

| | |
|--|-----|
| 1. Sprawozdanie | 2 |
| 1.1 Strona tytułowa | 3 |
| 1.1.1 Konta aplikacji | 5 |
| 1.1.2 Opis funkcjonalności podstawowej i dodatkowej | 6 |
| 1.2 Sprawozdanie wstępne (15 pkt) | 7 |
| 1.2.1 1 Dziedzina wykorzystania 1 pkt | 8 |
| 1.2.2 2 Moduły funkcjonalne, aktorzy, przypadki użycia 3 pkt | 11 |
| 1.2.3 3 Struktury relacyjnej bazy danych 2 pkt | 23 |
| 1.2.4 4 Użytkownicy bazodanowi 2 pkt | 31 |
| 1.2.5 5 Identyfikacja obiektów encji 2 pkt | 34 |
| 1.2.6 6 Diagram klas encji 1 pkt | 35 |
| 1.2.7 7 Diagramy sekwenacji oraz identyfikacja transakcji bazodanowych 3 pkt | 36 |
| 1.2.8 8 Konfiguracja uwierzytelniania w aplikacji 1 pkt | 50 |
| 1.3 Sprawozdanie szczegółowe (25 pkt) | 54 |
| 1.3.1 9 Diagram UML klas komponentów EJB/CDI 4 pkt | 55 |
| 1.3.2 10 Model bezpieczeństwa komponentów EJB/CDI 5 pkt | 57 |
| 1.3.3 11 Identyfikacja transakcji aplikacyjnych 6 pkt | 64 |
| 1.3.4 12 Zgłasiane wyjątki 4 pkt | 76 |
| 1.3.5 13 Interfejs użytkownika 3 pkt | 79 |
| 1.3.6 14 Zabezpieczenia interfejsu użytkownika 3 pkt | 87 |
| 1.3.7 15 Zmiany - projekt szczegółowy | 90 |
| 1.4 Sprawozdanie końcowe (10 pkt) | 91 |
| 1.4.1 16 Zabezpieczenie transmisji HTTP 1 pkt | 92 |
| 1.4.2 17 Weryfikacja poprawności danych 2 pkt | 94 |
| 1.4.3 18 Obsługa błędów 2 pkt | 98 |
| 1.4.4 19 Wersje językowe interfejsu użytkownika 1 pkt | 103 |
| 1.4.5 20 Wykaz akcji dostępnych z interfejsu użytkownika 4 pkt | 110 |
| 1.4.6 21 Zmiany - projekt końcowy | 166 |

Sprawozdanie

SPRAWOZDANIE

Z REALIZACJI LABORATORIUM PRZEDMIOTU

SIECIOWE SYSTEMY BAZ DANYCH

EDYCJA 2022

Strona tytułowa

Projekt SSBDXX: "Temat projektu"

Członkowie grupy projektowej i funkcje projektowe:

- Szymon Depcik 229862 Szef, repozytorium, kontrola zgodności
- Adrian Wojtasik 230040 Architektura.
- Hubert Krzemiński 229928 Wdrożenie.
- Damian Wasilewski 230032 Dokumentacja.
- Marcin Muszyński 229967 Kontrola jakości.
- Michał Tosik 230027 Kontrola jakości.
- Krzysztof Martyn 229955 Baza.
- Jakub Bugara 229852 Kontrola jakości,
- Imię Nazwisko nr_indeksu login, Funkcje pełnione w projekcie: szef, repozytorium, baza, dokumentacja, architektura, wdrożenie, kontrola jakości, kontrola zgodności. Login każdego uczestnika projektu powinien być zgodny z tym, którym uczestnik posługuje się w dostępie do serwera studdev.it.p.lodz.pl
 - lista realizowanych indywidualnie przypadków użycia przez członka grupy poza modelem obsługi kont (MOK)
 - na liście należy wymienić przynajmniej dwa przypadki użycia, podając oznaczenie modułu i numer przypadku (np. MRB.2), dla każdego przypadku użycia należy wcześniej utworzyć odpowiedni komponent w systemie rejestracji zadań. Wymienione przypadki użycia muszą także być odwzorowane w macierzy decyzyjnej.

Zasoby:

- Zarządzanie projektem, system rejestracji zadań JIRA: <https://atlas.it.p.lodz.pl/jira>
- Repozytorium projektu Bitbucket: <https://atlas.it.p.lodz.pl/bitbucket>
- System gromadzenia i wymiany informacji Confluence: <https://atlas.it.p.lodz.pl/confluence>
- Witryna aplikacji : <http://studapp.it.p.lodz.pl:80XX>
- Zarządzanie serwerem aplikacyjnym: <https://studapp.it.p.lodz.pl:48XX>

Macierz decyzyjna przypadków użycia

| P.U. | Opis | Gość | Administrator | Wynajmujący | Zgłaszający |
|--------|---|------|---------------|-------------|-------------|
| MOK.1 | Zarejestruj | V | | | |
| MOK.2 | Utwórz konto | | V | | |
| MOK.3 | Zablokuj konto | | V | | |
| MOK.4 | Odblokuj konto | | V | | |
| MOK.5 | Dołącz poziom dostępu do konta | | V | | |
| MOK.6 | Odlącz poziom dostępu od konta | | V | | |
| MOK.7 | Zmień własne hasło | | V | V | V |
| MOK.8 | Zmień hasło innego użytkownika | | V | | |
| MOK.9 | Edytuj dane własnego konta | | V | V | V |
| MOK.10 | Edytuj dane konta innego użytkownika | | V | | |
| MOK.11 | Wyloguj | | V | V | V |
| MOK.12 | Aktualizacja profilu firmy | | | | V |
| MOK.13 | Weryfikacja wiarygodności konta firmy | | V | | |
| MOK.14 | Logowanie | V | | | |
| MOK.15 | Reset hasła | V | | | |
| MZ.1 | Dodawanie ofert | | | | V |
| MZ.2 | Usuwanie ofert | | V | | V |
| MZ.3 | Edycja ofert | | V | | V |
| MO.1 | Przeglądanie profili firm ogłaszających | V | V | V | V |
| MO.2 | Przeglądanie aktualnych ofert | V | V | V | V |
| MO.3 | Filtrowanie przeglądanych ofert | V | V | V | V |
| MW.1 | Możliwość wynajęcia techniki | | | V | |

| | | | | | |
|----------|--------------------------------------|--|--|---|--|
| MW.2 | Ocenianie technik | | | V | |
| SYSTEM 1 | Wysyłanie maili do weryfikacji konta | | | | |
| SYSTEM 2 | Usuwanie niepotwierdzonych kont | | | | |

Spis rozdziałów sprawozdania

Sprawozdanie wstępne

- 1 Dziedzina wykorzystania
- 2 Moduły funkcjonalne, aktorzy, przypadki użycia
- 3 Struktury relacyjnej bazy danych
- 4 Użytkownicy bazodanowi
- 5 Identyfikacja obiektów encji
- 6 Diagram klas encji
- 7 Diagramy sekwencji oraz identyfikacja transakcji bazodanowych
- 8 Konfiguracja uwierzytelniania w aplikacji

Sprawozdanie szczegółowe

- 9 Diagram UML klas komponentów EJB
 - 10 Schemat bezpieczeństwa komponentów EJB
 - 11 Identyfikacja transakcji aplikacyjnych
 - 12 Zgłaszone wyjątki
 - 13 Interfejs użytkownika
 - 14 Zabezpieczenia interfejsu użytkownika
 - 15 Zmiany - projekt szczegółowy
 - 16 Zabezpieczenie transmisji HTTP
 - 17 Weryfikacja poprawności danych
 - 18 Obsługa błędów
 - 19 Wersje językowe interfejsu użytkownika
 - 20 Opis przypadków użycia MVCv2
 - 21 Zmiany - projekt końcowy
- ...

Konta aplikacji

Konto Administratora

- Login: admin
- Hasło: ssbd01
- Poziom dostępu: Admin

Konto Zgłaszającego

- Login: serviceProvider
- Hasło: ssbd01
- Poziom dostępu: ServiceProvider

Konto Wynajmującego

- Login: renter
- Hasło: ssbd01
- Poziom dostępu: Renter

Opis funkcjonalności podstawowej i dodatkowej

Aplikacja będzie służyć do ogłaszaania usług oraz wynajmowania technik estradowych na dany okres czasu. Dostępne będą 2 rodzaje kont dla użytkowników aplikacji. Pierwszy rodzaj konta to konto osoby/firmy która chce ogłosić swoje usługi. Po weryfikacji oraz potwierdzeniu przez administrację, że dana osoba prowadzi działalność wynajmu technik estradowych.

użytkownik tego konta będzie mógł wystawić ogłoszenie określając termin w jakim jest dostępny, stawkę, oraz inne wartości związane z estradą.

Drugi rodzaj konta to konto klienta który chce wynająć technikę estradową na dany termin. Po rejestracji i zalogowaniu konta, klient może przeglądać oferty wystawione w aplikacji i zdecydować się na wybór jednej z przedstawionych ofert.

Sprawozdanie wstępne (15 pkt)

Sprawozdanie wstępne

1 Dziedzina wykorzystania 1 pkt

Przeznaczenie aplikacji

Aplikacja będzie służyć do wynajmowania technik estradowych na dany czas, z podziałem na konta dla klientów i firm oferujących wynajem technik estradowych na dany dzień.

Funkcjonalności

- konta użytkowników z różnymi poziomami dostępu, tj. rejestracja konta, logowanie i wylogowanie, zmiana hasła, modyfikacja poziomów dostępu konta administratora, blokowanie/odblokowanie kont przez administratora, modyfikacja danych konta/profilu;
- weryfikacja kont firm przez administratorów;
- dodawanie, modyfikacja i usuwanie ofert firm ogłaszających;
- przeglądanie profili firm ogłaszających;
- przeglądanie i filtrowanie aktualnych ofert;
- wynajmowanie ofert przez klientów i późniejsza ich ocena.

Reguły biznesowe

- jedna oferta może zostać zarezerwowana przez tylko jednego klienta,
- jeden użytkownik może mieć tylko jedno konto założone na dany adres e-mail,
- każde konto musi mieć potwierdzony e-mail, żeby było aktywowane,
- dane firmy ogłaszającej muszą zostać zweryfikowane przez administratora, żeby konto było aktywne,
- administrator może modyfikować dane kont i ofert oraz resetować hasła,
- administrator ma dostęp do widoków konta ogłaszającego i konta wynajmującego,
- konto ogłaszającego i wynajmującego mają tylko jeden, odpowiadający im widok,
- użytkownicy mogą modyfikować dane swoich kont,
- użytkownicy kont ogłaszających mogą dodawać nowe oferty, a także modyfikować i usuwać je, jeśli nie zostały zarezerwowane.

Konkurencja

Konkurencja będzie zachodzić dla ofert; kilku użytkowników mogą w tym samym momencie chcieć wynająć konkretną ofertę na dany dzień.

Charakterystyka aplikacji (OLTP/OLAP) z uzasadnieniem

Aplikacja posiada charakterystykę OLTP ze względu na potrzebę stworzenia systemu wymagającego dostępności najnowszej wersji danych w czasie rzeczywistym, który zapewni spójność wprowadzanych danych, możliwość efektywnego obsługi konkurencji o zasoby (oferty), co może zostać zrealizowane przy pomocy przetwarzania transakcyjnego.

Stanowość/bezstanowość aplikacji w warstwie logiki biznesowej, format wymiany danych pomiędzy warstwą prezentacji a warstwą logiki biznesowej

Aplikacja w warstwie logiki biznesowej będzie stanowa. Formatem wymiany danych jest DTO.

Stos technologiczny warstwy logiki biznesowej

- JDK 11
- Jakarta EE 8.0
- JUnit 5.8.1
- Eclipse Persistence 2.7.7
- Project Lombok 1.18.16
- Maven 3.3.2
- Bitbucket v7.6.0 (repozytorium kodu)
- IntelliJ 2021.3.3
- JDBC PostgreSQL 42.2.1

Słownik nazw obiektów w modelu danych

| Obiekt | Opis | Cechy | Cykl życia | Zależności | Ograniczenia |
|--------|------|-------|------------|------------|--------------|
|--------|------|-------|------------|------------|--------------|

| | | | | | |
|------------------------|--|---|---|-------------------------------------|---|
| Account | Zawiera informacje o koncie użytkownika, a także status aktywności i potwierdzenia konta. | id, login, password, active, confirmed, accessLevel, name, surname, email, phoneNumber, version | Tworzony w momencie utworzenia konta. | AccessLevel | Brak ograniczenia liczności. Może być powiązany z każdym z poziomów dostępu (przynajmniej 1). |
| AccessLevels | Reprezentuje poziom dostępu, jaki może być przypisany do konta. | id, accessLevel, active, account, version | Tworzony w momencie utworzenia konta. | Account | Maksymalnie 3 (Admin, Renter, ServiceProvider) unikalne dla konta. Powiązany z jednym kontem. |
| AdminDetails | Reprezentuje poziom dostępu Admin. | id | Tworzony w momencie dodania poziomu dostępu Admin do konta. | AccessLevel | Brak ograniczenia liczności. Powiązany z jednym poziomem dostępu. |
| RenterDetails | Reprezentuje poziom dostępu Renter. Zawiera też dodatkową informację o koncie użytkownika typu klient. | id, userName | Tworzony w momencie dodania poziomu dostępu Renter do konta. Usuwany w momencie usunięcia poziomu dostępu Renter z konta. | AccessLevel, Rate, UserOffer | Brak ograniczenia liczności. Powiązany z jednym poziomem dostępu. Może być powiązany z wieloma ocenami. Może być powiązany z wieloma rezerwacjami użytkownika. |
| ServiceProviderDetails | Reprezentuje poziom dostępu ServiceProvider. Zawiera też dodatkowe informacje o koncie użytkownika typu ogłaszający. | id, serviceName, NIP, address, description, logoUrl | Tworzony w momencie dodania poziomu dostępu ServiceProvider do konta. Usuwany w momencie usunięcia poziomu dostępu ServiceProvider z konta. | AccessLevel, Rate | Brak ograniczenia liczności. Powiązany z jednym poziomem dostępu. Może być powiązany z wieloma ocenami. |
| Rate | Zawiera ocenę wystawioną ogłoszeniem przez klienta. | id, renterId, serviceProviderId, value, version | Tworzony w momencie dodania oceny ogłoszającego. | RenterDetail, ServiceProviderDetail | Brak ograniczenia liczności. Powiązany z jednym klientem. Powiązany z jednym ogłoszającym. |
| UserOffer | Przyporządkowuje oferty użytkownikom, którzy je zarezerwowali. | id, renterId, offerDateId, version | Tworzony w momencie rezerwacji oferty przez klienta. | OfferDate, RenterDetail | Brak ograniczenia liczności. Powiązany z jedną rezerwacją. Powiązany z jednym klientem. |

| | | | | | |
|-------------------|---|--|---|----------------------------------|--|
| OfferDate | Zawiera informacje o utworzonej przez użytkownika rezerwacji. | id, offerId, date, version | Tworzony w momencie rezerwacji oferty przez klienta. | UserOffer, Offer | Brak ograniczenia liczności. Powiązany z jedną rezerwacją użytkownika. Powiązany z jedną ofertą. |
| Offer | Zawiera informacje o ofercie utworzonej przez ogłaszającego. | id, serviceProviderId, description, price, version | Tworzony w momencie dodania oferty przez ogłaszającego, usuwany w momencie usunięcia oferty. | OfferDate, ServiceProviderDetail | Brak ograniczenia liczności. Może być powiązany z wieloma rezerwacjami. Powiązany z jednym ogłaszającym. |
| VerificationToken | Token przypisany do konta, umożliwiający na potwierdzanie rejestracji oraz resetu hasła | id, generatedToken, creationDate, expiryDate | Tworzony w momencie wysłania maila do użytkownika z potwierdzeniem rejestracji oraz resetu hasła, usuwany w momencie potwierdzenia maila przez użytkownika lub gdy token wygaśnie | Account | Brak ograniczenia liczności. Powiązany z jednym kontem. |

