******

***Bazy Danych***

***Inżynieria Oprogramowania***

***Dokumentacja projektu***

***AUTORZY:***

|  |  |
| --- | --- |
| *Adrian*  *Amadeusz*  *Kamil*  *Łukasz*  *Łukasz Łukasz Bartosz Kamil* | *Łysakowski Łakoma Łopuszański Łokaj Malec*  *Marcola*  *Magryś Przybyło* |

*L05 2EF-DI*

# **Spis treści**

1. [***Opis projektu i cel projektu*** 3](#_Toc454202424)

2. [***Użytkownicy*** 3](#_Toc454202425)

3. [***Diagram ERD*** 5](#_Toc454202426)

4. [***Diagram przypadków użycia***](#_Toc454202427) 8

5. [***Opis wybranego środowiska programistycznego***](#_Toc454202427) 9

6. [***Opis bazy danych*** 1](#_Toc454202428)3

7. [***Opis skryptów*** 1](#_Toc454202429)6

8. [***Interfejs*** 1](#_Toc454202430)8

9. [***Podsumowanie***](#_Toc454202431) 20

***9.***[***Załączone diagramy sekwencji***](#_Toc454202431) 21

1. **Opis projektu i cel projektu**

Celem projektu jest opracowanie systemu informatycznego wspierającego proces organizacji konferencji naukowych, kongresów, sympozjów, szkoleń i innych wydarzeń wymagających wcześniejszej rejestracji gości. W zakresie: rejestracji użytkowników, recenzji artykułów naukowych, uczestnictwa w konferencji, zakwaterowania.

1. **Użytkownicy**

W systemie możemy wyróżnić następujących aktorów:  
1. ADMINISTRATOR SYSTEMU/ORGANIZATOR  
2. RECENZENT  
3. UCZESTNIK

Uprawnienia i zadania poszczególnych aktorów w systemie:

* ADMINISTRATOR SYSTEMU:  
  - dodawanie, wyświetlanie i edytowanie kont użytkowników   
  - dodawanie, wyświetlanie i edytowanie konferencji.

Dane podstawowe przy tworzeniu konferencji: temat konferencji, miejsce i data konferencji, liczba wystąpień, długość trwania, maksymalna liczba uczestników, opłata za udział, możliwe opcje zakwaterowania uczestnika.

Po stworzeniu konferencji wyświetlana jest jeszcze informacja o liczbie dostępnych miejsc. Jeśli konferencja jest płatna, to przy jej zapisie jest obowiązek płatności poprzez przekierowanie do PayU.

- zatwierdzanie artykułów/referatów do konferencji, dodanych przez uczestnika  
- przypisywanie recenzentów (minimum 3) do wybranych konferencji  
- generowanie raportów o kalendarzu konferencji, liczbie uczestników, oraz wpływach

* UCZESTNIK:

- zakładanie konta użytkownika.

Dane podstawowe przy założeniu konta: Imie, Nazwisko, e-mail  
- przeglądanie dostępnych konferencji, wraz z danymi na jej temat (dane podstawowe o konferencji + nazwiska autorów i nazwiska recenzentów, kolejność wystąpień)  
- możliwość zapisania się na każdą z konferencji

System generuje maila z potwierdzeniem zapisu na konferencje, oraz dodaje uczestnika do listy uczestników danej konferencji z oznaczeniem wybranej opcji zakwaterowania.

- możliwość zgłoszenia artykułu na konferencje. Zgłoszenie artykułu będzie składało się z 3 części:

- pole tekstowe – podanie tematu konferencji

- miejsce na wrzucenie pliku – skrócony opis poruszanego tematu przez autora, który będzie widoczny dla wszystkich użytkowników

- miejsce na wrzucenie pliku – pełny referat/artykuł, który będzie widoczny wyłącznie dla organizatora oraz wyznaczonych przez organizatora recenzentów

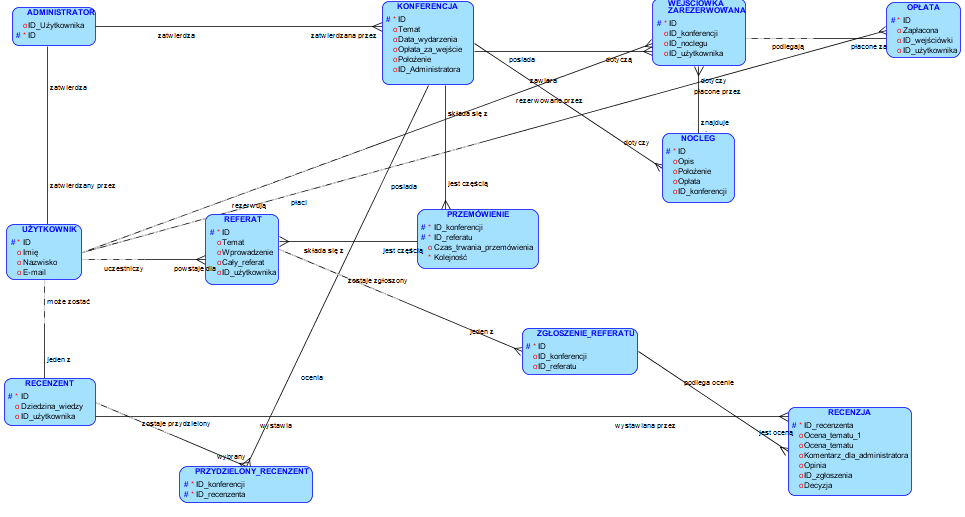
* RECENZENTA  
  - Zakładanie konta recenzenta. Dane podstawowe przy założeniu konta: Imie, Nazwisko, e-mail, obszar w którym recenzent się specjalizuje

Możliwość oceny zgłoszonego artykułu do wybranej konferencji. Ocena składająca się z następujących pól:

- pole tekstowe – słowna ocena artykułu widoczna przez autora i organizatora  
- pole tekstowe – komentarz dla organizatora – niewidoczny dla autora  
- lista rozwijana ze skalą 1-5 oceniająca zgodność artykułu z tematem konferencji  
- lista rozwijana ze skalą 1-5 oceniająca wartość merytoryczną artykułu  
- lista rozwijana z trzema opcjami (przyjąć, zmodyfikować, odrzucić) – jako werdykt recenzenta

