

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica  
w Krakowie**



**Wydział Elektrotechniki, Automatyki, Informatyki i  
Inżynierii Biomedycznej**

**Wyszukiwarka  
Książek**

**Autorzy:**

Adrian Jaśkowiec  
Mateusz Mirecki  
Weronika Hłaszek  
Karolina Surówka

# Spis treści

<b>Ogólny opis systemu</b>	<b>3</b>
Cel systemu	3
Efekt biznesowy	3
Kryterium	3
<b>Opis budowy aplikacji:</b>	<b>5</b>
Umieszczenie folderów i plików	5
Klasa WolneLekturyAPI:	6
Metody klasowe/prywatne:	6
Atrybuty publiczne:	7
Klasa DatabaseOperations	8
Obsługa i wykorzystywane technologie	9
Diagramy ERD	10
<b>Analiza i projekt</b>	<b>11</b>
<b>Diagramy sekwencyjne</b>	<b>13</b>
Diagram sekwencji rejestracji	13
Diagram sekwencji logowania	14
Diagram sekwencji dodawania książki i oznaczania jako przeczytanej	15
<b>Diagramy aktywności</b>	<b>16</b>
<b>Lista możliwości (funkcji systemu)</b>	<b>17</b>
Użytkownik zalogowany	17
Użytkownik niezalogowany	17
<b>Słownik pojęć</b>	<b>18</b>
<b>Analiza Ryzyka</b>	<b>19</b>
<b>Architektura systemu</b>	<b>20</b>
<b>Lista narzędzi planowanych do użycia przy realizacji projektu</b>	<b>21</b>
<b>Stos technologiczny</b>	<b>22</b>
Python	22
Flask	22
Vue	22
SQLite	22
Git	23
Bootstrap	23
Javascript	23

<b>Prototyp interfejsu</b>	<b>24</b>
<b>Ogólny diagram przypadków użycia</b>	<b>27</b>
Login	27
Registration	28
Browsing as user	30
Adding book to library	31
Deleting book from library	31
<b>Projekt testów</b>	<b>32</b>
Cele testów funkcjonalnych:	32
Narzędzia wykorzystywane do testowania:	32
<b>Bibliografia:</b>	<b>34</b>

# **Ogólny opis systemu**

## **Cel systemu**

Celem opisywanego przez nas systemu informatycznego jest usprawnienie wyszukiwania i możliwość zapisywania książek udostępnionych na stronie wolnelektury.pl dla użytkownika korzystającego z przeglądarki.

## **Efekt biznesowy**

System ma za zadanie zapewnić sprawne wyszukanie dostępnych książek, i dać możliwość użytkownikowi odznaczania, które już przeczytał, a które dopiero chce. System pozwala również na sortowanie według podanych kategorii jak tytuł, autor czy rodzaj literacki dzieła.

## **Kryterium**

Większa wygoda obsługi ulubionych dzieł dla użytkownika i nowe możliwości przechowywania wybranych tytułów.

Udziałowiec	Opis
Użytkownik końcowy	<p>Typ: udziałowiec aktywny</p> <p>Użytkownik korzysta z aplikacji i ma możliwość kontaktu z twórcami poprzez email aby składać uwagi lub pomysły na dodatkowe funkcje, które mogłyby uatrakcyjnić i przyspieszyć czas wykonywania danych zadań</p>
Zespół projektowy	<p>Typ: udziałowiec aktywny</p> <p>Informatycy mogą mieć inny punkt widzenia niż użytkownicy zatem na prośby użytkowników zmieniają i dopasowują system do konkretnych potrzeb klientów</p>

# Opis budowy aplikacji:

## Umieszczenie folderów i plików

W folderze routes jest plik all\_routes.py, gdzie znajdują się wszystkie endpointy naszej aplikacji

W folderze controllers jest plik all\_controllers.py, gdzie są wszystkie kontrolery

W folderze models są modele danych z których korzystamy w pliku database.py są informacje dotyczące struktury bazy danych, w wolne\_lektury\_api.py jest klasa której metody zwracają nam potrzebne informacje z api

W folderze templates są templatki stron które będą renderowane po stronie użytkownika

Pliki w folderze źródłowym:

\_\_init\_\_.py wymagany plik przez flaska do stworzenia paczki

app.py startuje aplikację i pobiera potrzebne klasy/endpointy do działania aplikacji

cli\_commands.py odpowiada za stworzenie własnych komend do obsługi bazy danych

config.py odpowiada za zdefiniowanie globalnych zmiennych jak np. ścieżka do bazy danych

initialize\_objects.py odpowiada za inicjalizowanie rozszerzeń używanych w aplikacji

README.md - informacje jak obsługiwać projekt

requirements.txt - lista wymaganych rozszerzeń

# Klasa WolneLekturyAPI:

Co to jest za klasa:

Klasa ta służy do pobrania całego dostępnego API z Wolnych Lektur i przechowywania pobranych informacji w odpowiednich słownikach, dostępnych w każdej chwili do odczytu.

## Metody klasowe/prywatne:

- `fetch_api()` - pobiera całe użyteczne api z wolnych lektur. Wykonuje to za pomocą poszczególnych funkcji: `get_books()`, `get_authors()`, `get_epochs()`, `get_genres()`, `get_kinds()`
- `get_books()` - pobiera dostępne API ze strony ["https://wolnelektury.pl/api/books"](https://wolnelektury.pl/api/books) i uzupełnia słownik `books_list` o pobrane książki
- `get_authors()` - pobiera dostępne API ze strony `"https://wolnelektury.pl/api/authors/"` i uzupełnia słownik `authors_list` o pobranych autorów
- `get_epochs()` - pobiera dostępne API ze strony `"https://wolnelektury.pl/api/epochs/"` i uzupełnia słownik `epochs_list` o pobrane epoki
- `get_genres()` - pobiera dostępne API ze strony `"https://wolnelektury.pl/api/genres/"` i uzupełnia słownik `genres_list` o pobrane gatunki
- `get_kinds()` - pobiera dostępne API ze strony `"https://wolnelektury.pl/api/kinds/"` i uzupełnia słownik `kinds_list` o pobrane rodzaje literackie

