ZADANIE PROJEKTOWE   
1 PROJEKTOWANIE SYSTEMU BAZ DANYCH   
1.1. Analiza dziedziny problemu   
1.2. Główne zadania do rozwiązania   
1.3. Główne zapytania według poziomów dostępu użytkowników  
1.4. Normalizacja relacji oraz jej uzasadnienie   
1.5. Budowa grafu schematu modelu infologicznego oraz weryfikacja jego poprawności

2 REALIZACJA SYSTEMU BAZ DANYCH I OPROGRAMOWANIA   
2.1. Uzasadnienie wyboru systemu zarządzania bazą danych (SZBD) i środowiska programistycznego   
2.2. Schemat strukturalny programu i jego opis   
2.3. Opis interfejsu użytkownika i narzędzi jego tworzenia   
2.4. Przeznaczenie głównych procedur i funkcji   
2.5. Realizacja złożonych zapytań selekcyjnych i raportów dla użytkowników różnych poziomów

3 BEZPIECZEŃSTWO I FUNKCJE SERWISOWE SYSTEMU   
3.1. Rejestracja i uwierzytelnianie użytkowników z różnym dostępem   
3.2. Archiwizacja, przywracanie i inne funkcje serwisu oraz konfiguracji   
3.3. Prowadzenie dziennika aktywności użytkowników systemu i jego przetwarzanie

4 INSTRUKCJE UŻYTKOWANIA I WSPARCIA DLA UŻYTKOWNIKÓW WEDŁUG POZIOMU DOSTĘPU   
4.1 Instrukcje dla administratora systemu   
4.2 Instrukcje dla pracownika

WNIOSKI

ZAŁĄCZNIK

Cele Projectu

Utwórz bazę danych przy użyciu wybranego systemu DBMS i aplikacji programowej (lub witryny internetowej), która umożliwia wprowadzanie danych, przeglądanie, edycję i wykonywanie wszystkich żądań użytkowników zgodnie z zidentyfikowanymi zadaniami dotyczącymi przepływów informacji w danym obszarze tematycznym.

Ogólne wymagania dotyczące bazy danych i jej działania:

1. Przeprowadzić kompleksowe badanie obszaru tematycznego;

2. Zidentyfikuj główne przepływy informacji i zadania użytkownika, które należy rozwiązać;

3. Określić rodzaj, zasady tworzenia i typy danych podstawowych dla dokumentów przychodzących i wychodzących;

4. Identyfikować anomalie występujące podczas pracy z obiektami danego obszaru tematycznego przy wykonywaniu podstawowych operacji na danych oraz przeprowadzać normalizację relacji;

5. Zbuduj model infologiczny obszaru tematycznego i sprawdź jego poprawność;

6. Dla odpowiedniego modelu infologicznego należy wykonać implementację w jednym z relacyjnych systemów DBMS;

7. Opracuj indywidualne formularze (i strony witryny) dla użytkowników o różnym poziomie zaawansowania, zapewniając przyjazny interfejs zorientowany na tematykę. Formularze te powinny umożliwiać nawigację po tabeli i rozwiązywanie zadań użytkownika.

8. Rozważ możliwość archiwizacji i przywrócenia tabel bazy danych lub całej bazy danych; tworzenie kopii bazy danych w celu przeniesienia do innego miejsca pracy; Administrator może przeglądać dziennik działań użytkownika, korzystając z odpowiednich zapytań (filtrów).

9. Twórz zapytania różnego typu i raporty dla zapytań podstawowych. Poprzez próbkowanie zapytań o różnym stopniu złożoności upewnij się, że niezbędne informacje są dostarczane użytkownikom na różnych poziomach

10. Zapewnij kontrolę danych podczas wprowadzania informacji według typu, zakresu, wartości i integralności danych;

11. Przeprowadzanie testów i debugowania produktu programowego;

12. Opracować instrukcje dla użytkowników o różnym poziomie zaawansowania;

13. Przygotuj sprawozdanie z pracy kursowej.

# Implementacja BD

## 1.1.

Jako obszar tematyczny wybrano „Zarządzanie kredytami w organizacji mikrofinansowej (MFO)”. Opracowany system ma na celu automatyzację procesów księgowania klientów, udzielania pożyczek, ustalania terminów spłat, tworzenia harmonogramów spłat, a także analiz wewnętrznych i raportowania.

Oprogramowanie umożliwia menadżerom tworzenie klientów, udzielanie pożyczek, monitorowanie ich statusu, tworzenie harmonogramów płatności i przetwarzanie faktycznych płatności. Użytkownicy z uprawnieniami administratora mogą przeglądać dziennik akcji, ograniczać dostęp poza godzinami pracy i tworzyć kopię zapasową bazy danych.

Analizując przedmiot zagadnienia, postawiono sobie za zadanie stworzenie systemu informatycznego zapewniającego przejrzyste, kontrolowane i efektywne zarządzanie transakcjami finansowymi, charakterystycznymi dla instytucji mikrofinansowych. Główny nacisk położono na automatyzację operacji pożyczkowych i księgowych oraz monitorowanie dotrzymywania terminów.

Podkreślono następujące główne podmioty:

1. Klient

Przechowuje dane osobowe osoby ubiegającej się o pożyczkę.

• imię – pełne imię i nazwisko klienta

• telefon – numer kontaktowy

• dateofbirth – data urodzenia

• customertypeid – typ klienta (osoba fizyczna, osoba prawna itp.)

2. Pożyczka

Informacje o konkretnej pożyczce udzielonej klientowi.

• kwota – całkowita kwota pożyczki

• termmonths – semestr w miesiącach

• interestrate – stopa procentowa

• creationtime – data utworzenia

• statusid – status (wydany, wykupiony, przeterminowany)

• loantypeid – typ pożyczki

• customerid – relacja z klientem

3. Harmonogram płatności (PaymentSchedule)

Zawiera zaplanowane spłaty pożyczki.

• termin płatności – planowana data płatności

• kwota – kwota

• statusid – status (1 – oczekujący, 2 – opłacony)

4. Płatność

Rzeczywista płatność dokonana przez klienta.

• data płatności – data płatności

• kwota – kwota

• loanid – pożyczka, do której należy

• paymenttypeid – rodzaj płatności (gotówka, przelew bankowy itp.)

5. Użytkownik systemu (Pracownik)

Pracownik odpowiedzialny za organizację.

• imię – imię i nazwisko pracownika

• login, hasło – dane uwierzytelniające

• positionid – rola (menedżer, administrator)

6. Pozycja

Rola użytkownika w systemie.

• positionname – nazwa stanowiska

(służy do autoryzacji i kontroli dostępu)

7. Dziennik aktywności (UserLog)

Rejestruje wszystkie istotne działania użytkownika w systemie.

● user\_id – kto wykonał akcję

● akcja – typ akcji (utwórz, zaktualizuj itp.)

● table\_name – na jakim elemencie wykonywana jest akcja?

● szczegóły – treść zmian

● znacznik czasu – czas wykonania akcji

8. Rodzaj pożyczki / statusy / typy klientów / rodzaje płatności

**1.2. Główne zadania do rozwiązania**

Aby zapewnić efektywne funkcjonowanie systemu zarządzania operacjami kredytowymi w organizacji mikrofinansowej, wdrożono szereg kluczowych zadań, które zapewniają podstawowe kompetencje pracownikom o różnych poziomach dostępu.

1. Autoryzacja użytkownika

Użytkownicy muszą mieć możliwość zalogowania się do systemu przy użyciu nazwy użytkownika i hasła.

Kategorie użytkowników: manager, administrator.

Wymagania dostępowe: rejestracji dokonuje administrator. Samodzielna rejestracja nie jest dostępna. Autoryzacja jest kontrolowana za pomocą tokenów JWT i ograniczona zgodnie z rolą użytkownika.

2. Zarządzanie klientami

Menedżerowie mogą dodawać nowych klientów, przeglądać ich dane oraz edytować lub usuwać informacje według potrzeb.

Kategorie użytkowników: manager, administrator.

Wymagania dostępu: tylko dla autoryzowanych użytkowników posiadających odpowiednią rolę.

3. Udzielanie i obsługa pożyczek

Menedżer może zorganizować dla klienta nową pożyczkę, określając jej kwotę, okres, rodzaj i stopę procentową. Możliwa jest również zmiana parametrów istniejącej pożyczki i sprawdzenie jej aktualnego statusu.

Kategorie użytkowników: manager, administrator.

Wymagania dostępu: Dostępne wyłącznie dla upoważnionych pracowników.

4. Generowanie harmonogramów płatności

Podczas tworzenia lub edycji pożyczki automatycznie generowany jest harmonogram zaplanowanych płatności, uwzględniający okres i kwotę.

Kategorie użytkowników: systemowe (automatyczne), przeglądające – menedżer.

Wymagania dostępu: Tylko dla zarejestrowanych menedżerów i administratorów.

5. Prawdziwe przetwarzanie płatności

Menedżer rejestruje rzeczywiste płatności klientów. System automatycznie aktualizuje harmonogram: zmienia status płatności na „zapłacone” lub przywraca je do „oczekujące” po usunięciu.

Kategorie użytkowników: manager.

Wymagania dostępu: wyłącznie dla upoważnionych pracowników.

6. Przeglądanie informacji według klienta

System umożliwia przeglądanie historii pożyczek, płatności i harmonogramów spłat dla każdego klienta osobno.

Kategorie użytkowników: manager, administrator.

Wymagania dostępu: wyłącznie dla upoważnionych użytkowników.

7. Zarządzanie okresem księgowym

Administrator może ustalić okres rozliczeniowy, w którym praca z bazą danych będzie dozwolona. Po upływie okresu system blokuje wszystkie operacje za wyjątkiem zmiany samego okresu.

Kategorie użytkowników: superadministrator.

Wymagania dostępu: Tylko upoważniony administrator.

8. Prowadzenie dziennika aktywności

System automatycznie rejestruje wszystkie istotne działania użytkownika – dodawanie, edycję i usuwanie rekordów. Administrator może przeglądać dziennik, korzystając z filtrów i sortowania.

Kategorie użytkowników: administrator.

Wymagania dostępu: Tylko upoważniony administrator.

2. Kopia zapasowa i przywracanie

Administrator może tworzyć kopie zapasowe bazy danych, ładować je lub przywracać system do poprzedniego stanu.

