**Uniwersytet Technologiczno –Przyrodniczy w Bydgoszczy**

Wydział Telekomunikacji, Informatyki i Elektrotechniki



Projekt

Programowanie aplikacji internetowych

**Dokumentacja projektu**

**„Serwis umożliwiający organizację imprez”**

***COVID AFTER-PARTY***

*Michael Szychulski,*

*Nr indeksu: 113115*

*Dominik Wiśniewski,*

*Nr indeksu: 113131*

*Michał Sulecki,*

*Nr indeksu: 113112*

*Informatyka stosowana*

*Semestr VI*

*grupa 2*

# Specyfikacja projektu

1. Ogólny opis projektu

System z założenia ma umożliwić interakcje pomiędzy organizatorem wydarzenia, a uczestnikami. Organizator tworzy nowe wydarzenie, określa liczbę gości, miejsce, termin i pozostałe szczegóły wydarzenia. Po ustaleniu szczegółów organizator może opublikować wydarzenie, aby było dostępne dla uczestników. Założyciel może dodatkowo wysyłać zaproszenia do grupy jego np. wcześniejszych uczestników, za pośrednictwem e-maili. Uczestnicy imprez po wejściu na aplikacje maja dostępną wyszukiwarkę okolicznych wydarzeń. Dostępne jest wyszukiwanie interesujących wydarzeń poprzez system tagów, które określają tematykę. Następnie po wybraniu wydarzenia można przejrzeć szczegóły, które zostały ustalone przez organizatora i zarezerwować swoje miejsce na wydarzeniu, lub określić swoje uczestnictwo. Dostępna jest też komunikacja mailowa między założycielem wydarzenia/osobą odpowiedzialną, a uczestnikami.

Poprzez listę dostępnych wydarzeń i możliwość rezerwacji miejsca, założyciel wydarzenia od razu wie, ile gości może się spodziewać i jakie środki należy przygotować. Użytkownicy od razu mają informację, ile miejsc jest jeszcze dostępnych i mogą łatwiej zapisać się na dane wydarzenie. Dodatkowo aplikacja przewiduje wyszukiwanie wydarzeń wedle tematyki co znacząco ułatwi wyszukiwanie interesujących wydarzeń.

1. Cel i przeznaczenie projektu

Projekt ma umożliwić organizację wydarzeń zbiorowych poprzez poprawę widoczności, przystępne i szybkie zakładania, publikację i zarządzanie wydarzeniami i listą uczestników .

Dostarczenie serwisu do zarządzania wydarzeniami zbiorowymi realizujący funkcjonalności związane z : //wymienić.

Przeznaczeniem projektu jest implementacja aplikacji na chmurze obliczeniowej, do której można będzie uzyskać dostęp przez przeglądarkę. Aplikacja ma za zadanie umożliwić przejrzysty, szybki i przystępny sposób na zakładanie i zarządzanie wydarzeniami zbiorowymi i listą uczestników, oraz na przeglądanie i wyrażanie uczestnictwa w opublikowanych wydarzeniach. Aplikacja jest skierowana do każdej grupy wiekowej

1. Specyfikacja wymagań na oprogramowanie
   1. Wymagania funkcjonalne:
      1. Rejestracja użytkownika z wymaganymi polami:
         1. Imię – pole tekstowe \*
         2. Nazwisko – pole tekstowe \*
         3. Adres Email – pole tekstowe
         4. Hasło – pole tekstowe \*
         5. Powtórzenie hasła – pole tekstowe
         6. Data urodzenia – data
         7. Miasto zamieszkania – pole tekstowe \*
         8. Numer telefonu – pole tekstowe \*
      2. Logowanie za pomocą email i hasła dla zarejestrowanego konta użytkownika
      3. Tworzenie i zarządzanie wydarzeniami
         1. Utworzenie wydarzenia z polami:
            1. Nazwa – pole tekstowe \*
            2. Lokalizacja \*

Miasto– pole tekstowe \*

Kod pocztowy– pole tekstowe \*

ulica – pole tekstowe \*

numer budynku – int \*

numer mieszkania - int

* + - * 1. Data i godzina wydarzenia - data\*
        2. Data i godzina publikacji - data
        3. Status aktywności – enum [ACTIVE / COMPLETED / CANCELLED / DRAFT]
        4. Status widoczności – enum [PUBLIC/PRIVATE]\*
        5. Lista tagów – pole tekstowe
        6. Lista gości –

połączenie wiele-do-wielu między Tabelami Event – User ze statusami:

Accepted

Denied

Pending

* + - * 1. Maksymalna liczba gości - int
        2. Opis – pole tekstowe \*
        3. Zdjęcia - plik [jpg / png]
        4. Ograniczenie wiekowe – int \*
        5. Wymagana zgoda organizatora - boolean
      1. Anulowanie wydarzenia
      2. Edycja informacji wydarzenia
    1. Wysyłanie prośby o dołączenie do wydarzenia:  
       Użytkownik po wyszukaniu wydarzenia decyduje się dołączyć za pomocą przycisku do listy gości. Jeżeli wydarzenie wymaga akceptacji organizatora, to uczestnik dopisany jest do listy gości ze statusem PENDING.
    2. Zarządzanie zaproszeniami
       1. Wyszukiwanie użytkowników do zaproszenia po:
          1. Imieniu
          2. Nazwisku
          3. Adresie email
          4. Mieście
          5. Dacie urodzenia
       2. Wysyłanie zaproszeń
       3. Wyświetlanie i zarządzanie otrzymanych zaproszeń poprzez widok
          1. Akceptuj

