Bazy danych I

Biblioteka — dokumentacja projektu

Rok akademicki 2023/2024

1. Projekt koncepcji, założenia. 3
   a. Definicja tematu projektu 3
   b. Analiza wymagań użytkownika 3
   c. Zaprojektowanie funkcji 3
2. Projekt diagramów (konceptualny) 4
   a. Budowa i analiza diagramu przepływu danych (DFD) 4
   b. Zdefiniowanie encji oraz ich atrybutów 4
   c. Zaprojektowanie relacji pomiędzy encjami 7
3. Projekt logiczny 8
   a. Projektowanie tabel, kluczy, indeksów 8
   b. Słowniki danych 9
   c. Analiza zależności funkcyjnych i normalizacja tabel 10
   d. Zaprojektowanie operacji na danych 11
4. Projekt funkcjonalny 23
   a. Interfejsy do prezentacji, edycji i obsługi danych 23
   b. Wizualizacja danych 26
   c. Zdefiniowanie panelu sterowania aplikacji 27
   d. Makropolecenia 29
5. Dokumentacja 29
   a. Wprowadzanie danych 29
   b. Dokumentacja użytkownika. 29
   c. Opracowanie dokumentacji technicznej. 30
   d. Wykaz literatury 31

1. Projekt koncepcji, założenia.

a. Definicja tematu projektu

Tematem projektu jest projekt bibliotecznej bazy danych z uwzględnieniem odpowiedniego interfejsu, który z nią się komunikuje. Oprócz uzyskiwania danych o poszczególnych pozycjach bibliotecznych, przewidziana jest również funkcjonalność związana z wypożyczaniem kolejnych pozycji, przedłużaniem ich oraz oddawaniem. Jest to niejako główny cel tego projektu, choć w jego ramach możliwe jest również wyliczenie zadłużenia wynikającego z przetrzymania, uzyskanie informacji o książkach, które są obecnie wypożyczone, a także wgląd do wybranych rankingów (jak np. najchętniej czytany autor).

b. Analiza wymagań użytkownika

Użytkownik, korzystając z bazy danych, powinien mieć możliwość dodania nowego wypożyczenia, przedłużenia lub zakończenia obecnego, powinien również mieć wgląd do listy czytelników, listy książek, listy pozycji; oprócz tego użytkownik powinien móc generować informacje o zadłużeniu poszczególnego czytelnika, a także móc to zadłużenie wyzerować. Przewidziana jest również możliwość dodania nowego czytelnika i usunięcia starego (wyrejestrowanie z biblioteki, usunięcie danych osobowych).

c. Zaprojektowanie funkcji

W bazie danych realizowane są funkcje:

- wypożyczenie — realizowane jest sprawdzanie wypożyczenia — czytelnik nie może wypożyczyć więcej niż 3 pozycje oraz nie może dokonać wypożyczenia, jeśli jego zadłużenie jest większe niż 50zł; walidacja danych: nie można wypożyczyć już wypożyczonej pozycji, ponadto licznik liczby dostępnych egzemplarzy zmniejsza się o 1. Wypożyczenia dokonuje się na 30 dni. Można wypożyczyć daną pozycję powtórnie, jeśli wcześniej się ją oddało.
- oddanie — wraz z nim obliczane jest zadłużenie (jeśli występuje przetrzymanie pozycji), licznik dostępnych egzemplarzy się zwiększa o 1. Walidacja danych: nie można oddać niewypożyczonej pozycji.
- przedłużenie — termin zwiększa się o kolejne 30 dni. Przedłużyć jedną pozycję można maksymalnie trzy razy.
- liczenie zadłużenia — dokonuje się przy oddaniu pozycji, naliczana jest złotówka za każde dwa dni. Zadłużenie można sprawdzić dla każdego czytelnika oddzielnie
- pełne informacje o książce — uzyskanie informacji o autorach i kategoriach danej książki

- dodanie użytkownika, wyzerowanie mu zadłużenia, usunięcie użytkownika
- obliczanie liczby obecnie wypożyczonych przez użytkownika pozycji

Funkcje realizowane są przy użyciu funkcji składowanych (język PL/PGSQL), wyzwalaczy, podzapytań, widoków, wyrażeń CTE, a także wyrażeń warunkowych CASE.

2. Projekt diagramów (konceptualny)

a. Budowa i analiza diagramu przepływu danych (DFD)

Przepływ danych odbywa się na zasadzie wyboru odpowiednich, wcześniej zdefiniowanych zapytań (przegląd podstawowych relacji, wywołanie rankingów) lub na przekazaniu odpowiednich parametrów do przygotowanych funkcji.
Dane wejściowe przekazujemy wówczas, gdy chcemy uzyskać informacje o jakiejś konkretnej książce (będzie to identyfikator książki), gdy chcemy dokonać wypożyczenia (identyfikator użytkownika, identyfikator pozycji, data), sprawdzić czyjeś zadłużenie lub przedłużyć komuś daną pozycję.
Przekazując dane wejściowe, posługujemy się identyfikatorami w celu uproszczenia zapytań, a także łatwiejszej walidacji danych. Dane wejściowe są walidowane, przekazywane do odpowiednio przygotowanych zapytań lub funkcji, po czym zwracają dane wyjściowe w postaci tekstu pojawiającego się w odpowiednim oknie — wyniku zapytania do bazy. Oprócz wyniku pojawiają się również informacje zwrotne, przykładowo — czy udało się dokonać wypożyczenia, a jeśli nie, to z jakiego powodu.
Dane wprowadzane przez użytkownika są przechowywane, gdy tworzony jest nowy rekord — w przypadku wprowadzenia nowego użytkownika do serwisu, utworzenia nowego wypożyczenia, przedłużenia, a także obliczenia zadłużenia. W innych przypadkach dane wejściowe są zapominane zaraz po zakończeniu operacji.
Przechowywana jest historia wypożyczeń, lecz nie przechowuje się historii zadłużenia po jego umorzeniu.
Elementy decydujące o przepływie danych to w większości pola tekstowe i przyciski, zdefiniowane szerzej w interfejsie.

b. Zdefiniowanie encji oraz ich atrybutów

Przy projekcie bazy danych zdefiniowano następujące encje:

czytelnik — przechowuje rekordy dotyczące danego czytelnika

jego atrybuty to:

- czytelnik_id — **klucz główny** encji typu SERIAL NOT NULL
- imie — imię czytelnika, typu varchar(32) NOT NULL
- nazwisko — typu varchar(32) NOT NULL
- zadluzenie — typu int4

ksiazka — przechowuje rekordy dotyczące danej książki
atrybuty:

- ksiazka_id — **klucz główny** encji typu SERIAL NOT NULL
- nazwa — czyli tytuł książki typu varchar(128) NOT NULL
- liczba_egzemplarzy — typu int4

autor — przechowuje informacje o autorach

atrybuty:

- autor_id — **klucz główny** encji typu SERIAL NOT NULL
- nazwa — czyli imię i nazwisko autora, może być podane również przez skrót imienia, np. 'J.R.R. Tolkien', lub 'J.K. Rowling'
- data_urodzenia — dokładna lub przybliżona data urodzenia danego autora typu DATE NOT NULL

między encjami **ksiazka** i **autor** istnieje związek logiczny wiele do wielu (M:N). Realizacja tego związku odbywa się przez zdefiniowanie encji asocjacyjnej:
ksiazka_autor

atrybuty:

- ksiazka_id — **klucz obcy,** odwołuje się do klucza encji **ksiazka**
- autor_id —**klucz obcy,** odwołuje się do klucza głównego encji **autor**

Para atrybutów (ksiazka_id, autor_id) jest kluczem głównym.

kategoria — przechowuje informacje o kategoriach.

atrybuty:

- kategoria_id — **klucz główny** encji typu SERIAL NOT NULL
- kat_nazwa — nazwa kategorii typu varchar(32) NOT NULL

między encjami **ksiazka** i **kategoria** istnieje związek logiczny wiele do wielu (M:N). Realizacja tego związku odbywa się przez zdefiniowanie encji asocjacyjnej:
ksiazka_kategoria

atrybuty:

- ksiazka_id — **klucz obcy,** odwołuje się do klucza encji **ksiazka**
- kategoria_id —**klucz obcy,** odwołuje się do klucza głównego encji **kategoria**

Para atrybutów (ksiazka_id, kategoria_id) jest kluczem głównym.

wydawnictwo — przechowuje informacje o wydawnictwach.

atrybuty:

- wydawnictwo_id — klucz główny typu SERIAL NOT NULL
- wyd_nazwa — nazwa wydawnictwa typu varchar(32) NOT NULL

pomiędzy encjami **wydawnictwo** i **ksiazka** istnieje związek logiczny wiele do wielu (M:N). Realizacja tego związku w encji **pozycja**. Nie jest to jednak encja czysto asocjacyjna.

pozycja — czyli fizyczna reprezentacja danej książki, wydana przez określone wydawnictwo
atrybuty:

- pozycja_id — klucz główny typu SERIAL NOT NULL
- ksiazka_id — klucz obcy odwołujący się do encji **ksiazka** typu int4 NOT NULL
- wydawnictwo_id — klucz obcy odwołujący się do encji **wydawnictwo** typu int4 NOT NULL
- rok_wydania — informuje o roku wydania danej pozycji, typu int4 NOT NULL
- isborrowed — przechowuje informację o tym, czy pozycja jest pożyczona, czy nie, typu bool

  Para atrybutów (ksiazka_id, wydawnictwo_id) mogłaby być kluczem kandydującym, lecz podczas projektowania bazy okazało się, że wówczas wydawnictwo nie mogłoby kilka razy wydać tej samej pozycji, co okazało się kluczową przeszkodą, gdyż wówczas baza danych nie mogłaby oddać realnie stanu rzeczywistego bibliotek.