Słownik pozostałych pojęć i obiektów

- technika estradowa - sceny, instalacje multimedialne, profesjonalne nagłośnienie czy komputerowo sterowane systemy oświetlenia,
- klient - użytkownik, który jest zainteresowany zarezerwowaniem technik estradowych w systemie,
- ogłaszający, firma ogłaszaająca - firma oferująca wynajem technik estradowych.

2 Moduły funkcjonalne, aktorzy, przypadki użycia 3 pkt

Lista modułów funkcjonalnych

- MOK (Moduł Obsługi Kont)
- MZ (Moduł Zgłaszającego)
- MO (Moduł Ogłoszeń)
- MW (Moduł Wynajmującego)

Lista poziomów dostępu (aktorów)

- **Administrator** - użytkownicy z tym poziomem dostępu mogą zarządzać, edytować i weryfikować konta w systemie. Pierwsze konto z poziomem dostępu administratora zostanie utworzone przed wdrożeniem aplikacji. Kolejne konta użytkowników z tym poziomem dostępu będą tworzone i nadawane będą im uprawnienia przez innego administratora, jeśli zajdzie potrzeba tworzenia nowych kont administracyjnych.
- **Wynajmujący** - konto z poziomem dostępu wynajmującego pełni rolę klientów w modelu biznesowym, posiadają możliwość wynajęcia usługi technik estradowych.
W celu założenia konta z poziomem dostępu wynajmującego należy przejść przez standardowy dla aplikacji proces rejestracji, podczas którego należy wybrać przeznaczenie konta - w tym przypadku będzie to skorzystanie z dostępnych usług.
- **Zgłaszający** - rolą poziomu zgłaszającego jest rola usługodawcy, użytkownicy mają możliwość wystawienia swoich ofert usług technik estradowych. Założenie funkcjonującego konta zgłaszającego przebiega podobnie, jak w przypadku wynajmującego, z różnicą w wyborze przeznaczenia konta - tutaj należy wybrać opcję, która jasno mówi o tym, że chcemy zaoferować swoje usługi. Wybranie takiej opcji sprawi, że będziemy musieli zweryfikować wiarygodność swojej firmy poprzez podanie koniecznych do tego informacji. Proces rejestracji w tym momencie zostanie zakończony, jednakże aby w pełni korzystać z usług oferowanych przez aplikację, będziemy musieli poczekać na zweryfikowanie naszego konta przez administratora.
- **Gość** - gościami są osoby niezarejestrowane, potencjalni użytkownicy aplikacji.
Dla uzyskania poziomu dostępu gościa, nie jest konieczne zakładanie konta. Bezpośrednio po przejściu na stronę aplikacji, uzyskujemy ten przywilej, który pozwala nam na przeglądanie oferty oraz rejestracji
- W aplikacji nie może dojść do przypadku, w którym użytkownik bez praw administratora będzie pełnił rolę zarówno zgłaszającego i wynajmującego. Wykluczony jest również przypadek, w którym gość posiada poziom dostępu większy, niż pozwala mu na jego rolę.

