2019

Pubweb - dokumentacja

grupa

dubiński jan grębowski łukasz kaleta joanna ogonowski aleksander oniszczuk maria szymczyk kornel walkowiak paweł

Spis treści

[1. Opis 3](#_Toc10904015)

[2. Wymagania 4](#_Toc10904016)

[2.1. Opis Metody MoSCoW 4](#_Toc10904017)

[2.2. Wymagania funkcjonalne 4](#_Toc10904018)

[2.3. Wymagania niefunkcjonalne 5](#_Toc10904019)

[3. Postanowienia ogólne 7](#_Toc10904020)

[3.1. Ogólna architektura serwisów 7](#_Toc10904021)

[4. Opis funkcjonalności projektu 8](#_Toc10904022)

[4.1. Rodzaje użytkowników 8](#_Toc10904023)

[4.2. Rejestracja I logowanie 9](#_Toc10904024)

[4.3. Wyszukiwanie pubów 9](#_Toc10904025)

[4.4. Dodawanie pubu 9](#_Toc10904026)

[4.5. Zarządzanie profilem 9](#_Toc10904027)

[4.6. Wysylanie zgłoszeń 9](#_Toc10904028)

[4.7. Panel administratora 9](#_Toc10904029)

[5. Operacje 10](#_Toc10904030)

[5.1. Serwis autoryzacyjny 10](#_Toc10904031)

[5.2. Serwis ról 10](#_Toc10904032)

[5.3. Serwis użytkowników 10](#_Toc10904033)

[5.4. Serwis znajomych 11](#_Toc10904034)

[5.5. Serwis zgód 12](#_Toc10904035)

[5.6. Serwis pubów 12](#_Toc10904036)

[5.7. Serwis recenzji 15](#_Toc10904037)

[6. Ogólny opis rozwiązań architektonicznych 16](#_Toc10904038)

[6.1. 4+1 Views 16](#_Toc10904039)

[6.1.1. Widok fizyczny rozwiązania 16](#_Toc10904040)

[6.1.2. Widok logiczny, diagram klas 17](#_Toc10904041)

[6.1.3. Widok procesów 18](#_Toc10904042)

[6.1.3.1. Diagramy aktywności 18](#_Toc10904043)

[6.1.3.2. Diagramy sekwencji 20](#_Toc10904044)

[6.1.4. Widok implementacji 22](#_Toc10904045)

[6.1.4.1. Diagram komponentów 22](#_Toc10904046)

[6.1.5. Biznesowe przypadki użycia 23](#_Toc10904047)

[7. Diagramy związków encji 24](#_Toc10904048)

[7.1. Diagram związków encji serwisów użytkowników i autoryzacji 24](#_Toc10904049)

[7.2. Diagram związków encji serwisów pubów i recenzji 25](#_Toc10904050)

[8. Modelowanie decyzji architektonicznych w postaci MAD 2.0 26](#_Toc10904051)

[9. Testowanie 28](#_Toc10904052)

[9.2 Testy interfejsów mikroserwisów 28](#_Toc10904053)

[9.2 Testy integracyjne backend 28](#_Toc10904054)

[9.3 Testy manualne 28](#_Toc10904055)

[9.3.1 Pierwsza iteracja testów funkcjonalnych 28](#_Toc10904056)

[10. Analiza zgodności z RODO 30](#_Toc10904057)

[10.1. Definicje wykorzystywanych pojęć 30](#_Toc10904058)

[10.2. Prezentacja wymagań i rozwiązań 31](#_Toc10904059)

[11. Zagadnienia bezpieczeństwa 35](#_Toc10904060)

[11.1. Dostęp do danych 35](#_Toc10904061)

[11.2. JWT 35](#_Toc10904062)

[11.3. Przesyłanie danych ( https, szyfrowanie) 36](#_Toc10904063)

[11.4. Serwery 37](#_Toc10904064)

[11.5. Przechowywanie haseł 37](#_Toc10904065)

[11.6. Kodowanie wyjść 37](#_Toc10904066)

[11.7. Rozmiar oraz przepustowość komunikatów 37](#_Toc10904067)

[11.8. Prepared query 38](#_Toc10904068)

[12. Zagadnienia niezawodności 39](#_Toc10904069)

[12.1. Odporność na awarie 39](#_Toc10904070)

[12.2. Replikacja danych 39](#_Toc10904071)

[12.3. Usterki 39](#_Toc10904072)

# Opis

PubWeb jest serwisem internetowym przeznaczonym dla pasjonatów spędzania wolnego czasu i kulturalnego spożywania alkoholu w różnego rodzaju pubach. Jest to miejsce, gdzie odnaleźć można informacje na temat pubów z całej Polski. Serwis umożliwia wyszukania lokalu według interesujących nas cech, miejscowości i nazwy. W wyborze pomagają oceny i recenzje umieszczone przez użytkowników. PubWeb jest jednak czymś więcej niż zwykłą wyszukiwarką miejsc, pozwala bowiem na zaplanowanie miejsc do odwiedzenia i zaznaczenie tych, w których już byliśmy. Serwis oferuje również możliwość dodawania znajomych i sugerowania się ich opiniami i wyborami. Przeznaczony jest dla osób powyżej 18 roku życia.

# Wymagania

W poniższym fragmencie opisano w sposób spriorytetyzowany wymagania funkcjonalne i niefunkcjonalne systemu. Przy opisie priorytetów realizacji poszczególnych wymagań posłużono się opisaną dalej metodą MoSCoW.

## Opis Metody MoSCoW

Metoda **MoSCoW** jest techniką priorytetyzacji wykorzystywaną w analizie biznesowej i przy tworzeniu oprogramowania w celu osiągnięcia wspólnego zrozumienia pomiędzy interesariuszami co do znaczenia jakie ma dla nich dostarczenie każdego z wymagań. Wyróżnia się następujące kategorie wymagań według metody MoSCoW:

* **M** – MUST (*musi być*): Opisuje wymaganie, które *musi być* spełnione w końcowym, finalnym rozwiązaniu
* **S** – SHOULD (*powinien być*): Reprezentuje pozycję o wysokim priorytecie, która *powinna być* zawarta w rozwiązaniu, jeżeli jest to możliwe
* **C** – COULD (*może być*): Opisuje wymaganie, które jest postrzegane jako pożądane, ale niekonieczne. Zostanie ono zawarte, jeżeli pozwolą na to czas i zasoby
* **W** – WON’T (*nie będzie*): Reprezentuje wymaganie, które – za zgodą interesariuszy – *nie będzie* implementowane w danym wydaniu, ale może być rozpatrzone w przyszłości

## Wymagania funkcjonalne

MUST:

* system umożliwia utworzenie konta umożliwiającego korzystanie z usług systemu
* system umożliwia logowanie
* system umożliwia dodawanie recenzji pubów
* system umożliwia dodawanie nowych pubów
* użytkownik ma możliwość wyszukania w bazie pubów na podstawie nazw lub zadanych słów kluczowych, lub miasta

SHOULD:

* system umożliwia personalizacje profilu użytkownika
* system umożliwia komentowanie recenzji
* system umożliwia dodawanie zdjęć pubów
* system umożliwia dodanie pubu do listy do odwiedzenia
* system przedstawia rekomendacje pubów

COULD:

* system umożliwia dodawanie znajomych
* system zawiera system rekomendacji pubów
* system prowadzi dziennik aktywności użytkownika

WON’T:

* system umożliwia wyszukiwanie pubów w wybranej lokalizacja
* system wyświetla lokalizację pubów na mapie
* system umożliwia logowanie za pomocą innych platform

## Wymagania niefunkcjonalne

MUST:

* system opiera się na architekturze rozproszonej w postaci mikroserwisów
* struktura systemu umożliwia dodawanie nowych funkcjonalności i dalszy rozwój
* system musi zapewnić skalowalność
* system jest zgodny z RODO [Patrz: Analiza zgodności z RODO]
* wszyscy użytkownicy mogą zgłaszać błędy na dedykowany adres email
* system spełnia wymagania bezpieczeństwa [Patrz: Kwestie bezpieczeństwa]
* system implementuje protokół OAuth 2.0
* w systemie implementowane są następujące role różniące się poziomami uprawnień
  + niezalogowany użytkownik
  + użytkownik
  + administrator
* system powinien realizować strukturę zabezpieczeń w dostępie do jego zasobów w oparciu o System uprawnień nadawanych Rolom
* interfejs klienta otwiera się w przeglądarkach:
* Chrome 68 +
* Mozilla 61 +

SHOULD:

* Interfejs klienta jest dostosowany do urządzeń mobilnych
* System zapewnia najlepszy możliwy User Experience
* Językiem systemu jest język polski
* W aplikacja pola obowiązkowe są oznaczone w inny sposób niż pola nieobowiązkowe

COULD:

* w systemie implementowana jest oddzielna rola o poziomie uprawnień większym niż użytkownik dla użytkownika-właściciela pubu
* System cenzuruje słowa wulgarne

WON’T:

* System posiada angielską wersję językową
* System posiada dedykowaną aplikacje mobilną

# Postanowienia ogólne

## Ogólna architektura serwisów

Finalnie Mikroserwisy zaimplementowane zostały w Javie. Interfejsy serwisów opisane są w poniższym dokumencie, dodatkowo dla serwisu autoryzacyjnego, użytkowników, zgód i znajomych wygenerowane zostały specyfikacje w formacie OPENAPI - [*https://swagger.io/specification/*](https://swagger.io/specification/) . Do uwierzytelnienia w serwisach wykorzystywane są tokeny JWT. Ich działanie opisuje dokument JWT.docx.

Wyróżniamy 7 następujących serwisów:

User service – serwis udostępniający operacje i informacje o użytkownikach.

Permission service – serwis udostępniający operacje i informacje o zgodach.

Friend service – serwis udostępniający informacje i operacje dotyczące relacji pomiędzy poszczególnymi użytkownikami.

Pub service – serwis udostępniający operacje i informacje o pubach.

Rating service – serwis udostępniający informacje i operacje dotyczące ocen i recenzji pubów tworzonych przez użytkowników

Authservice – serwis pozwalający na autoryzację użytkownika

Role service – niewielki serwis służący do wystawiania informacji o roli użytkownika.