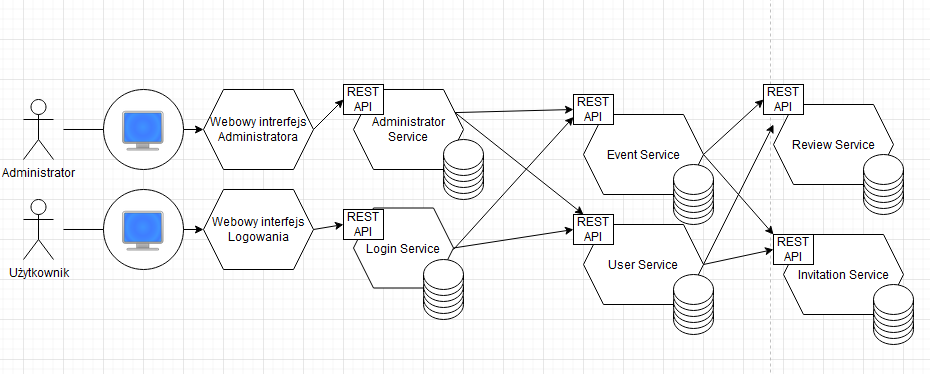
Akceptacja jest możliwa do momentu, gdy maksymalna liczba gości dla wydarzenia nie zostanie osiągnięta

* + - * 1. Odrzuć
    1. Wyszukiwanie wydarzeń za pomocą:
       1. Tagów
       2. Lokalizacji
       3. Daty
       4. Nazwie
       5. Minimalnym wieku
    2. Zarządzanie komentarzami dla zakończonych wydarzeń, w których użytkownik wziął udział
       1. Dodawanie
          1. Ocena - int[1 – 5] \*
          2. Komentarz
       2. Edycja
       3. Usuwanie
    3. Wysyłanie zapytań do organizatora wydarzenia
       1. Formularz z polem tekstowym i przyciskiem, który po naciśnięciu wysyła maila do organizatora wydarzenia
  1. Wymagania na bazę danych:  
     Relacyjna baza danych oparta na mySQL. Baza będzie posiadała jednego użytkownika z pełnymi uprawnieniami. Dostęp do niego będzie można uzyskać za pomocą panelu administratora aplikacji.
  2. Wymagania bezpieczeństwa:  
     Użytkownik, który się zaloguje otrzymuje token, i za jego pomocą jest on autoryzowany w serwisie. Token zostaje otrzymany na określony czas. Akcje użytkownika odświeżają token. Po minięciu określonego czasu bez żadnej akcji użytkownika, zostaje on wylogowany.  
     Każdy użytkownik może być jednocześnie organizatorem jak i uczestnikiem wydarzeń, przez co jego uprawnienia zależne są od jego pozycji względem wydarzenia.  
     Dodatkowo aplikacja będzie posiadała konto administratora z większymi uprawnieniami.  
     Zabezpieczenie aplikacji zostanie zintegrowane za pomocą Spring Security.
  3. Wymagania sprzętowo-wydajnościowe:  
     Jednostka obliczeniowa z dostępem do Internetu na której uruchomiony będzie aplikacja obsługująca klientów i bazę danych.   
     Minimalne wymagania serwera:  
     1 TB pojemności dyskowej  
     16 GB RAM  
     Procesor 6-cio rdzeniowy, 12-to wątkowy  
     Przepustowość sieciowa powyżej 10Mbps pobierania i wysyłania
  4. Wymagania na testowanie:  
     Przeprowadzone zostaną testy jednostkowe sprawdzającą logikę wykonywaną na danym endpoincie. Pokrycie kodu zakładane jest na powyżej 50%. Planowane jest kilka większych testów funkcjonalnych i integracyjnych, które sprawdzą współpracę poszczególnych części systemu. Będzie to sprawdzenie kluczowych i bazowych funkcjonalności, gdyż większa ilość takich testów może przełożyć się negatywnie na wydajność możliwego użycia pipeline’a (rozszerzenie funkcjonalności wchodzi na główną gałąź projektu po zapewnieniu, że wszystkie poprzednie testy działają). Kluczowe funkcjonalności:
     1. Tworzenie i zarządzanie wydarzeniami
     2. Wysyłanie, odbieranie, akceptowanie i odrzucanie zaproszeń
     3. Rejestracja użytkownika
     4. Logowanie użytkownika

1. Architektura oprogramowania

Projekt zostanie wykonany jako aplikacja sieciowa, gdzie:

* Część serwerowa – backend – oparty zostanie na języku Java z wykorzystaniem frameworka Spring
* Część kliencka – frontend – napisana zostanie za pomocą JavaScript i JSX i wykorzystaniem frameworka React.js i wymaganych pomniejszych pakietów do stylizacji

