techtarget.com/whatis/definition/use-case-diagram

26. What is Docker?. [Електронний ресурс] URL: https://aws.amazon.com/docker/?nc1=h_ls

27. Wiegers, Karl E. Software Requirements. Redmond, WA: Microsoft Press. 2003.

28. Основи фінансового аналізу за даними спрощеної звітності [Електронний ресурс] URL: https://magazine.faaf.org.ua/osnovi-finansovogo-analizu-za-danimi-sproschenoi-zvitnosti.html

29. Парус-бухгалтерія [Електронний ресурс] URL: http://www.parus.ua/ru/161/

30. Управління торговим підприємством [Електронний ресурс] URL: http://1c.ua/v8/RegionalSolutions_UA_UTP.php

НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України
НУБіП України