System zarządzania biblioteką szkolną.

Use Case – opis głównych funkcjonalności systemu.

System zarządzania biblioteką szkolną ma na celu usprawnienie procesu wypożyczania i zarządzania zasobami bibliotecznymi. Poniżej znajdują się główne funkcjonalności systemu:

1. Zarządzanie Uczniami:
   * Dodawanie, edycja i usuwanie uczniów.
   * Przeglądanie informacji o uczniach (imię, nazwisko, klasa).
2. Zarządzanie Książkami:
   * Dodawanie nowych książek do systemu (tytuł, kategoria, autor, wydawca).
   * Aktualizacja informacji o istniejących książkach.
   * Wyszukiwanie książek po tytule, autorze lub kategorii.
3. Wypożyczanie Książek:
   * Przypisywanie książek do uczniów.
   * Śledzenie daty wypożyczenia i zwrotu.
   * Monitorowanie zaległych zwrotów.
4. Zarządzanie Kategoriami:
   * Dodawanie nowych kategorii książek.
   * Przeglądanie listy dostępnych kategorii.
5. Zarządzanie Autorami i Wydawcami:
   * Dodawanie nowych autorów i wydawców.
   * Przeglądanie listy autorów i wydawców.
6. Raporty i Statystyki:
   * Generowanie raportów o aktywności wypożyczeń w danym okresie.
   * Statystyki dotyczące liczby książek w poszczególnych kategoriach.
   * Lista uczniów z zaległymi zwrotami.

Aktorzy

1. Administrator:
   * Zarządza wszystkimi aspektami systemu (uczniowie, książki, kategorie, autorzy, wydawcy).
   * Generuje raporty i statystyki.
2. Uczeń:
   * Przegląda dostępne książki.
   * Wypożycza i zwraca książki.

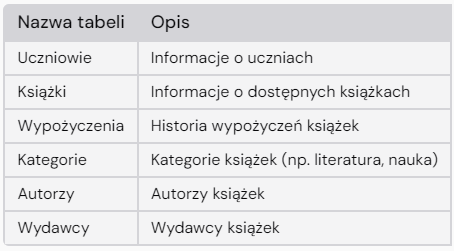
Główne Scenariusze Użycia

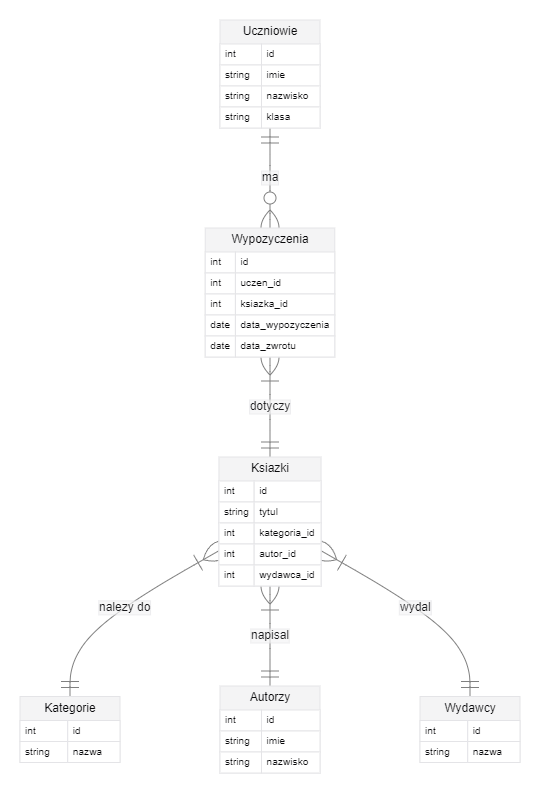
1. Dodanie Nowej Książki:
   * Administrator wybiera opcję dodania nowej książki.
   * Wypełnia formularz (tytuł, kategoria, autor, wydawca).
   * System zapisuje książkę w bazie danych.
2. Wypożyczenie Książki:
   * Uczeń wybiera książkę z listy dostępnych.
   * System przypisuje książkę do ucznia, zapisując datę wypożyczenia.
3. Zwrot Książki:
   * Uczeń zwraca książkę.
   * System aktualizuje rekord wypożyczenia, zapisując datę zwrotu.
4. Generowanie Raportu:
   * Administrator wybiera zakres dat.
   * System generuje raport z aktywności wypożyczeń w danym okresie.