1. Architektura oprogramowania  
   
2. Opis elementów składowych oprogramowania:
   1. Administrator Service określa serwis udostępniany administratorom, który służy za bramę do wykonywania wszystkich możliwych czynności na innych serwisach
   2. Login Service jest to element służący do autoryzowania użytkowników wewnątrz aplikacji
   3. Event Service zawiera główną funkcjonalność zarządzania wydarzeniami //do dopisania
   4. User Service wykorzystywany jest w celu zarządzania wszystkimi możliwymi funkcjonalnościami użytkownika
   5. Review Service dostępny jest z poziomu Eventu, gdzie oferuje funkcjonalności związanie z wyświetlaniem ocen i komentarzy, a także z poziomu User Service, za którego pomocą mogą być wystawiane takowe komentarze
   6. Invitation Service określa z poziomu Eventów możliwość wysyłania zaproszeń dla użytkowników, którzy z poziomu swojego serwisu mogą je zaakceptować lub odrzucić.
3. Alokacja wymagań w architekturze oprogramowania

W adnotacji do punktów z opisu 4.b) odwołujemy się do opisanych funkcjonalności w 3.a):

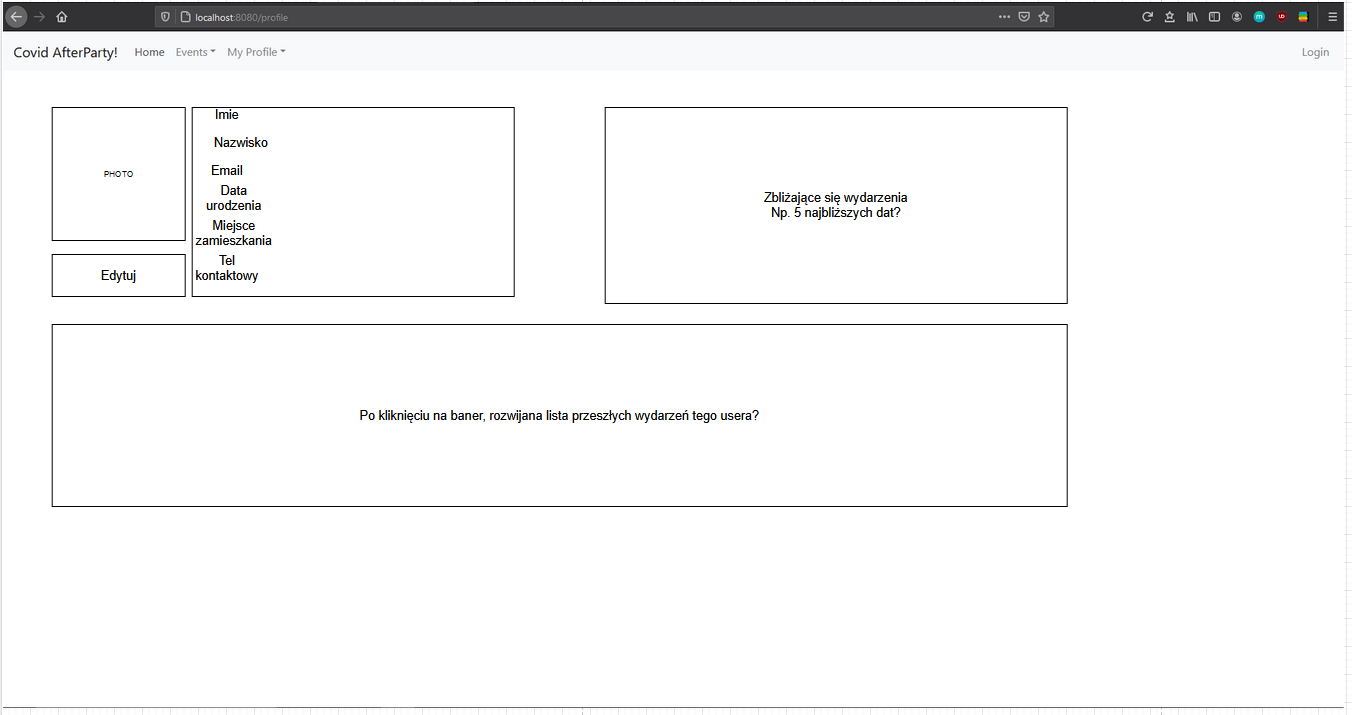
* 1. Realizuje:  
     Wszystkie funkcjonalności poza rejestracją, ze względu na pełnie uprawnień
  2. Realizuje:  
     3.a) i oraz 3.a) ii
  3. Realizuje:  
     3.a) iii oraz 3.a) viii
  4. Realizuje:  
     3. a) iv oraz 3.a) vi
  5. Realizuje:  
     3.a) vii
  6. Realizuje:  
     3.a) v oraz 3.a) viii

1. Alokacja wymagań - opisana
   1. Administrator Service
      1. Wszystkie funkcjonalności poza rejestracją, ze względu na pełnie uprawnień
   2. Login Service
      1. Rejestracja użytkownika z wymaganymi polami:
      2. Logowanie za pomocą email i hasła dla zarejestrowanego konta użytkownika
   3. Event Service
      1. Organizator:
         1. Tworzenie i zarządzanie wydarzeniami
      2. Uczestnik:
         1. Wysyłanie zapytań do organizatora wydarzenia
         2. Wyszukiwanie wydarzeń
   4. User Service
      1. Wysyłanie prośby o dołączenie do wydarzenia
   5. Review Service
      1. Zarządzanie komentarzami dla zakończonych wydarzeń, w których użytkownik wziął udział
   6. Invitation Service
      1. Organizator:
         1. Wyszukiwanie użytkowników do zaproszenia
         2. Wysyłanie zaproszeń
      2. Uczestnik:
         1. Wyświetlanie i zarządzanie otrzymanych zaproszeń poprzez widok
2. Link do repozytorium GitHub z zestawem oprogramowania:

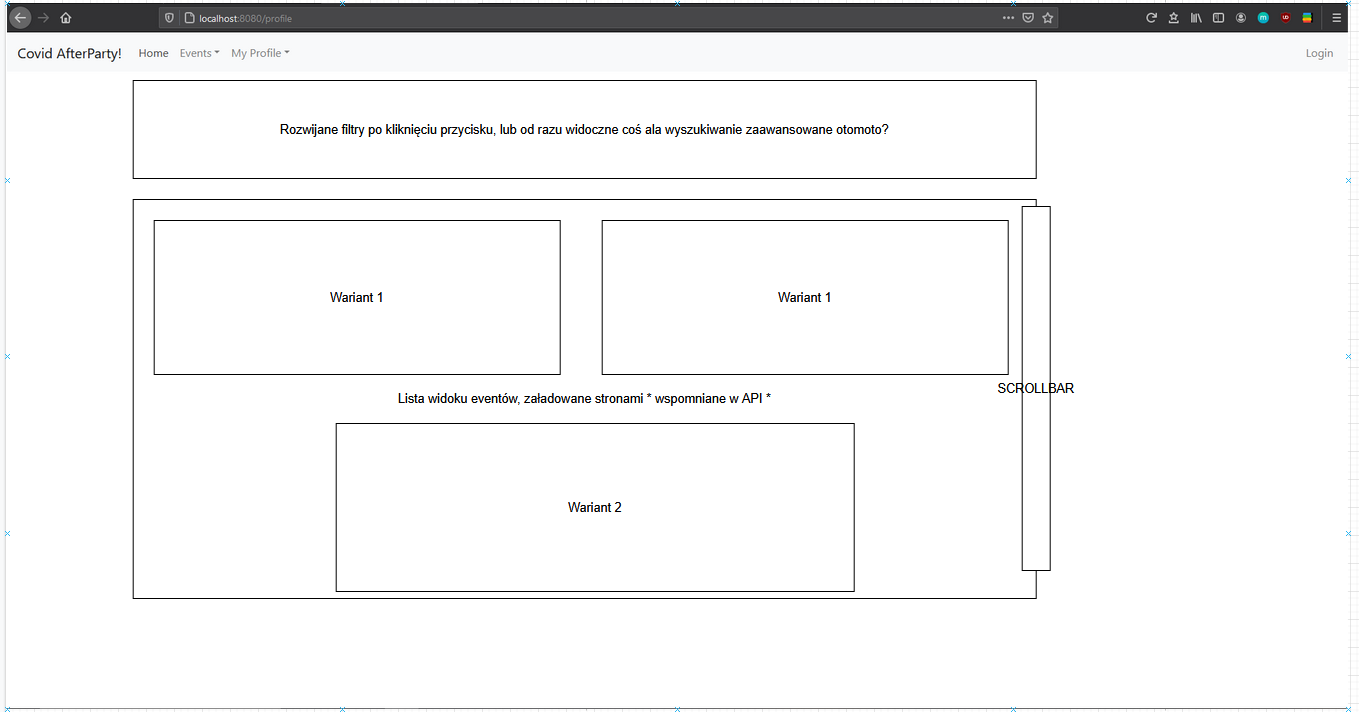
<https://github.com/MCtyskie/pai_project>

Szkice widoków:

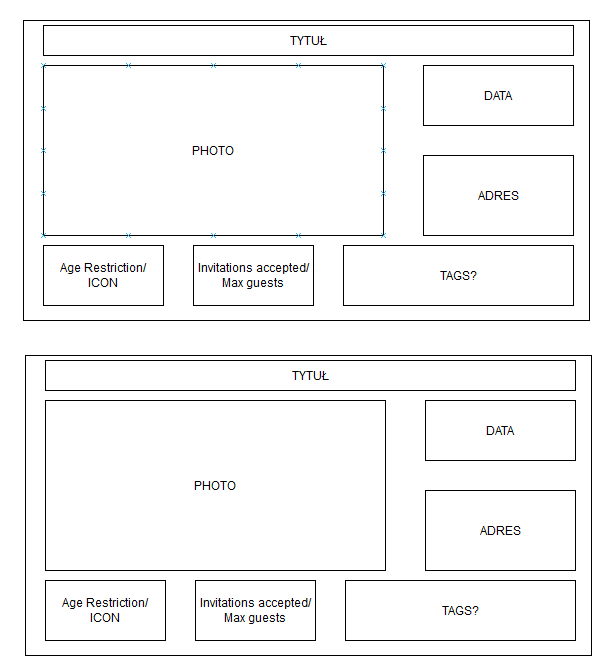
Widok panelu użytkownika \*może warto dodać tutaj przycisk/widok dodaj oceny do wydarzeń\*:



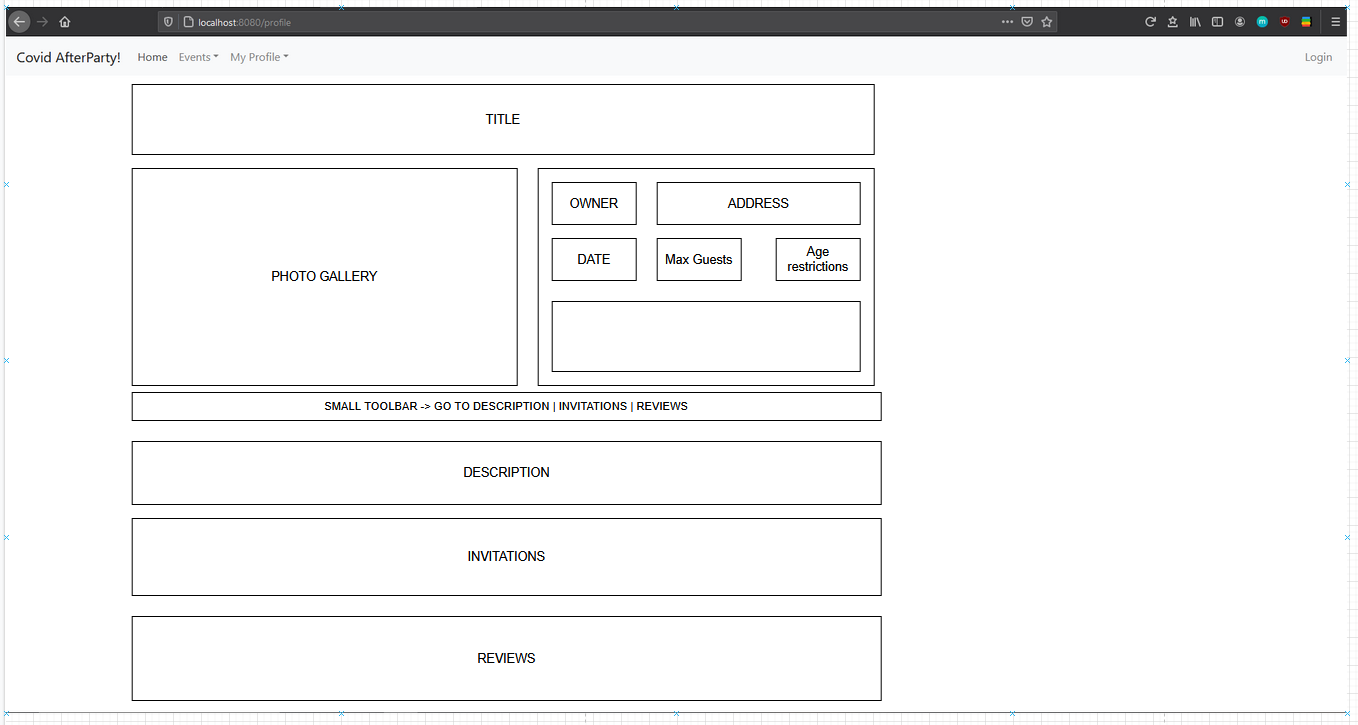
Widok eventów:



Widok szczegółów okienka eventu w powyższym widoku:



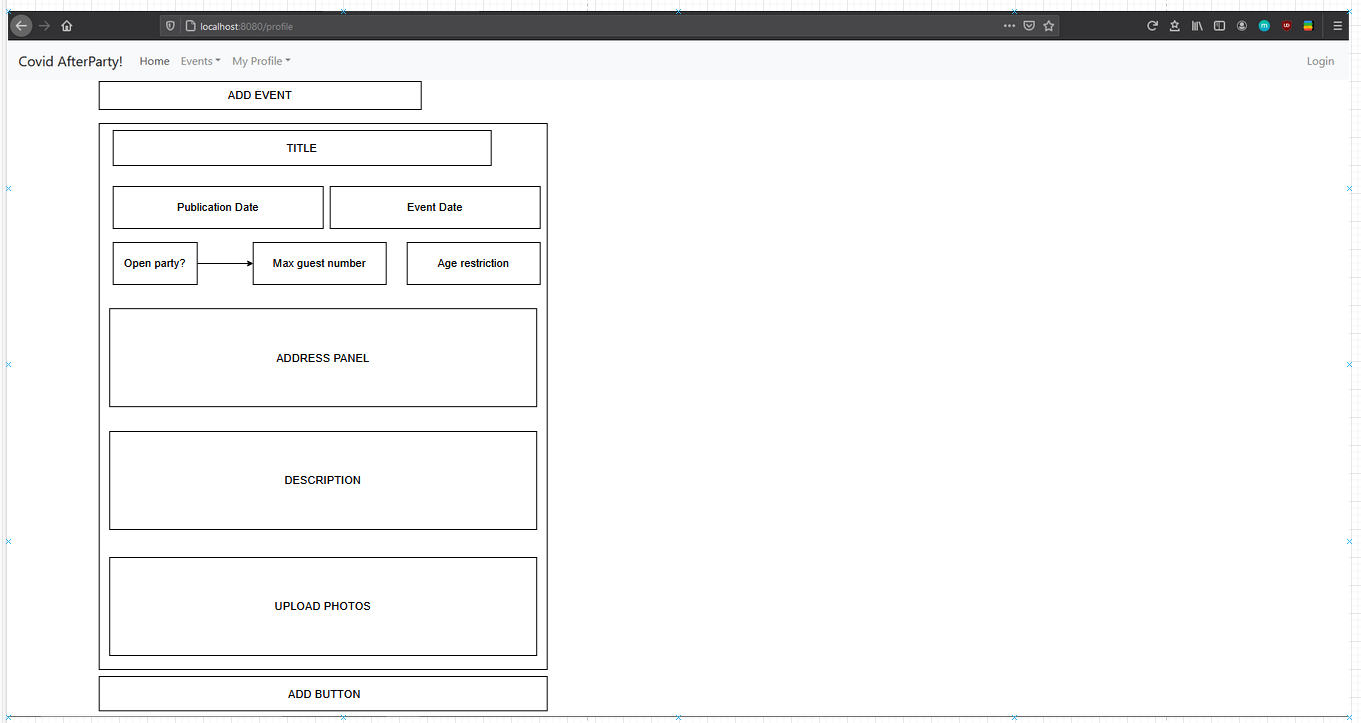
Widok pojedynczego eventu\*wszystkie dane na wzór produktu z x-komu):