Diagramy przypadków użycia

Diagram użycia dla Gościa

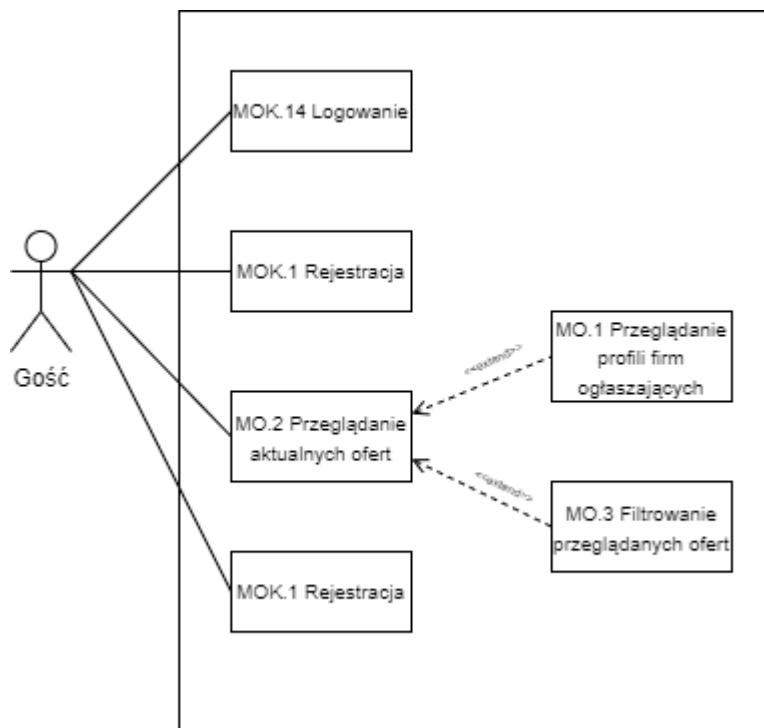


Diagram użycia dla Administratora

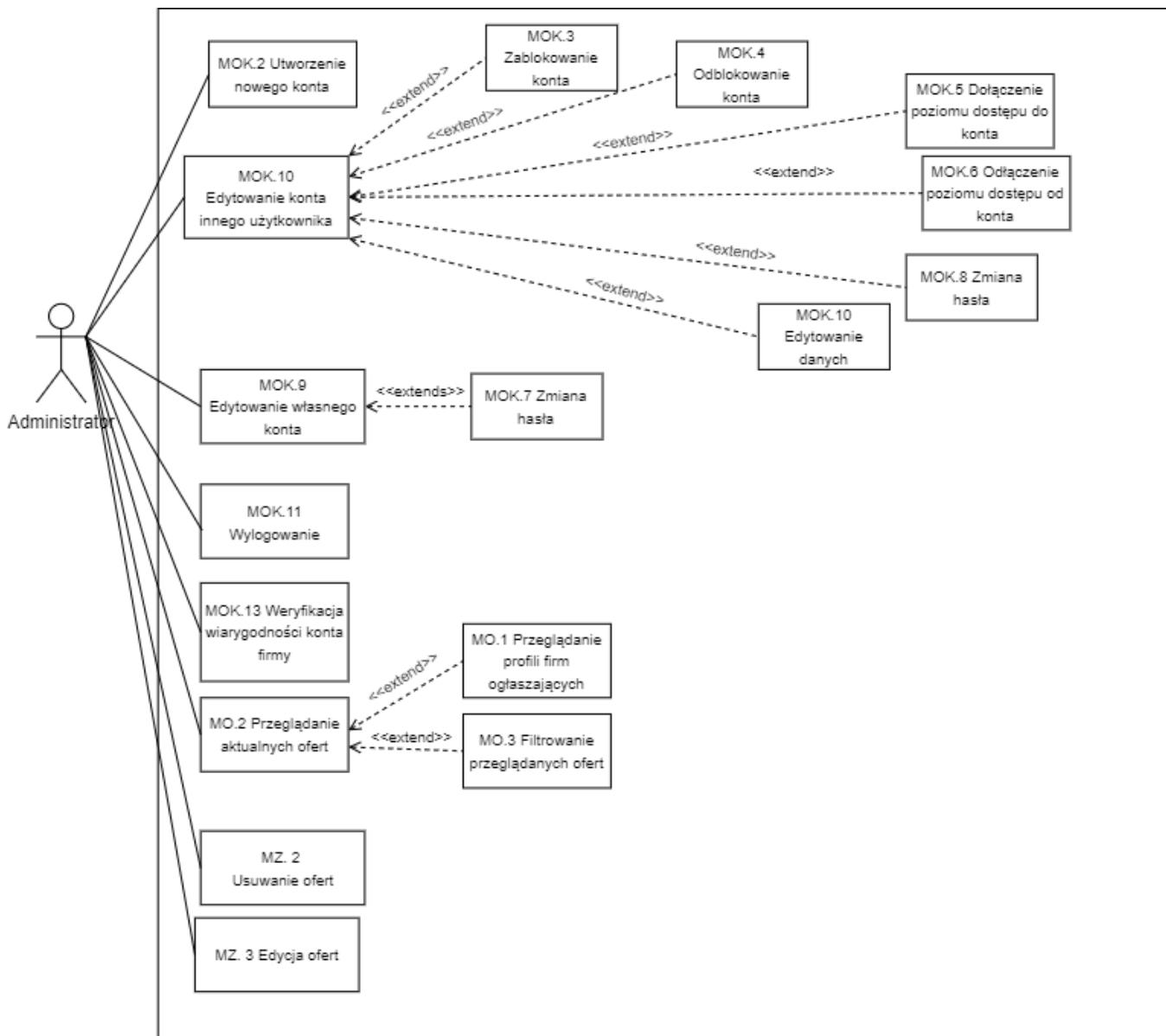


Diagram użycia dla Wynajmującego

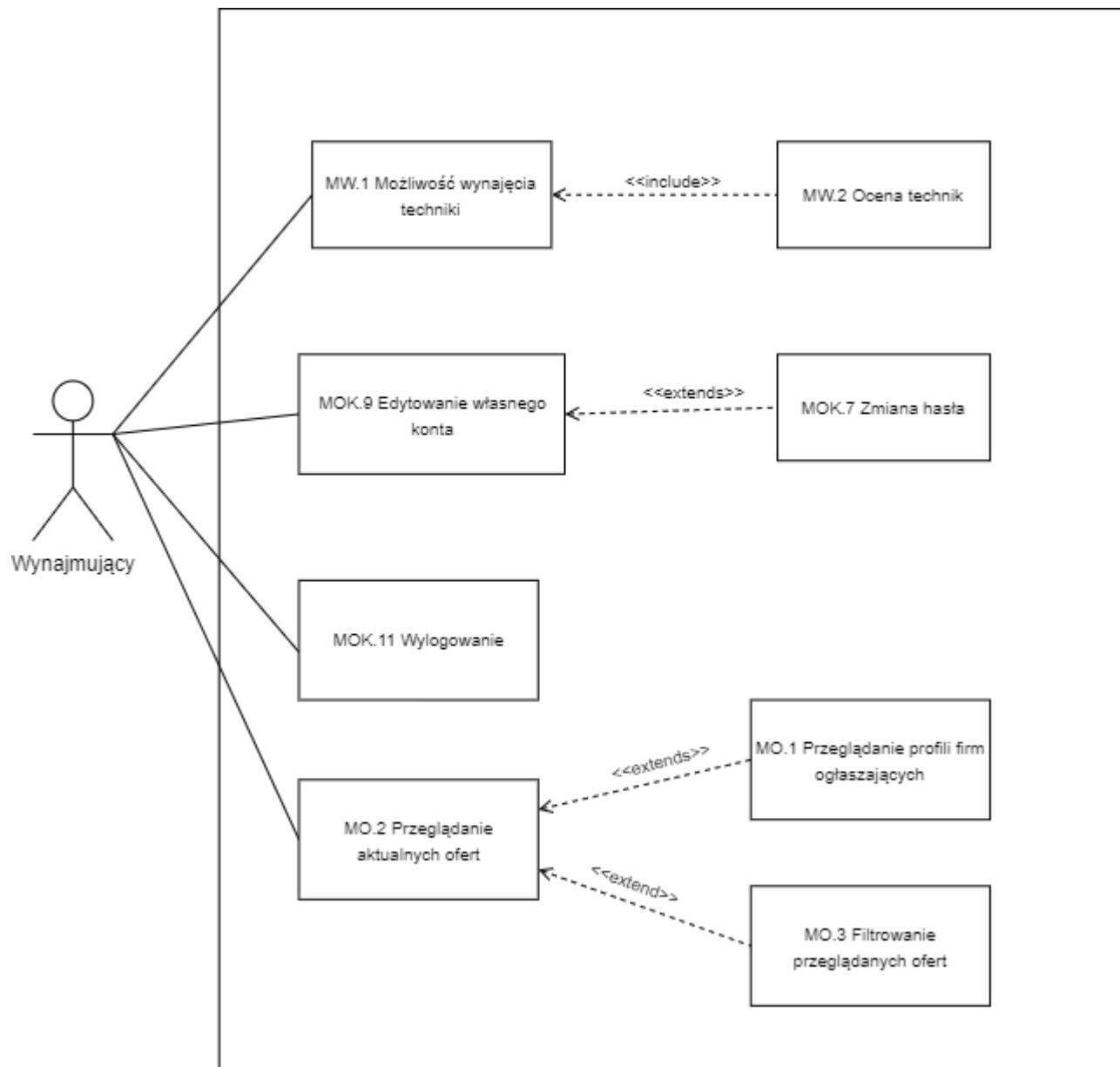


Diagram użycia dla Zgłaszającego

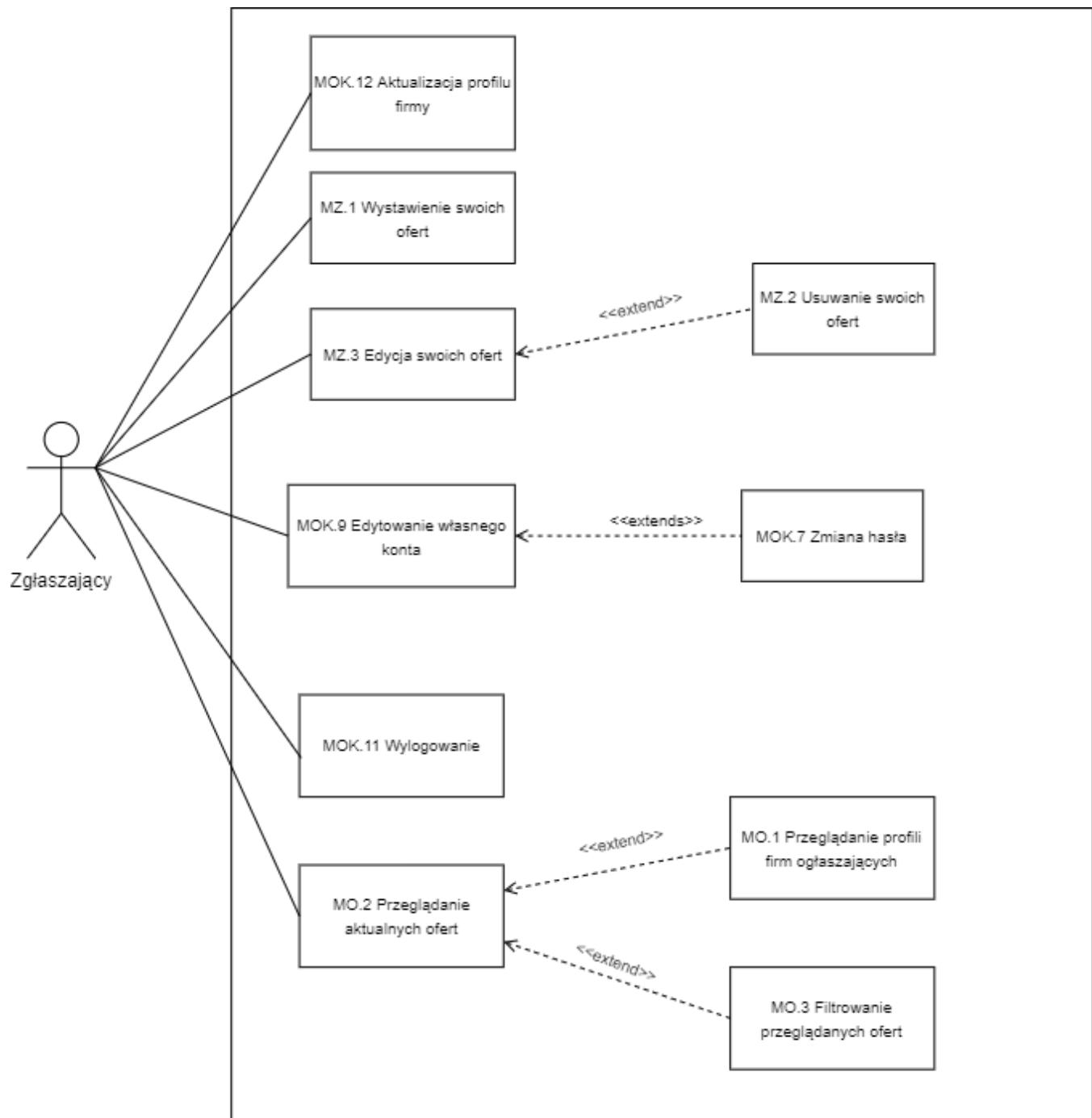
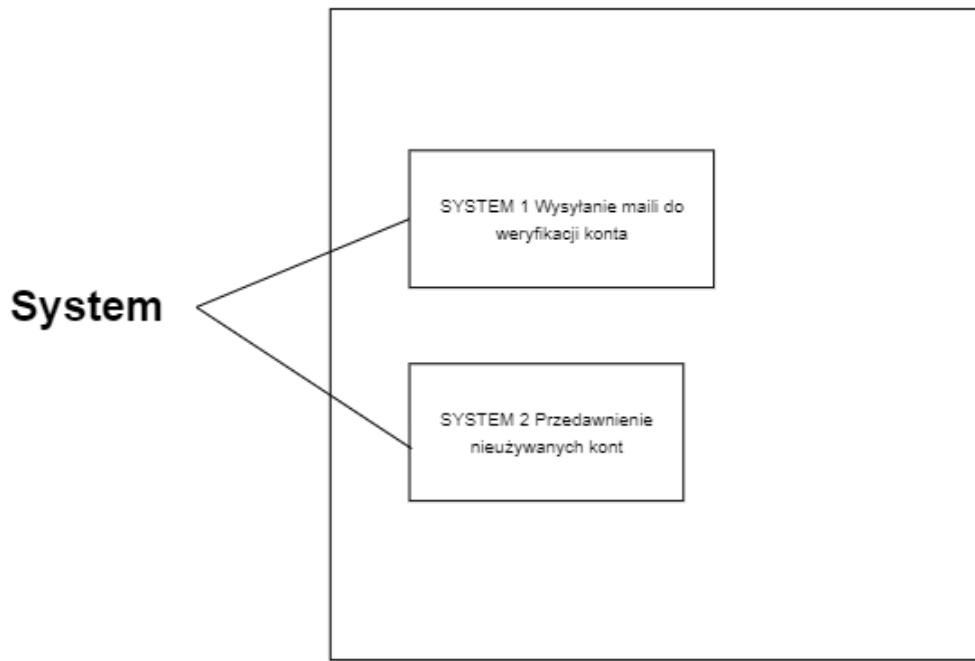


Diagram użycia dla Systemu



| Przypadek użycia | MOK. 1 | Przypadek użycia | MOK. 2 |
|---------------------|--|---------------------|---|
| Nazwa | Zarejestruj | Nazwa | Utwórz konto |
| Aktorzy | Gość | Aktorzy | Administrator |
| Warunki wstępne | Użytkownik jest niezalogowany | Warunki wstępne | Konto posiada uprawnienia administratora |
| Warunki końcowe | Użytkownik rejestruje nowe konto w aplikacji | Warunki końcowe | Administrator tworzy nowe konto |
| Główny scenariusz | <ol style="list-style-type: none"> Na stronie głównej aplikacji, gość używa przycisku „Zarejestruj” w celu przejścia do strony z formularzem rejestracji Użytkownik wybiera typ konta osoby wynajmującej Użytkownik uzupełnia formularz rejestracyjny zgodny z wyborem typu konta Użytkownik zatwierdza poprawne wypełnienie formularza klikając przycisk „Zarejestruj” System przetwarza wprowadzone dane i sprawdza, czy podany użytkownik może zostać utworzony System wykonuje operację dodania nowego konta użytkownika System wysyła mail aktywacyjny na skrzynkę pocztową użytkownika Użytkownik aktywuje konto poprzez link aktywacyjny z otrzymanego maila Użytkownikowi wyświetla się powiadomienie o pomyślnym założeniu konta | Główny scenariusz | <ol style="list-style-type: none"> W panelu administratora, użytkownik wybiera przycisk „Utwórz nowe konto” Użytkownik wypełnia formularz Użytkownik zatwierdza wypełniony formularz klikając przycisk „Utwórz konto” System przetwarza wprowadzone dane oraz sprawdza, czy nowy użytkownik może zostać utworzony System wykonuje operacje dodawania nowego konta Użytkownik zostaje powiadomiony o pomyślnym utworzeniu nowego konta użytkownika |
| Poboczny scenariusz | | Poboczny scenariusz | <p>6a. Użytkownik o podanych danych nie może zostać utworzony</p> <p>7a. System powiadamia administratora o niepomyślnym utworzeniu nowego konta</p> |

| | |
|---------------------|--|
| Poboczny scenariusz | <p>2a. Użytkownik wybiera typ konta osoby zgłaszającej</p> <p>6a. Użytkownik zostaje powiadomiony o tym, że proces zakładania konta czeka na akceptację ze strony administratora.</p> <p>6b. Podany użytkownik nie może zostać utworzony. W tej sytuacji gość zostaje cofnięty do punktu 3</p> <p>7a. Administrator weryfikuje autentyczność osoby zakładającej konto.</p> <p>8a. System wykonuje operację dodania nowego konta użytkownika</p> <p>9a. Użytkownikowi wyświetla się powiadomienie o pomyślnym założeniu konta</p> |
|---------------------|--|

| Przypadek użycia | MOK. 3 |
|---------------------|---|
| Nazwa | Zablokuj konto |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora Wybrane konto do zablokowania posiada stan odblokowany |
| Warunki końcowe | Administrator zablokowuje wybrane konto użytkownika |
| Główny scenariusz | <ol style="list-style-type: none"> Administrator w panelu administratora wyszukuje wybranego użytkownika Administrator w liście użytkowników, na pozycji konkretnego użytkownika, naciska przycisk statusu użytkownika System przetwarza operację, sprawdza, czy wybrany użytkownik posiada stan odblokowany System zablokuje konto Administrator otrzymuje powiadomienie o pomyślnym zablokowaniu konta |
| Poboczny scenariusz | 4a. Administrator otrzymuje powiadomienie o braku możliwości zablokowania konta |

| Przypadek użycia | MOK. 4 |
|---------------------|---|
| Nazwa | Odblokuj konto |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora Wybrane konto do odblokowania posiada stan zablokowany |
| Warunki końcowe | Administrator odblokowuje wybrane konto użytkownika |
| Główny scenariusz | <ol style="list-style-type: none"> Administrator w panelu administratora wyszukuje wybranego użytkownika Administrator w liście użytkowników, na pozycji konkretnego użytkownika, naciska przycisk statusu użytkownika System przetwarza operację, sprawdza, czy wybrany użytkownik posiada stan zablokowany System odblokowuje konto Administrator otrzymuje powiadomienie o pomyślnym odblokowaniu konta |
| Poboczny scenariusz | 4a. Administrator otrzymuje powiadomienie o braku możliwości odblokowania konta |

| Przypadek użycia | MOK. 5 |
|------------------|---|
| Nazwa | Dołącz poziom dostępu do konta |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora Wybrane konto nie posiada dodawanego poziomu dostępu |
| Warunki końcowe | Administrator dodaje nowy poziom dostępu do konta użytkownika |

| Przypadek użycia | MOK. 6 |
|------------------|---|
| Nazwa | Odlacz poziom dostępu od konta |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora Wybrane konto posiada odłączany poziom dostępu |
| Warunki końcowe | Administrator odłącza poziom dostępu od konta użytkownika |

| | | | |
|---------------------|---|---------------------|---|
| Główny scenariusz | <ol style="list-style-type: none"> Administrator w panelu administratora naciska przycisk "Dodaj poziom dostępu" Administrator uzupełnia formularz odpowiedniego poziomu dostępu, który zamierza dodać Administrator naciska przycisk „Dodaj poziom dostępu” w celu potwierdzenia operacji System przetwarza operację dodawania poziomu dostępu wybranemu użytkownikowi, sprawdza, czy użytkownik nie posiada już dodawanego poziomu dostępu System wykonuje operację dodawania poziomu dostępu wybranemu użytkownikowi System powiadamia administratora o pomyślnym wykonaniu operacji dodawania poziomu dostępu | Główny scenariusz | <ol style="list-style-type: none"> Administrator w panelu administratora naciska przycisk "Dodaj poziom dostępu" Administrator naciska przycisk usuwania interesującego go poziomu dostępu, który chce odłączyć od konta użytkownika System przetwarza operację odłączania poziomu dostępu wybranemu użytkownikowi, sprawdza, czy użytkownik posiada odłączany poziom dostępu System wykonuje operację odłączania poziomu dostępu wybranemu użytkownikowi System powiadamia administratora o pomyślnym wykonaniu operacji odłączania poziomu dostępu |
| Poboczny scenariusz | <p>6a. Podany poziom dostępu nie mógł zostać dodany do wybranego użytkownika</p> <p>7a System powiadamia administratora o niepomyślnym zakończeniu operacji</p> | Poboczny scenariusz | <p>6a. Podany poziom dostępu nie mógł zostać odłączony od wybranego użytkownika</p> <p>7a. System powiadamia administratora o niepomyślnym zakończeniu operacji</p> |

| Przypadek użycia | MOK. 7 |
|---------------------|---|
| Nazwa | Zmień własne hasło |
| Aktorzy | Administrator, Wynajmujący, Zgłaszający |
| Warunki wstępne | Użytkownik jest zalogowany |
| Warunki końcowe | Hasło zalogowanego użytkownika zostało zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> Użytkownik naciska przycisk "Moje konto" Użytkownik naciska przycisk „Zmień hasło” Użytkownik wypełnia formularz zmiany hasła Użytkownik zatwierdza zmianę naciskając przycisk „Zmień hasło” System wyświetla użytkownikowi okno dialogowe z prośbą o potwierdzenie wykonania operacji zmiany hasła Użytkownik naciska przycisk „Zmień hasło” potwierdzając operację System przetwarza dane wprowadzone do formularza, sprawdza zgodność i poprawność haseł System wykonuje operacje zmiany hasła użytkownika System powiadamia użytkownika o pomyślnym wykonaniu operacji zmiany hasła |
| Poboczny scenariusz | <p>6a. Użytkownik naciska przycisk „Anuluj” przerywając operację</p> <p>8a. System informuje użytkownika o niepoprawnie wypełnionym formularzu i cofa użytkownika do punktu nr 3</p> |