## Metody publiczne:

- Brak, korzystanie z klasy za pomocą publicznych atrybutów

## Atrybuty publiczne:

- **books\_list** - słownik wszystkich książek pobranych z API Wolnych Lektur. Każda książka zawiera takie pola, jak:
  - "title" - tytuł książki
  - "author" - autor książki
  - "epoch" - epoka z której pochodzi książka
  - "genre" - gatunek książki
  - "kind" - rodzaj literacki książki
  - "simple\_thumb" - link do zdjęcia okładki książki
  - "url" - link do strony książki
  - "id" - unikalne id książki
- **authors\_list** - słownik wszystkich autorów pobranych z API Wolnych Lektur. Każdy autor zawiera takie pola, jak:
  - "name" - imię i nazwisko autora
  - "id" - unikalne id autora
- **epochs\_list** - słownik wszystkich epok pobranych z API Wolnych Lektur. Każda epoka zawiera takie pola, jak:
  - "name" - nazwa epoki
  - "id" - unikalne id epoki
- **genres\_list** - słownik wszystkich gatunków pobranych z API Wolnych Lektur. Każdy gatunek zawiera takie pola, jak:
  - "name" - nazwa gatunku
  - "id" - unikalne id gatunku
- **kinds\_list** - słownik wszystkich rodzajów literackich pobranych z API Wolnych Lektur. Każdy rodzaj literacki zawiera takie pola, jak:
  - "name" - nazwa rodzaju literackiego
  - "id" - unikalne id rodzaju literackiego



## Sposób użytkowania:

Klasa jest inicjalizowana na początku uruchamiania strony internetowej i automatycznie pobiera dane do atrybutów publicznych. Klasa jest globalna i statyczna, czyli jest tylko jedna na cały projekt i dostęp do poszczególnych słowników odbywa się za pomocą: `WolneLekturyApi.[lista]`, np. Aby uzyskać wszystkie książki należy wywołać `WolneLekturyApi.books_list`.

## Klasa DatabaseOperations

Co to jest za klasa:

Klasa ta służy do wykonywania operacji na bazach danych. Podczas metod wykorzystujących hasło, wykonywane jest hashowanie za pomocą klucza prywatnego, dostępnego tylko na serwerze.

Metody publiczne:

- `add_user (username, password)` - Dodaj użytkownika do bazy danych z książkami oraz do bazy danych z logowaniem
- `delete_user (username)` - Usuń użytkownika z bazy danych z książkami oraz z bazy danych z logowaniem
- `update_password (new_password)` - Zmień hasło użytkownika
- `add_book (username, book_id)` - Dodaj książkę do biblioteki
- `remove_book (username , book_id)` - Usuń książkę z biblioteki
- `check_if_username_exists (username)` - sprawdza czy użytkownik o danym loginie istnieje w bazie
- `check_if_password_matches_username (username, password)` - sprawdza czy wpisane hasło jest poprawne

## Sposób użytkowania:

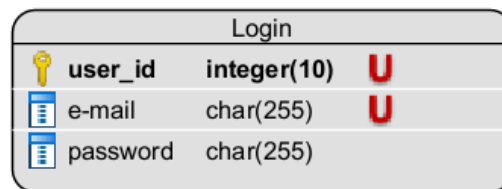
Klasa jest globalna i statyczna, czyli jest tylko jedna na cały projekt i dostęp do zasobów odbywa się poprzez nazwa klasy i metoda (nie są tworzone instancje klasy). Aby użyć np. metody `add_user` należy wywołać ją za pomocą: `DatabaseOperations.add_user(username, password)`. Hasło podawane do funkcji jest jako zwykły tekst, ponieważ metody obsługujące hasło hashują je w sobie przed operacjami na nich.

## Obsługa i wykorzystywane technologie

- Za backend odpowiada framework **flask**
- Za frontend odpowiada **bootstrap**, framework **vue** wraz z rozszerzeniem do vue czyli **vuetify**
- Za bazę danych odpowiada rozszerzenie **SQLAlchemy**, której obiekt podpinamy do instancji flaski
- Za modele bazodanowe odpowiada rozszerzenie **Marshmallow**, którego obiekt możemy zainicjalizować mając obiekt bazy
- Za rozdzielenie endpointów odpowiada rozszerzenie **Blueprint**, które podpinamy do instancji flaski
- Za pomocą funkcji **render\_template** przekazujemy całą stronę do użytkownika
- Do bezwzględnego zdefiniowania ścieżek używamy bibliotek **os** oraz **sys**

# Diagramy ERD

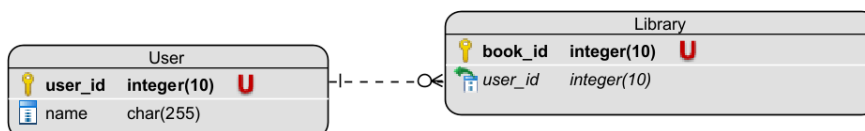
## Schemat bazy danych logowania (ERD)



## Specyfikacja kwerend

Tabela	Opis tabeli	Atrybut	Opis
Login	Tabela zawierająca dane logowania	user_id	Unikalny numer identyfikujący użytkownika
		e-mail	Adres e-mail użytkownika
		password	Zaszyfrowane hasło użytkownika