Widok własnych zaproszeń usera po zalogowaniu się

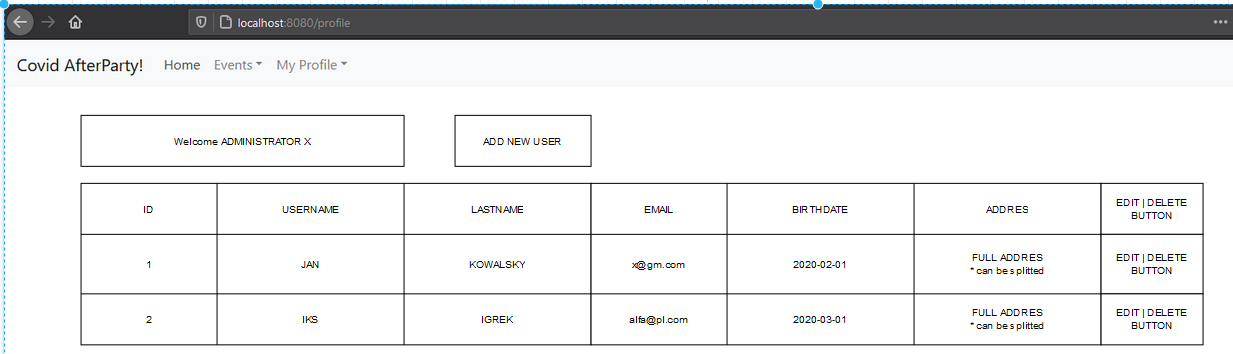
Widok zorganizowanych/organizowanych eventów

Widok dodawania eventów



Widok danych komentarzy?\*

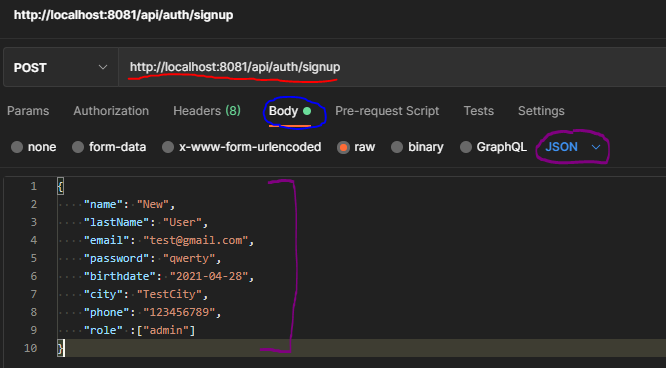
Widok admina dla podglądu wszystkich userów

