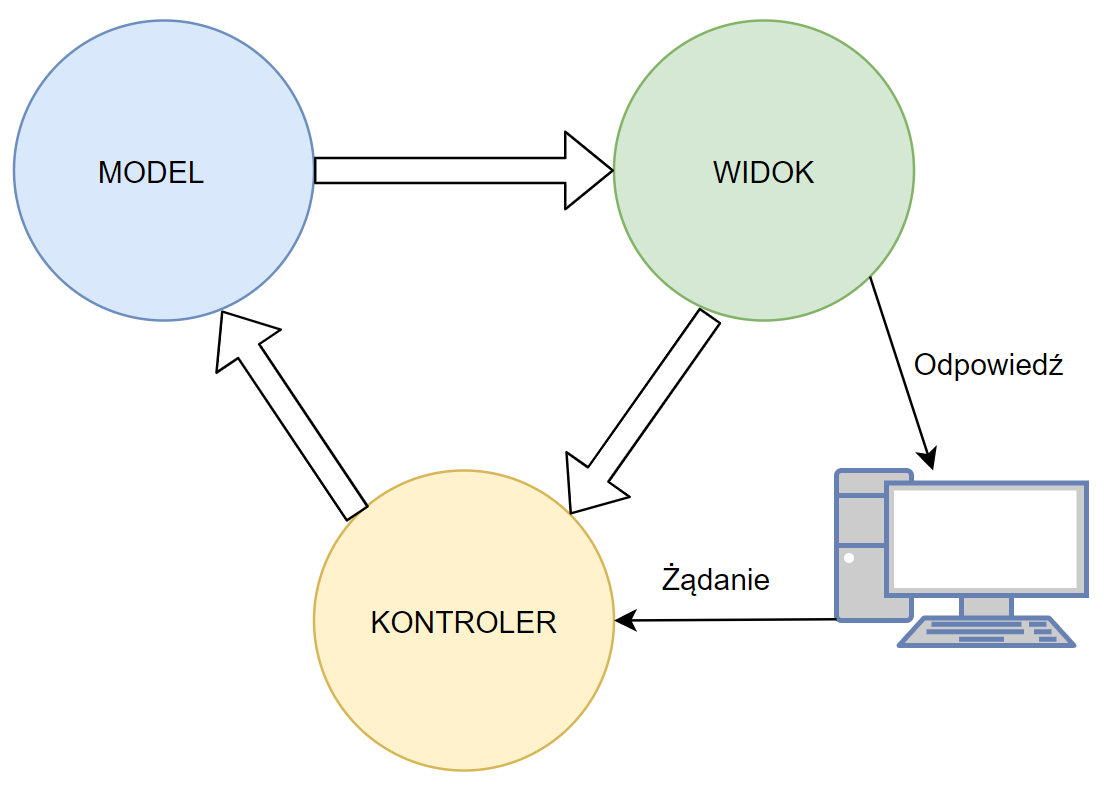
System wspomagania pracy firmy sprzątającej

**Etap 3**

**Mateusz Kita Z710**

# Projekt architektury systemu

System oparty będzie na stylu architektury MVC(Model-Widok-Kontroler), dzięki temu aplikacja będzie podzielona na 3 niezależne od siebie warstwy. Ułatwi to utrzymanie systemu kosztem zwiększonej złożoności.



W celu integracji części front i back-endowej zostanie stworzone API, po którym będzie komunikowała się aplikacja.

# Projekt bazy danych

## Dokładny opis bazy danych

Systemem bazy danych wybranym do utworzenia systemu jest MongoDB. Jest to nie relacyjny system gdzie dokumenty zapisywane są w formacie BSON, czyli Binarny JSON.