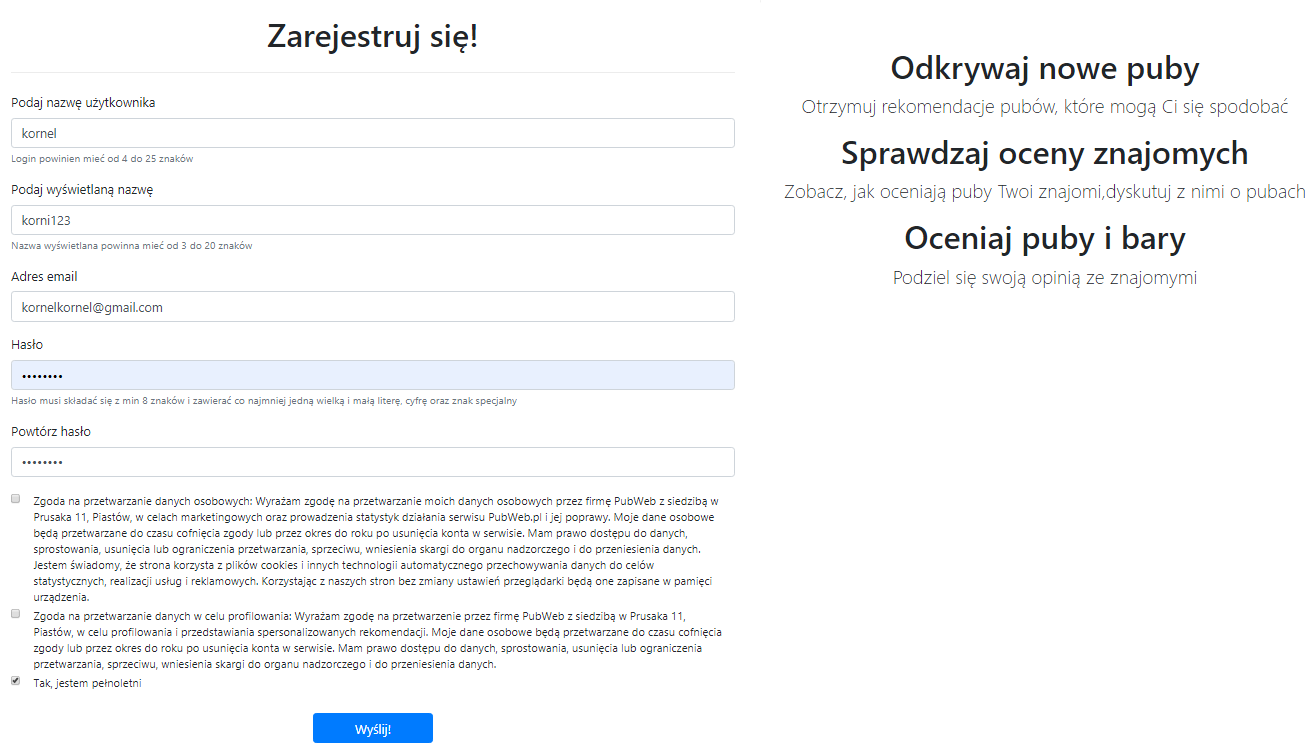
# Opis kluczowych funkcjonalności projektu

## Rodzaje użytkowników

* Anonymous – jest to użytkownik niezalogowany. Ma on możliwosć przeglądania pubów i recenzji zgromadzonych w serwisie. Nie posiada żadnych uprawnień.
* User – jest to użytkownik zalogowany, który dokonał wcześniejszej pomyślnej rejestracji w serwisie. Posiada on szereg uprawnień. Może on dodawć nowe puby (co wymaga akceptacji adminitratora) i recenzje. Wysyłać I odbierać zaproszeniado znajomych. Ma możliwość oznaczania pubów, który chciałby odwiedzić lub już w nich był. Może on również sugerować się opiniami swoich znajomych. Użytkownik może tównież wysłać zgłoszenie do administracji dotyczące uwag technicznych, pubu lub niekulturalnej recenzji.
* Adminisrator – Jest to specjalny rodzaj user’a. Ma on dostęp do panelu administracyjnego. Może on kontrolować wszystkie treści zamieszczane w serwisie I wszystkich zwykłych użytkowników. Panel administracyjny pozwala na akceptację dodanego pubu, usuwanie recenzji i użytkowników oraz przeglądanie zgłoszeń.

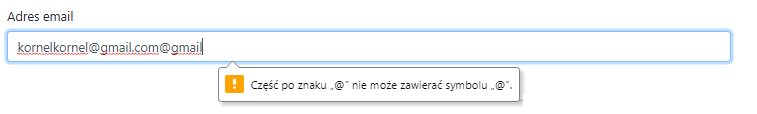
## Rejestracja i logowanie

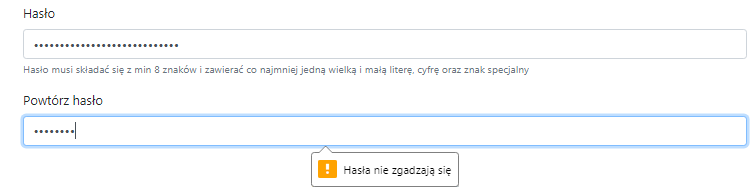
*Rys.1 ekran powitalny użytkownika anonimowego*

Do podstron logowania i rejestracji w serwisie można przejść za pomocą przycisków 1 i 2 widocznych na pasku górnym użytkownika anonimowego. Po kliknięciu przycisku 2 otwiera się podstrona z formularzem rejestracyjnym.

*Rys.2 Przycięty ekran rejestracji*

Rejestracja odbywa się poprzez uzupełnienie pół w widocznym powyżej formularzu. Wyrażanie zgody na przetwarzanie danych osobowych i danych w celu profilowania jest opcjonalne. Wymagane jest zaznaczenie bycia pełnoletnim. Po kliknięciu przycisku wyślij system sprawdza poprawność wprowadzonego adresu email i zgodność haseł. Jeśli wszystko przebiega bez przeszkód konto zostaje utworzone. W przeciwnym wypadku wyświetlony zostaje adekwatny komunikat. Przykładowo:





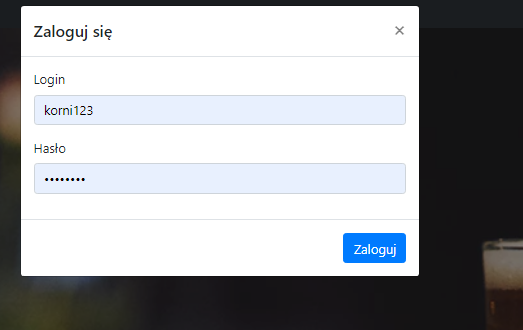
*Rys 3,4 Zgłaszanie błędów przy rejestracji*

Jeśli wszystko przebiegnie pomyślnie w prawym górnym rogu pojawi się adekwatny komunikat:



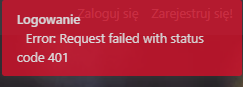
*Rys 5. Komunikat poprawnej rejestracji.*

Po dokonaniu rejestracji możemy zalogować się klikając przycisk 1 widoczny na rysunku 1. Pojawi się okienkologowania:



*Rys 6. Okno logowania*

Przy niepoprawnych danych zwrocony zostanie błąd:



*Rys 5. Komunikat błędnego logowania*

W przypadku podania poprawnych danych zostaniemy poinformowani komunikatem „Logowanie pomyślne” i przeniesieni na stronę użytkownika zalogowanego:

## Wyszukiwanie pubów i uzytkowników

## Dodawanie pubu

## Zarządzanie profilem

## Wysyłanie zgłoszeń

## Panel administratora

# Operacje

## Serwis autoryzacyjny

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje dodatkowe** |
| /auth/check | GET | Zwraca informacje o tokenie JWT | {}, token w headerze Authorization | CheckAuthResponse | Brak |  |
| /auth/signin | POST | Przyjmuje login i hasło użytkownika w celu wygenerowania tokenu JWT | LoginRequest | jwtAuthenticationResponse | Brak |  |

## Serwis ról

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje**  **dodatkowe** |
| /admin/roles | GET | Odczytuje rolę użytkownika | { } | Lista ról użytkownika | admin |  |
| /admin/users/{userId}/ roles | POST | Przypisuje rolę użytkownika | lista identyfikatorów ról | Lista ról użytkownika | admin |  |

## Serwis użytkowników

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje**  **dodatkowe** |
| /users | GET | Pobiera listę użytkowników i ich danych publicznych | {} | [GetUserResponse], tylko informacje widoczne w kontekście JWT z zapytania | Zalogowany | Stronnicowanie |
| /users/{id} | GET | Pobiera dane konkretnego użytkownika |  | GetUserResponse, tylko informacje widoczne w kontekście JWT z zapytania | Zalogowany |  |
| /users | POST | Dodaje nowego użytkownika | SignUpRequest | GetUserResponse | Brak |  |
| /users/{id} | DELETE | Usuwa dane użytkownika o podanym id | {} | {} | Kontekst własny |  |
| /users/{id}/displaySettings | PUT | Aktualizuje dane wyświetlania użytkownika | UDisplaySettings | GetUserResponse | Kontekst własny |  |
| /users/{id}/personalInformation | PUT | Aktualizuje dane osobiste użytkownika | UserPersonalInfoDto | GetUserResponse | Kontekst własny + zgoda | Zgoda o przetwarzaniu danych osobowych |
| /users/{id}/friends | GET | Pobiera dane o znajomościach użytkownika | {} | FriendshipInfo[] | Zalogowany |  |
| /users/{id}/permission | GET | Pobiera dane o zgodach (aktywnych), które zaakceptował dany użytkownik | {} | AcceptedPermission[] | Kontekst własny |  |
| /users/{id}/avatar | GET | Pobiera avatar/zdjęcie użytkownika | {} | String z awatarem w Base64 | Zalogowany |  |
| /users/{id}/avatar | POST | Dodaje avatar/zdjęcie użytkownika | AddAvatarReqeust | {} | Kontekst własny |  |
| /users/{id}/avatar | PUT | Dodaje avatar/zdjęcie użytkownika | AddAvatarReqeust | {} | Kontekst własny |  |
| /users/{id}/avatar | DELETE | Usuwa avatar/zdjęcie użytkownika |  | {} | Kontekst własny |  |