## Schemat bazy danych (ERD)



## Specyfikacja kwerend

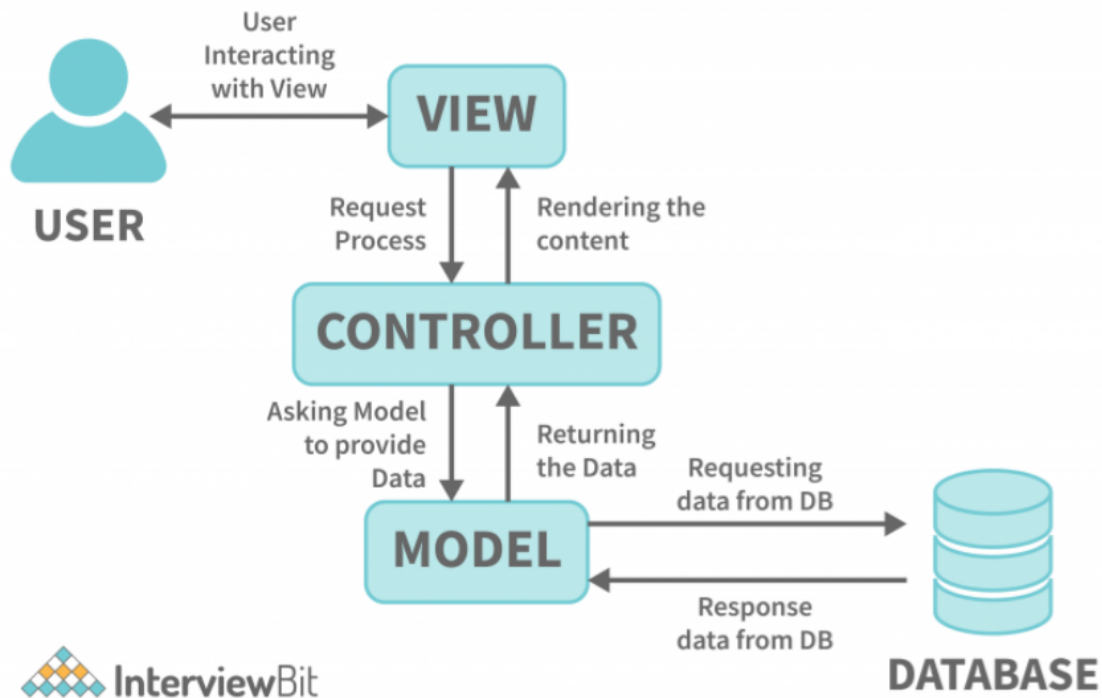
Tabela	Opis tabeli	Atrybut	Opis
User	Tabela zawierająca dane użytkownika	user_id	Unikalny numer identyfikujący użytkownika
		name	Nazwa użytkownika
Library	Tabela połączeń książek z użytkownikiem	book_id	Unikalny numer identyfikujący książkę
		user_id	Numer identyfikujący konto użytkownika

# Analiza i projekt

System zostanie zrealizowany w architekturze trójwarstwowej model-widok-kontroler dostosowanej do potrzeb aplikacji webowych.

Zdecydowaliśmy się na wybór aplikacji webowej, z uwagi na dostęp do aplikacji z dowolnego komputera oraz możliwość późniejszej rozbudowy systemu o elementy interesujące z punktu widzenia użytkownika. Api z którego korzystamy wymaga dostępu do internetu więc aplikacja webowa daje pewność, że użytkownik ma dostęp do internetu.

Z uwagi na fakt, że zdecydowaliśmy się na aplikację webową platforma nie ma większego znaczenia, system przystosowany jest do użytkowania w standardowych przeglądarkach, przy czym zaleca się stosowanie Google Chrome. Ogólny schemat architektury:



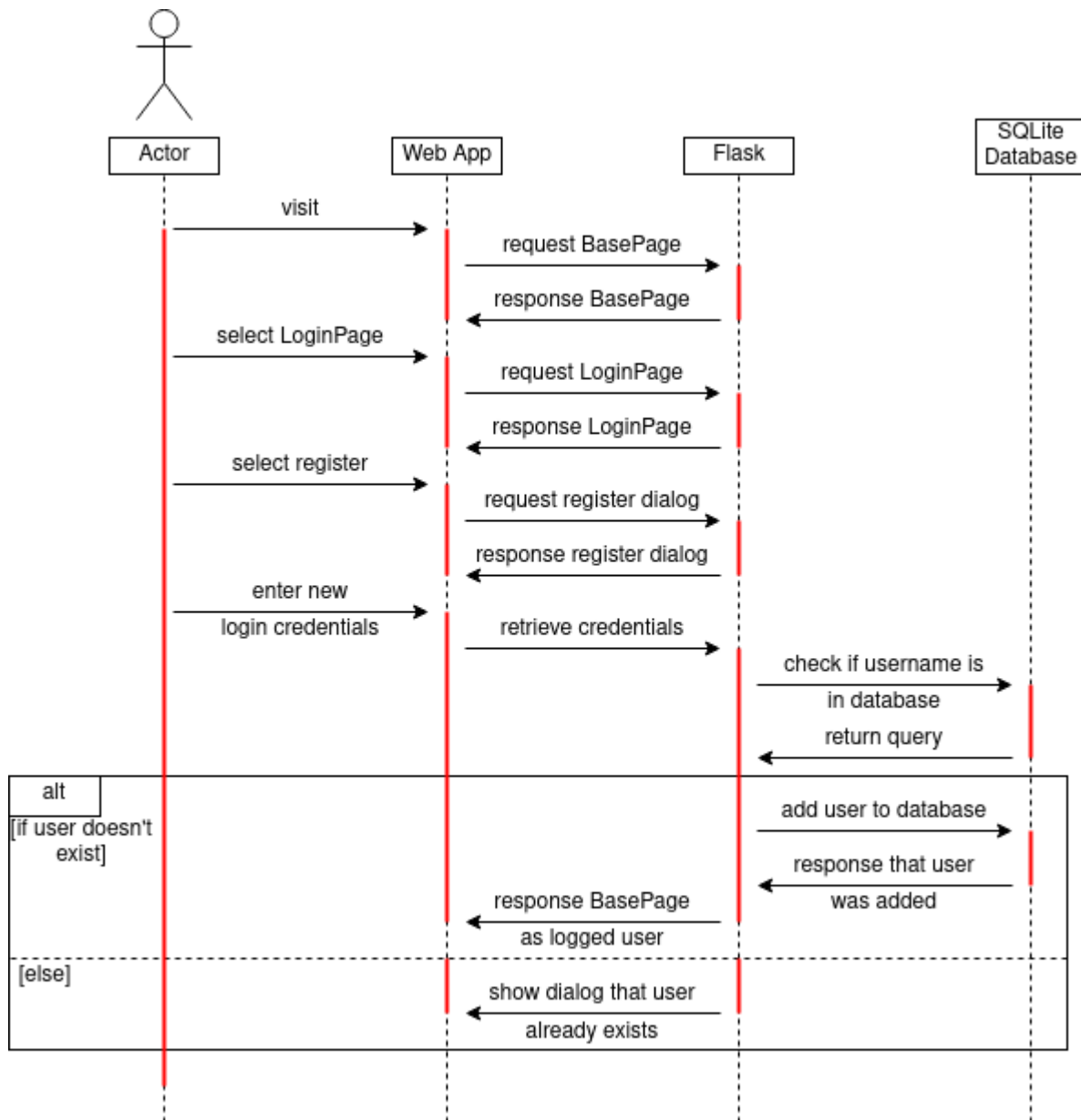
1. **Model** – odpowiada za logikę biznesową. Obejmuje wszystkie obiekty, które służą do wykonywania operacji związanych z implementacją funkcjonalności aplikacji np. obiekty bazodanowe.

