НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

004.9:658.1

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
_____Болбот І. М., д. т. н., проф._____	_____Голуб Б.Л., к.т.н., доцент_____
_____	_____
____ _____ **2024 р.**	____ _____ **2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Аналітична система управління фінансами малого підприємства»

Спеціальність _____121 «Інженерія програмного забезпечення»_____
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____д.т.н., проф._____	_____	____Семко В. В.____
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник магістерської кваліфікаційної роботи

_____д.е.н., проф._____	_____	__Руденський Р. А.__
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав _____ ____Возний О. І.___
(підпис) (ПІБ студента)

КИЇВ – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

доцент к.т.н. Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
20__ р.

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

Возний Олександр Ігорович
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія програмного забезпечення»
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи «Аналітична система управління фінансами малого підприємства»

затверджена наказом ректора НУБіП України від " 09 " жовтня 2024 р. № 1505 «С»
Термін подання завершеної роботи на кафедру 29.11.2024
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: наукові статті з дослідження проблематики проведення та розвитку малої підприємницької діяльності: методів аналізу фінансового становища підприємства: методів прогнозування фінансових результатів бізнесу: матеріали із описом розробки прикладних програм

Перелік питань, що підлягають дослідженню:

1. Проведення системного аналізу об'єкта;
2. Аналіз існуючих рішень;
3. Формулювання вимог та структуру для аналітичної системи управління фінансами малого підприємства;
4. Розробка структури інформаційного забезпечення системи;
5. Розробка алгоритму аналізу даних, та самої аналітичної системи.

Перелік графічного матеріалу (за потреби)

Дата видачі завдання

Керівник магістерської кваліфікаційної роботи Руденський Р.А.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання Возний О. І.

(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DDD – (Domain-driven design) – підхід до проектування програмного забезпечення, який фокусується на моделюванні програмного забезпечення відповідно до вхідних даних, отриманих від експертів у цій галузі

FastAPI – сучасний веб-фреймворк для побудови RESTful API на Python

MVC – (model–view–controller) – патерн проектування програмного забезпечення

ORM – (Object Relational Mapping) – техніка, яка використовується для створення "мосту" між об'єктно-орієнтованими програмами і реляційними базами даних

PDF – (Portable Document Format) – формат файлів

PostgreSQL – безкоштовна реляційна база даних

Python – мова програмування високого рівня

REST API – архітектурний стиль програмного забезпечення

SQL – (Structured Query Language) – специфічна мова програмування, призначена для управління даними

UML – (Unified Modeling Language) – мова моделювання загального призначення для візуалізації проектування системи

БД – база даних

ВВП – валовий внутрішній продукт

СКБД – система управління базами даних

ФОП – (Фізична особа – підприємець) – згідно з українським законодавством, фізична особа, котра реалізує свою здатність до праці шляхом самостійної діяльності з метою отримання прибутку

ВСТУП

Бізнес відіграє важливу роль в економіці. Основні цілі бізнесу включають забезпечення прибутку, створення робочих місць, задоволення потреб споживачів і сприяння економічному зростанню. Він генерує прибуток і сприяє економічному зростанню країни.

Актуальність такої розробки зумовлена стрімким розвитком малого бізнесу в Україні за тенденцією розвинених держав. Це накладається на сучасні умови бізнес-середовища, які характеризуються високою динамічністю та нестабільністю. Малий бізнес, котрий часто працює з обмеженими фінансовими ресурсами, особливо вразливий до економічних коливань, змін у законодавстві, зростання податкових та кредитних зобов'язань.

Об’єкт: система моніторингу та контролю контрактів та платіжної дисципліни

Предмет: методи та механізми управління фінансовими ризиками в контексті забезпечення платоспроможності малого підприємства

Мета: розробка аналітичної системи для фінансового прогнозування малого підприємства

Методи дослідження. У цьому дослідженні використано комплексний підхід для створення та оцінки аналітичної системи управління фінансами малого підприємства. Було проведено огляд літератури для аналізу сучасних методів і підходів до управління фінансовими ризиками та оцінки платоспроможності. Системний аналіз використовувався для вивчення бізнес-процесів малого підприємства, зокрема моніторингу платіжної дисципліни та контролю контрактів. Для проектування системи був застосований метод моделювання UML для створення діаграм класів та послідовностей.

Наукова новизна цього дослідження полягає у створенні спеціалізованої аналітичної системи, яка дозволяє малим підприємствам автоматизувати моніторинг фінансових потоків, прогнозувати ризики та забезпечувати

платоспроможність. Особливість підходу полягає в інтеграції механізмів оцінки фінансових ризиків та обробки даних у реальному часі, що дає змогу ефективно управляти дебіторською заборгованістю та контролювати виконання контрактних зобов'язань. На відміну від типових облікових систем, які переважно орієнтовані на податкову звітність, запропоноване рішення фокусується на прогнозуванні та наданні інструментів для моделювання різних сценаріїв розвитку бізнесу.

Зміст поставлених завдань. Для досягнення поставленої мети необхідно:

- провести системний аналіз об’єкта;
- сформувати вимоги та структуру до аналітичної системи управління фінансами малого підприємства;
- розробити структуру інформаційного забезпечення системи;
- розробити алгоритми аналізу даних, та розробити саму аналітичну систему.

Апробація результатів дослідження. Основні положення роботи були представлені у вигляді тез "Аналітична система управління фінансами малого підприємства" на XV Міжнародна науково-практична конференція молодих вчених "Інформаційні технології: економіка, техніка, освіта" 2024. Із тезами подання можна ознайомитися у додатку А.

Структура роботи. Структурні елементи роботи розміщені в такій послідовності.

Перший розділ розглядає предметну область, здійснює постановку задачі дослідження та аналізує стан автоматизації сучасних тепличних господарств.

Другий розділ присвячено моделюванню системи. Були розроблені діаграми прецедентів використання, які дозволяють отримати передумови для аналізу поведінки системи.

Третій розділ присвячено розробці та реалізації системи. Було побудовано логічну модель даних та реляційну БД. Розглянуто розробку програмного забезпечення системи. Були розроблені алгоритми роботи системи.

Четвертий розділ розглядає результати дослідження, реалізацію аналітичного алгоритму системи та перспективи розвитку.

Обсяг роботи 51 сторінка, 13 рисунків, 3 таблиці, 1 додаток, 20 посилань на літературні джерела.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Малий бізнес – це вид підприємницької діяльності, що характеризується відносно невеликим розміром компанії, обмеженими фінансовими ресурсами, невеликою кількістю співробітників та низьким рівнем обсягів виробництва чи реалізації. У більшості країн критеріями визначення малого бізнесу є кількість працівників (як правило, до 50 осіб), річний дохід та балансова вартість активів, які є нижчими за встановлені межі. Малий бізнес зазвичай орієнтований на локальні ринки, має меншу бюрократичну структуру та гнучкість, що дозволяє швидко адаптуватися до змін у ринкових умовах. В Україні малий бізнес виконує важливу роль у створенні робочих місць, стимулюванні конкуренції та інновацій, а також є основою для розвитку середнього бізнесу в майбутньому. [1]

В Україні спостерігається розвиток малого підприємництва за тенденцією розвинених держав. Станом на грудень 2021 малий бізнес складає 25% ВВП, середній 27%, частка великого бізнесу складає 22% ВВП. У порівнянні із попередніми роками видно, що частка малого бізнесу у структурі ВВП зростає із 13% за 2013 рік, середній бізнес має відносно стабільну частку, а великий показує падіння своєї частки із 28%. Із цього можна зробити висновок, що малий бізнес розвивається і своєю часткою у структурі господарювання наближається до Європейського рівня в 56%.

Спілка українських підприємців (СУП) у дослідженні 2020 року нарахувала в Україні понад 1 млн підприємців. Більшість із них мають вищу освіту та розпочали справу в середньому 8,4 років тому завдяки власним заощадженням. Згідно їхнього звіту, майже половина малого та середнього бізнесу працюють у сфері роздрібної торгівлі [2].

Попри те, що більшість малих підприємств не надто динамічно розвивається, а дрібним бізнесом часто займаються просто для виживання, це все ж важлива альтернатива соціальному утриманству, тобто життю на виплати від

держави. Малий бізнес стає рятівним кругом і амортизатором економічних негараздів, тому він відіграє не стільки продуктивну економічну роль, скільки соціальну. Також він є початком для створення більшого бізнесу [19].

Бізнес – це один з рушіїв, що веде до інновацій, розробки нових технологій і впровадження нових ідей. Це сприяє підвищенню продуктивності, якості та ефективності виробництва, а також стимулює конкуренцію та розвиток нових ринків.

Проте ведення власного бізнесу для більшості людей видається неможливим та надто складним. В умовах постійних ризиків та життєвих випробувань така діяльність видається ризикованим кроком, що тягне за собою велику відповідальність та обов'язки.

В умовах поточної ринкової ситуації, де Україна прагне до частки малого бізнесу рівня Європейського Союзу, така форма діяльності видається все привабливішою для мотивованих людей. Із початком ведення бізнесу його власник зустрічається із багатьма складнощами, новими викликами і досвідом. Для полегшення цієї діяльності існують програмні рішення, що допомагають власникам самостійно управляти фінансами власної справи та вести звітність для сплати податків [3].

Методи та механізми управління фінансовими ризиками для малого підприємства є ключовими елементами в забезпеченні його стабільної платоспроможності, особливо в умовах високої економічної динамічності та невизначеності. Малий бізнес часто стикається з фінансовими ризиками, такими як ризики ліквідності, кредитні ризики, ринкові коливання та операційні ризики. Впровадження системного підходу до управління цими ризиками є важливим кроком для підтримки фінансової стійкості та забезпечення виживання підприємства [20].

Одним із фундаментальних методів управління є планування грошових потоків та оптимізація бюджету, що передбачає побудову прогнозів на основі історичних даних і поточних фінансових показників. Ефективний контроль

грошових потоків дає можливість підприємству передбачати потенційні фінансові труднощі, зокрема шляхом встановлення порогових значень для оборотних коштів, що дозволяє виявити та запобігти розривам у фінансуванні. Системи cash flow-аналітики також сприяють своєчасному моніторингу грошових залишків, забезпечуючи можливість оперативної реакції на зміни у фінансовому становищі підприємства.

Додатковим механізмом є диверсифікація доходів та управління портфелем клієнтів, що дозволяє знизити залежність від окремих джерел доходу і пом’якшити вплив непередбачених факторів, таких як сезонні коливання попиту. Це також допомагає підприємству адаптуватися до зовнішніх змін, не втрачаючи стабільного фінансового становища.

Застосування сучасного фінансового програмного забезпечення для аналізу та прогнозування ризиків є важливим інструментом для малого бізнесу, адже воно забезпечує можливість автоматизованого контролю основних фінансових показників. Такі системи дозволяють проводити своєчасний аналіз ринкових умов, виявляти відхилення від запланованих показників і оцінювати ефективність фінансової політики компанії. Наприклад, автоматизовані системи можуть допомогти в інтеграції фінансових даних, що дозволяє оперативно створювати звіти і надавати детальні прогнози щодо платоспроможності підприємства.

Таким чином, комплексний підхід до управління фінансовими ризиками включає як стратегічні методи, так і технічні інструменти, які дозволяють ефективно підтримувати фінансову стабільність малого бізнесу в умовах мінливого ринкового середовища.

1.2 Аналіз наявних рішень

Taxer.ua – це вебплатформа для автоматизації бухгалтерських і податкових процесів, спеціально розроблена для малого бізнесу та приватних підприємців. Основні функції сервісу включають створення та подання податкових

декларацій, формування рахунків і рахунків-фактур, а також відстеження податкових термінів. Ця система також містить податковий календар і можливість отримання юридичної консультації, що дозволяє підприємцям спростити процес звітності та зменшити ризик помилок. Завдяки функціоналу, який полегшує обробку документів і контроль за термінами оплати податків, Taxer.ua допомагає користувачам заощадити час, підвищити ефективність управління податковими зобов'язаннями та знизити адміністративне навантаження на підприємця [17].

Аналізуючи Taxer.ua в контексті розробки аналітичної системи для управління фінансами малого підприємства, можна зробити висновок, що цей сервіс є частковим рішенням, яке ефективно вирішує завдання з обліку та податкової звітності, але не забезпечує комплексної фінансової аналітики та прогнозування, що є важливими для стратегічного управління. Розробка власної системи може включати подібні функції для автоматизації обліку та звітності, але також розширити функціональність у напрямку фінансового прогнозування, аналізу грошових потоків, оцінки ризиків та управління контрактами. Це дозволить надавати користувачам більш повну інформацію для прийняття обґрунтованих фінансових рішень, орієнтованих на довгострокове планування.