| Przypadek użycia | MOK. 8 |
|---------------------|--|
| Nazwa | Zmień hasło innego użytkownika |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora |
| Warunki końcowe | Hasło wybranego użytkownika zostało zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> Administrator w panelu administratora wyszukuje wybranego użytkownika Administrator naciska przycisk zmiany hasła wybranego użytkownika Administrator wypełnia formularz zmiany hasła Administrator zatwierdza zmianę naciskając przycisk „Zmień hasło” System wyświetla użytkownikowi okno dialogowe z prośbą o potwierdzenie wykonania operacji zmiany hasła Administrator naciska przycisk „Zmień hasło” potwierdzając operację System przetwarza dane wprowadzone do formularza, sprawdza zgodność i poprawność haseł System wykonuje operacje zmiany hasła użytkownika System powiadamia administratora o pomyślnym wykonaniu operacji zmiany hasła |
| Poboczny scenariusz | <p>6a. Administrator naciska przycisk „Anuluj” przerywając operację</p> <p>8a. System informuje administratora o niepoprawnie wypełnionym formularzu i cofa użytkownika do punktu nr 3</p> |

| Przypadek użycia | MOK. 9 |
|------------------|---|
| Nazwa | Edytuj dane własnego konta |
| Aktorzy | Administrator, Wynajmujący, Zgłaszający |
| Warunki wstępne | Użytkownik jest zalogowany |

| Przypadek użycia | MOK. 10 |
|------------------|---|
| Nazwa | Edytuj dane innego użytkownika |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora |

| | |
|---------------------|--|
| Warunki końcowe | Dane zalogowanego użytkownika zostały zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik naciska przycisk "Moje konto" 2. Użytkownik naciska przycisk „Modyfikuj konto” 3. Użytkownik naciska ikonę edycji wybranego pola danych w profilu 4. Użytkownik edytuje dane znajdujące się w polu 5. Użytkownik zatwierdza zmianę naciskając przycisk „Zapisz” 6. System przetwarza dane wprowadzone do pola, sprawdza poprawność danych 7. System wykonuje operacje zmiany danych użytkownika 8. System powiadamia użytkownika o pomyślnym wykonaniu operacji zmiany danych |
| Poboczny scenariusz | <p>4a. Użytkownik naciska przycisk „Odrzuć zmiany” przerywając operację</p> <p>6a. System informuje użytkownika o niepoprawnie wprowadzonych danych</p> |

| Przypadek użycia | MOK. 11 |
|---------------------|---|
| Nazwa | Wyloguj |
| Aktorzy | Administrator, Wynajmujący, Zgłaszający |
| Warunki wstępne | Użytkownik jest zalogowany |
| Warunki końcowe | Użytkownik zostaje wylogowany |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik naciska przycisk „Wyloguj” 2. System wyświetla użytkownikowi okno dialogowe z prośbą o potwierdzenie wykonania operacji wylogowania 3. Użytkownik naciska przycisk „Wyloguj” potwierdzając wylogowanie 4. System sprawdza, czy możliwa jest operacja wylogowania 5. System wykonuje operacje wylogowania użytkownika 6. System informuje użytkownika o pomyślnym wylogowaniu |
| Poboczny scenariusz | 5a. System informuje użytkownika o niepoprawnym wylogowaniu |

| | |
|---------------------|--|
| Warunki końcowe | Dane wybranego użytkownika zostały zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Administrator w panelu administratora wyszukuje wybranego użytkownika 2. Administrator naciska przycisk "Modyfikuj konto" przy wybranym użytkowniku" 3. Administrator naciska ikonę edycji wybranego pola danych w profilu 4. Administrator edytuje dane znajdujące się w polu 5. Administrator zatwierdza zmianę naciskając przycisk „Zatwierdź” 6. System wyświetla administratorowi okno dialogowe z prośbą o potwierdzenie wykonania operacji edycji pola 7. Administrator naciska przycisk „Potwierdź” potwierdzając operację 8. System przetwarza dane wprowadzone do pola, sprawdza poprawność danych 9. System wykonuje operacje zmiany danych użytkownika 10. System powiadamia administratora o pomyślnym wykonaniu operacji zmiany danych |
| Poboczny scenariusz | <p>5a. Administrator naciska przycisk „Odrzuć zmiany” przerywając operację</p> <p>9a. System informuje administratora o niepoprawnie wprowadzonych danych</p> |

| Przypadek użycia | MOK. 12 |
|---------------------|---|
| Nazwa | Aktualizacja profilu firmy |
| Aktorzy | Zgłaszający |
| Warunki wstępne | Użytkownik posiada uprawnienia zgłoszającego |
| Warunki końcowe | Dane profilu firmy zalogowanego użytkownika zostały zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik naciska przycisk "Moje konto" 2. Użytkownik naciska przycisk „Modyfikuj konto” 3. Użytkownik naciska ikonę edycji wybranego pola danych w profilu 4. Użytkownik edytuje dane znajdujące się w polu 5. Użytkownik zatwierdza zmianę naciskając przycisk „Zatwierdź” 6. System wyświetla użytkownikowi okno dialogowe z prośbą o potwierdzenie wykonania operacji edycji pola 7. Użytkownik naciska przycisk „Potwierdź” potwierdzając operację 8. System przetwarza dane wprowadzone do pola, sprawdza poprawność danych 9. System wykonuje operacje zmiany danych użytkownika 10. System powiadamia użytkownika o pomyślnym wykonaniu operacji zmiany danych |
| Poboczny scenariusz | <p>5a. Użytkownik naciska przycisk „Odrzuć zmiany” przerywając operację</p> <p>9a. System informuje użytkownika o niepoprawnie wprowadzonych danych</p> |

| Przypadek użycia | MOK. 13 |
|------------------|---------|
|------------------|---------|

| Przypadek użycia | MOK. 14 |
|------------------|---------|
|------------------|---------|

| | |
|---------------------|---|
| Nazwa | Weryfikacja wiarygodności firmy |
| Aktorzy | Administrator |
| Warunki wstępne | Użytkownik posiada uprawnienia administratora Wybrany użytkownik posiada stan niezweryfikowany |
| Warunki końcowe | Konto wybranego użytkownika zostaje zweryfikowane |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Administrator w panelu administratora przechodzi do zakładki „Konta oczekujące na weryfikację” 2. Administrator wybiera z listy konto do weryfikacji 3. Administrator weryfikuje dane podane przez osobę chcącą założyć konto 4. Administrator naciska przycisk „Oznacz jako zweryfikowane” w celu potwierdzenia prawdziwości danych 5. System sprawdza, czy weryfikowany użytkownik może zostać uznany jako zweryfikowany 6. System wykonuje operację weryfikowania podanego użytkownika 7. System informuje administratora o pomyślnym zweryfikowaniu użytkownika 8. System informuje weryfikowanego użytkownika o potwierdzeniu jego konta |
| Poboczny scenariusz | 6a. System informuje administratora o braku możliwości zweryfikowania wybranego użytkownika |

| | |
|---------------------|--|
| Nazwa | Logowanie |
| Aktorzy | Gość |
| Warunki wstępne | Użytkownik jest niezalogowany |
| Warunki końcowe | Użytkownik jest zalogowany |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik naciska przycisk „Zaloguj”, która przeniesie go do formularza logowania 2. Użytkownik wypełnia formularz danych potrzebnych do zalogowania 3. Użytkownik naciska przycisk „Zaloguj”, aby podjąć próbę zalogowania w systemie 4. System weryfikuje wprowadzone dane przez użytkownika, sprawdza, czy podane konto istnieje, potwierdzone i zweryfikowane 5. System loguje użytkownika do aplikacji 6. System informuje użytkownika o pomyślnym zalogowaniu |
| Poboczny scenariusz | 5a. System informuje użytkownika o nieistnieniu konta (?) lub niepoprawnych danych logowania |

| Przypadek użycia | MOK.15 |
|---------------------|---|
| Nazwa | Reset hasła |
| Aktorzy | Gość |
| Warunki wstępne | Użytkownik jest niezalogowany |
| Warunki końcowe | Hasło należące do konta z podanym adresem email, zostało zmienione |
| Główny scenariusz | <ol style="list-style-type: none"> 1.Użytkownik naciska przycisk „Zaloguj”, która przeniesie go do formularza logowania 2.Użytkownik naciska przycisk "Przywróć hasło" 3.Użytkownik podaje adres email na który ma zostać wysłany link ze zmianą hasła 5.System sprawdza czy podany mail istnieje w bazie danych. Jeśli tak wysyła na ten mail link do zmiany hasła 4.Użytkownik naciska w link wysłany na podany adres email 5.System przenosi użytkownika do odpowiedniego formularza na stronie 6.Użytkownik wpisuje nowe hasło oraz je powtarza 7.Użytkownik naciska przycisk "Przywróć hasło" 8.Jeśli podane hasła spełniają wymogi, system informuje użytkownika o poprawnej zmianie hasła |
| Poboczny scenariusz | 5a.System sprawdza czy podany mail istnieje w bazie danych. Jeśli nie, na stronie pojawi się stosowny komunikat 8a.Jeśli podane hasła nie spełniają wymogów, system informuje użytkownika o nieudanej zmianie hasła |

| Przypadek użycia | MZ. 1 |
|------------------|--|
| Nazwa | Dodawanie ofert |
| Aktorzy | Zgłaszący |
| Warunki wstępne | Zalogowany użytkownik ma uprawnienia zgłaszącego |

| Przypadek użycia | MZ. 2 |
|------------------|--------------------------|
| Nazwa | Usuwanie ofert |
| Aktorzy | Administrator, Zgłaszący |

| | | | |
|---------------------|--|--------------------------|---|
| Warunki końcowe | Użytkownik dodał nową ofertę | Warunki wstępne | Użytkownik posiada uprawnienia administratora w przypadku głównego scenariusza nr 1 Użytkownik posiada uprawnienia zgłaszającego w przypadku głównego scenariusza nr 2 Firma posiada przynajmniej jedną ofertę |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Moje oferty” 2. Użytkownik naciska przycisk „Dodaj ofertę” 3. Użytkownik wypełnia formularz danych potrzebnych do utworzenia oferty 4. Użytkownik zatwierdza wprowadzone dane naciskając przycisk „Dodaj ofertę” 5. System wyświetla okno dialogowe z prośbą o potwierdzenie wykonania operacji dodawania nowej oferty 6. Użytkownik naciska przycisk „Dodaj” potwierdzając operację dodawania oferty 7. System przetwarza dane wprowadzone do formularza nowej oferty 8. System dodaje nową ofertę 9. System informuje użytkownika o dodaniu nowej oferty | Warunki końcowe | Użytkownik usunął ofertę |
| Poboczny scenariusz | <p>6a. Użytkownik naciska przycisk „Anuluj” przerywając operację dodawania oferty</p> <p>8a. System informuje użytkownika o niepoprawnych danych wprowadzonych do formularza</p> | Główny scenariusz nr 1 | <ol style="list-style-type: none"> 1. Administrator z listy ofert przechodzi na profil firmy, której ofertę chce usunąć 2. Administrator naciska przycisk „Usuń ofertę” dedykowany do oferty, którą zamierza usunąć 3. System wyświetla okno dialogowe z prośbą o potwierdzenie wykonania operacji usunięcia oferty 4. Administrator naciska przycisk „Usuń ofertę” potwierdzając operację usunięcia oferty 5. System sprawdza, czy dana oferta może zostać usunięta 6. System wykonuje operację usunięcia wybranej oferty 7. System informuje administratora o pomyślnym usunięciu oferty |
| | | Główny scenariusz nr 2 | <ol style="list-style-type: none"> 1. Zgłaszący przechodzi do zakładki „Moje oferty” w profilu firmy 2. Zgłaszący naciska przycisk „Usuń ofertę” dedykowany do oferty, którą zamierza usunąć 3. System wyświetla okno dialogowe z prośbą o potwierdzenie wykonania operacji usunięcia oferty 4. Zgłaszący naciska przycisk „Usuń ofertę” potwierdzając operację usunięcia oferty 5. System sprawdza, czy dana oferta może zostać usunięta 6. System wykonuje operację usunięcia wybranej oferty 7. System informuje zgłaszącego o pomyślnym usunięciu oferty |
| | | Poboczny scenariusz nr 1 | <p>4a. Administrator naciska przycisk „Anuluj” w celu przerwania operacji</p> <p>6a. System informuje użytkownika o braku możliwości usunięcia wybranej oferty</p> |
| | | Poboczny scenariusz nr 2 | <p>4a. Zgłaszący naciska przycisk „Anuluj” w celu przerwania operacji</p> <p>6a. System informuje użytkownika o braku możliwości usunięcia wybranej oferty</p> |

| Przypadek użycia | MZ. 3 |
|------------------|---|
| Nazwa | Edycja ofert |
| Aktorzy | Administrator, Zgłaszący |
| Warunki wstępne | <p>Użytkownik posiada uprawnienia administratora w przypadku głównego scenariusza nr 1</p> <p>Użytkownik posiada uprawnienia zgłaszającego w przypadku głównego scenariusza nr 2</p> <p>Firma posiada przynajmniej jedną ofertę</p> |
| Warunki końcowe | Użytkownik edytował ofertę |

| Przypadek użycia | MO. 1 |
|------------------|--|
| Nazwa | Przeglądanie profili firm ogłaszających |
| Aktorzy | Gość, Administrator, Wynajmujący, Zgłaszący |
| Warunki wstępne | Przynajmniej jedna firma istnieje w systemie |
| Warunki końcowe | Użytkownik wyświetla profil wybranej firmy |

