Projekt bazy danych systemu zarządzania biblioteką

Karolina Kominiak
Dominika Cyganek

Spis treści

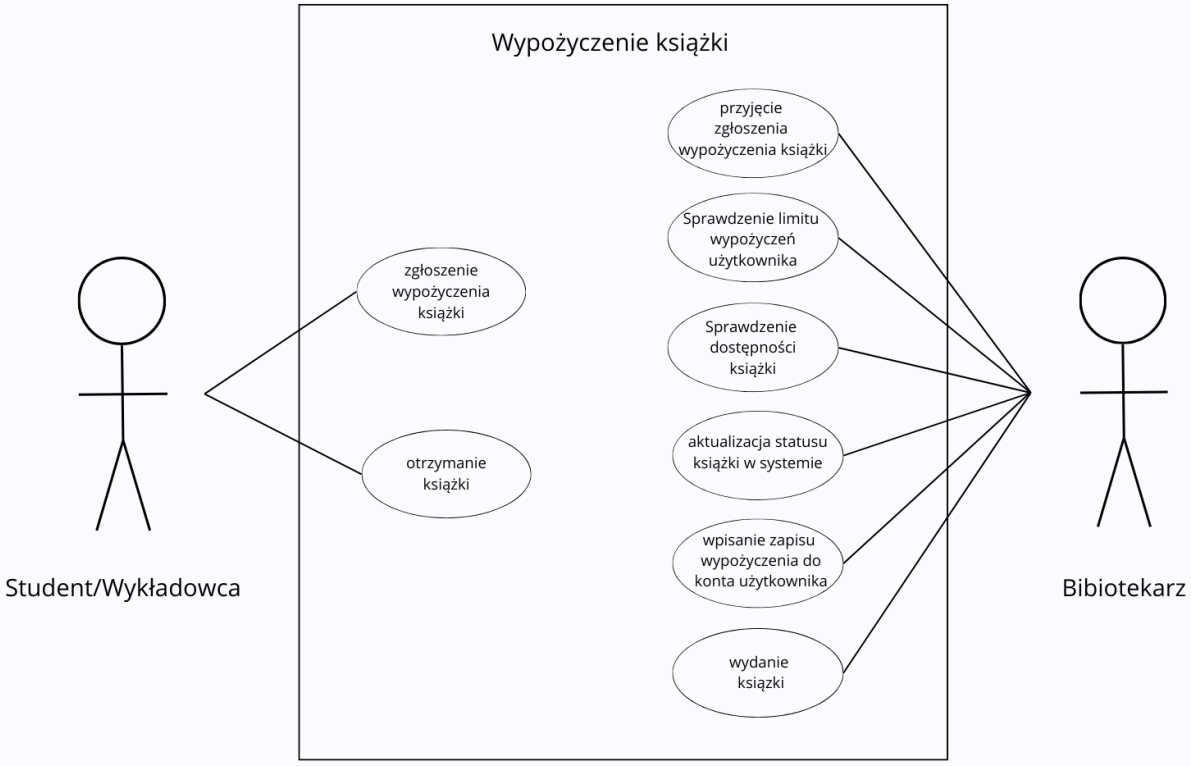
Wprowadzenie

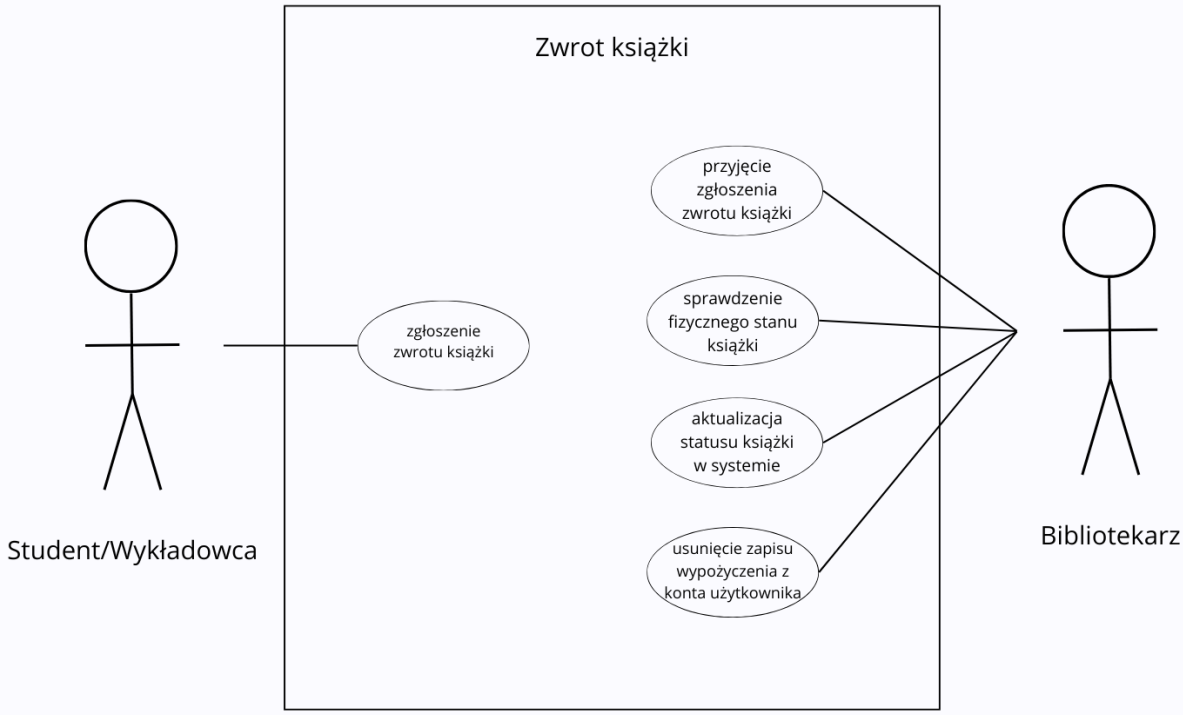
Celem projektu jest stworzenie bazy danych systemu zarządzania biblioteką uczelni, która umożliwi skuteczne administrowanie procesami zachodzącymi w bibliotece oraz obsługę potrzeb jej użytkowników.

