МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

МИКИТЮК Вікторія Віталіївна

Інформаційний застосунок управління операціями небанківської фінансової установи/ Information Application for Operations Management of a Non-Banking Financial Institution

спеціальність: 122 - Комп'ютерні науки

освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконала студентка групи КНз-41

В.В. Микитюк

_____

Науковий керівник

д.т.н., доцент М.З. Домбровський

_____

Кваліфікаційну роботу допущено до захисту

«___» _____ 20__ р.

Завідувач кафедри

_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ – 2026

Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
МИКИТЮК Вікторії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Інформаційний застосунок управління операціями небанківської фінансової установи/Information Application for Operations Management of a Non-Banking Financial Institution

керівник роботи к.т.н., доцент, М.З. Домбровський
затверджений наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: методичні рекомендації кафедри, нормативні документи щодо діяльності небанківських фінансових установ, документація Python, Flask, SQLAlchemy, PostgreSQL/SQLite, матеріали з проєктування баз даних та вебзастосунків.

4. Основні питання, які потрібно розробити:
− проаналізувати предметну область небанківської установи на прикладі ломбарду;
− визначити функціональні та нефункціональні вимоги до інформаційного застосунку;
− спроєктувати алгоритмічне та інформаційне забезпечення системи;
− розробити структуру бази даних і ERD-модель;
− реалізувати інформаційно-обчислювальний застосунок мовою Python із використанням Flask та SQLAlchemy;
− провести тестування основних функцій системи.

5. Перелік графічного матеріалу у роботі:
- структурна схема системи;
- блок-схема алгоритму створення договору;
- ERD-модель бази даних;
- UML-діаграма класів.

6. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студентка _____ В.В. Микитюк
підпис

Керівник роботи _____М.З. Домбровський
підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Інформаційний застосунок управління операціями небанківської фінансової установи» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом 54 сторінок і містить 6 рисунків, 17 таблиць, 4 додатки та 27 використаних джерел.

Кваліфікаційна робота присвячена розробленню інформаційного застосунку управління операціями небанківської фінансової установи. Як приклад предметної області розглянуто діяльність ломбарду, що передбачає облік клієнтів, заставного майна, договорів, платежів і статусів виконання фінансових зобов'язань.

У роботі проаналізовано предметну область небанківської установи, визначено основні операційні напрями ломбарду та встановлено потребу в автоматизації облікових операцій. Проведено огляд існуючих інформаційних застосунків для автоматизації ломбардної діяльності, визначено їхні функціональні можливості, переваги та недоліки.

Розроблено алгоритмічне та інформаційне забезпечення системи. Визначено основні сутності бази даних, спроєктовано структуру таблиць, описано зв'язки між ними та сформовано ERD-модель. Запропонована структура бази даних забезпечує зберігання інформації про клієнтів, користувачів, заставне майно, договори та платежі.

Програмну реалізацію інформаційного застосунку виконано мовою Python із використанням фреймворку Flask, бібліотеки SQLAlchemy та вебтехнологій HTML, CSS, Bootstrap і Jinja2.

Проведено тестування основних функцій інформаційно-обчислювального застосунку. Результати тестування підтвердили працездатність системи та відповідність реалізованого функціоналу поставленим вимогам.

Ключові слова: ІНФОРМАЦІЙНИЙ ЗАСТОСУНОК, УПРАВЛІННЯ ОПЕРАЦІЯМИ, НЕБАНКІВСЬКА ФІНАНСОВА УСТАНОВА, ЛОМБАРД, БАЗА ДАНИХ, PYTHON, FLASK, SQLALCHEMY, ВЕБЗАСТОСУНОК.

ANNOTATION

The qualification thesis on the topic “Information Application for Operations Management of a Non-Banking Financial Institution” for obtaining the Bachelor’s degree in specialty 122 “Computer Science” of the educational program “Computer Science” consists of 54 pages and contains 6 figures, 17 tables, 4 appendices and 27 references.

The qualification work is devoted to the development of an information application for operations management of a non-banking financial institution. The activity of a pawnshop is considered as an example of the subject area because it includes operations with clients, pledged property, contracts, payments, and statuses of financial obligations.

The paper analyzes the subject area of a non-banking financial institution, identifies the main operational areas of a pawnshop, and determines the need to automate operations management. Existing information applications for pawnshop automation are reviewed, and their functional capabilities, advantages, and disadvantages are identified.

The algorithmic and information support of the system is developed. The main database entities are defined, the table structure is designed, relationships between them are described, and an ERD-model is formed. The proposed database structure provides storage of information about clients, users, pledged property, contracts, and payments.

The implementation of the information application is performed using Python, the Flask framework, the SQLAlchemy library, and web technologies such as HTML, CSS, Bootstrap, and Jinja2.

The main functions of the information application are tested. The testing results confirm the operability of the system and the compliance of the implemented functionality with the specified requirements.

Keywords: INFORMATION APPLICATION, OPERATIONS MANAGEMENT, NON-BANKING FINANCIAL INSTITUTION, PAWNSHOP, DATABASE, PYTHON, FLASK, SQLALCHEMY, WEB APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних.

СУБД — система управління базами даних.

КР — кваліфікаційна робота.

ІС — інформаційна система.

НбФУ — небанківська фінансова установа.

ERP — Enterprise Resource Planning, система планування ресурсів підприємства.

CRM — Customer Relationship Management, система управління взаємовідносинами з клієнтами.

POS — Point of Sale, програмно-апаратна платформа для обліку продажів, платежів і торговельних операцій у місці обслуговування клієнта.

ПІБ — прізвище, ім’я, по батькові.

ERD — Entity Relationship Diagram, діаграма «сутність-зв’язок».

UML — Unified Modeling Language, уніфікована мова моделювання.

ORM — Object-Relational Mapping, об’єктно-реляційне відображення.

SQL — Structured Query Language, мова структурованих запитів.

HTTP — HyperText Transfer Protocol, протокол передавання гіпертексту.

URL — Uniform Resource Locator, уніфікований покажчик ресурсу в мережі Інтернет.

HTML — HyperText Markup Language, мова розмітки гіпертексту.

CSS — Cascading Style Sheets, каскадні таблиці стилів.

API — Application Programming Interface, програмний інтерфейс застосунку.

CRUD — Create, Read, Update, Delete, базові операції з даними.

ВСТУП

У сучасних умовах цифровізації фінансового сектору особливого значення набуває автоматизація процесів обліку та управління інформацією у небанківських фінансових установах. До таких установ належать фінансові компанії, кредитні спілки, ломбарди та інші організації, діяльність яких пов’язана з обробленням значних обсягів інформації про клієнтів, договори, фінансові операції та майнові об’єкти.

Одним із напрямів діяльності небанківських установ є надання фінансових послуг ломбардами. Особливістю ломбардної діяльності є необхідність одночасного ведення обліку клієнтів, заставного майна, договорів, платежів і строків виконання зобов’язань. За відсутності інформаційної системи такі процеси часто виконуються за допомогою паперових документів або електронних таблиць, що підвищує ризик помилок та ускладнює пошук необхідних даних.

Актуальність теми полягає у необхідності створення інформаційного застосунку управління операціями небанківської фінансової установи, який забезпечуватиме підтримку основних операцій з клієнтами, заставним майном, договорами, платежами та звітною інформацією. У межах роботи діяльність ломбарду розглядається як приклад небанківської фінансової установи, а автоматизація облікових дій є складовою ширшого завдання — організації та контролю операцій установи із використанням сучасних засобів програмування, СУБД та вебтехнологій.

Метою кваліфікаційної роботи є розроблення інформаційного застосунку управління операціями небанківської фінансової установи, який на прикладі ломбарду забезпечує підтримку основних операцій з клієнтами, заставним майном, договорами, платежами та звітною інформацією.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область небанківської фінансової установи на прикладі ломбарду;
- конкретизувати операції ломбарду, які потребують інформаційної підтримки;

- дослідити існуючі інформаційні застосунки для управління операціями ломбардів та споріднених небанківських установ;

- визначити функціональні та нефункціональні вимоги до застосунку;

- спроєктувати алгоритмічне та інформаційне забезпечення системи;

- розробити структуру бази даних і ERD-модель;

- реалізувати інформаційно-обчислювальний застосунок мовою Python із використанням Flask, SQLAlchemy та СУБД SQLite;

- провести тестування основних операцій застосунку та оцінити отримані результати.

Об’єктом дослідження є процеси управління операціями небанківської фінансової установи. Предметом дослідження є методи, моделі та програмні засоби розроблення інформаційно-обчислювального застосунку для підтримки операцій клієнтського обліку, заставного майна, договорів, платежів і звітності на прикладі ломбарду.

Практичне значення отриманих результатів полягає у створенні інформаційного застосунку, який може бути використаний як демонстраційний прототип системи управління операціями ломбарду та як основа для подальшого розширення функціональності небанківської фінансової установи.

Під час розроблення застосунку використовувалися мова програмування Python, фреймворк Flask, СУБД SQLite з можливістю подальшого переходу на PostgreSQL, ORM SQLAlchemy та сучасні вебтехнології для створення користувацького інтерфейсу.

Результати кваліфікаційної роботи апробовані та опубліковані в матеріалах V Всеукраїнська науково-практичної конференції Інтелектуальні комп’ютерні системи та мережі, Західноукраїнський національний університет, факультет комп’ютерних інформаційних технологій, кафедра комп'ютерної інженерії 20 травня (Додаток Г).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ НЕБАНКІВСЬКОЇ УСТАНОВИ

1.1 Опис операцій небанківської фінансової установи

Небанківські фінансові установи займають окреме місце у фінансовій системі, оскільки надають послуги, пов’язані з кредитуванням, платежами, страхуванням, фінансовим посередництвом та іншими операціями, не будучи банками. До таких установ належать фінансові компанії, кредитні спілки, страхові організації, ломбарди та інші суб’єкти, діяльність яких потребує належного обліку, контролю й оброблення інформації [1, 2].

У межах даної кваліфікаційної роботи як приклад небанківської установи розглядається ломбард. Ломбард здійснює надання короткострокових фінансових позик фізичним особам під заставу рухомого майна. Особливістю такої діяльності є необхідність одночасного обліку клієнтів, заставного майна, договорів, платежів, термінів повернення коштів та статусів виконання зобов’язань [1].

Операційна діяльність ломбарду як небанківської фінансової установи може бути подана як сукупність взаємопов’язаних операцій: ідентифікація клієнта, реєстрація предмета застави, попереднє оцінювання майна, визначення суми позики, формування договору, реєстрація платежів, контроль строків виконання зобов’язань і зміна статусу договору або застави. Саме ці операції потребують інформаційної підтримки з боку розроблюваного застосунку.

Для ломбарду важливо, щоб інформація зберігалася впорядковано та була доступною для швидкого пошуку. У разі використання паперових документів або розрізнених електронних таблиць виникають типові проблеми: дублювання даних, складність пошуку історії клієнта, ризик помилок під час розрахунків, відсутність єдиного контролю за договорами та платежами. Саме тому доцільним є створення застосунку управління операціями, який автоматизує основні дії працівника та забезпечує централізоване зберігання даних.

Інформаційний застосунок управління операціями ломбарду має забезпечувати роботу з такими основними інформаційними об’єктами: клієнти, користувачі системи, заставне майно, договори, платежі, статуси договорів, категорії заставного майна та звітна інформація.

У межах інформаційної системи сутностями вважаються основні об’єкти предметної області, дані про які зберігаються у базі даних. Для розроблюваного застосунку такими сутностями є клієнт, користувач, предмет застави, договір, платіж, категорія майна та статус. Клієнт є однією з ключових сутностей, оскільки саме з ним пов’язуються договори, застави та платежі. Для кожного клієнта необхідно зберігати персональні й контактні дані, а також історію взаємодії з установою.

Основними учасниками операцій у системі є адміністратор, працівник ломбарду та клієнт. Адміністратор відповідає за керування користувачами та контроль доступу до системи. Працівник ломбарду виконує операції з клієнтами, договорами, заставами й платежами. Клієнт безпосередньо не працює із системою, але його дані є ключовим об’єктом обліку.

Діяльність ломбарду доцільно описувати як сукупність операційних напрямів: робота з клієнтами, робота із заставним майном, оформлення договорів, облік платежів і формування звітності. Кожен операційний напрям має власний набір дій, але всі вони пов’язані спільною базою даних.

У процесі функціонування ломбарду формується значний обсяг даних, який повинен бути логічно структурований. Саме база даних є центральним елементом системи, оскільки вона забезпечує зберігання, цілісність і взаємозв’язок інформації. Для такої предметної області доцільно використовувати реляційну модель даних, оскільки вона дозволяє чітко описати зв’язки між клієнтами, договорами, заставами та платежами.

Інформаційний застосунок повинен підтримувати такі основні потоки даних: введення інформації про клієнта; додавання інформації про заставне майно; формування договору; реєстрація платежів; оновлення статусів договорів і застав; формування звітів для працівників установи.

Вхідною інформацією системи є персональні дані клієнтів, характеристики заставного майна, параметри договорів, суми платежів, дати операцій та облікові дані користувачів. Вихідною інформацією є сформовані записи договорів, звіти щодо активних і прострочених договорів, інформація про платежі, залишок заборгованості та стан заставного майна.