Po ocenie recenzenta, autor artykułu otrzymuję mailowo informacje o nowej recenzji to jego artykułu

1. **Diagram ERD**



Rysunek 4.1 Diagram ERD wymodelowany za pomocą programu datamodeler

Zależności między encjami:

**Zależności jeden do wielu:**

-Administrator-Konferencja(Administrator musi zatwierdzić konferencje, konferencja musi zostać zatwierdzona

-Użytkownik-Referat(Użytkownik wysłuchuje referatów, referat musi mieć użytkników chętnych do wysłuchania)

-Recenzent-Przydzielony\_Recenzent(Recenzent może zostać przydzielony do jakiejś konferencji, Recenzent przydzielony musi zostać wybrany z puli recenzentów)

-Zgłoszenie\_Referatu-Recenzja(Każdy zgłoszony referat musi zostać oceniony)

-Konferecja-Przydzielony\_Recenzent(Każda konferencja musi posiadać przydzielonych recenzentów)

-Referat-Zgłoszenie\_Referatu(Każdy referat musi zostać zgłoszony)

-Konferencja-Przemówienie(Każda konferencja składa się z przemówień)

-Konferencja-Nocleg(Do każdej konferencji muszą być przypisane noclegi)

-Nocleg-Wejściówka\_Zarezerwowana(Do każdej wejściówki zarezerwowanej przypisane są noclegi)

-Użytkownik-Wejściówka\_Zarezerwowana(Użytkownik może zarezerwować wejściówkę)

-Konferencja-Wejściówka\_Zarezerwowana(Konferencja musi zawierać wejściówki)

- Zgłoszenie\_Referatu-Recenzje(Każdy zgłoszony referat musi zostać oceniony)

-Recenzent-Recenzja(Każdy recenzent musi pisać recenzje)

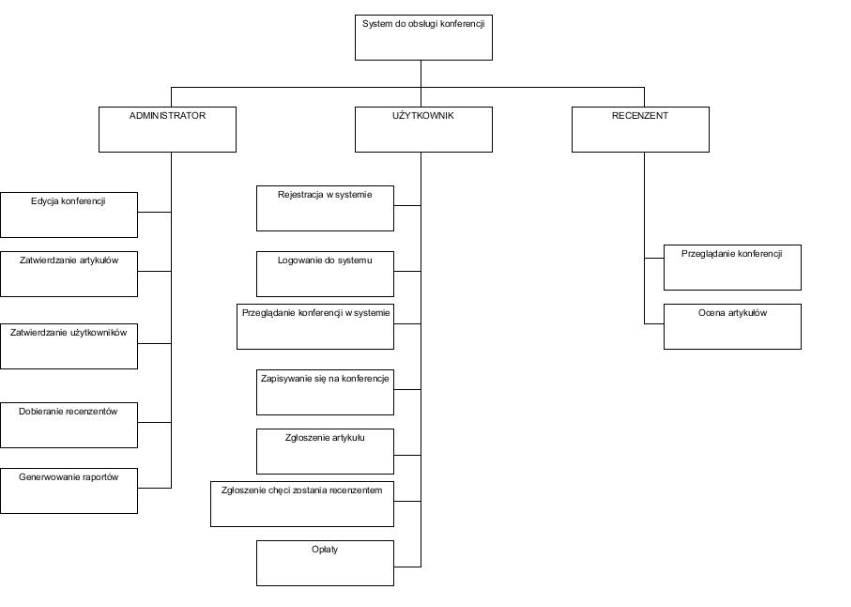
-Wejściówka\_Zarezerwowana-Opłata(Za każdą wejściówkę trzeba zaplacić)

-Referat-Przemówienie(Każde przemówienie składa się z kilku referatów)

**Zależności jeden do jednego:**

-Administrator-Użytkownik(Użytkownik musi zostać zatwierdzony przez administratora)

-Użytkownik-Recenzent(Użytkownik może zostać recenzentem)



Rysunek 4.2 Diagram hierarchii funkcji.

*System do obsługi konferencji*

1.Administrator

1.1Edycja konferencji

1.2Zatwierdzanie artykułów

1.3Zatwierdzanie użytkowników

1.4Dobieranie recenzentów

1.5Generowanie raportów

2. Użytkownik

2.1 Rejestracja w systemie

2.2 Logowanie do systemu

2.3 Przeglądanie konferencji w systemie

2.4 Zapisywanie się na konferencje

2.5 Zgłoszenie artykułu

2.6 Zgłoszenie chęci zostania recenzentem

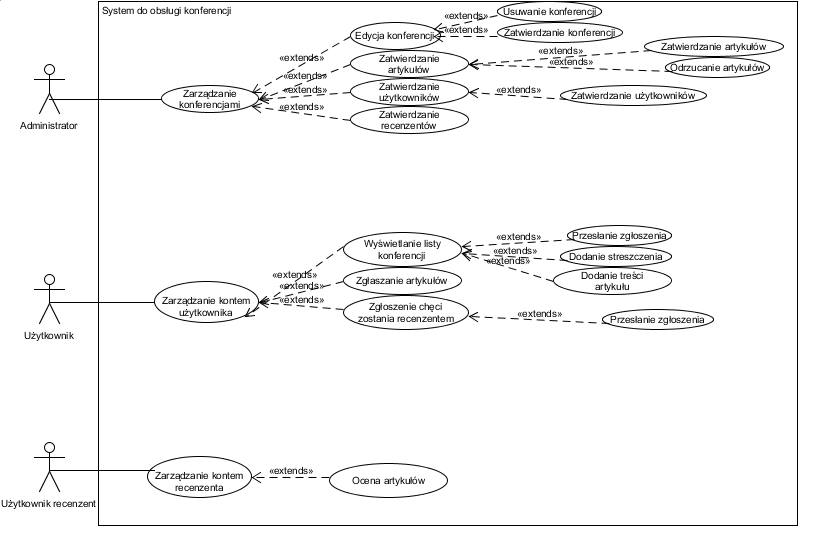
2.7 Opłaty

3. Recenzent

3.1 Przeglądanie konferencji

3.2 Ocena artykułów

1. **Diagram przypadków użycia**



**Diagram przypadków użycia** jest graficzne przedstawienie [przypadków użycia](https://pl.wikipedia.org/wiki/Przypadek_u%C5%BCycia), aktorów oraz związków między nimi, występujących w danej dziedzinie przedmiotowej. Diagram przypadków użycia w języku [UML](https://pl.wikipedia.org/wiki/Unified_Modeling_Language) służy do modelowania funkcjonalności systemu. Powyższy diagram powstał na podstawie zaprojektowanej już aplikacji oraz przedstawia jej funkcjonalność, został on wyrysowany przy pomocy programu UMLet, który używany jest również na laboratorium.