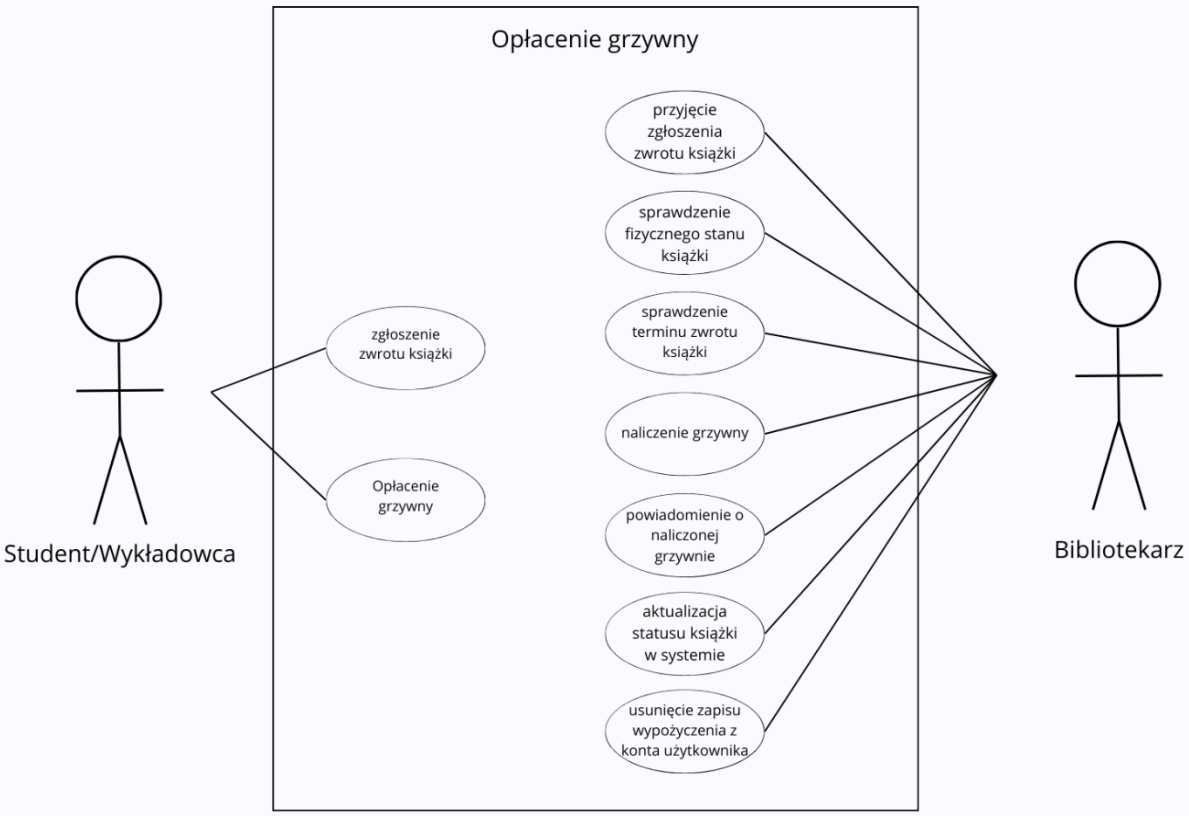
W pierwszej fazie projektu, opisanej w poniższym raporcie, skupiono się na fragmencie bazy danych obejmującym tabele związane z osobami korzystającymi z zasobów biblioteki (studenci i wykładowcy) oraz z jej funkcjonowaniem (wypożyczenia i grzywny). Dodatkowo zaprogramowano i opisano prototyp aplikacji internetowej połączonej z bazą danych, możliwej do wykorzystania w bibliotece.

Dokumentacja przypadków użycia

Poniżej przedstawiono diagramy przypadków użycia najistotniejszych procesów zachodzących w bibliotece celem ułatwienia specyfikacji wymagań stawianych systemowi oraz nakreślenia ram jego funkcjonalności.






Opłacenie grzywny
przyjęcie zgłoszenia zwrotu książki
sprawdzenie fizycznego stanu książki
sprawdzenie terminu zwrotu książki
naliczenie grzywny
powiadomienie o naliczonej grzywnie
aktualizacja statusu książki w systemie
usunięcie zapisu wypożyczenia z konta użytkownika
zgłoszenie zwrotu książki
Opłacenie grzywny
Student/Wykładowca
Bibliotekarz

Wymagania

W pierwszej fazie projektu skupiamy się na wymaganiach funkcjonalnych, aby móc stworzyć zalążek działającego systemu, mając również na uwadze dalsze etapy rozwoju w trakcie implementacji. Na podstawie diagramów przypadków użycia oraz specyfikacji zadania, można określić następujące wymagania:

W1 System powinien przechowywać informacje na temat wszystkich książek będących własnością biblioteki: ich tytuły, imiona i nazwiska autorów, nazwy wydawnictw, ich numery ISBN oraz ich daty i miejsca wydania.

W2 System powinien przechowywać informacje na temat wszystkich wypożyczeń, zwrotów oraz naliczonych grzywien.

W3 System powinien przechowywać podstawowe dane osobowe użytkowników biblioteki.

W4 System powinien umożliwiać użytkownikowi logowanie się do swojego konta użytkownika oraz wylogowanie się z niego.

W5 System powinien umożliwiać studentowi wypożyczanie maksymalnie 5 książek jednocześnie, każdej z nich na maksymalnie 14 dni.

W6 System powinien umożliwiać wykładowcy wypożyczanie maksymalnie 10 książek jednocześnie, każdej z nich na maksymalnie 21 dni.

W7 System powinien automatycznie naliczać grzywnę w przypadku niezwrócenia książki w terminie o wysokości 0.5 zł za każdy dzień po upłynięciu terminu zwrotu.

W8 System powinien umożliwiać użytkownikowi biblioteki spłatę naliczonej grzywny.

W9 System powinien umożliwiać użytkownikowi biblioteki przeglądanie zasobów biblioteki.

W10 System powinien umożliwiać użytkownikowi biblioteki pozyskanie szczegółowych informacji o każdej książce będącej własnością biblioteki.

W11 System powinien umożliwiać użytkownikowi zarejestrowanie zwrotu książki.

W12 System powinien umożliwiać użytkownikowi pozyskanie informacji o swoich wypożyczeniach, zarówno aktualnych jak i archiwalnych, oraz o ich statusach.

W13 System powinien umożliwiać użytkownikowi pozyskanie informacji o swoich grzywnach oraz o ich statusie.

W14 System powinien zapewniać bezpieczeństwo bazy danych i nie narażać jej integralności.

Opis koncepcji rozwiązania

Wybór technologii

Projekt systemu został stworzony z myślą o prostocie użytkowania, skalowalności i wydajności. Jako technologię wybrano relacyjną bazę danych głównie ze względu na strukturę przechowywanych w systemie informacji – dane są uporządkowane, schematyczne i jednorodne. Do realizacji wykorzystano dialekt MySQL, ze względu na jego zalety, takie jak uniwersalność, skalowalność czy ochronę danych, które będą miały istotne znaczenie na dalszych etapach rozwoju projektu.
Prototyp aplikacji internetowej napisano w języku Python, przy użyciu framework'a Flask. Jego główną zaletą jest modularność, znacznie ułatwiająca zarządzanie kodem oraz integrację z innymi komponentami systemu, takimi jak baza danych, a także prostota i elastyczność.

Koncepcja realizacji wymagań systemu

W poniższej tabeli przedstawiono plan realizacji wymagań określonych w sekcji *Wymagania*.