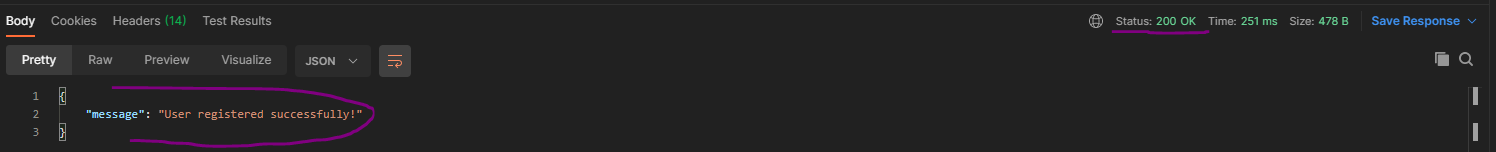


Widok admina rozszerza funkcjonalności innych widoków o CRUD-a

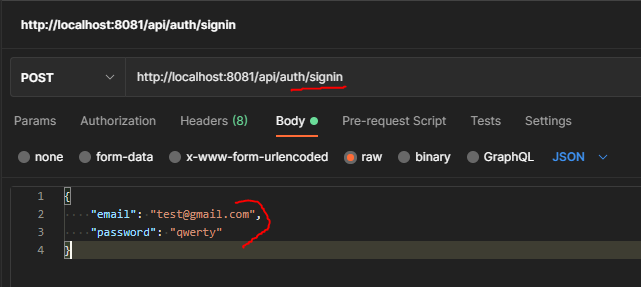
|  |  |  |
| --- | --- | --- |
| Składania zapytania | Opis | Wynik |
| POST  api/auth/signin | Przesłanie emaila i hasła w celach autoryzacji | Response 200 Ok + JSON JWT TOKEN:  {  "responseID": 4,  "email": "test@gmail.com",  "lastname": "User",  "roles": [  "ROLE\_ADMIN"  ],  "accessToken": "eyJhbGciOiJIUzUxMiJ9.  eyJzdWIiOiJ0ZXN0QGdtYWlsL  mNvbSIsImlhdCI6MTYyMDMzOTEyNywiZ  XhwIjoxNjIwNDI1NTI3fQ  .dKYJL43HnJ4l2hvMGqcnvd\_3wb2cuIb  SsaDx7ItN7t4rSurDM7UjtFOTThs  \_peEXcqVIAQpg8pcyq6ZOtyhcOQ",  "tokenType": "Bearer"  }  Response 403 Forbidden /  Response 401 Unathorized |
| POST  api/auth/signup | Przesłanie formularza danych w celach zarejestrowania użytkownika | Response 200 Ok/ 201 Created  Response 400 Bad Request |
| GET  /user/profile | Pobranie informacji użytkownika do wyświetlenia menu użytkownika i jego panelu na podstawie przesłanego tokenu JWT – rozkodowanie pierwszej części jako ID | Response 200 Ok z JSON:  {  “name”: “Jan”,  “lastName”: “Kowalsky”,  “email”: [jankow@o2.pl](mailto:jankow@o2.pl),  “birthdate”: “2000-01-01”  “city”: “Bydgoszcz”.  “phone”: “123456789”  }  Or Response 404 Not Found |
| PUT  /user/edit | Modifykacja wartości aktualnego użytkownika | Response 200 Ok/ 201 Created  Response 400 Bad Request |
| DELETE  /user/delete | Usunięcie konta użytkownika na platformie | Response 200 OK albo 204 No content? |
| GET  /user/users | Pobranie wszystkich użytkowników dostępne dla roli ADMIN | Response 200 Ok z JSON:  [  {USER1},  {USER2},…  ] |
| POST  /user/create\_user | Tworzenie nowego użytkownika z informacji posanych w Body Requestu | Response 200 OK albo 400 BAD REQUEST |
| GET  /user/get\_user\_by\_id?id=? | Pobranie pełnych informacji o użytkowniku o podanym ID – tylko dla ADMIN’a | Resposne 200 OK z JSON: [  {USER1}  ] lub 404 NOT FOUND |
| GET  /event/incoming\_events | Pobranie najbliższych 0-5 zbliżających się eventów dla użytkownika po dacie wydarzenia od najbliższej | Response 200 Ok z JSON:  {  {EVENT1},  {EVENT2},  {EVENT3},  {EVENT4},  {EVENT5}  }  Lub Reponse 204 No content z pustym JSON {} |
| GET  /event/finished\_events | Pobranie najbliższych 0-5 zakończonych eventów dla użytkownika po dacie wydarzenia od najbliższej | Response 200 Ok z JSON:  {  {EVENT1},  {EVENT2},  {EVENT3},  {EVENT4},  {EVENT5}  }  Lub Reponse 204 No content z pustym JSON {} |
| GET  /event/events | Pobranie listy wszystkich eventów  // trzeba pomyśleć o paginacji, np. 20 i po wbiciu na 15 doładowujemy kolejne 20 itd. Coś jak wall na fejsie  Zapytanie z filtrami: | JSON:  {  „title”: „meloinferno”,  „date”: „2021-04-19 19:00:00”,  „adress”: “Bydgoszcz 85-435 Puławska 13”,  \*\*albo osobno każde pole adresu, ja je złącze we froncie  „ageRestriction”: True,  „invitationsAccepted”: 23,  „maxGuests”: 100,  \*\*ALBO „occupied”: „23/100”,  „picture”: “JPG” or sth?,  \*\*opcjonalnie „tags”:  },  {  //kolejna instancja eventu  },… |
| GET  /event/details?id=? | Pobranie dwóch list dla danego eventu, listy zaproszeń i listy wystawionych komentarzy | JSON:  {  “invitations”:{Invitation1, Invitation2,…}  “reviews”:{Review1, Review2,…}  } |
| PUT  /event/edit?id=? | Aktualizacja pól danego obiektu Event przez ownera/moderatora | Response 200 OK  Response 404 Not Found |
| DELETE  /event/delete?id=? | Usunięcie danego wydarzenia i jego wpisu z listą zaproszonych | Response 200 OK  Response 404 Not Found |
| POST  /event/create\_event | Dodanie nowego obiektu imprezy | Response 201 Created  Response 400 Bad Request |
| GET  /event/events\_for\_owner | Pobranie listy eventów dla danego użytkownika, w których jest on organizatorem/moderatorem | Tak jak w /events tylko po filtracji |
| GET  /event/cities | Pobranie listy Miast w których występują Eventy | Response 200 OK z JSON: [  City1,  City2,…  ] |
| GET  /event/events\_filter | Pobranie listy eventów które spełniają kryteria podane w filtrze  Body:  JSON  {      "city":"",      "tags":"",      "date\_start":"",      "date\_end":"",      "time\_start":"",      "time\_end":""  } | Response 200 OK z JSON:  [  {EVENT1},  {EVENT2},…  ] lub 404 NOT FOUND |
| GET  /event/events\_to\_review | Pobranie listy eventów dla których użytkownik może wystawić Review | Response 200 OK z JSON:  [  {EVENT1},  {EVENT2},…  ] |
| GET  /reviews/get\_reviews | Pobranie listy wystawionych komentarzy dla danego użytkownika | Response 200 OK z JSON:  [  {REVIEW1},  { REVIEW2},…  ] |
| GET  /reviews/get\_review\_by\_id?id=? | Pobranie informacji dla Review o podanym ID | Response 200 OK z JSON:  [  {REVIEW}  ] lub 404 NOT FOUND |
| PUT  /reviews/update?id=? | Uaktualnienie istniejącego Review | Response 200 OK lub 404 NOT FOUND |
| DELETE  /reviews/delete?id=? | Usunięcie Review o podanym ID | Response 200 OK lub 404 NOT FOUND |
| GET  /reviews/review\_for\_event?id=? | Pobranie listy Review wystawionych dla Eventu o podanym ID | Response 200 OK z JSON:  [  {REVIEW1},  { REVIEW2},…  ] lub 404 NOT FOUND |
| GET  /reviews/review\_check | Sprawdzenie czy dany Review jest dostępny dla użytkownika do wystawienia | Response 200 OK lub 404 NOT FOUND |
| POST  /reviews/add\_review | Dodanie nowego komentarza dla danego id eventu od danego uzytkownika | Response 200 OK lub 400 BAD REQUEST |
| GET  /invitation/invitations | Pobranie listy zaproszeń dla użytkownika jako uczestnika | Response 200 OK z JSON:  [  {INV1}.  {INV2},…  ] |
| POST  /invitation/add | Dodanie Invitation | Response 200 OK lub 404 NOT FOUND |
| PUT  /invitation/update?id=? | Modyfikacja statusu/całościowa edycja danego zaproszenia | Response 200 OK lub 404 NOT FOUND |
| DELETE  /invitation/delete?id=? | Usunięcie podanego Invitation o podanym ID | Response 200 OK lub 404 NOT FOUND |
| POST  /invitation/join?eventID=? | Próba dołaczenia obecnie zalogowanego użytkownika do danego Eventu | Response 200 OK  Response 404 NOT FOUND  lub  Response 400 BAD REQUEST |
| POST  /invitation/invite\_to\_event?  eventID=?&invitedID=? | Utworzenie zaproszenia na dane wydarzenie danemu użytkownikowi | Response 200 OK  Response 404 NOT FOUND  lub  Response 401 UNAUTHORIZED |
| GET  /invitation/inv\_per\_event  ?eventID=? | Pobranie listy Invitation powiązanych z danym Eventem | Response 200 OK z JSON:  [  {INV1},  {INV2},…  ] |
| GET  /invitation/inv\_per\_invited  ?eventID=? | Pobranie listy Invitation powiązanych z danym zaproszonym Userem | Response 200 OK z JSON:  [  {INV1},  {INV2},…  ] |
| GET  /invitation/inv\_per\_inviter  ?eventID=? | Pobranie listy Invitation powiązanych z danym zapraszającym Userem | Response 200 OK z JSON:  [  {INV1},  {INV2},…  ] |