## Serwis znajomych

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje**  **Dodatkowe** |
| [/friends/request/{id}](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/deleteRequestUsingDELETE) | DELETE | Usuwa informacje o zaproszeniu do znajomych | {} |  | Tylko nadawca |  |
| friends/requests | GET | Pobiera informacje o wszystkich zaproszeniach do znajomych | {} | Lista zaproszeń do znajomych | Kontekst własny |  |
| [/friends/requests](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/sendRequestUsingPOST) | POST | Wysyła zaproszenie do znajomych | SendFriendshipReequest | Informacje o wysłanym zaporoszeniu | Kontekst własny |  |
| [/friends/requests](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/sendRequestUsingPOST)/{id} | GET | Pobiera informacje o zaproszeniu do znajomych | {} | zaproszenie do znajomych |  |  |
| [/friends/requests](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/sendRequestUsingPOST)/{id}/cancel | POST | Odrzuca zaproszenie do znajomych | {} | Informacje o odrzuconym zaproszeniu | Kontekst własny | Tylko zaproszenia przychodzące do danego użytkownika |
| [/friends/requests](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/sendRequestUsingPOST)/{id}/confirm | POST | Potwierdza dodanie do znajomych | {} | Informacje o potwierdzonym zaproszeniu | Kontekst własny | Tylko zaproszenia przychodzące do danego użytkownika |
| /requests/received | GET | Pobiera informacje o zaproszeniach wychodzących | {} | Lista zaproszeń otrzymanych | Kontekst własny |  |
| [/friends](http://34.90.111.128:8762/pubweb-user-service/swagger-ui.html?fbclid=IwAR2QPjvuyZJl843-cRSLKqvFZkxWMcHWqJkz0XVL2Yle5Dq-WMAJI7rxQpg#/operations/friends-request-controller/getSentRequestsUsingGET)/requests/sent | GET | Pobiera informacje o zaproszeniach wysłanych | {} | Lista zaproszeń wysłanych | Kontekst własny |  |

## Serwis zgód

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje**  **Dodatkowe** |
| /permissions | GET | Pobiera wszystkie informacje o zgodach | {} | permissionInfo[] | Brak |  |
| /permissions/{id} | GET | Pobiera dane jednej ze zgód | {} | PermissionInfo | Brak |  |
| /permission/accepted | GET | Pobiera dane o zaakceptowanych zgodach | {} | AcceptedPermission[] | Kontekst własny |  |
| /permission/{id}/accept | POST | Akceptuje zgodę o danym id | {} | AcceptedPermission[] – lista po akceptacji | Kontekst własny |  |
| /permission/{id}/revoke | POST | Wycofuje zgodę o danym id | {} | AcceptedPermission[] – lista po wycofaniu akceptacji | Kontekst własny |  |

## Serwis pubów

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Znaczenie** | **Typ przyjmowany** | **Typ zwracany** | **Wymagane uprawnienia** | **Informacje**  **Dodatkowe** |
| /pub | GET | lista pubow |  | PubDto | brak |  |
| /pub/{pubId} | GET | Pobiera informacje szczegółowe | {} | PubDto | brak |  |
| [/pub/{pubId}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/deleteUsingDELETE) | DELETE | Usuwa pub | {} | {} | admin |  |
| [/pub/{pubId}/picture](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/getAllPicturesUsingGET) | GET | Pobiera zdjecia w formacie Base64 | {} | Lista [Base64 photo + informacje ] | brak | Wymaga konwersji |
| [/pub/{pubId}/picture/{pictureId}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/getOnePictureUsingGET) | GET | Pobiera jedno zdjęcie | {} {} | Base64 photo + informacje | brak | Wymaga konwersji |
| [/pub/{pubId}/tags](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/addTagUsingPOST) | POST | Dodaje tagi do pubu | {} | List<TagDto>, aktualna lista tagów pubu | zalogowany |  |
| [/pub/{pubId}/tags](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/setTagsUsingPUT) | PUT | Ustawia tagi pubu | {} | List<TagDto>, aktualna lista tagów pubu | Zalogowany |  |
| [/pub/{pubId}/tags/{tagId}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-controller/deleteTagUsingDELETE) | DELETE | Usuwa informacje o tagach pubu | {} | List<TagDto>, aktualna lista tagów pubu | admin |  |
| [/request](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/addUsingPOST) | POST | Prośba o dodanie pubu | PubRegistrationRequestDto | PubRegistrationRequestInfo | Kontekst własny |  |
| [/request/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/getOneUsingGET_1) | GET | Pobranie prośby dodania pubu | {} | PubRegistrationRequestInfo | admin | Puby przychodzące do danego użytkownika |
| [/request/{requestId}/accept](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/acceptUsingPOST) | POST | Akceptacja prośby dodania pubu | {} | PubRegistrationRequestAcceptanceResponse | admin | Uwidacznia pub w serwisie |
| [/request/{requestId}/cancel](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/cancelUsingPOST) | POST | Anulowanie prośby dodania pubu | {} | PubRegistrationRequestInfo | admin |  |
| [/request/{requestId}/picture](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/getPictureUsingGET) | GET | Pobiera zdjęcie z prośby o dodanie pubu | {} | Base64PubPhoto | admin | Wymaga konwersji |
| [/request/{requestId}/picture](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/addPictureUsingPUT) | PUT | Dodaje zdjęcie do prośby o dodanie pubu | Base64PubPhoto  + info | PubPhoto info | Kontekst własny |  |
| [/request/accepted](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/getAcceptedUsingGET) | GET | Akceptacja prośby o dodanie pubu | {} | List<PubRegistrationRequestInfo> | admin | Uwidacznia wszystkie puby w serwisie |
| [/request/pending](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/pub-request-controller/getPendingUsingGET) | GET | Prośba o dodanie pubu oczekująca | {} | List<PubRegistrationRequestInfo> | admin |  |
| /request/rejected | GET | Prośba o dodanie pubu odrzucona | {} | List<PubRegistrationRequestInfo> | admin |  |
| [/suggest](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/suggest-controller/getSuggestionsUsingGET) | GET | Pobiera sugerowane do wyświetlenia puby |  | List<PubSuggestion> | Kontekst własny | Sugerowanie po tagach |
| [/tag](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/getAllUsingGET_1) | GET | Pobiera wszystkie tagi |  | List<TagDto> | Brak |  |
| [/tag](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/registerUsingPOST) | POST | Dodaje tag do puli istniejących | CreateUpdateTagRequest | TagDto | zalogowany |  |
| [/tag/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/getOneUsingGET_2) | GET | Pobiera konkretny tag | {} | TagDto | zalogowany |  |
| [/tag/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/updateTagUsingPUT) | PUT | Uaktualnia nazwę konkretnego tagu | shortTagInfo + id | TagDto | Zalogowany |  |
| [/tag/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/deleteTagUsingDELETE_1) | DELETE | Usuwa konkretny tag | {} |  | admin |  |
| /tag/{id}/cancelSubscription | POST | Usuwa zainteresowanie użytkownika danym tagiem | {} | List< TagDto> - obecnie subskrybowane tagi | Kontekst własny |  |
| [/tag/{id}/subscribe](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/subscribeUsingPOST) | POST | Dodaje zainteresowanie użytkownika danym tagiem | {} | List<TagDto> - obecnie subskrybowane tagi | Kontekst własny |  |
| [/tag/subscribed](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/tag-controller/getSubscriptionsUsingGET) | GET | Pobiera jakimi tagami zainteresowany jest uzytkownik |  | List<TagDto> - obecnie subskrybowane tagi | zalogowany |  |
| [/visits/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/getOneUsingGET_3) | GET | Dodaje datę odwiedzenia pubu | {} | PubVisitInfo | Kontekst własny |  |
| [/visits/{id}](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/deleteUsingDELETE_1) | DELETE | Usuwa wizytę | {} | {} | Kontekst własny |  |
| [/visits/visited](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/getVisitedUsingGET) | GET | Pobiera listę informacji o wizytach w pubach | userId - id usera | < PubVisitInfo> | Kontekst własny |  |
| [/visits/visited](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/markVisitedUsingPOST) | POST | Oznaczenie odwiedzenia pubu |  | PubVisitInfo | Kontekst własny |  |
| [/visits/wish](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/getWishesUsingGET) | GET | Zwraca puby, które chcemy odwiedzić |  | Lista VisitWish | Kontekst własny |  |
| [/visits/wish](http://34.90.111.128:8762/pubweb-pub-service/swagger-ui.html?fbclid=IwAR0m2aLHRBp_IihGdZvKJsW9t1__YazqeR7R7VhF130gJUGrNYoK4yqXpUc#/operations/visit-controller/addToWishListUsingPOST) | POST | Oznacza pub, jako ten który chcemy odwiedzić |  | VisitWish | Kontekst własny |  |

## Serwis recenzji

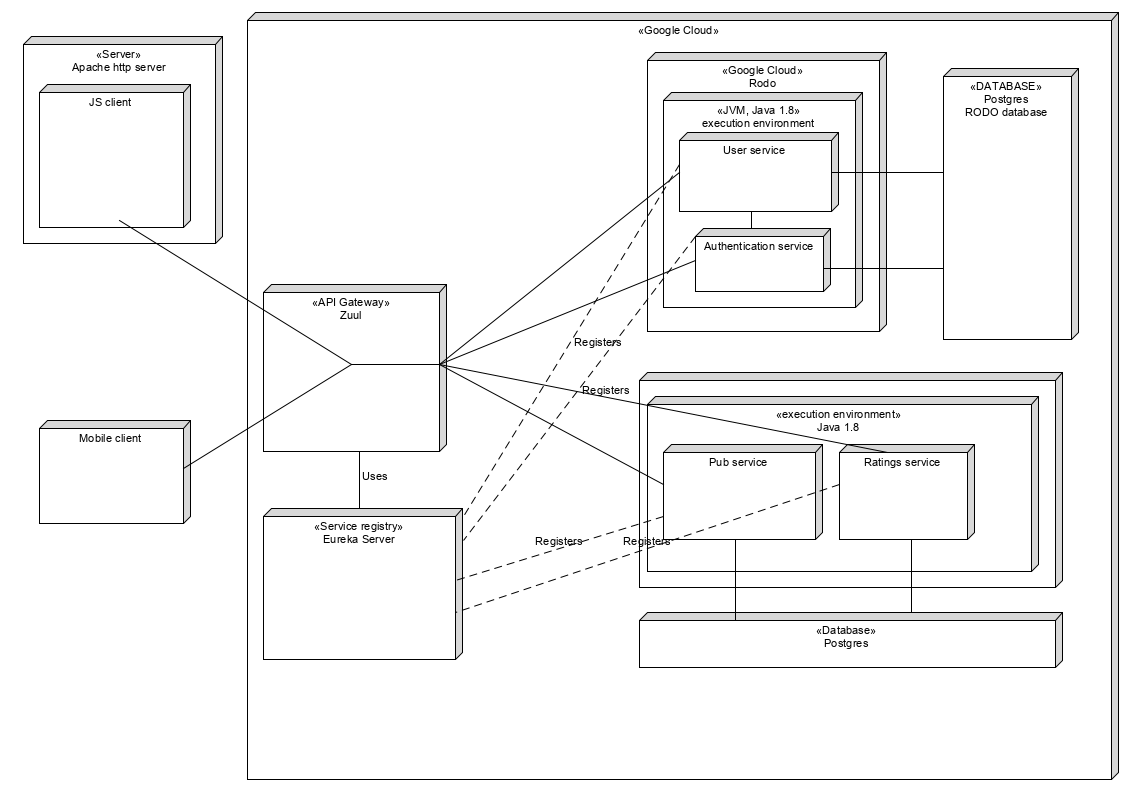
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ścieżka względna** | **Metoda HTTP** | **Działanie** | **Parametry** | **Informacje zwrotne** | **Wymagane uprawnienia** | **Informacje**  **Dodatkowe** |
| /reviews | GET | Pobiera recenzje i oceny pubów |  | Lista ReviewInfo + history + pubID | brak |  |
| /reviews | POST | Umieszcza recenzję i ocenę pubów | ReviewInfo + PubId | ReviewInfo + history + pubID | zalogowany |  |
| /reviews | DELETE | Usuwa recenzję i ocenę | reviewId | status | admin | Obejmuje recenzje pochodzące od wszystkich użytkowników |
| /reviews/{id} | GET | Pobiera konkretną recenzję i ocenę | {} | ReviewInfo + history + pubID | brak |  |
| /reviews/{id} | PUT | Zmienia konkretną recenzję i ocenę | Description + rate | ReviewInfo + history + pubID | Kontekst własny |  |
| /review/{id}/history | GET | Wyświetla recenzję i ocenę przed jej edycją | {} | Review History | Kontekst własny |  |
| /review/average | GET | Pobiera średnią ocenę pubu |  | Rating + pubId | brak |  |
| /review/top | GET | Pobiera najlepszą ocenę pubu |  | Lista [rating + pubId] | brak |  |