Kategorie użytkowników: administrator.

Wymagania dostępu: Tylko administrator.

1.3. Podstawowe zapytania według poziomów dostępu użytkownika

System organizacji zajmującej się mikrofinansami implementuje szereg zapytań SQL odpowiadających kluczowym operacjom, pogrupowanym według jednostek i dostępu.

Administrator może tworzyć, modyfikować i usuwać użytkowników (pracowników). Na przykład:

• Tworzenie nowego użytkownika:

WSTAW DO pracownicy (imię, login, hasło, id stanowiska) WARTOŚCI ('Olena Ivanenko', 'admin', '<hashowane hasło>', 1);

• Aktualizacja użytkownika:

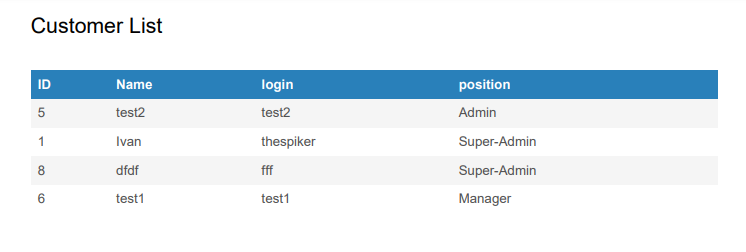
AKTUALIZACJA pracowników SET name = 'Olena Petrovna', login = 'admin2', password = '<new\_hash>', positionid = 2 WHERE id = 3;

• Usuwanie użytkownika:

USUŃ Z pracowników, GDZIE id = 3;

• Poszukiwanie pracowników:

WYBIERZ e.id, e.name, e.login, p.positionname Z pracownicy e DOŁĄCZ stanowiska p NA e.positionid = p.id GDZIE e.name ILIKE '%olena%' LUB e.login ILIKE '%admin%';



Rysunek 1.1 – Przykład raportu użytkownika

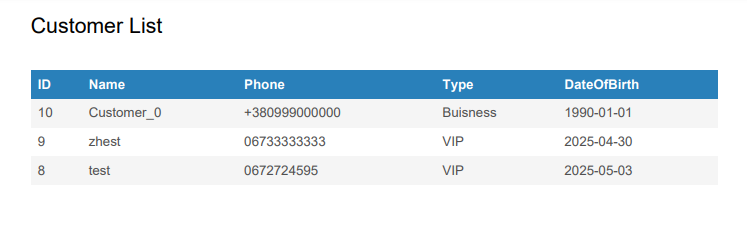
Klienci – to osoby, które otrzymują pożyczki. Menedżerowie mogą wykonywać następujące zapytania:

• Dodawanie klienta:  
INSERT INTO customers (name, phone, dateofbirth, customertypeid) VALUES ('Iwan Petrenko', '+380991112233', '1990-01-01', 1);

• Edycja klienta:  
UPDATE customers SET name = 'Iwan P.', phone = '+380999000111', dateofbirth = '1991-05-12', customertypeid = 2 WHERE id = 5;

• Usuwanie klienta:  
DELETE FROM customers WHERE id = 5;

• Wyszukiwanie klientów:  
SELECT \* FROM customers WHERE name ILIKE '%iwan%' OR phone ILIKE '%099%';



Rysunek 1.2 – Przykład raportu dotyczącego klientów

Pożyczki są tworzone dla klientów za pomocą następujących zapytań:

• Tworzenie pożyczki:

INSERT INTO loans (customerid, amount, termmonths, interestrate, statusid, loantypeid, creationtime) VALUES (5, 12000.00, 6, 12.5, 1, 1, CURRENT\_DATE);

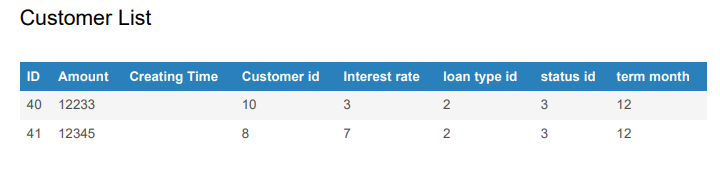
• Edycja pożyczki:

UPDATE loans SET amount = 15000.00, termmonths = 12, interestrate = 14.0 WHERE id = 10;

• Usuwanie pożyczki:

DELETE FROM loans WHERE id = 10;

• Przegląd pożyczek klienta:

SELECT \* FROM loans WHERE customerid = 5;

Rysunek 1.3 – Przykład raportu dotyczącego pożyczek

Harmonogramy płatności są tworzone automatycznie, ale mogą być również aktualizowane ręcznie:

• Dodanie płatności do harmonogramu:  
INSERT INTO payment\_schedules (loanid, duedate, amount, statusid) VALUES (10, '2025-06-10', 2000.00, 1);

• Oznaczenie jako „opłacone”:  
UPDATE payment\_schedules SET statusid = 2 WHERE id = 15;

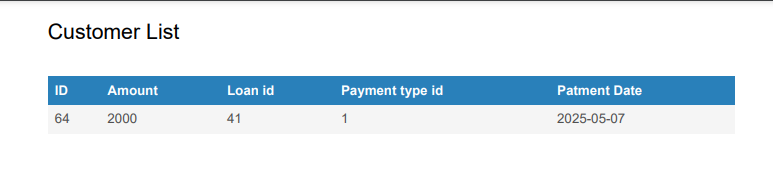
• Usunięcie harmonogramu płatności:  
DELETE FROM payment\_schedules WHERE id = 15;

Płatności są rejestrowane przez menedżera po otrzymaniu środków:

• Utworzenie płatności:  
INSERT INTO payments (loanid, amount, paymentdate, paymenttypeid) VALUES (10, 2000.00, CURRENT\_DATE, 1);

• Edycja płatności:  
UPDATE payments SET amount = 2500.00, paymenttypeid = 2 WHERE id = 20;

• Usunięcie płatności:  
DELETE FROM payments WHERE id = 20;



Rysunek 1.4 – Przykład raportu dotyczącego płatności

Okres rozliczeniowy jest określany przez administratora, który kontroluje dopuszczalne daty zmian w systemie:

• Ustawienie nowego okresu:  
UPDATE accounting\_period SET start\_date = '2025-01-01', end\_date = '2025-12-31' WHERE id = 1;

Dziennik operacji rejestruje wszystkie krytyczne operacje:

• Utworzenie wpisu:  
INSERT INTO user\_logs (user\_id, action, table\_name, details, timestamp) VALUES (2, 'створення', 'customers', 'Створено клієнта Іван Петренко', NOW());

• Wyszukiwanie według tabeli:  
SELECT \* FROM user\_logs WHERE table\_name = 'loans';

• Wyszukiwanie według użytkownika:  
SELECT \* FROM user\_logs WHERE user\_id = 2 ORDER BY timestamp DESC;

Kopia zapasowa systemu jest wykonywana przez administratora:

• Tworzenie kopii bazy:  
pg\_dump -U postgres -d curs -F c -f backup\_2025\_05\_10.dump

• Przywracanie:  
pg\_restore -U postgres -d curs -c backup\_2025\_05\_10.dump

Wszystkie te zapytania są zaimplementowane jako API-endpointy oraz instrukcje SQL, do których mają dostęp wyłącznie osoby upoważnione zgodnie ze swoją rolą (menedżer lub administrator). System wspiera niezawodne logowanie, analizę danych i kontrolę działań w czasie rzeczywistym.

1.4. Normalizacja relacji i jej uzasadnienie

Pierwsza postać normalna (1NF)  
Początkowe tabele bazy danych już spełniają wymagania 1NF. Wszystkie atrybuty są atomowe – zawierają niepodzielne wartości, bez list czy powtarzających się grup w komórkach. Na przykład tabela Customers ma tylko jedno pole phone, a nie phone1, phone2 itd. Powtarzalne struktury, takie jak płatności związane z pożyczkami, zostały przeniesione do osobnych tabel Payments, co jest poprawnym podejściem. Wszystkie tabele mają jasno określony klucz główny – zwykle pole id.  
Zatem nie stwierdzono naruszeń 1NF, a obecna struktura nadaje się do dalszej normalizacji.

Druga postać normalna (2NF)  
2NF zakłada, że każdy atrybut niekluczowy powinien zależeć od całego klucza głównego, a nie tylko od jego części. W naszej bazie wszystkie klucze główne są proste (jednopolowe), więc formalnie nie występują częściowe zależności.  
Jednak zaobserwowano znaczną duplikację danych w tabelach: typy klientów (CustomerTypes), typy pożyczek (LoanTypes), statusy (LoanStatuses, ScheduleStatuses), typy płatności (PaymentTypes) i stanowiska (Positions). Powtarzające się wartości zwiększały ryzyko anomalii aktualizacji.  
Aby usunąć nadmiarowość, wartości te zostały przeniesione do osobnych tabel słownikowych, a w głównych tabelach zastąpione przez klucze obce (\*\_id). To umożliwia centralne zarządzanie typami, unikanie błędów (np. ortograficznych) i ułatwia konserwację.

Trzecia postać normalna (3NF)  
3NF eliminuje zależności przechodnie – sytuacje, gdy jeden atrybut niekluczowy zależy od innego niekluczowego. W większości tabel takich zależności nie wykryto – wszystkie atrybuty zależą wyłącznie od klucza głównego.  
Wyjątkiem była tabela Contracts, która zawierała pole customer\_id, mimo że klient jest już zdefiniowany poprzez loan\_id. To tworzyło zależność przechodnią: contract\_id → loan\_id → customer\_id. Aby uniknąć redundancji, pole customer\_id zostało usunięte, a informacje o kliencie są uzyskiwane przez relację Contracts → Loans → Customers.  
Inne atrybuty takie jak status, position, type nie są już przechowywane jako pola tekstowe, lecz odnoszą się do odpowiednich tabel słownikowych. To zapewnia czystość strukturalną, spójność danych i łatwiejszą konserwację.