Wymaganie	Sposób realizacji
W1	Informacje dotyczące książek będą przechowywane w tabeli **TAB3** oraz tabeli słownikowej przechowującej imiona i nazwiska autorów **TAB1**.
W2	Informacje dotyczące zwrotów i grzywien będą przechowywane w tabelach **TAB2** i **TAB8**.
W3	Informacje użytkowników biblioteki będą przechowywane w tabeli ogólnej **TAB5** oraz szczegółowych `Student` (**TAB6**) i `Wykladowca` (**TAB7**), w zależności od grupy, do której użytkownik należy.
W4	Dane logowania będą przechowywane w tabeli **TAB4**. Proces logowania na poziomie aplikacji będzie umożliwiać funkcja **F3**, a wylogowania – funkcja **F4**.
W5, W6	Na tabeli **TAB8** zostaną zdefiniowane dwa triggery - **T1** i **T2**, sprawdzające limit wypożyczeń oraz definiujące czas, na jaki użytkownik może wypożyczyć książkę, w zależności od grupy, do której należy. Sam proces wypożyczania na poziomie bazy danych umożliwi procedura składowana **P3**, a na poziomie aplikacji – funkcja **F7**.
W7	Naliczanie grzywny będzie odbywać się przy zwrocie książki. Na poziomie bazy danych proces ten umożliwi procedura składowana **P4**.
W8	Spłatę grzywny umożliwią użytkownikowi: na poziomie bazy danych – procedura składowana **P6**, a na poziomie aplikacji – funkcja **F11**.
W9	Przeglądanie dostępnych zasobów biblioteki umożliwi widok **V1** wykorzystany wraz z funkcją aplikacji **F5**.
W10	Szczegółowe informacje o książkach dostępne będą dzięki procedurze składowanej **P1** oraz funkcji **F6** w aplikacji.

W11 Zarejestrowanie zwrotu książki umożliwią: procedura składowana **P4** na poziomie bazy danych oraz funkcja **F9** na poziomie aplikacji.

W12 Pozyskanie informacji o wypożyczeniach użytkownika umożliwią procedura składowana **P2** oraz funkcja **F8** w aplikacji.

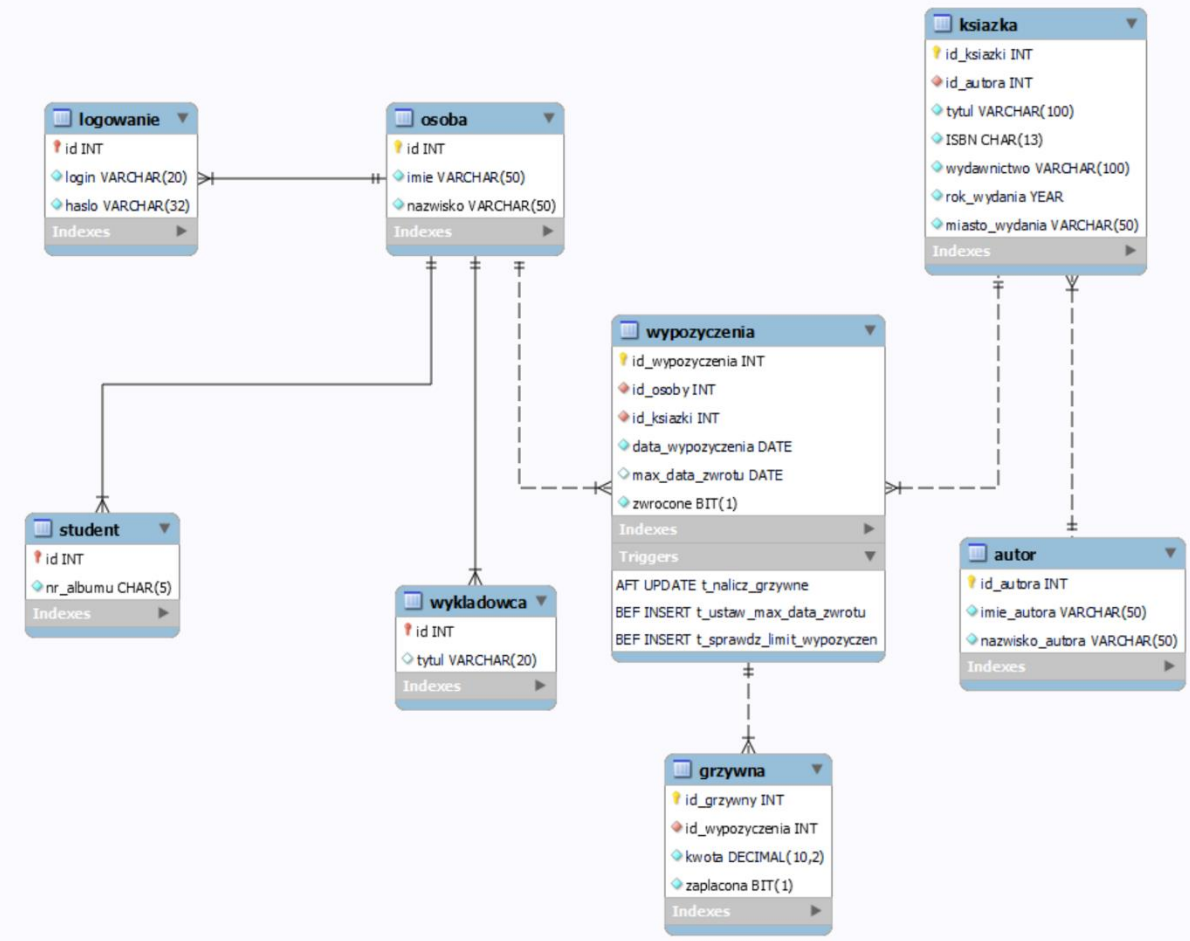
W13 Pozyskanie informacji o grzywnach użytkownika umożliwią procedura składowana **P5** oraz funkcja **F10** w aplikacji.

W14 Wszystkie operacje dokonywane przez użytkowników odbywają się przy pomocy procedur składowanych oraz widoku, co stanowi barierę bezpieczeństwa dla bazy danych.

Przyjęte założenia i uproszczenia

1. W związku z nieokreśleniem konkretnej liczby dni na wypożyczenie, limitów wypożyczeń oraz wysokości grzywny, przyjęto wartości opisane w sekcji *Wymagania*.
2. W pierwszej wersji projektu nie przewidziano specjalnej funkcjonalności dla pracowników biblioteki bądź administratorów i władz uczelni.
3. Nie opracowano funkcjonalności związanej z rejestracją użytkowników czy zmianą hasła, w założeniu powyższe czynności wykonywane są przez administratora.

Dokumentacja schematu bazy danych



Dokumentacja kompletu użytych elementów

Dokumentacja tabel bazy danych

Tabele ułożone w kolejności alfabetycznej, nie w kolejności implementacji. Domyślnie wartości są `NOT NULL`, chyba, że sprecyzowano inaczej.

TAB1	**`Autor`** Tabela słownikowa przechowująca informacje o wszystkich autorach książek, których prace dostępne są w bibliotece.	
`id_autora`	`INT (PK)`	Automatycznie inkrementowany identyfikator autora.
`imie_autora`	`VARCHAR(50)`	Przechowuje imię autora książki.
`nazwisko_autora`	`VARCHAR(50)`	Przechowuje nazwisko autora książki. *Komentarz*: połączenie imię + nazwisko jest unikalne.