Для реалізації інформаційного застосунку доцільно використати веборієнтований підхід. Такий підхід дає змогу працювати із системою через браузер, спрощує оновлення інформаційно-обчислювального застосунку та дозволяє централізовано керувати базою даних. Для демонстраційного застосунку вебзастосунок є зручним варіантом, оскільки його можна продемонструвати на локальному сервері без складного розгортання.

Як основну мову програмування обрано Python. Ця мова є зручною для створення вебзастосунків, роботи з базами даних і реалізації бізнес-логіки. Для серверної частини доцільно використати Flask, оскільки він має просту структуру, не потребує надмірної конфігурації та підходить для невеликих і середніх інформаційних систем [5, 7].

Окремої уваги потребує питання якості даних, що вводяться у систему. Для ломбарду помилка у номері документа, сумі позики або даті завершення договору може призвести до некоректного обліку фінансових зобов’язань. Тому інформаційний застосунок повинен не лише зберігати дані, а й виконувати початкову перевірку правильності їх введення.

У практичній діяльності ломбарду важливим є швидкий доступ до історії взаємодії з клієнтом. Працівник повинен мати можливість побачити, які договори укладалися з клієнтом раніше, які предмети передавалися у заставу, чи були прострочення та які платежі виконувалися. Така інформація допомагає приймати більш обґрунтовані рішення під час оформлення нового договору.

Ще одним важливим аспектом є контроль статусів. Предмет застави може перебувати у різних станах: доступний для договору, у заставі, повернений клієнту або реалізований. Договір також може бути активним, закритим чи простроченим. Наявність уніфікованих статусів дозволяє уникати неоднозначності та спрощує формування звітів.

Предметна область ломбарду характеризується наявністю як транзакційної, так і аналітичної складової. Транзакційна складова пов’язана з щоденним введенням і зміною записів, а аналітична — з переглядом активних договорів, прострочень, платежів за період і структури заставного майна. Обидві складові мають бути враховані під час проєктування системи.

Розроблення інформаційного застосунку також повинно враховувати можливість подальшого розширення. Наприклад, до системи можна додати друк договорів, експорт звітів, завантаження фотографій заставного майна, журнал дій користувачів або інтеграцію з іншими сервісами. Тому структура бази даних і коду застосунку повинна бути достатньо гнучкою.

Основні операційні напрями ломбарду наведено у таблиці 1.1.

Таблиця 1.1 – Основні операційні напрями ломбарду як небанківської фінансової установи

Операційний напрям	Основні операції	Результат
Облік клієнтів	Реєстрація, редагування, пошук, перегляд історії	Картка клієнта та історія взаємодії
Облік заставного майна	Додавання предмета, опис, оцінювання, зміна статусу	Запис про предмет застави
Робота з договорами	Створення, продовження, закриття, контроль строків	Фінансове зобов’язання клієнта
Облік платежів	Реєстрація оплат, контроль боргу, історія платежів	Фінансова історія договору
Звітність	Формування списків активних і прострочених договорів	Узагальнена інформація для працівника

Отже, операційна діяльність ломбарду є достатньо структурованою для проєктування інформаційного застосунку управління операціями. Вона містить чіткі об’єкти даних, логічні зв’язки між ними та практичні операції, які можуть бути автоматизовані засобами Python, СУБД і реляційних баз даних.

1.2 Огляд і аналіз існуючих інформаційних застосунків

Для автоматизації діяльності ломбардів можуть використовуватися як спеціалізовані інформаційні системи, так і універсальні системи обліку, CRM або ERP-рішення. Спеціалізовані системи зазвичай орієнтовані саме на операції із заставним майном, договорами, платежами та звітністю. Універсальні системи мають ширші можливості, але часто потребують додаткового налаштування під конкретну предметну область.

Метою аналізу аналогів є не повне копіювання їхньої функціональності, а визначення типових можливостей, які мають бути враховані під час розроблення власного інформаційного застосунку. Для порівняння розглянуто спеціалізовані інформаційні застосунки, що використовуються у сфері автоматизації ломбардів.

Одним із прикладів інформаційно-обчислювального застосунку для ломбардів є Bravo Store Systems. Система позиціонується як POS-платформа для ломбардів і спеціалізованої роздрібної торгівлі, що підтримує облік позик, інвентаризацію, відповідність вимогам та управління операціями [16].

Іншим прикладом є PawnMate, який подається як хмарна платформа для ломбардів і магазинів, що працюють із викупом, продажем та комісійними товарами. На офіційному сайті акцент зроблено на швидкому виконанні транзакцій, роботі з кількома форматами ломбардного бізнесу та підтримці звітності [17].

Порівняння існуючих рішень показує, що сучасні системи автоматизації ломбардів зазвичай підтримують облік клієнтів, облік заставного майна, оформлення договорів, фінансові операції, пошук і фільтрацію даних, формування звітності, розмежування доступу користувачів і контроль історії операцій.

Разом з цим такі системи можуть мати недоліки для використання у межах невеликої установи або навчального застосунку. До таких недоліків належать висока вартість, закритий вихідний код, складність адаптації, залежність від постачальника, надлишкова функціональність або потреба у спеціальному налаштуванні.

Порівняльну характеристику розглянутих інформаційних застосунків наведено у таблиці 1.2 [16, 17].

Таблиця 1.2 – Порівняльна характеристика інформаційних застосунків

Критерій	Спеціалізовані системи	Універсальні CRM/ERP	Розроблюваний застосунок
Облік клієнтів	Підтримується	Підтримується після налаштування	Передбачено
Облік заставного майна	Підтримується	Потребує адаптації	Передбачено
Формування договорів	Підтримується	Потребує шаблонів	Передбачено

Продовження таблиці 1.2.

Критерій	Спеціалізовані системи	Універсальні CRM/ERP	Розроблюваний застосунок
Облік платежів	Підтримується	Частково	Передбачено
Можливість зміни структури БД	Обмежена	Обмежена або складна	Повністю контрольована
Відкритість коду	Зазвичай відсутня	Залежить від системи	Повна
Придатність для навчального проєкту	Обмежена	Обмежена	Висока

Аналіз аналогів показує, що для кваліфікаційної роботи доцільно створити власний інформаційний застосунок, який реалізує базову, але завершену функціональність. Такий підхід дозволяє не лише описати предметну область, а й практично реалізувати структуру бази даних, алгоритми оброблення інформації та інтерфейс користувача.

Власна система не повинна повторювати всі можливості промислових інформаційних систем. Її основне завдання полягає у демонстрації принципів побудови інформаційного застосунку управління операціями небанківської установи. Тому особливу увагу необхідно приділити не кількості функцій, а правильності структури даних, логіці взаємодії модулів, коректності фінансових розрахунків та можливості тестування.

Під час аналізу аналогів також враховано, що промислові системи часто містять функції, які виходять за межі бакалаврської роботи: інтеграцію з касовим обладнанням, комплексну бухгалтерію, багатофіліальну структуру, спеціальні модулі відповідності місцевому законодавству та розширену аналітику. Для навчального інформаційного застосунку доцільно реалізувати лише ядро предметної області.

У розроблюваному застосунку акцент зроблено на прозорості структури даних. Це означає, що кожна сутність предметної області має бути відображена окремою таблицею, а кожна операція повинна мати зрозумілий зв’язок з операцією. Такий підхід спрощує пояснення системи під час захисту та полегшує тестування.

Також важливо, щоб інформаційний застосунок мав відкриту і зрозумілу структуру коду. На відміну від готових комерційних систем, власна реалізація

дозволяє показати логіку роботи маршрутів, моделей бази даних, шаблонів інтерфейсу та алгоритмів оброблення даних.

1.3 Постановка задачі дослідження

На основі аналізу предметної області та існуючих інформаційних застосунків сформульовано задачу кваліфікаційної роботи: розробити інформаційний застосунок управління операціями небанківської установи на мові програмування Python для автоматизації обліку основних операцій ломбарду.

Розроблювана система повинна забезпечувати централізоване зберігання даних, зручну роботу з клієнтами, реєстрацію заставного майна, формування договорів, облік платежів і перегляд основної звітної інформації. Система має бути реалізована як демонстраційний застосунок, придатний для запуску на локальному комп’ютері та демонстрації під час захисту.

Метою розроблення є створення інформаційно-обчислювального застосунку, який дозволяє автоматизувати типові операції ломбарду та показує практичне використання Python для роботи з базою даних.

Функціональні вимоги до інформаційного застосунку включають авторизацію користувачів, облік клієнтів, облік заставного майна, формування договорів, реєстрацію платежів, пошук інформації та формування звітності. Для адміністратора передбачається керування користувачами, а для працівника ломбарду — виконання основних облікових операцій.

Модуль роботи з клієнтами призначений для додавання нових клієнтів, редагування даних, пошуку клієнта та перегляду пов’язаної з ним інформації. Для клієнта зберігаються прізвище, ім’я, по батькові, номер телефону, адреса, паспортні дані або інший ідентифікаційний документ.

Модуль заставного майна повинен забезпечувати реєстрацію предметів застави, визначення категорії, зазначення опису, оцінної вартості та поточного статусу. Один клієнт може мати кілька предметів застави, тому між клієнтом і заставним майном необхідно передбачити відповідний зв’язок у базі даних.

Модуль договорів реалізує створення договору на основі даних клієнта та заставного майна. У договорі фіксуються сума позики, відсоткова ставка, дата початку, дата завершення, статус договору та пов’язаний предмет застави. Договір є центральним елементом системи, оскільки він поєднує клієнта, заставу та платежі.

Модуль платежів повинен забезпечувати реєстрацію платежів за договором. Для кожного платежу зберігаються сума, дата, тип платежу та договір, до якого він належить. На основі платежів можна визначати фінансову історію договору.

Нефункціональні вимоги до системи включають надійність, зручність використання, безпеку, масштабованість і простоту супроводу. Надійність системи забезпечується використанням структурованої бази даних, первинних і зовнішніх ключів, перевіркою обов’язкових полів та коректною обробкою помилок.

Безпека системи передбачає зберігання паролів у хешованому вигляді, перевірку прав доступу та обмеження функцій для різних ролей користувачів. Оскільки система працює з персональними даними клієнтів, необхідно уникати відкритого зберігання паролів та неконтрольованого доступу до бази даних [12, 14, 15].

Для реалізації інформаційно-обчислювального застосунку обрано Python, Flask, SQLAlchemy, SQLite або PostgreSQL, HTML, CSS, Bootstrap, Jinja2 та Werkzeug Security. Використання Flask є доцільним, оскільки цей фреймворк дозволяє швидко створити робочий вебзастосунок із маршрутизацією, шаблонами, формами та підключенням до бази даних [5, 7, 8, 10-13].

Очікуваним результатом виконання роботи є демонстраційний інформаційний застосунок управління операціями небанківської фінансової установи, який реалізує базові процеси діяльності ломбарду як прикладу такої установи. Система повинна дозволяти створювати записи клієнтів, додавати заставне майно, формувати договори, реєструвати платежі та переглядати основну інформацію через вебінтерфейс.

Під час постановки задачі важливо визначити межі демонстраційної реалізації. У межах цієї роботи система не претендує на повну заміну промислового програмного комплексу, а реалізує основні операції, які демонструють принципи

побудови інформаційного застосунку управління операціями. Такий підхід відповідає практичному характеру кваліфікаційної роботи.

До основних обмежень демонстраційної версії можна віднести відсутність інтеграції з платіжними сервісами, електронним документообігом і зовнішніми державними реєстрами. Водночас ці обмеження не впливають на можливість перевірити основну логіку системи, структуру бази даних і роботу ключових модулів.

Очікується, що демонстраційний застосунок буде запускатися локально, а база даних створюватиметься автоматично під час ініціалізації застосунку. Це спрощує демонстрацію системи та дозволяє швидко перевірити роботу основних сценаріїв без складного серверного налаштування.

Під час розроблення також враховано вимоги академічної доброчесності. Теоретичні положення подаються з посиланнями на використані джерела, а реалізована частина застосунку побудована як власна реалізація, що демонструє розуміння принципів роботи вебзастосунку та бази даних.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна структура проєктованої системи

Інформаційний застосунок управління операціями небанківської фінансової установи призначений для автоматизації основних операцій, пов’язаних з обліком клієнтів, заставного майна, договорів і платежів. У межах кваліфікаційної роботи система розглядається на прикладі ломбарду, оскільки ця предметна область має чітко визначені інформаційні об’єкти та логічні зв’язки між ними.

Розроблювана система має веборієнтовану архітектуру. Це означає, що користувач взаємодіє із системою через веббраузер, а основна бізнес-логіка виконується на серверній частині застосунку. Дані зберігаються у базі даних, до якої звертається серверна частина через спеціальний програмний рівень доступу до даних.

Загальна структура системи містить користувацький інтерфейс, серверну частину, модуль бізнес-логіки, модуль роботи з базою даних, базу даних, модуль авторизації та модуль звітності. Користувацький інтерфейс забезпечує введення, перегляд і редагування інформації. Через нього працівник установи може додавати клієнтів, створювати записи про заставне майно, оформлювати договори, реєструвати платежі та переглядати звітну інформацію.