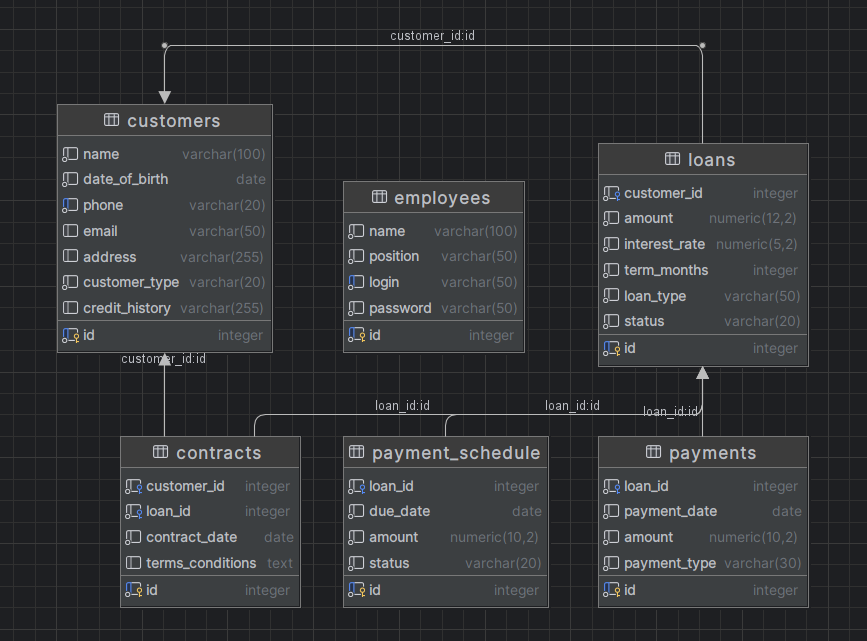
Wniosek po normalizacji do 3NF:

* Zredukowano duplikację danych, usunięto zależności przechodnie
* Poprawiono spójność i konserwowalność danych dzięki tabelom słownikowym
* Utworzono wyraźne relacje między encjami (poprzez klucze obce)
* Uniknięto potencjalnych anomalii wstawiania, aktualizacji i usuwania danych
* Schemat bazy danych jest teraz gotowy na skalowanie i rozbudowę funkcjonalności

Struktura bazy danych jest logicznie spójna, przejrzysta i elastyczna, co ułatwia rozwój złożonej logiki biznesowej systemu mikrofinansowego.

**1.5. Budowa diagramu grafowego modelu infologicznego i weryfikacja jego poprawności**

Po przeprowadzeniu analizy strukturalnej dziedziny tematycznej oraz zakończeniu etapów normalizacji, opracowano **model infologiczny danych** w postaci **diagramu ER**.  
Diagram ten stanowi logiczną wizualizację głównych obiektów systemu mikrofinansowego oraz relacji między nimi.

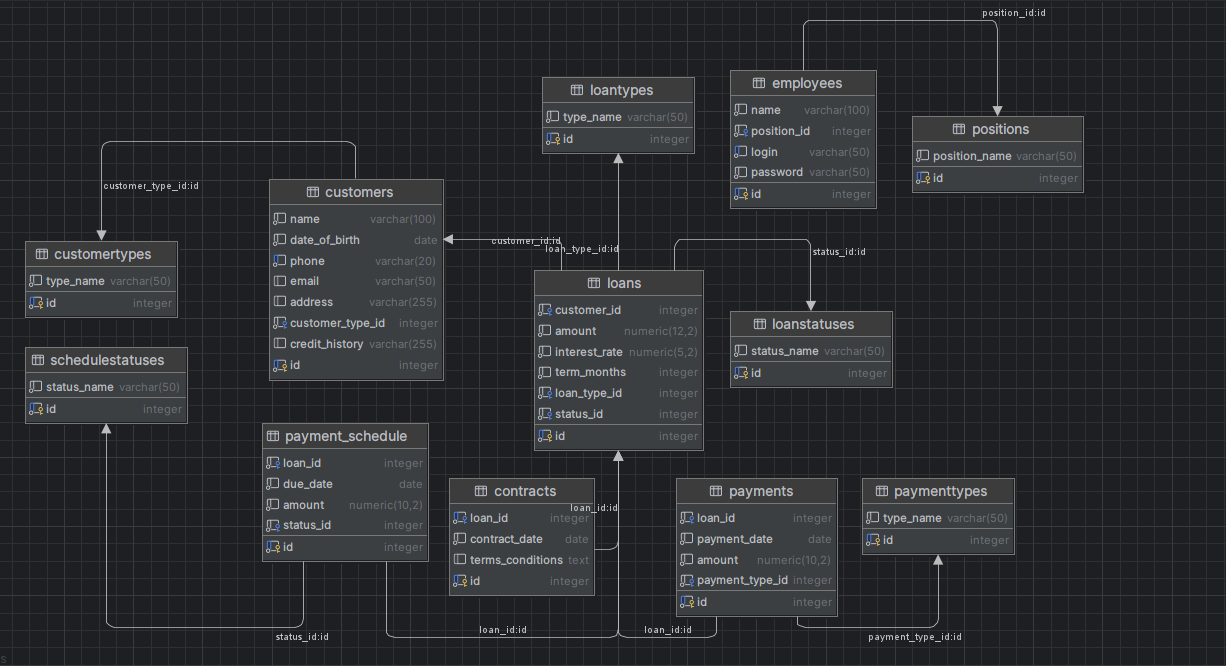


Rysunek 1.5 – Początkowy diagram ER

Początkowa wersja diagramu ER zawierała kluczowe encje: klienci, pracownicy, pożyczki, harmonogramy płatności, typy i statusy pożyczek, umowy oraz płatności.  
Encje te zostały zdefiniowane na podstawie analizy wymagań systemowych oraz procesu normalizacji danych. Przedstawiono typowe relacje między tabelami, na przykład relację „jeden do wielu” między klientami a pożyczkami lub pożyczkami a ich harmonogramami spłat.

W toku dalszego rozwoju systemu pojawiła się potrzeba rozszerzenia funkcjonalności, w szczególności w zakresie wsparcia narzędzi administracyjnych.  
W rezultacie dodano nowe tabele:

* Logs – do rejestrowania aktywności użytkowników w systemie (dodawanie, edycja, usuwanie danych),
* Settings – do przechowywania parametrów konfiguracyjnych systemu, w tym dat rozpoczęcia i zakończenia okresu rozliczeniowego.



Rysunek 1.6 – Uzupełniony diagram ER

Zaktualizowany diagram ER (zob. rys. 1.6) przedstawia pełny model struktury informacyjnej, rozszerzony o elementy niezbędne do zarządzania systemem.  
W modelu wyraźnie określono typy relacji:

* 1:N – na przykład jeden pracownik może obsługiwać wielu klientów lub pożyczek,
* N:1 – pożyczki odwołują się do słowników typów i statusów,
* 1:1 – między umową a konkretną pożyczką.

Wszystkie pola każdej tabeli zostały zweryfikowane pod kątem przynależności tylko do jednej encji, a wszystkie klucze obce zapewniają spójność i integralność relacji. Klucze główne gwarantują unikalność każdego obiektu w obrębie systemu.

W rezultacie końcowy diagram ER stał się logicznym odzwierciedleniem zaprojektowanego systemu, gotowym do implementacji w postaci fizycznej bazy danych w PostgreSQL lub innym systemie relacyjnych baz danych.

**2 REALIZACJA SYSTEMU BAZ DANYCH I OPROGRAMOWANIA**

**2.1. Uzasadnienie wyboru systemu zarządzania bazą danych oraz środowiska programistycznego**

Do realizacji oprogramowania organizacji mikropożyczkowej wybrano system zarządzania bazą danych **PostgreSQL**, który zapewnia wysoką wydajność, wsparcie dla transakcji, integralność danych oraz elastyczność w pracy z różnymi strukturami danych. PostgreSQL łatwo integruje się z technologiami **.NET** oraz oferuje szeroki zakres narzędzi do obsługi dużych zbiorów informacji.

Jako środowisko programistyczne dla części serwerowej wybrano **Microsoft Visual Studio 2022**, ponieważ zapewnia wygodną pracę z **ASP.NET Core Web API**, wspiera diagnostykę, integrację z Git oraz umożliwia wykonywanie testów jednostkowych i integracyjnych.

Do budowy frontendu wykorzystano **Next.js** – nowoczesny framework oparty na React, który wspiera SSR (Server Side Rendering) i ma efektywną integrację z **Firebase**. Dzięki Firebase zrealizowano przechowywanie plików oraz uwierzytelnianie. Wykorzystanie **Redux Toolkit** pozwoliło na centralne zarządzanie stanem aplikacji.

Komunikacja między frontendem a serwerem odbywa się poprzez REST API zbudowane na **ASP.NET Core**. Dostęp do bazy danych realizowany jest za pomocą **Entity Framework Core**, co pozwala na pracę z danymi jako obiektami oraz automatyczne tworzenie i aktualizację schematu bazy danych.

**2.2. Schemat strukturalny programu i jego opis**

System jest zbudowany zgodnie z zasadą podziału ról. Punktem wejścia jest autoryzacja użytkownika, po której następuje routing do odpowiedniego interfejsu użytkownika zgodnie z jego rolą (administrator lub pracownik).

Administrator posiada pełny zestaw funkcji: zarządzanie pracownikami, klientami, pożyczkami, płatnościami, okresami rozliczeniowymi, a także dostęp do dziennika działań i kopii zapasowych.

Pracownik ma ograniczony zakres możliwości: przegląd klientów, tworzenie pożyczek, edycja i spłata płatności. Wszystkie działania są logowane, a system dostępu realizowany jest za pomocą tokenów JWT oraz atrybutów autoryzacyjnych na poziomie kontrolerów.

**2.3. Opis interfejsu użytkownika oraz narzędzi jego tworzenia**

Interfejs frontendowy zrealizowano w oparciu o **Next.js** z użyciem **Material UI** do budowy komponentów. Każdy użytkownik po zalogowaniu widzi interfejs dopasowany do swojej roli.

Interfejs administratora zawiera dashboard z dynamicznymi wykresami, strony do zarządzania klientami, pożyczkami, płatnościami oraz okresami rozliczeniowymi. Dostępne są również funkcje generowania raportów PDF oraz przeglądania działań użytkowników poprzez dziennik.

Interfejs pracownika jest uproszczony i ukierunkowany na codzienne zadania operacyjne. Zastosowano strukturę komponentową z podziałem na strony, co zapewnia skalowalność i łatwość utrzymania kodu. Integracja z Firebase służy do obsługi plików i uwierzytelniania.