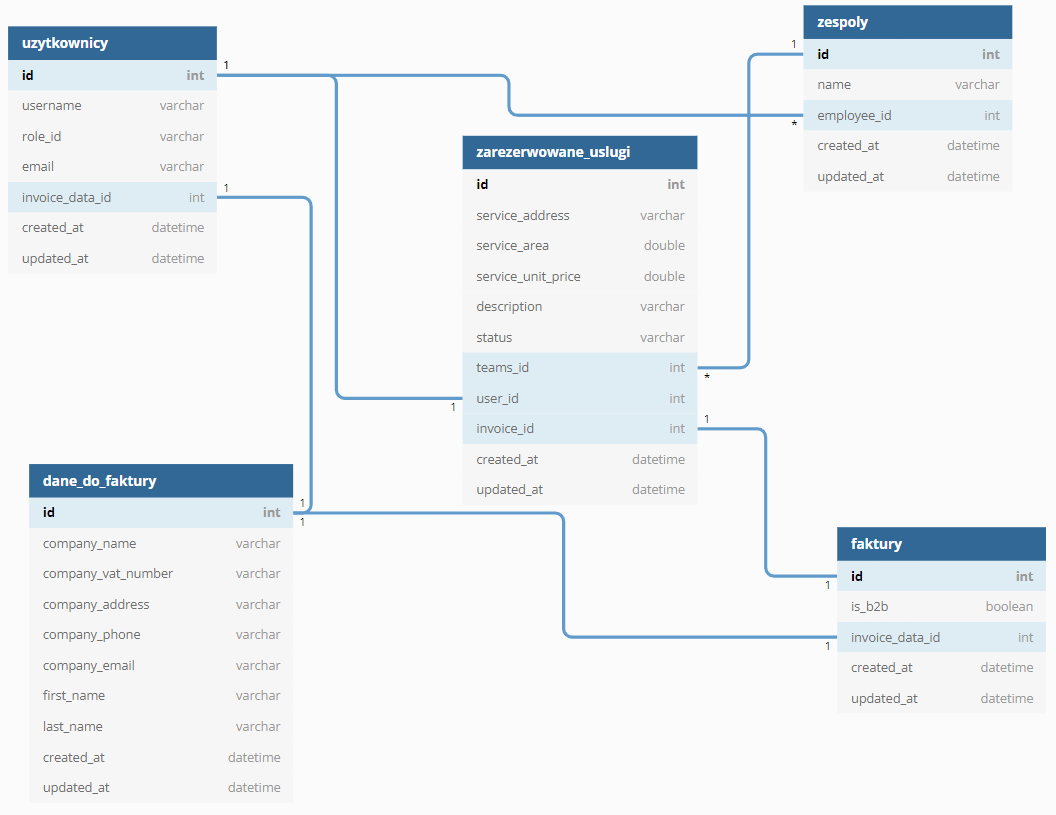


Technologia ta została wybrana ponieważ pozwala na szybki i elastyczny proces rozwijania aplikacji z NodeJS, który został użyty w części back-endowej systemu.

Docelowo baza danych składała się będzie z 5 tabel:

* uzytkownicy - przechowuje dane dotyczące kont użytkowników systemu,
* zespoly - przechowuje dane zespołów sprzątających – listę pracowników należących do zespoły oraz przypisane zlecenia,
* zarezerwowane\_uslugi - przechowuje informacje o zarezerwowanych usługach,
* faktury - przechowuje informacje o wystawionych fakturach,
* dane\_do\_faktury - przechowuje dane klienta potrzebne do wystawienia faktur