Taxer.ua зосереджений на автоматизації рутинних операцій і податкових зобов’язань, що відповідає базовим потребам підприємців в обліку. Проте малий бізнес в Україні часто стикається з проблемами непередбачуваних грошових потоків, сезонними коливаннями, змінами в податковому законодавстві та економічною нестабільністю, які потребують більш гнучких інструментів для стратегічного планування. У таких умовах система, що пропонує додаткову аналітичну підтримку для прогнозування доходів, управління витратами та моніторингу фінансових ризиків, може забезпечити підприємству стійкість і адаптивність до змін.

Таким чином, можна зробити висновок, що платформа Taxer.ua є корисною для малих підприємств, які прагнуть полегшити податкову звітність і забезпечити відповідність вимогам законодавства. Однак для побудови

довгострокової стратегії розвитку бізнесу та адаптації до ринкових змін потрібна система з розширеними можливостями фінансової аналітики. Розробка такої системи для малого підприємства дозволить об’єднати базові облікові функції з інструментами для аналізу та прогнозування, які необхідні для стабільного розвитку в умовах високої динамічності сучасного бізнес-середовища.

Skynum.ua – це платформа для автоматизації обліку фінансів та управління бізнесом, орієнтована на малий та середній бізнес. Вона пропонує комплексні інструменти для управління складом, контролю за фінансами, ведення взаєморозрахунків із постачальниками і клієнтами, а також для створення різноманітних фінансових звітів [18].

Завдяки хмарному доступу, Skynum дозволяє користувачам працювати віддалено, без необхідності встановлення програмного забезпечення на локальних серверах, що є зручним для власників малого бізнесу, яким потрібно контролювати діяльність підприємства на ходу. Платформа дозволяє вести облік товарів по модифікаціях та серійних номерах, відстежувати терміни придатності продукції та використовувати різні одиниці виміру для товарів, що зручно для різних видів підприємницької діяльності, включаючи роздрібну торгівлю та сферу послуг.

Skynum також дозволяє генерувати фінансові звіти та відстежувати статистику продажів, що допомагає користувачам аналізувати фінансовий стан компанії і приймати більш обґрунтовані управлінські рішення, підтримуючи платоспроможність та конкурентоспроможність малого бізнесу.

Платформа дозволяє користувачам слідкувати за балансом, переглядати історію замовлень та оплат, контролювати обігові кошти, що важливо для стабільності малого бізнесу. Крім того, система надає графічні фінансові звіти, які полегшують аналіз грошових потоків і управління бюджетом. Це рішення також адаптоване для підприємств, що працюють у сфері роздрібної торгівлі, забезпечуючи комплексний складський і товарний облік.

1.3 Постановка завдання

Задачі дослідження:

1. Дослідити особливості фінансових ризиків для малого бізнесу, зокрема ризики кредитні та операційні ризики, що можуть впливати на платоспроможність.

2. Дослідити можливість створення алгоритму аналізу грошових потоків і моделювання можливих ризиків.

3. Визначити механізми моніторингу виконання контрактних зобов'язань і надання звітності про стан рахунків для вчасного виявлення можливих ризикових ситуацій. Це дозволить контролювати й оцінювати балансові показники для збереження фінансової стійкості підприємства.

4. Розробити процес подання податкових звітностей та автоматизувати його.

5. Визначити спосіб відображення стану дебіторської та кредиторської заборгованості, обсяг майбутніх витрат і зобов’язань, які потребують погашення, для надання користувачу можливості наочно бачити фінансовий стан компанії в реальному часі.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Визначення користувачів системи

Перед початком моделювання аналітичної системи управління фінансами малого підприємства потрібно визначити ким є цільовий користувач такої системи та його потреби.

Із попередніх розділів одразу можна виокремити власника малого підприємства. Він зазвичай виконує стратегічні функції управління бізнесом і контролює фінансовий стан підприємства. У невеликих компаніях він часто поєднує цю роль із виконанням адміністративних обов'язків.

Власник малого підприємства потребує ефективних інструментів для управління фінансами, зокрема наступні:

- моніторинг загального фінансового стану. Власнику важливо мати оперативний доступ до ключових фінансових даних, таких як доходи, витрати, баланс рахунків, та динаміка грошових потоків. Ці дані дозволяють йому оцінювати фінансову стабільність та платоспроможність компанії для ухвалення стратегічних рішень щодо нових інвестицій чи контрактів;
- аналіз ефективності. Для забезпечення зростання бізнесу власнику потрібна можливість відстежувати ключові показники ефективності та порівнювати їх із плановими значеннями. Це дозволяє оцінювати прибутковість і визначати ефективність фінансових рішень;
- прогнозування. Моделі прогнозування на основі історичних даних та поточних контрактів допомагають оцінити ризики та планувати розвиток бізнесу. Інструменти для створення сценаріїв дають змогу власнику змоделювати різні варіанти дій;
- простота управління. Оскільки власник часто не має глибоких технічних чи бухгалтерських знань, інтерфейс має бути інтуїтивно зрозумілим. Крім того, автоматичні нагадування про ризики чи важливі фінансові дати підвищують зручність використання системи [4].

Власник малого підприємства може зіткнутися з проблемами забезпечення фінансової прозорості, контролю за ресурсами та уникнення ризиків, пов'язаних із недотриманням законодавства або неправильним розподілом коштів [16]. Також може виникнути банальна нестача часу, або бажання диверсифікації своїх обов'язків. У таких випадках він може найняти до себе на роботу фінансового менеджера або бухгалтера.

Фінансовий менеджер – це спеціаліст, який відповідає за управління фінансовими ресурсами підприємства, аналіз фінансових даних, прогнозування, та розробку стратегій для досягнення фінансової стабільності [15]. Основні обов'язки фінансового менеджера включають:

- бюджетування, або ж розподіл фінансових ресурсів для оптимізації витрат та доходів;
- оцінка ефективності діяльності підприємства через аналіз ключових показників, таких як прибутковість та ліквідність;
- ризик-менеджмент та виявлення потенційних фінансових ризиків та розробка стратегій для їх мінімізації.

Бухгалтер – це фахівець, який забезпечує ведення фінансового та податкового обліку підприємства. Основні завдання бухгалтера включають:

- облік. Точна фіксація всіх фінансових операцій (доходи, витрати, платежі, податки);
- звітність. Складання фінансових звітів для власників бізнесу, податкових органів, інвесторів чи партнерів;
- контроль дотримання законодавства. Перевірка відповідності облікової документації до чинних нормативно-правових вимог.

Фінансовий менеджер допомагає стратегічно планувати розвиток бізнесу, тоді як бухгалтер фокусує увагу на операційному управлінні та дотриманні юридичних норм, звільняючи власника бізнесу від рутинної роботи з документами та надаючи йому більше часу для реалізації стратегічних цілей [5].

Також існує ринок із надання бухгалтерських послуг для окремих власників бізнесу. У таких випадках працівникам потрібно мати інструмент для здійснення управління фінансів для декількох клієнтів. Це актуально і для власника малого бізнесу, якщо він, наприклад, керує декількома магазинами і хоче розділити їх управління в системі.

2.2 Діаграма прецедентів

Діаграма прецедентів (use case diagram) необхідна для візуалізації взаємодії між користувачами (акторами) та функціями системи. Вона допомагає зрозуміти, які завдання система повинна виконувати, орієнтуючись на потреби користувачів. Цей інструмент сприяє визначенню вимог до системи, забезпечує прозорість для зацікавлених сторін і дозволяє уникнути помилок та не опустити щось важливе на ранніх етапах розробки.

Крім того, діаграма слугує основою для подальшого проектування та тестування системи. Вона допомагає планувати архітектуру, створювати інші моделі (наприклад, послідовності чи активності) і формулювати тестові сценарії. У контексті системи управління фінансами малого підприємства, діаграма прецедентів відображає основний функціонал (створення контрактів, формування звітів або контроль заборгованості), забезпечуючи ефективність і зручність у використанні.

Для системи було створено діаграму для визначення потреб користувачів (рис. 2.1). Щоб краще розуміти кожен із представлених прецедентів представлено пояснення.

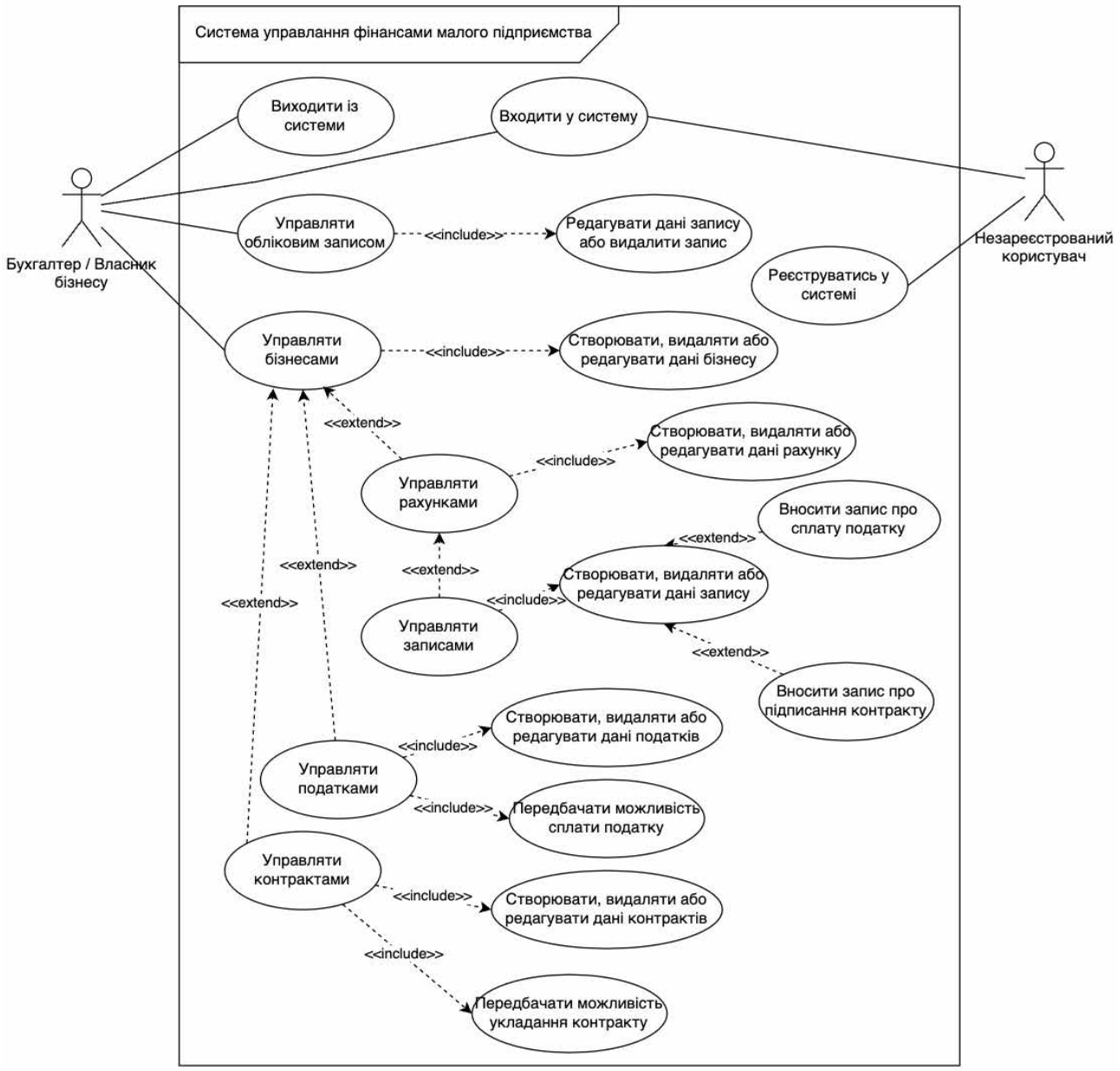

Рис 2.1 Діаграма прецедентів системи

Реєстрація у системі – це одна із двох можливих точок входу в систему. Користувач, котрий перший раз знайомиться із системою, має створити власний обліковий запис. Це необхідно для того, щоб система мала змогу ідентифікувати користувача та підпорядковувати йому створені ним об’єкти. Користувач повинен бути проінформованим із якою ціллю це робиться, так як для реєстрації необхідні його особисті дані.

Вхід у систему – друга із можливих точок входу. Користувач що вже має обліковий запис, може скористатися даними авторизації та увійти в систему. Система повинна обробити цю дію та представити користувачу функціонал застосунку.

Наступні прецеденти системи доступні лише користувачу, що пройшов авторизацію. Таким користувачем може бути власник бізнесу або бухгалтер.