**2.4. Przeznaczenie głównych procedur i funkcji**

W systemie zaimplementowano kilka bloków funkcjonalnych pogrupowanych według kategorii:

* **Autoryzacja** – tworzenie tokenów JWT w celu weryfikacji praw dostępu.
* **Dziennik działań** – rejestrowanie wszystkich zmian w systemie: tworzenie, aktualizacja, usuwanie rekordów.
* **Klienci** – tworzenie, edycja, filtrowanie, usuwanie klientów.
* **Pożyczki** – tworzenie pożyczek z harmonogramem spłat, aktualizacja statusów, edycja warunków.
* **Płatności** – obsługa płatności, sprawdzanie kwot, aktualizacja harmonogramów, automatyczna zmiana statusów.
* **Raporty** – generowanie dokumentów PDF na podstawie tabel i filtrów.
* **Kopie zapasowe** – tworzenie i przywracanie kopii bazy danych za pomocą kontrolera **BackupController**.

**2.5. Realizacja złożonych zapytań i raportów dla użytkowników różnych poziomów**

Zapytania do bazy danych realizowane są przez **Entity Framework Core**. Złożone wybory wykonuje się za pomocą **LINQ** oraz metod filtrujących. Na przykład:

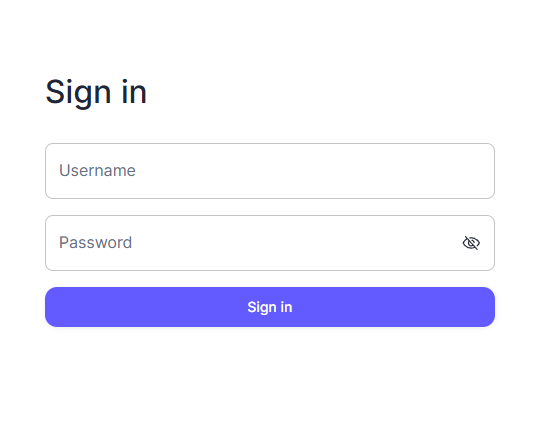
* Pobranie pożyczek klienta wraz ze wszystkimi płatnościami:  
  .Include(l => l.Payments).ThenInclude(p => p.PaymentType);
* Generowanie statystyk statusów pożyczek:  
  .GroupBy(l => l.Statusid).Select(g => new { Status = g.Key, Count = g.Count() });
* Wyszukiwanie logów po tabeli i akcji:  
  .Where(l => l.TableName == "Loans" && l.Action == "Updated");

Do generowania raportów w formacie PDF wykorzystano bibliotekę **DinkToPdf**. Stworzono serwis **PdfService**, który przyjmuje HTML, generuje z niego dokument i zwraca go jako strumień do pobrania. Układ tworzy się przez **StringBuilder** w odpowiednim serwisie.

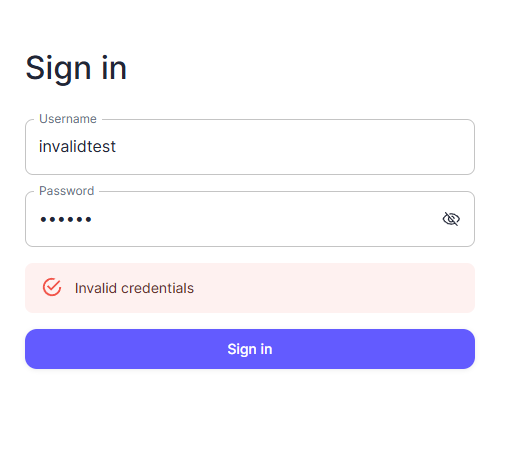
Dzięki elastyczności zapytań LINQ administrator może tworzyć raporty według okresów, działań użytkowników lub filtrów po encjach.

System wspiera złożoną logikę pobierania danych, dynamiczne tworzenie raportów oraz zapewnia pełną kontrolę nad kluczowymi procesami organizacji mikropożyczkowej

витоку бази.



**Rysunek 3.1 – Formularz logowania**

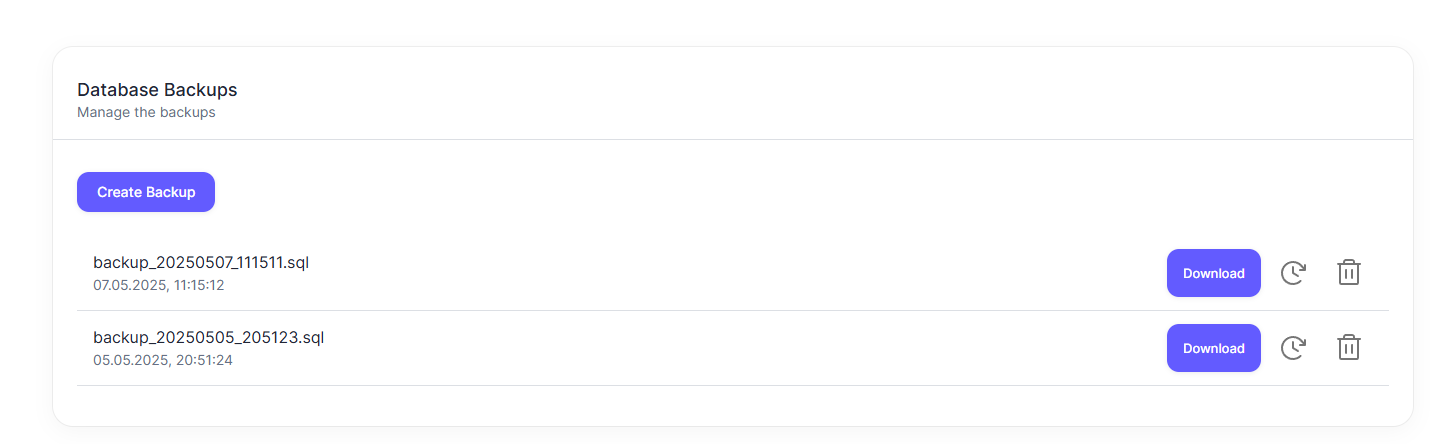


**Rysunek 3.2 – Formularz logowania przy nieprawidłowym wprowadzeniu danych**

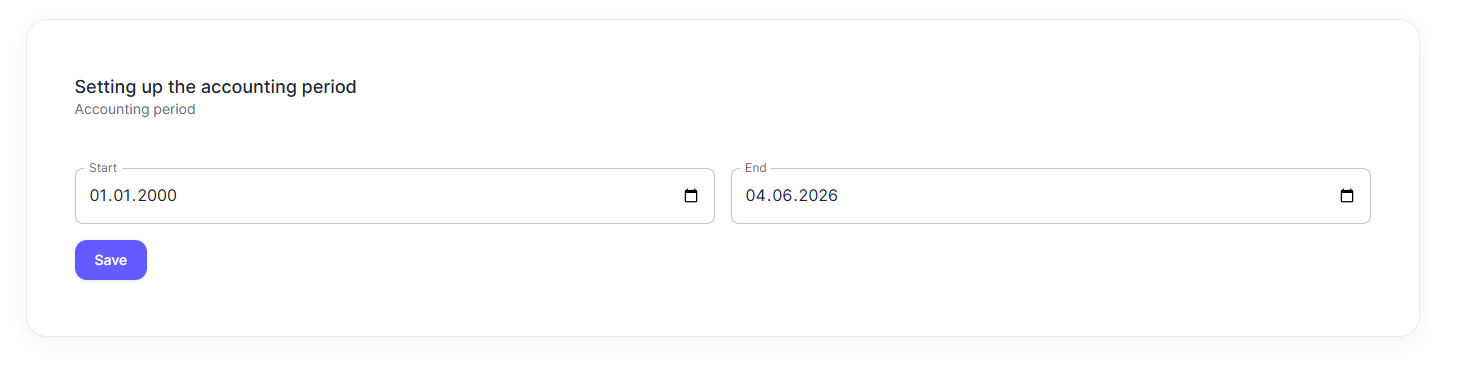
Proces uwierzytelniania jest realizowany poprzez wydanie tokenu JWT, który klient przesyła w nagłówku Authorization przy każdym żądaniu. Serwer waliduje token i na jego podstawie określa prawa dostępu użytkownika. Kontrolery API są chronione atrybutem [AuthorizeRole], który ogranicza dostęp wyłącznie do autoryzowanych użytkowników z odpowiednią rolą.

**3.2. Archiwizacja, przywracanie i inne funkcje serwisu oraz ustawienia**

W systemie zaimplementowano serwis tworzenia kopii zapasowych bazy danych, dostępny tylko dla administratorów. Na stronie zarządzania kopiami administrator może utworzyć nową kopię zapasową (plik .back), pobrać ją lokalnie lub wykonać przywracanie z istniejącego pliku. Realizowane jest to za pomocą poleceń SQL BACKUP DATABASE i RESTORE DATABASE, które wywoływane są przez część serwerową za pomocą procesu System.Diagnostics.Process.



**Rysunek 3.3 – Kontrola archiwizacji i przywracania danych**

Ustawienia systemu obejmują zarządzanie okresem rozliczeniowym oraz informacjami kontaktowymi. Okres rozliczeniowy wyznacza zakres dat, w którym dopuszczalna jest praca z klientami i kredytami. Jeśli bieżąca data wykracza poza ten okres, dostęp do głównej funkcjonalności jest blokowany dla ról innych niż administrator.