2. **Widok** – warstwa prezentacji. Widok odpowiedzialny jest za prezentację użytkownikowi wyników działania logiki biznesowej (Modelu) na witrynie internetowej w przeglądarce użytkownika.

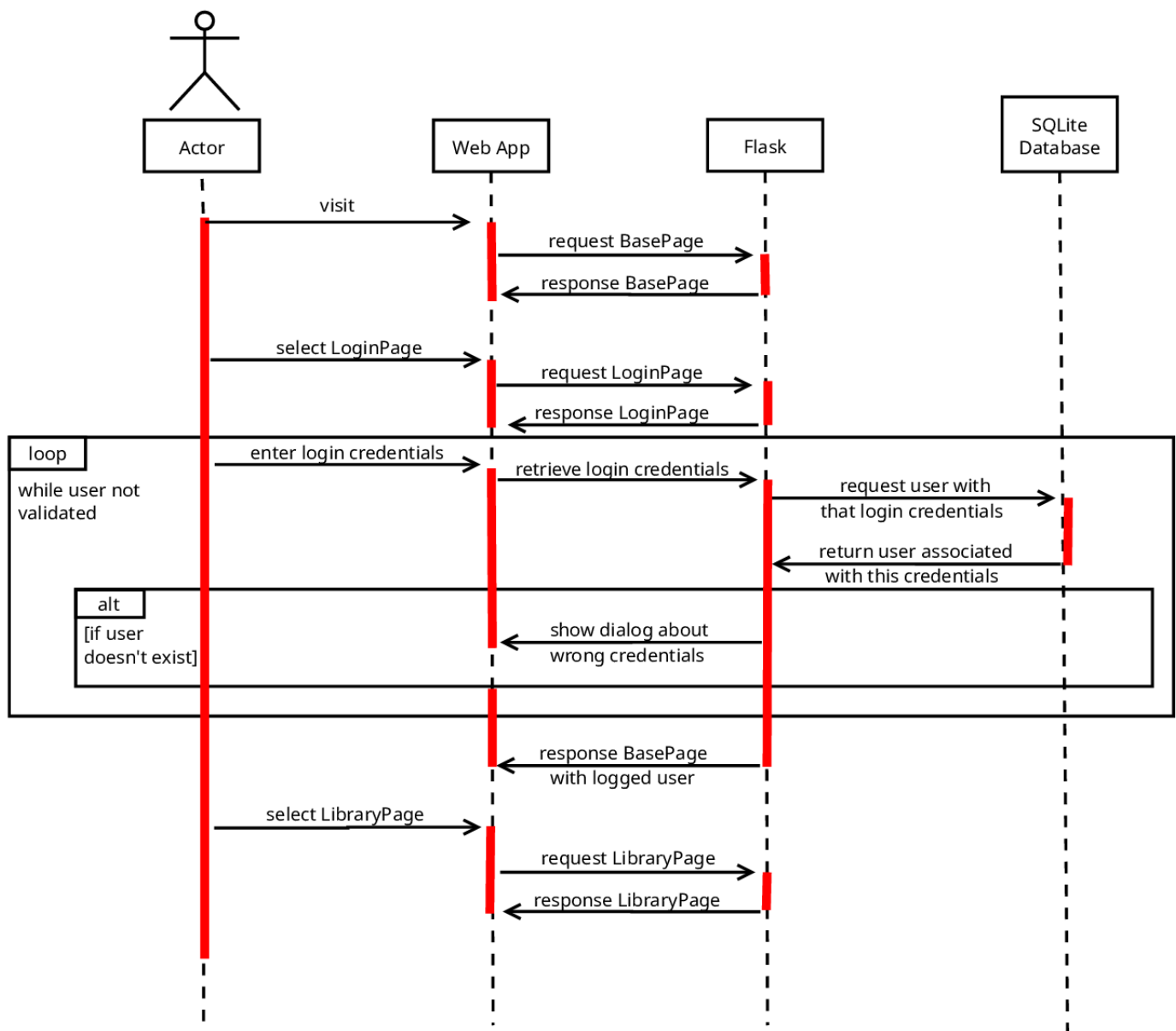
3. **Kontroler** – obsługuje żądania użytkownika. Wszelkie żądania deleguje do odpowiednich metod Modelu, który zwraca inny obiekt, który kontroler może przekazać użytkownikowi w formie graficznej na stronie.

