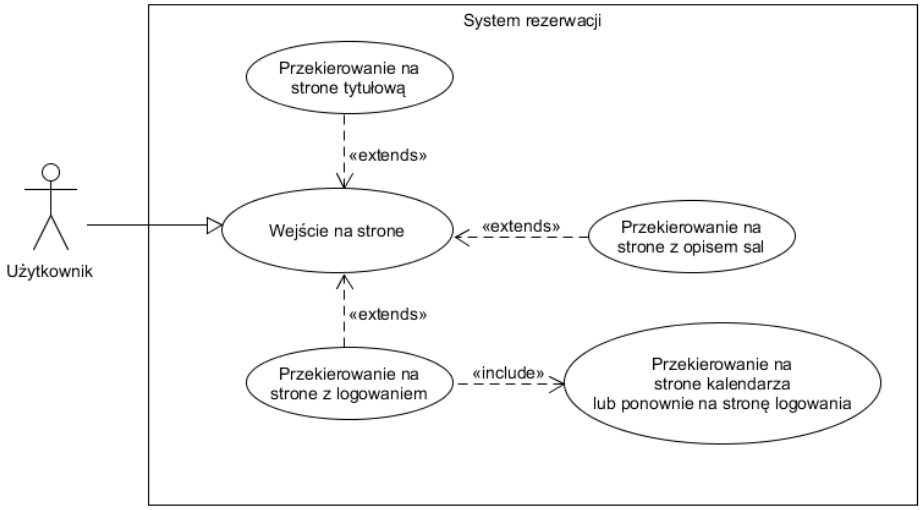
Rola: zajęcie się backend’em i realizaję otrzymanych zadań

Funkjonalności:

1. Przekierowywanie stron

Jest to początek projektu lecz już sama zmiana strony może być kwestionowana, ponieważ przekierowywanie na inne strony po zrobieniu czegoś jest nie praktyczne ponieważ tracimy dane z poprzedniej strony lub jeżeli je przekażemy w sesji lub requestem musimy poczekać na odświeżenie strony. Napisanie funkcjonalności w jqery na ajaxie pozwoli nie odświeżać strony lecz nie jest to potrzebne na przykład dla przechodzenia między stroną tytułową a stroną admina czy stroną logowania, tym bardziej że ajax nie zmieni nam struktury strony tylko wydzielony element, dlatego dodanie dodatkowej strony jest łatwiejsze i bardziej.

Diagram użycia dla logowania:



Otworzenie nowego linku przyciskiem czyli wklejenie linku do htmla i określenie linku w urlach projektu

test

urlpatterns = [url(app/login/', views.login, name='login'),]

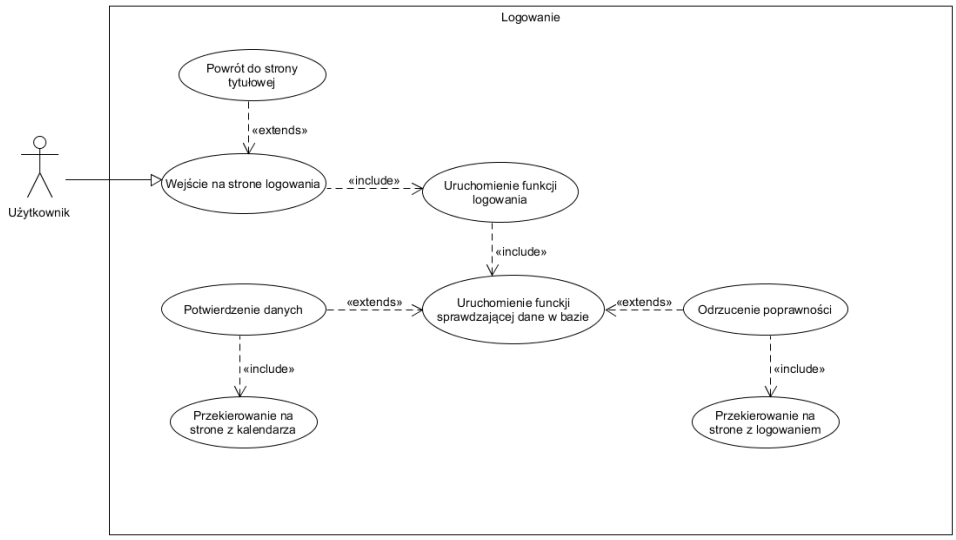
Po wejściu w funkcję login włączającą się po wpisaniu danego adresu można też przejść na inną stronę wpisując

Return redirect(‘app/login.html’)

Testem jest zwykłe naciśnięcie przycisku i sprawdzenie czy strona się otworzy.

2. Logowanie

Logowanie było jednym z elementów które trzeba było zrobić przez specyfikacje systemu, który posiadał zarejestrowane osoby które mogły rezerwować sale i resztę która nie mogła. Pierwszym problemem jest to jak określić czy ktoś jest zarejestrowany. Normalnie porównywalibyśmy dane w logowaniu z danymi w bazie, lecz danej bazy nie posiadaliśmy, wiedzieliśmy natomiast że poprzedni system korzysta ze API które łączy się z bazą danych firmy, stwierdziliśmy więc że połączymy się z przykładowym API które wiedzieliśmy że posiada jednego użytkownika po czym wysyłaliśmy jego dane jeżeli logowany user wpisywał dane które były w naszej wbudowanej bazie stworzonej razem z projektem django, takie rozwiązanie pozwala na łatwiejszą implementację zmian na prawdziwe API oraz pozwala na działanie naszego projektu. Następnym problemem jest szyfrowanie hasła aby nie było podatne na podsłuchiwanie.