**Rysunek 3.4 – Kontrola okresu rozliczeniowego**

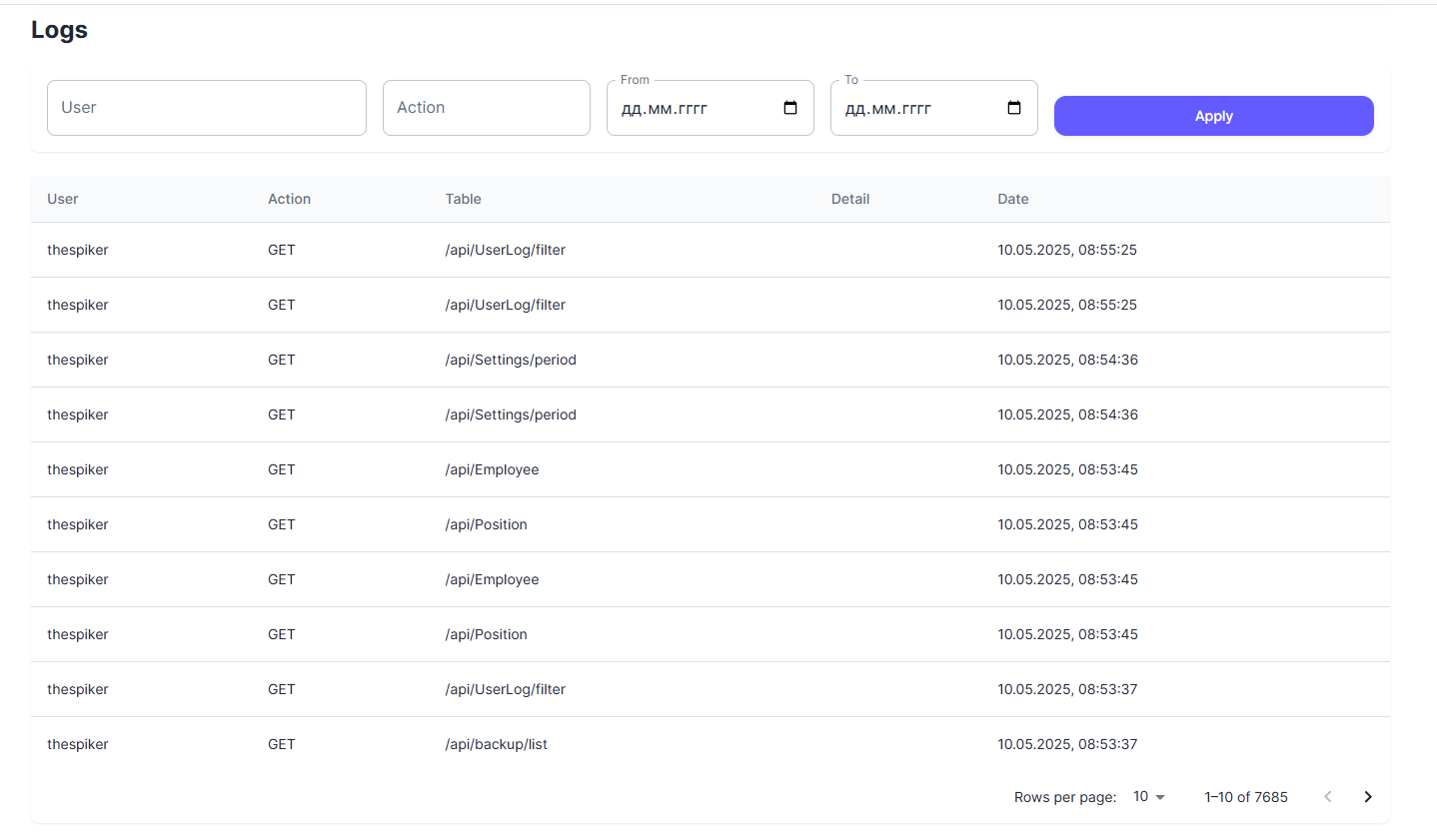
**3.3. Prowadzenie dziennika aktywności użytkowników systemu oraz jego przetwarzanie**

**Wszystkie kluczowe działania w systemie są rejestrowane w dzienniku aktywności (tabela UserLogs). Obejmuje to tworzenie, aktualizację i usuwanie klientów, kredytów, płatności, kopii zapasowych i innych. Do tego celu stworzono LogService, który jest wywoływany w odpowiednich serwisach podczas wykonywania akcji użytkownika.**

**Każdy wpis w dzienniku zawiera: user\_id, nazwę akcji, nazwę tabeli, szczegóły zmiany oraz znacznik czasu (timestamp). Dziennik jest tylko do odczytu i nie można go edytować.**

**Administrator może przeglądać dziennik działań na osobnej stronie, gdzie dostępne są filtry według typu akcji, daty oraz pracownika.**

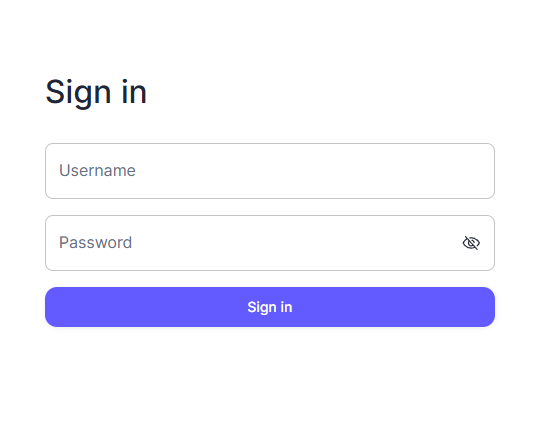
**Dziennik umożliwia audyt działań pracowników, śledzenie błędów oraz odtwarzanie chronologii zmian w bazie danych. Jest to krytycznie ważne dla systemu, który przetwarza dane finansowe i osobowe klientów.**



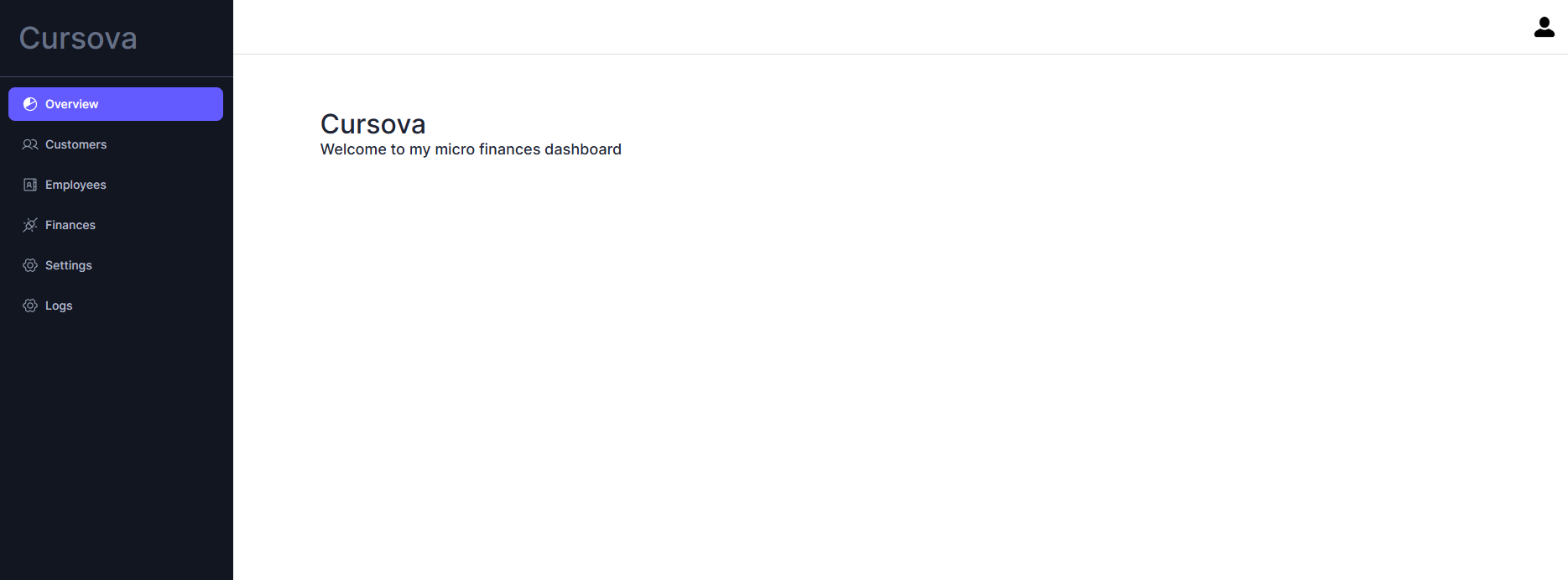
**Rysunek 3.5 – System logowania**

**4 INSTRUKCJE DLA UŻYTKOWNIKÓW WEDŁUG POZIOMU DOSTĘPU**

**Przed rozpoczęciem korzystania z systemu każdy użytkownik musi się uwierzytelnić, wprowadzając login oraz hasło na stronie logowania. Jeśli użytkownik nie jest zalogowany, system blokuje dostęp do jakiejkolwiek funkcjonalności. Po zalogowaniu system automatycznie rozpoznaje poziom dostępu użytkownika i otwiera odpowiedni panel.**