Вихід із системи – користувач може вийти зі свого облікового запису для можливості входу в інший обліковий запис або захисту даних від інших користувачів пристрою, із якого виконано вхід.

Управління обліковим записом – система повинна дозволяти змінювати дані запису авторизованому користувачу а також видаляти його із усіма пов’язаними даними.

Управління бізнесами – власник бізнесу керуватиме бізнесами, якими володіє у реальному житті, дані про які вніс у систему. Це передбачає їх створення, видалення, та редагування даних про них.

Можливість управління рахунками, податками та контрактами (створення, видалення, та редагування) розширюють прецедент управління бізнесами, оскільки є доступними у системі лише після його наявності. Управління рахунками також додає можливість управління записами – створення, видалення, та редагування), а також окремо вносити запис про підписання контракту або сплату податку, якщо такі існують у системі.

В свою чергу, управління податками та контрактами включає у себе передбачення можливості їх укладання або сплати у майбутньому, грунтуючись на уже наявних даних у системі та встановлених нею прогнозів використання коштів. Ці прецеденти є ключовими у розроблюваній системі та досліджуваній проблематиці.

2.3 Use-Case та User Story

Use-Case – це опис взаємодії між користувачем (або зовнішньою системою) та програмним продуктом, що фокусується на досягненні певної цілі користувача. Use-Case визначає, як система має поводитися в різних сценаріях, і описує, які кроки необхідно виконати для досягнення бажаного результату.

Компоненти Use-Case:

1. Заголовок: Коротка назва сценарію, що відображає основну ідею (наприклад, "Створення контракту").
2. Актор: Хто взаємодіє із системою (наприклад, фінансовий менеджер).
3. Передумови: Умови, які мають бути виконані перед початком сценарію (наприклад, користувач має бути авторизований у системі).
4. Кроки: Чіткий опис дій, які виконуються актором та системою.
5. Результат: Очікуваний результат або вихідний стан (наприклад, контракт успішно створено і збережено в базі даних).
6. Альтернативні сценарії: Що станеться, якщо щось піде не за основним сценарієм (наприклад, у разі помилки в даних система надсилає повідомлення про необхідність виправлень).

Для даної роботи було пропрацьовано декілька Use-Case що описують використання основного аналітичного функціоналу системи, а саме створення нових контрактів. Дані Use-Case наведені у таблицях 2.1 – 2.3.

Таблиця 2.1

Use-Case для алгоритму створення контрактів

Актор	Власник малого підприємства, фінансовий менеджер.
Передумови	Власник має обліковий запис у системі, та створений профіль бізнесу.
Опис	Актор вводить дані контракту, такі як сума, опис, тип (дебетовий або кредитний), та дату. Система перевіряє фінансові дані, оцінює платоспроможність на основі поточного балансу, історичних даних та майбутніх зобов'язань. У разі успішної оцінки контракту присвоюється статус "очікування".
Результат	Контракт збережено в системі для подальшої обробки.
Альтернативний сценарій	Якщо система виявляє ризик неплатоспроможності, актор отримує попередження з детальними причинами.

Таблиця 2.2

Use-Case аналіз можливості виконання контракту

Актор	Власник малого підприємства, фінансовий менеджер.
Передумови	У системі вже є створені записи доходів, витрат, кредиторської та дебіторської заборгованості.
Опис	Актор обирає контракт для аналізу. Система обчислює поточний баланс, прогнозує надходження, враховує майбутні податкові платежі та перевіряє наявність кредиторської заборгованості. У разі позитивного результату видається підтвердження можливості виконання контракту.
Результат	Актор отримує рішення щодо здійсненності контракту та рекомендації.

Таблиця 2.3

Use-Case збереження чернетки контракту

Актор	Власник малого підприємства.
Передумови	Власник хоче внести дані контракту, але ще не готовий підтвердити його створення.
Опис	Актор заповнює необхідні дані, але натискає кнопку що збереже контракт як чернетку. Система фіксує всі введені дані без їх подальшої обробки.
Результат	Актор отримує рішення щодо здійсненності контракту та рекомендації.

User Story:

1. Як власник малого підприємства, я хочу створити контракт із зазначенням умов і суми, щоб система могла перевірити його здійсненність і уникнути ризиків неплатежів.

2. Як фінансовий менеджер, я хочу отримувати автоматичний аналіз фінансових ризиків під час створення контракту, щоб забезпечити платоспроможність компанії.

3. Як користувач, я хочу, щоб система відображала список чернеток контрактів, щоб я міг вносити зміни перед їх остаточним затвердженням.

4. Як фінансовий менеджер, я хочу, щоб система враховувала майбутні податкові зобов’язання під час перевірки контракту, щоб мінімізувати ризики перевитрати коштів.

5. Як власник, я хочу отримувати сповіщення, якщо контракт не відповідає фінансовим критеріям, щоб мати можливість переглянути умови або ухвалити інші рішення.

6. Як фінансовий менеджер, я хочу, щоб система автоматично враховувала попередні дані про контракти та фінансові операції для точного прогнозу.

7. Як фінансовий менеджер, я хочу мати доступ до інструментів прогнозування грошових потоків, щоб ефективно планувати витрати та уникати ризику дефолту.

8. Як бухгалтер, я хочу автоматично формувати звіти для податкових органів, щоб заощадити час і уникнути помилок у звітності.

9. Як власник малого підприємства, я хочу отримувати попередження про ризики невиконання контрактів або строки сплати податків, щоб уникнути штрафів і фінансових втрат.

10. Як бухгалтер, я хочу вводити дані про контракти та платежі у зручному інтерфейсі, щоб уникнути ручних помилок і прискорити обробку документів.

11. Як власник малого підприємства, я хочу швидко знаходити інформацію про фінансовий стан кожного контракту, щоб планувати витрати та доходи.

3 РОЗРОБКА СИСТЕМИ

3.1 Побудова фізичної моделі даних

На фізичній моделі бази даних зображений готовий до реалізації у СКБД дизайн БД для програмного рішення проблематики роботи. Для кожної із сутностей системи виділена окрема таблиця, у якій вони зберігатимуться. Із структурою таблиці можна ознайомитися на рисунку 3.1. Далі описано кожен із її елементів та зв’язків, які моделюються на цьому рівні проектування.

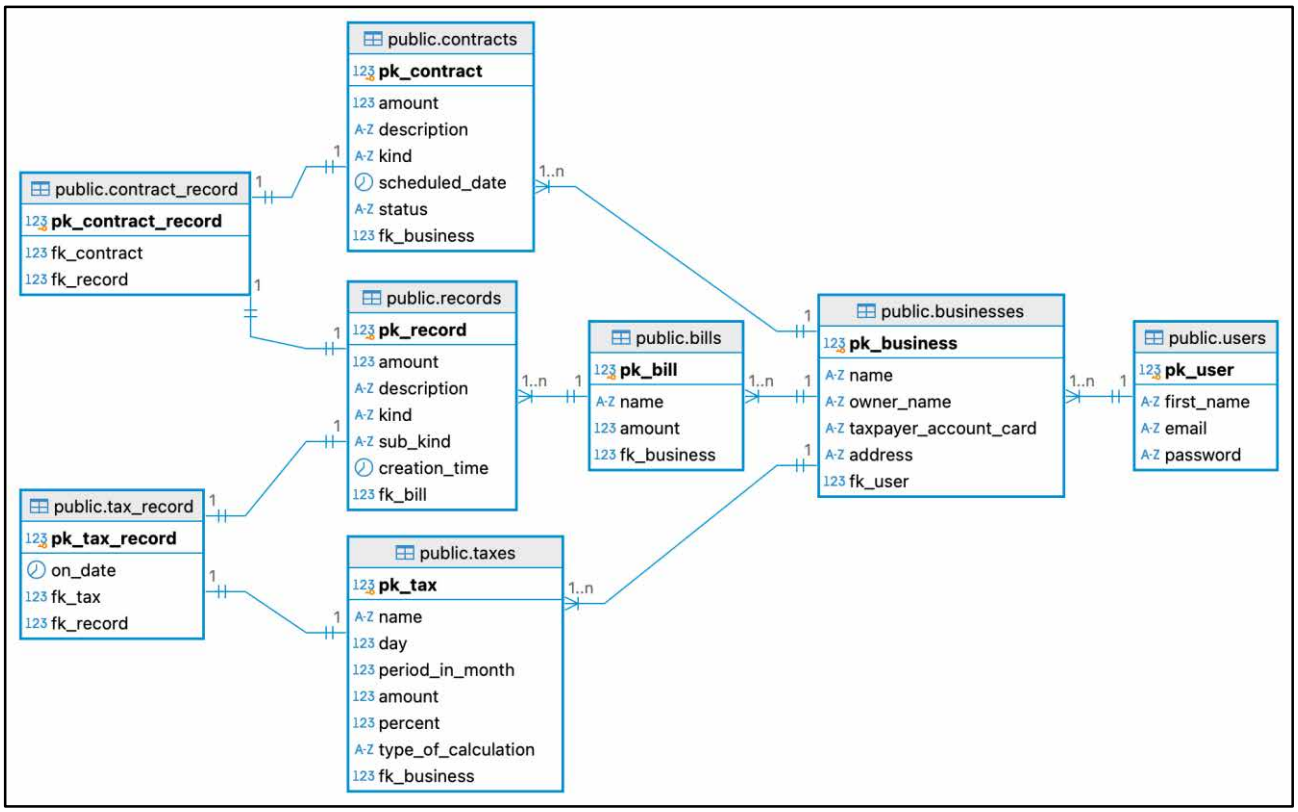

Рис. 3.1 Фізична модель даних

У таблиці "users" знаходяться дані користувачів, кожен із яких має унікальний ідентифікатор, а також поля імені ("first_name"), електронної пошти ("email") та паролю ("password") які, хоча не є необхідними для обчислювального алгоритму, надають можливість користування системою кільком користувачам, включаючи можливість захисту від небажаного доступу до приватних даних інших. За необхідності користувач може створити кілька акаунтів для із різними електронними поштами і керувати кількома окремими бізнесами.

Один користувач може мати кілька бізнесів, і кожен бізнес керується лише одним користувачем, що визначено зв’язком "один до багатьох" між таблицею користувачів та бізнесів "business". Пов’язаною з ним інформацією, що зберігається у базі, є назва бізнесу ("name"), ім’я власника бізнесу ("owner_name"), що фактично може відрізнятись від імені людини, що ним керує у системі, реєстраційний номер облікової картки платника податків, що необхідний для формування податкової звітності ("taxpayer_account_card") та адреса, за якою він зареєстрований ("address"). Із даними бізнесу напряму пов’язані дані контрактів (таблиця "contracts"), рахунків ("bills") та податків ("taxes") і їхній зв’язки також мають відношення один до багатьох (один бізнес може мати кілька пов’язаних рахунків і т.д.).

Для запису рахунку зберігаються його назва (поле "name") та сума коштів, що доступна на ньому ("amount").

Щоб реалізувати можливість витрати із рахунку або надходження коштів на нього була створена таблиця записів ("records") із такою інформацією – сума витрати або надходження ("amount"), опис запису ("description"), його вид ("kind"), підвид ("sub_kind"), а також час, коли він був внесений ("creation_time"). Із рахунком може асоціюватися багато записів, що реалізовано у відповідному зв’язку.

Дані сутності контрактів зберігаються у таблиці "contracts" і містять дані про суму цього контракту (поле "amount"), його опис ("description"), тип контракту що вказує на те чи це дебетова чи кредитна заборгованість ("kind"), час коли очікується що він буде здійснений ("scheduled_date") і його статус ("status").

Для сутності податків окрема таблиця "taxes" – містить поля імені ("name"), дня сплати ("day"), періоду в місяцях між сплатою цих податків ("period_in_month"), суми до сплати, якщо це фіксований податок ("amount"), відсотку від суми суми надходжень, якщо потрібен такий розрахунок ("percent") та способу його обчислення для випадків коли податок має мінімальну суму в

котрій потрібно враховувати як поле "amount" так і "percent" ("type_of_calculation").

Оскільки записи можуть як бути не пов’язаними із податками та контрактами, так і виходити із них, для встановлення цього зв’язку були створені проміжні таблиці "tax_record" та "contract_record". У них створюються записи лише в тому випадку, якщо запис спродуковано підписанням контракту або сплатою податку. Таблиця "tax_record" також містить поле "on_date", оскільки один податок сплачується періодично і потрібно розуміти за який період його було сплачено.

3.2 Реалізація бізнес-логіки аналітичної системи

Для створення цієї системи було обрано патерн дизайну систем Model-View-Controller.