# Diagramy sekwencyjne

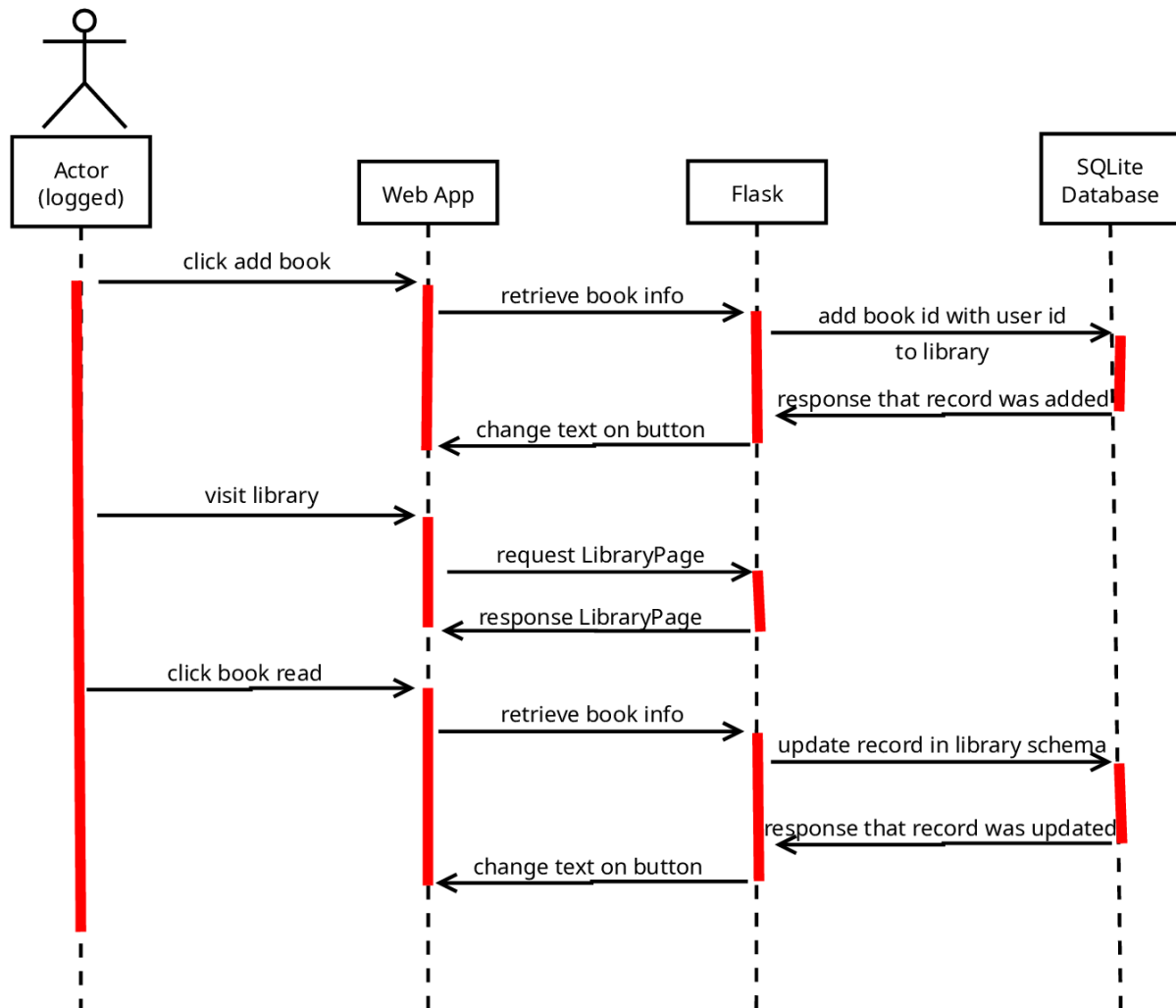
## Diagram sekwencji rejestracji



# Diagram sekwencji logowania

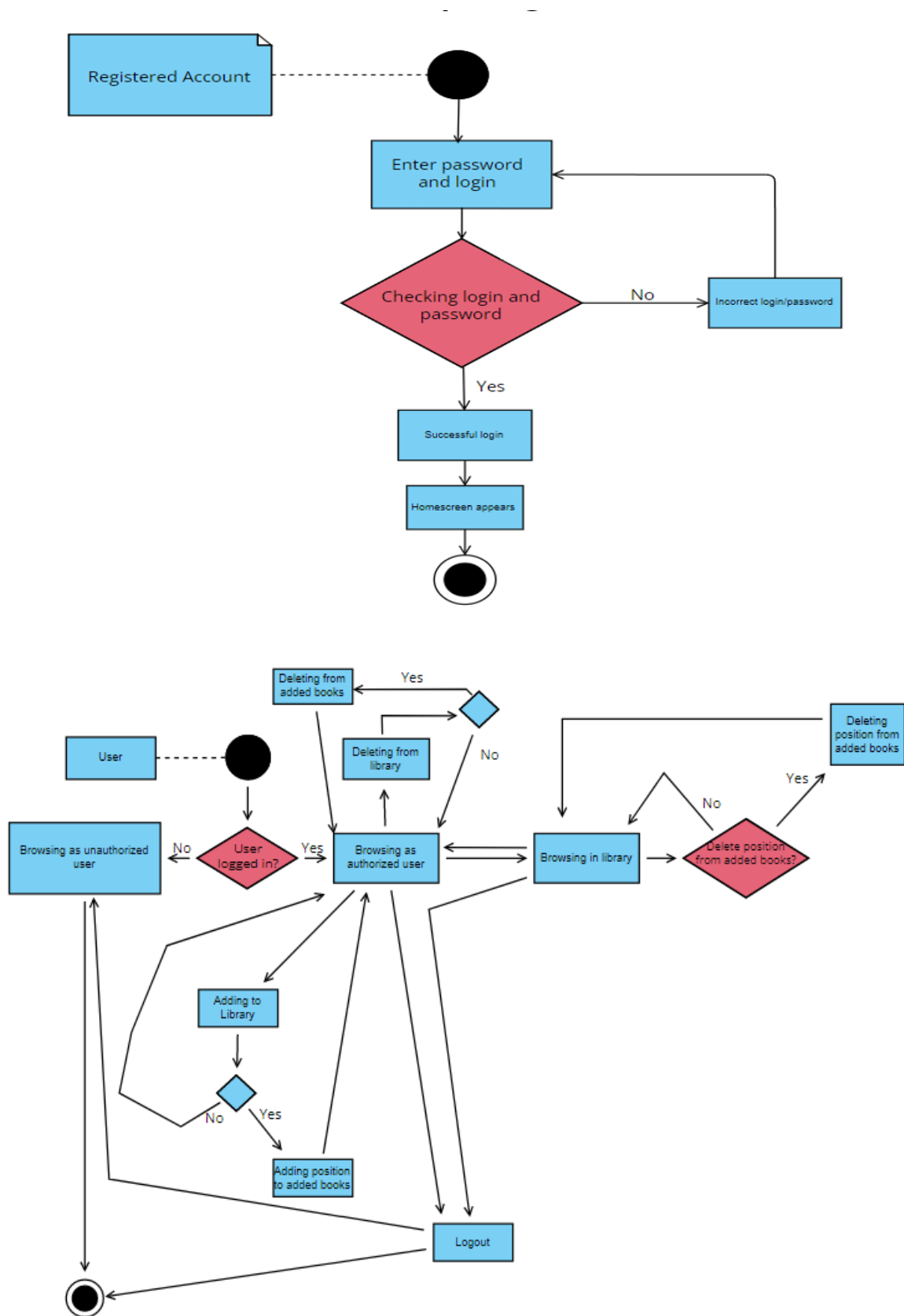


# Diagram sekwencji dodawania książki i oznaczania jako przeczytanej





# Diagramy aktywności



# **Lista możliwości (funkcji systemu)**

## **Użytkownik zalogowany**

- przeglądanie/filtrowanie dostępnych książek z portalu wolnelektury.pl
- dodawanie wybranych książek do własnej biblioteki
- wyszukiwanie/usuwanie w bibliotece dodanych książek

## **Użytkownik niezalogowany**

- przeglądanie/filtrowanie dostępnych książek z portalu wolnelektury.pl

# Słownik pojęć

**System** – aplikacja służąca do wyszukiwania i zapisywania książek dostępnych w serwisie wolnelektury.pl. Jest to system dostępny online, dostęp do Systemu jest możliwy z niezalogowanymi użytkownikami – korzystanie z Systemu wymaga nieuprzedniego zalogowania (tj. podania Loginu i Hasła użytkownika), jednak dzięki logowaniu otrzymujemy dostęp do dodatkowych funkcji (np. zapisywanie)

**Użytkownik** – podstawowy klient Systemu. Lista dostępnych funkcjonalności zależna jest od tego czy jest zalogowany. Użytkownik posiada unikalne Login oraz Hasło umożliwiające zalogowanie się do systemu.