TAB2	**`Grzywna`** Przechowuje informacje o naliczonych grzywnach.	
`id_grzywny`	`INT (PK)`	Automatycznie inkrementowany identyfikator grzywny.
`id_wypozyczenia`	`INT (FK)`	Identyfikator wypożyczenia, którego dotyczy grzywna. Jako `FOREIGN KEY` łączy tabele `Grzywna` i `Wypozyczenie`.
`kwota`	`DECIMAL(10,2)`	Naliczona kwota grzywny. Jest automatycznie inkrementowana o 0.5 zł z każdym dniem po przekroczeniu `Wypozyczenie.max_data_wypozyczenia`.
`zaplacona`	`BIT`	Przechowuje informację o tym, czy grzywna została opłacona pod postacią wartości logicznej (`0` – nieopłacona, `1` – opłacona).

TAB3	**Ksiazka** Przechowuje informacje o wszystkich książkach dostępnych w bibliotece.	
id_ksiazki	INT (**PK**)	Przechowuje automatycznie inkrementowany identyfikator książki.
id_autora	INT (**FK**)	Przechowuje identyfikator autora; jako FOREIGN KEY łączy tabele Ksiazka i Autor.
tytul	VARCHAR(100)	Przechowuje tytuł książki.
ISBN	CHAR(13)	Przechowuje numer ISBN książki.
wydawnictwo	VARCHAR(100)	Przechowuje nazwę wydawnictwa, które wydało książkę.
rok_wydania	YEAR	Przechowuje rok wydania książki.
miasto_wydania	VARCHAR(50)	Przechowuje nazwę miasta, w którym wydano książkę.

TAB4	**Logowanie** Przechowuje informacje wykorzystywane do logowania użytkowników.	
id	INT (**PK**, **FK**)	Przechowuje identyfikator osoby logującej się; jako FOREIGN KEY łączy tabele Logowanie i Osoba.
login	VARCHAR(20)	Przechowuje login użytkownika.
haslo	VARCHAR(32)	Przechowuje hasło użytkownika. *Komentarz*: przyjęto uproszczenie polegające na przechowywaniu haseł w formie ciągu znakowego, co w komercyjnej bazie danych narażałoby bezpieczeństwo użytkowników i należałoby zastosować hash.

TAB5	**Osoba** Przechowuje informacje o wszystkich osobach korzystających z biblioteki.	
id	INT (**PK**)	Automatycznie inkrementowany identyfikator osoby.
imie	VARCHAR(50)	Przechowuje imię osoby.
nazwisko	VARCHAR(50)	Przechowuje nazwisko osoby.

TAB6	**Student** Przechowuje informacje o wszystkich studentach korzystających z biblioteki.	
id	INT (**PK**, **FK**)	Przechowuje identyfikator studenta; jako FOREIGN KEY łączy tabele Student i Osoba.
nr_albumu	CHAR(5)	Przechowuje unikalny pięciocyfrowy numer albumu stduenta.

TAB7	**Wykladowca** Przechowuje informacje o wszystkich wykładowcach korzystających z biblioteki.	
id	INT (**PK**, **FK**)	Przechowuje identyfikator wykładowcy; jako FOREIGN KEY łączy tabele Wykladowca i Osoba.
tytul	VARCHAR(20)	Przechowuje skróconą formę tytułu naukowego/zawodowego wykładowcy. Może przyjmować wartości NULL.

TAB8	**Wypozyczenia** Przechowuje informacje o wszystkich wypożyczeniach w bibliotece.	
id_wypozyczenia	INT (**PK**)	Przechowuje automatycznie inkrementowany identyfikator wypożyczenia.
id_osoby	INT (**PK**, **FK**)	Przechowuje identyfikator osoby wypożyczającej; jako FOREIGN KEY łączy tabele Wypozyczenia i Osoba.
id_ksiazki	INT (**FK**)	Przechowuje identyfikator książki, której dotyczy wypożyczenie; jako FOREIGN KEY łączy tabele Wypozyczenia i Ksiazka.
data_wypozyczenia	DATE	Przechowuje datę wypożyczenia książki.
max_data_zwrotu	DATE	Przechowuje maksymalną datę, do której książka musi zostać zwrócona celem uniknięcia naliczenia grzywny. Obliczana przy pomocy triggera. Może przyjmować wartości NULL.
zwrocone	BIT	Przechowuje informację o tym, czy książka została zwrócona pod postacią wartości logicznej (0 – niezwrócona, 1 – zwrócona).

Komentarz dotyczący wyboru kluczy głównych

We wszystkich tabelach zdecydowano o wykorzystaniu automatycznie inkrementowanych liczb całkowitych, ponieważ spełniają wszystkie zasady jakie stawia się kluczom głównym (jednoznacznie identyfikują wiersze, są niezmienne w trakcie życia wiersza, są krótkie i atomowe) oraz nie narażają bezpieczeństwa użytkowników systemu.

Dokumentacja triggerów

T1 Trigger `t_ustaw_max_data_zwrotu` (na tabeli `Wypozyczenia`) – jego zadaniem jest automatyczne ustawienie maksymalnej daty, do której książka musi zostać zwrócona do biblioteki, żeby uniknąć naliczenia grzywny. Sprawdza, czy osobą wypożyczającą jest student, czy wykładowca i zgodnie z tą informacją oblicza stosowny maksymalny termin zwrotu (14 dni dla studenta i 21 dni dla wykładowcy).

T2 Trigger `t_sprawdz_limit_wypozyczen` (na tabeli `Wypozyczenia`) – ma za zadanie zapewnienie przestrzegania przez użytkowników biblioteki limitów wypożyczeń. W przypadku jeśli student chce wypożyczyć więcej niż 5 książek, bądź wykładowca więcej niż 10 książek, trigger spowoduje, że procedura wypożyczenia się nie powiedzie.

Dokumentacja widoków

V1 Widok `v_ksiazki` – wyświetla listę wszystkich dostępnych w bibliotece książek i informacji o nich. Dodatkowo widok pozwala użytkownikom wyświetlić jedynie książki, które nie są w danym momencie wypożyczone, co znacznie skraca czas dokonywania wypożyczeń.

Dokumentacja procedur składowanych

P1 Procedura `p_info_ksiazka` – służy do pobierania i zwracania szczegółowych informacji o konkretnej książce z bazy danych na podstawie jej identyfikatora.

P2 Procedura `p_wypozyczenia` – zwraca szczegółowe informacje o wszystkich wypożyczeniach dokonanych przez konkretnego użytkownika na podstawie jego identyfikatora.

P3 Procedura `p_wypozycz_ksiazke` – umożliwia użytkownikowi dokonanie nowego wypożyczenia (tj. dodania nowego wpisu do tabeli `Wypozyczenia`). Służy do rejestrowania wszystkich wypożyczeń dokonywanych w bibliotece.