**Rysunek 4.1 – Formularz logowania do konta osobistego**

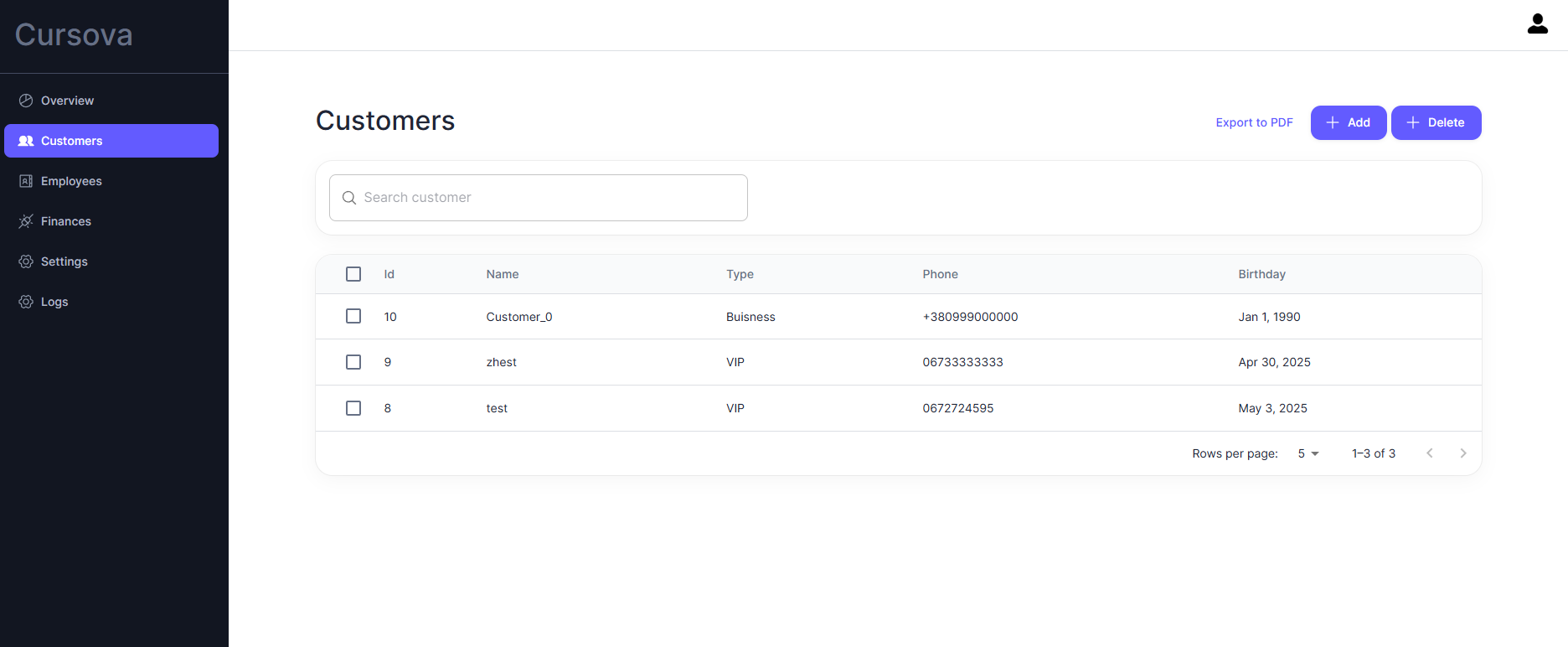


**Rysunek 4.2 – Strona widoczna po zalogowaniu do konta osobistego**

4.1 Instrukcje dotyczące użytkowania i obsługi dla administratora systemu

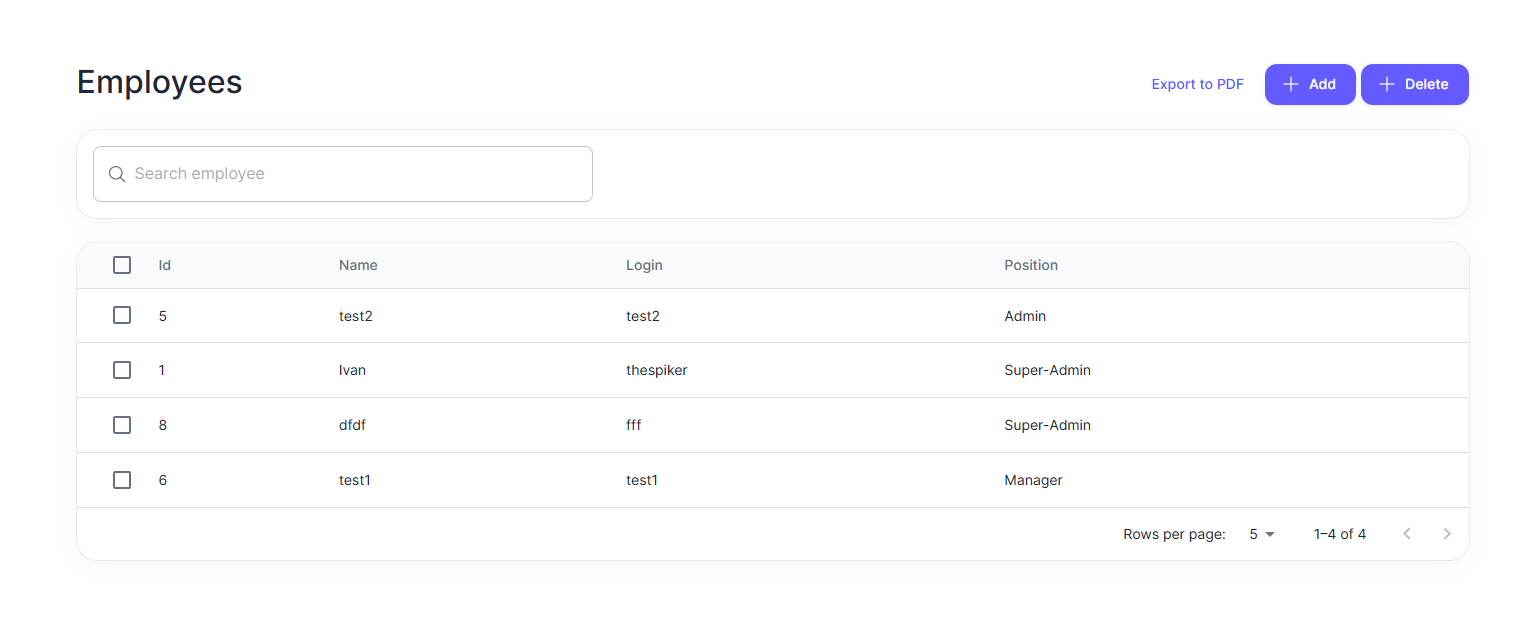
Po uwierzytelnieniu administrator trafia do głównego panelu sterowania, gdzie dostępne są rozszerzone funkcje zarządzania systemem. Interfejs administratora został zaprojektowany z naciskiem na wygodne zarządzanie krytycznymi danymi.  
Dostępne funkcje administratora:

* Użytkownicy. Administrator może tworzyć, edytować lub usuwać konta pracowników. Można przypisać każdemu użytkownikowi rolę oraz sprawdzić jego ostatnie działania. Interfejs obsługuje paginację i wyszukiwanie.

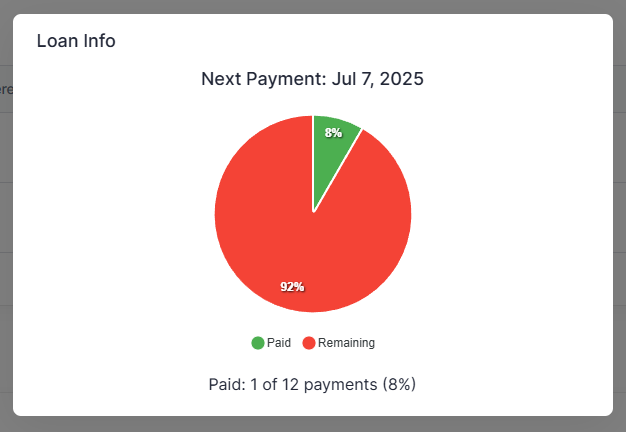


**Rysunek 4.3 – Strona zarządzania użytkownikami**

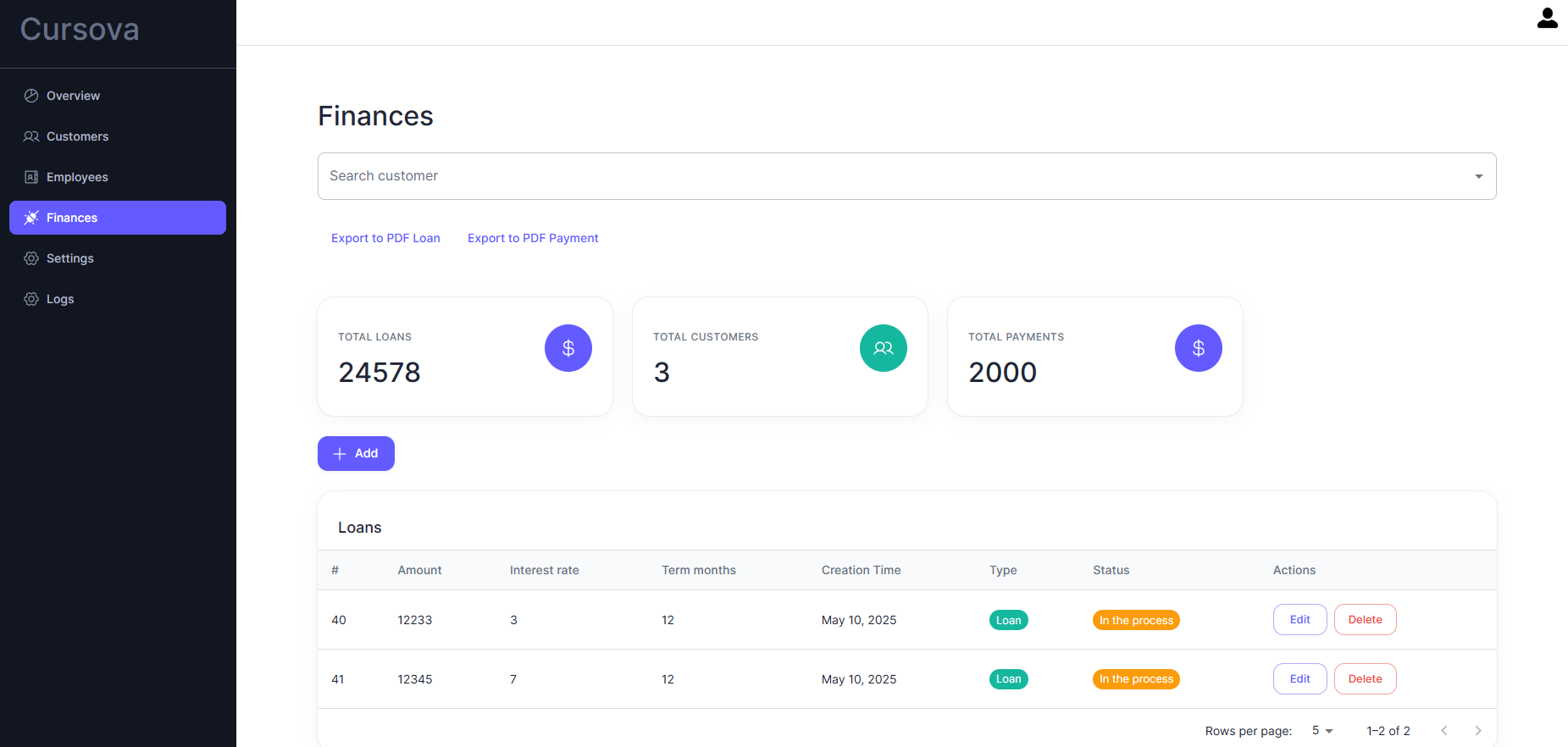
**Klienci.** Strona zarządzania klientami pozwala tworzyć nowe rekordy, aktualizować dane (imię i nazwisko, numer telefonu, typ klienta) oraz wykonywać wyszukiwanie i paginację.