| | | | |
|--------------------------|---|-------------------|---|
| Główny scenariusz nr 1 | <ol style="list-style-type: none"> 1. Administrator przechodzi do zakładki „Oferty” 2. Administrator naciska przycisk „Edytuj ofertę” dedykowany do oferty, которую замерза edytować 3. Administrator edytuje wybrane pola oferty 4. Administrator naciska przycisk „Potwierdź” w celu zatwierdzenia wprowadzonych zmian 5. System wyświetla okno dialogowe z prośbą o potwierdzenie wykonania operacji edycji oferty 6. Administrator naciska przycisk „Potwierdź” w celu potwierdzenia operacji 7. System przetwarza edytowane pola oferty, sprawdza czy dana oferta może zostać edytowana oraz sprawdza poprawność wprowadzonych danych 8. System wykonuje operację edycji oferty 9. System informuje administratora o pomyślnym zakończeniu operacji edycji oferty | Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Firmy” 2. Użytkownik na liście ofert wybiera interesującą go pozycję 3. Użytkownik naciska na nazwę firmy wymienionej w ofercie |
| Główny scenariusz nr 2 | <ol style="list-style-type: none"> 1. Zgłaszący przechodzi do zakładki „Moje oferty” w profilu firmy 2. Zgłaszący naciska przycisk „Edytuj ofertę” dedykowany do oferty, которую zamierza edytować 3. Zgłaszący edytuje wybrane pola oferty 4. Zgłaszący naciska przycisk „Potwierdź” w celu zatwierdzenia wprowadzonych zmian 5. System wyświetla okno dialogowe z prośbą o potwierdzenie wykonania operacji edycji oferty 6. Zgłaszący naciska przycisk „Potwierdź” w celu potwierdzenia operacji 7. System przetwarza edytowane pola oferty, sprawdza czy dana oferta może zostać edytowana oraz sprawdza poprawność wprowadzonych danych 8. System wykonuje operację edycji oferty 9. System informuje zgłaszącego o pomyślnym zakończeniu operacji edycji oferty | | |
| Poboczny scenariusz nr 1 | <p>4a. Administrator naciska przycisk „Anuluj” w celu przerwania operacji</p> <p>8a. System informuje administratora o braku możliwości wykonania operacji edycji oferty</p> | | |
| Poboczny scenariusz nr 2 | <p>4a. Zgłaszący naciska przycisk „Anuluj” w celu przerwania operacji</p> <p>8a. System informuje zgłaszącego o braku możliwości wykonania operacji edycji oferty</p> | | |

| Przypadek użycia | MO. 2 |
|-------------------|--|
| Nazwa | Przeglądanie aktualnych ofert |
| Aktorzy | Gość, Administrator, Wynajmujący, Zgłaszący |
| Warunki wstępne | Przynajmniej jedna oferta istnieje w systemie |
| Warunki końcowe | Użytkownik wyświetla listę dostępnych ofert |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Oferty” 2. Użytkownik na liście ofert wybiera interesującą go pozycję |

| Przypadek użycia | MO. 3 |
|------------------|--|
| Nazwa | Filtrowanie przeglądanych ofert |
| Aktorzy | Gość, Administrator, Wynajmujący, Zgłaszący |
| Warunki wstępne | Przynajmniej jedna oferta istnieje w systemie |
| Warunki końcowe | Użytkownik wyświetla zaktualizowaną listę ofert, z nałożonymi filtrami |

| | |
|--------------------------|--|
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Oferty” 2. Użytkownik wybiera z listy interesującą go filtry ofert 3. Użytkownik naciska przycisk „Wyszukaj” celem zastosowania filtrów 4. System filtruje oferty względem podanych przez użytkownika filtrów 5. System wyświetla użytkownikowi zaktualizowaną listę |
|--------------------------|--|

| Przypadek użycia | MW. 1 |
|-------------------------|---|
| Nazwa | Możliwość wynajęcia techniki |
| Aktorzy | Wynajmujący |
| Warunki wstępne | Użytkownik posiada uprawnienia wynajmującego W systemie istnieje przynajmniej jedna dostępna do wynajęcia oferta |
| Warunki końcowe | Użytkownik wynajmuje oferowaną technikę estradową |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Oferty” 2. Użytkownik wybiera interesującą go ofertę 3. Użytkownik wybiera okres czasu, przez który chce wynająć firmę 4. Użytkownik naciska przycisk „Wynajmij” 5. System wyświetla okno dialogowe z prośba o potwierdzenie wynajęcia techniki 6. Użytkownika naciska przycisk „Wynajmij” w celu potwierdzenia wykonania operacji 7. System sprawdza, czy wybrana oferta jest dostępna w konkretnym okresie czasu 8. System wykonuje operacje wynajęcia firmy 9. System informuje użytkownika o pomyślnym wynajęciu techniki |
| Poboczny scenariusz | 6a. Użytkownik naciska przycisk „Anuluj” w celu przerwania operacji 8a. System informuje użytkownika o braku możliwości wynajęcia techniki |

| Przypadek użycia | MW. 2 |
|-------------------------|--|
| Nazwa | Ocenianie technik |
| Aktorzy | Wynajmujący |
| Warunki wstępne | Użytkownik posiada uprawnienia wynajmującego Użytkownik skorzystał z przynajmniej jednej oferty wynajmu techniki |
| Warunki końcowe | Użytkownik ocenia ofertę techniki |
| Główny scenariusz | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki „Moje wynajmy” 2. Użytkownik wybiera ofertę, której wynajem się już skończył 3. Użytkownik ocenia ofertę w podanej skali 4. System sprawdza, czy dana technika może zostać oceniona 5. System dodaje ocenę użytkownika do oferty firmy 6. System informuje użytkownika o wykonaniu operacji oceny oferty |
| Poboczny scenariusz | 5a. System informuje użytkownika o braku możliwości oceny oferty |

| Przypadek użycia | SYSTEM 1 |
|-------------------------|--|
| Nazwa | Wysyłanie maili do weryfikacji konta |
| Aktorzy | System |
| Warunki wstępne | Użytkownik założył konto |
| Warunki końcowe | Użytkownik aktywował konto |
| Główny scenariusz | <ol style="list-style-type: none"> 1. System po skończonej operacji dodawania konta, wysyła maila do nowego użytkownika na podany w formularzu rejestracyjnym adres mailowy |

| Przypadek użycia | SYSTEM 2 |
|-------------------------|--|
| Nazwa | Usuwanie niepotwierdzonych kont |
| Aktorzy | System |
| Warunki wstępne | Użytkownik posiada konto Użytkownik nie potwierdził konta |
| Warunki końcowe | Konto użytkownika jest usuwane |
| Główny scenariusz | <ol style="list-style-type: none"> 1. System wykrywa, że użytkownik nie potwierdził konta przez określoną ilość czasu. 2. System usuwa użytkownika |

3 Struktury relacyjnej bazy danych 2 pkt

Tabela access_level

```
create table access_level
(
    id serial
        constraint access_level_pkey
            primary key,
    access_level varchar(31),
    active boolean,
    version bigint,
    account_id bigint
        constraint fk_access_level_account_id
            references account
);
```

id - identyfikator poziomu dostępu, klucz główny tabeli access_level

access_level - nazwa poziomu dostępu

active - określa czy poziom dostępu jest aktualnie aktywny

version - wersja encji

account_id - identyfikator konta, klucz obcy łączący z tabelą account

Indexes:

"access_level_pkey" PRIMARY KEY, btree (id)

"access_level_account_fk" btree (account_id)

Foreign-key constraints:

"fk_access_level_account_id" FOREIGN KEY (account_id) REFERENCES account(id)

Referenced by:

TABLE "admin_details" CONSTRAINT "fk_admin_details_id" FOREIGN KEY (id) REFERENCES access_level(id)

TABLE "offer" CONSTRAINT "fk_offer_service_provider_id" FOREIGN KEY (service_provider_id) REFERENCES access_level(id)

TABLE "rate_renter_details" CONSTRAINT "fk_rate_renter_details_renters_id" FOREIGN KEY (renters_id) REFERENCES access_level(id)

TABLE "rate" CONSTRAINT "fk_rate_service_provider_id" FOREIGN KEY (service_provider_id) REFERENCES access_level(id)

TABLE "renter_details" CONSTRAINT "fk_renter_details_id" FOREIGN KEY (id) REFERENCES access_level(id)

TABLE "service_provider_details" CONSTRAINT "fk_service_provider_details_id" FOREIGN KEY (id) REFERENCES access_level(id)

TABLE "user_offer" CONSTRAINT "fk_user_offer_renter_id" FOREIGN KEY (renter_id) REFERENCES access_level(id)

Tabela admin_details

```
create table admin_details
(
    id bigint not null
        constraint admin_details_pkey
            primary key,
    constraint fk_admin_details_id
        references access_level
);
```

id - identyfikator, klucz główny tej tabeli, będący jednocześnie kluczem obcym łączącym z tabelą access_level

Indexes:

"admin_details_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"fk_admin_details_id" FOREIGN KEY (id) REFERENCES access_level(id)

Tabela renter_details

```
create table renter_details
(
    id bigint not null
    constraint renter_details_pkey
        primary key
    constraint fk_renter_details_id
        references access_level,
    user_name varchar(255)
    constraint renter_details_user_name_key
        unique
);
```

id - identyfikator, klucz główny tej tabeli, będący jednocześnie kluczem obcym łączącym z tabelą access_level

user_name - nazwa użytkownika, jest unikatowa

Indexes:

"renter_details_pkey" PRIMARY KEY, btree (id)

"renter_details_user_name_key" UNIQUE CONSTRAINT, btree (user_name)

Foreign-key constraints:

"fk_renter_details_id" FOREIGN KEY (id) REFERENCES access_level(id)

Tabela service_provider_details

```
create table service_provider_details
(
    id bigint not null
    constraint service_provider_details_pkey
        primary key
    constraint fk_service_provider_details_id
        references access_level,
    address varchar(255)
    constraint service_provider_details_address_key
        unique,
    description varchar(255),
    logo_url varchar(255),
    nip varchar(255)
    constraint service_provider_details_nip_key
        unique,
    service_name varchar(255)
    constraint service_provider_details_service_name_key
        unique
);
```

id - identyfikator, klucz główny tej tabeli, będący jednocześnie kluczem obcym łączącym z tabelą access_level

service_name - nazwa świadczonej usługi

address - adres działalności, jest unikatowy

description - opis działalności

logo_url - adres loga działalności

nip - numer NIP działalności, jest unikatowy

service_name - nazwa usługi, jest unikatowa

Indexes:

```
"service_provider_details_pkey" PRIMARY KEY, btree (id)
"service_provider_details_address_key" UNIQUE CONSTRAINT, btree (address)
"service_provider_details_nip_key" UNIQUE CONSTRAINT, btree (nip)
"service_provider_details_service_name_key" UNIQUE CONSTRAINT, btree (service_name)
```

Foreign-key constraints:

```
"fk_service_provider_details_id" FOREIGN KEY (id) REFERENCES access_level(id)
```

Tabela account

```
create table account
(
    id serial
        constraint account_pkey
        primary key,
    active boolean,
    confirmed boolean,
    email varchar(255)
        constraint account_email_key
        unique,
    login varchar(255)
        constraint account_login_key
        unique,
    name varchar(255),
    password varchar(255),
    phone_number varchar(255),
    surname varchar(255),
    version bigint
);
```

id - identyfikator, klucz główny tej tabeli

active - określa czy konto jest aktywne

confirmed - pole określające czy konto zostało potwierdzone

email - email użytkownika, jest unikatowy

login - login użytkownika, jest unikatowy

name - imię użytkownika

password - hasło użytkownika

phone_number - numer telefonu użytkownika

surname - nazwisko użytkownika

version - wersja encji

Indexes:

```
"account_pkey" PRIMARY KEY, btree (id)  
"account_email_key" UNIQUE CONSTRAINT, btree (email)  
"account_login_key" UNIQUE CONSTRAINT, btree (login)
```

Referenced by:

```
TABLE "access_level" CONSTRAINT "fk_access_level_account_id" FOREIGN KEY (account_id) REFERENCES account(id)
```

```
TABLE "verification_token" CONSTRAINT "fk_verification_token_account_id" FOREIGN KEY (account_id) REFERENCES account(id)
```

Tabela rate

```
create table rate  
(  
id serial  
constraint rate_pkey  
primary key,  
average_rate double precision,  
rates_number integer,  
version bigint,  
service_provider_id bigint  
constraint fk_rate_service_provider_id  
references access_level  
);
```

id - identyfikator, klucz główny tej tabeli

average_rate - średnia ocena

rates_number - liczba ocen

version - wersja encji

service_provider_id - identyfikator świadczącego usługę, klucz obcy łączący z tabelą service_provider_details

Indexes:

```
"rate_pkey" PRIMARY KEY, btree (id)
```

Foreign-key constraints:

```
"fk_rate_service_provider_id" FOREIGN KEY (service_provider_id) REFERENCES access_level(id)
```

Referenced by:

```
TABLE "rate_renter_details" CONSTRAINT "fk_rate_renter_details_ratecollection_id" FOREIGN KEY (ratecollection_id) REFERENCES rate(id)
```

Tabela offer

```
create table offer  
(  
id serial  
constraint offer_pkey  
primary key,  
active boolean,  
description varchar(255),  
price integer,  
title varchar(255),  
version bigint,  
service_provider_id bigint  
constraint fk_offer_service_provider_id  
references access_level  
);
```

id - identyfikator, klucz główny tej tabeli

active - określa czy oferta jest aktywna

description - opis oferty

price - cena usługi

title - tytuł oferty

version - wersja encji

service_provider_id - identyfikator świadczącego usługę, klucz obcy łączący z tabelą access_level

Indexes:

"offer_pkey" PRIMARY KEY, btree (id)

"offer_service_provider_fk" btree (service_provider_id)

Foreign-key constraints:

"fk_offer_service_provider_id" FOREIGN KEY (service_provider_id) REFERENCES access_level(id)

Referenced by:

TABLE "offer_date" CONSTRAINT "fk_offer_date_offer_id" FOREIGN KEY (offer_id) REFERENCES offer(id)

Tabela offer_date

```
create table offer_date
(
    id serial
    constraint offer_date_pkey
        primary key,
    date date,
    version bigint,
    offer_id bigint
    constraint fk_offer_date_offer_id
        references offer
);
```

id - identyfikator, klucz główny tej tabeli

date - data oferty

version - wersja encji

offer_id - identyfikator oferty, klucz obcy łączący z tabelą offer

Indexes:

"offer_date_pkey" PRIMARY KEY, btree (id)

"offer_date_offer_fk" btree (offer_id)

Foreign-key constraints:

"fk_offer_date_offer_id" FOREIGN KEY (offer_id) REFERENCES offer(id)

Referenced by:

TABLE "user_offer" CONSTRAINT "fk_user_offer_offer_date_id" FOREIGN KEY (offer_date_id) REFERENCES offer_date(id)