Między encjami **czytelnik** i **pozycja** znajduje się relacja **wypozyczenie** realizująca główną funkcjonalność biblioteki, czyli wypożyczanie. Encje **czytelnik** i **pozycja** są w związku wiele-do-wielu (M:N).

wypozyczenie — realizuje wypozyczenie oraz przechowuje kluczowe informacje dotyczące danego wypożyczenia

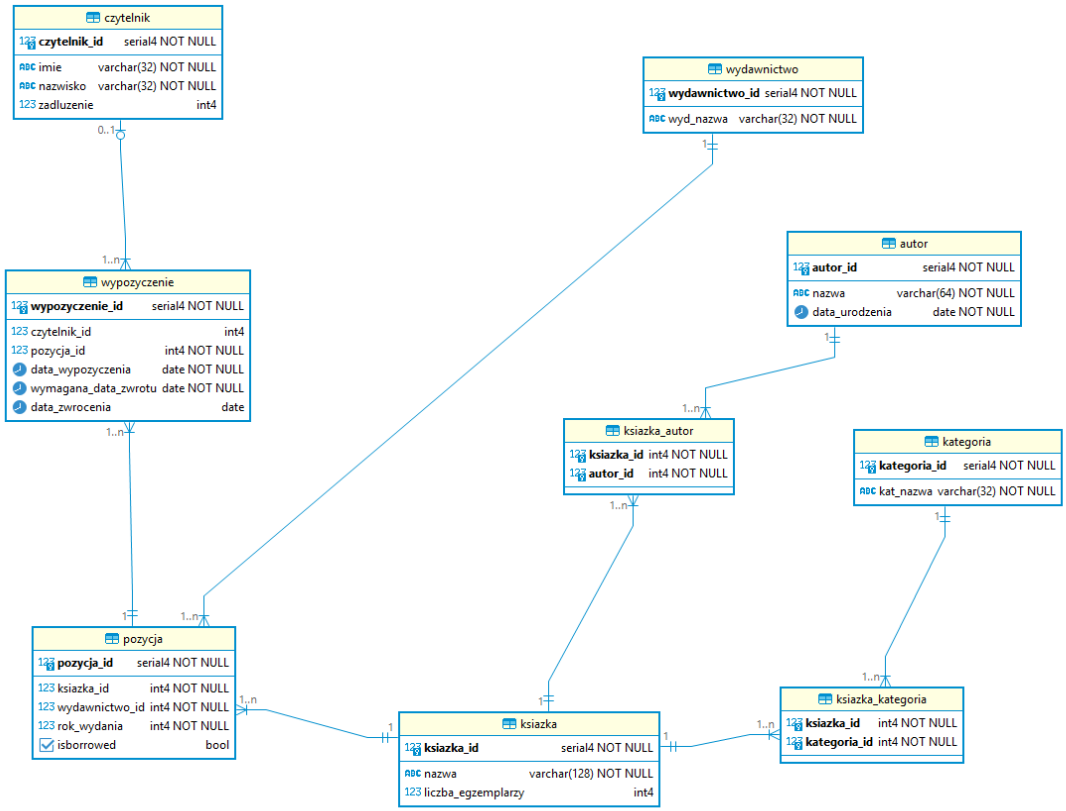
atrybuty:

- wypozyczenie_id — **klucz główny** typu SERIAL NOT NULL
- czytelnik_id — **klucz obcy** odwołujący się do encji **czytelnik**, typu int4
- pozycja_id — **klucz obcy** odwołujący się do encji **pozycja**, typu int4 NOT NULL
- data_wypozyczenia — typu DATE NOT NULL
- wymagana_data_zwrotu — data, do której trzeba zwrócić pozycję, domyślnie 30 dni od daty wypożyczenia, typu DATE NOT NULL
- data_zwrocenia — faktyczna data zwrócenia pozycji, jeśli jest NULL, to oznacza, że pozycja jest w tym momencie wypożyczona, typu DATE

czytelnik_id we właściwościach klucza obcego ma ustawione ON DELETE SET NULL, w wypadku jakiegoś czytelnika, pozostaje jego historia wypożyczeń (rekord nie jest usuwany).

c. Zaprojektowanie relacji pomiędzy encjami

Związki i definicje kluczy zostały podane podczas definiowania encji. W projekcie występuje wiele relacji wiele do wielu — jeden czytelnik może pożyczyć klika książek, a jedna pozycja może być pożyczona przez wielu czytelników (choć w różnym czasie).
Podobnież jedna książka może być wydana przez wiele wydawnictw, a jedno wydawnictwo może wydać wiele książek; tak samo można założyć, że jedna książka może mieć wiele tytułów i kategorii.
Projektując relacje, uwzględniono również pewien poziom abstrakcji: o ile książka jest tworem w pewnym sensie abstrakcyjnym, o tyle pozycja jest fizyczną reprezentacją, pewnym konkretnym egzemplarzem leżącym gdzieś na półce bibliotecznej lub u czytelnika w domu.



Rysunek 1. ERD projektu bibliotecznej bazy danych.

3. Projekt logiczny

a. Projektowanie tabel, kluczy, indeksów

Struktura bazy danych została zaprojektowania w oparciu o wyżej przyjęte założenia i z wykorzystaniem wyżej podanych typów danych (INT4, VARCHAR, DATE, SERIAL, z użytymi kwalifikatorami NOT NULL).

Wszystkie tabele zostały zawarte w ramach konkretnego *schematu* **biblioteka**. Użyty został System Zarządzania Bazą Danych PostgreSQL.

Projekt bazy danych w języku SQL:

```
create schema biblioteka;

create table biblioteka.ksiazka(
      ksiazka_id SERIAL primary key,
      nazwa VARCHAR(128) not null,
      liczba_egzemplarzy int
);

create table biblioteka.autor(
      autor_id SERIAL primary key,
      nazwa VARCHAR(64) not null,
      data_urodzenia DATE not null
);

create table biblioteka.ksiazka_autor(
      ksiazka_id int not null,
      autor_id int not null,
      CONSTRAINT ksiazka_autor_pk primary key(ksiazka_id, autor_id),
      CONSTRAINT ksiazka_autor_autor_id_fk foreign key (autor_id)
references biblioteka.autor(autor_id),
      CONSTRAINT ksiazka_autor_ksiazka_id_fk foreign key (ksiazka_id)
references biblioteka.ksiazka(ksiazka_id)
);

create table biblioteka.kategoria(
      kategoria_id SERIAL primary key,
      kat_nazwa varchar(32) not null
);
create table biblioteka.ksiazka_kategoria(
      ksiazka_id int not null,
      kategoria_id int not null,
      constraint ksiazka_kategoria_pk primary key (ksiazka_id,
kategoria_id),
      constraint ksiazka_kategoria_ksiazka_id_fk foreign key (ksiazka_id)
references biblioteka.ksiazka(ksiazka_id),
      constraint ksiazka_kategoria_kategoria_id_fk foreign key
(kategoria_id) references biblioteka.kategoria(kategoria_id)
);

create table biblioteka.wydawnictwo(
      wydawnictwo_id serial primary key,
      wyd_nazwa varchar(32) not null
```

```
);

create table biblioteka.pozycja(
	pozycja_id SERIAL primary key,
	ksiazka_id int not null,
	wydawnictwo_id int not null,
	rok_wydania int not null,
	isBorrowed boolean,
	constraint pozycja_ksiazka_id_fk foreign key (ksiazka_id) references
biblioteka.ksiazka(ksiazka_id),
	constraint pozycja_wydawnictwtwo_id_fk foreign key (wydawnictwo_id)
references biblioteka.wydawnictwo(wydawnictwo_id)
);

create table biblioteka.czytelnik(
	czytelnik_id serial primary key,
	imie varchar(32) not null,
	nazwisko varchar(32) not null,
	zadluzenie int
);


create table biblioteka.wypozyczenie(
	wypozyczenie_id SERIAL primary key,
	czytelnik_id int not null,
	pozycja_id int not null,
	data_wypozyczenia date not null,
	wymagana_data_zwrotu date not null,
	data_zwrocenia date,
	constraint wypozyczenie_czytelnik_id_fk foreign key(czytelnik_id)
references biblioteka.czytelnik(czytelnik_id) on delete set null,
	constraint wypozyczenie_pozycja_id_fk foreign key(pozycja_id)
references biblioteka.pozycja(pozycja_id) on delete set null
);
```

b. Słowniki danych

czytelnik	**typ**
czytelnik_id	SERIAL4 NOT NULL
imie	VARCHAR(32) NOT NULL
nazwisko	VARCHAR(32) NOT NULL
zadluzenie	INT4

wypozyczenie	**typ**
wypozyczenie_id	SERIAL4 NOT NULL
czytelnik_id	INT4
pozycja_id	INT4 NOT NULL
data_wypozyczenia	DATE NOT NULL
wymagana_data_zwrotu	DATE NOT NULL
data_zwrocenia	DATE

pozycja	typ
pozycja_id	SERIAL4 NOT NULL
ksiazka_id	INT4 NOT NULL
wydawnictwo_id	INT4 NOT NULL
rok_wydania	INT4 NOT NULL
isborrowed	BOOL

ksiazka	typ
ksiazka_id	SERIAL4 NOT NULL
nazwa	VARCHAR(128) NOT NULL
liczba_egzemplarzy	INT4

wydawnictwo	typ
wydawnictwo_id	SERIAL4 NOT NULL
wyd_nazwa	VARCHAR(32) NOT NULL

autor	typ
autor_id	SERIAL4 NOT NULL
nazwa	VARCHAR(32) NOT NULL
data_urodzenia	DATE NOT NULL

kategoria	typ
kategoria_id	SERIAL4 NOT NULL
kat_nazwa	VARCHAR(32) NOT NULL

Encje czysto asocjacyjne zostały pominięte, każdy typ danych tam to INT4 NOT NULL.
Dziedzina narzuca dosyć duże ograniczenia: w większości dane są przechowywane przez INT4, czy też INTEGER: czyli zakres od -2147483648 do +2147483647, choć nawet w obrębie tej dziedziny interesują nas raczej wartości dodatnie.
Nazwy kategorii, autorów czy czytelników są ograniczone przez VARCHAR o maksymalnej długości 32 znaków. Jedyny wyjątek zaistniał przy tytule książki — maksymalnie może być on długi na 128 znaków. Używany jest również typ DATE, tam gdzie konieczne jest takie ograniczenie (czyli daty), oraz typu BOOL, który narzuca zero-jedynkowe ograniczenie na dany atrybut. Wszystkie użyte typy danych w tym systemie zarządzania bazą danych są więc skalarne i niepodzielne.

c. Analiza zależności funkcyjnych i normalizacja tabel

W tym podpunkcie dokonamy sprawdzenia zgodności z normą 3NF.