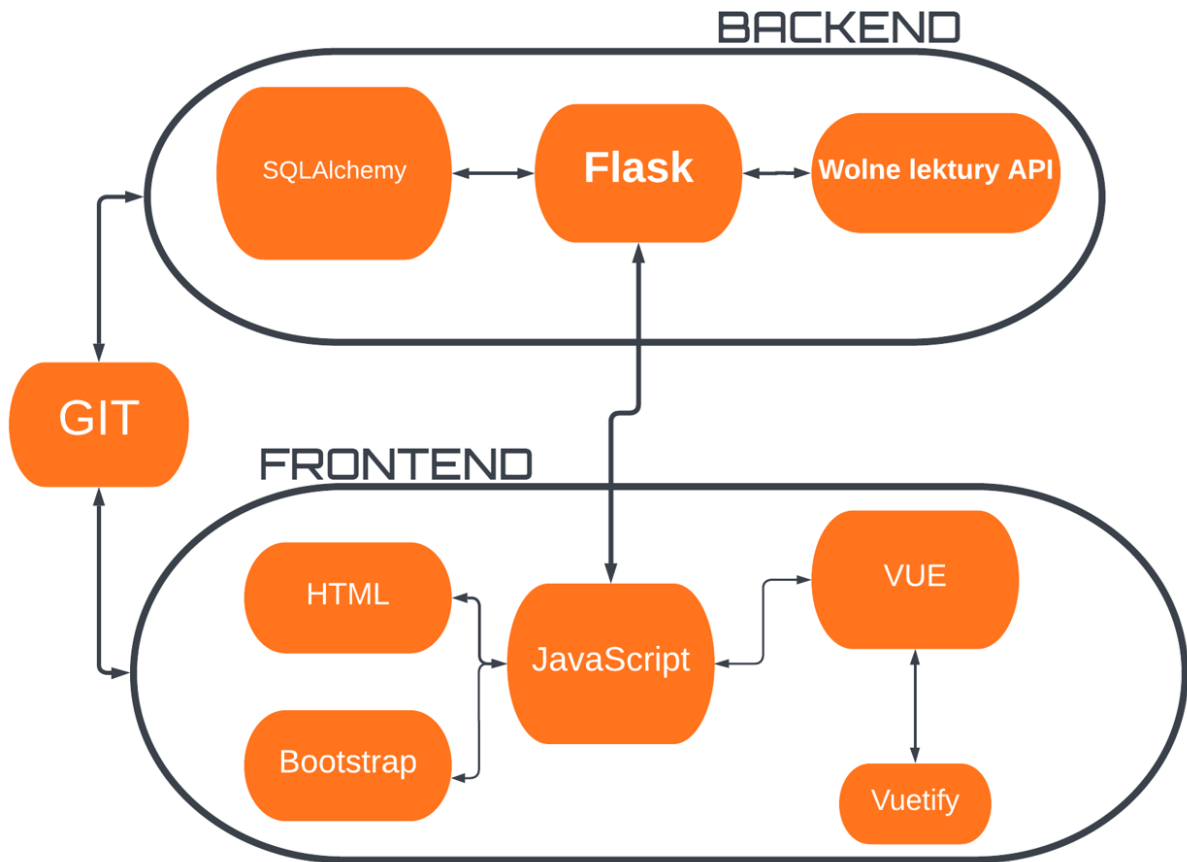
**Rola** – parametr określający zakres uprawnień i dostęp do funkcjonalności Systemu dla danego Użytkownika. W Systemie wyróżniamy dwie role użytkownika zalogowanego oraz niezalogowanego. Szczegółowy zakres funkcjonalności dostępnych dla każdej z ról został opisany powyżej w punkcie Lista możliwości systemu

**Sekcja** - część witryny internetowej odpowiedzialną za konkretną funkcję. Jej rozmiar jest dynamiczny w zależności od ilości danych i szerokości urządzenia z którego korzysta użytkownik

# Analiza Ryzyka

	Risk	Low	Medium	High	How to deal
Technical problems	Problem wewnętrzny zewnętrznego dostawcy	-	-	+	Informacje wyświetlane na stronie o błędzie
	Utrata kodu	+	-	-	korzystanie z autozapisu/ kod synchronizowany z gitem
	Awaria sprzętu	-	+	-	możliwa szybka naprawa/zastąpienie nowym
	Problemy z dostępem do internetu	-	+	-	skorzystanie z sieci publicznej
Problems of human nature	Zmęczenie	-	+	-	analiza swojego trybu życia
	Brak czasu	-	+	-	lepsza organizacja pracy dnia
	Choroba	+	-	-	szybkie wdrożenie leczenia
	Brak motywacji	-	+	-	wsparcie reszty członków grupy
	Brak dostatecznej wiedzy/kompetencji/umiejętności	-	+	-	systematyczne i regularne studiowanie i dążenie problemu, a także wsparcie reszty grupy

# Architektura systemu



# **Lista narzędzi planowanych do użycia przy realizacji projektu**

- Github
- Discord
- Trello
- Google docs
- Visual studio code
- Diagram.net
- Visual paradigm

# Stos technologiczny

## Python

Python został wybrany na główny język projektu ze względu na bardzo dobrą jego znajomość wśród osób w grupie. Także do pythona istnieje dużo bibliotek umożliwiających proste połączenie z innymi technologiami oraz bibliotek ułatwiających wiele etapów prac przy projekcie.