# Ogólny opis rozwiązań architektonicznych

W celu zrealizowania celów stawianych systemowi zdecydowaliśmy się wprowadzić architekturę opartą na mikroserwisach. Dają one możliwość konteneryzacji poszczególnych części aplikacji, a w efekcie wdrożenie w środowiskach chmurowych jest nieskomplikowane. Wzorce architektoniczne opisane są przy diagramie fizycznym.

# 4+1 Views

## Widok fizyczny rozwiązania



Zuul – jest to proxy ze stajni Netflixa. Zapewnia jeden wspólny interfejs dla wszystkich serwisów (realizacja wzorca *API gateway* nazywanego dalej bramą API). Pozwala ukryć implemntację i rzeczywiste rozmieszczenie serwisów. Klient aplikacji zna jedynie adres bramy API. Daje to możliwość zarówno łatwiejszego dostęp do API jak i oddzielenie interfejsów oraz implementacji. Możliwe jest również zdefiniowanie ustawień bezpieczeństwa (w omawianym przypadku niewykorzystywane, zarządzanie nimi odbywa się na poziomie aplikacji).

Eureka – jest to serwer *service discovery*. W przypadku mikroserwisów ręczne zmiany konfiguracji w momencie pojawienia się nowej instacji lub przeniesienia instniejącej instancji mikroserwisu byłyby żmudne. Eureka jest rejestrem serwisów, który wystawia API REST, dzięki czemu mikroserwis sam może zarejestrować swoją obecność pod wskazanym adresem (np. przy uruchomieniu).

Współpraca Zuul i Eureka: ze względu na wspólnego dostawcę Zuul i Eureka posiadają mechanizm integracji. Zuul odczytuje rejestr Eureki i trasuje zapytania do poszczególnych serwisów.

## Widok logiczny, diagram klas

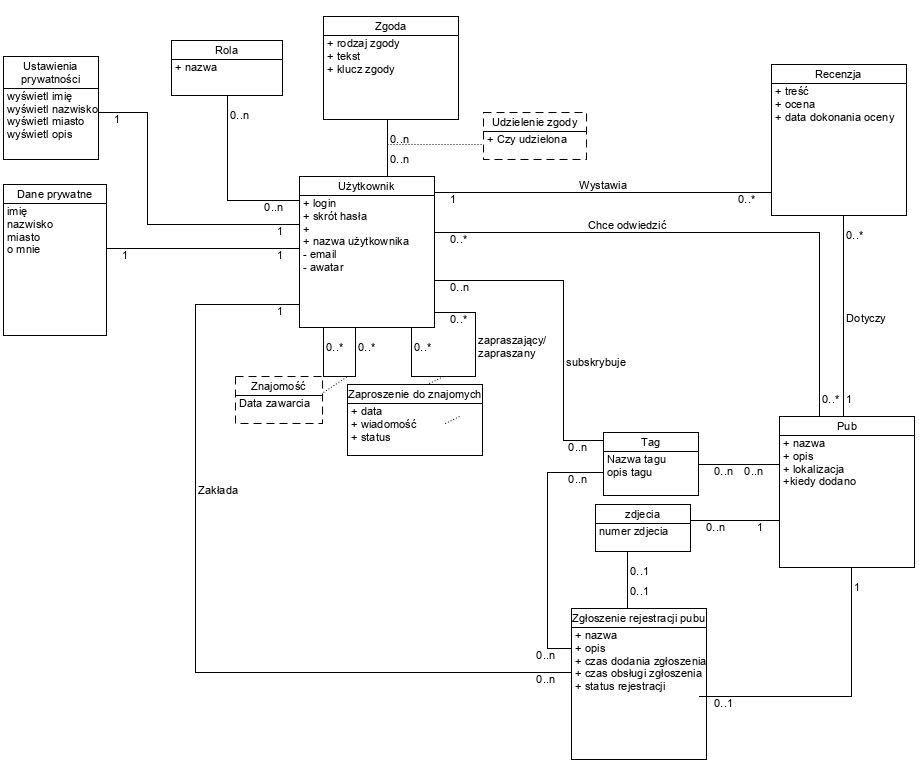
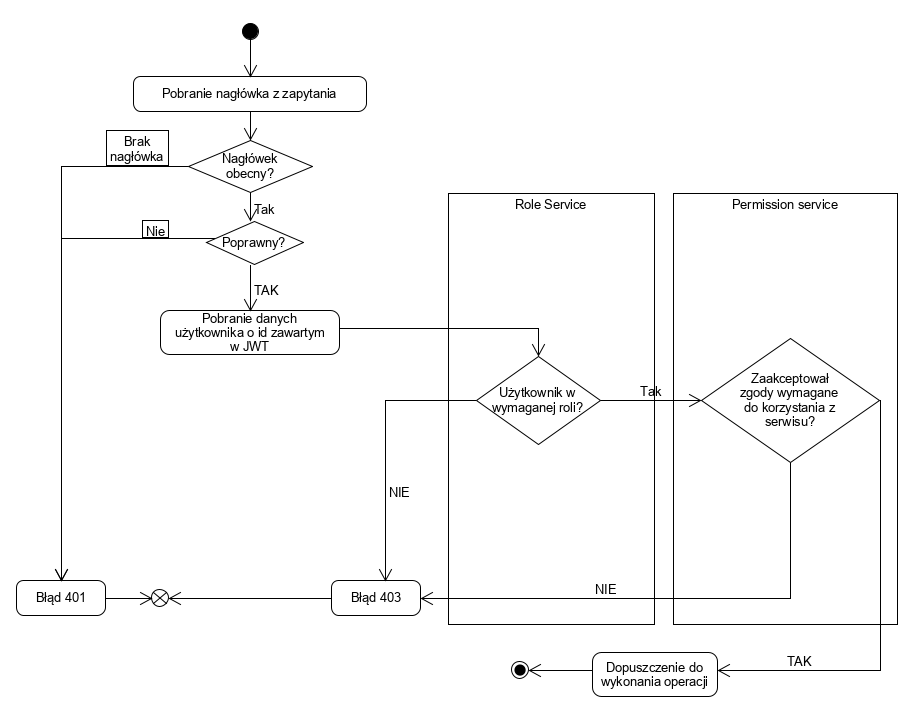


Diagram klas obrazuje logiczne zależności pomiędzy poszczególnymi komponentami systemu.