Серверна частина відповідає за оброблення запитів користувача, перевірку введених даних, виконання бізнес-правил і взаємодію з базою даних. У цій роботі для реалізації серверної частини використовується мова програмування Python та вебфреймворк Flask.

Модуль бізнес-логіки містить правила, за якими система виконує основні операції. До таких правил належать перевірка коректності даних клієнта, формування договору, визначення статусу договору, розрахунок платежів і перевірка строків виконання зобов’язань.

Модуль роботи з базою даних забезпечує створення, читання, оновлення та видалення записів. Для цього використовується SQLAlchemy, що дозволяє працювати з таблицями бази даних через класи Python.

База даних є центральним елементом системи. У ній зберігаються відомості про клієнтів, користувачів, договори, заставне майно, платежі та статуси. Структура бази даних повинна забезпечувати цілісність інформації та мінімізувати дублювання даних.

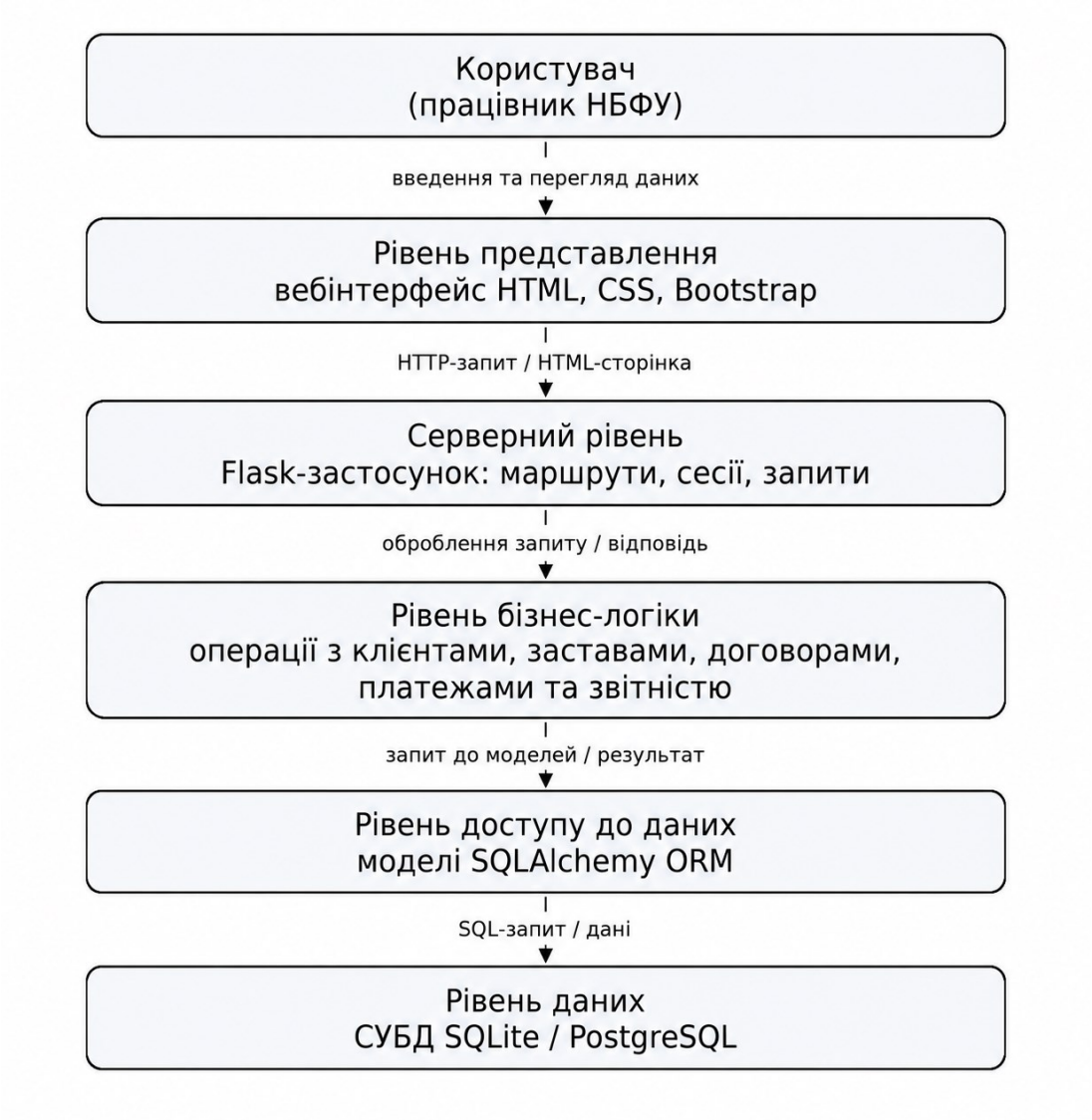

Рисунок 2.1 – Загальна архітектура інформаційного застосунку управління операціями

Функціональна структура інформаційного застосунку складається з окремих підсистем, кожна з яких відповідає за певний напрям роботи: підсистема авторизації користувачів, підсистема обліку клієнтів, підсистема обліку заставного

майна, підсистема роботи з договорами, підсистема обліку платежів, підсистема звітності та підсистема адміністрування.

Ролі користувачів проєктованої системи наведено у таблиці 2.1.

Таблиця 2.1 – Ролі користувачів системи

Роль користувача	Основні можливості
Адміністратор	Керування користувачами, перегляд усіх даних, адміністрування системи
Працівник ломбарду	Робота з клієнтами, заставами, договорами, платежами та звітами

Архітектура інформаційно-обчислювального застосунку побудована за принципом поділу системи на три логічні рівні: рівень представлення, рівень бізнес-логіки та рівень даних. Такий підхід дозволяє спростити розроблення, тестування та подальше розширення системи.

2.2 Алгоритмічне забезпечення

Алгоритмічне забезпечення системи визначає послідовність дій, які виконуються під час основних операцій. Для розроблюваного інформаційного застосунку важливо описати алгоритми авторизації, реєстрації клієнта, створення договору, реєстрації платежу та оновлення статусу договору.

Авторизація є початковим етапом роботи із системою. Користувач відкриває сторінку входу, вводить логін і пароль, після чого система перевіряє наявність користувача у базі даних і порівнює введений пароль із хешованим значенням. Якщо дані правильні, створюється сеанс користувача, і він переходить на головну сторінку системи.

Реєстрація клієнта виконується перед створенням договору або додаванням заставного майна. Працівник відкриває сторінку додавання клієнта, вводить ПІБ, телефон, адресу та документ клієнта. Система перевіряє заповнення обов’язкових полів і унікальність номера документа. Якщо дані коректні, створюється новий запис у таблиці клієнтів.

Додавання заставного майна виконується з прив’язкою до конкретного клієнта. Працівник обирає клієнта, вводить назву предмета, категорію, опис і оцінну вартість. Система перевіряє коректність введених даних і створює запис у таблиці заставного майна.

Договір є центральним об’єктом системи. Під час його створення працівник обирає клієнта і предмет застави, вводить суму позики, відсоткову ставку, дату початку та дату завершення договору. Система перевіряє коректність дат і фінансових значень, після чого створює договір і змінює статус заставного майна.

Платіж відображає фінансову операцію за договором. Працівник відкриває сторінку договору, додає суму платежу, дату та тип платежу. Після перевірки даних запис зберігається у базі, а історія платежів за договором оновлюється.

Контроль строків договору необхідний для виявлення прострочених зобов’язань. Система отримує поточну дату, вибирає активні договори та порівнює дату завершення договору з поточною датою. Якщо строк завершився, статус договору змінюється на «прострочений».

Алгоритм авторизації має забезпечувати не лише перевірку логіна та пароля, а й захист від доступу до сторінок без активного сеансу. Тому кожен маршрут, пов’язаний з даними клієнтів, договорів або платежів, повинен перевіряти наявність ідентифікатора користувача у сесії.

Алгоритм створення договору містить кілька перевірок, які мають практичне значення. По-перше, система повинна перевірити наявність клієнта і предмета застави. По-друге, дата завершення договору повинна бути пізнішою за дату початку. По-третє, сума позики не повинна бути нульовою або від’ємною.

Алгоритм реєстрації платежу також передбачає перевірку коректності введеної суми. Якщо користувач випадково вводить нуль або від’ємне значення, система повинна відхилити таку операцію. Це запобігає появі некоректних фінансових записів у базі даних.

Для контролю прострочених договорів може використовуватися періодична перевірка дат завершення. У демонстраційній версії така перевірка виконується під час відкриття сторінки звіту, але у промисловій системі її доцільно реалізувати як фонове завдання, що запускається за розкладом.

Алгоритмічне забезпечення повинно бути достатньо простим для реалізації, але повним з погляду основних бізнес-сценаріїв. Тому в роботі описано саме ті алгоритми, які безпосередньо використовуються у програмному модулі та можуть бути перевірені під час тестування.

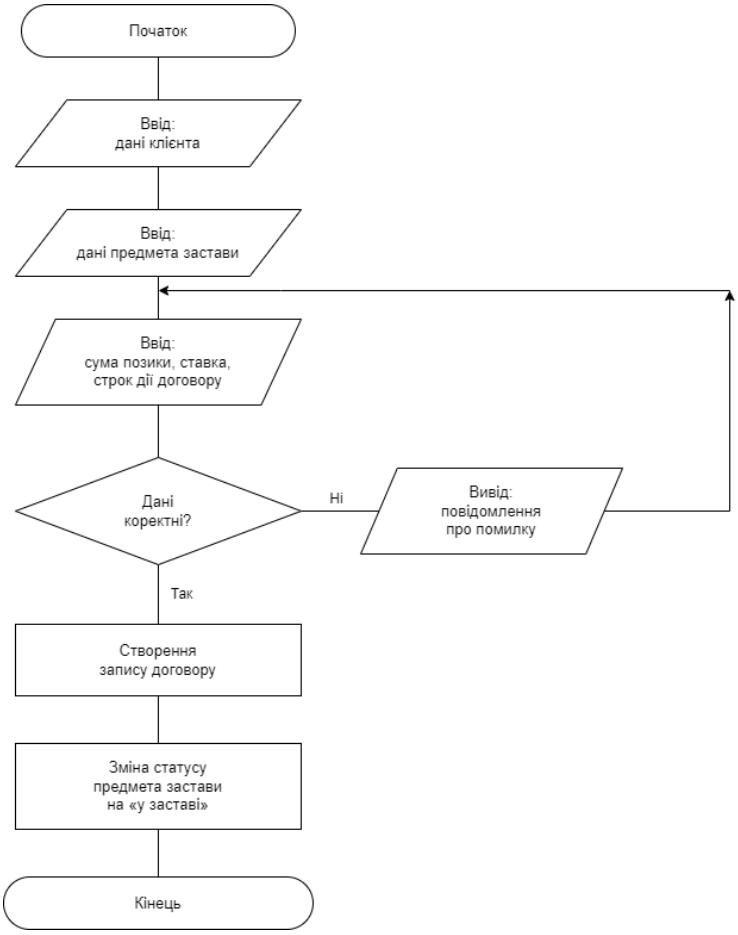

Рисунок 2.2 – Блок-схема алгоритму створення договору

2.3 Інформаційне забезпечення

Інформаційне забезпечення є однією з основних складових інформаційного застосунку управління операціями небанківської установи. Воно визначає склад даних, що використовуються в системі, їхню структуру, взаємозв’язки, способи зберігання та оброблення.

У межах розроблюваного застосунку інформаційне забезпечення формується на основі аналізу предметної області ломбарду. Основними інформаційними об’єктами системи є клієнти, користувачі, заставне майно, договори, платежі,

категорії майна та статуси договорів. Кожен із цих об’єктів має власний набір атрибутів і пов’язаний з іншими об’єктами предметної області.

База даних повинна забезпечувати зберігання структурованої інформації, швидкий пошук необхідних записів, підтримку зв’язків між таблицями, цілісність і несуперечність даних, можливість розширення структури та захист від втрати або дублювання інформації.

Для реалізації інформаційного забезпечення доцільно використати реляційну модель даних. Такий підхід дозволяє подати предметну область у вигляді таблиць, між якими встановлюються зв’язки за допомогою первинних і зовнішніх ключів [19, 20].

Вхідну інформацію системи наведено у таблиці 2.2.

Таблиця 2.2 – Вхідна інформація системи

Назва інформації	Джерело надходження	Призначення
Дані користувача	Форма авторизації	Вхід до системи
Дані клієнта	Форма клієнта	Реєстрація та облік клієнтів
Дані заставного майна	Форма застави	Облік предметів застави
Дані договору	Форма договору	Створення фінансового зобов’язання
Дані платежу	Форма платежу	Реєстрація фінансової операції
Параметри пошуку	Поля фільтрації	Пошук інформації у базі даних

Вихідну інформацію системи наведено у таблиці 2.3.

Таблиця 2.3 – Вихідна інформація системи

Назва інформації	Форма подання	Призначення
Список клієнтів	Таблиця	Перегляд і пошук клієнтів

Продовження таблиці 2.3.