P4 Procedura `p_zwroc_ksiazke` – obsługuje proces zwrotu książki w systemie. Aktualizuje status zwrotu w tabeli `Wypozyczenia`, oblicza ewentualne opóźnienie, i jeśli takie występuje, na jego podstawie nalicza odpowiednią grzywnę. Procedura uwzględnia bieżącą datę zwrotu i porównuje ją z maksymalną dopuszczalną datą zwrotu, aby określić wysokość należnej kary za opóźnienie (przyjmując karę 0.5 zł za każdy dzień zwłoki).

P5 Procedura `p_grzywny` – służy do pobierania i wyświetlania szczegółowych informacji o grzywnach nałożonych na konkretnego użytkownika biblioteki.

P6 Procedura `p_oplac_grzywne` – aktualizuje status grzywny w tabeli `Grzywna` (`zaplacona`), oznaczając ją jako zapłaconą i jednocześnie rejestrując fakt uiszczenia płatności na podstawie identyfikatora grzywny.

Kod SQL

Kod służący do wygenerowania opracowanego schematu bazy danych

Tabele

TAB1

```
CREATE TABLE Autor (
    id_autora INT AUTO_INCREMENT PRIMARY KEY,
    imie_autora VARCHAR(50) NOT NULL,
    nazwisko_autora VARCHAR(50) NOT NULL,
    UNIQUE (imie_autora, nazwisko_autora)
);
```

TAB2

```
CREATE TABLE Grzywna (
    id_grzywny INT AUTO_INCREMENT PRIMARY KEY,
    id_wypozyczenia INT NOT NULL,
    kwota DECIMAL(10,2) NOT NULL,
    zaplacona BIT DEFAULT 0 NOT NULL,
    FOREIGN KEY (id_wypozyczenia) REFERENCES Wypozyczenia(id_wypozyczenia)
);
```

TAB3

```
CREATE TABLE Ksiazka (
    id_ksiazki INT AUTO_INCREMENT PRIMARY KEY,
    id_autora INT NOT NULL,
    tytul VARCHAR(100) NOT NULL,
    ISBN CHAR(13) NOT NULL,
    wydawnictwo VARCHAR(100) NOT NULL,
    rok_wydania YEAR NOT NULL,
    miasto_wydania VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_autora) REFERENCES Autor(id_autora)
);
```

TAB4

```
CREATE TABLE Logowanie (
    id INT PRIMARY KEY,
    login VARCHAR(20) NOT NULL,
    haslo VARCHAR(32) NOT NULL,
    FOREIGN KEY (id) REFERENCES Osoba(id)
);
```

TAB5

```
CREATE TABLE Osoba (
    id INT AUTO_INCREMENT PRIMARY KEY,
    imie VARCHAR(50) NOT NULL,
    nazwisko VARCHAR(50) NOT NULL
);
```

TAB6

```
CREATE TABLE Student (
    id INT PRIMARY KEY,
    nr_albumu CHAR(5) NOT NULL UNIQUE,
    FOREIGN KEY (id) REFERENCES Osoba(id)
);
```

TAB7

```
CREATE TABLE Wykladowca (
    id INT PRIMARY KEY,
    tytul VARCHAR(20),
    FOREIGN KEY (id) REFERENCES Osoba(id)
);
```

TAB8

```
CREATE TABLE Wypozyczenia (
    id_wypozyczenia INT AUTO_INCREMENT PRIMARY KEY,
    id_osoby INT NOT NULL,
    id_ksiazki INT NOT NULL,
    data_wypozyczenia DATE NOT NULL,
    max_data_zwrotu DATE,
    zwrocone BIT DEFAULT 0 NOT NULL,
    FOREIGN KEY (id_osoby) REFERENCES Osoba(id),
    FOREIGN KEY (id_ksiazki) REFERENCES Ksiazka(id_ksiazki)
);
```

Triggery

T1

```
DELIMITER //
CREATE TRIGGER t_ustaw_max_data_zwrotu
BEFORE INSERT ON Wypozyczenia
FOR EACH ROW
BEGIN
    DECLARE userType CHAR(1);

    IF EXISTS (SELECT 1 FROM Student WHERE id = NEW.id_osoby) THEN
        SET userType = 'S';
    ELSEIF EXISTS (SELECT 1 FROM Wykladowca WHERE id = NEW.id_osoby) THEN
        SET userType = 'W';
    END IF;

    IF userType = 'S' THEN
        SET NEW.max_data_zwrotu = DATE_ADD(NEW.data_wypozyczenia,
                                           INTERVAL 14 DAY);
    ELSEIF userType = 'W' THEN
        SET NEW.max_data_zwrotu = DATE_ADD(NEW.data_wypozyczenia,
                                           INTERVAL 21 DAY);
    END IF;
END // DELIMITER ;
```

T2

```
DELIMITER //
CREATE TRIGGER t_sprawdz_limit_wypozyczen
BEFORE INSERT ON Wypozyczenia
FOR EACH ROW
BEGIN
    DECLARE liczba_wypozyczen INT;
    DECLARE max_wypozyczen INT;

    IF EXISTS (SELECT 1 FROM Student WHERE id = NEW.id_osoby) THEN
        SET max_wypozyczen = 5;
    ELSEIF EXISTS (SELECT 1 FROM Wykladowca WHERE id = NEW.id_osoby) THEN
        SET max_wypozyczen = 10;
    END IF;

    SELECT COUNT(*) INTO liczba_wypozyczen
    FROM Wypozyczenia
    WHERE id_osoby = NEW.id_osoby AND zwrocone = 0;

    IF liczba_wypozyczen >= max_wypozyczen THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Przekroczony limit wypożyczeń.';
    END IF;
END // DELIMITER ;
```

Procedury składowane

P1

```
DELIMITER //
CREATE PROCEDURE p_info_ksiazka(IN ksiazka_id INT)
BEGIN
    SELECT K.id_ksiazki, K.tytul, A.imie_autora, A.nazwisko_autora,
           K.ISBN, K.wydawnictwo, K.rok_wydania, K.miasto_wydania
    FROM Ksiazka K
    JOIN Autor A ON K.id_autora = A.id_autora
    WHERE K.id_ksiazki = ksiazka_id;
END // DELIMITER ;
```

P2

```
DELIMITER //
CREATE PROCEDURE p_wypozyczenia(IN osoba_id INT)
BEGIN
    SELECT W.id_wypozyczenia, W.id_ksiazki, K.tytul,
           A.imie_autora, A.nazwisko_autora,
           W.data_wypozyczenia, W.max_data_zwrotu, W.zwrocone
    FROM Wypozyczenia W
    JOIN Ksiazka K ON W.id_ksiazki = K.id_ksiazki
    JOIN Autor A ON K.id_autora = A.id_autora
    WHERE W.id_osoby = osoba_id
    ORDER BY data_wypozyczenia DESC, tytul ASC;
END // DELIMITER ;
```

P3

```
DELIMITER //
CREATE PROCEDURE p_wypozycz_ksiazke(IN p_id_osoby INT, IN p_id_ksiazki INT,
                                    IN p_data_wypozyczenia DATE)
BEGIN
    INSERT INTO Wypozyczenia (id_osoby, id_ksiazki, data_wypozyczenia)
    VALUES (p_id_osoby, p_id_ksiazki, p_data_wypozyczenia);
END // DELIMITER ;
```