Aby schemat spełniał normę 3NF, musi też spełniać normę 2NF i 1NF.
Z założenia wyjściowe dziedziny mają być skalarne i niepodzielne, co potwierdziliśmy w poprzednim podpunkcie dokumentacji.

Relacja jest w pierwszej postaci normalnej, gdy każdy atrybut niekluczowy funkcyjnie zależy od klucza głównego.
I faktycznie, w każdej relacji klucz główny jednoznacznie definiuje pozostałe atrybuty: w miejscu, gdzie tak mogłoby nie być (wypozyczenie) wprowadziliśmy dodatkowo identyfikator wypożyczenia jako klucz główny.
Schemat spełnia normę 1NF.
Czy spełnia jednak normę 2NF?
wypozyczenie: {wypozyczenie_id}
F: {wypozyczenie_id} —> {czytelnik_id, pozycja_id, data_wypozyczenia, wymagana_data_zwrotu, data_zwrocenia}

Norma 2NF zachodzi, gdy zachodzi norma 1NF, a ponadto atrybuty wchodzące w skład klucza głównego są w pełni funkcjonalnie zależne od całego klucza głównego, a nie tylko od jego części.

W bibliotecznej bazie danych nie wykorzystujemy kluczy złożonych z kilku atrybutów, więc z pewnością norma 2NF jest spełniona.

Relacja jest w 3NF wtedy i tylko wtedy, gdy wszystkie **niekluczowe** atrybuty są wzajemnie niezależne oraz nieredukowalnie zależne od klucza głównego.

Analizując atrybuty w poszczególnych encjach, w szczególności w encjach **wypozyczenie** i **pozycja**, można dojść do wniosku, że niekluczowe atrybuty rzeczywiście są wzajemnie niezależne od siebie. Każdy atrybut można by było bowiem zaktualizować niezależnie od innych.
Każdy niekluczowy atrybut zależy również nieprzechodnie od klucza głównego
wypozyczenie:
F: {wypozyczenie_id} —> {czytelnik_id, pozycja_id, data_wypozyczenia, wymagana_data_zwrotu, data_zwrocenia}
{czytelnik_id} ----//--> {pozycja_id} *(nie zależy)*
{pozycja_id} ------//--> {wymagana_data_zwrotu}
itd.…

Można więc stwierdzić, że schemat biblioteki spełnia normę 3NF.

d. Zaprojektowanie operacji na danych

W założeniach projektu zdefiniowaliśmy następujące funkcjonalności:
wypożyczenie, oddanie, przedłużenie, liczenie zadłużenia, pełne informacje o książce, dodanie użytkownika, wyzerowanie zadłużenia, usunięcie użytkownika, obliczenie liczby wypożyczonych obecnie pozycji.

Zostały zdefiniowane odpowiednie kwerendy w języku SQL.

- wypożyczenie — odbywa się przez wstawienie do tabeli wypozyczenie odpowiedniego rekordu:

```
insert into biblioteka.wypozyczenie (czytelnik_id, pozycja_id,
data_wypozyczenia, wymagana_data_zwrotu)
values (
      1, 4, TO_DATE('20.11.2002', 'DD.MM.YYYY'), TO_DATE('20.11.2002',
'DD.MM.YYYY') + 30
);
```

Jednocześnie, by zachować integralność bazy danych, uruchamiają się odpowiednie wyzwalacze, które ustawiają wypożyczenie danej pozycji (atrybut *isborrowed*) oraz zmniejszają licznik dostępnych egzemplarzy danej książki. Odbywa się jednocześnie walidacja danych.

Na początek definiujemy pomocniczą funkcję składowaną *liczba_ksiazek*, która ustawia poprawnie atrybut *liczba_egzemplarzy*:

```
create or replace function liczba_ksiazek() returns void as
$$
      declare
            counter int := 0;
            var_r int;
            var_p int;
      begin
            for var_r in (select s.ksiazka_id from biblioteka.ksiazka s
group by s.ksiazka_id)
            loop
                  for var_p in (select p.pozycja_id from biblioteka.pozycja
p where p.isborrowed = false and var_r = p.ksiazka_id group by
p.pozycja_id)
                  loop
                        counter = counter + 1;
                  end loop;
            update biblioteka.ksiazka s set liczba_egzemplarzy = counter
where s.ksiazka_id = var_r;
            counter = 0;
            end loop;
      end;
$$ language plpgsql;
```

Walidacja wypożyczenia dokonuje się przez wyzwalacz i dwie funkcje składowane (jedna dokonuje nadpisania danych, druga sprawdza, czy wypożyczenie jest dozwolone):

```
create or replace function wypozyczenie(integer) returns text as
$$
      declare
            id_p alias for $1;
            id_k int;
            borrowed_check boolean := (select p.isborrowed from
biblioteka.pozycja p where p.pozycja_id = id_p);
            zwrot text;
      begin
            zwrot := 'Niepowodzenie.';
            if borrowed_check = false then
```

```
            id_k = (select p.ksiazka_id from biblioteka.pozycja p where
p.pozycja_id = id_p group by p.ksiazka_id);
            update biblioteka.pozycja p set isborrowed = true where
p.pozycja_id = id_p;
            update biblioteka.ksiazka s set liczba_egzemplarzy =
s.liczba_egzemplarzy - 1 where s.ksiazka_id = id_k;
            zwrot := 'Operacja wypozyczenia udana.';
            end if;
            return zwrot;
      end;
$$ language plpgsql;
```

Ta funkcja sprawdza *legalność* wypożyczenia:

```
create or replace function wypozyczenie_check() returns trigger as
$$
      declare
            tekst text;
      begin
            if (select c.zadluzenie from biblioteka.czytelnik c where
czytelnik_id = NEW.czytelnik_id) is null OR (select c.zadluzenie from
biblioteka.czytelnik c where czytelnik_id = NEW.czytelnik_id) < 30 then
                  if (select count(*) as liczba_pozycji from
biblioteka.wypozyczenie w group by w.czytelnik_id, w.data_zwrocenia
                        having w.czytelnik_id = new.czytelnik_id and
w.data_zwrocenia is NULL) is null or (select count(*) as liczba_pozycji
from biblioteka.wypozyczenie w group by w.czytelnik_id, w.data_zwrocenia
                        having w.czytelnik_id = new.czytelnik_id and
w.data_zwrocenia is NULL) < 3 then
                              tekst = (select wypozyczenie(new.pozycja_id));
                              if tekst = 'Niepowodzenie.' then
                                    raise warning 'Nie mozna wypozyczyc juz
wypozyczonej pozycji!';
                                    return null;
                              end if;
                              raise warning 'Wypozyczono pozycje przez
czytelnika %.', new.czytelnik_id;
                              perform liczba_ksiazek();
                              return new;
                  end if;
                  raise warning 'Osiagnieto maksymalna liczbe pozycji!';
                  perform liczba_ksiazek();
                  return null;
            end if;
            raise warning 'Zadluzenie przekracza dozwolona wartosc!';
            return null;
      end;
$$ language plpgsql;
```

Wyzwalacz:

```
create trigger new_wypozyczenie before insert on biblioteka.wypozyczenie
for each row execute procedure wypozyczenie_check();
```

- Oddanie, przedłużenie, wyliczenie zadłużenia

Dokonuje się przez wpisanie daty oddania do atrybutu *data_zwrocenia* w relacji **wypozyczenie**.
Analogicznie działa **przedłużenie** pozycji, z tym, że wtedy *data_zwrocenia* pozostaje NULL, a zmienia się *wymagana_data_zwrotu*.
Przykład przedłużenia:

```
update biblioteka.wypozyczenie set wymagana_data_zwrotu =
wymagana_data_zwrotu + 30
where wypozyczenie_id = 3;
```

Przykład oddania:

```
update biblioteka.wypozyczenie set data_zwrocenia = TO_DATE('20.12.2002',
'DD.MM.YYYY')
where wypozyczenie_id = 3;
```

Do walidacji oddania wykorzystywana jest funkcja składowana i wyzwalacz:

Funkcja składowa dbająca o integralność danych:

```
create or replace function oddanie(integer) returns text as
$$
	declare
		id_p alias for $1;
		id_k int;
		borrowed_check boolean := (select p.isborrowed from
biblioteka.pozycja p where p.pozycja_id = id_p);
		zwrot text;
	begin
		zwrot = 'Niepowodzenie.';
		if borrowed_check = true then
			id_k = (select p.ksiazka_id from biblioteka.pozycja p
where p.pozycja_id = id_p group by p.ksiazka_id);
			update biblioteka.pozycja p set isborrowed = false where
p.pozycja_id = id_p;
			update biblioteka.ksiazka s set liczba_egzemplarzy =
s.liczba_egzemplarzy + 1 where s.ksiazka_id = id_k;
			zwrot = 'Operacja oddania udana.';
		end if;
	return zwrot;
	end;
$$ language plpgsql;
```

Funkcja składowa walidująca *poprawność* oddania:

```
create or replace function oddanie_check() returns trigger as
$$
declare
	tekst text;
	zadluzenie_val int;
begin
	if old.data_zwrocenia is null and new.data_zwrocenia is not null then
		tekst := (select oddanie(new.pozycja_id));
		if tekst = 'Niepowodzenie.' then
			raise warning 'Nie mozna oddac niewypozyczonej pozycji!';
			return NULL;
		end if;
```

```
            -- miejsce na wyliczenie zadluzenia
            if new.data_zwrocenia - new.wymagana_data_zwrotu > 1 then
                  zadluzenie_val = ( new.data_zwrocenia -
new.wymagana_data_zwrotu )/2 * 1;
                  update biblioteka.czytelnik c set zadluzenie =
zadluzenie_val where c.czytelnik_id = new.czytelnik_id;
            end if;
            raise warning 'Pomyslnie oddano.';
            perform liczba_ksiazek();
            return new;
      end if;

      if new.wymagana_data_zwrotu - new.data_wypozyczenia > 120 then
            raise warning 'Nie mozna przedluzyc pozycji.';
            return NULL;
      end if;
      if new.wymagana_data_zwrotu <> old.wymagana_data_zwrotu then
            raise warning 'Przedluzono pozycje.';
      end if;
      return new;
end;
$$ language plpgsql;
```

Funkcja walidująca jednocześnie oblicza i wpisuje zadłużenie. Nie pozwala również przedłużyć pozycji o dłużej niż 3 miesiące (30 + 90).

Wyzwalacz:

```
create trigger oddanie_lub_przedluzenie before update on
biblioteka.wypozyczenie
for each row execute procedure oddanie_check();
```

Procedury powyżej obsługują zarówno oddawanie, jak i przedłużanie, liczenie zadłużenia.

- Pełne informacje o książce

Z racji na dwie różne relacje N:M (wiele autorów i wiele kategorii), nie jest łatwo dobrać odpowiednią kwerendę do szybkiego wylistowanie pełnej informacji o danej książce.
W tym projekcie zrobiono to przez złożenie dwóch różnych widoków i zdefiniowanie odrębnego widoku.

Widok informacji o autorach książek:

```
create view ksiazka_info as
select k.ksiazka_id, a.nazwa as Autor, k.nazwa as Tytul,
k.liczba_egzemplarzy from ((biblioteka.ksiazka k join
biblioteka.ksiazka_autor ka using (ksiazka_id)) join biblioteka.autor a
using (autor_id))
group by ksiazka_id, Autor, Tytul, liczba_egzemplarzy;
```

Widok informacji o kategoriach książek:

```
create view ksiazka_kat_info as
select k.nazwa as tytul, k2.kat_nazwa as Kategoria from
((biblioteka.ksiazka k join biblioteka.ksiazka_kategoria kk using
(ksiazka_id)) join biblioteka.kategoria k2 using (kategoria_id))
group by tytul, kategoria;
```

Widok informacji o autorach i kategoriach książki:

```
create view ksiazka_info_aut_kat as
select k.ksiazka_id, k.autor, k.tytul, ki.kategoria from ksiazka_info k
join ksiazka_kat_info ki using (tytul)
group by ksiazka_id, k.autor, k.tytul, ki.kategoria;

select * from ksiazka_info_aut_kat where ksiazka_id = 5 order by kategoria,
autor;
```

Widok został tu użyty, by nieco uprościć późniejszą interakcję bazy danych z interfejsem.

- dodanie użytkownika, wyzerowanie mu zadłużenia, usunięcie użytkownika

Te operacje można prosto wykonać w języku SQL:

Dodanie użytkownika:

```
insert into biblioteka.czytelnik (imie, nazwisko) values (
	'Arkadiusz', 'Korzeniak'
);
```

Wyzerowanie mu zadłużenia:

```
update biblioteka.czytelnik c set zadluzenie = null where c.czytelnik_id =
1;
```

W ramach interfejsu zostało zaimplementowane dodatkowe ograniczenie: wyzerowanie zadłużenia oraz usunięcie użytkownika należy dodatkowo zautoryzować.

Usunięcie użytkownika:

```
delete from biblioteka.czytelnik c where c.czytelnik_id = 1;
```

Nie ma mechanizmów uniemożliwiających usunięcie jakiegoś czytelnika, za to w tabeli **wypozyczenie** w miejscu *czytelnik_id* pojawia się NULL.

- obliczanie liczby obecnie wypożyczonych przez użytkownika pozycji

Odbywa się również przez widok z użyciem CTE i wyrażenia CASE. Skomplikowanie tej operacji wynika z braku możliwości policzenia daty zwrócenia, gdy jej wartość jest NULL. Użycie widoku ułatwia późniejszą komunikację z interfejsem.

```
create view czytelnik_info as
with cte as (select w.czytelnik_id, w.data_zwrocenia, sum(case when
w.data_zwrocenia is null then 1 else 0 end) as liczba_pozycji from
biblioteka.wypozyczenie w group by w.czytelnik_id, w.data_zwrocenia
                        )
select c.czytelnik_id, c.imie, c.nazwisko, sum(cte.liczba_pozycji) from
biblioteka.czytelnik c join cte using(czytelnik_id)
group by c.czytelnik_id, c.imie, c.nazwisko ;

select * from czytelnik_info order by nazwisko;
```

Oprócz tego, uwzględnione jest wyświetlanie informacji o użytkowniku w powiązaniu z jego obecnymi wypożyczeniami:

```
create view uzytkownik as
select c.czytelnik_id, c.imie, c.nazwisko, p.pozycja_id, k.nazwa,
w.wymagana_data_zwrotu from ((((biblioteka.czytelnik c join
biblioteka.wypozyczenie w using (czytelnik_id)) join biblioteka.pozycja p
using (pozycja_id))) join biblioteka.ksiazka k using (ksiazka_id))
where w.data_zwrocenia is null;

select * from uzytkownik order by nazwisko;
```

A także rankingi:

- Najchętniej czytana książka:

```
with cte as(select count(*) as liczba_wypozyczen, k.nazwa as Tytul from
((biblioteka.wypozyczenie w join biblioteka.pozycja p using (pozycja_id))
join biblioteka.ksiazka k using (ksiazka_id))
group by k.nazwa)
select rank() over (order by liczba_wypozyczen) as Miejsce, Tytul from cte;
```

- Najchętniej czytany autor:

```
with cte as(select count(*) as liczba_wypozyczen, a.nazwa as Autor from
((((biblioteka.wypozyczenie w join biblioteka.pozycja p using (pozycja_id))
join biblioteka.ksiazka k using (ksiazka_id)) join biblioteka.ksiazka_autor
ka using (ksiazka_id)) join biblioteka.autor a using (autor_id))
group by a.nazwa)
select rank() over (order by liczba_wypozyczen) as Miejsce, Autor from cte;
```

- Najchętniej czytana kategoria:

```
with cte as(select count(*) as liczba_wypozyczen, k2.kat_nazwa as Kategoria
from ((((biblioteka.wypozyczenie w join biblioteka.pozycja p using
(pozycja_id)) join biblioteka.ksiazka k using (ksiazka_id)) join
biblioteka.ksiazka_kategoria kk using (ksiazka_id)) join
biblioteka.kategoria k2 using (kategoria_id))
group by Kategoria)
select rank() over (order by liczba_wypozyczen) as Miejsce, Kategoria from
cte;
```