MVC – це патерн у дизайні програмного забезпечення, який зазвичай використовується для реалізації користувацьких інтерфейсів, даних або ж бізнес-логіки та логіки управління. Він чітко проводить межу між бізнес-логікою та відображенням що презентує систему для взаємодії. Таке "розділення завдань" забезпечує кращий розподіл навантаження та покращення обслуговування і підтримки [6].

Три частини цього патерну проектування інформаційних систем описуються таким чином:

- модель (Model) – керує даними та бізнес-логікою;
- представлення (View) – керує макетом та відображенням;
- контролер (Controller) – спрямовує та керує командами до частин моделі та представлення.

Для розробки бізнес логіки даної системи було вирішено застосувати підхід DDD (Domain-Driven Design) – підхід до розробки бізнес-логіки, що виділяє її у домен.

Домен – це предметна область, до якої належить додаток, який розробляється. Наприклад, якщо компанія займається видачею кредитів чи обміном валют, такий домен називатиметься фінансовим. Іноді в компаніях одночасно йде розробка декількох паралельно активних бізнес-доменів. Організація може займатись продажем і транспортуванням (ремонтом, обслуговуванням або будь-чим ще) одночасно.

Своєю чергою, Domain-Driven Design, або DDD – це підхід до розробки програмного забезпечення, який реалізує конкретну модель предметної області та вирішує конкретну задачу бізнесу. Вона ставить бізнес-логіку на перший план і пропонує підходи та патерни для розробки складних додатків. DDD надає інструменти та практики, які допомагають розробникам та бізнес-експертам спільно працювати над моделюванням домену, використовуючи спільну мову та концепції, які є зрозумілими для цих сторін [14].

Доменна логіка є "серцем" додатку і повинна бути основною областю концентрації при проектуванні та розробці. Додаток розбивається на окремі контексти, що мають чітко визначені межі та відповідальності.

DDD дозволяє зменшити складність розробки та підтримки додатків, забезпечує більшу гнучкість та розширюваність, а також дозволяє ефективніше відповідати на зміни в бізнес-вимогах. Однак, використання DDD вимагає певного рівня експертизи та знань і може бути витратним за часом на початкових етапах розробки.

На прикладі розроблювальної системи, можна виділити домен "податок", у якого буде діяльність – "створити податок".

У самому ж коді це буде представлено класом, що описує доменну сутність з її властивостями, та функціями, що описують та проводять необхідні процеси моделюючи при цьому реальні.

Програмні системи тяжіють до хаосу. Коли починається побудова нової системи, у інженерів є грандіозні ідеї, щодо чистого та добре впорядкованого коду, але з часом виявляється, що він накопичує помилки і крайні випадки

(проблема або ситуація, яка виникає лише за межами нормальних робочих параметрів – зокрема, коли кілька змінних або умов навколишнього середовища одночасно знаходяться на екстремальних рівнях, навіть якщо кожен параметр знаходиться в межах встановленого діапазону для цього параметра [7]), і в кінцевому підсумку перетворюється на складну до розуміння павутину класів-менеджерів і утилітарних модулів. Інженери приходять до того, що розумно розшарована архітектура розвалилася сама по собі. Хаотичні програмні системи характеризуються одноманітністю функцій: API-обробники, які знають домен, надсилають електронну пошту та ведуть журнали; класи "бізнес-логіки", які не виконують жодних обчислень, але виконують ввід/вивід; і все це пов'язано з усім іншим так, що зміна будь-якої частини системи стає небезпечною. Це настільки поширене явище, що інженери-програмісти мають власний термін для позначення хаосу: антипаттерн "Велика куля бруду" (Big Ball of Mud) (рис. 3.2) [8].

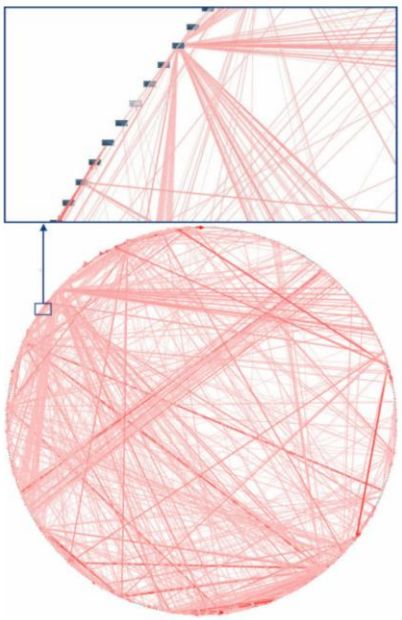

Рис. 3.2 Діаграма залежностей класів у великих системах [8]

Тому даний підхід розглядає методи подолання такої заплутаності залежностей шляхом відокремлення доменів у модуль, від якого йтимуть всі залежності, але який натомість не буде залежати від іншої системи та своїх

реалізацій.

Одним із кроків є зміна порядку та структури побудови системи. У традиційних реалізаціях використовується шарова архітектура (рис. 3.3), у якій компоненти послідовно залежать одне від одного та реалізація бази даних стоїть в голові залежностей, через що бізнес-логіка стає залежною від цієї реалізації. Це означатиме, що будь-які зміни в базі даних вестимуть зміни у бізнес-логіці, що неправильно, враховуючи те, що програмні інженери пишуть системи, які направлені на те, щоб задовольнити потреби бізнесу, а не вносити у нього зміни обумовлені внутрішньою архітектурою. У такому підході міграція на іншу СУБД, що кардинально відрізняється від застосовуваної, може потягнути за собою переписування системи та великі витрати часу.

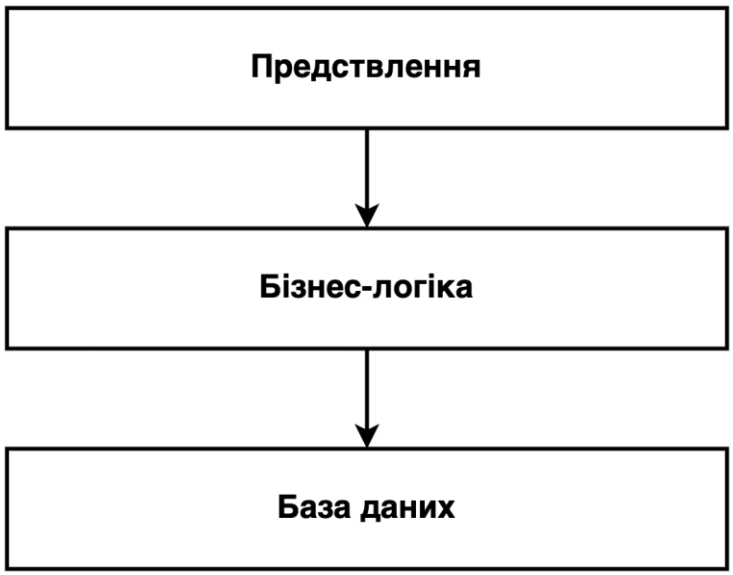

Рис. 3.3 Шарова архітектура системи

У DDD ставиться за ціль те, щоб доменна модель не мала залежності, так, щоб проблеми з інфраструктурою не вносили корективи в доменну модель і не сповільнювали модульні тести або здатність вносити зміни.

Тому доменна модель ставиться у центрі системи так, щоб усі залежності йшли від неї. Така архітектура називається цибулевою (рис. 3.4).

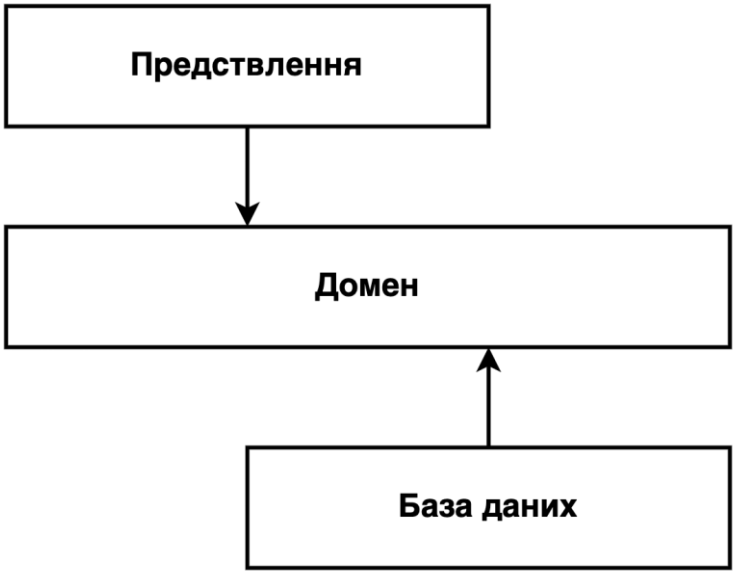

Рис. 3.4 – Цибулева архітектура системи

На рисунку видно, що база даних тепер залежить від домену. Це досягається використанням патерну репозиторій. Патерн Репозиторій (Repository pattern) є шаблоном проектування, який використовується для розділення логіки доступу до даних від іншої логіки додатку. Він реалізує абстракцію між доменом додатку і способом, яким дані зберігаються та отримуються з бази даних або іншого джерела.

Основна ідея патерну репозиторій полягає в тому, щоб мати окремий шар (клас або набір класів), відповідальний за роботу з сховищем даних, таким як база даних. Цей шар називається репозиторієм. Він надає єдиний інтерфейс для взаємодії з даними без прямого залучення доменної логіки до деталей роботи з базою даних [9].

Таким чином із застосуванням вищезгаданих підходів було зпроектовано та розроблено програмну реалізацію аналітичної системи обліку фінансів. Відповідно до структури фізичної моделі даних, для кожного її об’єкту був створений доменний відповідник (рис. 3.5). Це забезпечує дотримання цибулевої архітектури системи де всі компоненти повинні залежати від доменної частини.

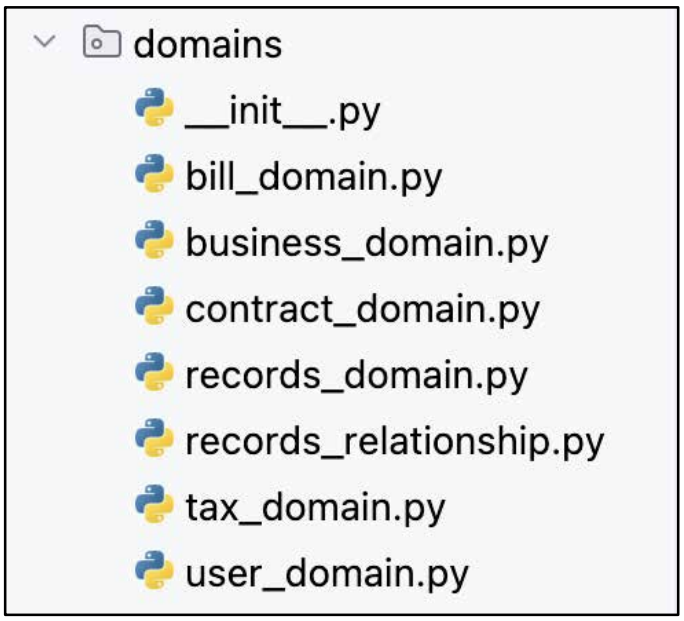

Рис. 3.5 Структура створених доменів

Кожен домен є класом у системі, котрий диктує вигляд бази даних, де зберігаються дані. Під кожен такий клас створений сервіс який має власний набір функцій для виконання операцій над даними. Обов'язковою є реалізація базових операцій створення, редагування, читання та видалення об'єктів. Інші функції вже залежать від специфіки свого домену. Із основним функціоналом можна ознайомитися у 4 розділі цієї роботи, де описані результати дослідження.

Окрім цього, для реалізації патерну Репозиторій було створено абстрактний репозиторій із яким взаємодіє доменна область (таким чином попереджаються залежності). Під час запуску застосунку на місце абстрактного інтерфейсу репозиторію підставляється потрібна реалізація. Це дозволяє спроектувати тестовий репозиторій, що зберігатиме дані у пам'яті для реалізації unit-тестування у подальшій роботі (на даному етапі роби цього не було реалізовано).

Із реалізацією репозиторіїв система набуває наступного виду (рис. 3.6). Як видно на рисунку, доменна область напряму не взаємодіє із БД, як це реалізується в традиційних системах, а використовує інтерфейс наданий репозиторієм.