Tabela user_offer

```

create table user_offer
(
id serial
constraint user_offer_pkey
primary key,
version bigint,
renter_id bigint
constraint fk_user_offer_renter_id
references access_level,
offer_date_id bigint
constraint fk_user_offer_offer_date_id
references offer_date
);

```

id - identyfikator, klucz główny tej tabeli

version - wersja encji

renter_id - identyfikator wynajmującego, klucz obcy łączący z tabelą access_level

offer_date_id - identyfikator daty oferty, klucz obcy łączący z tabelą offer_date

Indexes:

```

"user_offer_pkey" PRIMARY KEY, btree (id)

"user_offer_offer_date_fk" btree (offer_date_id)

"user_offer_renter_fk" btree (renter_id)

```

Foreign-key constraints:

```

"fk_user_offer_offer_date_id" FOREIGN KEY (offer_date_id) REFERENCES offer_date(id)

"fk_user_offer_renter_id" FOREIGN KEY (renter_id) REFERENCES access_level(id)

```

Tabela verification_token

```

create table verification_token
(
id serial
constraint verification_token_pkey
primary key,
creation_date timestamp,
expiry_date timestamp,
generated_token varchar(255)
constraint verification_token_generated_token_key
unique,
version bigint,
account_id bigint
constraint fk_verification_token_account_id
references account
);

```

id - identyfikator, klucz główny tej tabeli

creation_date - data utworzenia tokenu

expiry_date - data wygaśnięcia tokenu

generated_token - wygenerowany token, jest unikatowy

account_id - identyfikator konta, klucz obcy łączący z tabelą account

Indexes:

```
"verification_token_pkey" PRIMARY KEY, btree (id)  
"verification_token_account_fk" btree (account_id)  
"verification_token_generated_token_key" UNIQUE CONSTRAINT, btree (generated_token)  
  
Foreign-key constraints:  
"fk_verification_token_account_id" FOREIGN KEY (account_id) REFERENCES account(id)
```

Tabela rate_renter_details - tabela umożliwiająca relację "wiele do wielu" między tabelami rate oraz renter_details

```
create table rate_renter_details  
(  
renters_id bigint not null  
constraint fk_rate_renter_details_renters_id  
references access_level,  
ratecollection_id bigint not null  
constraint fk_rate_renter_details_ratecollection_id  
references rate,  
constraint rate_renter_details_pkey  
primary key (renters_id, ratecollection_id)  
);
```

obie kolumny są razem kluczem głównym tabeli (dany renter może ocenić daną ofertę tylko raz)

renters_id - identyfikator wynajmującego, klucz obcy łączący z tabelą access_level

ratecollection_id - identyfikator ocenianej oferty, klucz obcy łączący z tabelą rate

Indexes:

```
"rate_renter_details_pkey" PRIMARY KEY, btree (renters_id, ratecollection_id)  
  
Foreign-key constraints:  
"fk_rate_renter_details_ratecollection_id" FOREIGN KEY (ratecollection_id) REFERENCES rate(id)  
"fk_rate_renter_details_renters_id" FOREIGN KEY (renters_id) REFERENCES access_level(id)
```

Dane inicjujące

```
-- Tworzenie konta Administratora  
INSERT INTO Account(id, login, password, name, surname, email,  
phone_number, confirmed, active, version) VALUES(nextval(  
'account_id_seq'), 'admin',  
'67e357d90262fd9f6dccdb11d18955204b8c02d12574bef45e0d6d0546b7377'  
, 'Ssbadmin', 'Ssbd', 'ssbd01admin@ias.pl', '1111111111', true,  
true, 1);  
  
INSERT INTO Access_level(id, access_level, active, account_id,  
version) VALUES(nextval('access_level_id_seq'), 'Admin', true, 1,  
1);  
  
INSERT INTO Admin_details(id) VALUES(1);  
  
-- Tworzenie konta Service providera  
INSERT INTO Account(id, login, password, name, surname, email,  
phone_number, confirmed, active, version) VALUES(nextval(  
'account_id_seq'), 'serviceProvider',  
'67e357d90262fd9f6dccdb11d18955204b8c02d12574bef45e0d6d0546b7377'  
, 'Ssbbserviceprovider', 'Ssbd', 'ssbd01serviceprovider@ias.pl',  
'2222222222', true, true, 1);  
  
INSERT INTO Access_level(id, access_level, active, account_id,  
version) VALUES(nextval('access_level_id_seq'), 'ServiceProvider'  
, true, 2, 1);  
  
INSERT INTO Service_provider_details(id, service_name, NIP,  
address, description, logo_url) VALUES(2, 'service', '0123456789'
```

```

, 'New York', 'Best services', 'url123');

INSERT INTO Rate(id, average_rate, rates_number, version,
service_provider_id) VALUES(nextval('rate_id_seq'), 0, 0, 1, 2);

-- Dodanie oferty
INSERT INTO Offer(id, title, description, price, active,
service_provider_id, version) VALUES(nextval('offer_id_seq'),
'Tytull1', 'Opis oferty', 100, true, 2, 1);

INSERT INTO Offer(id, title, description, price, active,
service_provider_id, version) VALUES(nextval('offer_id_seq'),
'Tytul2', 'Oferta 2', 200, true, 2, 1);

INSERT INTO Offer(id, title, description, price, active,
service_provider_id, version) VALUES(nextval('offer_id_seq'),
'Tytul3', 'Oferta 3', 350, true, 2, 1);

INSERT INTO Offer(id, title, description, price, active,
service_provider_id, version) VALUES(nextval('offer_id_seq'),
'Tytul4', 'Oferta 4', 520, true, 2, 1);

-- Tworzenie konta Rentera
INSERT INTO Account(id, login, password, name, surname, email,
phone_number, confirmed, active, version) VALUES(nextval(
'account_id_seq'), 'renter',
'67e357d90262fd9f6dccdb11d18955204b8c02d12574bef45e0d6d0546b7377',
,'Ssbdrenter', 'Ssbd', 'ssbd01renter@ias.pl', '333333333', true,
true, 1);

INSERT INTO Access_level(id, access_level, active, account_id,
version) VALUES(nextval('access_level_id_seq'), 'Renter', true, 3
, 1);

INSERT INTO Renter_details(id, user_name) VALUES(3, 'renter1');

INSERT INTO Offer_date(id, date, version, offer_id) VALUES(nextval(
'offer_date_id_seq'), '2020-03-21', 1, 1);

INSERT INTO User_offer(id, version, renter_id, offer_date_id)
VALUES(nextval('user_offer_id_seq'), 1, 3, 1);

-- Tworzenie konta Rentera
INSERT INTO Account(id, login, password, name, surname, email,
phone_number, confirmed, active, version) VALUES(nextval(
'account_id_seq'), 'renter2',
'67e357d90262fd9f6dccdb11d18955204b8c02d12574bef45e0d6d0546b7377',
,'Ssbdrenter', 'Ssbd', 'ssbd01renter2@ias.pl', '333333333', true,
true, 1);

INSERT INTO Access_level(id, access_level, active, account_id,
version) VALUES(nextval('access_level_id_seq'), 'Renter', true, 4
, 1);

INSERT INTO Renter_details(id, user_name) VALUES(4, 'renter2');

-- Tworzenie konta Rentera
INSERT INTO Account(id, login, password, name, surname, email,
phone_number, confirmed, active, version) VALUES(nextval(
'account_id_seq'), 'renter3',
'67e357d90262fd9f6dccdb11d18955204b8c02d12574bef45e0d6d0546b7377',
,'Ssbdrenter', 'Ssbd', 'ssbd01renter3@ias.pl', '333333333', true,
true, 1);

INSERT INTO Access_level(id, access_level, active, account_id,
version) VALUES(nextval('access_level_id_seq'), 'Renter', true, 5
, 1);

INSERT INTO Renter_details(id, user_name) VALUES(5, 'renter3');

```

4 Użytkownicy bazodanowi 2 pkt

Lista użytkowników bazodanowych:

- ssbd01 - użytkownik bazodanowy który jest właścicielem tabel
- ssbd01admin - administrator bazy danych
- ssbd01auth - użytkownik który wykorzystywany będzie na potrzeby uwierzytelniania użytkowników w wielodostępnej aplikacji
- ssbd01mok - konto stworzone na potrzeby modułu obsługi konta *MOK*
- ssbd01mo - konto stworzone na potrzeby modułu ogłoszeń *MO*
- ssbd01mz - konto stworzone na potrzeby modułu zgłaszającego *MZ*
- ssbd01mw - konto stworzone na potrzeby modułu wynajmującego *MW*

Tabela uprawnień użytkowników bazodanowych.

| Nazwa tabeli | Uprawnienia użytkowników bazodanowych |
|--------------------------|---|
| account | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |
| access_level | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |
| renter_details | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |
| service_provider_details | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |

| | |
|--------------------|---|
| admin_details | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |
| offer | ssbd01 = arwd ssbd01admin = arwd ssbd01mok = r ssbd01mo = r ssbd01mz = arwd ssbd01mw = r |
| offer_date | ssbd01 = arwd ssbd01admin = arwd ssbd01mok = r ssbd01mo = r ssbd01mz = r ssbd01mw = ar |
| user_offer | ssbd01 = arwd ssbd01admin = arwd ssbd01mok = r ssbd01mo = r ssbd01mz = r ssbd01mw = ar |
| rate | ssbd01 = arwd ssbd01admin = arwd ssbd01mok = r ssbd01mo = r ssbd01mz = r ssbd01mw = arwd |
| verification_token | ssbd01 = arwd ssbd01admin = arwd ssbd01auth = r ssbd01mok = arwd ssbd01mo = r ssbd01mz = r ssbd01mw = r |

Legenda akronimów praw dostępu:

r - "read" - SELECT - umożliwia odczyt danych z tabeli.

a - "append" - INSERT - umożliwia dodawanie nowych wierszy w tabeli.

w - "write" - UPDATE - umożliwia aktualizację wierszy w tabeli.

d - "delete" - DELETE - umożliwia usuwanie wierszy w tabeli.

5 Identyfikacja obiektów encji 2 pkt

Zestawienie klas encji

| Obiekt encji | Tabele | Klucz główny | Typ |
|------------------------|--------------------------|--------------|------|
| Account | Account | Id | Long |
| AccessLevel | Access_level | Id | Long |
| RenterDetails | Renter_details | Id | Long |
| ServiceProviderDetails | Service_provider_details | Id | Long |
| AdminDetails | Admin_details | Id | Long |
| UserOffer | User_offers | Id | Long |
| OfferDate | Offer_date | Id | Long |
| Offer | Offers | Id | Long |
| Rate | Rates | Id | Long |
| VerificationToken | Verification_token | Id | Long |

Związki między obiektami klas encji

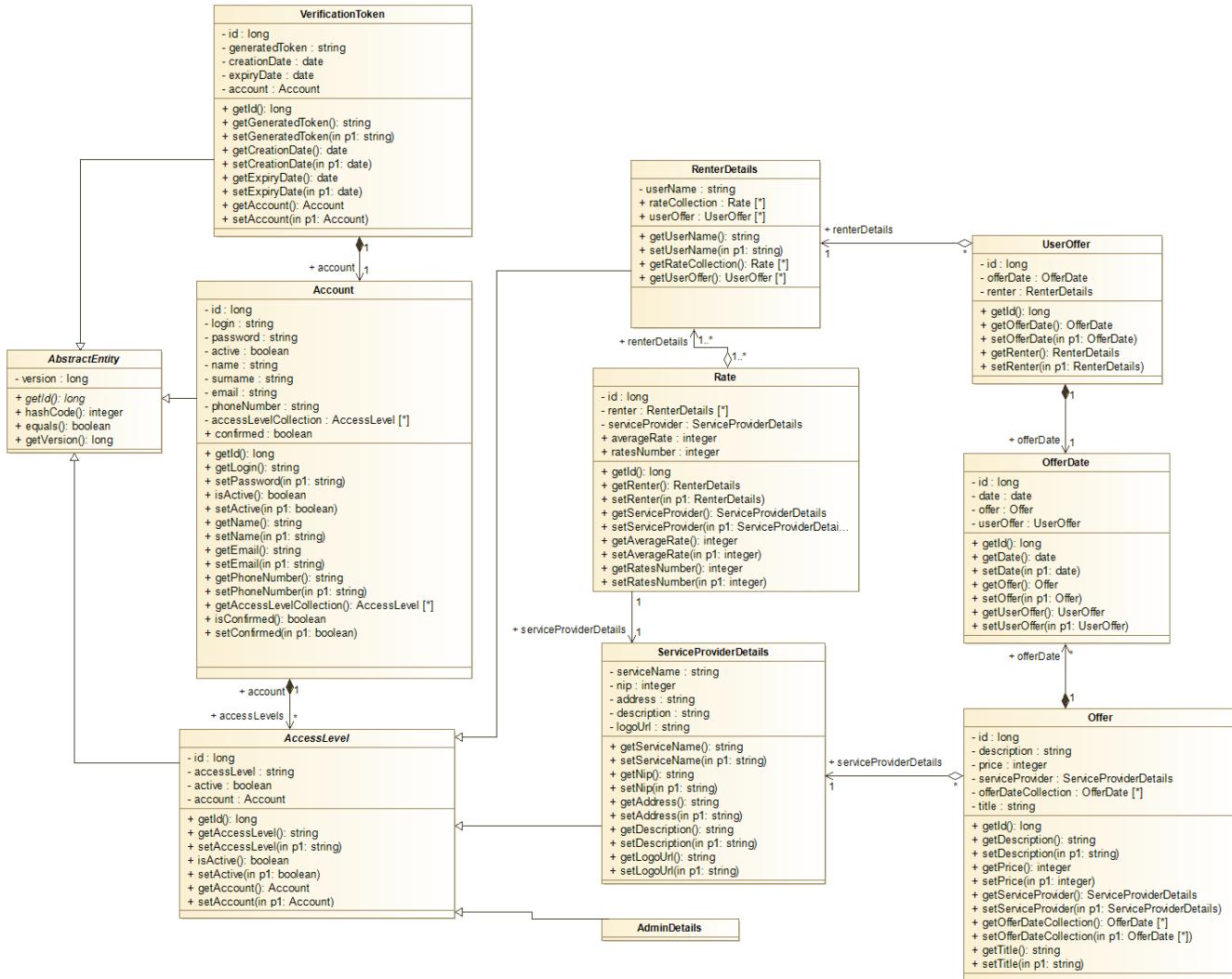
| | Account | AccessLevel | RenterDetails | ServiceProviderDetails | Rate | UserOffer | OfferDate | Offer | AdminDetails | VerificationToken |
|------------------------|----------------------|----------------------|---------------|------------------------|-------------------------|----------------------|----------------------|-------------------------|--------------|----------------------|
| Account | | W 2K [1] (P, R) T | | | | | | | | K 1K [1] (P, R) N |
| AccessLevel | K 2K [0..*] (P) T | | | | | | | | | |
| RenterDetails | | | | | K 1K [0..1] (P, R) N | K 1K [1] (P, R) N | | | | |
| ServiceProviderDetails | | | | | K 1K [1] (P, R) N | | | K 1K (P, R) N | | |
| Rate | | | W 1K [1] N | W 1K [1] (P, R) N | | | | | | |
| UserOffer | | | W 1K [1] N | | | | W 1K [1] N | | | |
| OfferDate | | | | | | K 1K [1] (P, R) N | | K 1K [0..*] (P, R) N | | |
| Offer | | | | W 1K [1] (P, R) N | | | W 1K [1] (P, R) N | | | |
| AdminDetails | | | | | | | | | | |
| VerificationToken | W 1K [1] N | | | | | | | | | |