Zdefiniowana również została funkcja RESET, która restartuje bazę danych i uzupełnia ją przykładowymi wartościami. Funkcja ta wykonuje skrypt w języku SQL:

```
drop schema biblioteka cascade;
create schema biblioteka;

create table biblioteka.ksiazka(
	ksiazka_id SERIAL primary key,
	nazwa VARCHAR(128) not null,
	liczba_egzemplarzy int
);

create table biblioteka.autor(
	autor_id SERIAL primary key,
	nazwa VARCHAR(64) not null,
	data_urodzenia DATE not null
);

--encja asocjacyjna autorów (m:n)

create table biblioteka.ksiazka_autor(
	ksiazka id int not null,
	autor_id int not null,
	CONSTRAINT ksiazka_autor_pk primary key(ksiazka_id, autor_id),
	CONSTRAINT ksiazka autor autor id fk foreign key (autor id) references
biblioteka.autor(autor_id),
	CONSTRAINT ksiazka_autor_ksiazka_id_fk foreign key (ksiazka_id) references
biblioteka.ksiazka(ksiazka_id)
);

create table biblioteka.kategoria(
	kategoria_id SERIAL primary key,
	kat_nazwa varchar(32) not null
);
--encja asocyjacyjna kategorii (m:n)
create table biblioteka.ksiazka_kategoria(
	ksiazka_id int not null,
	kategoria id int not null,
	constraint ksiazka_kategoria_pk primary key (ksiazka_id, kategoria_id),
	constraint ksiazka_kategoria_ksiazka_id_fk foreign key(ksiazka_id) references
biblioteka.ksiazka(ksiazka id),
	constraint ksiazka_kategoria_kategoria_id_fk foreign key(kategoria_id) references
biblioteka.kategoria(kategoria_id)
);

create table biblioteka.wydawnictwo(
	wydawnictwo_id serial primary key,
	wyd_nazwa varchar(32) not null
);

create table biblioteka.pozycja(
	pozycja_id SERIAL primary key,
	ksiazka_id int not null,
	wydawnictwo_id int not null,
	rok_wydania int not null,
	isBorrowed boolean,
	constraint pozycja_ksiazka_id_fk foreign key (ksiazka_id) references
biblioteka.ksiazka(ksiazka_id),
	constraint pozycja_wydawnictwtwo_id_fk foreign key (wydawnictwo_id) references
biblioteka.wydawnictwo(wydawnictwo_id)
);

create table biblioteka.czytelnik(
	czytelnik_id serial primary key,
	imie varchar(32) not null,
	nazwisko varchar(32) not null,
	zadluzenie int
);


--encja asocjacyjna -> wypozyczenie

create table biblioteka.wypozyczenie(
	wypozyczenie_id SERIAL primary key,
	czytelnik_id int,
	pozycja_id int not null,
	data_wypozyczenia date not null,
	wymagana_data_zwrotu date not null,
	data_zwrocenia date,
```

```
        constraint wypozyczenie_czytelnik_id_fk foreign key(czytelnik_id) references
biblioteka.czytelnik(czytelnik_id) on delete set null,
        constraint wypozyczenie_pozycja_id_fk foreign key(pozycja_id) references
biblioteka.pozycja(pozycja_id) on delete set null
);


insert into biblioteka.czytelnik (imie, nazwisko) values (
        'Arkadiusz', 'Korzeniak'
);
insert into biblioteka.czytelnik (imie, nazwisko) values (
        'Arkadiusz', 'Dworzeniak'
);
insert into biblioteka.czytelnik (imie, nazwisko) values (
        'Mariusz', 'Corzeniak'
);

insert into biblioteka.wydawnictwo ( wyd_nazwa) values (
        'Grek'
);

insert into biblioteka.wydawnictwo ( wyd nazwa) values (
        'Roma'
);


insert into biblioteka.kategoria ( kat_nazwa) values(
       'kryminal'
);
insert into biblioteka.kategoria ( kat_nazwa) values(
       'fantasy'
);
insert into biblioteka.kategoria (kat_nazwa) values(
       'romans'
);
insert into biblioteka.kategoria ( kat_nazwa) values(
       'powiesc przygodowa'
);
insert into biblioteka.kategoria ( kat_nazwa) values(
       'powiesc historyczna'
);

insert into biblioteka.autor (nazwa, data_urodzenia) values(
        'J.R.R. Tolkien', TO_DATE('03.01.1852','DD.MM.YYYY')
);


insert into biblioteka.ksiazka ( nazwa, liczba_egzemplarzy) values(
       'Druzyna Pierscienia', 2
);
insert into biblioteka.ksiazka_autor values (1, 1);
insert into biblioteka.ksiazka_kategoria values (1,2);
insert into biblioteka.ksiazka_kategoria values (1,4);

insert into biblioteka.ksiazka (nazwa, liczba_egzemplarzy) values(
        'Dwie Wieze', 2
);
insert into biblioteka.ksiazka_autor values (2, 1);
insert into biblioteka.ksiazka_kategoria values (2,2);
insert into biblioteka.ksiazka_kategoria values (2,4);

insert into biblioteka.ksiazka ( nazwa, liczba_egzemplarzy) values(
        'Powrot Krola', 2
);
insert into biblioteka.ksiazka_autor values (3, 1);
insert into biblioteka.ksiazka kategoria values (3,2);
insert into biblioteka.ksiazka_kategoria values (3,4);

insert into biblioteka.ksiazka( nazwa, liczba egzemplarzy) values(
       'Romeo i Julia', 5
);
insert into biblioteka.autor ( nazwa, data_urodzenia) values(
        'William Shakespeare', TO_DATE('01.04.1564', 'DD.MM.YYYY')
);
insert into biblioteka.ksiazka_autor values (4,2);
insert into biblioteka.ksiazka kategoria values (4,3);
```

```
insert into biblioteka.ksiazka (nazwa, liczba_egzemplarzy)values (
	'Potop', 1
);
insert into biblioteka.autor (nazwa, data_urodzenia )values('Henryk Sienkiewicz',
TO_DATE('05.05.1846', 'DD.MM.YYYY'));
insert into biblioteka.ksiazka_autor values (5,3);
insert into biblioteka.ksiazka_autor values (5,1);
insert into biblioteka.ksiazka_kategoria values(5,3);
insert into biblioteka.ksiazka_kategoria values(5,4);
insert into biblioteka.ksiazka_kategoria values(5,5);



insert into biblioteka.pozycja ( ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	1, 1, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	1, 1, 2003, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	1, 2, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	1, 1, 2012, false
);

insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	2, 1, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	2, 1, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	2, 2, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	2, 2, 2002, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	3, 1, 2005, false
);

insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	3, 1, 2004, false
);

insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	4, 2, 2013, false
);
insert into biblioteka.pozycja (ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	 4, 2, 2013, false
);
insert into biblioteka.pozycja ( ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
 4, 2, 2004, false
);

insert into biblioteka.pozycja ( ksiazka_id, wydawnictwo_id, rok_wydania, isborrowed) values (
	 5, 2, 1994, false
);



create or replace function liczba_ksiazek() returns void as
$$
	declare
		counter int := 0;
		var_r int;
		var_p int;
	begin
		for var_r in (select s.ksiazka_id from biblioteka.ksiazka s group by
s.ksiazka_id)
		loop
			for var_p in (select p.pozycja_id from biblioteka.pozycja p where
p.isborrowed = false and var_r = p.ksiazka_id group by p.pozycja_id)
			loop
				counter = counter + 1;
			end loop;
```

```
                update biblioteka.ksiazka s set liczba_egzemplarzy = counter where s.ksiazka_id
= var_r;
                counter = 0;
                end loop;
        end;
$$ language plpgsql;

create or replace function wypozyczenie(integer) returns text as
$$
        declare
                id_p alias for $1;
                id_k int;
                borrowed_check boolean := (select p.isborrowed from biblioteka.pozycja p where
p.pozycja_id = id_p);
                zwrot text;
        begin
                zwrot := 'Niepowodzenie.';
                if borrowed_check = false then
                id_k = (select p.ksiazka_id from biblioteka.pozycja p where p.pozycja_id = id_p
group by p.ksiazka_id);
                update biblioteka.pozycja p set isborrowed = true where p.pozycja_id = id_p;
                update biblioteka.ksiazka s set liczba_egzemplarzy = s.liczba_egzemplarzy - 1
where s.ksiazka_id = id_k;
                zwrot := 'Operacja wypozyczenia udana.';
                end if;
                return zwrot;
        end;
$$ language plpgsql;


create or replace function oddanie(integer) returns text as
$$
        declare
                id_p alias for $1;
                id_k int;
                borrowed_check boolean := (select p.isborrowed from biblioteka.pozycja p where
p.pozycja_id = id_p);
                zwrot text;
        begin
                zwrot = 'Niepowodzenie.';
                if borrowed_check = true then
                        id_k = (select p.ksiazka_id from biblioteka.pozycja p where p.pozycja_id
= id_p group by p.ksiazka_id);
                        update biblioteka.pozycja p set isborrowed = false where p.pozycja_id =
id_p;
                        update biblioteka.ksiazka s set liczba_egzemplarzy =
s.liczba_egzemplarzy + 1 where s.ksiazka_id = id_k;
                        zwrot = 'Operacja oddania udana.';
                end if;
        return zwrot;
        end;
$$ language plpgsql;



create or replace function wypozyczenie_check() returns trigger as
$$
        declare
                tekst text;
        begin
                if (select c.zadluzenie from biblioteka.czytelnik c where czytelnik_id =
NEW.czytelnik_id) is null OR (select c.zadluzenie from biblioteka.czytelnik c where
czytelnik_id = NEW.czytelnik_id) < 30 then
                        if (select count(*) as liczba_pozycji from biblioteka.wypozyczenie w
group by w.czytelnik id, w.data zwrocenia
                                having w.czytelnik_id = new.czytelnik_id and w.data_zwrocenia is
NULL) is null or (select count(*) as liczba_pozycji from biblioteka.wypozyczenie w group by
w.czytelnik id, w.data zwrocenia
                                having w.czytelnik_id = new.czytelnik_id and w.data_zwrocenia is
NULL) < 3 then
                                        tekst = (select wypozyczenie(new.pozycja_id));
                                        if tekst = 'Niepowodzenie.' then
                                                raise warning 'Nie mozna wypozyczyc juz
wypozyczonej pozycji!';
                                                return null;
                                        end if;
```

```
                                        raise warning 'Wypozyczono pozycje przez czytelnika %.',
new.czytelnik_id;
                                        perform liczba_ksiazek();
                                        return new;
                        end if;
                        raise warning 'Osiagnieto maksymalna liczbe pozycji!';
                        perform liczba_ksiazek();
                        return null;
                end if;
                raise warning 'Zadluzenie przekracza dozwolona wartosc!';
                return null;
        end;
$$ language plpgsql;



create trigger new_wypozyczenie before insert on biblioteka.wypozyczenie
for each row execute procedure wypozyczenie_check();


create or replace function oddanie_check() returns trigger as
$$
declare
        tekst text;
        zadluzenie_val int;
begin
        if old.data_zwrocenia is null and new.data_zwrocenia is not null then
                tekst := (select oddanie(new.pozycja id));
                if tekst = 'Niepowodzenie.' then
                        raise warning 'Nie mozna oddac niewypozyczonej pozycji!';
                        return NULL;
                end if;
                -- miejsce na wyliczenie zadluzenia
                if new.data_zwrocenia - new.wymagana_data_zwrotu > 1 then
                        zadluzenie val = ( new.data zwrocenia - new.wymagana data zwrotu )/2 *
1;
                        update biblioteka.czytelnik c set zadluzenie = zadluzenie_val where
c.czytelnik id = new.czytelnik id;
                end if;
                raise warning 'Pomyslnie oddano.';
                perform liczba_ksiazek();
                return new;
        end if;

        if new.wymagana_data_zwrotu - new.data_wypozyczenia > 120 then
                raise warning 'Nie mozna przedluzyc pozycji.';
                return NULL;
        end if;
        if new.wymagana_data_zwrotu <> old.wymagana_data_zwrotu then
                raise warning 'Przedluzono pozycje.';
        end if;
        return new;
end;
$$ language plpgsql;


create trigger oddanie_lub_przedluzenie before update on biblioteka.wypozyczenie
for each row execute procedure oddanie_check();


--test funkcjonalnosci
insert into biblioteka.wypozyczenie (czytelnik_id, pozycja_id, data_wypozyczenia,
wymagana_data_zwrotu)
values (
        1, 7, TO_DATE('20.11.2002', 'DD.MM.YYYY'), TO_DATE('20.12.2002', 'DD.MM.YYYY')
);

update biblioteka.wypozyczenie set data_zwrocenia = TO_DATE('20.12.2002', 'DD.MM.YYYY')
where czytelnik id = 1 and data zwrocenia is null;

insert into biblioteka.wypozyczenie (czytelnik_id, pozycja_id, data_wypozyczenia,
wymagana_data_zwrotu)
values (
        1, 2, TO_DATE('20.11.2002', 'DD.MM.YYYY'), TO_DATE('20.12.2002', 'DD.MM.YYYY')
);
```

```
create view uzytkownik as
select c.czytelnik_id, c.imie, c.nazwisko, p.pozycja_id, k.nazwa, w.wymagana_data_zwrotu from
((((biblioteka.czytelnik c join biblioteka.wypozyczenie w using (czytelnik_id)) join
biblioteka.pozycja p using (pozycja_id))) join biblioteka.ksiazka k using (ksiazka_id))
where w.data_zwrocenia is null;


select * from uzytkownik order by nazwisko
create view ksiazka_info as
select k.ksiazka_id, a.nazwa as Autor, k.nazwa as Tytul, k.liczba_egzemplarzy from
((biblioteka.ksiazka k join biblioteka.ksiazka_autor ka using (ksiazka_id)) join
biblioteka.autor a using (autor_id))
group by ksiazka_id, Autor, Tytul, liczba_egzemplarzy;

create view ksiazka_kat_info as
select k.nazwa as tytul, k2.kat_nazwa as Kategoria from ((biblioteka.ksiazka k join
biblioteka.ksiazka kategoria kk using (ksiazka id)) join biblioteka.kategoria k2 using
(kategoria_id))
group by tytul, kategoria;



create view ksiazka_info_aut_kat as
select k.ksiazka_id, k.autor, k.tytul, ki.kategoria from ksiazka_info k join ksiazka_kat_info
ki using (tytul)
group by ksiazka id, k.autor, k.tytul, ki.kategoria;

create view czytelnik_info as
with cte as (select w.czytelnik id, w.data zwrocenia, sum(case when w.data zwrocenia is null
then 1 else 0 end) as liczba_pozycji from biblioteka.wypozyczenie w group by w.czytelnik_id,
w.data_zwrocenia
                                )
select c.czytelnik id, c.imie, c.nazwisko, sum(cte.liczba pozycji) from biblioteka.czytelnik c
join cte using(czytelnik_id)
group by c.czytelnik_id, c.imie, c.nazwisko ;
```