## Diagram tabel



## Opisy pól i ich atrybutów

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nazwa tabeli | Opis tabeli | Nazwa pola | Typ pola | Opis pola |
| uzytkownicy | Przechowuje dane dotyczące kont użytkowników systemu | id | int | Klucz główny |
|  |  | username | varchar | Nazwa użytkownika, potrzebna do zalogowania się |
|  |  | role\_id | varchar | Nazwa roli użytkownika |
|  |  | invoice\_data\_id | int | Klucz obcy. Id w tabeli dane\_do\_faktury |
|  |  | created\_at | datetime | Data utworzenia rekordu |
|  |  | updated\_at | datetime | Data ostatniej modyfikacji |
| dane\_do\_faktury | Przechowuje dane klienta potrzebne do wystawienia faktur | id | int | Klucz główny |
|  |  | company\_name | varchar | Nazwa przedsiębiorstwa |
|  |  | company\_vat\_number | varchar | Numer NIP |
|  |  | company\_address | varchar | Adres przedsiębiorstwa |
|  |  | company\_phone | varchar | Numer telefonu |
|  |  | company\_email | varchar | Email |
|  |  | first\_name | varchar | Imię |
|  |  | last\_name | varchar | Nazwisko |
|  |  | created\_at | datetime | Data utworzenia rekordu |
|  |  | updated\_at | datetime | Data ostatniej modyfikacji |
| zarezerwowane\_usługi | Przechowuje informacje o zarezerwowanych usługach | id | int | Klucz główny |
|  |  | service\_address | varchar | Adres miejsca usługi |
|  |  | service\_area | double | Powierzchnia do sprzątania |
|  |  | service\_unit\_price | double | Cena za 1m2 usługi sprzątania |
|  |  | status | varchar | Status zlecenia |
|  |  | description | varchar | Opis zlecenia |