Legenda:

- W - właściciel związku znajduje się w wierszu
- K - właściciel związku znajduje się w kolumnie
- 1K - związek jednokierunkowy
- 2K - związek dwukierunkowy
- [...] - Liczność związku
- (...) - operacje kaskadowe w związku, gdzie R - REMOVE, P - PERSIST, RF - REFRESH, M - MERGE
- T - związek od momentu wiązania może ulec zmianie
- N - związek od momentu wiązania nie może ulec zmianie

6 Diagram klas encji 1 pkt

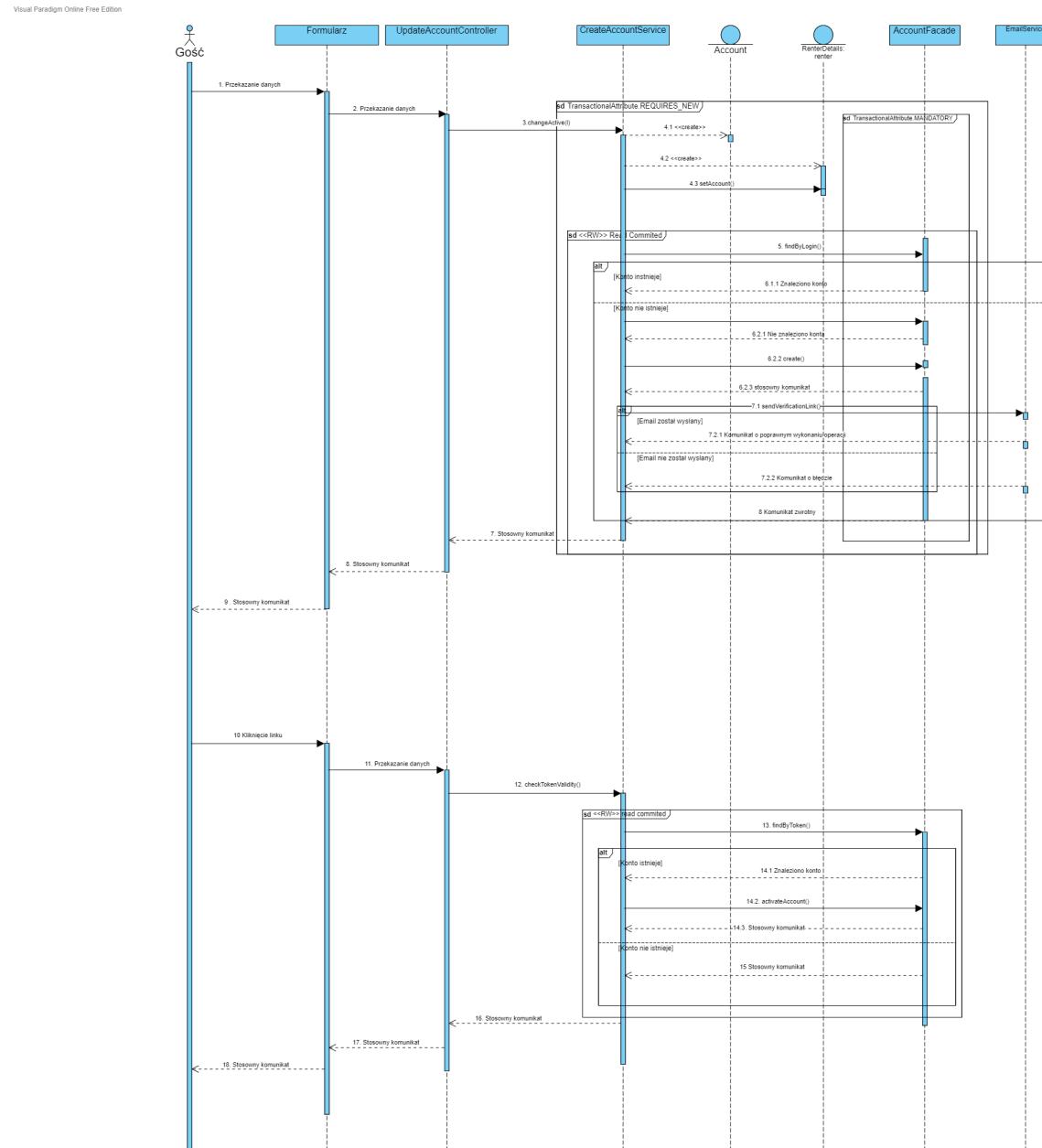
Diagram klas encji



Obraz 1. Diagram klas encyjnych dla modułu MOK.

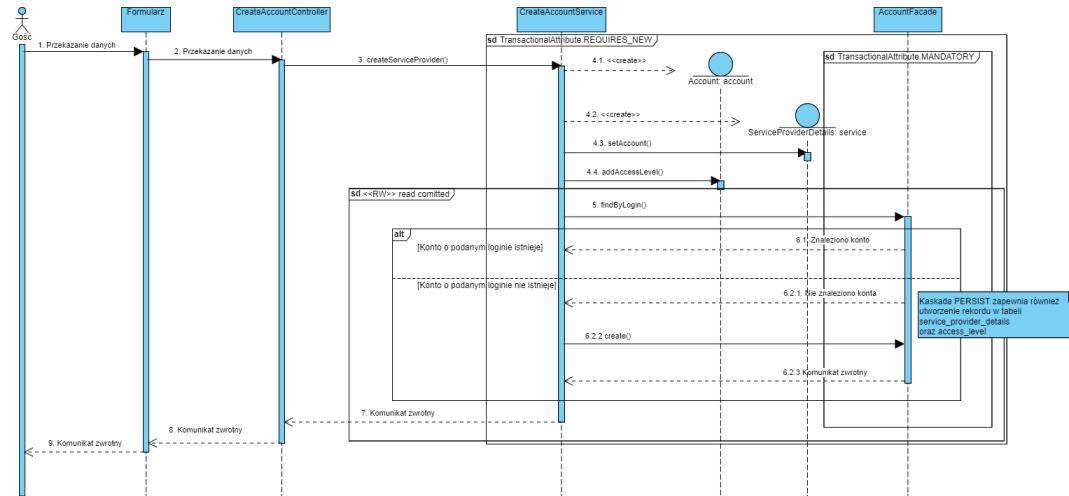
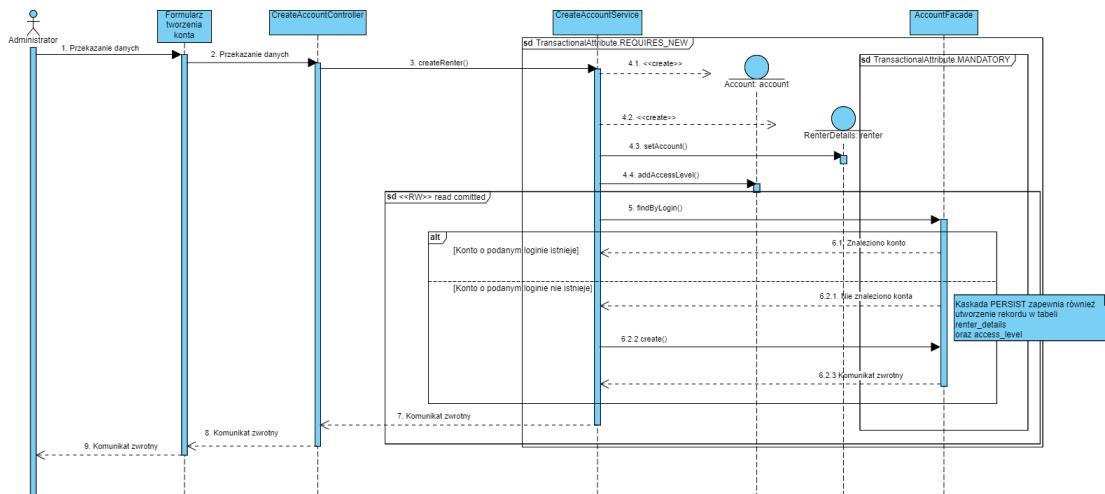
7 Diagramy sekwencji oraz identyfikacja transakcji bazodanowych 3 pkt

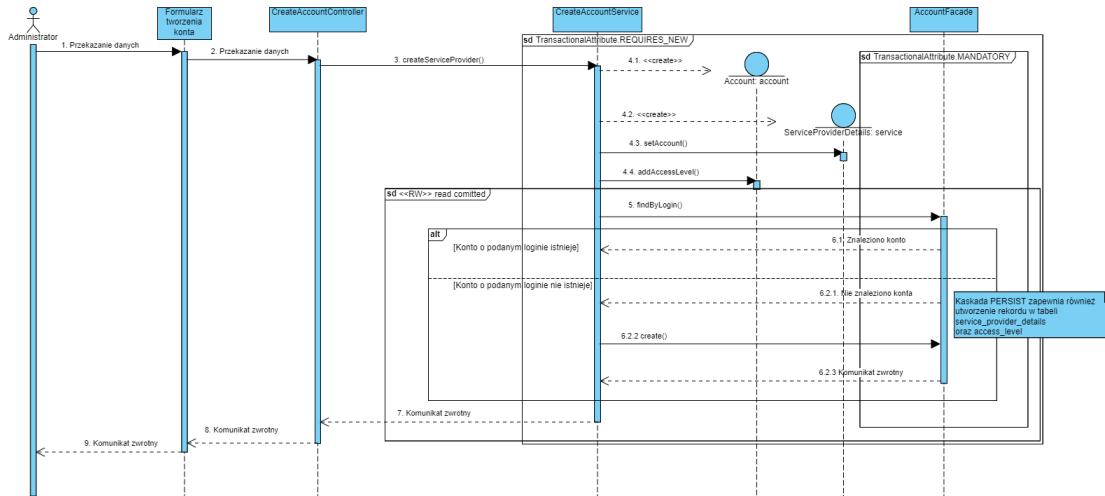
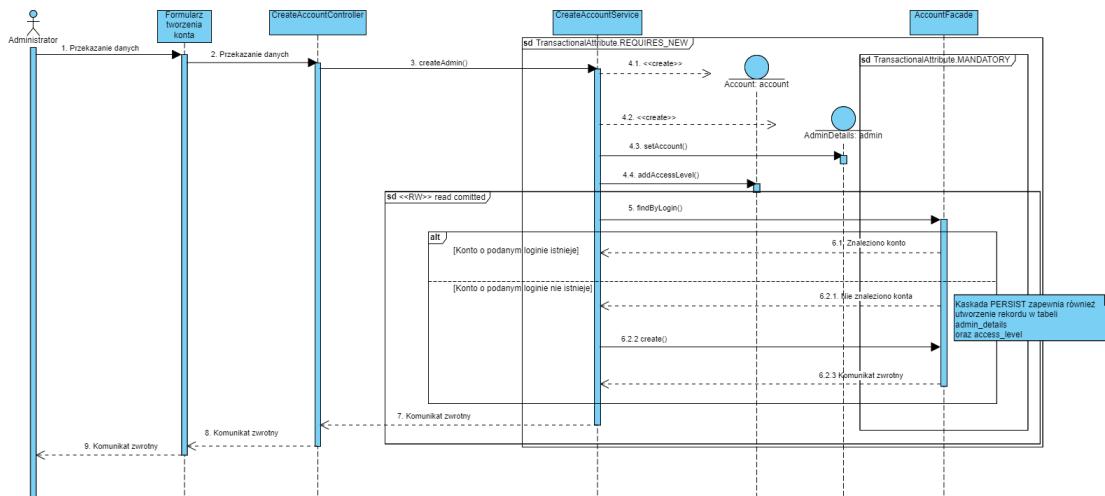
MOK.1.1 (Rejestracja użytkownika wynajmującego usługi)