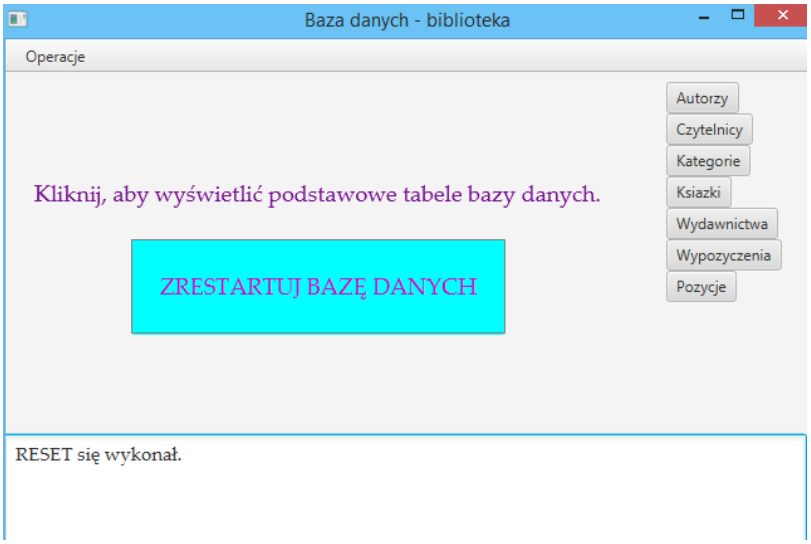
4. Projekt funkcjonalny

a. Interfejsy do prezentacji, edycji i obsługi danych

Interfejs składa się z kilku okienek, które można przeglądać w jednym czasie.
Na początku znajduje się okienko, które umożliwia połączenie się z bazą danych ElephantSQL. Przy udanym połączeniu się, wyświetla się komunikat o sukcesie oraz otwiera się główne okno aplikacji.

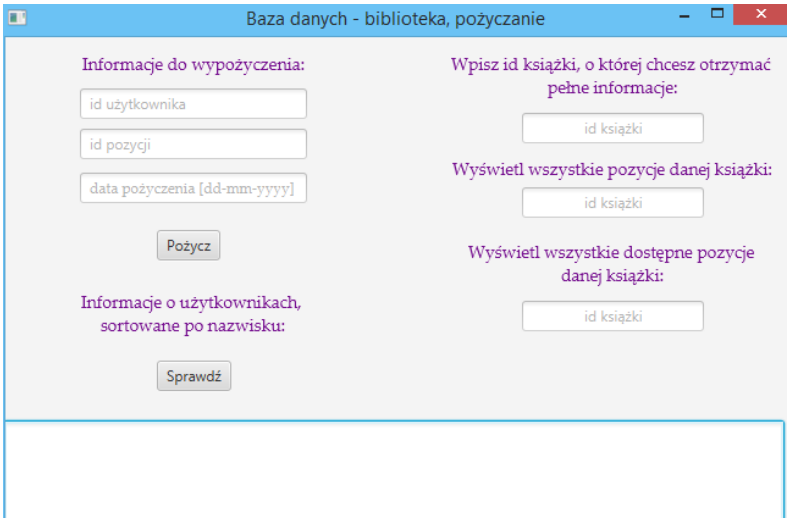
Główne okno umożliwia przegląd wszystkich tabel, które nie są tabelami asocjacyjnymi, a także umożliwia zrestartowanie bazy danych do pierwotnego stanu.

Przegląd tabel odbywa się przez wciśnięcie odpowiednich przycisków.



Rysunek 2: Główne menu programu

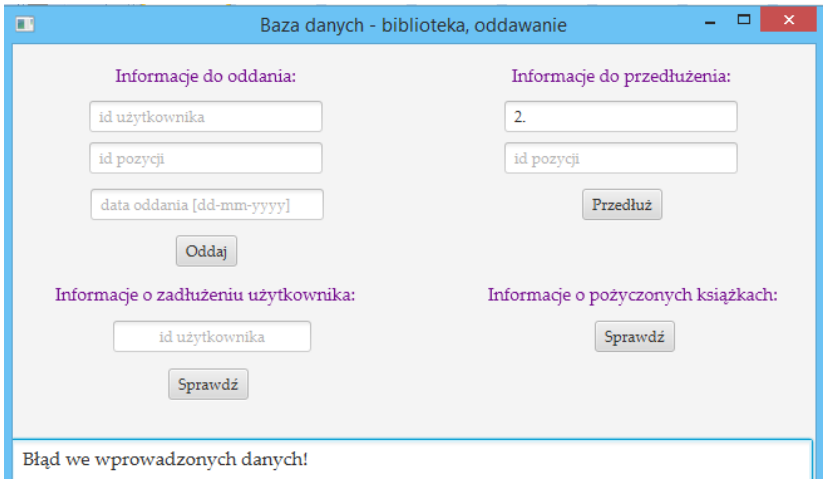
Realizacja wypożyczenia i uzyskiwanie informacji o danej książce, a także o dostępnych pozycjach danej książki odbywa się przez wpisanie do formularza odpowiednich identyfikatorów: czytelnika, książki, w przypadku wypożyczenia również daty w jednoznacznie określonym formacie dd-mm-rrrr (dzień-miesiąc-rok). Użytkownik może pozyskać odpowiednie identyfikatory z tabeli z menu głównego.



Rysunek 3: Menu pożyczania i uzyskiwania informacji o danej książce.

Dane potwierdzane są albo przez przycisk ENTER (w przypadku formularzy z jednym polem) lub przez kliknięcie odpowiedniego przycisku.