Назва інформації	Форма подання	Призначення
Картка клієнта	Вебсторінка	Перегляд детальної інформації
Список застав	Таблиця	Контроль заставного майна
Список договорів	Таблиця	Облік фінансових зобов’язань
Історія платежів	Таблиця	Контроль фінансових операцій
Звіт активних договорів	Таблиця	Аналіз поточних договорів
Звіт прострочених договорів	Таблиця	Контроль проблемних операцій

Концептуальне проєктування бази даних передбачає визначення основних сутностей предметної області, їхніх атрибутів і зв’язків між ними. Для інформаційного застосунку управління операціями небанківської установи визначено сутності User, Client, PawnItem, Contract, Payment, Category та Status [18, 19].

Основні сутності бази даних наведено у таблиці 2.4.

Таблиця 2.4 – Основні сутності бази даних

Сутність	Призначення
User	Зберігання даних користувачів системи
Client	Зберігання інформації про клієнтів
PawnItem	Облік заставного майна
Contract	Облік договорів
Payment	Облік платежів
Category	Класифікація заставного майна
Status	Визначення стану договорів і застав

На основі визначених сутностей та зв'язків між ними побудовано концептуальну модель бази даних у вигляді ERD-діаграми. Структуру бази даних і взаємозв'язки між таблицями наведено на рисунку 2.3

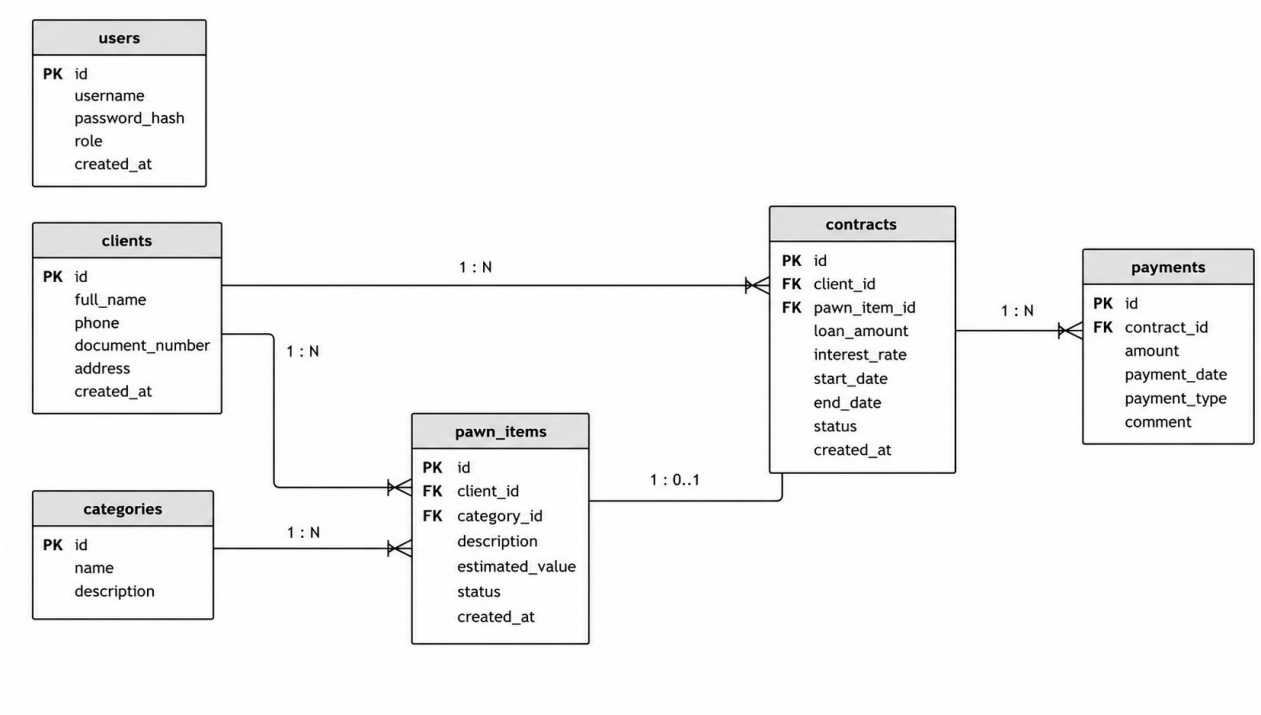

Рисунок 2.3 – ERD-модель бази даних застосунку управління операціями

Основні зв’язки між сутностями побудовані відповідно до логіки роботи ломбарду. Один клієнт може мати багато предметів заставного майна та багато договорів. Один предмет застави може бути пов’язаний з одним договором. Один договір може мати багато платежів. Одна категорія може використовуватися для багатьох предметів застави.

Структуру таблиці clients наведено у таблиці 2.5.

Таблиця 2.5 – Структура таблиці clients

Поле	Тип даних	Призначення
id	integer	Первинний ключ
full_name	varchar	ПІБ клієнта
phone	varchar	Номер телефону
document_number	varchar	Номер документа
address	varchar	Адреса проживання
created_at	datetime	Дата реєстрації

Структуру таблиці pawn_items наведено у таблиці 2.6.

Таблиця 2.6 – Структура таблиці pawn_items

Поле	Тип даних	Призначення
id	integer	Первинний ключ
client_id	integer	Зовнішній ключ на clients
category_id	integer	Зовнішній ключ на categories
name	varchar	Назва предмета застави
description	text	Опис предмета
estimated_value	numeric	Оцінна вартість
status	varchar	Поточний статус

Структуру таблиці contracts наведено у таблиці 2.7.

Таблиця 2.7 – Структура таблиці contracts

Поле	Тип даних	Призначення
id	integer	Первинний ключ
client_id	integer	Зовнішній ключ на clients
pawn_item_id	integer	Зовнішній ключ на pawn_items
loan_amount	numeric	Сума позики
interest_rate	numeric	Відсоткова ставка
start_date	date	Дата початку договору
end_date	date	Дата завершення договору
status	varchar	Статус договору

Структуру таблиці payments наведено у таблиці 2.8.

Таблиця 2.8 – Структура таблиці payments

Поле	Тип даних	Призначення
id	integer	Первинний ключ
contract_id	integer	Зовнішній ключ на contracts
amount	numeric	Сума платежу
payment_date	date	Дата платежу
payment_type	varchar	Тип платежу
comment	text	Коментар до платежу

Фізична модель бази даних визначає спосіб реалізації даталогічної моделі у конкретній системі керування базами даних. Для реалізації інформаційного застосунку може використовуватися PostgreSQL, а для спрощеної демонстраційної версії — SQLite [9, 10].

У фізичній моделі необхідно забезпечити створення таблиць, визначення типів даних, встановлення первинних і зовнішніх ключів, обмеження цілісності та індексування полів для швидкого пошуку. Індекси доцільно створити для полів clients.full_name, clients.phone, contracts.status, contracts.end_date та payments.payment_date.

Для забезпечення цілісності даних у базі використовуються первинні та зовнішні ключі. Первинний ключ однозначно ідентифікує кожен запис таблиці, а зовнішній ключ встановлює зв’язок між записами різних таблиць. Наприклад, договір не може існувати без клієнта, а платіж не може існувати без договору.

Окремо враховано питання нормалізації даних. Категорії заставного майна винесено в окрему таблицю, що дозволяє уникнути повторення однакових назв категорій у таблиці предметів застави. Такий підхід зменшує дублювання та спрощує редагування довідкової інформації.

У практичній системі також доцільно передбачити резервне копіювання бази даних, журналювання важливих дій користувачів і механізми відновлення після помилок. У межах демонстраційного застосунку ці функції можуть бути описані як перспективи розвитку, але структура системи повинна дозволяти їх додавання у майбутньому.

Для реалізації бази даних використовується ORM SQLAlchemy. ORM дозволяє описати таблиці бази даних у вигляді класів Python. Це спрощує роботу з даними та зменшує кількість SQL-коду, який необхідно писати вручну [8].

Під час проєктування бази даних необхідно забезпечити мінімізацію дублювання даних і логічну узгодженість структури. Запропонована структура приводиться до третьої нормальної форми, оскільки сутності клієнтів, застав, договорів, платежів і категорій винесені в окремі таблиці, а зв’язки між ними реалізовані зовнішніми ключами.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Реалізація інформаційно-обчислювального застосунку

Реалізація інформаційного застосунку управління операціями небанківської установи виконується мовою програмування Python. Для створення вебзастосунку обрано фреймворк Flask, який дозволяє реалізувати серверну частину системи, маршрути оброблення запитів, шаблони сторінок і взаємодію з базою даних [5, 7].

Розроблюваний інформаційно-обчислювальний застосунок має забезпечувати виконання основних функцій, визначених у попередніх розділах: авторизацію користувачів, роботу з клієнтами, облік заставного майна, створення договорів, реєстрацію платежів і перегляд звітної інформації.

Інформаційний застосунок реалізується як вебзастосунок, який запускається на локальному сервері. Користувач відкриває систему у браузері та працює з нею через вебінтерфейс. Такий варіант є зручним для демонстрації, тестування та подальшого розширення.

Для реалізації інформаційного застосунку використано Python, Flask, SQLAlchemy, SQLite або PostgreSQL, HTML, CSS, Bootstrap, Jinja2 і Werkzeug Security. Мова Python обрана через простоту синтаксису, наявність великої кількості бібліотек і зручність роботи з базами даних [5, 7, 8, 10-13].

Фреймворк Flask використовується для створення серверної частини вебзастосунку. Його перевагою є невеликий обсяг базової конфігурації та можливість поступово додавати потрібні модулі. Flask дозволяє швидко створити маршрути, обробляти HTTP-запити, передавати дані у шаблони та організувати структуру застосунку [7].

Бібліотека SQLAlchemy використовується для взаємодії з базою даних. Вона дозволяє описувати таблиці у вигляді класів Python і виконувати операції з даними без прямого написання великої кількості SQL-запитів [8].

Для демонстраційної версії може використовуватися SQLite, оскільки ця система керування базами даних не потребує окремого серверного налаштування. Для повноцінного впровадження доцільніше використовувати PostgreSQL,

оскільки вона краще підходить для багатокористувацьких систем і забезпечує вищий рівень надійності [9, 10].

Розроблена структура файлів інформаційно-обчислювального застосунку має такий вигляд:

```
pawnshop_module/
│
├── app.py
├── config.py
├── models.py
├── requirements.txt
│
├── templates/
│   ├── base.html
│   ├── login.html
│   ├── dashboard.html
│   ├── clients.html
│   ├── client_form.html
│   ├── pawn_items.html
│   ├── contracts.html
│   ├── contract_form.html
│   ├── payments.html
│   └── reports.html
│
├── static/
│   └── css/style.css
└── instance/
    └── database.db
```

Файл app.py є головним файлом запуску застосунку. У ньому створюється Flask-застосунок, підключаються маршрути, база даних і додаткові компоненти. Файл models.py містить описи моделей бази даних, а папка templates містить HTML-шаблони сторінок.

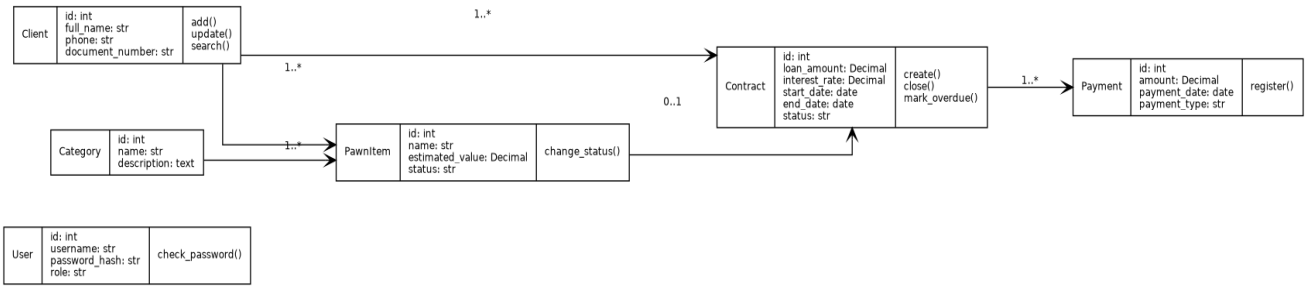

Рисунок 3.1 – UML-діаграма класів програмного модуля

Інформаційний застосунок складається з кількох логічних підсистем: авторизації, роботи з клієнтами, роботи із заставним майном, договорів, платежів та звітності. Така структура спрощує супровід коду та дозволяє розділити логіку застосунку на окремі функціональні частини.

Підсистема авторизації відповідає за вхід користувачів до системи. Вона перевіряє логін і пароль, створює сеанс користувача та обмежує доступ до сторінок для неавторизованих осіб. Для підвищення безпеки паролі зберігаються у вигляді хешів за допомогою інструментів Werkzeug Security.

Підсистема роботи з клієнтами дозволяє додавати нових клієнтів, редагувати дані, переглядати список клієнтів і здійснювати пошук. Підсистема заставного майна використовується для обліку предметів, які клієнти передають у заставу. Кожен предмет застави пов’язаний із конкретним клієнтом.