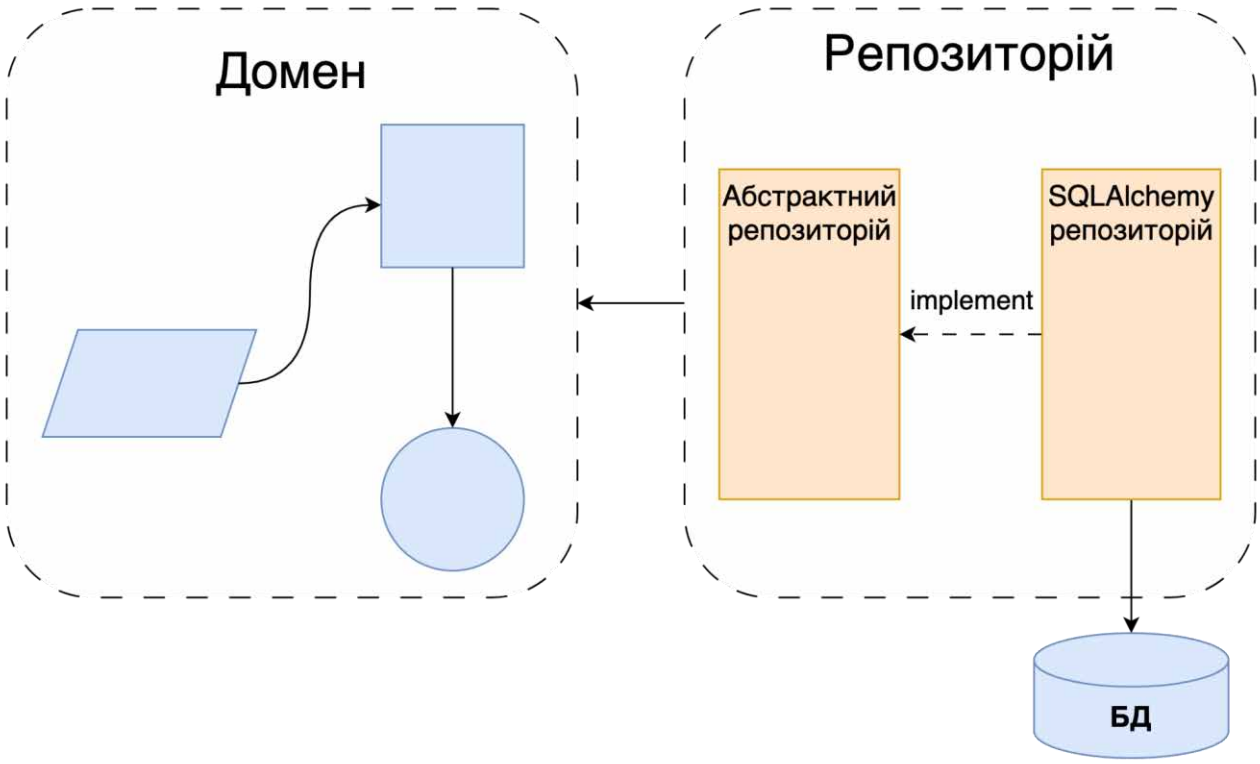

Рис. 3.6 Реалізація патерну Репозиторій у системі

3.3 Розробка представлення інформаційної системи

У процесі розробки програмної реалізації роботи було вирішено, що доцільним представленням даної системи буде API ресурс із котрим буде легко взаємодіяти та просто розробляти без витрати часу на надлишкові речі у рамках магістерської роботи, як ось візуальний інтерфейс.

API (Application Programming Interface) – це набір інструментів, протоколів і стандартів, що дозволяє одній програмі взаємодіяти з іншою. API визначає, як програмне забезпечення або сервіси можуть обмінюватися даними та функціональністю, надаючи стандартизований спосіб для реалізації цих взаємодій.

Основна функція API – спрощення інтеграції між різними програмами та системами, забезпечуючи доступ до даних чи функцій, без необхідності знати технічні деталі реалізації іншого додатка.

Для даної роботи було реалізовано REST API. Цей вид використовує протокол HTTP для передачі даних у форматах JSON чи XML. Дуже популярний завдяки простоті та масштабованості. У даному випадку дані передаються у форматі JSON.

Для поточної системи були створені ресурси, котрі відповідають доменній структурі застосунку (рис. 3.7). У такий спосіб розробка представлення полегшується через чіткість розуміння наступного об’єму для роботи.

```
7   app = FastAPI()
8
9   app.add_middleware(
10      CORSMiddleware,
11      allow_origins=ORIGINS,
12      allow_credentials=True,
13      allow_methods=['*'],
14      allow_headers=['*'],
15  )
16
17  app.include_router(bills_api.router)
18  app.include_router(businesses_api.router)
19  app.include_router(records_api.router)
20  app.include_router(users_api.router)
21  app.include_router(contracts_api.router)
```

Рис. 3.7 Структура ресурсів у застосунку

3.4 Обґрунтування вибору засобів програмування та середовища розробки

Для реалізації бізнес логіки було обрано мову Python. Python є відмінним вибором для написання бізнес-логіки:

- хистий і зрозумілий синтаксис, що полегшує розробку та розуміння коду. Це забезпечує швидке і ефективне написання бізнес-логіки без великої

кількості зайвого коду. Також цьому допомагають вбудовані бібліотеки структур даних що допомагають чітко описувати сутності описані у вимогах бізнесу;

- делика екосистема, що включає багато корисних бібліотек та фреймворків для розробки бізнес-логіки. Такими є бібліотеки, що полегшують розрахунки та взаємодію із великими масивами даних. Також можливо підібрати бібліотеки, що допомагатимуть проводити статистичний аналіз даних;
- Python є кросплатформною мовою програмування, що дозволяє запускати створені застосунки на платформах Windows, macOS і Linux без модифікації коду. Це дозволяє розгортати бізнес-логіку на різних середовищах та забезпечує більшу гнучкість для розробників у виборі оточення для розробки;
- хоча Python не є найшвидшою мовою програмування, вона забезпечує прийнятну швидкість виконання для багатьох бізнес-застосунків. Крім того, є можливість використання сторонніх бібліотек написаних на C та C++, що дозволяє оптимізувати виконання важких обчислень. За потреби ці бібліотеки можуть бути написані і самостійно;
- велика та активна спільнота розробників, яка надає підтримку, документацію та приклади коду. Це спрощує вирішення проблем та дозволяє швидко знайти відповіді на запитання. Також це означає, що мова залишатиметься актуальною ще довгий час.

Після визначення мови, за допомогою якої буде розроблятися бізнес логіка застосунку, було обрано бібліотеки, для використання на початковому етапі розробки.

Для реалізації бізнес сутностей обрано бібліотеку dataclasses, яка є частиною стандартної бібліотеки Python, починаючи з версії 3.7. Вона надає простий спосіб оголошення та роботи з "data classes" (класами даних) в мові Python.

Data classes використовуються для опису даних, які мають обмежену кількість полів, і для яких потрібно автоматично генерувати стандартні методи ініціалізації, представлення та логічних операцій. Без використання бібліотеки dataclasses методи доводиться писати вручну, що може бути часо- та

ресурсозатратно. Також використання dataclasses дозволяє виразити структуру даних одним компактним класом. Це поліпшує читабельність коду та зрозумілість структури даних, що важливо чистоти коду.

Важливим плюсом цієї бібліотеки є те, що вона є сумісною із іншими подібними бібліотеками, що використовуватимуться в інших частинах застосунку.

Для реалізації взаємодії із базою даних було обрано бібліотеку SQLAlchemy. Це потужна бібліотека для роботи з базами даних у мові програмування Python. Вона надає високорівневий інтерфейс ORM (Object-Relational Mapping) та доступ до низькорівневих SQL запитів до реляційних баз даних.

ORM (Object-Relational Mapping) – це технологія, яка дозволяє взаємодіяти з реляційною базою даних, використовуючи об'єктно-орієнтований підхід. Вона вирішує проблему взаємодії між об'єктами програми і записами в базі даних.

За допомогою ORM можна виконувати операції з базою даних, такі як створення, зчитування, оновлення та видалення даних, використовуючи об'єктну модель замість написання прямих SQL-запитів. ORM перетворює об'єкти в реляційні структури даних (таблиці, рядки, стовпці) і забезпечує автоматичне створення та виконання SQL-запитів.

ORM приховує складність SQL-запитів та деталі взаємодії з базою даних, дозволяючи програмістам працювати з об'єктами та методами, які більш природні для мови програмування. Автоматична генерація SQL-запитів дозволяє уникнути ручного написання та повторення коду.

Також ORM допомагає уникнути SQL-ін'єкцій та інших потенційних вразливостей безпеки шляхом використання параметризованих запитів і захищених методів доступу до даних.

SQLAlchemy надає можливість імперативного та декларативного опису БД. Різниця між цими підходами відноситься до того, як описується структура та взаємозв'язки між таблицями бази даних.

Імперативний підхід – це підхід, де використовуються класи та об'єкти для створення та взаємодії з таблицями бази даних. Програміст визначає таблиці, стовпці та їх взаємозв'язки, створюючи класи моделей (ORM-класи) вручну. При використанні імперативного підходу, визначається структура бази даних через програматичне створення об'єктів, налаштування відношень та використання методів класів для збереження, оновлення та видалення даних. Імперативний підхід дає повний контроль над структурою бази даних і можливості детального налаштування. Користувач можете програмно впливати на кожен аспект створення та взаємодії з базою даних. Однак, це також вимагає більшого обсягу коду для опису моделей та їх взаємозв'язків. Деякі складні операції, такі як генерація SQL-запитів, можуть ускладнюватися при використанні імперативного підходу.

Декларативний підхід – це підхід, де використовуються декларативні класи для опису структури таблиць та взаємозв'язків. Замість того, щоб вручну створювати класи моделей та конфігурувати їх вручну, використовується базовий клас declarative_base() і визначаються класи моделей, що успадковуються від нього. SQLAlchemy використовує механізми метаданих та рефлексії для автоматичного створення таблиць та їх структури на основі опису класів моделей. Програміст описує взаємозв'язки між таблицями за допомогою зв'язків між класами. Декларативний підхід дає більш простий спосіб визначення структури бази даних, приховуючи багато деталей технічного рівня.

Слідуючи книзі Architecture Patterns with Python, для реалізації патерну репозиторій було обрано імперативний спосіб створення зв’язку із базою даних. Це дозволяє описати сутності домену ізолюючі їх від інших частин системи, а у цьому випадку недопускаючи залежність домену від вибраної ORM, а також описати таблиці фізичної бази даних за всіма вимогами.

Для реалізації представлення застосунку було прийнято рішення створити API сервіс, що буде отримувати запити які надходять, використовувати методи доменної області та повертати результат роботи системи.

API (Application Programming Interface) – це набір правил і протоколів, які визначають спосіб взаємодії між різними програмами. API визначає, які функції, методи та процедури доступні для використання, а також формати даних, які можуть передаватися між програмами [10].

API може бути реалізовано у вигляді бібліотеки, набору функцій, вебсервісу або спеціального протоколу обміну даними. Він надає інтерфейс, через який програми можуть спілкуватися, обмінюватися інформацією та виконувати певні операції.

REST (Representational State Transfer) API – це архітектурний стиль для розробки веб-сервісів, який використовує стандартні протоколи HTTP для передачі і обміну даними між клієнтом і сервером. REST API базується на принципах ресурсо-орієнтованої архітектури, де ресурси ідентифікуються унікальними URL-адресами [13].

REST API розділяє клієнтську і серверну частини, що дозволяє їх незалежно розвивати та масштабувати. Проте це вимагає того, що кожен запит до REST API повинен містити всю необхідну інформацію для його обробки, без залежності від попередніх запитів. Сервер не зберігає стан клієнта між запитами. Тому це накладає додаткові умови на продуману реалізацію, щоб сервер із мінімальної кількості вхідних даних міг виконати бажані операції.

Для реалізації цього архітектурного стилю в Python було обрано FastAPI, що є високопродуктивним фреймворком для розробки API. Він поєднує в собі швидкість виконання, простоту використання та автоматичну документацію API [11].

Основні переваги FastAPI:

- *висока швидкість* через базування на бібліотеці Starlette та використовує корутини (механізм програмування, який дозволяє виконувати обчислення асинхронно та кооперативно; є спеціальним видом підпрограм, які можуть бути призупинені та відновлені у певних точках виконання) asyncio, що дозволяє досягти високої продуктивності та швидкості виконання запитів.

- *підтримка типованої анотації* функцій та автоматичну генерацію схеми даних API за допомогою бібліотеки Pydantic. Це полегшує валідацію вхідних даних та видачу відповідей з правильною структурою. Також така типованість додає стабільності самого коду та допомагає у його підтримці.
- *підтримка асинхронних запитів* та використання асинхронних бібліотек, що дозволяє розробникам створювати швидкі та ефективні веб-додатки.
- *легка інтеграція із Python-екосистемою* що дозволяє взаємодіяти з іншими бібліотеками та фреймворками Python, такими як вибрана ORM SQLAlchemy. Це дає можливість використовувати потужні інструменти для роботи з базами даних, асинхронних задач та інших функцій у сполученні з FastAPI.