Korzyści

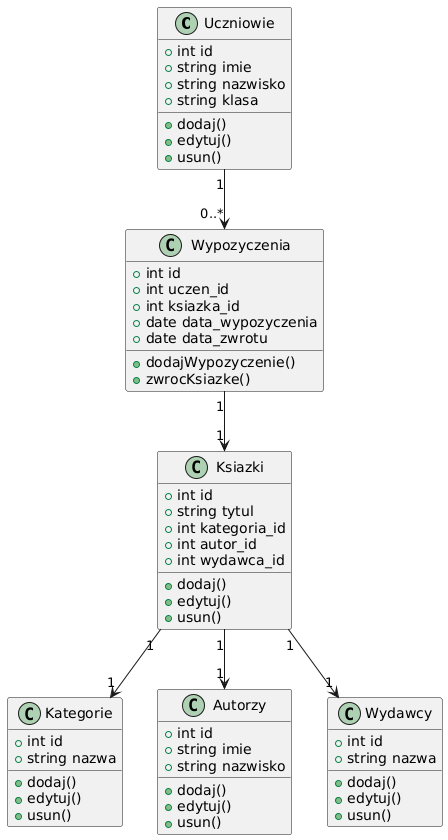
* Usprawnienie procesu wypożyczania książek.
* Łatwe monitorowanie stanu zasobów bibliotecznych.
* Automatyzacja generowania raportów i statystyk.

Dane, które będą wykorzystane w projekcie:



2. Diagram ERD

3.Diagram UML



4.Opis technologii

1. Python

W tym projekcie Python będzie używany do:

- Tworzenia logiki aplikacji: Zarządzanie uczniami, książkami, wypożyczeniami itp.

- Interakcji z bazą danych: Wykonywanie zapytań SQL, aktualizacja danych, pobieranie informacji.

- Tworzenia prostego interfejsu użytkownika: Można użyć bibliotek takich jak `Tkinter` lub `PyQt` do stworzenia aplikacji desktopowej.

2. SQLite

SQLite to lekka, wbudowana baza danych, która nie wymaga oddzielnego serwera. Jest idealna do mniejszych projektów, takich jak system zarządzania biblioteką szkolną. SQLite oferuje:

- Prostą obsługę: Baza danych jest przechowywana w jednym pliku, co ułatwia zarządzanie i przenoszenie projektu.

- Obsługę SQL: Pełna obsługa standardowych zapytań SQL, takich jak `SELECT`, `INSERT`, `UPDATE`, `DELETE`.

- Integrację z Pythonem: SQLite jest domyślnie zintegrowany z Pythonem poprzez moduł `sqlite3`.

3. Moduł `sqlite3` w Pythonie

Moduł `sqlite3` jest wbudowaną biblioteką Pythona, która umożliwia interakcję z bazą danych SQLite. Oto przykładowe operacje:

- Połączenie z bazą danych:

```python

import sqlite3

Połączenie z bazą danych (lub utworzenie nowej, jeśli nie istnieje)

conn = sqlite3.connect('biblioteka.db')

cursor = conn.cursor()

```

- Tworzenie tabel:

```python

cursor.execute('''

CREATE TABLE IF NOT EXISTS Uczniowie (

id INTEGER PRIMARY KEY AUTOINCREMENT,

imie TEXT NOT NULL,

nazwisko TEXT NOT NULL,

klasa TEXT

)

''')

conn.commit()

```

- Dodawanie danych:

```python

cursor.execute('''

INSERT INTO Uczniowie (imie, nazwisko, klasa)

VALUES ('Jan', 'Kowalski', '3A')

''')

conn.commit()

```

- Pobieranie danych:

```python

cursor.execute('SELECT FROM Uczniowie')

uczniowie = cursor.fetchall()

for uczen in uczniowie:

print(uczen)

```

- Zamykanie połączenia:

```python

conn.close()

```

4. Prosty interfejs użytkownika

Do stworzenia prostego interfejsu użytkownika można użyć biblioteki `Tkinter`, która jest wbudowana w Python. Przykładowe użycie:

```python

import tkinter as tk

from tkinter import messagebox

def dodaj\_ucznia():

imie = entry\_imie.get()

nazwisko = entry\_nazwisko.get()

klasa = entry\_klasa.get()

Tutaj można dodać kod do dodania ucznia do bazy danych

messagebox.showinfo("Sukces", f"Dodano ucznia: {imie} {nazwisko}")

Tworzenie okna aplikacji

root = tk.Tk()

root.title("System Biblioteki Szkolnej")

Pola do wprowadzenia danych

tk.Label(root, text="Imię").grid(row=0, column=0)

entry\_imie = tk.Entry(root)

entry\_imie.grid(row=0, column=1)

tk.Label(root, text="Nazwisko").grid(row=1, column=0)

entry\_nazwisko = tk.Entry(root)

entry\_nazwisko.grid(row=1, column=1)

tk.Label(root, text="Klasa").grid(row=2, column=0)

entry\_klasa = tk.Entry(root)

entry\_klasa.grid(row=2, column=1)

Przycisk do dodania ucznia

tk.Button(root, text="Dodaj ucznia", command=dodaj\_ucznia).grid(row=3, column=0, columnspan=2)

Uruchomienie aplikacji

root.mainloop()