|  |  | teams\_id | int | Klucz obcy. Id w tabeli zespoly |
|  |  | user\_id | int | Klucz obcy. Id w tabeli uzytkownicy |
|  |  | invoice\_id | int | Klucz obcy. Id w tabeli faktury |
|  |  | created\_at | datetime | Data utworzenia rekordu |
|  |  | updated\_at | datetime | Data ostatniej modyfikacji |
| faktury | Przechowuje informacje o wystawionych fakturach | id | int | Klucz główny |
|  |  | is\_b2b | boolean | Informacja czy klient posiada działalność gospodarczą |
|  |  | invoice\_data\_id | int | Klucz obcy. Id w tabeli dane\_do\_faktury |
|  |  | created\_at | datetime | Data utworzenia rekordu |
|  |  | updated\_at | datetime | Data ostatniej modyfikacji |
| zespoły | Przechowuje dane zespołów sprzątających – listę pracowników należących do zespoły oraz przypisane zlecenia. | id | int | Klucz główny |
|  |  | name | varchar | Nazwa zespołu |
|  |  | employee\_id | int | Klucz obcy. Id w tabeli uzytkownicy |
|  |  | created\_at | datetime | Data utworzenia rekordu |
|  |  | updated\_at | datetime | Data ostatniej modyfikacji |

## Opis mechanizmów bazy danych

Baza danych będzie dodawała unikalny identyfikator do każdego nowo utworzonego rekordu.

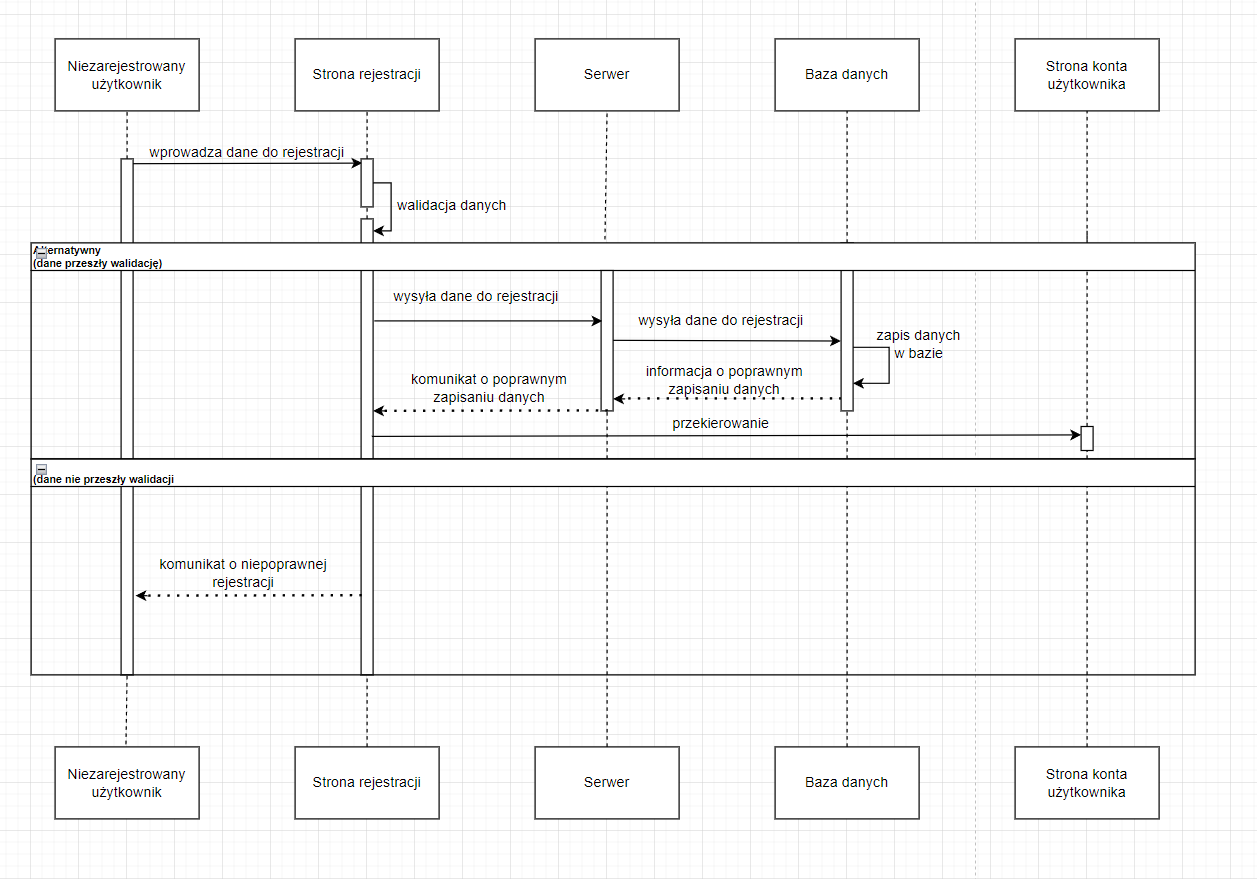
Komunikacja back-endu aplikacji z bazą danych będzie odbywała się za pomocą biblioteki *mongoose(*[*https://www.npmjs.com/package/mongoose*](https://www.npmjs.com/package/mongoose)*).* Biblioteka ta posiada prostą składnię do modelowania obiektów bazy danych oraz adapter do utworzenia połączenia z bazą danych.