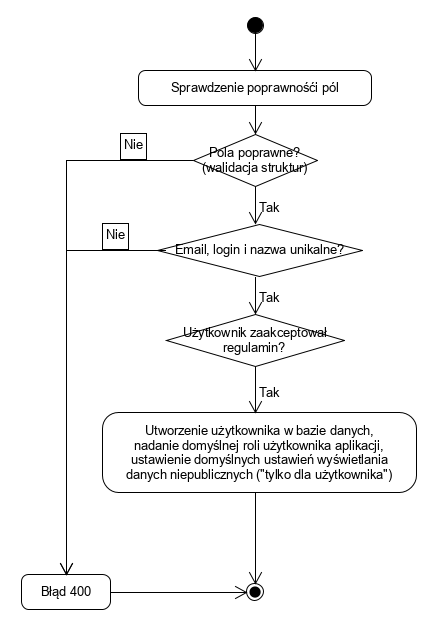
## Widok procesów

## Diagramy aktywności

* Autoryzacja operacji

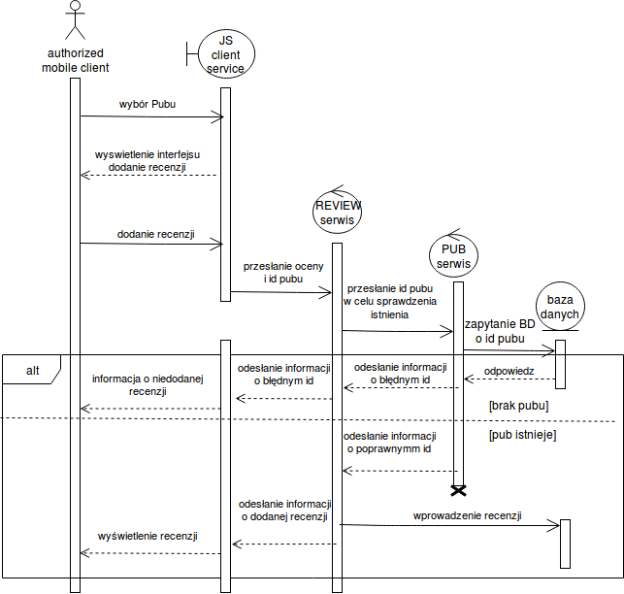


* Tworzenie konta

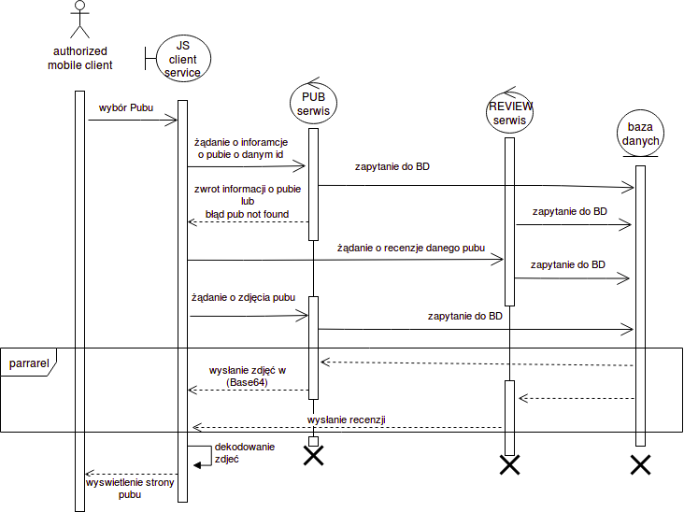


## Diagramy sekwencji

* Dodanie recenzji pubu

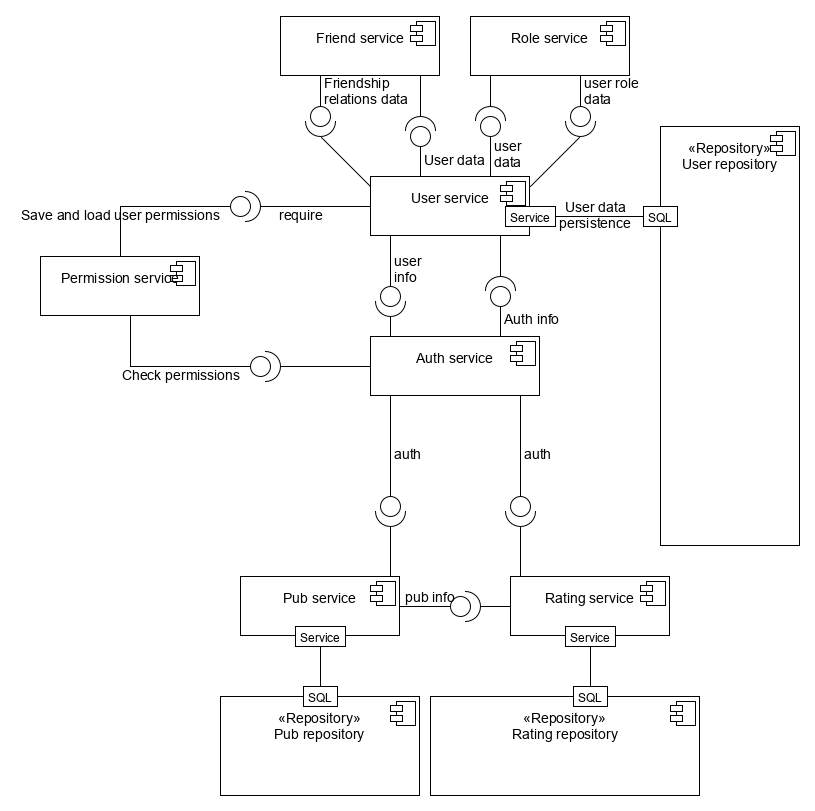


* Wyświetlanie strony pubu

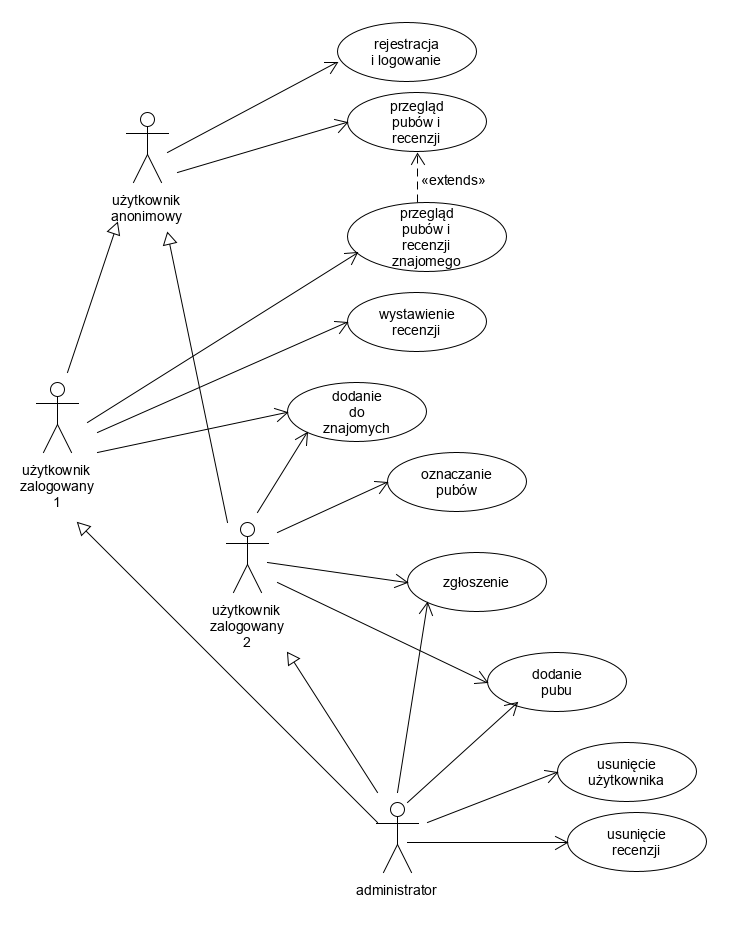


## Widok implementacji

## Diagram komponentów

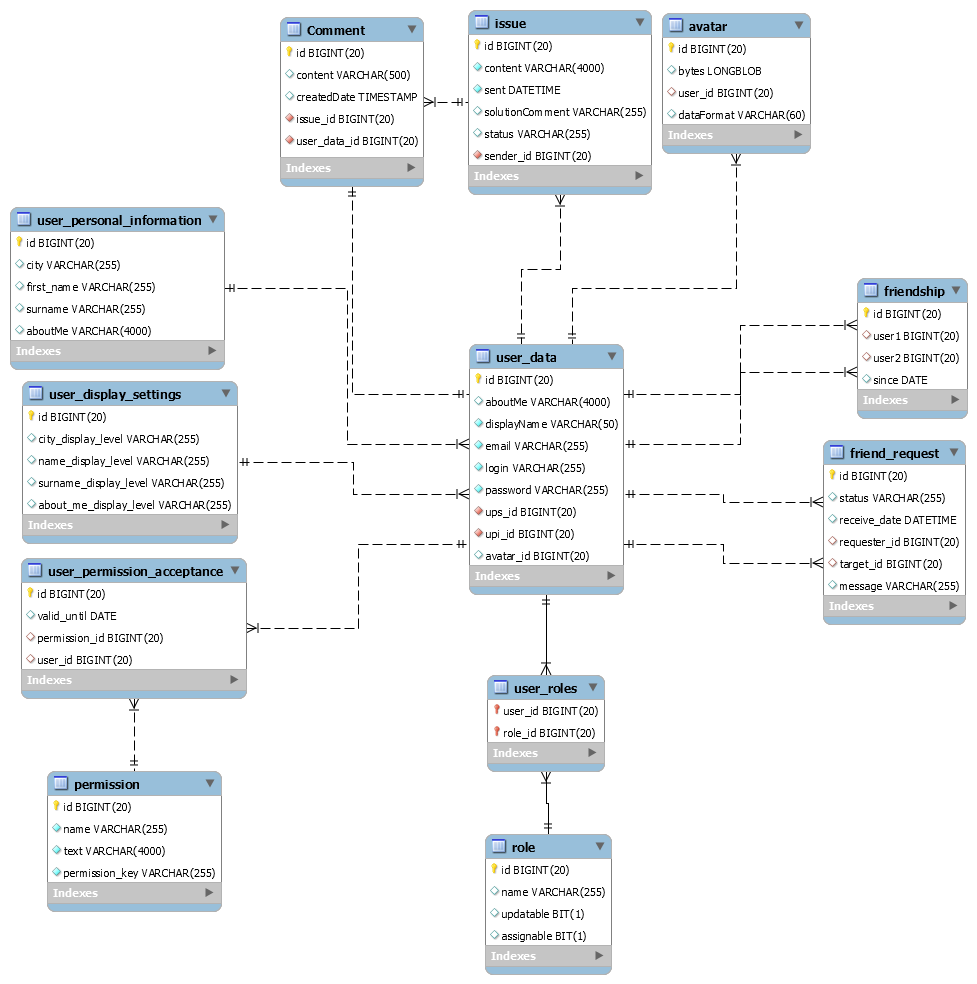


## Biznesowe przypadki użycia

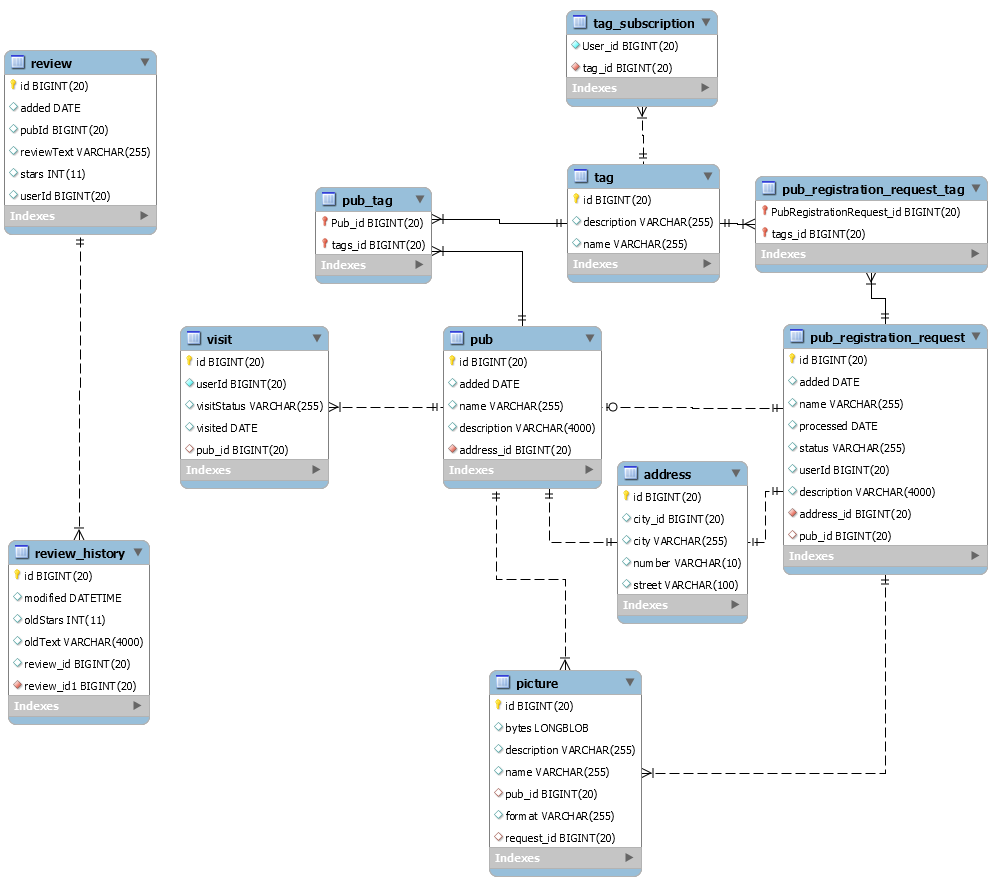


# Diagramy związków encji

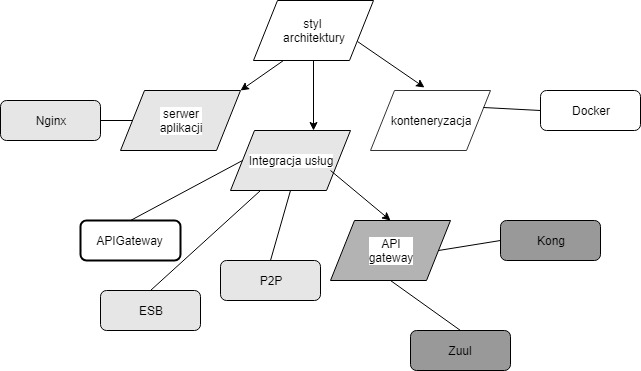
## Diagram związków encji serwisów użytkowników i autoryzacji