P4

```
DELIMITER //
CREATE PROCEDURE p_zwroc_ksiazke (IN p_id_wypozyczenia INT)
BEGIN
    DECLARE v_max_data_zwrotu DATE;
    DECLARE v_data_zwrotu DATE;
    DECLARE v_opoznienie INT;
    DECLARE v_kwota DECIMAL(10,2);

    SELECT max_data_zwrotu, CURDATE() INTO v_max_data_zwrotu, v_data_zwrotu
    FROM Wypozyczenia
    WHERE id_wypozyczenia = p_id_wypozyczenia;

    UPDATE Wypozyczenia
    SET zwrocone = 1
    WHERE id_wypozyczenia = p_id_wypozyczenia;

    SET v_opoznienie = DATEDIFF(v_data_zwrotu, v_max_data_zwrotu);

    IF v_opoznienie > 0 THEN
        SET v_kwota = v_opoznienie * 0.5;
        INSERT INTO Grzywna (id_wypozyczenia, kwota)
        VALUES (p_id_wypozyczenia, v_kwota);
    END IF;
END // DELIMITER ;
```

P5

```
DELIMITER //
CREATE PROCEDURE p_grzywny (IN osoba_id INT)
BEGIN
    SELECT id_grzywny, Grzywna.id_wypozyczenia, tytul,
           imie_autora, nazwisko_autora, data_wypozyczenia,
           DATEDIFF(CURDATE(), max_data_zwrotu) AS opoznienie, kwota,
           zaplacona
    FROM Grzywna
    JOIN Wypozyczenia
        ON Grzywna.id_wypozyczenia = Wypozyczenia.id_wypozyczenia
    JOIN Ksiazka ON Wypozyczenia.id_ksiazki = Ksiazka.id_ksiazki
    JOIN Autor ON Ksiazka.id_autora = Autor.id_autora
    WHERE Wypozyczenia.id_osoby = osoba_id
    ORDER BY opoznienie DESC, tytul ASC;
END // DELIMITER ;
```

P6

```
DELIMITER //
CREATE PROCEDURE p_oplac_grzywne (IN p_id_grzywny INT)
BEGIN
    UPDATE Grzywna
    SET zaplacona = 1
    WHERE id_grzywny = p_id_grzywny;
END // DELIMITER ;
```

Widoki

V1

```
DELIMITER //
CREATE VIEW v_ksiazki AS
    SELECT K.id_ksiazki, K.tytul, A.imie_autora, A.nazwisko_autora, K.ISBN,
           K.wydawnictwo, K.rok_wydania, K.miasto_wydania
    FROM Ksiazka K
    JOIN Autor A ON K.id_autora = A.id_autora
    LEFT JOIN Wypozyczenia W ON K.id_ksiazki = W.id_ksiazki
    WHERE W.id_ksiazki IS NULL;
// DELIMITER ;
```

Kod służący do wprowadzenia przykładowych danych do poszczególnych tabel systemu

TAB1

```
INSERT INTO Autor (imie_autora, nazwisko_autora) VALUES
    ('Valentina','Alto'),
    ('Filip','Sala'),
    ('Aditya','Bhargava'),
    ('Ben','Forta'),
    ('Eric','Matthes'),
    ('Aurelien','Geron'),
    ('Josef','Brockmann-Muller'),
    ('Rishal','Hurbans'),
    ('Michael','Dawson'),
    ('Donella H.','Meadows'),
    ('Andrzej','Pikoń'),
    ('Cory','Althoff'),
    ('Roland','Zimek'),
    ('Michał','Wiszniewski'),
    ('Robert C.','Martin');
```

TAB2

```
-- Wiersze do tej tabeli dodawane są za pomocą triggera. Przykład manualnego
dodania danych:
INSERT INTO Grzywna (id_wypozyczenia, kwota, zaplacona) VALUES
    (1, 10.00, 1),
    (2, 7.50, 0),
    (3, 2.50, 1),
    (4, 12.50, 1),
    (5, 5.00, 0);
```

TAB3

```
INSERT INTO Ksiazka (id_autora, tytul, ISBN, wydawnictwo, rok_wydania,
miasto_wydania) VALUES
    (1, 'Generatywna sztuczna inteligencja z ChatGPT i modelami OpenAi',
        '9820284921872', 'Helion', 2024, 'Warszawa'),
    (1, 'Generatywna sztuczna inteligencja z ChatGPT i modelami OpenAi',
        '9820284921872', 'Helion', 2024, 'Warszawa'),
    (2, 'ChatGPT. Podstawy i proste zastosowania',
        '9873867392810', 'Helion', 2024, 'Warszawa'),
    (2, 'ChatGPT. Podstawy i proste zastosowania',
        '9873867392810', 'Helion', 2024, 'Warszawa'),
    (2, 'ChatGPT. Podstawy i proste zastosowania',
        '9873867392810', 'Helion', 2024, 'Warszawa'),
    (2, 'ChatGPT. Podstawy i proste zastosowania',
        '9873867392810', 'Helion', 2024, 'Warszawa'),
    (3, 'Algorytmy. Ilustrowany przewodnik',
        '9873620198322', 'Helion', 2017, 'Warszawa'),
    (3, 'Algorytmy. Ilustrowany przewodnik',
        '9873620198322', 'Helion', 2017, 'Warszawa'),
    (3, 'Algorytmy. Ilustrowany przewodnik',
        '9873620198322', 'Helion', 2017, 'Warszawa'),
    (3, 'Algorytmy. Ilustrowany przewodnik',
        '9873620198322', 'Helion', 2017, 'Warszawa'),
    (3, 'Algorytmy. Ilustrowany przewodnik',
```

```
    '9873620198322', 'Helion', 2017, 'Warszawa'),
(4, 'SQL w mgnieniu oka. Opanuj język zapytań w 10 minut dziennie',
    '9867291827109', 'Helion', 2020, 'Warszawa'),
(5, 'Python. Instrukcje dla programisty',
    '9872736520917', 'Helion', 2023, 'Kraków'),
(6, 'Uczenie maszynowe z użyciem Scikit-Learn, Keras i TensorFlow',
    '9872617289283', 'Helion', 2023, 'Gdańsk'),
(7, 'Systemy siatek w projektowaniu graficznym',
    '9872091736251', 'd2d.pl', 2020, 'Łódź'),
(7, 'Systemy siatek w projektowaniu graficznym',
    '9872091736251', 'd2d.pl', 2020, 'Łódź'),
(8, 'Algorytmy sztucznej inteligencji. Ilustrowany przewodnik',
    '9837263986293', 'Helion', 2021, 'Kraków'),
(8, 'Algorytmy sztucznej inteligencji. Ilustrowany przewodnik',
    '9837263986293', 'Helion', 2021, 'Kraków'),
(8, 'Algorytmy sztucznej inteligencji. Ilustrowany przewodnik',
    '9837263986293', 'Helion', 2021, 'Kraków'),
(9, 'Python dla każdego. Podstawy programowania',
    '9836271928371', 'Helion', 2014, 'Kraków'),
(9, 'Python dla każdego. Podstawy programowania',
    '9836271928371', 'Helion', 2014, 'Kraków'),
(10, 'Myślenie systemowe. Wprowadzenie',
     '9837485032917', 'Helion', 2022, 'Warszawa'),
(11, 'AutoCAD 2024 PL. Pierwsze kroki',
     '9827364928172', 'Helion', 2023, 'Gdańsk'),
(11, 'AutoCAD 2024 PL. Pierwsze kroki',
     '9827364928172', 'Helion', 2023, 'Gdańsk'),
(12, 'Informatyk samouk', '9374293840591', 'Helion', 2022, 'Kraków'),
(12, 'Informatyk samouk', '9374293840591', 'Helion', 2022, 'Kraków'),
(12, 'Informatyk samouk', '9374293840591', 'Helion', 2022, 'Kraków'),
(12, 'Informatyk samouk', '9374293840591', 'Helion', 2022, 'Kraków'),
(12, 'Informatyk samouk', '9374293840591', 'Helion', 2022, 'Kraków'),
(13, 'Python na maturze. Rozwiązania i analiza wybranych zadań
    programistycznych', '8273927483910', 'Helion', 2021, 'Szczecin'),
(14, 'Python na start! Programowanie dla nastolatków',
     '9837462781920', 'Helion', 2017, 'Gdańsk'),
(15, 'Czysta architektura. Struktura i design oprogramowania.
      Przewodnik dla profesjonalistów',
     '9374635284398', 'Helion', 2022, 'Warszawa'),
(15, 'Czysta architektura. Struktura i design oprogramowania.
      Przewodnik dla profesjonalistów',
     '9374635284398', 'Helion', 2022, 'Warszawa'),
(15, 'Czysta architektura. Struktura i design oprogramowania.
      Przewodnik dla profesjonalistów',
     '9374635284398', 'Helion', 2022, 'Warszawa');
```