Однією із вимог є формування книги обліку доходів для ФОП. Для цього було обрано спосіб її генерації як PDF документа що буде відображатися користувачу у вікні браузера, та буде доступний для завантаження.

Для формування цього документа було обрано бібліотеку Python pdfkit. Ця бібліотека надає можливість генерації файлів PDF з HTML (стандартизована мова розмітки документів) або текстових даних. Вона використовує зовнішній інструмент під назвою wkhtmltopdf, який конвертує HTML-або текстові файли в формат PDF. Також за потреби ширшої кастомізації може застосовуватися CSS (спеціальна мова, яка використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних).

Для формування шаблону звітності буде використано інструмент Jinja2. Це потужна бібліотека для шаблонізації в Python. Вона надає можливість розділення логіки програми та представлення даних шляхом використання шаблонів що пишуться із використанням HTML. Jinja2 використовує синтаксис схожий на Python, що робить цей інструмент зрозумілим та легким для використання. Також підтримуються різні функціональні блоки, такі як цикли та умови, що дозволяє краще представляти дані.

PDF документ дозволяє у статичній формі представити звітність без необхідності відображати на стороні користувача динамічні таблиці з даними. Також при відображенні PDF документів у браузері за допомогою компонента object, можна без додаткової реалізації надавати користувачу можливість завантажити цей звіт для локального збереження.

Система розроблялася в середовищі PyCharm.

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python. Воно призначене для полегшення розробки, тестування та налагодження Python-проектів, надаючи розширені можливості та інструменти для продуктивної роботи.

PyCharm надає зручне середовище для роботи з віртуальними середовищами, пакетним менеджером і залежностями проекту.

3.5 Організаційна структура прикладного програмного забезпечення

Організаційну структуру кожної частини системи можна представити як діаграму пакетів. Це структурна діаграма, яка відображає взаємозв'язки між пакетами в програмному проекті. Вона допомагає візуалізувати організацію проекту, його модульну структуру та залежності між компонентами.

На діаграмі пакетів (рис. 3.8) компоненти зображуються у вигляді прямокутників, які містять назву пакета. Зв'язки між пакетами показуються у вигляді стрілок або ліній, що вказують на залежності між пакетами. Зв'язки можуть вказувати на залежність одного пакета від іншого, імпорт, використання чи включення пакету в інший [12].

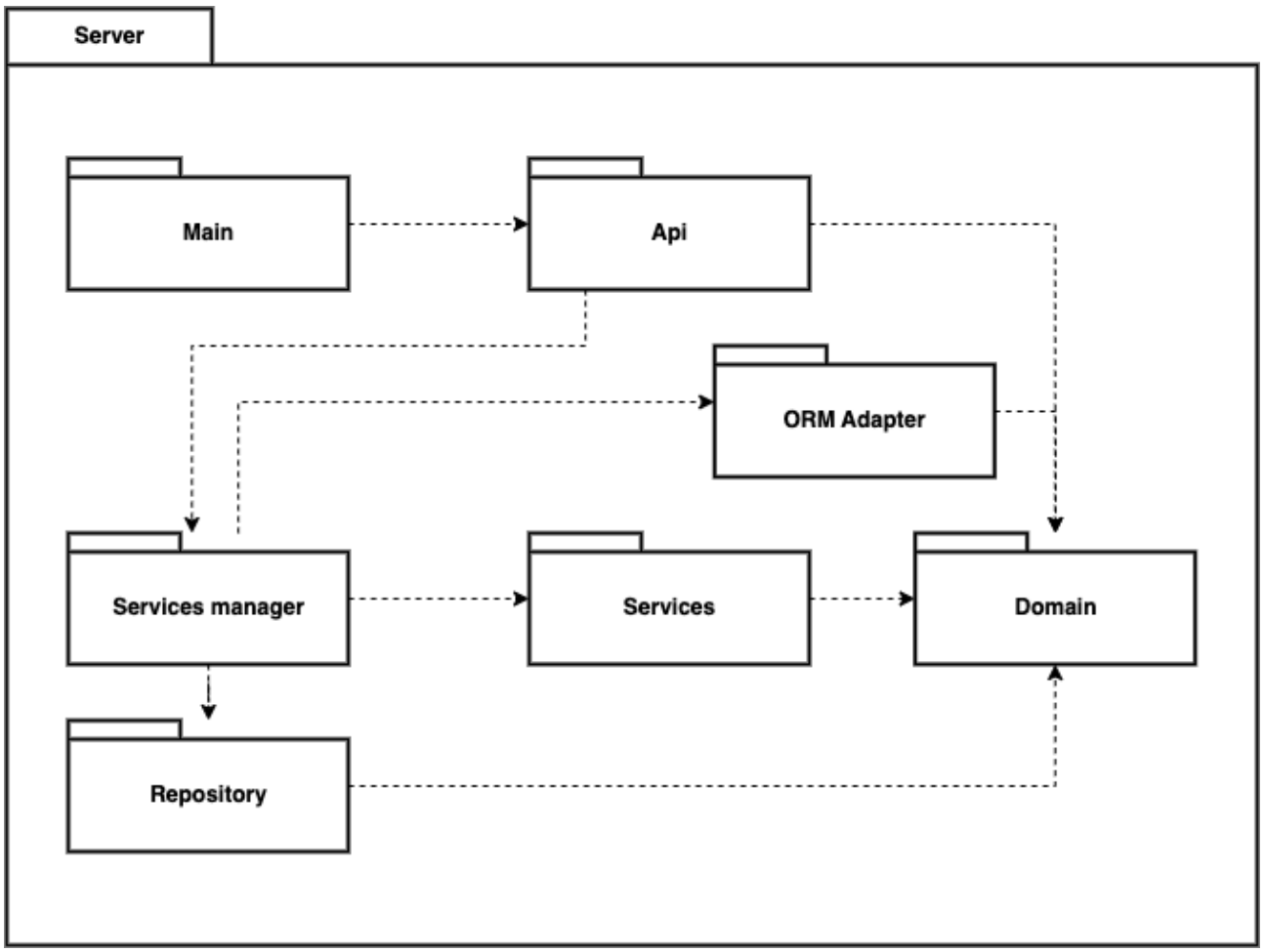

Рис. 3.8 Діаграма пакетів серверної частини додатку

Діаграма дозволяє орієнтуватися у структурі проекту, відображає організацію коду на рівні пакетів та допомагає виявити можливі проблеми залежностей між компонентами. Вона є потужним інструментом для аналізу та візуалізації архітектури програмного проекту і полегшує його розробку та підтримку.

Domain. Як було розглянуто в пункті 3.2, домен – це модуль з описом бізнес сутностей системи. На діаграмі видно, що було збережено вимогу незалежності домена від інших частин системи. Сам модуль описує класи із якими проходить взаємодія у системі без орієнтації на реалізацію.

Services. Цей модуль має залежність лише від пакету Domain. У ньому описані процеси, які відбуваються у системі і які маніпулюють сутностями із Domain. Цей модуль розпорошений на файли та відповідні класи кожен із яких відповідає за власну сутність домену. Кожен сервіс при створені очікує

екземпляр абстрактного класу репозиторію із яким він може взаємодіяти для зберігання даних у сховищі.

Ці два пакети описують бізнес логіку без залежностей від інших компонентів. За потреби вони можуть бути винесеними в окрему бібліотеку яка використовуватиметься для різних реалізацій системи. До прикладу, окрім API ресурсу можна створити gRPC сервіс для реалізації мобільних та стільничних (desktop) застосунків що можуть взаємодіяти із сервером виконуючи віддалені функції, що значно швидше у порівнянні з використанням API. Також може бути потреба у заміні фреймворку, що також стає можливим при використанні такої архітектури.

Repository. Цей пакет залежить лише від домену для того, аби використовувати специфічні помилки пов’язані із доменом, щоб в реалізації можна було їх обробляти та відповідно змінювати поведінку.

ORM Adapter. Цей пакет пов’язаний із обраною ORM та також використовує домен. У ньому імперативно пов’язуються сутності із домену та відповідні структури таблиць в базі даних. У результаті зміни об’єктів із домену якими маніпулює система, будуть відображатися у сховищі даних, яким виступає БД.

Services manager. Пакет, який являє собою вузол, що пов’язує всі вищеперелічені пакети. У ньому ініціалізуються всі сервіси, встановлюється сеанс із базою даних та ініціалізує репозиторій, передаючи його сервісам у використання.

Api. Цей пакет реалізує сервіси REST API, до яких буде звертатися користувач системи. Цей пакет розпорошений на окремі ресурси кожен з яких взаємодіє із своїм сервісом. Результатом цього є те, що цей пакет залежить від domain та services manager для правильної взаємодії з бізнес логікою.

Main. У цьому пакеті всі Api ресурси пов’язуються в один вузол. Він є вхідною точкою всього серверного застосунку.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Результати розробки системи

Результатом даної роботи став API ресурс (англ. application programming interface). Ресурс дозволяє користувачу створити свій обліковий запис, що є вхідною точкою у структурі системи. Після цього користувач може створити необхідну кількість бізнесів – структур, фінансами яких він розпоряджається. Це відповідає потребі користувачів, що ведуть бухгалтерію декількох бізнесів, володіють кількома бізнесами, або мають потребу в розділенні одного бізнесу на декілька. Звернення до системи здійснюється шляхом взаємодії із її API ресурсами. Їх зручне представлення забезпечується використання бібліотеки FastAPI що дозволяє використовувати свої ресурси через OpenAPI (рис. 4.1).

Рис. 4.1 Інтерфейс OpenAPI розробленої системи

Для ведення фінансової звітності користувач повинен створити рахунки, за якими він здійснює свої фінансові операції. Вони можуть відображати рахунки у

різних банках, рахунки в одному банку, або ж той логічний поділ власного капіталу котрий зручний користувачеві. Користувач може створити лише один рахунок, що відображає увесь капітал його бізнесу, система не ставить ніяких обмежень перед цим.

Фінансова звітність здійснюється шляхом ведення користувачем записів його фінансових операцій. Фінансові операції здійснюються поза даною системою, а користувач самостійно вносить дані котрі можуть відображати кожну окрему операцію, підсумок за день чи місяць. На основі цих даних користувач може сформувати книгу обліку доходів для подання податкової звітності. Дану звітність користувач отримує у вигляді файлу у форматі PDF. Приклад звіту показано на рисунку 4.2.

Книга обліку доходів фізичної особи-підприємця

Кравченко Арсеній Миколайович

- дата початку ведення 01.01.2023;
- РНОКПП 1234567890;
- адреса місця реєстрації фізичної особи-підприємця: м. Чернігів, вул. Доценка, 17, кв. 35;
- одиниця виміру доходу за господарськими операціями – грн з копійками;

ОБЛІК ДОХОДІВ ЗА 2023 РІК

Період	Сума доходів	Із них				Перевищення річного ліміту	ЄСВ	Податок
		Готівковий розрахунок	Безготівковий розрахунок	Безоплатно отриманий дохід	Повернення коштів			
січень	68900.0	0	68900.0	0	0	Немає	1474	3445.0
лютий	0	0	0	0	0	Немає	1474	0.0
березень	50489.0	0	50489.0	0	0	Немає	1474	2524.45
Всього, за 1 квартал	**119389.0**	**0**	**119389.0**	**0**	**0**	**Немає**	**4422**	**5969.45**
квітень	15000.0	0	0	20000.0	-5000.0	Немає	1474	750.0
травень	470.0	400.0	70.0	0	0	Немає	1474	23.5
червень	2640.0	2640.0	0	0	0	Немає	1474	132.0
Всього, за 2 квартал	**18110.0**	**3040.0**	**70.0**	**20000.0**	**-5000.0**	**Немає**	**4422**	**905.5**
липень	0	0	0	0	0	Немає	1474	0.0
серпень	2000000.0	0	2000000.0	0	0	Немає	1474	100000.0
вересень	0	0	0	0	0	Немає	1474	0.0
Всього, за 3 квартал	**2000000.0**	**0**	**2000000.0**	**0**	**0**	**Немає**	**4422**	**100000.0**
жовтень	0	0	0	0	0	Немає	1474	0.0
листопад	0	0	0	0	0	Немає	1474	0.0
грудень	0	0	0	0	0	Немає	1474	0.0
Всього, за 4 квартал	**0**	**0**	**0**	**0**	**0**	**Немає**	**4422**	**0.0**
Всього за рік	**2137499.0**	×	×	×	×	**Немає**	**17688**	**106874.95**

Рис. 4.2 Докладний вигляд звітності

Для проведення аналітики фінансів, а також побудови прогнозів на основі історичних даних і поточних фінансових показників було зроблено можливість створення контрактів. Ця сутність дозволяє створити у системі уявлення майбутнього прибутку, або витрати. Контракт для користувача є своєрідним нагадуванням та червоним прапорцем, котрий попереджає про майбутню витрату, або стає рятівним кругом (у випадку коли це майбутній прибуток) у спробах запланувати фінансові операції. При підтвердженні виконання

контракту створюється відповідний запис у системі для того, аби ці дані враховували у податковій звітності.