Підсистема договорів реалізує створення та облік договорів між клієнтом і небанківською фінансовою установою. Договір пов’язує клієнта, предмет застави та фінансові умови. Підсистема платежів забезпечує облік фінансових операцій за договорами, а підсистема звітності забезпечує перегляд узагальненої інформації.

Фрагменти програмного коду моделей бази даних і маршрутів Flask наведено у додатку А.

Під час реалізації моделей бази даних кожна таблиця описується окремим класом. Такий підхід полегшує читання коду, оскільки структура бази даних стає зрозумілою без перегляду SQL-скриптів. Наприклад, клас Client відповідає таблиці clients, а його атрибути описують поля цієї таблиці.

Важливою перевагою SQLAlchemy є можливість опису зв’язків між моделями. Завдяки цьому можна отримати всі договори конкретного клієнта або всі платежі за певним договором без ручного написання складних SQL-запитів. Це зменшує ймовірність помилок і прискорює розроблення.

Під час створення маршрутів Flask використовується принцип поділу операцій за URL-адресами. Наприклад, маршрут /clients відповідає за перегляд клієнтів, /clients/add — за додавання нового клієнта, /contracts — за перегляд договорів, а /payments/add — за додавання платежу. Така структура є зрозумілою та легко розширюється.

Для зручності користувача після виконання операцій застосовуються повідомлення. Якщо запис успішно додано, система повідомляє про це користувача. Якщо під час введення даних виникла помилка, система не зберігає некоректний запис, а показує відповідне попередження.

У демонстраційній реалізації основна увага приділяється простоті запуску та перевірки. Тому SQLite використовується як базова СУБД для локального запуску. При потребі система може бути перенесена на PostgreSQL шляхом зміни рядка підключення та виконання міграції структури бази даних.

3.2 Інтерфейс користувача

Користувацький інтерфейс є важливою частиною інформаційного застосунку, оскільки саме через нього працівник взаємодіє із системою. Інтерфейс повинен бути зрозумілим, логічним і зручним для виконання основних операцій.

Під час розроблення інтерфейсу враховано такі принципи: простота навігації, зрозумілі назви сторінок і кнопок, мінімальна кількість зайвих елементів, табличне подання основних даних, використання форм для введення інформації, відображення повідомлень про помилки та успішні дії.

Інтерфейс системи реалізовано за допомогою HTML, CSS, Bootstrap і шаблонізатора Jinja2. Bootstrap дозволяє створити адаптивний вигляд сторінок без складного ручного написання стилів [11, 13, 25, 26].

Для всіх сторінок використовується базовий шаблон base.html. Він містить загальну структуру HTML-документа, підключення Bootstrap, навігаційне меню та блок для відображення основного вмісту сторінки. Це дозволяє уникнути дублювання HTML-коду та забезпечити єдиний стиль усіх сторінок.

Сторінка авторизації використовується для входу користувача до системи. Вона містить форму з двома полями: логін і пароль. Після введення даних система перевіряє логін і пароль. У разі успішної авторизації користувач переходить на головну сторінку.

Головна сторінка системи призначена для швидкого перегляду загальної статистики. На ній відображається кількість клієнтів, договорів, активних і прострочених договорів. Головна сторінка виконує роль інформаційної панелі та дозволяє користувачу швидко перейти до потрібного розділу системи.

Приклад розробленого інтерфейсу користувача наведено на рисунку 3.2.

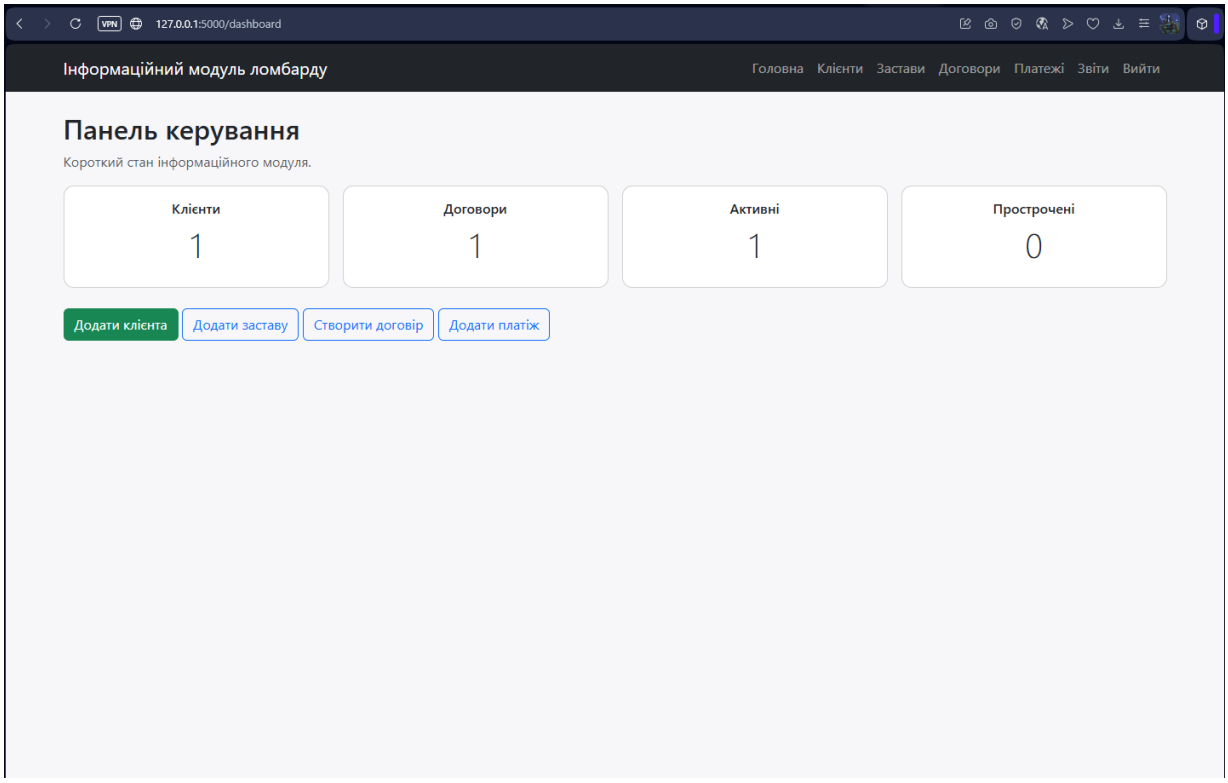

Рисунок 3.2 – Приклад інтерфейсу користувача розробленого застосунку

Сторінка обліку клієнтів містить таблицю зі списком клієнтів і форму пошуку. Користувач може швидко знайти клієнта за ПІБ. Табличне подання даних є зручним для облікових систем, оскільки дозволяє швидко переглядати багато записів.

Сторінка створення договору містить форму, у якій працівник обирає клієнта, предмет застави, суму позики, відсоткову ставку та строки дії договору. Форма створення договору є одним із ключових елементів інтерфейсу, оскільки через неї формується основний фінансовий запис системи.

Сторінка обліку платежів призначена для перегляду фінансових операцій, які виконуються за договорами. На цій сторінці користувач може побачити дату платежу, договір, суму, тип операції та коментар.

Сторінка звітів призначена для перегляду узагальненої інформації про роботу небанківської установи. У межах демонстраційної версії системи реалізовано звіт активних договорів, звіт прострочених договорів, звіт платежів за період і звіт заставного майна.

Під час створення інтерфейсу було враховано, що користувачами системи можуть бути працівники небанківської установи, які не мають глибоких технічних знань. Тому інтерфейс має бути простим, зрозумілим і не перевантаженим зайвими елементами.

Під час проєктування інтерфейсу важливо забезпечити однаковий підхід до розміщення елементів керування. Кнопки додавання нових записів розміщуються у верхній частині сторінки, таблиці займають основну частину робочої області, а поля пошуку розміщуються перед списками даних.

Форми введення повинні містити лише ті поля, які необхідні для виконання конкретної операції. Надмірна кількість полів ускладнює роботу користувача та підвищує ймовірність помилки. Тому для демонстраційної системи обрано мінімальний набір полів, достатній для опису клієнта, застави, договору та платежу.

Табличне подання даних дозволяє швидко порівнювати записи між собою. Наприклад, у списку договорів працівник може одразу побачити клієнта, предмет застави, суму позики, дату завершення та статус. Це спрощує щоденну роботу з інформацією.

Система також повинна бути зрозумілою під час демонстрації. Для цього сторінки мають логічні назви, а основні дії виконуються через очевидні кнопки: «Додати клієнта», «Створити договір», «Додати платіж», «Сформувати звіт». Такий підхід покращує сприйняття програмного продукту під час захисту.

Основні сторінки користувацького інтерфейсу наведено у таблиці 3.1.

Таблиця 3.1 – Основні сторінки користувацького інтерфейсу

Сторінка	Призначення
login.html	Вхід користувача до системи
dashboard.html	Перегляд загальної статистики
clients.html	Робота зі списком клієнтів
pawn_items.html	Облік предметів застави
contracts.html	Перегляд і створення договорів
payments.html	Реєстрація платежів
reports.html	Формування звітної інформації

Для повідомлення користувача про результат виконання операцій використовуються flash-повідомлення Flask. Наприклад, після успішного додавання клієнта система відображає повідомлення «Клієнта успішно додано». Якщо користувач не заповнив обов’язкові поля, система повідомляє про помилку [7].

3.3 Тестування застосунку управління операціями

Тестування застосунку управління операціями є необхідним етапом розроблення, оскільки воно дозволяє перевірити правильність виконання операцій, виявити помилки та оцінити відповідність реалізованого функціоналу поставленим вимогам.

У межах кваліфікаційної роботи тестування проводиться для основних операцій застосунку: авторизація користувачів, додавання клієнта, додавання заставного майна, створення договору, реєстрація платежу, формування звітів і перевірка обмежень введення даних.

Метою тестування є перевірка того, що система коректно виконує основні операції та правильно реагує на помилкові або неповні дані. Для перевірки працездатності інформаційного застосунку використовується функціональне тестування. Цей вид тестування передбачає перевірку окремих функцій системи з погляду користувача.

Тестування виконується вручну через вебінтерфейс системи. Такий підхід є достатнім для демонстраційної версії інформаційно-обчислювального застосунку, оскільки дозволяє перевірити основні сценарії роботи користувача.

План тестування основних операцій застосунку наведено на рисунку 3.3.

Рисунок 3.3 – План тестування застосунку управління операціями

Результати тестування авторизації користувача наведено у таблиці 3.2.

Таблиця 3.2 – Тестування авторизації користувача

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Вхід з правильними даними	Коректний логін і пароль	Перехід на головну сторінку	Успішно
Вхід з неправильним паролем	Коректний логін, неправильний пароль	Повідомлення про помилку	Успішно
Вхід з порожніми полями	Порожній логін і пароль	Повідомлення про помилку	Успішно
Перехід без входу	Неавторизований користувач	Перенаправлення на сторінку входу	Успішно

Результати тестування модуля клієнтів наведено у таблиці 3.3.

Таблиця 3.3 – Тестування модуля клієнтів

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Додавання нового клієнта	ПІБ, телефон, документ, адреса	Клієнт збережений у базі	Успішно
Додавання клієнта без ПІБ	Телефон і документ без ПІБ	Повідомлення про помилку	Успішно
Додавання клієнта без документа	ПІБ і телефон без документа	Повідомлення про помилку	Успішно
Повторний документ	Документ, який уже існує	Повідомлення про дублювання	Успішно
Пошук клієнта за ПІБ	Частина прізвища	Відображення знайдених записів	Успішно

Результати тестування модуля заставного майна наведено у таблиці 3.4.

Таблиця 3.4 – Тестування модуля заставного майна

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Додавання предмета застави	Клієнт, категорія, назва, вартість	Предмет збережений у базі	Успішно
Додавання застави без клієнта	Дані майна без клієнта	Повідомлення про помилку	Успішно
Додавання застави без вартості	Назва і категорія без вартості	Повідомлення про помилку	Успішно
Додавання застави з нульовою вартістю	Вартість 0	Повідомлення про помилку	Успішно

Результати тестування модуля договорів наведено у таблиці 3.5.

Таблиця 3.5 – Тестування модуля договорів

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Створення договору	Клієнт, застава, сума, ставка, дати	Договір створено	Успішно
Створення без клієнта	Дані договору без клієнта	Повідомлення про помилку	Успішно
Некоректна дата завершення	Дата завершення менша за дату початку	Повідомлення про помилку	Успішно
Нульова сума позики	Сума позики 0	Повідомлення про помилку	Успішно
Зміна статусу застави	Створення договору із заставою	Статус змінено на «у заставі»	Успішно

Результати тестування модуля платежів наведено у таблиці 3.6.

Таблиця 3.6 – Тестування модуля платежів

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Додавання платежу	Договір, сума, дата, тип	Платіж збережено	Успішно
Платіж без договору	Сума і дата без договору	Повідомлення про помилку	Успішно
Платіж без суми	Договір і дата без суми	Повідомлення про помилку	Успішно
Нульова сума платежу	Сума 0	Повідомлення про помилку	Успішно
Історія платежів	Наявні платежі за договором	Відображення списку платежів	Успішно

Результати тестування модуля звітності наведено у таблиці 3.7.