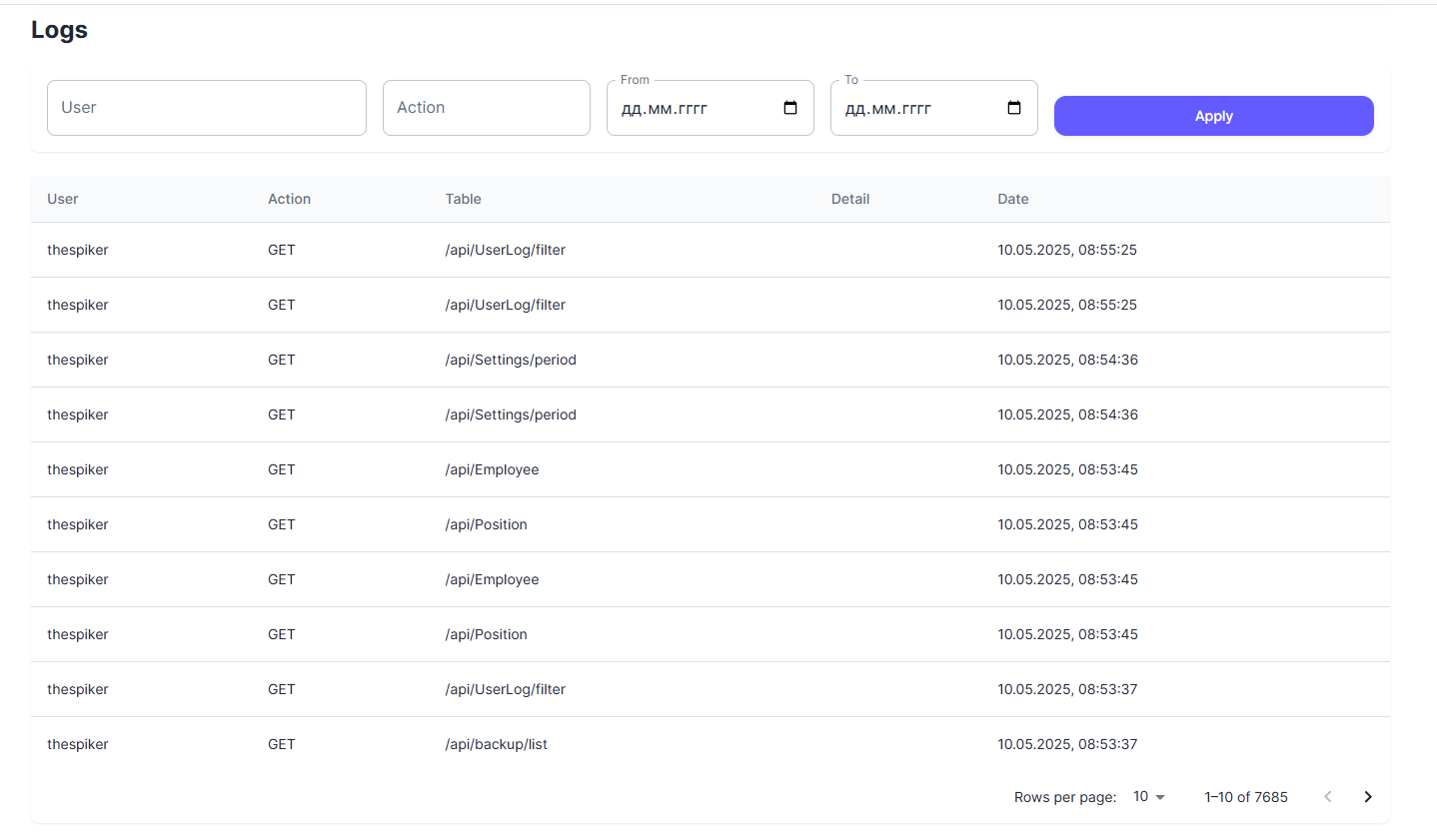
Rysunek 4.4 – Strona zarządzania klientami

**Kredyty.** Administrator ma dostęp do pełnej listy kredytów, które można tworzyć, edytować oraz przeglądać harmonogramy płatności. Każdy kredyt ma powiązaną historię płatności oraz status.

Rysunek 4.5 – Harmonogram płatności

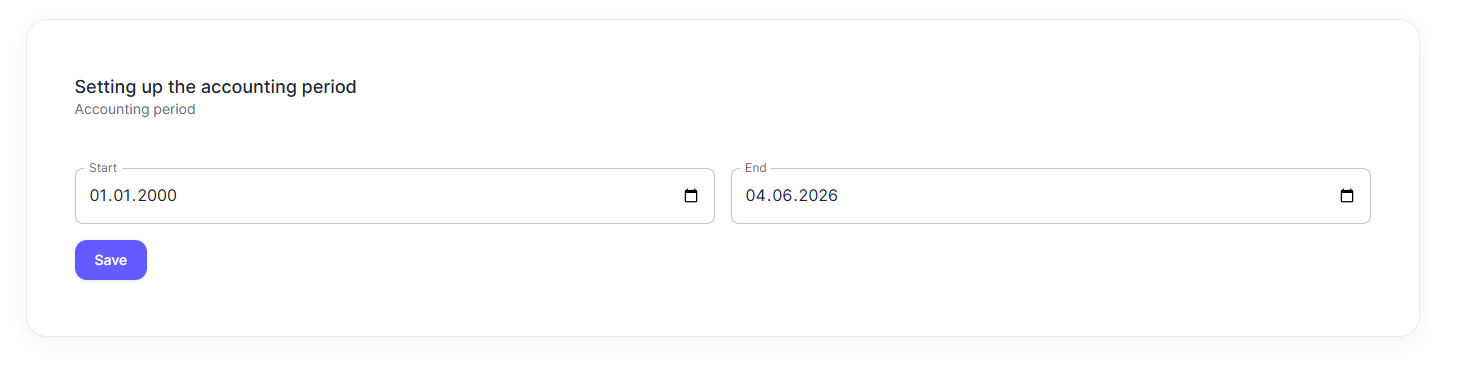
**Płatności.** Na stronie płatności administrator widzi wszystkie wpłaty, z możliwością edycji lub anulowania, a także aktualizacji harmonogramu spłaty kredytu. Dostępne jest również ręczne generowanie nowych płatności.

**Rysunek 4.6 – Strona zarządzania finansami**

**Dziennik zdarzeń.** Administrator widzi wszystkie zarejestrowane zdarzenia w systemie. Można wyszukiwać wpisy według e-maila, tabeli lub akcji oraz eksportować dziennik do formatu PDF

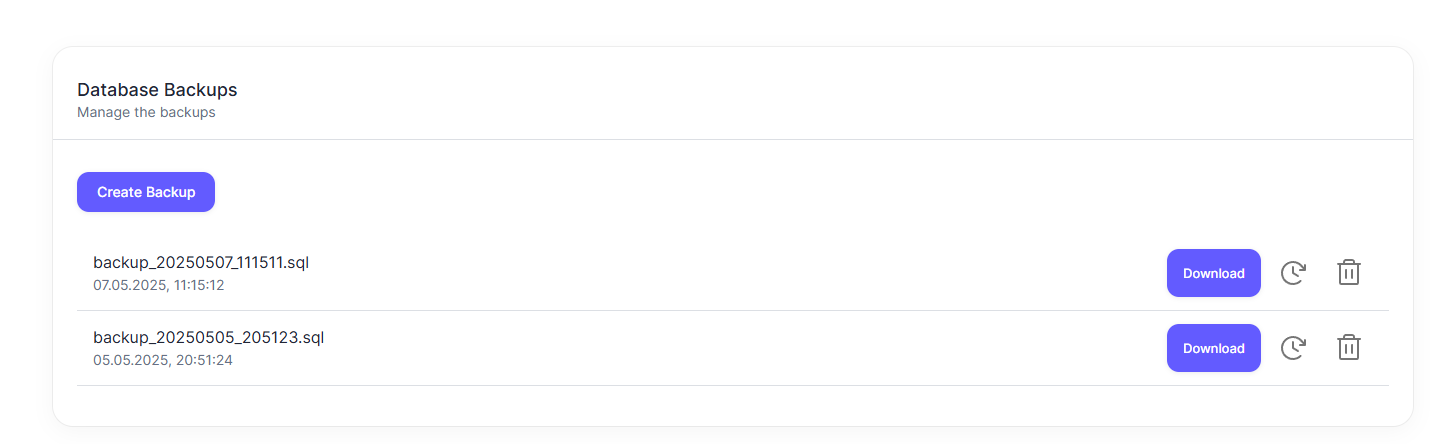
**Rysunek 4.7 – System logowania**

* **Okres rozliczeniowy.** W ustawieniach administrator definiuje okres, w którym dozwolona jest praca z systemem. Jeśli data wykracza poza ten zakres, pracownicy tracą dostęp do głównych funkcji.



Rysunek 4.8 – Kontrola okresu rozliczeniowego

* **Kopie zapasowe.** Strona do tworzenia, przeglądania i pobierania kopii zapasowych bazy danych. Możliwa jest również restauracja bazy danych z zapisanego pliku.