```

Podsumowanie technologii:

- Python: Język programowania do tworzenia logiki aplikacji i interfejsu użytkownika.

- SQLite: Lekka, wbudowana baza danych, idealna dla mniejszych projektów.

- Moduł `sqlite3`: Umożliwia łatwą interakcję z bazą danych SQLite w Pythonie.

- Tkinter: Prosta biblioteka do tworzenia interfejsu użytkownika w Pythonie.

5.Przykładowe zapytania

1. Wyszukanie wszystkich książek danego autora

```sql

SELECT Ksiazki.tytul

FROM Ksiazki

JOIN Autorzy ON Ksiazki.autor\_id = Autorzy.id

WHERE Autorzy.nazwisko = 'Sienkiewicz';

```

2. Lista uczniów, którzy wypożyczyli książki w danym miesiącu

```sql

SELECT Uczniowie.imie, Uczniowie.nazwisko

FROM Wypozyczenia

JOIN Uczniowie ON Wypozyczenia.uczen\_id = Uczniowie.id

WHERE MONTH(Wypozyczenia.data\_wypozyczenia) = 10; -- 10 odpowiada październikowi

```

3. Liczba książek w każdej kategorii

```sql

SELECT Kategorie.nazwa, COUNT(Ksiazki.id) AS liczba\_ksiazek

FROM Kategorie

JOIN Ksiazki ON Kategorie.id = Ksiazki.kategoria\_id

GROUP BY Kategorie.nazwa;

```

4. Sprawdzenie, które książki nie zostały zwrócone

```sql

SELECT Ksiazki.tytul, Uczniowie.imie, Uczniowie.nazwisko

FROM Wypozyczenia

JOIN Ksiazki ON Wypozyczenia.ksiazka\_id = Ksiazki.id

JOIN Uczniowie ON Wypozyczenia.uczen\_id = Uczniowie.id

WHERE Wypozyczenia.data\_zwrotu IS NULL;

```

5. Dodanie nowej książki

```sql

INSERT INTO Ksiazki (tytul, kategoria\_id, autor\_id, wydawca\_id)

VALUES ('Pan Tadeusz', 1, 1, 1);

```

6. Aktualizacja danych ucznia

```sql

UPDATE Uczniowie

SET klasa = '4B'

WHERE id = 1;

```

7. Usunięcie książki

```sql

DELETE FROM Ksiazki

WHERE id = 1;

```

8. Pobranie wszystkich wypożyczeń danego ucznia

```sql

SELECT Ksiazki.tytul, Wypozyczenia.data\_wypozyczenia, Wypozyczenia.data\_zwrotu

FROM Wypozyczenia

JOIN Ksiazki ON Wypozyczenia.ksiazka\_id = Ksiazki.id

WHERE Wypozyczenia.uczen\_id = 1;

```

9. Znajdowanie wszystkich książek wydanych przez danego wydawcę

```sql

SELECT Ksiazki.tytul

FROM Ksiazki

JOIN Wydawcy ON Ksiazki.wydawca\_id = Wydawcy.id

WHERE Wydawcy.nazwa = 'Wydawnictwo Literackie';

```

10. Statystyki wypożyczeń w danym roku

```sql

SELECT COUNT() AS liczba\_wypozyczen

FROM Wypozyczenia

WHERE YEAR(data\_wypozyczenia) = 2023;

```

Podsumowanie zapytań:

- SELECT: Pobieranie danych z tabel.

- JOIN: Łączenie tabel na podstawie kluczy obcych.

- GROUP BY: Grupowanie danych, np. po kategoriach.

- INSERT: Dodawanie nowych rekordów do tabel.

- UPDATE: Aktualizowanie istniejących rekordów.

- DELETE: Usuwanie rekordów z tabel.

Czy chcesz, abym rozwinął któreś z zapytań lub dodał więcej przykładów?