1. **Opis wybranego środowiska programistycznego**

### *Wykorzystywane technologie*

Warstwę aplikacji strony internetowej napisano w języku **Javascript**, wykorzystując runtime **node.js** w wersji 5.10.1 oraz menadżera pakietów (*package manager*) **npm** w wersji 3.8.3. Technologia ta jest oparta na silniku Chrome V8 i pozwala na szybkie tworzenie zarówno backendu jak i frontendu. Strona została wdrożona na platformie **heroku**.**com**. Strony nie są zapisywane w HTMLu, lecz jako szablony napisane w języku **jade**. Pozwala on na wykorzystanie elementów języka strukturalnego, takich jak if-else, pętle foreach, a nawet tworzenie zmiennych i klas wykorzystując język Javascript. Mimo tego, ma on postać języka znacznikowego.

W projekcie wykorzystano wiele pakietów (modułów), między innymi:

* *pg* – umożliwia nawiązanie połączenia i wykonywanie operacji na bazie danych PostgreSQL,
* *app = express()* - implementacja aplikacji, umożliwia obsługę requestów klientów oraz renderowanie stron napisanych w różnych językach szablonowych,
* *passport*– dzięki temu modułowi możliwe są autentykacja i logowanie, tworzenie cookies,

### *Wdrożenie na platformę*

Wdrożenie projektu na platformę polegało na wpisaniu 3 komend w oknie terminala (*PowerShell, Command Line*):

git add .

git commit -m "demo"

[master 9676450] demo

5 files changed, 56 insertions(+), 41 deletions(-)

git push heroku master

Counting objects: 10, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (10/10), done.

Na chwilę pisania tej dokumentacji, projekt na platformie jest w wersji 76.

### *Tworzenie dynamicznych stron WWW*

Niektóre ze stron są tworzone w sposób statyczny, tj. nie zależą od stanu bazy danych, a inne w sposób dynamiczny. Dynamika tych stron jest zrealizowana w następujący sposób (zostanie to pokazane na przykładzie przeglądu konferencji). Kiedy użytkownik klika link [Przegląd konferencji](https://system-conference-online.herokuapp.com/przeglad), zostaje on przekierowany na stronę */przeglad.* Funkcja obsługująca żądanie GET wygląda następująco:

**index.js**

*app*.get('/przeglad', function(req, res) {

pg.connect(*connectionString*, function(err, client, done) {

client.query('SELECT \* FROM conferences', function(err, result) {

done();

res.render('przeglad', {conferenceList: result.rows} );

});

});

});

Jak widać, połączenie z bazą zostaje nawiązane. Następnie, wysłane zostaje zapytanie SELECT, które zwraca wszystkie konferencje znajdujące się w bazie. Rezultat jest obiektem, który jest parametrem funkcji res**.**render. JSON zawierający obiekt reprezentujący konferencje, jest przesyłany do strony napisanej w języku szablonowym jade. Tam wykonuje się następująca pętla:

**przeglad.jade**

ul

each *conf* in conferenceList

li = *conf*.topic

Dla każdego elementu tablicy conferenceList, tworzony jest punkt listy nieuporządkowanej (ul – *unordered list*). Wynik może wyglądać następująco (zależy on od informacji przechowywanej w bazie danych):

Systemy ITS w komunikacji miejskiej

* Wybrane rozwiazania dla technologii Smart Grid
* Optymalizacja Data Center - trendy i wyzwania
* Smart City/ITS
* Zasady tworzenia aplikacji dla MSP
* Zdalne sterowanie obiektów

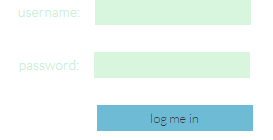
Podobnie stworzone są również następujące strony:

* */editconference –* dostęp tylko dla administratora, pokazuje wszystkie konferencje, pozwalając na usuwanie bądź zatwierdzanie
* */admin/paperlist –* dostęp tylko dla administratora, pokazuje wszystkie artykuły, pozwalając na usuwanie bądź zatwierdzanie
* */db –* pokazuje wszystkich użytkowników

## 

## *Logowanie*

Na powyższym rysunku przedstawiony jest panel logowania. Kiedy dane są wpisane przez użytkownika i klika on na przycisk **log me in**, request POST jest kierowany na tą samą stronę (*/login*), gdzie jest przyjmowany:

[](https://system-conference-online.herokuapp.com/login)Rysunek 6.1: Logowanie

***index.js***

*app*.post('/login', passport.authenticate('local')

passport.use(new *passportLocal*.Strategy(function(username, password, passportDone) {

pg.connect(*connectionString*, function(err, client, done) {

client.query('SELECT \* FROM users WHERE first\_name= $1 AND password\_enc=$2',

[username, password],

function(err, result) {

done();

if(result.rows.length > 0)

passportDone(null, {id: result.rows[0].id});

else

passportDone(null, false, {message: 'Something wrong.'});

});

});

Jak widać w powyższym kodzie, autentykacja odbywa się w module passport. Do bazy danych kierowane jest zapytanie SELECT, czy istnieje użytkownik o danej nazwie i haśle. Jeśli tak, to w sesji przechowywana jest informacja o jego id. Należało również napisać funkcje serializujące i deserializujące obiekt *user*. Po tym, jak użytkownik poprawnie się zaloguje, jest przekierowany na stronę opdowiedniego menu (w zależności od tego, czy jest to administrator czy recenzent), bądź na stronę główną.

Wylogowanie polega na wejściu na stronę */logout*, gdzie cookie *connect.sid* jest usuwane, a użytkownik jest wylogowywany. Operacja kończy się przekierowaniem na stronę główną.