## Flask

Jest to biblioteka do pythona umożliwiająca budowę stron internetowych w architekturze model-view-controller. Flask został wybrany do projektu ze względu na prostą składnię przy tworzeniu stron internetowych oraz zautomatyzowanie wielu żmudnych procesów typowych dla tworzenia stron internetowych.

## Vue

Jest to biblioteka do tworzenia interfejsu użytkownika na stronach internetowych. W porównaniu do innych bibliotek stworzonych w tym celu jak React, czy Angular, Vue wyróżnia się o wiele prostszym i szybszym tworzeniu interfejsu użytkownika. Także umożliwia ona korzystanie z biblioteki Vuetify, która wykorzystuje Vue i umożliwia proste tworzenie interfejsów według zasad Material Design.

## SQLite

SQLite jest to baza relacyjna. Wybrana została przez dobrą znajomość wśród grupy oraz ze względu na bardzo szybką możliwość wdrożenia z innymi wybranymi technologiami.

## **Git**

Git umożliwia zapisywanie kodu na każdym możliwym momencie pisania, przez co zawsze można wrócić do starszego kodu. Dzięki temu kod zawsze ma kopię zapasową i eliminuje to ryzyko utracenia kodu.

## **Bootstrap**

Bootstrap jest biblioteką do tworzenia responsywnego interfejsu użytkownika. Wybrany został do projektu ze względu na to, że oferuje wiele elementów interfejsów od razu gotowych do wstawienia na stronę i zapewnia responsywność tworzonego interfejsu.

## **Javascript**

Javascript został wykorzystany jako język do dynamicznego tworzenia frontendu. W nim wykorzystywane są inne technologie takie jak Bootstrap czy Vue. Także umożliwia dodawanie funkcjonalności do strony takiej jak co robi naciśnięcie danego elementu.



# Prototyp interfejsu

← Wyloguj się

 Biblioteka

## Wyszukiwarka Książek

"Pluń na wszystko, co minęło: na własną boleść i na cudzą nikczemność...  
Wybierz sobie jakiś cel, jakkolwiek i zacznij nowe życie."

Bolesław Prus

Podaj tytuł	Autor	Epoka	Gatunek	Rodzaj
	"Konrad Wallenrod"			
	Adam Mickiewicz			
	"Balladyna"			
	Juliusz Słowacki			
	"Sachem"			
	Henryk Sienkiewicz			
	"Ciemne ścieżki i jasne polany"			
	Andrzej Gronczewski			
	"Katarynka"			
	Bolesław Prus			

← Wyloguj się

 Biblioteka

## Wyszukiwarka Książek

"Pluń na wszystko, co minęło: na własną boleść i na cudzą nikczemność...  
Wybierz sobie jakiś cel, jakkolwiek i zacznij nowe życie."

Bolesław Prus

Podaj tytuł	Autor	Epoka	Gatunek	Rodzaj
	"Konrad Wallenrod"			
	Adam Mickiewicz			
	"Balladyna"			
	Juliusz Słowacki			
	"Sachem"			
	Henryk Sienkiewicz			
	"Ciemne ścieżki i jasne polany"			
	Andrzej Gronczewski			
	"Katarynka"			
	Bolesław Prus			

# Wyszukiwarka Książek

"Pluń na wszystko, co minęło: na własną boleść i na cudzą nikczemność...  
Wybierz sobie jakiś cel, jakkolwiek i zacznij nowe życie."

Bolesław Prus

**"Dziady"**    
Adam Mickiewicz

## Moja Biblioteka

### Dodane

**"Dziady"**  
Adam Mickiewicz

**"Sachem"**  
Henryk Sienkiewicz

### Przeczytane

**"Ciemne ścieżki i jasne polany"**  
Andrzej Gronczewski

**"Katarynka"**  
Bolesław Prus

**"Balladyna"**  
Juliusz Słowacki

## Zaloguj się

Nazwa użytkownika

 Nazwa użytkownika

Hasło


 Hasło

[Zarejestruj się](#)

Zaloguj się

## Zarejestruj się

Nazwa użytkownika

 Podaj nazwę użytkownika

Hasło

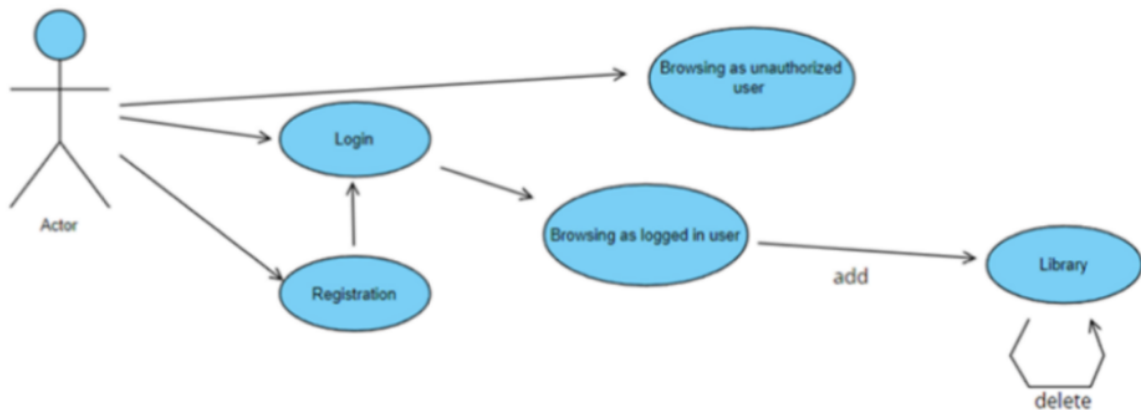
 Podaj hasło

Potwierdzenie hasła

 Potwierdź hasło

Zarejestruj się

# Ogólny diagram przypadków użycia



## Definicje przypadków użycia (dla wybranych akcji)

### Login

Aktorzy: Użytkownik

Krótki opis: Użytkownik chce zalogować się na stronie wyszukiwarki.

Zdarzenie wyzwalające (trigger): Użytkownik wybiera funkcję Zaloguj się

Warunki wstępne: Użytkownik nie może być zalogowany

Warunki końcowe dla sukcesu: Zmiana statusu użytkownika na „zalogowany”

Warunki końcowe dla niepowodzenia: Brak zmian statusu użytkownika, użytkownik zostanie powiadomiony o niepowodzeniu operacji.