Система дає користувачеві можливість створювати відображення податків, котрі він має сплачувати. Це дозволяє враховувати їх при плануванні майбутніх витрат котрі уособлює сутність контрактів.

4.2 Алгоритм створення контрактів

Основним результатом даної магістерської роботи є виокремлення сутності "Контракт". Вона відіграє ключову роль у контексті аналітичної системи управління фінансами малого підприємства, оскільки забезпечує централізоване управління зобов’язаннями підприємства та контролює фінансові потоки, пов’язані з виконанням договірних умов. Контракт – це основна одиниця обліку взаємодії з клієнтами, постачальниками, або партнерами, яка дозволяє формалізувати фінансові відносини та забезпечує прозорість і прогнозованість процесів.

Використовуючи дані контрактів, система може прогнозувати ризики невиконання зобов’язань, контролювати залишкові суми для сплати податків, а також оцінювати поточну платоспроможність підприємства.

Доцільність використання сутності "Контракт" зумовлена її здатністю бути джерелом актуальної інформації для аналізу, контролю та прийняття рішень. Такий підхід забезпечує як короткострокову стабільність, так і довгострокову фінансову стійкість підприємства.

У механізмі створення контракту криється основний аналітичний механізм який попереджує користувача від необдуманих дій та дозволяє прогнозувати його можливості на майбутнє.

На основі діаграми діяльності (рис. 4.3), демонстровано алгоритм створення контрактів в аналітичній системі управління фінансами малого підприємства. Діаграма охоплює процес створення контрактів, який починається

з запиту користувача (власника малого підприємства або фінансового менеджера). Система аналізує можливість укладання контракту, базуючись на фінансових даних підприємства, дотримуючись послідовних етапів.

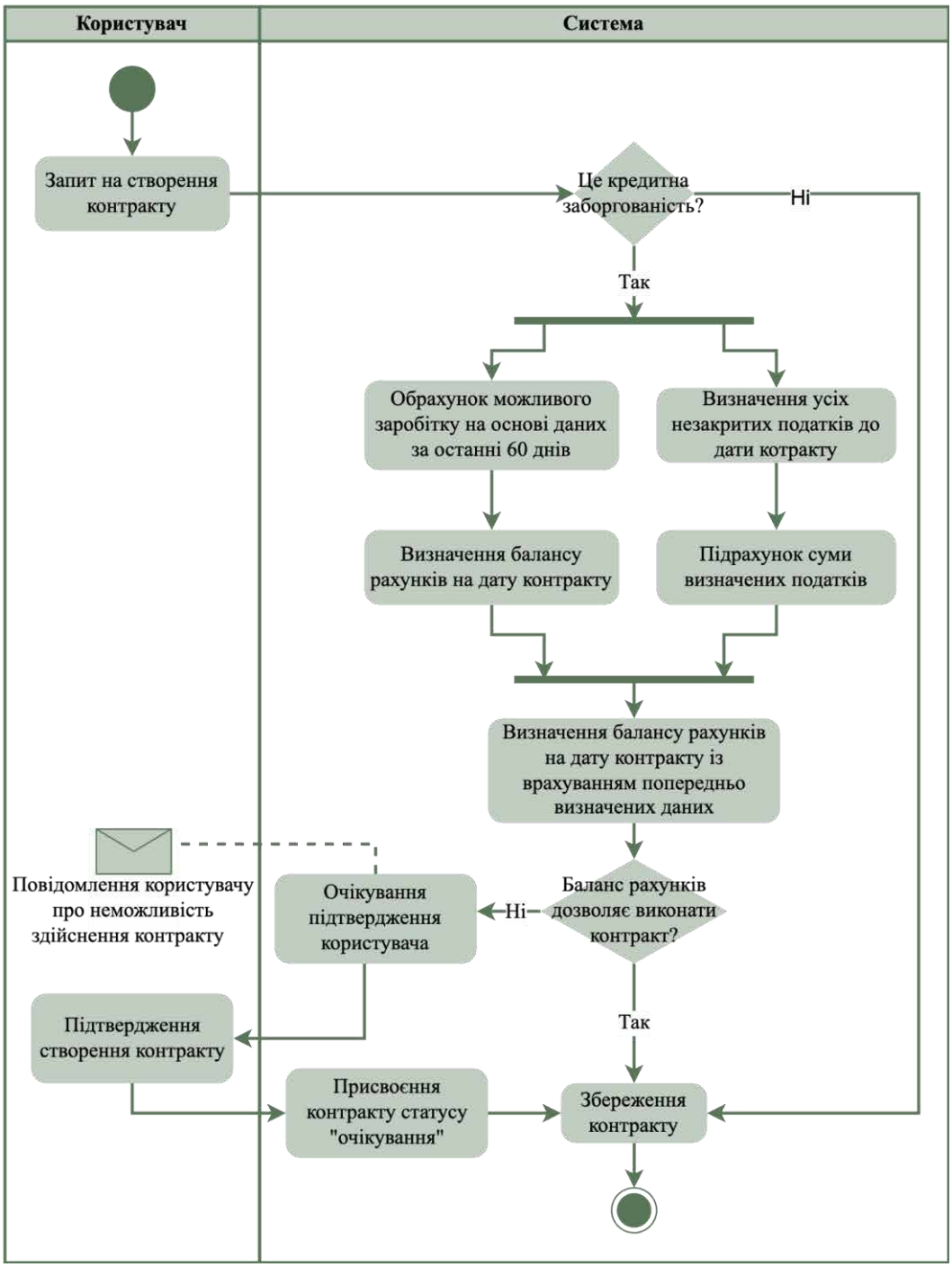

Рис. 4.3 Діаграма послідовності алгоритму створення контракту

Спочатку система перевіряє чи контракт є кредитною заборгованістю. У такому разі вона переходить до розрахунків можливого доходу на основі даних за останні 60 днів та аналізу балансу рахунків на дату контракту. Якщо кредитна

заборгованість є, система визначає всі податки, які повинні бути сплачені до моменту укладання контракту, та оцінює суми цих податків. Баланс рахунків, скоригований на заплановані податки, перевіряється на достатність для виконання контракту. Якщо баланс негативний, користувач отримує повідомлення про неможливість укладання контракту.

У випадку позитивного балансу система автоматично генерує контракт зі статусом "очікування". Вона надсилає користувачу повідомлення для підтвердження або редагування контракту. Після підтвердження статус змінюється на "активний", і контракт зберігається у системі. Із дизайном графічної реалізації даного процесу можна ознайомитися на рисунку 4.4.

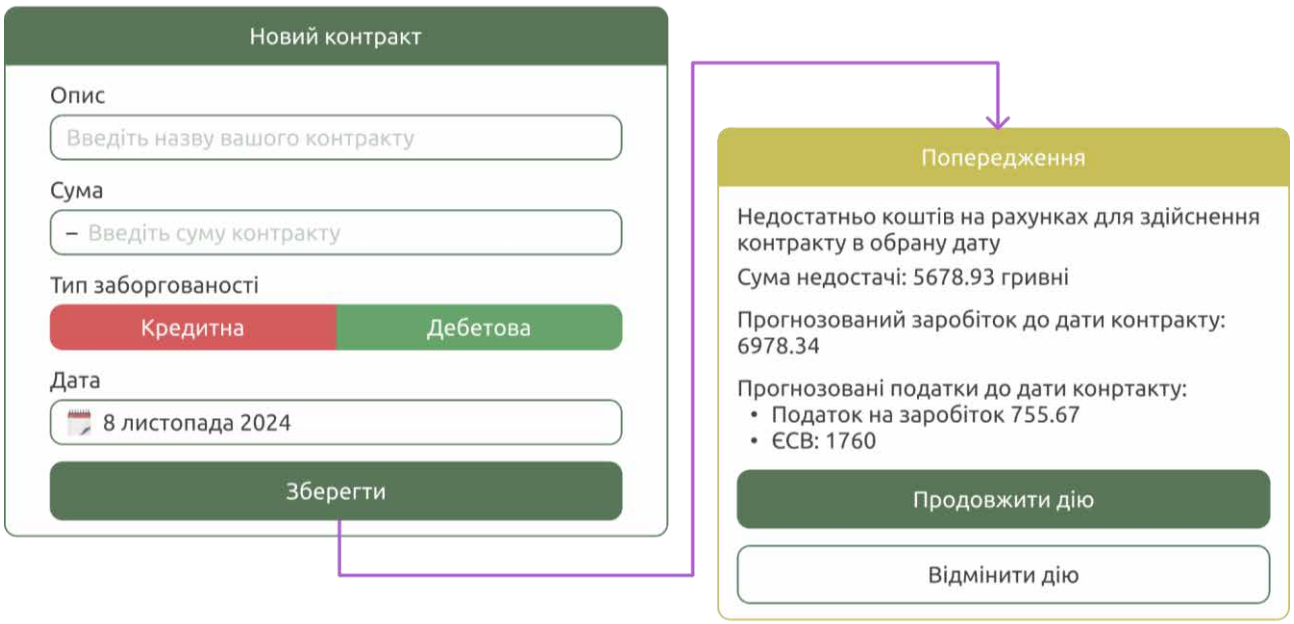

Рис. 4.4 Дизайн процесу створення контракту

Фундаментом для формування історичних даних стосовно яких здійснюється аналіз виступає сутність "Записи". Вона забезпечує збереження інформації про доходи та витрати підприємства, виконання контрактів та сплати податків.

4.3 Перспективи розвитку системи

Одним із перспективних напрямів розвитку аналітичної системи управління фінансами є впровадження інтерактивного календаря платежів. Цей інструмент буде ключовим у забезпеченні фінансової дисципліни та зниженні ризиків пропуску важливих дат для бізнесу. Календар автоматично відображатиме фінансові зобов’язання, зокрема терміни сплати податків, виконання контрактів і погашення кредитів, роблячи управління фінансами прозорішим.

Візуалізація в календарі виконуватиметься через кольорові позначення: зелене підсвічування сигналізуватиме про виконані зобов’язання, жовте попереджуватиме про наближення термінів (2-5 днів), а червоне вказуватиме на критичні чи прострочені дні. Це допоможе користувачам легко визначати пріоритети в управлінні фінансами.

Система також надсилатиме нагадування через email або месенджери, що сприятиме оперативному реагуванню на наближення важливих дат. Для днів із високим ризиком (червоне підсвічування) календар може пропонувати рекомендації, такі як корекція графіка платежів або перерозподіл коштів.

Такий функціонал значно спростить управління фінансовими потоками, допоможе уникати штрафів за прострочення і підтримуватиме фінансову стабільність малого бізнесу. Він стане інтуїтивно зрозумілим інструментом як для власника, так і для фінансового менеджера.

Генеративні алгоритми широко інтегрується у сучасному світі. Тому одним із напрямків може бути впровадження таких технологій для прогнозування фінансових потоків і ризиків, підвищення точності прогнозування та автоматизації рутинних завдань. Це дозволить системі автоматично виявляти аномалії в даних, аналізувати поведінку клієнтів і партнерів, а також пропонувати оптимальні стратегії для покращення фінансової стабільності підприємства. Наприклад, алгоритми машинного навчання можуть

адаптуватися до змінних ринкових умов, що зробить систему ще більш ефективною у прийнятті рішень.

Ще одним перспективним напрямком є інтеграція з мобільними платформами, що забезпечує постійний доступ до системи з будь-якого пристрою. Це відкриває можливості для гнучкого управління фінансами, навіть у віддаленому форматі. Інтеграція із сучасними платіжними системами та банківськими API дозволяє автоматизувати платежі, формувати платіжні графіки та мінімізувати ризики затримок у виконанні фінансових зобов'язань.

ВИСНОВКИ

У процесі виконання магістерської роботи була досягнута мета – розробити аналітичну систему управління фінансами малого підприємства, яка забезпечує ефективний моніторинг, аналіз і прогнозування фінансового стану бізнесу. Розроблена система дозволяє оптимізувати управління фінансовими ризиками, забезпечувати платоспроможність підприємства, а також надає власникам бізнесу та бухгалтерам необхідні інструменти для прийняття обґрунтованих рішень.

Завдяки системному аналізу об'єкта дослідження були визначені основні виклики та потреби малого бізнесу у фінансовому управлінні. Було сформовано вимоги до системи, які враховують необхідність моніторингу виконання контрактів і звітності щодо платіжної дисципліни. Розроблено структуру інформаційного забезпечення, яка дозволяє інтегрувати дані про доходи, витрати, зобов’язання.