### *Kontrola dostępu*

Zostały napisane funkcje, które uniemożliwiają wejście na strony, które nie są przeznaczone dla użytkownika niezalogowanego bądź są przeznaczone tylko dla użytkownika posiadającego odpowiednie uprawnienia. Są to funkcje: ensureAuthenticated, ensureAuthenticatedReviewer, ensureAuthenticatedAdmin. Najprostsza z nich – ensureAuthenticated – sprawdza tylko, czy użytkownik jest zalogowany:

function *ensureAuthenticated*(req, res, next) {

if(req.isAuthenticated())

return next();

else

res.sendStatus(403); //forbidden

}

Jeżeli tak, to wykona się następny callback, np. renderowanie odpowiedniej strony. W przeciwnym razie strona jest zakazana – wyświetli się odpowiedni komunikat. Użyto funkcji *sendStatus(code)*, ponieważ funkcja *send(code)* jest przestarzała.

W przypadku pozostałych 2 funkcji, sprawdza się id użytkownika, dokładniej: czy znajduje się on w odpowiedniej tabeli bazy danych (*administrators* lub *reviewers*).

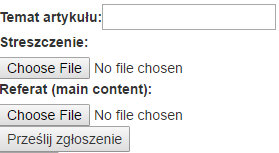
### *Rejestracja*

Rejestracja odbywa się za pośrednictwem formularza, na którym nowy użytkownik wpisuje swoje dane. Następnie, dane są insertowane do bazy danych w poniższej funkcji:

*//add a new user to the database*

*app*.post('/register'

### *Inne funkcjonalności*



Rysunek 6.2 – formularz do rejestracji artykułów

Rejestrowanie artykułów i recenzentów, a także ocena artykułu odbywa się za pośrednictwem formularzy. Przykładowy formularz jest pokazany na rys. 6.2. Po kliknięciu na przycisk *Prześlij zgłoszenie*, dane zostają zapisane w bazie danych:

//add paper to the database

*app*.post('/paper/register', function(req, res) {

pg.connect(*connectionString*, function(err, client, done) {

client.query('INSERT INTO papers (topic, introdution, main\_content,user\_id, id) VALUES ($1,$2,$3,10,$4)',

[req.body.paperTopic, req.body.paperIntro, req.body.paperMainContent,randomID],

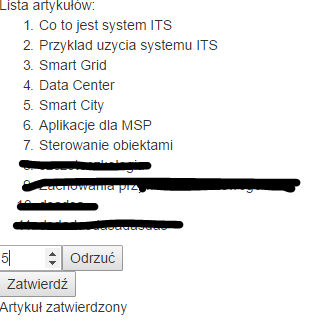
Ponieważ zapytanie używa literałów ($1, $2 …), to atak hakerski poprzez *SQL injection* jest niemożliwy. O bezpieczeństwo dba moduł pg (postgreSQL).

Administrator ma możliwość zatwierdzania i usuwania artykułów i konferencji, zob. rys. 6.3. Odrzucanie jest zrealizowane jako usunięcie danego wpisu z bazy danych:

client.query('DELETE FROM conferences WHERE id=$1',

[conferenceDAO.id],

function (err, result) {

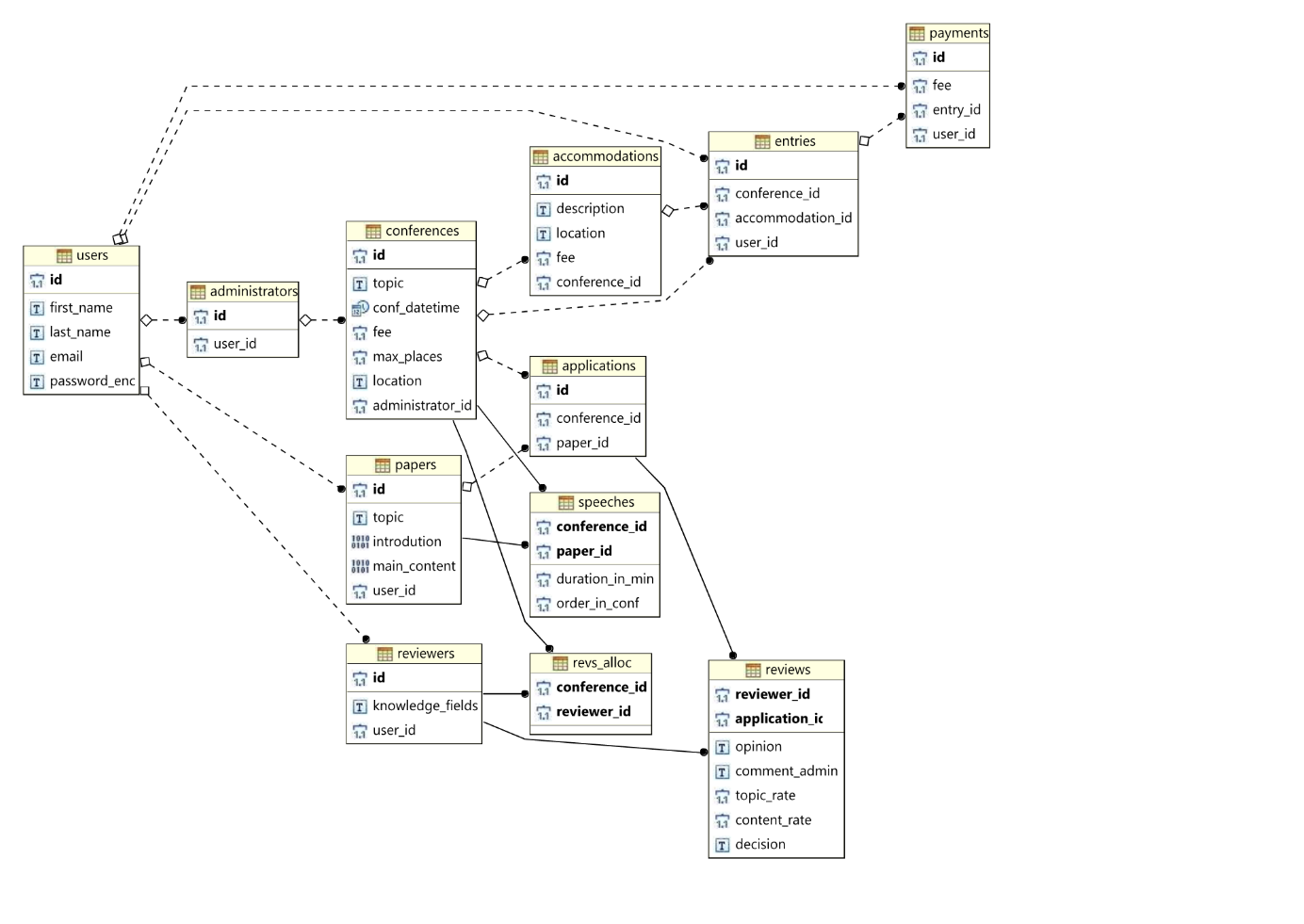


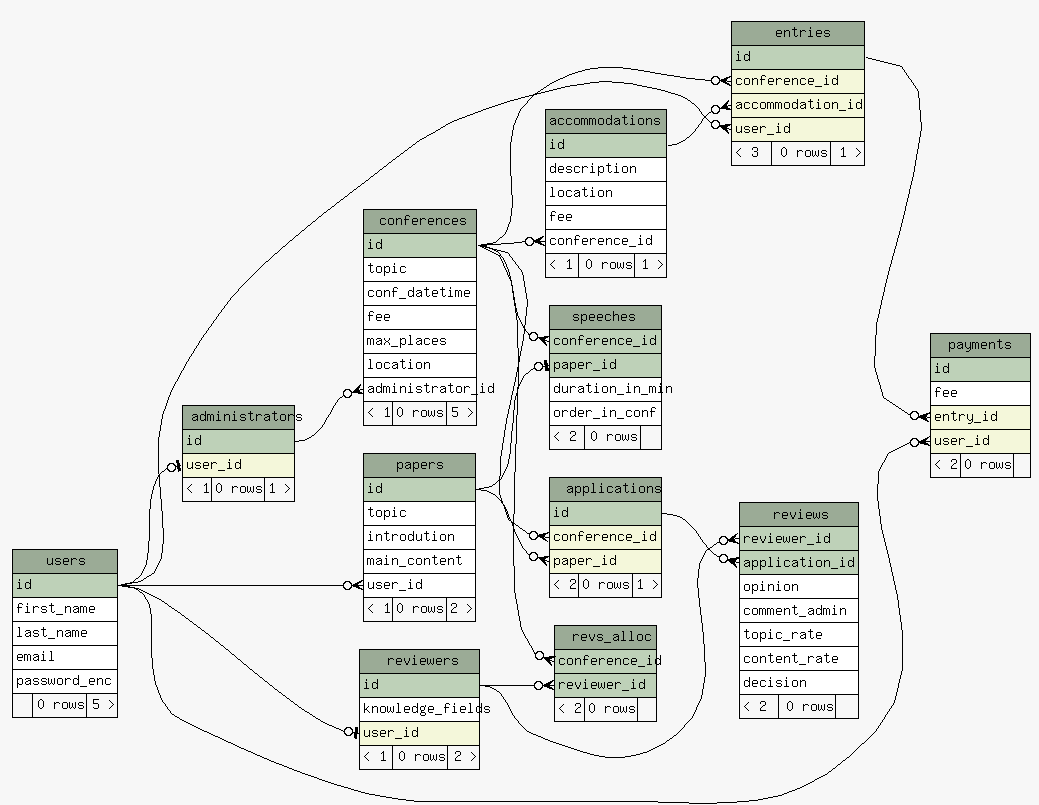
Rysunek 6.3 – zatwierdzenie I odrzucenie artykułów

1. **Opis bazy danych**

Stworzona baza danych zawiera następujące tabele:

* **users** – tabela przechowuje dane użytkowników – imie, nazwisko, email oraz hasło
* **administrators** – tabela przechowuje ID użytkowników, którzy są administratorami aplikacji
* **conferences** – przechowuje informacej na temat konferencji – temat, data, opłata, maksymalna ilość miejsc, miejsce oraz ID administrator
* **papers** – zawiera referaty, a dokładnie ich temat, krótki opis, zawartość (plik) oraz ID autora (użytkownika)
* **reviewers** – recenzenci, ich dziedziny wiedzy oraz ID użytkownika (konta)
* **accomodations** – zakwaterowanie, a mianowicie opis, miejsce, koszt oraz ID konferencji, dla której przewidziany jest nocleg
* **applications** – złożone aplikacje danych referatów zgłoszonych dla danych konferencji
* **speeches** – przemowy, jakie przewidziane zostały dla danej konferencji – w tym długość przemowy oraz kolejność
* **revs\_alloc** – tabela pomocniczna dla uniknięcia relacji wielu-do-wielu, zawiera powiązania między recenzentami a konferencjami
* **entries** – tabela zawierająca rezerwacje noclegu danego użytkownika, w tym ID konferencji oraz zakwaterowania
* **reviews** – recenzje, w tym opinia, komentarz administrator, ocena tematu i zawartości, a także decyzja czy została zaakceptowana
* **payments** – płatności za noclegi

Poniżej znajdują się diagramy ERD wygenerowane dla obecnie funkcjonującej bazy danych. Wykorzystane zostały program *DBeaver* (obsługa bazy danych) oraz *schemaSpy* (generowanie graficznych ilustracji na podstawie istniejącej bazy danych).



1. **Opis skryptów**

W celach przetestowania poprawności utworzenia bazy danych zostały stworzone skrypty w języku SQL. Za zadanie miały one sprawdzić czy istniejąca baza odpowiada w pełni utworzonemu diagramowi ERD, na podstawie którego miała ona powstać.

Pierwszy skrypt zakładał dodanie danych do bazy, gdyż to sprawdzałoby poprawne utworzenie pól, ich właściwości oraz odpowiednie relacje między tabelami. Część skryptu, przedstawiająca dodanie danych do tabeli użytkownicy:

INSERT INTO public.users

VALUES(1, 'Damian', 'Jagiello', 'djag@baza.pl', 'jagdam1410'),

(2, 'Adrianna', 'Klepka', 'aklep@baza.pl', 'kleadr93'),

(3, 'Glowoslaw', 'Jacykow', 'gjac@baza.pl', 'jacglo13'),

(4, 'Radziwil', 'Jankowski', 'rjan@baza.pl', 'janrad192'),

(5, 'Agata', 'Krol', 'akro@baza.pl', 'kroaga532'),

(6, 'Albert', 'Nowak', 'anow@baza.pl', 'nowalb456'),

(7, 'Katarzyna', 'Dabrowski', 'kdab@baza.pl', 'dabkat741'),

(8, 'Linda', 'Wojcik', 'lwoj@baza.pl', 'wojlin651'),

(9, 'Miroslaw', 'Kozlowski', 'mkoz@baza.pl', 'kozmir168'),

(10, 'Szymon', 'Grabowski', 'szgrab@baza.pl', 'graszy79'),

(11, 'Michal', 'Nowicki', 'mnow@baza.pl', 'nowmic348'),

(12, 'Wiktoria', 'Adamczyk', 'wadam@baza.pl', 'adamwik978'),

(13, 'Kacper', 'Zajac', 'kzaj@baza.pl', 'zajkac15'),

(14, 'Julia', 'Dudek', 'jdud@baza.pl', 'dudjul35'),

(15, 'Aleksandra', 'Wieczorek', 'awie@baza.pl', 'wieale325'),

(16, 'Bartosz', 'Majewski', 'bmaj@baza.pl', 'majbar98'),

(17, 'Filip', 'Witkowski', 'fwit@baza.pl', 'witfil23'),

(18, 'Zuzanna', 'Stepien', 'zste@baza.pl', 'stepzuz69'),

(19, 'Jan', 'Michalak', 'jmich@baza.pl', 'micjan232'),

(20, 'Zofia', 'Rutkowski', 'zrut@baza.pl', 'rutzof54');

Pozostałe dane zostały wprowadzone w ten sam sposób na podstawie schematu bazy danych. Wszystkie pola zostały uzupełnione bez błędów co potwierdza poprawność utworzenia bazy danych.

Kolejnym krokiem było stworzenie kilku prostych zapytań, które udowodniłyby poprawność wprowadzania danych oraz spójność bazy. Przykładowym zapytaniem jest wyświetlenie ilości wejściówek jaką zakupili użytkownicy realizowane w następujący sposób:

select u.first\_name || ' ' || u.last\_name, count(\*) as "Ilosc wejsciowek"

from users u, payments p

where u.id = p.user\_id

group by p.user\_id, u.first\_name, u.last\_name;

Zapytanie zwróciło odpowiednie wartości co świadczy o poprawnych relacjach między tabelami.

1. **Interfejs**

Szablon graficzny modeluje wygląd systemu jako strony www. Założeniem było, aby wykonany przez nas layout był prosty, czytelny i użyteczny a zarazem atrakcyjny dla oka. Do realizacji szablonu graficznego wybraliśmy schemat strony internetowej one-page. Taki szablon mieści się na jednej stronie internetowej i zapewnia bardziej płynny interfejs.

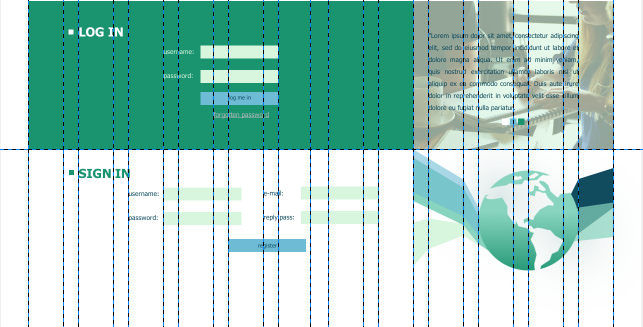
Pierwszym etapem utworzenia szablonu było rozplanowanie układu wszystkich elementów potrzebnych, by zapewnić użytkownikowi wszystkich przewidzianych przypadków użycia. Na stronie głównej takimi przypadkami są logowanie oraz rejestracja.



Rysunek 9.3 Panel logowania i rejestracji

Zdecydowaliśmy się dodać także menu w stopce szablonu, które ułatwia nawigację w systemie i kontakt. Do strony głównej zdecydowaliśmy się dodać elementy przybliżające ogólne informacje o projekcie.

Ważnym elementem w budowie strony www naszego systemu było zapewnienie responsywności. Oznacza to, że nasz szablon po zakodowaniu będzie dostosowywał się do rozdzielczości i wielkości okna użytkownika. Jest to niezbędny element w szablonach typu one-page. Aby strona była responsywna, należy nałożyć na nią siatkę (grid). Wybraliśmy grid 12c, co oznacza że szablon będzie podzielony na 12 kolumn. Od teraz, każdy element dodawany do schematu musi zawierać się między kolumnami i być dopasowanym do krawędzi.



Rysunek 9.4 Grid nałożony na panel logowania i rejestracji

Po zalogowaniu się mamy dostęp do menu. Aby mieć dostęp do panelu administratora musimy zalogować się jako administrator, aby uzyskać dostęp do panelu użytkownika musimy zalogować się jako administrator bądź użytkownik.



Rysunek 9.5 Widok na menu użytkownika



Rysunek 9.6 Widok na menu administratora

Jak widać w panelu administratora mamy możliwość edytować konferencje, zatwierdzać artykuły oraz użytkowników i akceptować recenzentów. Jako użytkownik możemy wyświetlić listę konferencji, zgłaszać artykuły i chęć bycia recenzentem.

Po stworzeniu interfejsu należało zakodować poszczególne elementy. I tak np. przyciski nawigacyjne na stronie, a przede wszystkim:

* menu administratora, zawierające przyciski do funkcjonalności takich jak:

-edycja konferencji,

-zatwierdzanie artykułów,

-zatwierdzanie użytkowników,

* menu uczestnika, zawierające przyciski do funkcjonalności takich jak:

-przeglądanie konferencji,

-zgłoszenie artykułu,

-zgłoszenie zostania recenzentem.

|  |  |
| --- | --- |
|  |  |
| a)Menu Uczestnika | b)Menu Administratora |

Na poszczególnych funkcjonalnych podstronach dodano przyciski „Powrót”, powodujący przeniesienie do poprzednio odwiedzonego menu.

Oba powyższe menu, mają przyciski powrotu na stronę główną oraz wylogowania.

Kolejnym elementem było stworzenie krótkiego opisu projektu na stronie głównej a następnie przetestowanie działania systemu pod wieloma kontami.

Przyciski nawigacyjne zostały wykonane w technologii JADE.