Таблиця 3.7 – Тестування модуля звітності

Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат
Перегляд активних договорів	Наявні активні договори	Відображення активних договорів	Успішно
Перегляд прострочених договорів	Договори з минулою датою завершення	Відображення прострочених договорів	Успішно
Звіт платежів за період	Дата початку і завершення	Відображення платежів у межах періоду	Успішно
Звіт без даних	Період без платежів	Порожня таблиця без помилки	Успішно

Окремо було перевірено реакцію системи на типові помилки користувача: порожні поля, повторний номер документа, некоректну дату завершення договору, нульову суму платежу та спробу доступу без авторизації. У кожному випадку система не зберігає помилкові дані, а повідомляє користувача про проблему.

Результати ручного функціонального тестування дозволили переконатися, що основні сценарії роботи виконуються у правильній послідовності. Після створення договору предмет застави змінює статус, після реєстрації платежу він з’являється в історії договору, а прострочені договори відображаються у відповідному звіті.

Для подальшого розвитку системи доцільно додати автоматизовані модульні тести. Вони можуть перевіряти роботу окремих функцій, моделей бази даних і маршрутів Flask без ручного введення даних через вебінтерфейс. Це підвищить надійність системи під час її розширення.

За результатами тестування встановлено, що розроблений інформаційний модуль виконує основні функції, визначені під час постановки задачі. Система дозволяє авторизувати користувача, вести облік клієнтів, додавати предмети застави, створювати договори, реєструвати платежі та формувати базові звіти.

ВИСНОВКИ

1. У кваліфікаційній роботі розв’язано задачу розроблення інформаційного застосунку управління операціями небанківської фінансової установи. Автоматизація облікових дій ломбарду розглянута як підпорядкована практична задача, що входить до ширшої системи підтримки операцій з клієнтами, заставним майном, договорами, платежами та звітністю.

2. Проведено аналіз предметної області небанківської фінансової установи на прикладі ломбарду. Операційну діяльність ломбарду подано як сукупність взаємопов’язаних операцій, що охоплюють ідентифікацію клієнта, реєстрацію застави, оцінювання майна, оформлення договору, облік платежів і контроль строків виконання фінансових зобов’язань.

3. Розроблено алгоритмічне та інформаційне забезпечення застосунку. Визначено загальну архітектуру системи, ролі користувачів, основні алгоритми виконання операцій, склад вхідної та вихідної інформації, структуру реляційної бази даних, ERD-модель і зв’язки між сутностями.

4. Реалізовано інформаційно-обчислювальний застосунок мовою Python із використанням Flask, SQLAlchemy, СУБД SQLite, HTML, CSS, Bootstrap і Jinja2. Застосунок забезпечує авторизацію користувача, роботу з клієнтами, заставним майном, договорами, платежами та звітною інформацією через вебінтерфейс.

5. Проведено тестування основних операцій застосунку. Перевірено авторизацію, додавання клієнтів, реєстрацію заставного майна, створення договорів, внесення платежів, формування звітів і реакцію системи на некоректні дані. Результати тестування підтвердили працездатність застосунку та можливість його використання як демонстраційного прототипу для подальшого розвитку системи управління операціями небанківської фінансової установи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про фінансові послуги та фінансові компанії : Закон України від 14.12.2021 № 1953-IX. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/1953-20 (дата звернення: 20.04.2026).

2. Національний банк України. Реєстри та переліки небанківських фінансових установ. URL: https://bank.gov.ua/ua/supervision/nonbanks/registers-lists (дата звернення: 20.04.2026).

3. Національний банк України. Ліцензування небанківських установ. URL: https://bank.gov.ua/ua/supervision/licensing-nonbanking/form (дата звернення: 20.04.2026).

4. Python Software Foundation. Python 3.12 Documentation. URL: https://docs.python.org/uk/3.12/index.html (дата звернення: 20.04.2026).

5. Python Software Foundation. sqlite3 — DB-API 2.0 interface for SQLite databases. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 20.04.2026).

6. Pallets Projects. Flask Documentation. Version 3.1.x. URL: https://flask.palletsprojects.com/ (дата звернення: 20.04.2026).

7. SQLAlchemy. SQLAlchemy Documentation. URL: https://docs.sqlalchemy.org/ (дата звернення: 20.04.2026).

8. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 20.04.2026).

9. SQLite. SQLite Documentation. URL: https://sqlite.org/docs.html (дата звернення: 20.04.2026).

10. Pallets Projects. Jinja Documentation. Version 3.1.x. URL: https://jinja.palletsprojects.com/ (дата звернення: 20.04.2026).

11. Pallets Projects. Werkzeug Documentation. Version 3.1.x. URL: https://werkzeug.palletsprojects.com/ (дата звернення: 20.04.2026).

12. Bootstrap Team. Bootstrap Documentation. Version 5.3. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 20.04.2026).

13. OWASP Foundation. Application Security Verification Standard. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 20.04.2026).

14. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 20.04.2026).

15. Bravo Store Systems. Pawn Shop POS Software. URL: https://www.bravostoresystems.com/point-of-sale-for-pawnbrokers (дата звернення: 20.04.2026).

16. PawnMate. Pawnshop Software Features and Benefits. URL: https://www.pawnmate.com/features-and-benefits/ (дата звернення: 20.04.2026).

17. Chen P. P. The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems. 1976. Vol. 1, No. 1. P. 9–36.

18. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Pearson, 2016. 1272 p.

19. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill Education, 2019. 1344 p.

20. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley Professional, 2002. 560 p.

21. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York: McGraw-Hill Education, 2019. 880 p.

22. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2015. 816 p.

23. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.

24. Mozilla Developer Network. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 20.04.2026).

25. Mozilla Developer Network. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 20.04.2026).

26. Комар М. П., Саченко А. О., Васильків Н. М., Гладій Г. М., Коваль В. С., Ліп’яніна-Гончаренко Х. В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122

«Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

27. Микитюк В.В. Інформаційний застосунок управління операціями небанківської фінансової установи// Інтелектуальні комп’ютерні системи та мережі : матеріали V Всеукраїнської науково-практичної конференції, 20 травня, Тернопіль. Тернопіль : Західноукраїнський національний університет, факультет комп’ютерних інформаційних технологій, кафедра комп’ютерної інженерії, 2026. С. 224-226.

Додаток А

Фрагменти програмного коду застосунку

```
from datetime import datetime
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()


class User(db.Model):
    __tablename__ = "users"

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password_hash = db.Column(db.String(255), nullable=False)
    role = db.Column(db.String(30), nullable=False, default="employee")
    created_at = db.Column(db.DateTime, default=datetime.utcnow)


class Client(db.Model):
    __tablename__ = "clients"

    id = db.Column(db.Integer, primary_key=True)
    full_name = db.Column(db.String(150), nullable=False)
    phone = db.Column(db.String(30), nullable=False)
    document_number = db.Column(db.String(50), unique=True, nullable=False)
    address = db.Column(db.String(200))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    pawn_items = db.relationship("PawnItem", backref="client", lazy=True)
    contracts = db.relationship("Contract", backref="client", lazy=True)


class Category(db.Model):
    __tablename__ = "categories"

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), unique=True, nullable=False)
    description = db.Column(db.Text)

    pawn_items = db.relationship("PawnItem", backref="category", lazy=True)


class PawnItem(db.Model):
    __tablename__ = "pawn_items"

    id = db.Column(db.Integer, primary_key=True)
    client_id = db.Column(db.Integer, db.ForeignKey("clients.id"), nullable=False)
```

```
            category_id = db.Column(db.Integer, db.ForeignKey("categories.id"),
nullable=False)
            name = db.Column(db.String(150), nullable=False)
            description = db.Column(db.Text)
            estimated_value = db.Column(db.Numeric(10, 2), nullable=False)
            status = db.Column(db.String(30), nullable=False, default="available")
            created_at = db.Column(db.DateTime, default=datetime.utcnow)

            contract = db.relationship("Contract", backref="pawn_item", uselist=False)


        class Contract(db.Model):
            __tablename__ = "contracts"

            id = db.Column(db.Integer, primary_key=True)
            client_id = db.Column(db.Integer, db.ForeignKey("clients.id"), nullable=False)
            pawn_item_id = db.Column(db.Integer, db.ForeignKey("pawn_items.id"),
nullable=False)
            loan_amount = db.Column(db.Numeric(10, 2), nullable=False)
            interest_rate = db.Column(db.Numeric(5, 2), nullable=False)
            start_date = db.Column(db.Date, nullable=False)
            end_date = db.Column(db.Date, nullable=False)
            status = db.Column(db.String(30), nullable=False, default="active")
            created_at = db.Column(db.DateTime, default=datetime.utcnow)

            payments = db.relationship("Payment", backref="contract", lazy=True)


        class Payment(db.Model):
            __tablename__ = "payments"

            id = db.Column(db.Integer, primary_key=True)
            contract_id = db.Column(db.Integer, db.ForeignKey("contracts.id"),
nullable=False)
            amount = db.Column(db.Numeric(10, 2), nullable=False)
            payment_date = db.Column(db.Date, nullable=False)
            payment_type = db.Column(db.String(50), nullable=False)
            comment = db.Column(db.Text)
            created_at = db.Column(db.DateTime, default=datetime.utcnow)
```

Додаток Б

SQL-структура таблиць

```
CREATE TABLE clients (
    id INTEGER PRIMARY KEY,
    full_name VARCHAR(150) NOT NULL,
    phone VARCHAR(30) NOT NULL,
    document_number VARCHAR(50) UNIQUE NOT NULL,
    address VARCHAR(200),
    created_at DATETIME
);

CREATE TABLE categories (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);

CREATE TABLE pawn_items (
    id INTEGER PRIMARY KEY,
    client_id INTEGER NOT NULL,
    category_id INTEGER NOT NULL,
    name VARCHAR(150) NOT NULL,
    description TEXT,
    estimated_value NUMERIC(10, 2) NOT NULL,
    status VARCHAR(30) NOT NULL,
    created_at DATETIME,
    FOREIGN KEY (client_id) REFERENCES clients(id),
    FOREIGN KEY (category_id) REFERENCES categories(id)
);

CREATE TABLE contracts (
    id INTEGER PRIMARY KEY,
    client_id INTEGER NOT NULL,
    pawn_item_id INTEGER NOT NULL,
    loan_amount NUMERIC(10, 2) NOT NULL,
    interest_rate NUMERIC(5, 2) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    status VARCHAR(30) NOT NULL,
    created_at DATETIME,
    FOREIGN KEY (client_id) REFERENCES clients(id),
    FOREIGN KEY (pawn_item_id) REFERENCES pawn_items(id)
);

CREATE TABLE payments (
    id INTEGER PRIMARY KEY,
    contract_id INTEGER NOT NULL,
    amount NUMERIC(10, 2) NOT NULL,
    payment_date DATE NOT NULL,
    payment_type VARCHAR(50) NOT NULL,
    comment TEXT,
    created_at DATETIME,
    FOREIGN KEY (contract_id) REFERENCES contracts(id)
);
```