TAB4

```
INSERT INTO Logowanie (id, login, haslo) VALUES
    (1, 'amejasi', 'haslo.123'),
    (2, 'hubegaj', 'piesek.789'),
    (3, 'lilib', 'kotek000'),
    (4, 'julczaj', 'czek0ladaBiala'),
    (5, 'adszew', '93716haslo0'),
    (6, 'matebar', 'MmM2290L'),
    (7, 'luand', 'LucjaJulianna.534'),
    (8, 'franole', 'JablKo0192'),
    (9, 'zuzada', 'paczki2019'),
    (10, 'bartsi', 'p@ssword.333'),
    (11, 'urszuk', 'orzechoweCiasto420'),
    (12, 'seweluk', 'NiebieskiDlugopis99'),
    (13, 'przemszym', 'LubiePizze1'),
    (14, 'barbzaw', 'Maksio.990'),
    (15, 'bozewit', '123ananas');
```

TAB5

```
-- Osoby-studenci
INSERT INTO Osoba (imie, nazwisko) VALUES
    ('Amelia', 'Jasińska'),
    ('Hubert', 'Gajewski'),
    ('Liliana', 'Brzezińska'),
    ('Julian', 'Czajkowski'),
    ('Adrianna', 'Szewczyk'),
    ('Mateusz', 'Baran'),
    ('Łucja', 'Andrzejczak'),
    ('Franciszek', 'Olejniczak'),
    ('Zuzanna', 'Dąbrowska'),
    ('Bartosz', 'Sikora');

-- Osoby-wykładowcy
INSERT INTO Osoba (imie, nazwisko) VALUES
    ('Urszula', 'Żukowska'),
    ('Seweryn', 'Łukasiewicz'),
    ('Przemysław', 'Szymański'),
    ('Barbara', 'Zawadzka'),
    ('Bożena', 'Witkowska');
```

TAB6

```
INSERT INTO Student (id, nr_albumu) VALUES
    (1, '43309'),
    (2, '43310'),
    (3, '43311'),
    (4, '43312'),
    (5, '43313'),
    (6, '43314'),
    (7, '43315'),
    (8, '43316'),
    (9, '43317'),
    (10, '43318');
```

TAB7

```
INSERT INTO Wykladowca (id, tytul) VALUES
    (11, 'dr'),
    (12, 'mgr'),
    (13, 'dr inż.'),
    (14, 'prof.'),
    (15, 'dr');
```

TAB8

```
INSERT INTO Wypozyczenia (id_osoby, id_ksiazki, data_wypozyczenia) VALUES
    (2, 1, '2024-05-22'),
    (2, 2, '2024-06-02'),
    (2, 3, '2024-06-19'),
    (11, 4, '2024-05-15'),
    (10, 14, '2024-06-01');
```

Aplikacja

Struktura projektu

```
PROJEKT_BIBLIOTEKA
│   main.py
│
├───static
│       style.css
│
└───templates
        grzywna_oplacona.html
        grzywny.html
        ksiazka_szczegoly.html
        lista.html
        login.html
        start.html
        wypozyczenia.html
        wypozyczono.html
        zwrocono.html
```

main.py Główny plik projektu; zawiera niezbędne funkcje i ich wywołania, ścieżki, łączy aplikację z bazą danych oraz renderuje odpowiednie szablony dokumentów.

/templates Zgodnie z konwencją nazewnictwa Flask jest to folder zawierający wszystkie szablony dokumentów.

/static Zgodnie z konwencją nazewnictwa Flask jest to folder zawierający wszystkie pliki stylów *CSS* oraz pliki *JavaScript*.

Dokumentacja zasobów interfejsu użytkownika

/static

style.css	Plik zawierający definicję stylów *CSS* różnych elementów aplikacji internetowej. Importuje czcionki, ustawia podstawowe style dla wykorzystywanych tagów *HTML*. Zapewnia estetyczne i spójne wyświetlanie komponentów strony.

/templates

grzywna_oplacona.html	Szablon dokumentu wyświetlanego po opłaceniu przez zalogowanego użytkownika grzywny.
grzywny.html	Szablon dokumentu wyświetlającego listę wszystkich grzywien zalogowanego użytkownika.
ksiazka_szczegoly.html	Szablon dokumentu wyświetlającego szczegółowe informacje o konkretnej książce, wybranej przez użytkownika z listy.
lista.html	Szablon dokumentu wyświetlającego listę wszystkich dostępnych do wypożyczenia książek.
login.html	Szablon strony logowania.
start.html	Szablon strony głównej.
wypozyczenia.html	Szablon dokumentu wyświetlającego listę książek wypożyczonych przez zalogowanego użytkownika.
wypozyczono.html	Szablon dokumentu wyświetlanego po wypożyczeniu książki przez zalogowanego użytkownika.
zwrocono.html	Szablon dokumentu wyświetlanego po dokonaniu zwrotu książki przez zalogowanego użytkownika.