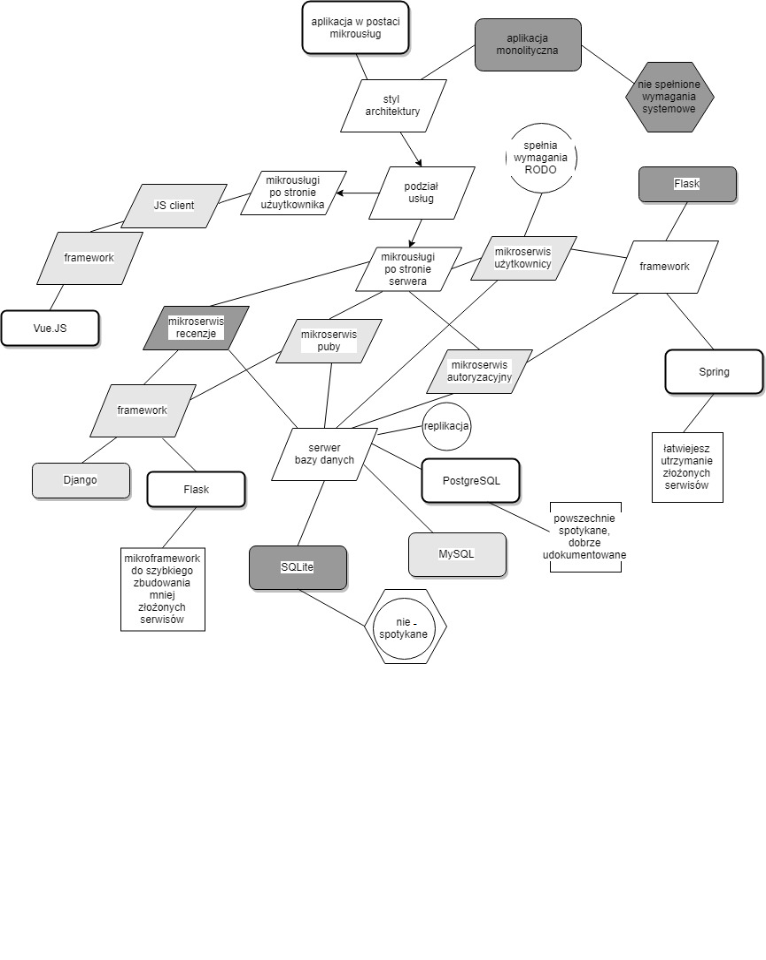


## Diagram związków encji serwisów pubów i recenzji



# Modelowanie decyzji architektonicznych w postaci MAD 2.0





# Testowanie

Podczas fazy testów aplikacji wykonywane były trzy rodzaje testów:

## 9.2 Testy interfejsów mikroserwisów

Testy interfejsów Rest Api wystawionych przez mikroserwisy przeprowadzane był w programie POSTMAN. Został zebrany zbiór zapytań wykorzystanych do sprawdzania poprawnego działania interfejsów.

## 9.2 Testy integracyjne backend

W aplikacji różne warstwy, zbudowane na różnych technologiach współpracują ze sobą. W uproszczeniu aplikację można podzielić na warstwę zapisu i odczytu danych, warstwę logiki oraz warstwę prezentacji. Testy integracyjne obejmują testy specyfikacji wyszukiwań danych w bazie. Tzn. sprawdzane jest czy zapytania budowane na poziome aplikacji są poprawne, a z bazy danych zwracane są poprawne rezultaty. Przed każdym z testów schemat bazy jest ponownie inicjalizowany, dane są przygotowywane per test.

Testy wykonywane są w frameworku Spock (<http://spockframework.org/>), który wspiera **BBD – Behaviour Driven Design. Testy w Spocku są znacznie czytelniejsze niż w Javowych frameworka typu jUnit, dodatkowo Spock posiada mocne wsparcie dla parametryzacji i wbudowane funkcje mockowania.**

**Wyniki testów pozwalają ocenić specyfikacje wyszukiwania (użytkowników, pubów, tagów) tworzone w aplikacji i czy ich współpraca z bazą danych są poprawne.**

## 9.3 Testy manualne

Testy manualne są najprostszą formą testów. Polegają na sprawdzenia funkcjonalności aplikacji poprzez jeden z wystawionych interfejsów, w tym przypadku testy funkcjonalne wykonano poprzez interfejs graficzny użytkownika. Sprawdzone zostało czy spełnione zostały wymagania określone w specyfikacji wymagań funkcjonalnych. Ponadto analizę podlegała szata graficzna interfejsu użytkownika.

### Pierwsza iteracja testów funkcjonalnych

|  |  |  |
| --- | --- | --- |
| Scenariusz testowy | wynik | uwagi |
| Rejestracja | ✓ |  |
| Logowanie | ✓ | Komunikat *request failed with status code 400* |
| Wylogowanie | ✓ |  |
| Wyszukiwanie Pubu przez użytkownika anonimowego | x |  |
| Wyszukiwanie Pubu przez użytkownika zalogowanego | ✓ |  |
| Wyszukiwanie Pubu po nazwie | ✓ |  |
| Wyszukiwanie pubów w danym mieście | ✓ |  |
| Wyszukiwanie Pubów po tagach | ✓ |  |
| Oznaczanie pubów do odwiedzenia | x |  |
| Oznaczanie pubów odwiedzonych | x |  |
| Dodawanie do znajomych | ✓ |  |
| Przyjmowanie zaproszenia do znajomych | ✓ |  |
| Wyświetlanie profilu | ✓ | Błędne wyświetlanie pubów odwiedzonych / do odwiedzenia |
| Uzupełnianie informacji o profilu | ✓ |  |
| Ustawienia zaawansowane profilu | ✓ | Brak możliwości zmiany zgód na przetwarzanie danych |
| Wysyłanie zgłoszeń do administratora | ✓ |  |
| Dodawanie pubu | ✓ |  |
| Dodawanie recenzji | ✓ |  |
| Usuwanie konta | x | Komunikat *request failed with status code 400* |
| Pobranie danych RODO | X |  |
| Odwiedzanie profilu znajomego | ✓ |  |
| Usuwanie znajomego | X |  |
| Wyświetlanie znajomych | ✓ |  |
| Wyszukiwanie znajomych | ✓ |  |
| Wyświetlanie ocen znajomych | X |  |
| Przeglądanie zgłoszeń użytkowników | ✓ |  |
| Przeglądanie recenzji z poziomu administratora | ✓ |  |
| Usuwanie recenzji | x | Komunikat *request failed with status code 400* |
| Przeglądanie użytkowników poziomu administratora | ✓ |  |
| Usuwanie użytkowników | x | Komunikat *request failed with status code 400* |
| Rozpatrywanie zgłoszeń | ✓ |  |
| Odświeżanie srony | x |  |

# Analiza zgodności z RODO

RODO jest skrótem od Rozporządzenia o Ochronie Danych Osobowych (ang*. General Data Protection Regulation* - GDPR). Jest to rozporządzenie Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych). Weszło ono w życie 25 maja 2018 r.

## Definicje wykorzystywanych pojęć

Podczas analizy zgodności systemu z RODO konieczna jest znajomość wykorzystywanych w ustawie pojęć. Jest ona niezbędna do poprawnego zrozumienia wymagań stawianych systemowi wynikających z tego rozporządzenia. W tym celu

***„dane osobowe”*** oznaczają informacje o zidentyfikowanej lub możliwej do zidentyfikowania osobie fizycznej („osobie, której dane dotyczą”); możliwa do zidentyfikowania osoba fizyczna to osoba, którą można bezpośrednio lub pośrednio zidentyfikować, w szczególności na podstawie identyfikatora takiego jak imię i nazwisko, numer identyfikacyjny, dane o lokalizacji, identyfikator internetowy lub jeden bądź kilka szczególnych czynników określających fizyczną, fizjologiczną, genetyczną, psychiczną, ekonomiczną, kulturową lub społeczną tożsamość osoby fizycznej

„***przetwarzanie***” oznacza operację lub zestaw operacji wykonywanych na danych osobowych lub zestawach danych osobowych w sposób zautomatyzowany lub niezautomatyzowany, taką jak zbieranie, utrwalanie, organizowanie, porządkowanie, przechowywanie, adaptowanie lub modyfikowanie, pobieranie, przeglądanie, wykorzystywanie, ujawnianie poprzez przesłanie, rozpowszechnianie lub innego rodzaju udostępnianie, dopasowywanie lub łączenie, ograniczanie, usuwanie lub niszczenie

**„ograniczenie przetwarzania”** oznacza oznaczenie przechowywanych danych osobowych w celu ograniczenia ich przyszłego przetwarzania

**„profilowanie”** oznacza dowolną formę zautomatyzowanego przetwarzania danych osobowych, które polega na wykorzystaniu danych osobowych do oceny niektórych czynników osobowych osoby fizycznej, w szczególności do analizy lub prognozy aspektów dotyczących efektów pracy tej osoby fizycznej, jej sytuacji ekonomicznej, zdrowia, osobistych preferencji, zainteresowań, wiarygodności, zachowania, lokalizacji lub przemieszczania się;

**„pseudonimizacja”** oznacza przetworzenie danych osobowych w taki sposób, by nie można ich było już przypisać konkretnej osobie, której dane dotyczą, bez użycia dodatkowych informacji, pod warunkiem że takie dodatkowe informacje są przechowywane osobno i są objęte środkami technicznymi i organizacyjnymi uniemożliwiającymi ich przypisanie zidentyfikowanej lub możliwej do zidentyfikowania osobie fizycznej

**„zgoda**” osoby, której dane dotyczą oznacza dobrowolne, konkretne, świadome i jednoznaczne okazanie woli, którym osoba, której dane dotyczą, w formie oświadczenia lub wyraźnego działania potwierdzającego, przyzwala na przetwarzanie dotyczących jej danych osobowych

**„naruszenie ochrony danych osobowych”** oznacza naruszenie bezpieczeństwa prowadzące do przypadkowego lub niezgodnego z prawem zniszczenia, utracenia, zmodyfikowania, nieuprawnionego ujawnienia lub nieuprawnionego dostępu do danych osobowych przesyłanych, przechowywanych lub w inny sposób przetwarzanych

## Prezentacja wymagań i rozwiązań

W tej części zawarto kolejny wymagania stawiane przed systemem, które wynikają z obowiązywania RODO oraz przyjęte rozwiązania odnoszące się do tych wymagań.