Додаток В

Скріншоти реалізованого інформаційного застосунку

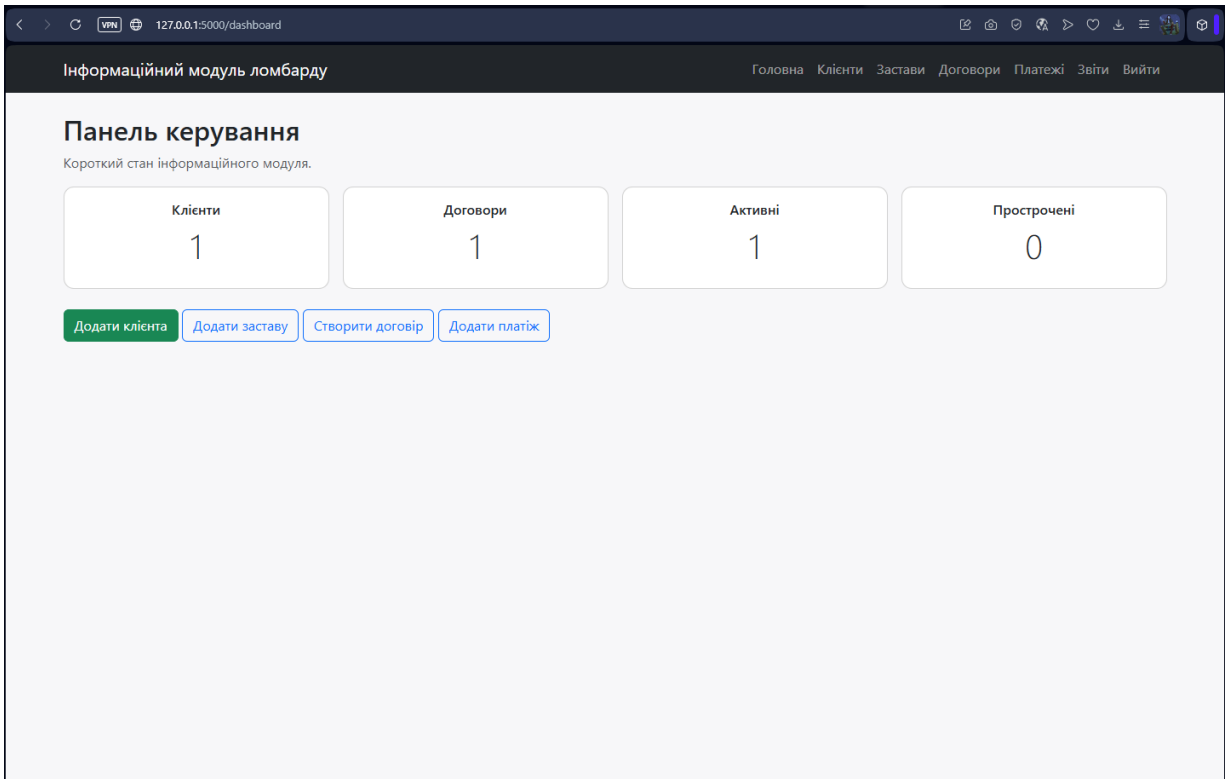

Рисунок В.1 – Панель керування інформаційного модуля

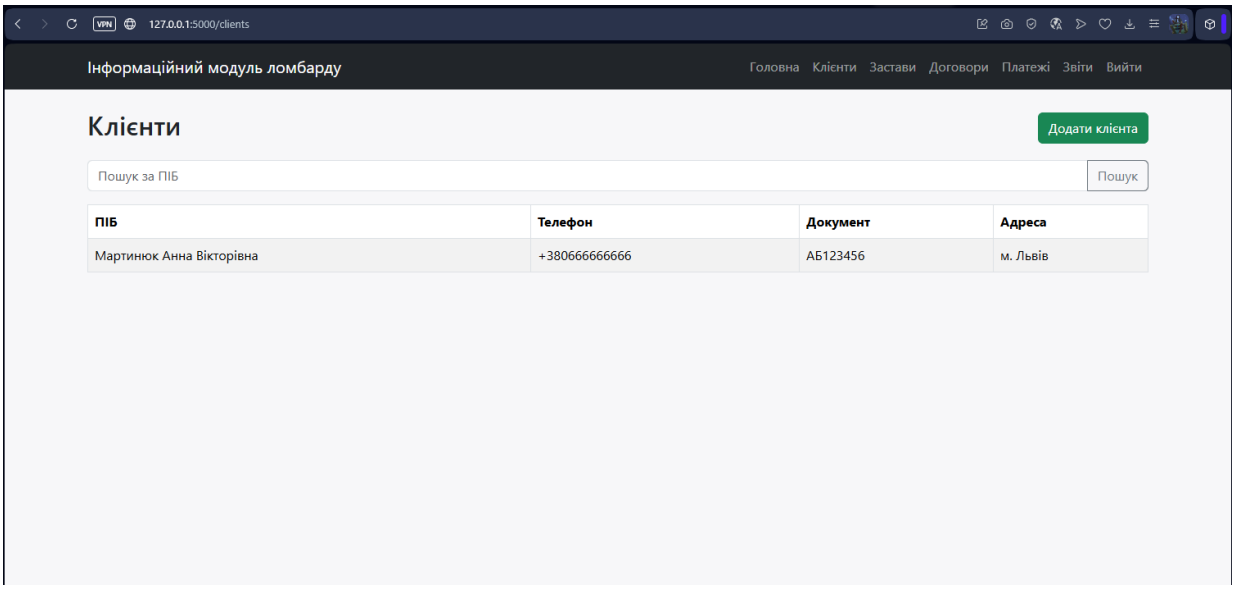

Рисунок В.2 – Сторінка обліку клієнтів

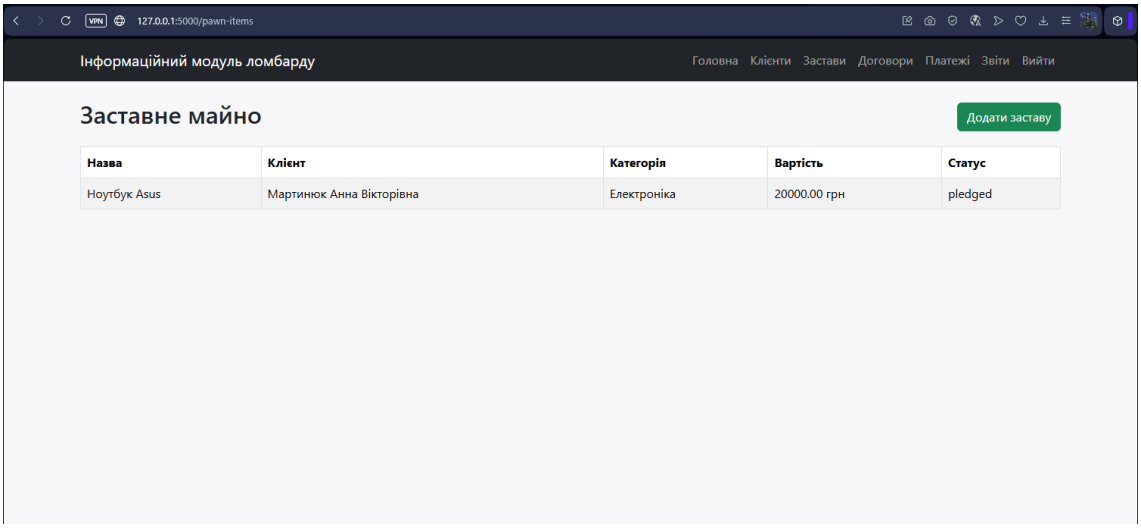

Рисунок В.3 – Сторінка обліку заставного майна

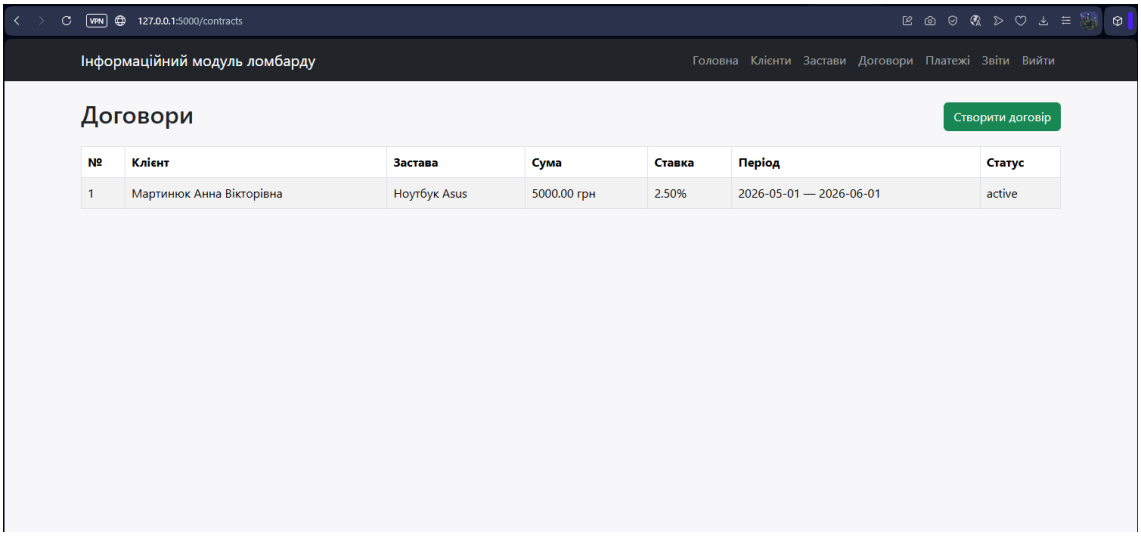

Рисунок В.4 – Сторінка обліку договорів

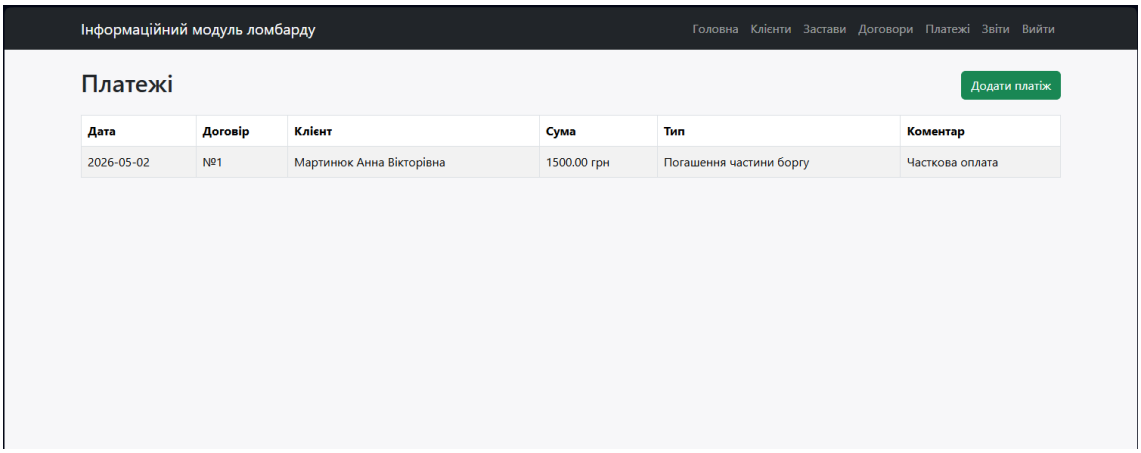

Рисунок В.5 – Сторінка обліку платежів

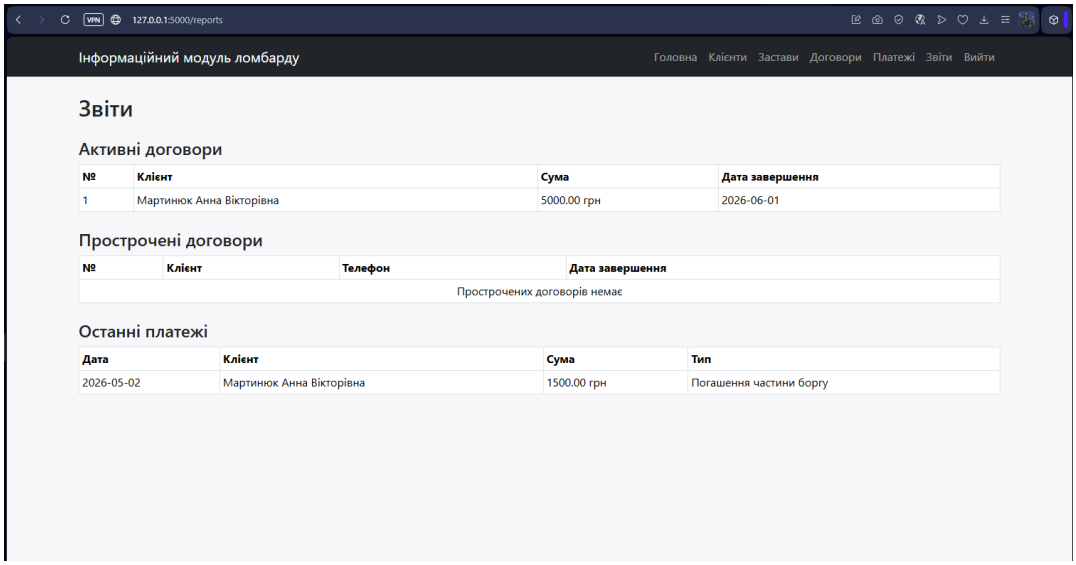

Рисунок В.6 – Сторінка звітів

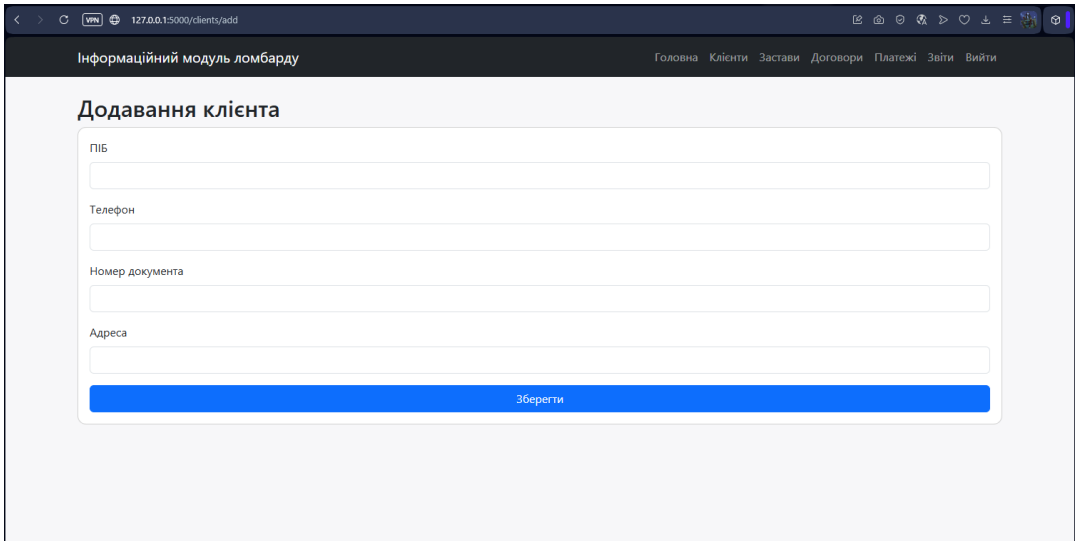

Рисунок В.7 – Форма додавання клієнта

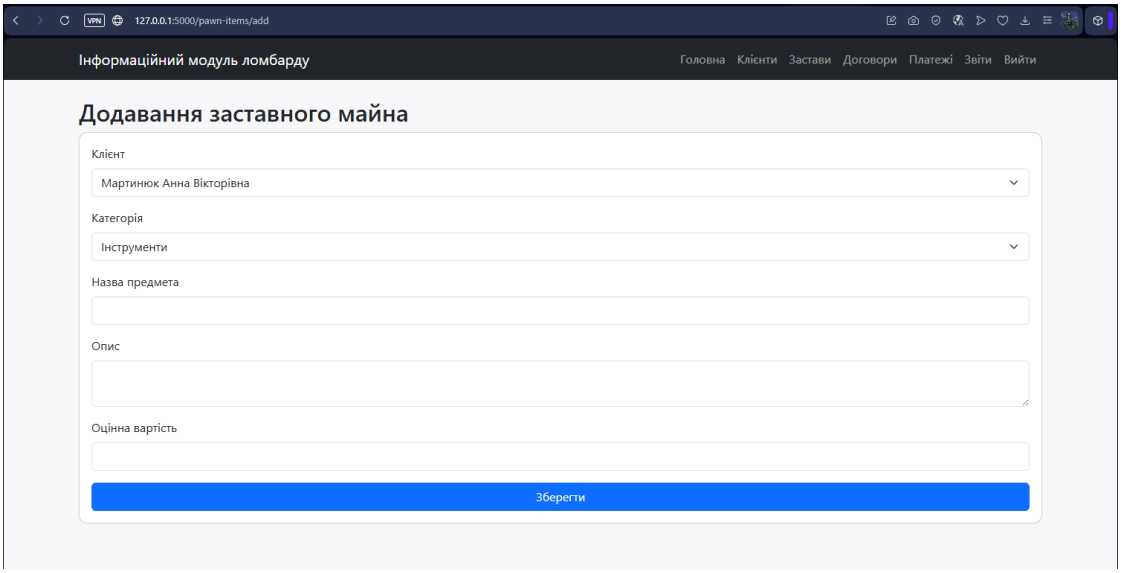

Рисунок В.8 – Форма додавання заставного майна

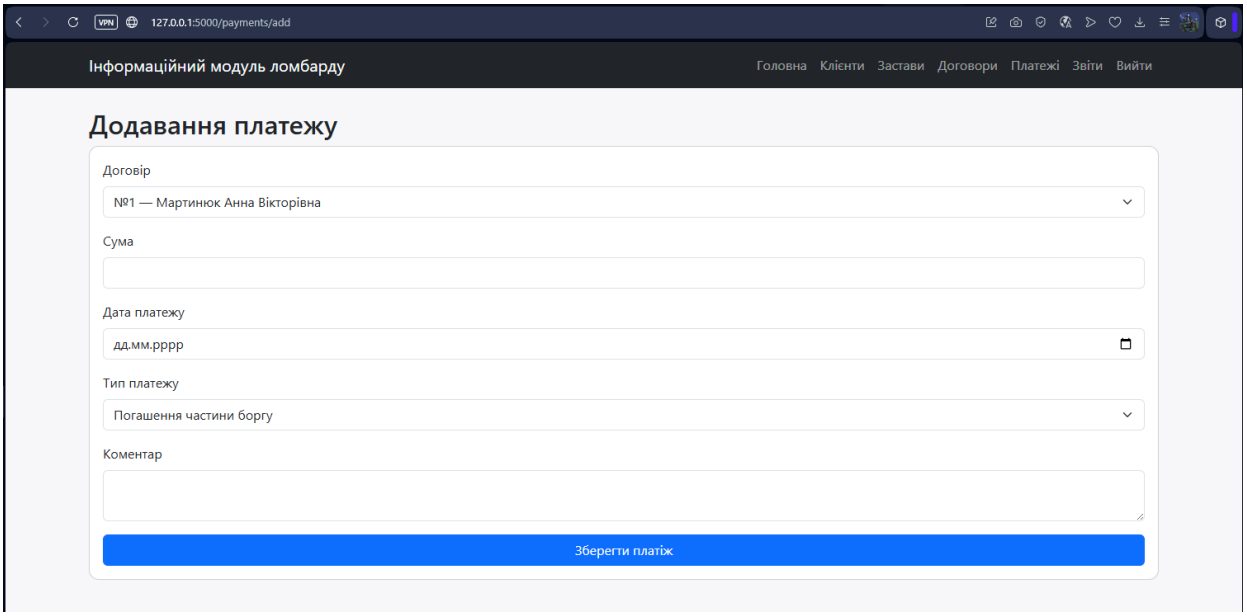

Рисунок В.9 – Форма додавання платежу

Додаток Г

Копія опублікованих результатів

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

WEST UKRAINIAN NATIONAL UNIVERSITY

Комп'ютерна Інженерія

IV ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

ІКСМ весна 2026

20 ТРАВНЯ 2026

Наукове товариство ЗУНУ

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2026

Сельський П.Р., д.м.н., професор, Тернопільський національний медичний університет імені І. Я. Горбачевського ;
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка» ;
Теслюк В.М., д.т.н., професор, НУ "Львівська політехніка" ;
Тимченко Л.І., д.т.н., професор, Державний університет інфраструктури та технологій;
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка" ;
Якименко І.З., к.т.н., доцент, Західноукраїнський національний університет;
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет ;
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад організаційного комітету:

Мельник Г.М. к.т.н., доцент, Західноукраїнський національний університет

Піцун О.Й. к.т.н., доцент, Західноукраїнський національний університет

Технічна підтримка:

Зінькевич О. В. студентка, Західноукраїнський національний університет

Кіт М. О. студентка, Західноукраїнський національний університет

Лебединський Т.В. студент,Західноукраїнський національний університет;

Кришталь Е.Р., студент, Західноукраїнський національний університет;

Метою конференції є представлення та обговорення наукових і практичних результатів, сприяння активізації творчої і інноваційної діяльності студентів, аспірантів і молодих вчених.
Конференція проводиться із залученням Ради Молодих Вчених та Студентського Наукового Товариства ФКІТ ЗУНУ.

IV Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ весна 2026), м. Тернопіль, ЗУНУ, 29 травня 2026 р. Тернопіль, 2026

Адреса:
Вул. О. Теліги, 8, корпус №6, Тернопіль
URL: https://ki.wunu.edu.ua/conference/
e-mail: kistudconference@gmail.com

Тези доповідей подаються за оригіналом рукопису

Микитюк В.В.
студентка 4 курсу ННІНОТ ЗУНУ
Науковий керівник к.т.н., доцент Домбровський М.З., кафедра ІОСУ ЗУНУ

ІНФОРМАЦІЙНИЙ ЗАСТОСУНОК УПРАВЛІННЯ ОПЕРАЦІЯМИ НЕБАНКІВСЬКОЇ ФІНАНСОВОЇ УСТАНОВИ

Вступ. Управління операціями небанківської фінансової установи пов'язане з постійним опрацюванням даних про клієнтів, майнові об'єкти, договори, платежі та стан виконання фінансових зобов'язань. Для таких установ важливо забезпечити не лише зберігання інформації, а й її цілісність, швидкий пошук, контроль строків і підтримку прийняття операційних рішень. Нормативною основою діяльності небанківських фінансових установ є законодавство України у сфері фінансових послуг [1] та регуляторні матеріали Національного банку України щодо небанківського сектору [2].

Актуальність розроблення інформаційного застосунку зумовлена тим, що в практичній діяльності невеликих фінансових установ частина операцій може виконуватися за допомогою паперових документів або окремих електронних таблиць. Це ускладнює контроль пов'язаних даних, підвищує ризик дублювання записів і не дає змоги швидко отримати узагальнену інформацію. У межах дослідження як приклад небанківської фінансової установи розглянуто ломбард, діяльність якого містить чітку послідовність операцій: реєстрацію клієнта, опис предмета застави, оформлення договору, внесення платежів і перегляд звітності.

Постановка задачі. Метою роботи є обґрунтування підходу до розроблення інформаційного застосунку управління операціями небанківської фінансової установи та опис програмної реалізації демонстраційного вебзастосунку на прикладі ломбарду. Об'єктом дослідження є процеси управління операціями небанківської фінансової установи, а предметом — методи, моделі та програмні засоби створення інформаційно-обчислювального застосунку для підтримки роботи з клієнтами, заставним майном, договорами, платежами та звітною інформацією.