Dokumentacja najważniejszych funkcji (`main.py`)

F1	`get_db_connection()`	Funkcja nawiązuje połączenie z bazą danych przy użyciu konfiguracji zapisanej w `db_config`. Jeśli połączenie się nie powiedzie z powodu nieprawidłowej nazwy użytkownika lub hasła lub brakującej bazy danych, wypisuje odpowiedni komunikat. W przypadku innego błędu wypisuje ogólny komunikat błędu.
F2	`hello()`	Funkcja sprawdzająca, czy użytkownik jest zalogowany: jeśli tak – przekierowuje użytkownika na stronę główną, jeśli nie – na stronę logowania.
F3	`login()`	Funkcja obsługuje proces logowania użytkownika. Jeśli żądanie jest metodą `POST`, pobiera nazwę użytkownika i hasło z formularza, następnie nawiązuje połączenie z bazą danych za pomocą funkcji `get_db_connection()`, po czym sprawdza zgodność danych przesyłanych przez formularz logowania z tabelą `Logowanie` w bazie danych. Jeśli dane logowania są poprawne, ustawia sesję użytkownika i przekierowuje do strony głównej. W przypadku nieprawidłowych danych logowania, zwraca odpowiedni komunikat. Również w przypadku problemów z połączeniem z bazą danych zwraca odpowiedni komunikat błędu.
F4	`logout()`	Funkcja umożliwiająca wylogowanie się użytkownika i tym samym zakończenie sesji.
F5	`lista_ksiazek()`	Funkcja pobiera listę książek z bazy danych korzystając z widoku `v_ksiazki`, aby wyrenderować ją w formie tabeli. Jeśli połączenie z bazą danych za pomocą funkcji `get_db_connection()` się powiedzie, wykonuje zapytanie SQL, które zwraca informacje o książkach. Następnie zamyka połączenie i przekazuje dane książek do szablonu *HTML* `lista.html`, aby wyświetlić je na stronie. W przypadku problemów z połączeniem z bazą danych, zwraca komunikat informujący o błędzie.
F6	`ksiazka_details(id_ksiazki)`	Funkcja pobiera szczegóły książki o podanym ID z bazy danych. Wykonuje procedurę `p_info_ksiazka` i pobiera jej wynik. Jeśli książka o podanym ID istnieje, renderuje szablon *HTML* z jej szczegółami, w przeciwnym razie generuje komunikat o braku książki. Obsługuje także błędy połączenia z bazą danych i błędy wywołania procedury.

F7	`wypozycz_ksiazke()`	Funkcja obsługuje proces wypożyczania książki przez użytkownika. Jeśli żądanie jest metodą POST, pobiera ID książki z formularza i sprawdza jego poprawność. Następnie nawiązuje połączenie z bazą danych przy pomocy funkcji `get_db_connection()` i sprawdza przez sesję czy użytkownik jest zalogowany. Jeśli powyższe warunki są spełnione, wykonuje procedurę składowaną `p_wypozycz_ksiazke`, aby zarejestrować wypożyczenie książki przez zalogowanego użytkownika. Po zakończeniu operacji zamyka połączenie i wyświetla komunikat potwierdzający wypożyczenie książki przy użyciu `wypozyczono.html`. Obsługuje błędy związane z połączeniem z bazą danych, wykonaniem operacji na bazie danych czy używaną metodą (jeśli nie jest nią POST).
F8	`wypozyczenia()`	Funkcja pobiera listę wypożyczeń zalogowanego użytkownika. Najpierw sprawdza, czy użytkownik jest zalogowany i jeśli nie, przekierowuje go do strony logowania, a następnie nawiązuje połączenie z bazą danych za pomocą funkcji `gt_db_connetion()`. Jeśli połączenie się powiedzie, wykonuje procedurę `p_wypozyczenia`, aby pobrać listę wypożyczeń zalogowanego użytkownika na podstawie jego ID. Wyniki wywołania procedury są pobierane za pomocą pętli i przypisywane do zmiennej `borrowings`. Po zakończeniu operacji połączenie z bazą danych jest zamykane, a dane są przekazywane do `wypozyczenia.html`, aby wyświetlić je na stronie. Funkcja obsługuje także błędy związane z połączeniem z bazą danych, wykonaniem procedury oraz inne błędy ogólne.
F9	`zwroc_ksiazke()`	Funkcja obsługuje proces zwracania książki przez zalogowanego użytkownika. Jeśli żądanie jest metodą POST, z formularza pobierane jest `id_wypozyczenia`, które jest dokonywane. Następnie za pomocą funkcji `get_db_connection()` nawiązywane jest połączenie z bazą danych i jeśli się ono powiedzie, wywoływana jest procedura składowana `p_zwroc_ksiazke` z podanym ID wypożyczenia. Po zakończeniu jej wykonywania, połączenie z bazą danych jest zamykane i renderowany jest szablon `zwrocono.html`. Funkcja obsługuje także błędy związane z połączeniem z bazą danych, wykonaniem procedury, metodą (jeśli nie jest nią POST), ID wypożyczenia oraz inne błędy ogólne.

F10	`grzywny()`	Funkcja pobiera listę grzywien zalogowanego użytkownika. Po sprawdzeniu czy użytkownik jest zalogowany, albo przekierowuje go do strony logowania, albo rozpoczyna nawiązanie połączenia z bazą danych poprzez funkcję `get_db_connection()`. Jeśli połączenie się uda, wykonuje procedurę składowaną `p_grzywny`, aby pobrać listę grzywien zalogowanego użytkownika na podstawie jego ID przechowywanego w sesji. Wyniki są pobierane za pomocą pętli i przechowywane w zmiennej `grzywny`. Po zakończeniu operacji połączenie z bazą danych jest zamykane, a dane są przekazywane do szablonu `grzywny.html`, aby wyświetlić je na stronie. Funkcja obsługuje błędy związane z połączeniem z bazą danych, wykonaniem procedury oraz inne błędy ogólne.
F11	`oplac_grzywne()`	Funkcja obsługuje proces opłacania grzywny za przetrzymanie książki. Początkowo sprawdza, czy żądanie jest metodą POST i jeśli tak, pobiera ID grzywny z formularza i sprawdza poprawność jej identyfikatora. Następnie nawiązuje połączenie z bazą danych przy pomocy funkcji `get_db_connection()` i wywołuje procedurę składowaną `p_oplac_grzywne` z podanym ID grzywny, aby zarejestrować jej spłatę. Na koniec zamyka połączenie z bazą danych i renderuje szablon potwierdzenia spłaty. Funkcja obsługuje błędy związane z identyfikatorem grzywny, połączeniem z bazą danych, wykonaniem procedury składowanej, metodą (jeśli nie jest nią POST) oraz inne błędy ogólne.

Spis załączników

Wszystkie dodatkowe pliki projektu znajdują się w załączonym archiwum `pliki_projektu.zip`.

1. `prezentacja_dzialania.mp4` – krótki film ilustrujący działanie stworzonej aplikacji oraz jej współpracę z bazą danych.
   Źródło muzyki do filmu: https://www.youtube.com/watch?v=Bt426WbUpow

2. `kod_zrodlowy` – folder zawierający wszystkie pliki kodu aplikacji - `/static`, `/templates` oraz `main.py`.