POSTMAN – signup & login & testing API:  
First you should create some account on DB that you can login by:



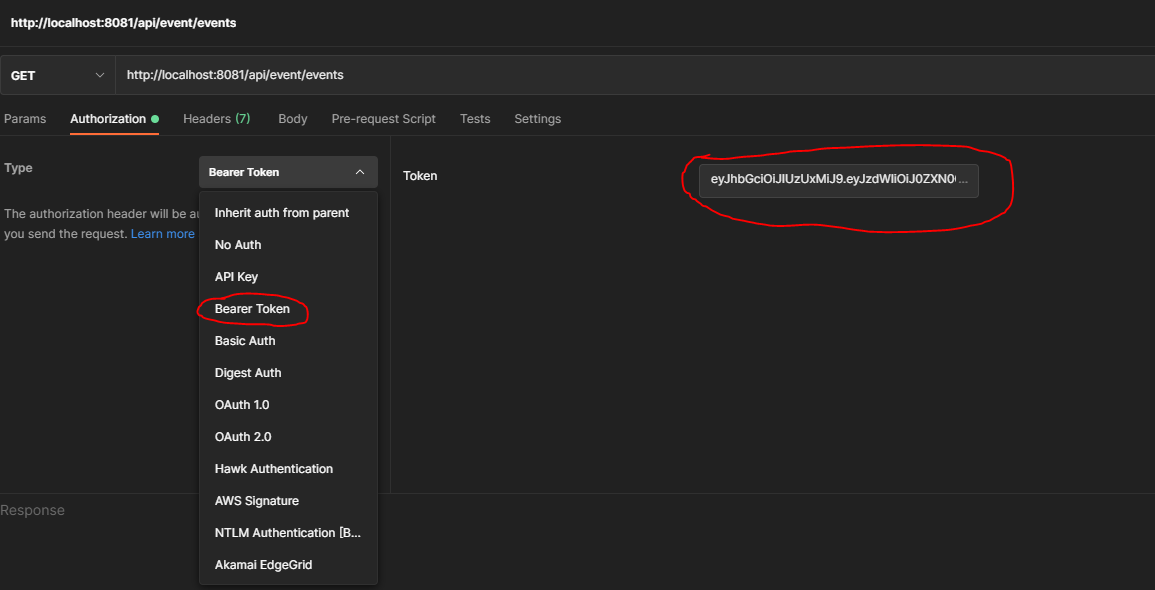


After positive response your account is created and you should POST email and password in body request on signin API endpoint:



In response you will receive access token and token type, copy access token:

  
Now go to Authorization tab on your other requests for API testing and choose accurate token type (here is a Bearer Token) and paste token that was assigned on step before:



Now you should be able to test and have access on API endpoints until your token expires.