Розроблено алгоритми аналізу грошових потоків, які дозволяють моделювати ризики та прогнозувати фінансові результати. Визначено та впроваджено механізми моніторингу виконання контрактів, які включають аналіз стану рахунків, дебіторської та кредиторської заборгованості, а також валідації фінансового стану при створенні контрактів що дозволяє прогнозувати їх виконання.

У системі автоматизовано процес формування податкової звітності. Це дозволяє покращити фінансову стійкість компанії. Також забезпечено візуалізацію фінансових показників, що надає власникам і фінансовим менеджерам можливість аналізувати балансові показники, оцінювати ризики та приймати стратегічні рішення на основі достовірних даних.

Розроблена система є гнучкою і може масштабуватися для інтеграції з іншими бізнес-інструментами, що відкриває перспективи її подальшого розвитку та адаптації до нових викликів у сфері управління фінансами малого підприємства.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шварц І. В. Особливості управління малими підприємствами в Україні. 2018. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/20116/Shvarts%20I.V.%20PFD%202018.pdf?sequence=3

2. Угніва С. Хребет національної економіки. Чим живе малий і середній бізнес в Україні та як він відвойовує свою частку у національному ВВП. NV. 2021. Вип. 48. URL: https://biz.nv.ua/ukr/economics/maliy-biznes-v-ukrajini-umovi-roboti-perspektivi-chim-zaroblyaye-novini-ukrajini-50203466.html (дата звернення 16.11.2024).

3. Терещенко Я. І. Проблеми та перспективи розвитку малого підприємництва в Україні. Двадцять друга всеукраїнська практично-пізнавальна інтернет-конференція. URL: http://naukam.triada.in.ua/i ndex.php/konferentsiji/52-dvadtsyat-druga-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/535-problemi-ta-perspektivi-rozvitku-malogo-pidpriemnitstva-v-ukrajini (дата звернення 16.11.2024).

4. Ефективне управління фінансами малого бізнесу. JoyPup: веб сайт. URL:https://joy-pup.com/ua/business-ua/efektivne-upravlinnya-finansami-malogo-biznesu/ (дата звернення 16.11.2024).

5. Управління фінансами в бізнесі. Yudey: веб сайт. URL: https://yudey.com.ua/urok-15-upravlinnia-finansamy-v-biznesi (дата звернення 16.11.2024).

6. MVC. MDN Web Docs: веб сайт. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення 16.11.2024).

7. Learn More! What are Edge Cases and Corner Cases? Medium : веб сайт. URL: https://medium.com/@dojobox.id/learn-more-what-are-edge-cases-and-corner-cases-f1e0321ed3ea (дата звернення 16.11.2024).

8. 8. Harry J.W. Percival, Bob Gregory. Architecture Patterns with Python. USA, O'Reilly, 2020. С 29–96

9. S. Millett, N. Tune. Patterns, Principles, and Practices of Domain-Driven Design. Germany, Wiley, 2015. С. 479–486.

10. What is an API? Mulesoft : веб сайт. URL: https://www.mulesoft.com/api/what-is-an-api (дата звернення 16.11.2024).

11. FastAPI framework. FastAPI official website : вебсайт. URL: https://fastapi.tiangolo.com/lo/ (дата звернення 16.11.2024).

12. Діаграми пакетів, компонентів і розміщення. Waykun : вебсайт. URL: https://ua.waykun.com/articles/diagrami-paketiv-komponentiv-i-rozmishhennja.php (дата звернення 16.11.2024).

13. Thiago C. A Formal Semantics for Use Case Diagram Via Event-B. J. Softw., 2017, 12.3: 189-200.

14. Domain-Driven Design: Simple Explanation. DEV: веб сайт. URL: https://dev.to/alexhyettdev/domain-driven-design-simple-explanation-23b0 (дата звернення 16.11.2024).

15. Бучко Т. Стратегічне управління фінансами підприємства. С. 104–105 URL: https://elartu.tntu.edu.ua/bitstream/lib/42940/2/MNPK_2023_Buchcko_T-Strategic_management_of_104-105.pdf

16. Олександренко І., Чиж Н. Управління фінансами підприємств малого бізнесу. 2022. URL: https://e-forum.com.ua/uk/journals/tom-12-3-2022/upravlinnya-finansami-pidpriyemstv-malogo-biznesu

17. Taxer.ua: веб сайт. URL: https://taxer.ua/uk/kb (дата звернення 16.11.2024).

18. Skynum: веб сайт. URL: https://skynum.com/ (дата звернення 16.11.2024).

19. Малий бізнес після війни. Українське радіо : веб сайт. 2023. URL: http://www.nrcu.gov.ua/news.html?newsID=100880 (дата звернення 16.11.2024).

20. Лактіонова О. А. Управління фінансовими ризиками. 2020. URL: https://r.donnu.edu.ua/bitstream/123456789/1460/1/%D0%9D%D0%B0%D0%B2%D1%87%20%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA%20%D0%A3%D0%A4%D0%A0%2027_10_2020.pdf

Додаток А

Тези "Аналітична система управління фінансами малого підприємства" на XV Міжнародній науково-практичній конференції молодих вчених "Інформаційні технології: економіка, техніка, освіта"

Сторінок 2

УДК 004.9:658.1

АНАЛІТИЧНА СИСТЕМА УПРАВЛІННЯ ФІНАНСАМИ МАЛОГО ПІДПРИЄМСТВА

Возний О.І., науковий керівник Руденський Р. А. д.ек.н., професор

Актуальність такої розробки зумовлена стрімким розвитком малого бізнесу в Україні за тенденцією розвинених держав. Це накладається на сучасні умови бізнес-середовища, які характеризуються високою динамічністю та нестабільністю. Малий бізнес, котрий часто працює з обмеженими фінансовими ресурсами, особливо вразливий до економічних коливань, змін у законодавстві, зростання податкових та кредитних зобов'язань. Ефективне управління фінансами стає вирішальним для його виживання і розвитку, а створення системи, яка дозволяє здійснювати аналітичну підтримку фінансових рішень, допомагає значно підвищити стабільність і конкурентоспроможність малого підприємства[1][2].

Однією з ключових проблем для малого бізнесу є непередбачуваність грошових потоків. Це пов’язано з нерегулярністю доходів, сезонними коливаннями попиту та невизначеністю щодо постійних клієнтів. У таких умовах прогнозування і контроль над фінансовими потоками є вирішальним. Аналітична система, котра допомагатиме відслідковувати баланс рахунків, виконувати оцінку можливих заробітків і витрат дозволить підприємству уникати дефіциту коштів, вчасно реагувати на фінансові ризики та підтримувати платоспроможність. Через наведені фактори виникає необхідність у системі управління та аналітики фінансів такої форми підприємництва.

Для створення системи що вирішуватиме описану проблематику було виділено сутність “Контракт”. Він є важливим елементом у розроблюваній системі, оскільки він виконує роль інструменту для відображення фінансових зобов'язань і можливостей компанії.

Контракт забезпечує фіксацію майбутніх фінансових надходжень (дебетова заборгованість) та витрат (кредитна заборгованість), що допомагає підприємству оцінити свої поточні та майбутні можливості.

На рисунку 1 діаграма діяльності показує процес створення контракту. Користувач ініціює запит; якщо це дебетова заборгованість, то такий контракт одразу буде створений, якщо кредитна – алгоритм продовжує опрацьовувати запит. Для цього обраховуються кілька складових:

1. Розраховується можливий заробіток на основі даних за останні 60 днів;
2. Визначається баланс рахунків на дату контракту (на нього можуть впливати також й інші контракти);
3. Перевіряються неоплачені та майбутні податки та обчислюється їх загальна сума.

Сукупність цих даних дозволяє з’ясувати чи контракт можна успішно здійснити. Якщо ні – користувачу надсилається повідомлення про неможливість виконання і система очікує підтвердження на продовження. Після підтвердження контракту йому присвоюється статус "очікування" і він зберігається в системі. Це дозволяє відокремити такі контракти для того аби користувач міг до них повернутися і зробити зміни. При збереженні змін алгоритм заново повторюється.

Таким чином користувач системи отримує можливість планувати бюджет та експериментувати із ним для знаходження оптимальних шляхів вирішення свого завдання.

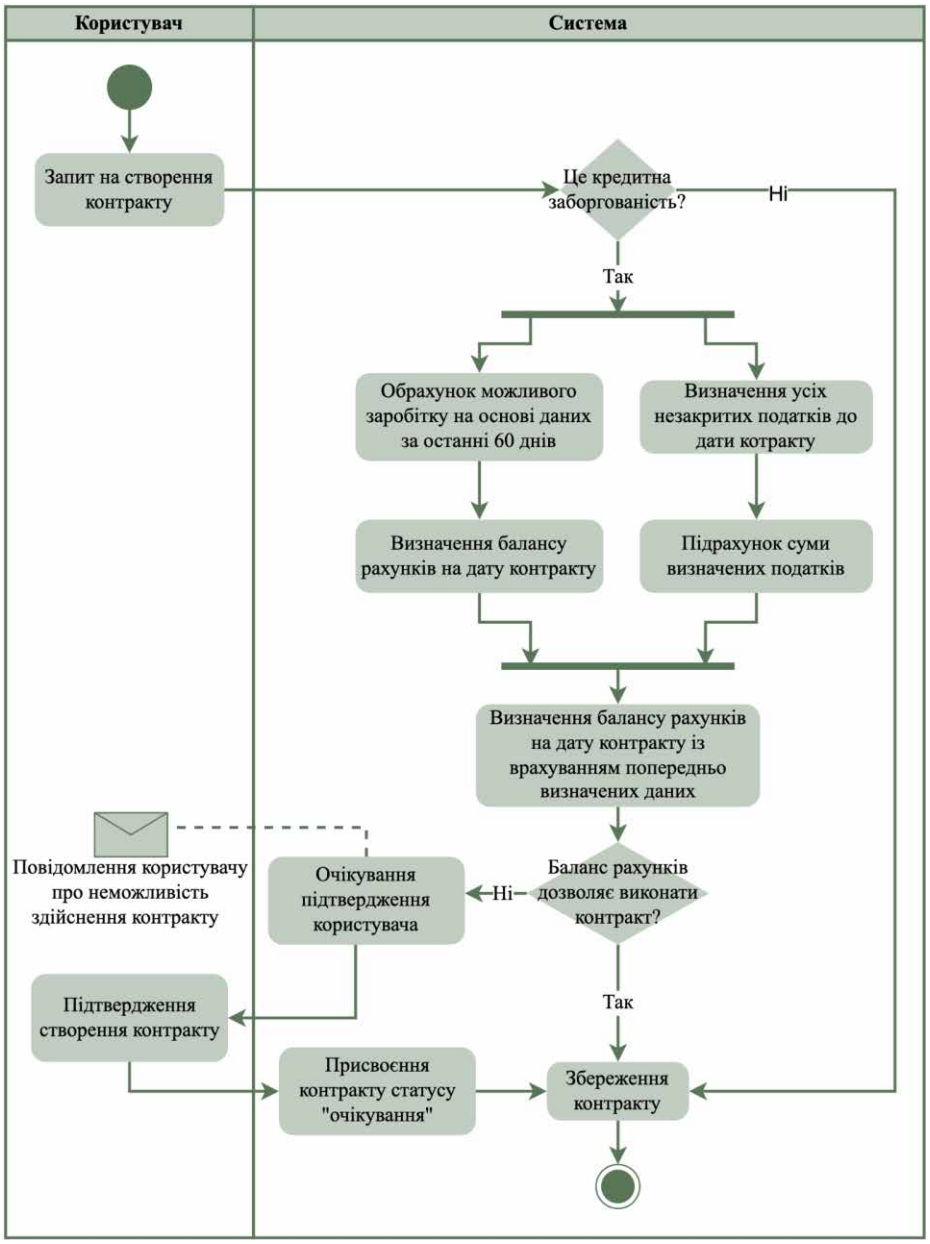

Рис. 1 Діаграма діяльності створення контракту

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Журнал Forbes Ukraine [Електронний ресурс]: “Terra incognita української економіки. Чотири висновки з дослідження середнього бізнесу Інституту економічних досліджень”. – Режим доступу: https://forbes.ua/business/terra-incognita-ukrainskoi-ekonomiki-chotiri-visnovki-z-doslidzhennya-serednogo-biznesu-institutu-ekonomichnikh-doslidzhen-31012024-18884 (дата звернення: 02.11.2024)
2. “Фактори розвитку малого бізнесу”, Рижко О.В [Електронний ресурс] – Режим доступу: https://economyandsociety.in.ua/journals/2_ukr/62.pdf (дата звернення: 04.11.2024)