Tak jak przy przekierowywaniu strony używane było wywołanie funkcji tak po kliknięciu w przycisk uruchomi się funkcja logowania. Musi zatem pobrać wpisane dane i je gdzieś zapisać by je porównać, w tym celu trzeba zrobić model który będziemy porównywać. Aby to zrobić użyłem gotowych modeli usera z frameworka do autentykacji w django. Następnie trzeba zaimplementować gotowe adresy z frameworka do autentykacji. Po stowrzeniu usera trzeba porównać jego dane z wpisywanymi danymi w logowaniu

```
path('accounts/', include('django.contrib.auth.urls'))
```

```
{% block content %}
  {% if form.errors %}
    Złe dane
  {% endif %}
  <form method="post" action="{% url 'login' %}">
    {% csrf_token %}
    <table>
      <tr>
        <td>{{ form.username.label_tag }}</td>
        <td>{{ form.username }}</td>
      </tr>
      <tr>
        <td>{{ form.password.label_tag }}</td>
        <td>{{ form.password }}</td>
      </tr>
    </table>
    <input type="submit" value="login" />
    <input type="hidden" name="next" value="{{ next }}" />
  </form>
{% endblock %}
```

Testem jest to że po wpisaniu dobrych danych zostaniemy przekierowani automatycznie do strony profilowej, która nie jest napisana więc otrzymując błąd a nie odpowiedz o złych danych logowanie działa.

Dodać usera można również w konsoli:

```
from django.contrib.auth.models import User
user = User.objects.create_user('login','email','haslo')
user.first_name = 'test'
user.last_name = 'test'
```

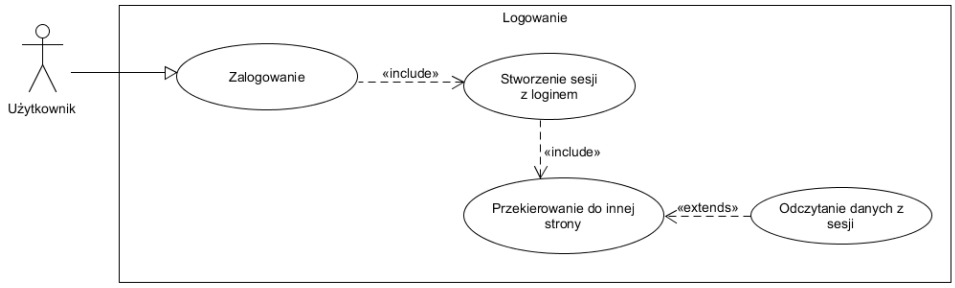
```
user.save()
```

Ta wersja logowania nie została dodana do projektu ponieważ pracując nad nią ktoś inny też zrobił logowanie, inaczej lecz prace trwały na podstawie innego kodu i nie było sensu go zmieniać

3. Autoryzacja

Autoryzacja jest potrzebna by nie prosić o logowanie po każdym renderowaniu nowej strony. Można było tak zrobić co było dość proste i zrobione prezemnie czyli dostanie się na stronę rezerwacji po logowaniu i zatwierdzenie rejestracji po ponownym logowaniu, lecz kłóci się to z założeniem projektu i nie rozwiązuje problemu zapisania rezerwacji na określonego usera. Trzeba zatem przekazać przy logowaniu login do następnej strony. Użyłem w celu frameworka sessions, lecz można to zrobić również za pośrednictwem requestów.

Diagram




```
def login(request):
      If request.moethod == ‘POST’:
         request.session['login'] = 'username'
         return render(request,'calendar’)


def calendar(request):
      if request.session.get('login'):
         response += " {0}
   <br>".format(request.session.get('name'))
         return HttpResponse(response)
      else:
         return redirect('login/')
```

Funkcja logowania może zatem tak wyglądać

```
def loginPage(request):
        if request.method == 'POST':
            username = request.POST.get('username')
            password = request.POST.get('password')
            params = {'login':username1,'password':password1}
            user = authenticate(request, email=username,
haslo=password)
            if user is not None:
                 login(request, user)
         Requset.session[‘login’] = ‘username’
                 return redirect('sale')
            else:
                messages.info(request, 'Username OR password is
incorrect')
        context = {}
           return render(request, 'accounts/login.html', context)
```

Nie jest to sprawdzone i napisane pod projekt przez co nie zostało zaimplementowane.

W praktyce moim zadaniem stało się pomaganie w pracy, rozwiązywanie błędów które trzeba rozwiązać przez znajomość dokumentacji lub znajdowanie małych błedów w kodzie, pomoc z postawieniem lokalnych serverów na początku pracy, aktywne uczestnictwo w rozmowach i komunikacja z prowadzącym oraz sama prezentacja projektu przed rokiem informatyki i zarządzania.