Realizacja oddania lub przedłużenia pozycji dokonuje się w osobnym oknie. Wprowadzanie informacji do formularza działa na identycznej zasadzie. Zawsze należy pilnować, by wpisywać identyfikator istniejący oraz by był on liczbą całkowitą. W przeciwnym wypadku pojawi się komunikat o błędzie, a zapytanie się nie wykona.



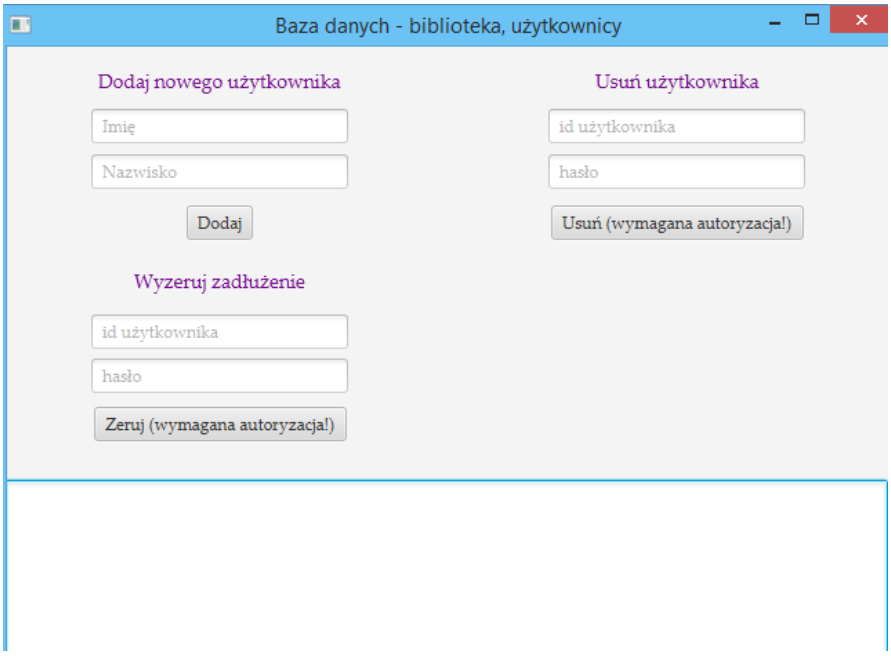
Rysunek 4: Menu do oddawania i przedłużania

Okno to umożliwia również uzyskanie informacji, kto obecnie posiada jaką książkę, kiedy ma ją zwrócić, a także ID użytkownika i ID pozycji.

Kolejne okno umożliwia wprowadzenie nowego czytelnika, a także wyzerowanie jego zadłużenia lub usunięcie użytkownika biblioteki. Wprowadza ono nowość: autoryzację. Przy wyzerowaniu i usunięciu czytelnika należy podać odpowiednie hasło, by przejść dalej, w przypadku podania błędnego hasła żadna operacja się nie wykona. **Hasłem jest 'maslo'.**

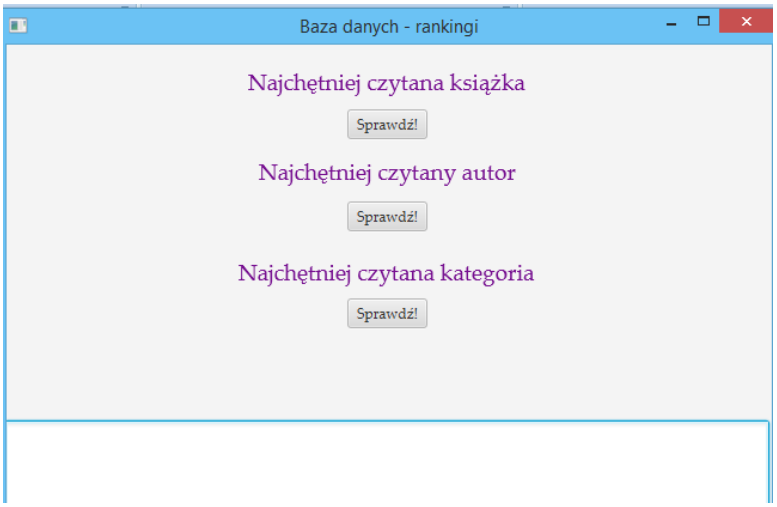
Dodanie użytkownika odbywa się przez formularz, w który wpisuje się imię i nazwisko danego użytkownika (nie mogą one być dłuższe niż 32 znaki każde), potwierdza się przez przycisk 'Dodaj'.

W przypadku usunięcia zadłużenia i usunięcia czytelnika wystarczy podać identyfikator użytkownika.



Rysunek 5: Menu użytkowników

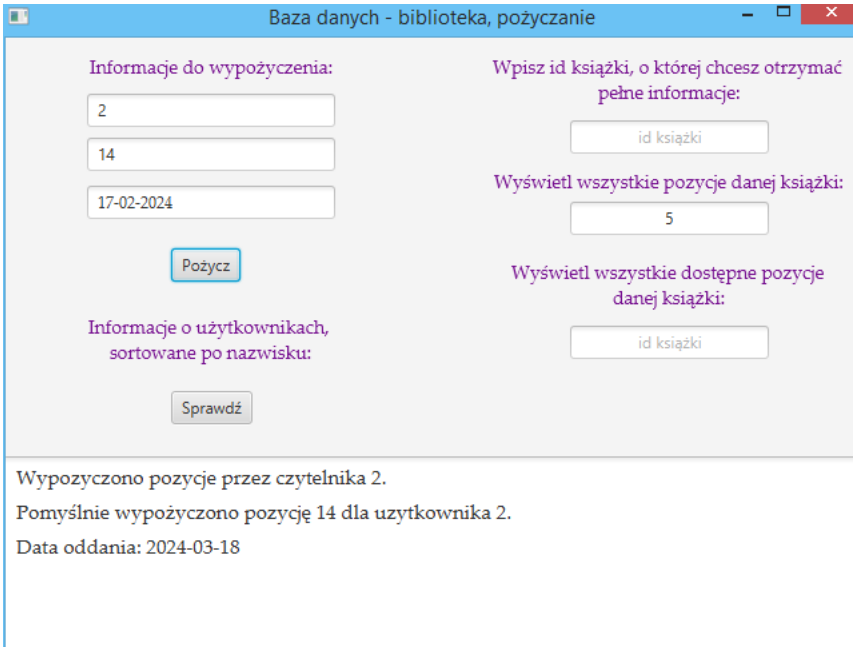
Ostatnie okno nie przewiduje formularzy do wprowadzania danych. Umożliwia ono podgląd wybranych rankingów, użytkownik aplikacji musi użyć odpowiednich przycisków do ich wyświetlenia.



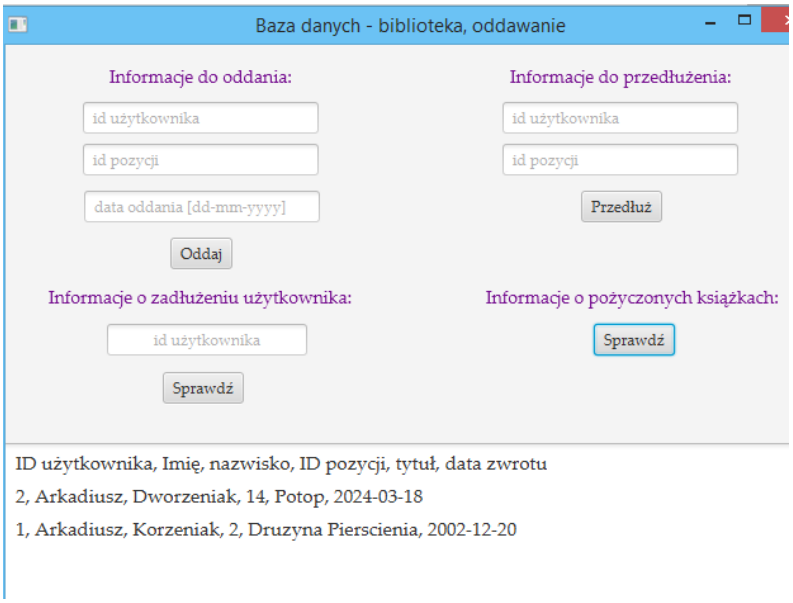
Rysunek 6: Menu rankingów

b. Wizualizacja danych

Raporty generowane przez bazę danych są wyświetlane w oknie dialogowym, które jest częścią każdego okna. Okno dialogowe informuje o pomyślności przeprowadzonej operacji (lub jej braku), a także wyświetla raporty. Każdy z nich ma odpowiednią strukturę: nagłówek, gdzie wypisane są poszczególne kolumny tabel po przecinku, oraz ciało struktury, gdzie wypisane są poszczególne rekordy (każdy od nowej linii) po przecinku.



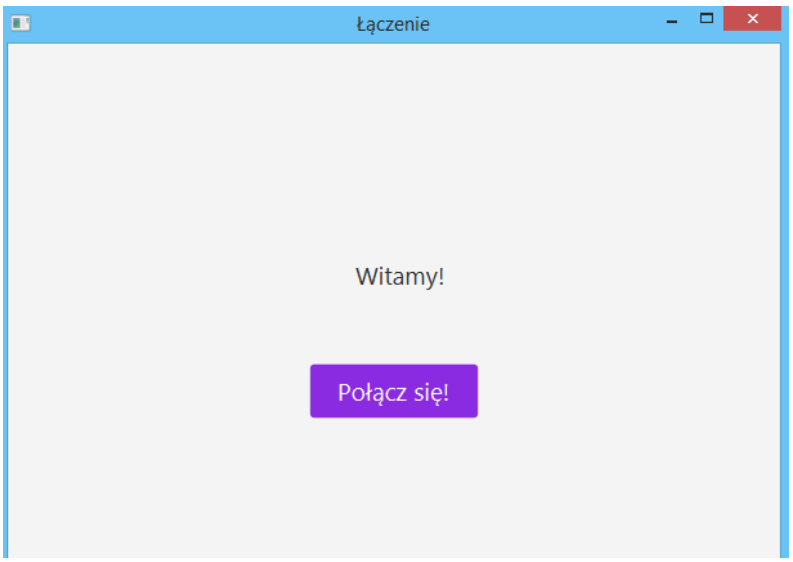
Rysunek 7: Komunikat o pomyślnym wyniku operacji.