Wymaganie 1: Privacy by design

Zgodnie z Art. 25 ust.1:

Uwzględniając stan wiedzy technicznej, koszt wdrażania oraz charakter, zakres, kontekst i cele przetwarzania oraz ryzyko naruszenia praw lub wolności osób fizycznych o różnym prawdopodobieństwie wystąpienia i wadze zagrożenia wynikające z przetwarzania, administrator – zarówno przy określaniu sposobów przetwarzania, jak i w czasie samego przetwarzania – wdraża odpowiednie środki techniczne i organizacyjne, takie jak pseudonimizacja, zaprojektowane w celu skutecznej realizacji zasad ochrony danych, takich jak minimalizacja danych, oraz w celu nadania przetwarzaniu niezbędnych zabezpieczeń, tak by spełnić wymogi niniejszego rozporządzenia oraz chronić prawa osób, których dane dotyczą.

Wymaganie 2: Privacy by default

Zgodnie z Art. 25 ust.2:

Administrator wdraża odpowiednie środki techniczne i organizacyjne, aby domyślnie przetwarzane były wyłącznie te dane osobowe, które są niezbędne dla osiągnięcia każdego konkretnego celu przetwarzania. Obowiązek ten odnosi się do ilości zbieranych danych osobowych, zakresu ich przetwarzania, okresu ich przechowywania oraz ich dostępności. W szczególności środki te zapewniają, by domyślnie dane osobowe nie były udostępniane bez interwencji danej osoby nieokreślonej liczbie osób fizycznych

**Realizacja 1+2:** System nie zbiera na temat użytkowników danych nie mających zastosowania przy jego działaniu. Zminimalizowana jest ilość danych wymaganych przy tworzeniu konta użytkownika w serwisie. Podanie dodatkowe informacje dotyczących użytkownika jest możliwe poprzez edycje profilu użytkownika i jest całkowicie dobrowolne. Dane użytkownika wykorzystywane są przez system rekomendacji, który przedstawia użytkownikowi spersonalizowane propozycje pubów do odpowiedzenia, jeżeli użytkownik wyrazi na to zgodę. 0 W bazie danych zawierającej dane osobowe użytkowników zastosowany jest mechanizm pseudonimizacji.

Wymaganie 3: Brak możliwości przetwarzania, zbierania i przechowywania danych osobowych bez jednej z podstawy wskazanych w przepisach

Zgodnie z Art. 6. Podstawy do przetwarzania danych to:

* umowa - przetwarzanie jest niezbędne do wykonania umowy
* zgoda wyrażona przez osobę, której dane dotyczą
* wypełnienie obowiązku prawnego
* realizacja prawnie uzasadnionych interesów administratora danych

**Realizacja 3:** Podstawą przetwarzania danych w systemie jest zgoda użytkownika. System przetwarza dane osobowe użytkownika jedynie w wypadku wyrażenia zgody przez użytkownika. Zgoda ta uzyskana jest w sposób prawidłowy, co opisuje *Wymaganie 4*.

Wymaganie 4: Pozyskanie zgody od użytkownika w sposób prawidłowy

Zgodnie z Art.7:

1. Jeżeli przetwarzanie odbywa się na podstawie zgody, administrator musi być w stanie wykazać, że osoba, której dane dotyczą, wyraziła zgodę na przetwarzanie swoich danych osobowych.

2. Jeżeli osoba, której dane dotyczą, wyrażą zgodę w pisemnym oświadczeniu, które dotyczy także innych kwestii, zapytanie o zgodę musi zostać przedstawione w sposób pozwalający wyraźnie odróżnić je od pozostałych kwestii, w zrozumiałej i łatwo dostępnej formie, jasnym i prostym językiem. Część takiego oświadczenia osoby, której dane dotyczą, stanowiąca naruszenie niniejszego rozporządzenia nie jest wiążąca.

3. Osoba, której dane dotyczą, ma prawo w dowolnym momencie wycofać zgodę. Wycofanie zgody nie wpływa na zgodność z prawem przetwarzania, którego dokonano na podstawie zgody przed jej wycofaniem. Osoba, której dane dotyczą, jest o tym informowana, zanim wyrazi zgodę. Wycofanie zgody musi być równie łatwe jak jej wyrażenie.

4. Oceniając, czy zgodę wyrażono dobrowolnie, w jak największym stopniu uwzględnia się, czy między innymi od zgody na przetwarzanie danych nie jest uzależnione wykonanie umowy, w tym świadczenie usługi, jeśli przetwarzanie danych osobowych nie jest niezbędne do wykonania tej umowy

Oznacz to, że:

• zgoda pozyskana musi być zawsze przed rozpoczęciem przetwarzania danych

• sposoby wyrażania zgody: pisemna, ustna; klarowne, potwierdzające czynność działanie

• automatyczne zaznaczanie zgody jest zabronione

• zgoda musi dotyczyć przetwarzania konkretnych danych, w konkretnym czasie, w konkretnym celu

• obowiązuje zakaz łączenia zgód na przetwarzanie danych

• zgoda musi być wyrażona bez przymusu

• zgoda musi być poparta rzetelną klauzulą informacyjną

• nie jest wyrażeniem zgody: milczenie, okienka domyślnie zaznaczone, niepodjęcie działania itp.

**Rozwiązanie 4:** Zgoda użytkownika na przetwarzanie jego danych osobowych uzyskiwana jest podczas jego rejestracji w systemie. System wyświetla użytkownikowi zapytanie o zgodę na wykorzystanie jego danych osobowych. Zapytanie to spełnia wymienione powyżej warunki. Rezygnacja ze zgodny przebiega w prosty sposób i jest możliwa poprzez jej odznaczenie w profilu użytkownika po zalogowaniu.

Wymaganie 5: Prawo użytkownika do przeniesienia danych

RODO wprowadza prawo osoby, której dane dotyczą do przeniesienia jej danych osobowych do innego administratora. Osoba, której dane są przetwarzane może poprosić o przekazanie danych wskazanemu przez nią podmiotowi. Dane należy przekazać w ustrukturyzowanym, powszechnie używanym formacie nadającym się do odczytu maszynowego (np. XML czy CSV).

**Rozwiązanie 5:** W aplikacji webowej podane są dane kontaktowe do administracji serwisu. Prośbę o przeniesienie danych do innego administratora należy wysłać na adres mailowy wskazany w danych kontaktowych. W treści wiadomości należy umieści token (kod), który uwierzytelni użytkownika. Token ten można wygenerować w profilu użytkownika po zalogowaniu. Mechanizm ten służy zwiększeniu bezpieczeństwa.

Wymaganie 6: Prawo użytkownika do bycia zapomnianym

Prawo do bycia zapomnianym to prawo do usunięcia danych. Osoba, której dotyczą dane ma prawo zażądać od administratora danych usunięcia dotyczących jej danych, a żądanie to musi być spełnione jeśli nie występują przesłanki, które to uniemożliwiają.

Bez wątpienia należy usunąć dane m.in. jeśli:

* nie są już one niezbędne do celów, w których zostały zebrane
* zgoda na podstawie, której przetwarzałeś dane została cofnięta
* dane były przetwarzane niezgodnie z prawem
* dane osobowe zostały zebrane w związku z oferowaniem usług społeczeństwa informacyjnego (np. w ramach portalu społecznościowego)

Są jednak sytuacje, w których możliwa jest odmowa spełnienia żądania usunięcia danych. Ma to miejsce m.in. jeśli przetwarzanie tych danych jest niezbędne:

* do wywiązania się z prawnego obowiązku
* do ustalenia, dochodzenia lub obrony roszczeń
* do korzystania z prawa do wolności wypowiedzi i informacji

**Rozwiązanie 6:** W aplikacji webowej podane są dane kontaktowe do administracji serwisu. Prośbę o usunięcie danych należy wysłać na adres mailowy wskazany w danych kontaktowych. W treści wiadomości należy umieści token (kod), który uwierzytelni użytkownika. Token ten można wygenerować w profilu użytkownika po zalogowaniu. Mechanizm ten służy zwiększeniu bezpieczeństwa.

Wymaganie 7: Prawo użytkownika do żądania ograniczenia przetwarzania danych

Wymaganie 8: Prawo użytkownika do otrzymania kopii danych

Rozwiązanie 8: Wszelkie dane o użytkowniku przechowywane przez aplikacje dostępne są z poziomu interfejsów aplikacji (wyłącznie dla użytkownika). Ponadto istnieje możliwość złożenia żądania wydania kopii danych przez formularz kontaktu z administracją serwisu. Kopia danych przesyłana jest na adres mailowy użytkownika.

Wymaganie 9: Prawo użytkownika do sprzeciwu wobec profilowania

Zgodnie z art. 4 pkt 4  RODO, profilowanie to:

 “dowolna forma zautomatyzowanego przetwarzania danych osobowych, które polega na wykorzystaniu danych osobowych do oceny niektórych czynników osobowych osoby fizycznej, w szczególności do analizy lub prognozy aspektów dotyczących efektów pracy tej osoby fizycznej, jej sytuacji ekonomicznej, zdrowia, osobistych preferencji, zainteresowań, wiarygodności, zachowania, lokalizacji lub przemieszczania się”.

**Rozwiązanie 9:** Aplikacja nie wyświetla treści personalizowanych w oparciu o dane osobowe (nie uwzględnia miasta pochodzenia). Aplikacja w wersji obecnej nie wyświetla reklam, ani sugestii. Profilowanie użytkownika nie zachodzi.

Wymaganie 10: Przejrzystość języka

Zabronione jest używanie trudnych, specjalistycznych sformułowań, tak samo jak utrudnianie zapoznania się z ważnymi informacjami, np. stosowania w komunikatach tzw. metody „drobnego druczku”.

**Rozwiązanie 10:** Tekst zgód napisany jest w przystępnej i zrozumiałej formie. Przejrzystość działania serwisu jest częścią jednego z wymagań niefunkcjonalnych serwisu = „zapewnienie możliwie najlepszego User Experience.