W celu zmiany statusu zarezerwowanej usługi na usługę anulowaną przez brak opłaty w bazie danych zastosowany zostanie wyzwalacz sprawdzający status płatności po 3 dniach od daty złożenia rezerwacji.

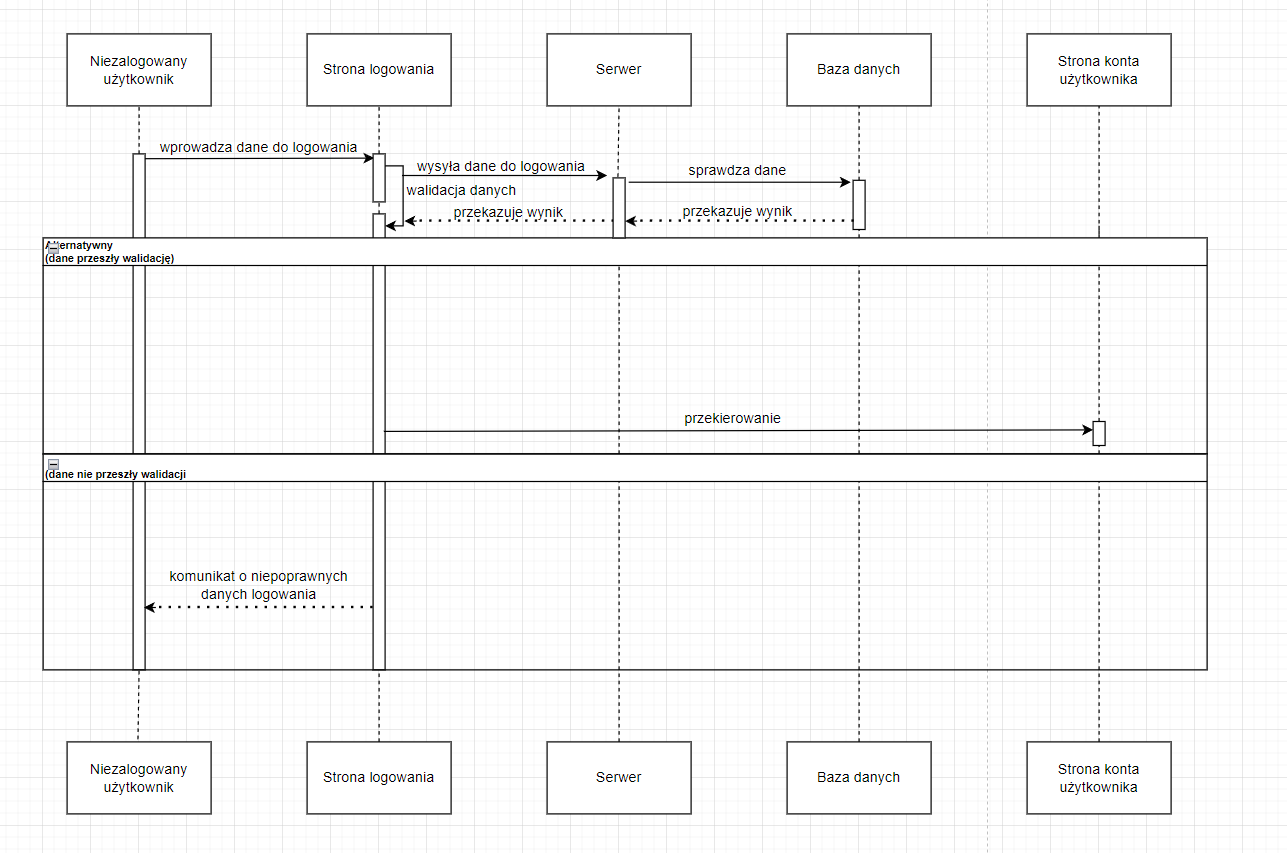
Zamówienia do systemu muszą być składane z przynajmniej tygodniowym wyprzedzeniem w celu ustalenia odpowiedniego harmonogramu prac ekip sprzątających. Próba rezerwacji z datą wcześniejszą nie powinna zakończyć się powodzeniem.