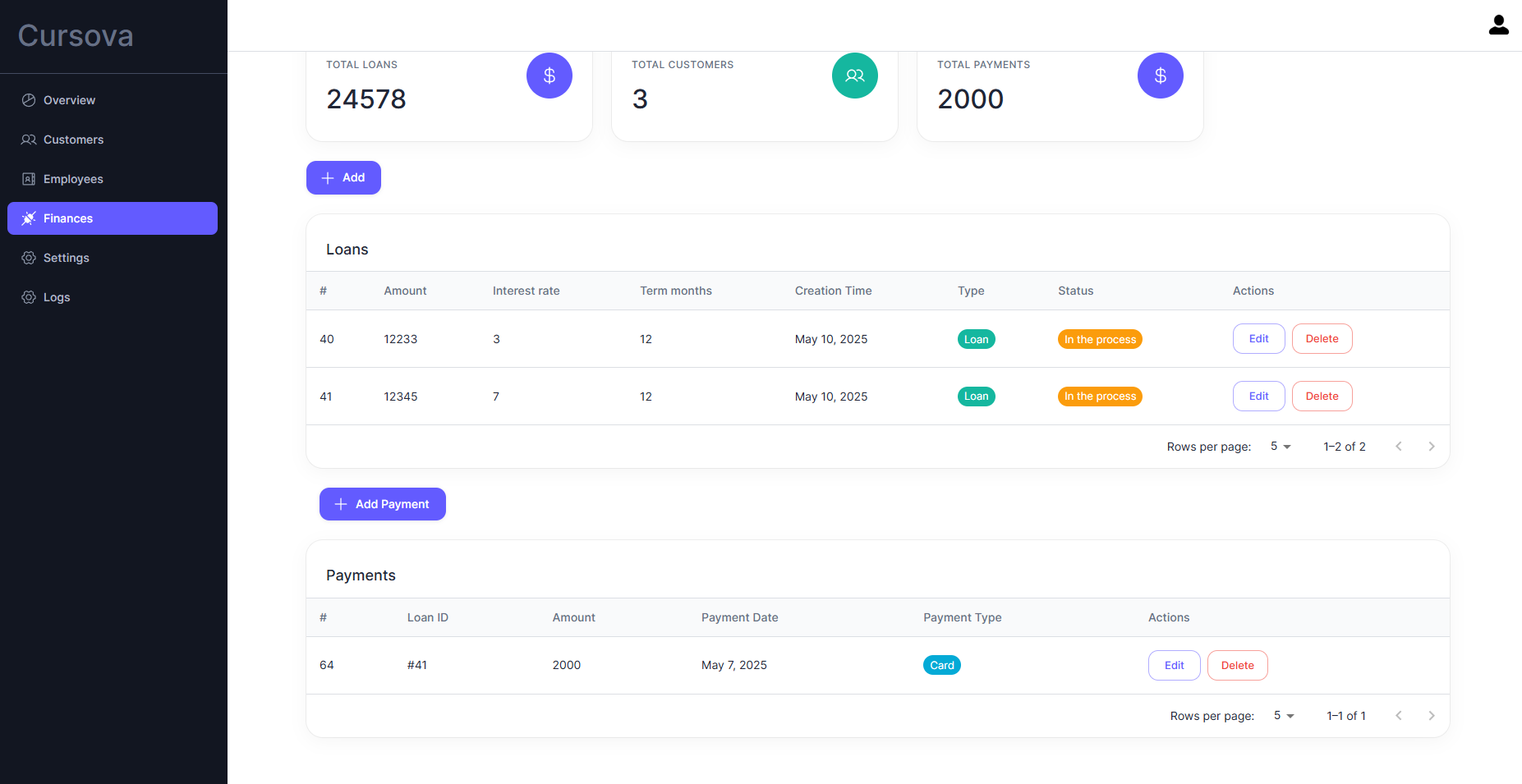


Rysunek 4.9 – Kontrola archiwizacji i przywracania danych

### 4.2 Instrukcje dotyczące użytkowania i obsługi dla pracowników

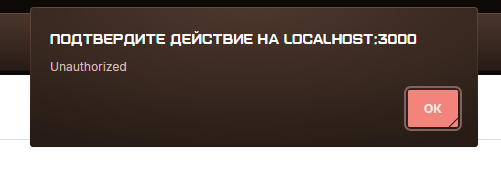
Po zalogowaniu się do systemu pracownik trafia do spersonalizowanego panelu z ograniczonym zestawem funkcji, skoncentrowanych na codziennej obsłudze klientów i wsparciu spraw kredytowych.  
Dostępne funkcje pracownika:

* **Praca z klientami.** Możliwość przeglądania listy klientów, wyszukiwania po imieniu i nazwisku lub numerze telefonu. Dostępne jest tworzenie nowych klientów, edytowanie istniejących rekordów, kontrola daty urodzenia, typu klienta oraz danych kontaktowych.
* **Tworzenie kredytów.** Pracownik może utworzyć kredyt dla wybranego klienta, określając kwotę, termin, typ i oprocentowanie. Po zapisaniu automatycznie generowany jest harmonogram płatności.



Rysunek 4.10 – Strona zarządzania finansami

* **Płatności.** Strona umożliwia dodawanie nowych płatności lub oznaczanie istniejących jako opłacone. Status płatności automatycznie aktualizuje harmonogram spłat. Można również przeglądać historię poprzednich płatności.
* **Ograniczenia dostępu.** Wszelkie próby pracownika dostępu do zasobów dostępnych tylko dla administratora (np. dziennik zdarzeń lub kopie zapasowe) są automatycznie blokowane wraz z komunikatem o braku odpowiednich uprawnień.



Rysunek 4.10 – Komunikat o braku uprawnień

W ten sposób system został zaprojektowany tak, aby każda kategoria użytkowników miała dostęp jedynie do funkcjonalności odpowiadającej ich roli w organizacji, co zapewnia bezpieczeństwo, integralność danych oraz kontrolę nad wszystkimi procesami

**WNIOSKI**

**W wyniku wykonanej pracy stworzono w pełni funkcjonalny system do automatyzacji działalności mikrofinansowej organizacji, który obejmuje pełny cykl życia pracy z klientami, kredytami i płatnościami. Oprogramowanie zostało zrealizowane z wykorzystaniem nowoczesnego stosu technologicznego: ASP.NET Core MVC dla części serwerowej, Entity Framework Core do komunikacji z bazą danych Microsoft SQL Server oraz Angular jako interfejs frontendowy (tam, gdzie jest to potrzebne). Podstawą systemu jest wzorzec architektoniczny MVC oraz wyraźny podział na serwisy, kontrolery i DTO, co zapewnia skalowalność i łatwość utrzymania kodu.**

**Podczas projektowania przeprowadzono pełną analizę dziedziny problemu, zidentyfikowano główne encje – klientów, kredyty, płatności, pracowników, harmonogramy płatności oraz działania użytkowników. Wszystkie dane zostały zorganizowane w strukturalny model relacyjny z wielopoziomową kontrolą dostępu. Dla każdego typu użytkownika – menedżera, administratora i superadministratora – zaimplementowano unikalne scenariusze dostępu odpowiadające poziomowi odpowiedzialności.**

**W systemie zrealizowano:  
• Ewidencję klientów i pracowników z rozróżnieniem według typów;  
• Tworzenie kredytów z automatycznym generowaniem harmonogramu płatności;  
• Rejestrację płatności z częściową lub pełną spłatą oraz aktualizacją statusów;  
• Logowanie działań użytkowników za pomocą serwisu LogService;  
• Tworzenie raportów PDF dotyczących płatności, kredytów oraz dziennika zdarzeń;  
• Tworzenie kopii zapasowych bazy danych oraz jej przywracanie z poziomu interfejsu administratora.**

**W celu zapewnienia bezpieczeństwa zaimplementowano hashowanie haseł, uwierzytelnianie oparte na rolach oraz ograniczenia działań użytkowników zgodnie z ich uprawnieniami. Wprowadzono także ograniczenia związane z okresem rozliczeniowym, które umożliwiają elastyczne blokowanie pracy w systemie poza dozwolonym czasem.**

**System przeszedł testy w warunkach rzeczywistego obciążenia, w tym scenariusze z setkami jednoczesnych żądań, co wykazało jego stabilność, wydajność i gotowość do skalowania.**

**Podsumowując, opracowane oprogramowanie spełnia postawione wymagania funkcjonalne i niefunkcjonalne, wykazuje wysoki poziom bezpieczeństwa, elastyczności i wygody dla użytkowników końcowych oraz może być skutecznie wdrożone w praktyce mikrofinansowych firm.**

# Wykorzystanie Swaggera do dokumentowania API

# W celu wygodnego testowania oraz dokumentowania usług sieciowych w projekcie zaimplementowano Swagger (OpenAPI). Swagger automatycznie generuje interaktywną dokumentację REST API na podstawie atrybutów kontrolerów ASP.NET Core. Interfejs Swaggera jest dostępny w trybie deweloperskim pod adresem: https://localhost:{port}/swagger

# Rozwiązanie to umożliwia:

# Przeglądanie wszystkich dostępnych endpointów (żądań HTTP);

# Zapoznanie się z modelami wejściowymi i wyjściowymi (DTO);

# Testowanie zapytań bezpośrednio z poziomu przeglądarki;

# Sprawdzanie autoryzacji przy użyciu tokena JWT.

# Główne grupy endpointów i ich przeznaczenie:

# 1. Uwierzytelnianie i autoryzacja

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| POST | /api/auth/login | Logowanie użytkownika i generowanie tokena JWT |
| POST | /api/auth/register | Rejestracja nowego użytkownika (dla administratora) |

# 2. Klienci

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/customers | Pobieranie listy klientów |
| POST | /api/customers | Dodawanie nowego klienta |
| PUT | /api/customers/{id} | Edycja danych klienta |
| DELETE | /api/customers/{id} | Usuwanie klienta |

# 3. Kredyty

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/loans | Pobieranie wszystkich kredytów |
| POST | /api/loans | Tworzenie nowego kredytu |
| GET | /api/loans/{id} | Szczegóły kredytu |
| DELETE | /api/loans/{id} | Usuwanie kredytu |

# 4. Płatności

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/payments | Pobieranie listy płatności |
| POST | /api/payments | Dodawanie nowej płatności |
| GET | /api/payments/{id} | Szczegóły płatności |

# 5. Harmonogramy płatności

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/paymentschedules | Przeglądanie harmonogramów płatności |
| GET | /api/paymentschedules/{id} | Szczegóły harmonogramu płatności |

# 6. Użytkownicy i pracownicy

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/users | Lista użytkowników systemu |
| POST | /api/users | Dodawanie nowego użytkownika |
| PUT | /api/users/{id} | Zmiana roli lub statusu |
| DELETE | /api/users/{id} | Usuwanie użytkownika |

# 7. Dziennik działań użytkowników

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/userlogs | Przegląd działań użytkowników w systemie |

# 8. Kopia zapasowa i okresy księgowe

| Metoda | Ścieżka | Opis |
| --- | --- | --- |
| GET | /api/accountingperiod | Sprawdzanie aktywnego okresu księgowego |
| PUT | /api/accountingperiod | Zmiana okresu księgowego |

# 9. Raporty i generacja PDF

# Raporty dotyczące kredytów, płatności oraz dziennika działań użytkowników generowane są za pomocą osobnych żądań, obsługiwanych przez odpowiednie serwisy. Do tworzenia plików PDF wykorzystano bibliotekę DinkToPdf.

# Swagger obsługuje również autoryzację poprzez token JWT — aby uzyskać dostęp do chronionych endpointów, użytkownik musi wprowadzić token w polu „Authorize” w interfejsie Swaggera. Bez ważnego tokena dostęp do większości zasobów jest zablokowana