# Zagadnienia bezpieczeństwa

## Dostęp do danych

Dostęp do danych znajdujących się w bazach danych systemu powinni otrzymać użytkownicy do nich uprawnieni, w związku z czym, w systemie jest stosowana identyfikacja użytkowników. Użytkownicy mogą uzyskać dostęp do swoich danych osobowych tylko i wyłącznie poprzez uwierzytelnienie w serwisie (login i hasło).

Aby zminimalizować ryzyko naruszenia ochrony danych osobowych dostęp do nich zostanie przydzielony jak najmniejszej liczbie członków zespołu. Wszystkie próby odczytu/modyfikacji danych wrażliwych muszą być jasno udokumentowane. Bazy danych zostaną umieszczone w kontenerach, więc port usługi bazy danych nie jest dostępny z sieci publicznej, co zapewnia nam większe bezpieczeństwo.

Usługi internetowe RESTful muszą być zabezpieczone przed wyciekami danych uwierzytelniających. Hasła, tokeny bezpieczeństwa i klucze API nie powinny pojawiać się w adresie URL, ponieważ może to prowadzić do wycieku poufnych informacji.

## JWT

JWT – Json Web Token jest otwartym standardem (RFC 7519) tworzenia tokenów dostępowych, które zawierają w sobie pewną porcję danych (payload). Siłą tokenów jest to, że mogą przenosić dowolną porcję danych i informacji o użytkowniku. Token składa się z trzech części:

Nagłówka – określa jaki algorytm szyfrowania został wykorzystany do utworzenia podpisu tokena.

Payloadu – tu znajdują się dekodowane w base64 uprawnienia/informacje przenoszone w tokenie.

Podpisu cyfrowego – podpis payloadu token-a. Podpisany kluczem prywatnym serwera (dostawcy token-a). Zabezpiecza token przed modyfikacją. Dzięki temu, że token podpisany jest kluczem prywatnym serwera autoryzacyjnego, za pomocą jego klucza prywatnego, zweryfikować można to czy wystawcą tokena faktycznie jest serwer autoryzacji.

Należy zaznaczyć iż domyślnie tokeny JWT nie posiadają funkcji pozwalających na wykluczenie przechwycenia tokena. Często w tokenach umieszcza się takie informacje jak ip, rodzaj przeglądarki (user-agent). Tokeny mają różne czasy życia, w przypadku aplikacji o wysokich wymaganiach bezpieczeństwa tokeny z pozwoleniami np. przelewu mogą być jednorazowe.

Uwaga! Użycie tokenów nie eliminuje konieczności użycia rozwiązań pozwalających zachować poufność przesyłanych informacji. Nie należy rozumieć tokenów JWT jako rozwiązań tej klasy co np. SSL.

Token przesyłany jest w nagłówku „Authorization” zapytania HTTP.

Rolą JWT w aplikacji Pubweb jest wykluczenie konieczności każdorazowego logowania loginem i hasłem oraz możliwość przekazywania informacji pomiędzy serwisami takich jak role i przywileje w spójnej formie. Użycie tokenów JWT pozwala również na ograniczenie konieczności odpytywania punktu autoryzacyjnego o dane użytkownika za każdym razem przez inne serwisy.

Standardowy schemat autoryzacji przy pomocy tokena JWT wygląda następująco:

1. Użytkownik wysyła hasło i login do serwera autoryzacyjnego

2. Serwer autoryzacyjny zwraca token JWT wraz z informacjami o tym jakie uprawnienia posiada użytkownik w aplikacji, do podpisu JWT użyty jest klucz prywatny serwera autoryzacyjnego

3. Użytkownik używa tokenu wygenerowanego przez centrum autoryzacji do wykonania operacji, w którymś z mikroserwisów.

4. Mikroserwis udziela lub nie udziela zgody na wykonanie operacji na podstawie:

4.1. Sprawdzenia za pomocą klucza publicznego serwera autoryzacji autentyczności JWT (porównanie podpisów).

4.2. W przypadku, gdy weryfikacja z 4.1. przebiegła pomyślnie serwis sprawdza czy w JWT zawarte są uprawnienia (np. Role/przywileje) wymagane do wykonania danej operacji

4.3. Mikroserwis wykonuje inne walidacje biznesowe charakterystyczne dla danej operacji

Powyższy schemat ma poniższe implikacje:

- Jeżeli dojdzie do rozszerzenia uprawnień użytkownika to po stronie użytkownika leży wygenerowanie i używanie nowego tokena o adekwatnych uprawnieniach

- Jeżeli dojdzie do ograniczenia uprawnień użytkownika to z punktu widzenia autoryzacji za pomocą JWT wejdzie ono w życie dopiero po wygaśnięciu poprzedniego tokena. Dla operacji o wysokim stopniu bezpieczeństwa potrzebny jest dodatkowy mechanizm autoryzacji (np. każdorazowe odpytywanie serwera autoryzacyjnego o uprawnienia użytkownika, którego dotyczy JWT).

## Przesyłanie danych ( https, szyfrowanie)

Komunikacja z serwerem odbywa się poprzez zastosowanie protokołu HTTPS. HTTPS zapewnia szyfrowanie asymetrycznie między klientem a serwerem, co pozwoli to na uzyskanie żądanej poufności oraz integralności transmisji danych.

Bezpieczne usługi REST powinny jedynie udostępniać punkty końcowe HTTPS. Pozwala to na ochronę poświadczeń uwierzytelniających podczas przesyłania, takich jak na przykład hasła, klucze API lub tokeny sieciowe JSON. Umożliwia to również klientom uwierzytelnianie usług oraz gwarantuje integralność przesyłanych danych.

Aby chronić przesyłane dane, należy stosować tzw. “dobre praktyki” TLS/SSL, takie jak zweryfikowane certyfikaty, odpowiednio chronione klucze prywatne, stosowanie wyłącznie bezpiecznych szyfrów. Dane prywatne muszą być szyfrowane w pamięci przy użyciu kluczy o odpowiedniej długości i w ściśle określonych warunkach dostępu, zarówno technicznych, jak i proceduralnych. Dane uwierzytelniające użytkownika muszą być hashowane niezależnie od tego, czy są one szyfrowane czy nie.

## Serwery

Dostęp do serwerów jest możliwy poprzez protokół komunikacyjny SSH. Zalogowanie do SSH jest możliwe tylko poprzez zastosowanie SSH Key. Logowanie przy pomocy hasła może zostać złamane atakiem brute force, a rozszyfrowanie SSH Key jest aktualnie niemal niemożliwe.

Całe oprogramowanie serwerów jest aktualizowane na bieżąco w celu zmniejszenia liczby podatności.

Serwery są odpowiednio skonfigurowane pod względem bezpieczeństwa poprzez zastosowanie zapór sieciowych (firewalls).

## Przechowywanie haseł

Hasła w systemie przechowywane są tylko i wyłącznie w postaci ich skrótów wykorzystując bezpieczną funkcję skrótu. Do utrzymywania skrótów wykorzystana jest bibliotek BCrypt.

Informacje niejawne dotyczące użytkowników, takie jak hasła, muszą być również chronione przy użyciu silnych, odpornych na kolizje funkcji skrótów, w celu znacznego ograniczenia ryzyka ujawnienia danych uwierzytelniających, jak również zapewnienia właściwej kontroli integralności.

## Kodowanie wyjść

Należy zapewnić, poprzez usługi internetowe, takie kodowanie informacji wysyłanych do klientów aby mogły być one wykorzystywane jako dane, a nie jako skrypty. Jest to szczególnie istotne, gdy klienci usług internetowych wykorzystują dane wyjściowe do renderowania stron HTML bezpośrednio lub pośrednio przy użyciu obiektów AJAX.

## Rozmiar oraz przepustowość komunikatów

Aplikacje internetowe, mogą być obiektem ataków DOS wykonywanych poprzez automatyczne wysyłanie do serwisów internetowych tysięcy obszernych komunikatów z protokołu SOAP. Może to doprowadzić do sparaliżowania atakowanej aplikacji, uniemożliwiając jej odpowiadanie na prawdziwe wiadomości. Aby zapobiec takiej sytuacji, rozmiar komunikatów SOAP powinien być ograniczony. Większy rozmiar (lub brak limitu w ogóle) zwiększa szanse na udany atak DoS.

Przepustowość komunikatów oznacza liczbę zapytań serwisów internetowych obsługiwanych w zadanym czasie. Należy dokonać konfiguracji zoptymalizowanej pod kątem maksymalnej przepustowości wiadomości, co pozwoli na uniknięcie sytuacji podobnych do DoS.

## Prepared query

W celu zapobiegania atakowi SQL injection zostanie wykorzystany mechanizm prepared query opierający się na wysyłaniu zapytania oraz danych do serwera bazy danych oddzielnie. Wszystkie znaki specjalne, sterujące SQL są escapowane przed wykonaniem skryptu.

# Zagadnienia niezawodności

## Odporność na awarie

Systemy rozproszone charakteryzują się wyższą odpornością na uszkodzenia, ponieważ zasoby mogą być zwielokrotnione oraz awarie zazwyczaj są częściowe. Awarie częściowe nie powodują zatrzymania całego systemu tylko pojedynczej składowej. Odporność na awarie zostanie zapewniona przez m.in. automatyczne restartowanie kontenerów Docker. Funkcja autorestart zapewnia automatyczne resetowanie kontenerów w przypadku ich zatrzymania bądź awarii. Zapewni nam to przywrócenie stanu części systemu sprzed awarii.

## Replikacja danych

Awaria niektórych systemów może wpływać na dostępność danych. Oprócz zastosowania auto restartowania kontenerów zastosowana również zostanie replikacja danych, czyli przechowywanie danych w kilku kopiach. Replikacja danych jest realizowana poprzez kopiowanie i przesyłanie danych między serwerami, i ich synchronizacji w celu zapewnienia spójności. Pozwala na skrócenie czasu dostępu do danych, oraz uniezależnienie się od czasowej niedostępności serwerów. Wadą jest konieczność aktualizowania repliki w przypadku zmiany danych źródłowych.

W systemie zastosowana jest baza danych PostgreSQL, która oferuję łatwą konfigurację replikacji danych. PostgreSQL zapewnia mechanizmy niezawodności opisane w https://www.postgresql.org/docs/9.3/wal-reliability.html.

## Usterki

Niezawodny system powinien posiadać mechanizm diagnozowania usterek i podejmowania odpowiednich działań. Działania te będą obejmować maskowanie usterek za pomocą całkowitego ukrycia i kontynuowania dalszej pracy; chwilowe lub całkowite przerwanie funkcjonowania w przypadku, gdy maskowanie jest niemożliwe do wykonania. Całkowite przerwanie jest wykonywane w ściśle określony sposób pozwalający na uniknięcie utraty lub spójności danych.