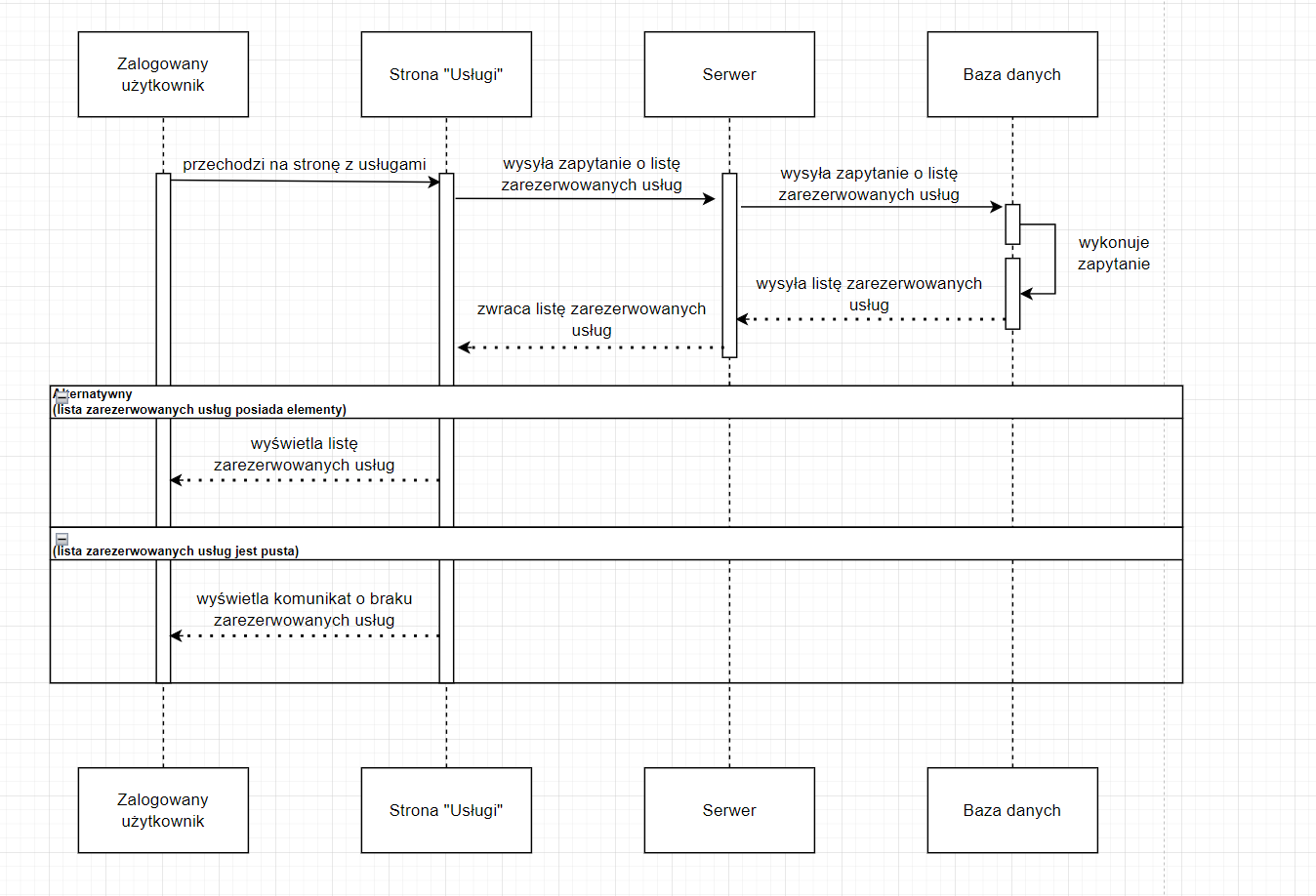
# Projekt algorytmów

1. Diagram sekwencji rejestracji użytkownika:



1. Diagram sekwencji logowania użytkownika:



3. Diagram sekwencji wyświetlania listy zarezerwowanych usług:  
  


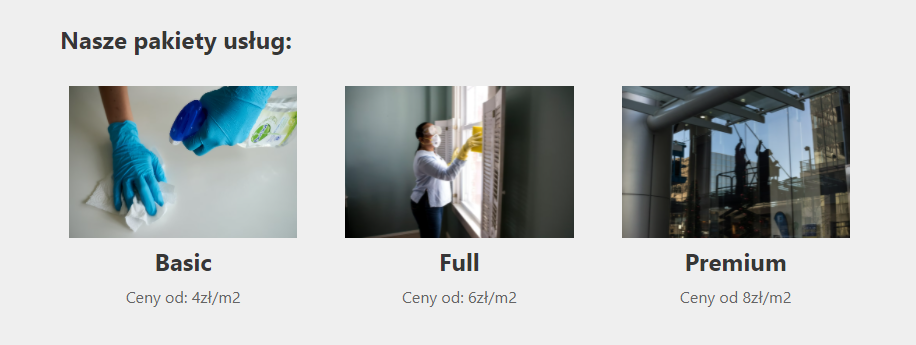
4.

# Projekt interfejsu użytkownika

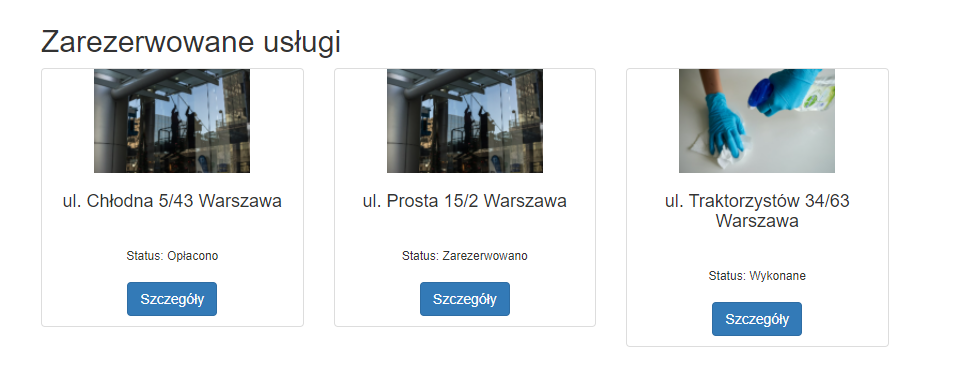
#### Ekran główny – widoczny dla użytkownika z dowolną rolą:



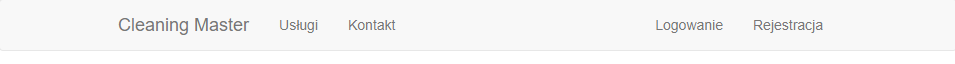
#### Nasze usługi:



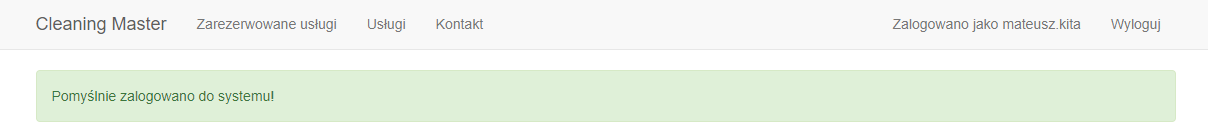
#### Widok listy zarezerwowanych usług:



#### Nagłówek dla niezalogowanego użytkownika – „gościa”:



#### Nagłówek dla zalogowanego użytkownika:



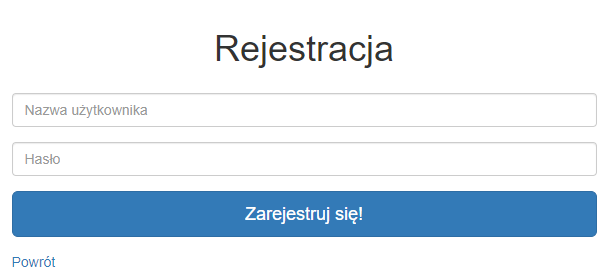
#### Stopka:



#### Widok formularza logowania:

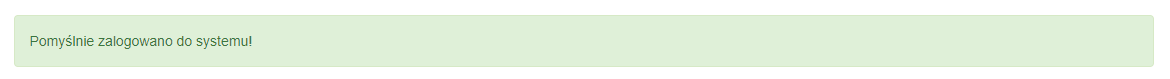


#### Widok formularza rejestracji nowego użytkownika:



#### Wiadomości informacyjne dla użytkowników:





# Projekt testów kontrolnych

Testami jednostkowymi pokryte zostanie około 80% kodu, ponieważ ustalono, że taki próg jest optymalnie dobrany pod względem bezpieczeństwa i czasu poświęconego na stworzenie i utrzymywanie testów jednostkowych.

Testami integracyjnymi zostaną pokryte kluczowe fragmenty aplikacji, od których zależy poprawne funkcjonowanie najistotniejszych części aplikacji. W szczególności komunikacja warstwy front-endowej z back-endową oraz zapis i odczyt danych z bazy danych.

Testy obciążeniowe – testy z wykorzystaniem narzędzia typu Gatling(<https://gatling.io/>) polegające na wysyłaniu dużej ilości testowych zapytań i sprawdzeniu dla ilu z nich otrzymano odpowiedź.

## Scenariusze testowe dla testów funkcjonalnych:

#### Proces rejestracji użytkownika

Podanie danych rejestracyjnych oraz kliknięcie przycisku zatwierdzającego wysłanie formularza rejestracyjnego powinno utworzyć nowego użytkownika w bazie danych z danymi takimi jak podał użytkownik.

#### Proces logowania użytkownika

Wpisanie prawidłowych danych do logowania powinno zmienić ustawić odpowiednią rolę użytkownikowi.

#### Wylogowanie użytkownika

Kliknięcie przycisku wylogowania powinno ograniczyć użytkownikowi zakres uprawnień z posiadanej roli do roli „Gościa”.

#### Sprawdzenie walidacji formularza

Wpisanie nieprawidłowych danych, takich jak cyfry do pola z imieniem powinno skutkować pojawieniem się wiadomości z informacją o błędzie.

#### Sprawdzenie poprawnego działania ról użytkownika

W zależności od posiadanej roli użytkownik powinien mieć możliwość wykonania odpowiednich akcji, a w przypadku braku uprawnień powinien dostać informację o odmowie dostępu.

#### Zmiana statusu zarezerwowanej usługi

Opłacenie usługi powinno wiązać się ze zmianą statusu na Zamówienie opłacone.

#### Przypisanie zespołu sprzątającego do zarezerwowanej usługi

Sprawdzenie poprawności mechanizmu przypisywania zespołu sprzątającego do konkretnej usługi.

#### Sprawdzenie poprawności działania generatora faktur

Podane przez klienta dane do faktury powinny znaleźć się w wygenerowanym dokumencie PDF.