Основний матеріал. Інформаційна модель застосунку побудована на основі реляційного підходу. Основними сутностями системи визначено користувача, клієнта, категорію майна, предмет застави, договір і платіж. Сутність клієнта пов'язана з предметами застави та договорами, договір поєднує клієнта, предмет застави й фінансові умови операції, а платежі формують історію виконання договору. Така структура дає змогу уникнути дублювання даних і забезпечити логічні зв'язки між основними операціями.

Основні модулі розроблюваного застосунку подано у таблиці 1.

Таблиця 1 – Основні модулі інформаційного застосунку управління операціями

Модуль застосунку	Основне призначення
Клієнти	Реєстрація, пошук і перегляд даних клієнта та пов'язаної інформації.
Заставне майно	Фіксація категорії, опису, оцінної вартості та статусу предмета.
Договори	Створення договору на основі клієнта і предмета застави, контроль строків.
Платежі та звітність	Реєстрація оплат і формування узагальненої інформації для працівника.

Для реалізації серверної частини застосунку використано мову програмування Python, яка має простий синтаксис і достатній набір засобів для веброзробки та роботи з базами даних [3]. Вебчастину реалізовано з використанням Flask, що забезпечує маршрутизацію запитів, роботу з шаблонами та формами [4]. Роботу з даними організовано через SQLAlchemy ORM, що дозволяє описувати таблиці у вигляді класів Python і виконувати базові операції з даними без

написання великої кількості SQL-запитів вручну [5]. У демонстраційній версії використано SQLite, оскільки ця СУБД не потребує окремого серверного налаштування та є зручною для локального запуску [6], а для подальшого впровадження системи у реальній установі може бути застосовано іншу серверну СУБД, зокрема PostgreSQL.

Архітектура застосунку передбачає поділ на рівень представлення, серверний рівень, рівень бізнес-логіки, рівень доступу до даних і рівень даних. Рівень представлення реалізовано засобами HTML, CSS, Bootstrap і шаблонів Jinja2. Серверний рівень приймає HTTP-запити, виконує маршрутизацію і передає дані до шаблонів. На рівні бізнес-логіки здійснюється перевірка введених даних, створення договорів, зміна статусів заставного майна та підготовка звітної інформації.

Оскільки система оперує персональними даними клієнтів і фінансовими записами, у ній передбачено авторизацію користувача, перевірку сеансу та зберігання паролів у хешованому форматі. Такі рішення відповідають загальним підходам до зменшення ризиків вебзастосунків, зокрема тим, що розглядаються у матеріалах OWASP щодо поширених загроз безпеці [7].

У процесі реалізації сформовано набір сторінок користувацького інтерфейсу: сторінку входу, панель керування, списки клієнтів, заставного майна, договорів, платежів і звітів. Працівник установи може створити запис клієнта, додати предмет застави, сформувати договір, зареєструвати платіж і переглянути основну звітну інформацію. Демонстраційний характер застосунку дає змогу перевірити послідовність операцій без складного серверного розгортання.

Тестування застосунку проводилося за основними сценаріями роботи користувача: вхід до системи, додавання клієнта, реєстрація предмета застави, створення договору, внесення платежу та перегляд звітів. Також перевірялася обробка некоректних або неповних даних. Результати перевірки підтвердили працездатність основних модулів і відповідність реалізованого функціоналу поставленим вимогам.

Висновки. Отже, запропонований інформаційний застосунок демонструє практичний підхід до автоматизації операцій небанківської фінансової установи на прикладі ломбарду. Поєднання вебінтерфейсу, реляційної бази даних, ORM-рівня та засобів авторизації дозволяє забезпечити централізоване зберігання інформації, підтримку ключових операцій і підготовку базової звітності. Розроблений підхід може бути адаптований до потреб інших невеликих небанківських фінансових установ.

Список літератури

1. Про фінансові послуги та фінансові компанії : Закон України від 14.12.2021 № 1953-IX. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/1953-20.

2. Національний банк України. Реєстри та переліки небанківських фінансових установ. URL: https://bank.gov.ua/ua/supervision/nonbanks/registers-lists.

3. Python Software Foundation. Python 3.12 Documentation. URL: https://docs.python.org/3.12/.

4. Pallets Projects. Flask Documentation. Version 3.1.x. URL: https://flask.palletsprojects.com/.

5. SQLAlchemy. SQLAlchemy Documentation. URL: https://docs.sqlalchemy.org/.

6. SQLite. SQLite Documentation. URL: https://sqlite.org/docs.html.

7. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/.