Rysunek 8: Przykładowy raport wygenerowany przez bazę danych.

c. Zdefiniowanie panelu sterowania aplikacji

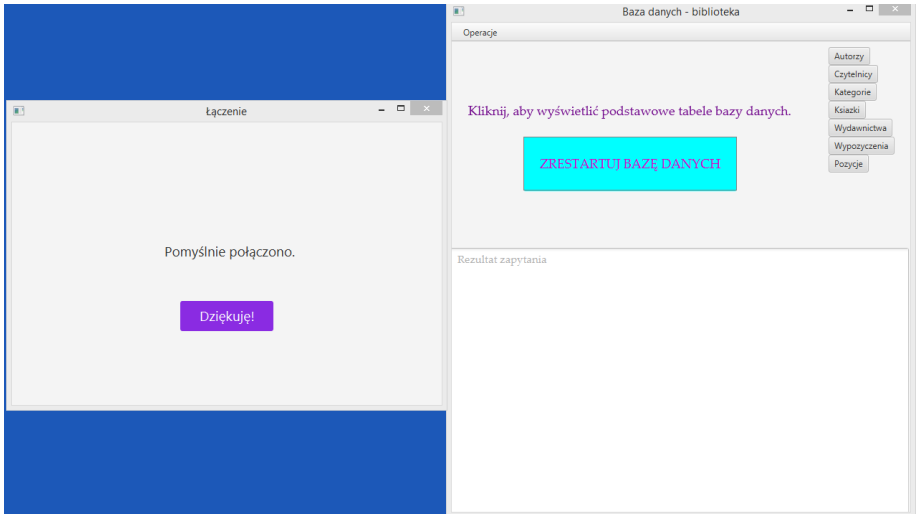
Po uruchomieniu aplikacji pojawi się prośba o połączenie się z bazą danych.



Rysunek 9: Okno powitalne.

By się połączyć, należy kliknąć przycisk "Połącz się".

Po połączeniu się, pojawia się okno *Baza danych - biblioteka*. Jest to główne okno sterujące, w nim jest opcja zrestartowania bazy danych, z niego też można przejść do innych okien.

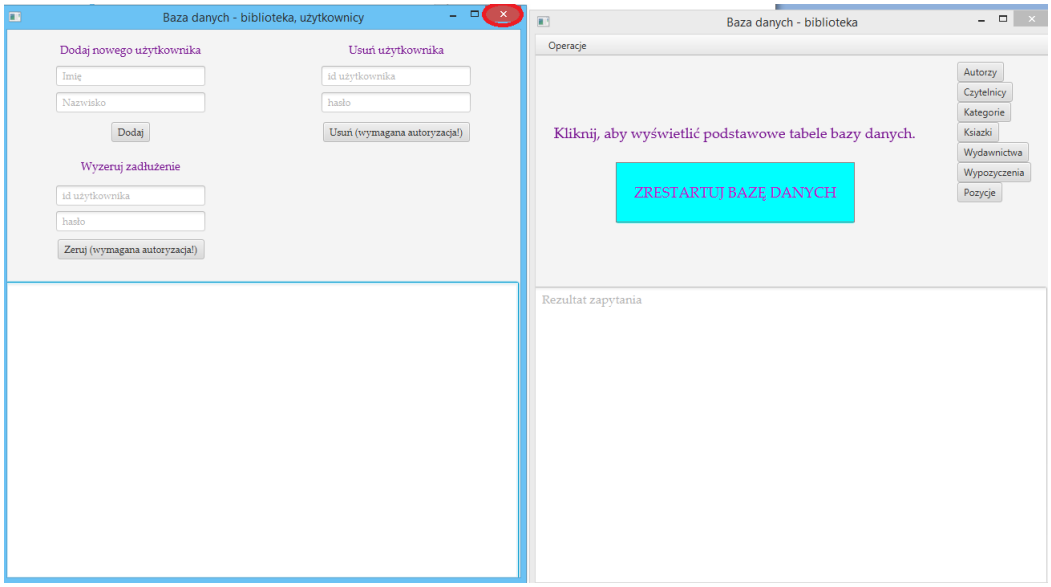


Rysunek 10: Pojawienie się głównego menu.

Aby otworzyć kolejne okna, należy z rozwijanego menu *Operacje* wybrać wybraną przez nas pozycję.

Rysunek 11: Rozwijane menu umożliwiające otworzenie kolejnych okien.

Każde okno można zamknąć przez użycie systemowego przycisku zamykania.



Rysunek 12: Prezentacja zamykania okna.

Program nie przestaje działać nawet wtedy, gdy zamkniemy główne menu sterujące. By zatrzymać program, należy zamknąć wszystkie okna. Po zamknięciu głównego menu nie ma jednak możliwości wywołania go ponownie, należy zrestartować program.

d. Makropolecenia

Aplikacja realizuje opcję zrestartowania bazy danych do stanu początkowego (3 użytkowników, 2 wypożyczenia). Makropolecenie jest realizowane przy pomocy Java DataBase Connectivity (JDBC).

5. Dokumentacja

a. Wprowadzanie danych

Dane są wprowadzane z klawiatury (ręcznie), odpowiednie zapytania można zrealizować również przez przyciśnięcie odpowiednich przycisków (zob. rozdział 4 niniejszej dokumentacji).

b. Dokumentacja użytkownika.

Na samym początku, aby się połączyć, wciśnij "Połącz się!". Nie próbuj uruchamiać więcej niż jednej instancji aplikacji. Jeśli coś pójdzie nie tak (np. brak internetu), pojawi się komunikat "Łączenie…". Jeśli usuniemy przeszkodę (np. zamkniemy drugą instancję aplikacji), możemy kliknąć jeszcze raz przycisk, by się połączyć.
Otworzy się główne menu programu, a w tamtym oknie pojawi się informacja o pomyślnym połączeniu. Instrukcje sterujące są szerzej opisane w podrozdziale c rozdziału 4 niniejszej dokumentacji.

Aby dokonać wypożyczenia, należy wybrać z rozwijanego menu opcję "Wypożycz".

Załóżmy, że czytelnik o danym imieniu i nazwisku chce pożyczyć jakąś książkę.
Jego ID można sprawdzić w menu głównym przy tabeli czytelnicy. Jeśli pożyczał już coś wcześniej, to w oknie wypożyczeń można wyświetlić informacje o jego ID i liczbie obecnie wypożyczonych pozycji. Należy kliknąć 'sprawdź' i odszukać informacje w raporcie posortowanym po nazwisku.

Następnie chcemy temu użytkownikowi przypisać daną pozycję. Najpierw w głównym menu wyświetlamy wszystkie dostępne książki, odczytujemy ID tej, która nas interesuje, następnie korzystamy z formularza w menu wypożyczeń, by wyświetlić dostępne pozycje danej książki i ich ID.

Wypożyczenie finalizujemy, wpisując ID użytkownika i ID pozycji, a także datę wypożyczenia. Uwaga! **Data musi być w formacie dd-mm-rrrr** (dzień, miesiąc, rok), inaczej wyświetli się błąd daty.

Przedłużenie lub oddanie realizujemy w menu 'oddawanie'. Tam również możemy sprawdzić zadłużenie danego użytkownika. Przedłużanie i oddawanie realizowane jest analogicznie: przez wprowadzenie ID użytkownika i ID pozycji, a w wypadku oddania — daty oddania. Domyślnie wypożyczenie przedłużane jest o 30 dni.

ID użytkownika i ID pozycji można pozyskać z dostępnego w tym menu raportu pod tytułem "Informacje o pożyczonych książkach", które nam wylistują obecne wypożyczenia, posortowane po nazwisku czytelnika.

Dodanie nowego użytkownika realizuje się w menu 'użytkownicy'.
Należy wpisać imię i nazwisko do formularza i kliknąć "Dodaj".

W przypadku wyzerowania zadłużenia i usunięcia użytkownika należy wcześniej pozyskać jego ID, a by przeprowadzić te operacje, należy podać również hasło. Hasłem jest 'maslo'.

Dostęp do rankingów jest w menu 'rankingi'. By wyświetlić odpowiednie raporty, należy po prostu wcisnąć przycisk.

Zrestartowanie bazy danych jest możliwe w menu głównym. Należy po prostu kliknąć niebieski przycisk 'ZRESTARTUJE BAZĘ DANYCH'.

c. Opracowanie dokumentacji technicznej.

Dokumentacja techniczna interfejsu (w języku Java) została wygenerowana przy pomocy narzędzia javadoc jako strona, przesłana osobno jako archiwum javadoc.zip.

d. Wykaz literatury

- wykład Bazy Danych I, dr inż. Antoni Dydejczyk
- https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html
- https://openjfx.io/javadoc/21/
- https://openjfx.io/javadoc/21/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html

Projekt został zrealizowany w języku Java, przy użyciu aplikacji klienta JavaFX oraz sterownika JDBC. Schemat ERD oraz wstępny projekt bazy został wykonany w DBeaver.