Przykładowy przycisk – powrót na stronę główną

…

form(action='/')

input(name='Powrot', value='Powrót na stronę główną', type='submit')

…

1. **Podsumowanie**

Udało się wykonać:

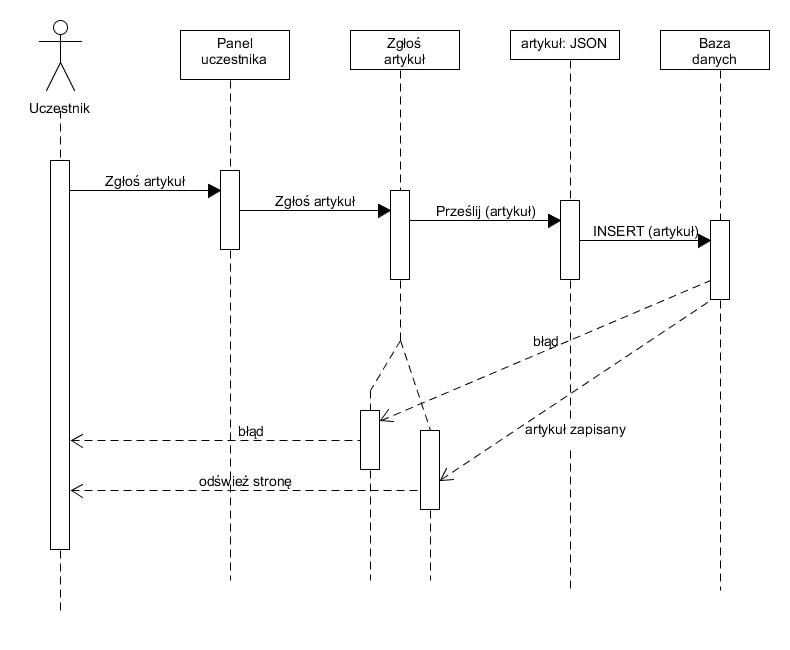
* Bazę danych,
* Stronę internetową,
* Wdrożenie na platformę,
* Rejestrację nowych użytkowników w systemie,
* Logowanie do systemu na różnego typów konta,
* Przeglądanie, zatwierdzanie oraz usuwanie listy konferencji.

Nie udało się zrealizować następujących zadań:

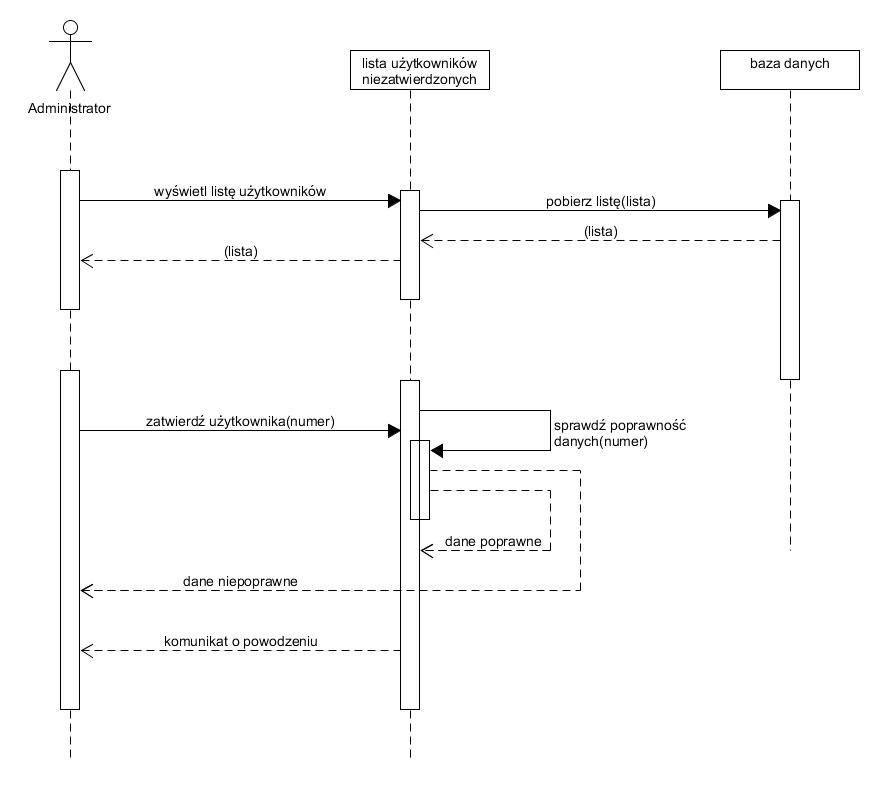
* Dodawanie konferencji,
* Brak podglądu ocen artykułów,
* Usługi realizacji opłat za pomocą systemu *PayPal*.

Powodem niezrealizowania wszystkich założeń oraz pomysłów było skomplikowanie projektu oraz brak odpowiedniej ilości czasu.