Scenariusz główny:

1. Użytkownik klika przycisk **Zaloguj się**
2. System przenosi użytkownika do modułu logowania
2. Użytkownik wprowadza adres e-mail oraz hasło
3. System weryfikuje dane. Weryfikacja zakończona sukcesem
4. Status użytkownika zostaje zmieniony na „zalogowany”. System przenosi użytkownika na stronę główną

Scenariusz alternatywny:

- 3.a System weryfikuje dane. Weryfikacja zakończona niepowodzeniem
- 3.a.1 System wyświetla informację o niepoprawnych danych
- 3.a.2 Użytkownik poprawia błędy w danych
- 3.a.3 System weryfikuje dane. Weryfikacja zakończona sukcesem

## Registration

Aktorzy: Użytkownik

Krótki opis: Użytkownik chce zarejestrować się na stronie wyszukiwarki.

Zdarzenie wyzwalające (trigger): Użytkownik wybiera funkcję Zaloguj się

Warunki wstępne: Użytkownik nie może być zalogowany i nie posiada zarejestrowanego konta

Warunki końcowe dla sukcesu: Dodanie nowego użytkownika do bazy kont

Warunki końcowe dla niepowodzenia: Brak zmian w bazie, użytkownik zostanie poinformowany o niepowodzeniu.

Scenariusz główny:

1. Użytkownik klika przycisk **Zaloguj się**
2. System przenosi użytkownika do modułu logowania
3. Użytkownik klika przycisk **Zarejestruj się** pod oknem logowania
4. System przenosi użytkownika do modułu rejestrowania
5. Użytkownik wprowadza adres e-mail oraz dwukrotnie hasło
6. System weryfikuje dane, łączy się z bazą i zapisuje dane nowego użytkownika
7. Wyświetlana jest informacja o aktywowaniu konta, status użytkownika zostaje zmieniony na „zalogowany”, system przenosi użytkownika na stronę główną

Scenariusz alternatywny:

- 6.a System weryfikuje dane, użytkownik wprowadził adres e-mail z niepoprawną składnią lub adres e-mail już istnieje w bazie
  - 6.a.1 System wyświetla informacje o niepoprawnych danych
  - 6.a.2 Użytkownik poprawia ewentualne błędy w adresie e-mail
  - 6.a.3 System weryfikuje dane, łączy się z bazą i zapisuje dane nowego użytkownika

# Browsing as user

Aktorzy: Użytkownik

Krótki opis: Użytkownik chce wyszukać książkę

Zdarzenie wyzwajające (trigger): Użytkownik dane w pole wyszukiwania.

Warunki wstępne: Użytkownik znajduje się na stronie głównej wyszukiwarki.

Warunki końcowe dla sukcesu: Wyświetlenie wyszukiwanej pozycji

Warunki końcowe dla niepowodzenia: Brak wyszukiwanej pozycji

Scenariusz główny:

1. Użytkownik wpisuje tytuł książki/wybiera autora/wybiera epokę/wybiera gatunek/wybiera rodzaj
2. System wyświetla użytkownikowi pozycje spełniające wybrane kryteria przy wyszukiwaniu

Scenariusz alternatywny:

2.a System nie odnalazł pozycji spełniających kryteria użytkownika

2.a.1 Użytkownik poprawia wprowadzone dane i wybrane kryteria

2.a.2 System wyświetla użytkownikowi pozycje spełniające wybrane kryteria przy wyszukiwaniu

## Adding book to library

Aktorzy: Użytkownik

Krótki opis: Użytkownik chce dodać książkę do swojej biblioteki.

Zdarzenie wyzwajające (trigger): Użytkownik klika przycisk Dodaj.

Warunki wstępne: Użytkownik musi być zalogowany i znajdować się na stronie głównej z wyświetlanymi pozycjami.

Warunki końcowe dla sukcesu: Dodanie książki do sekcji Dodane w bibliotece użytkownika

Scenariusz główny:

1. Użytkownik klika przycisk **Dodaj** znajdujący się przy wybranej książce
2. System dodaje książkę do sekcji Dodane w bibliotece użytkownika

## Deleting book from library

Aktorzy: Użytkownik

Krótki opis: Użytkownik chce usunąć książkę ze swojej biblioteki

Zdarzenie wyzwajające (trigger): Użytkownik klika przycisk Usuń

Warunki wstępne: Użytkownik musi być zalogowany, znajduje się na stronie głównej z wyświetlanymi pozycjami lub znajduje się w swojej bibliotece

Warunki końcowe dla sukcesu: Usunięcie książki z sekcji Dodane z biblioteki użytkownika

Scenariusz główny:

1. Użytkownik klika przycisk **Usuń** znajdujący się przy wybranej książce
2. System usuwa książkę z sekcji Dodane z biblioteki użytkownika.



# Projekt testów

## Cele testów funkcjonalnych:

- Prawidłowość działania danej funkcji
- Interfejs użytkownika
- Interfejs API
- Kompletność wymagań funkcjonalnych i niefunkcjonalnych
- Zabezpieczenia

## Narzędzia wykorzystywane do testowania:

**Unittest** - framework, za pomocą którego utworzymy testy jednostkowe dotyczące poszczególnych funkcji kontrolerów - BACKEND

Testów wydajnościowych nie będzie jako, że wydajność zależy od zewnętrznych dostawców, czyli od hostingu oraz organizacji udostępniającej api. Od użytkownika za to nie wymaga się większych mocy obliczeniowych niż tych niezbędnych do korzystania z przeglądarki.

## Testowane funkcje:

- Prawidłowe Logowanie, Rejestracja
  - `add_user (username, password)`
  - `delete_user (username)`
  - `update_password (new_password)`
  - `check_if_email_exists (email)`
  - `check_if_password_matches_username (username, password)`
- Testowanie bazy danych
- Testowanie DatabaseOperations
- Poprawne sortowanie i wyświetlanie książek
  - `get_books()`
  - `get_authors()`
  - `get_epochs()`
  - `get_genres()`
  - `get_kinds()`
- Poprawne działanie przycisków: 'Dodaj', 'Usuń' - (na Stronie Głównej jak i w Bibliotece)
  - `add_book (username, book_id)`
  - `remove_book (username , book_id)`

# Bibliografia:

- <https://www.interviewbit.com/blog/mvc-architecture/>