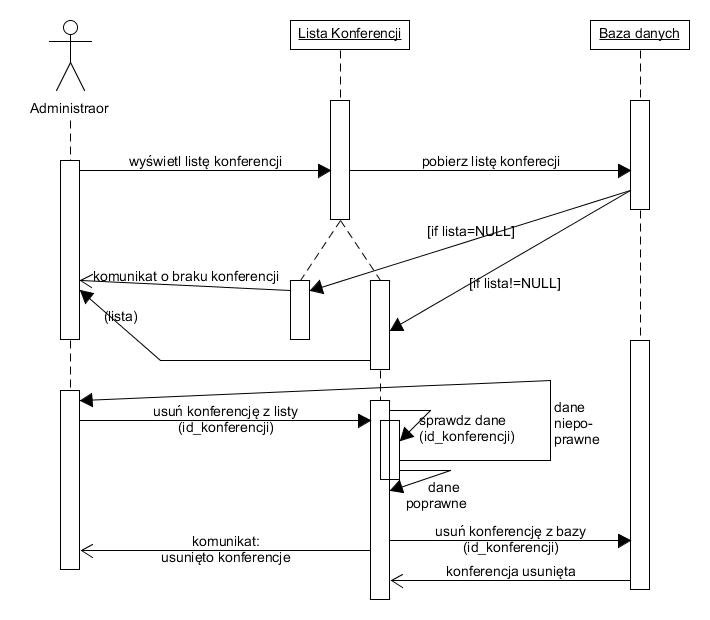
**10. Załączone diagramy sekwencji**



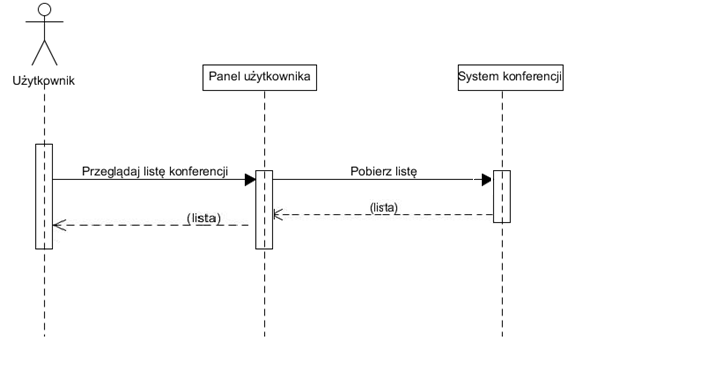
1. Zgłaszanie artykułu



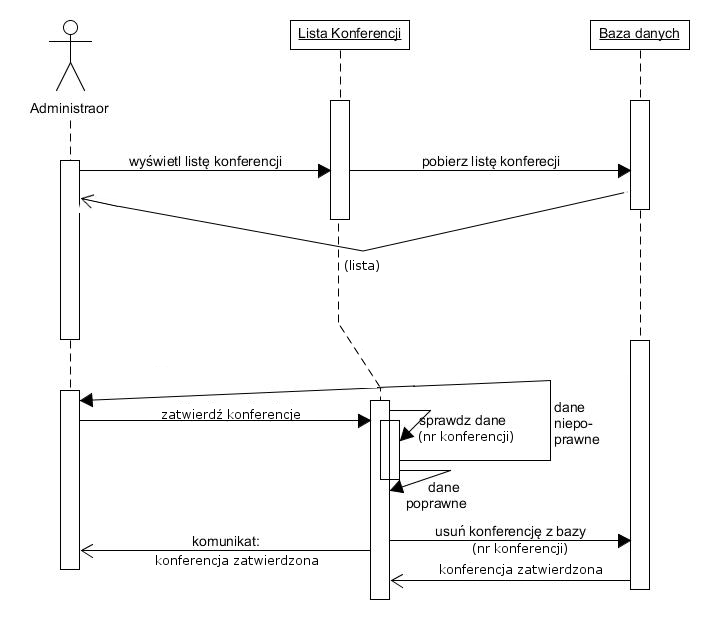
2. Zatwierdzanie użytkowników



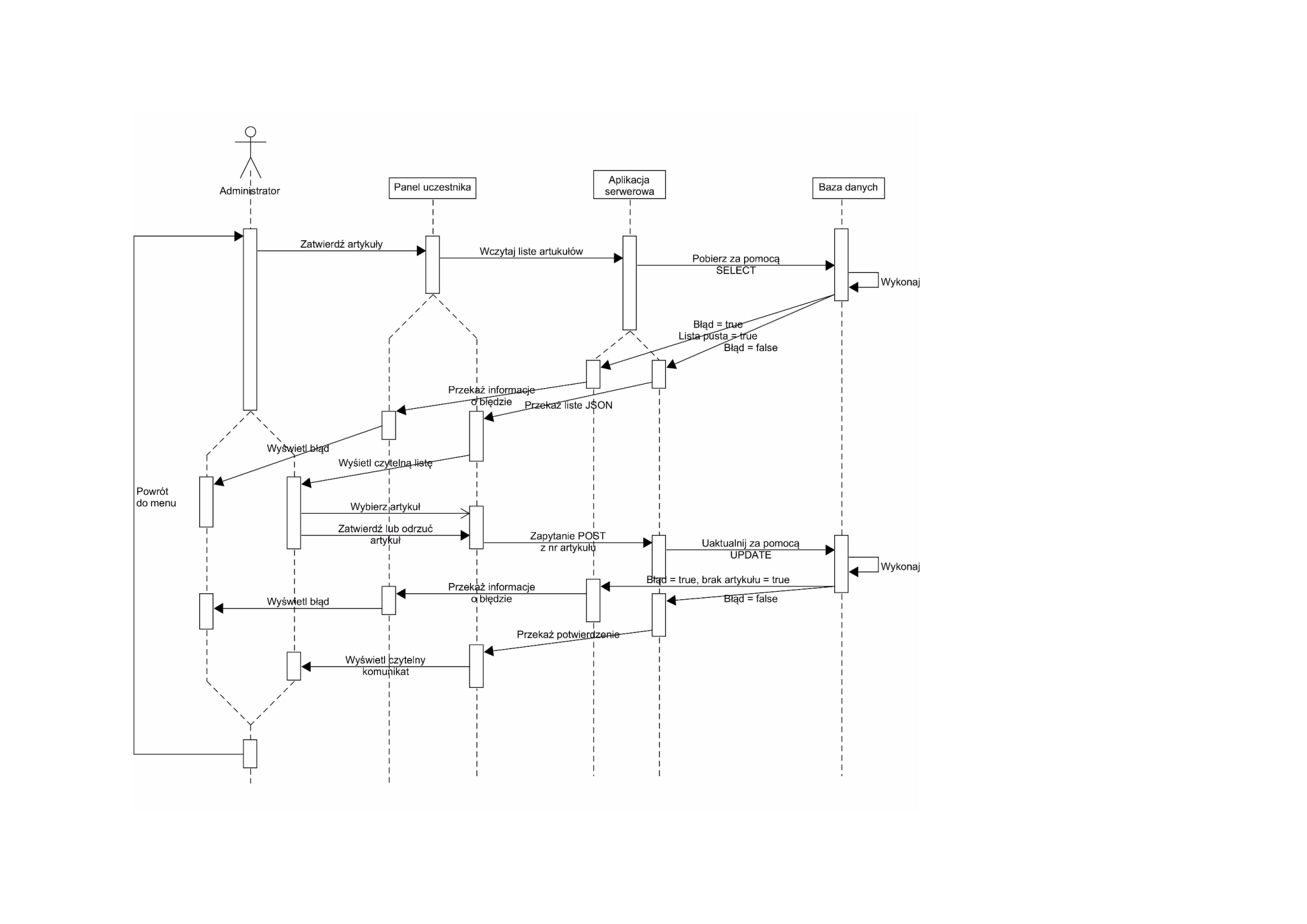
3. Usuwanie konferencji



4.Przeglądanie listy konferencji (użytkownik)



5. Dodanie konferencji do listy



6. Zatwierdzanie artykułu