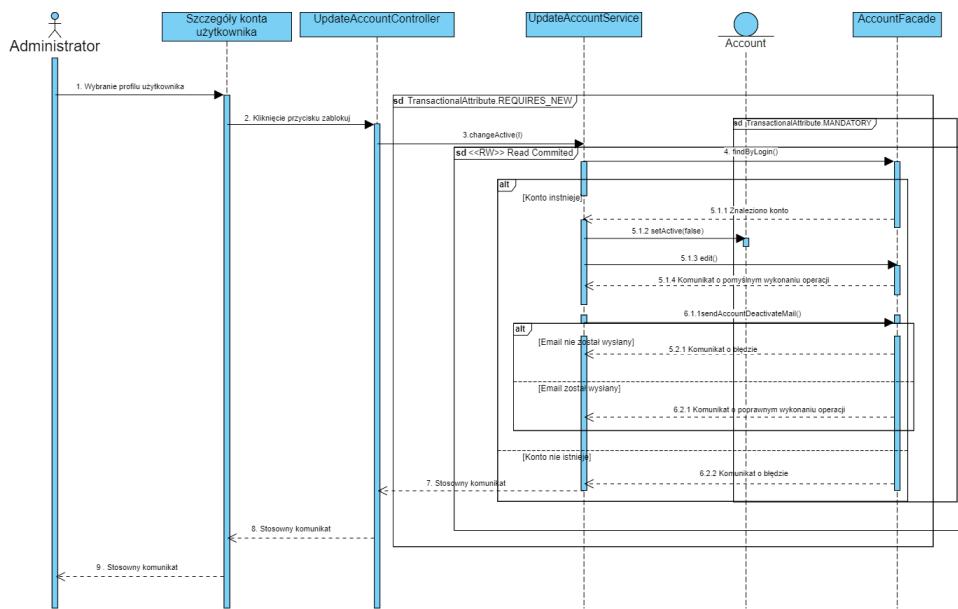
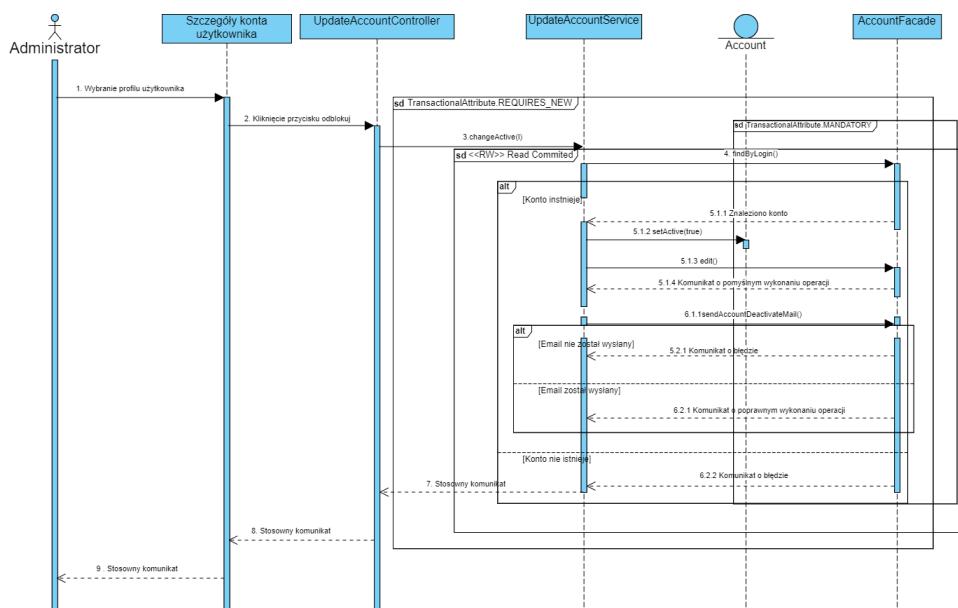


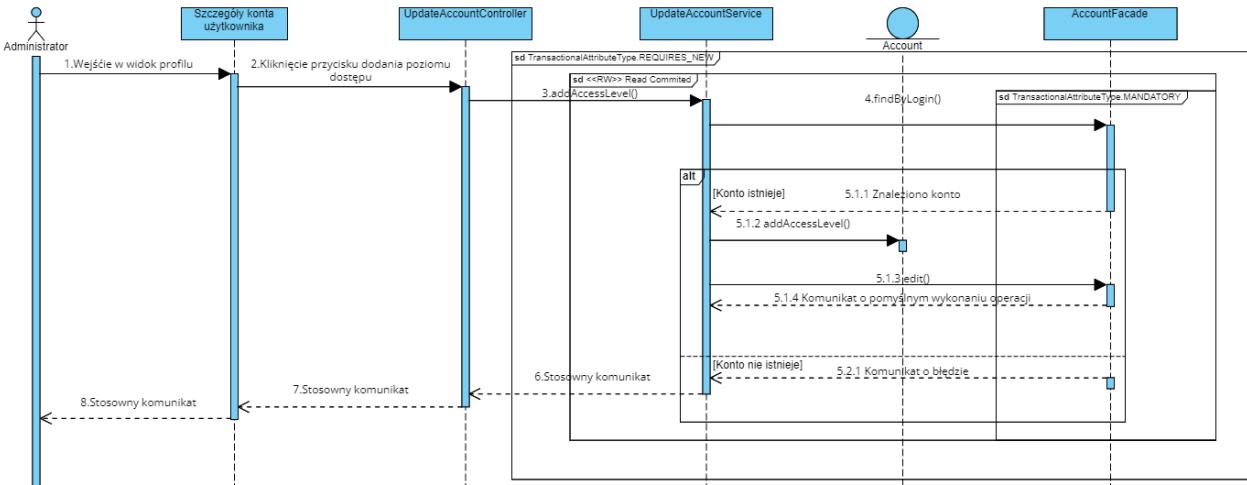
Visual Paradigm Online Free Edition

MOK.1.2 (Rejestracja użytkownika zapewniającego usługi)

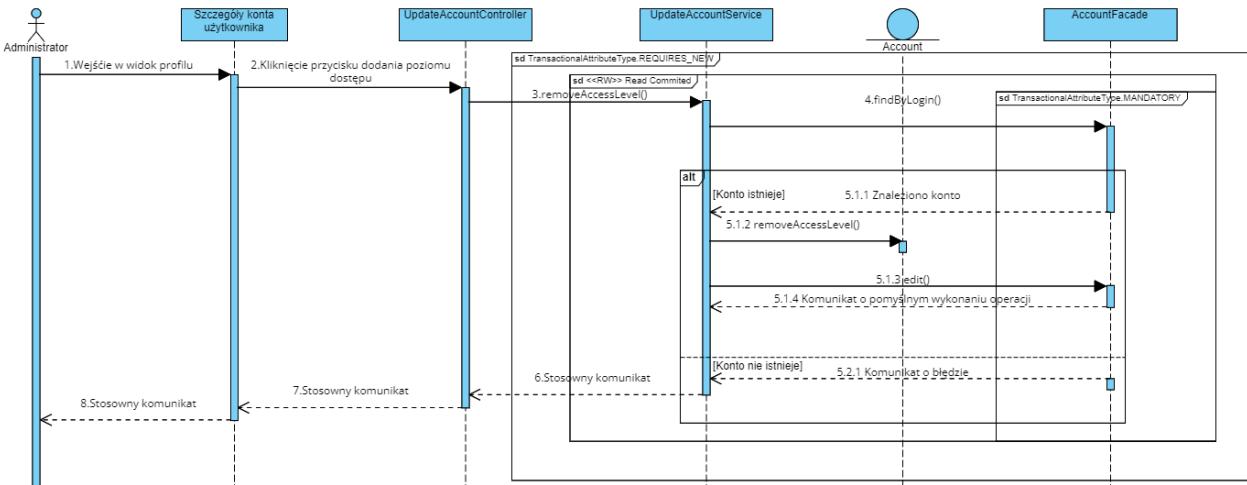
**MOK.2.1 (Utworzenie przez administratora konta użytkownika wynajmującego usługi)****MOK.2.2 (Utworzenie przez administratora konta użytkownika zapewniającego usługi)**

**MOK.2.3 (Utworzenie przez administratora konta administratora)****MOK.3**

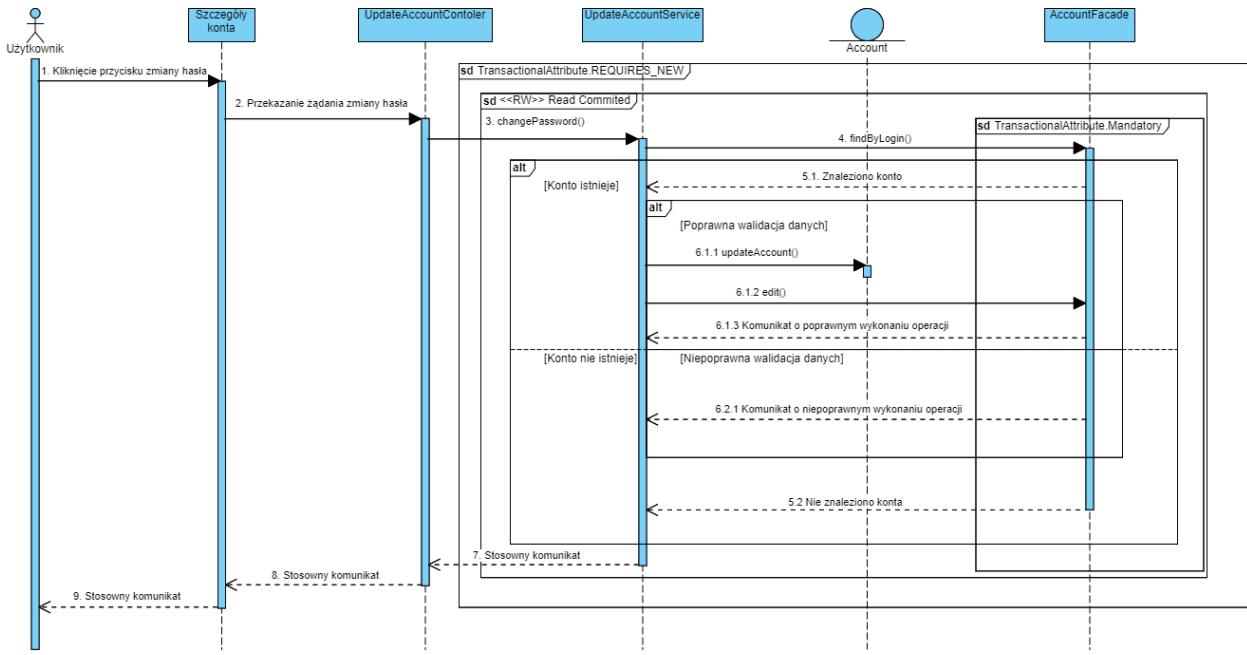
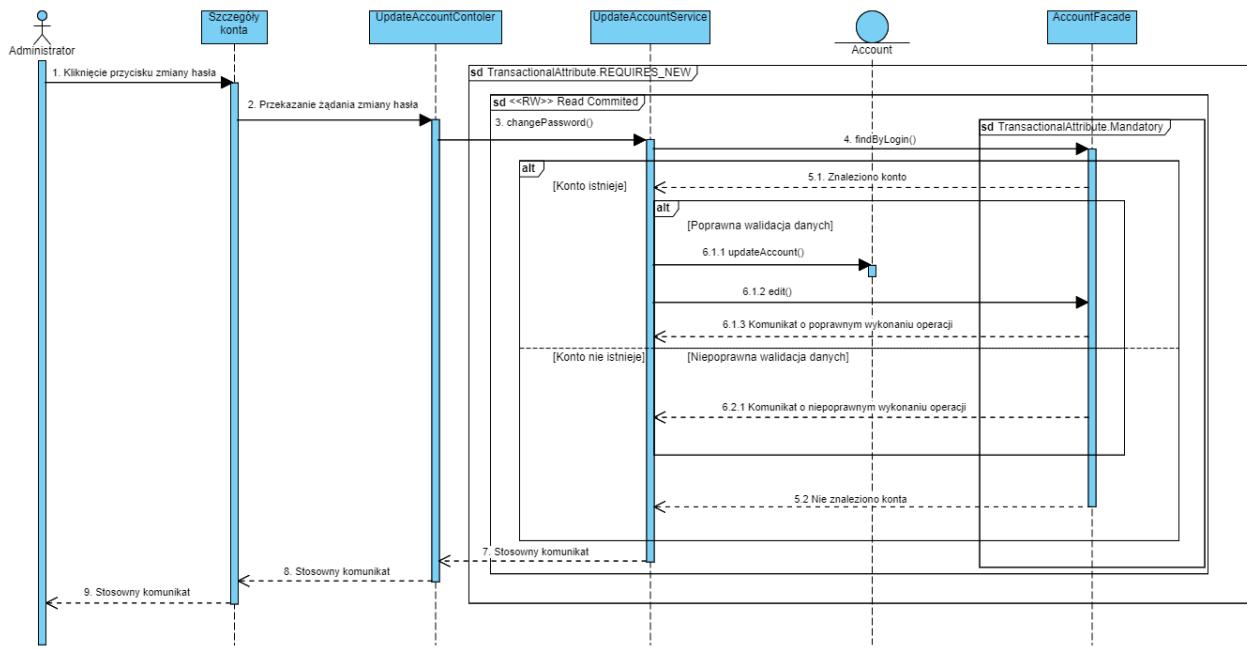
**MOK.4****MOK.5**

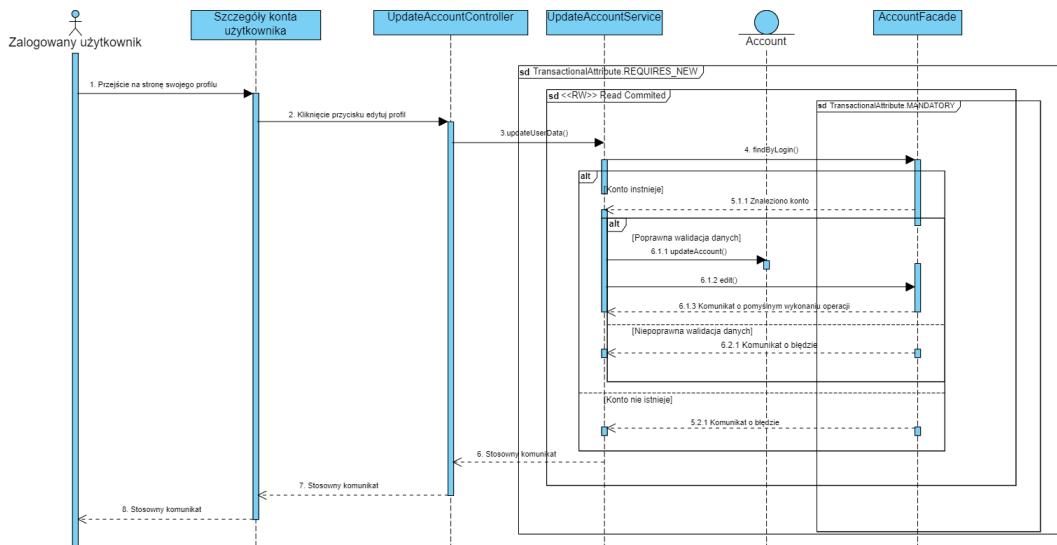
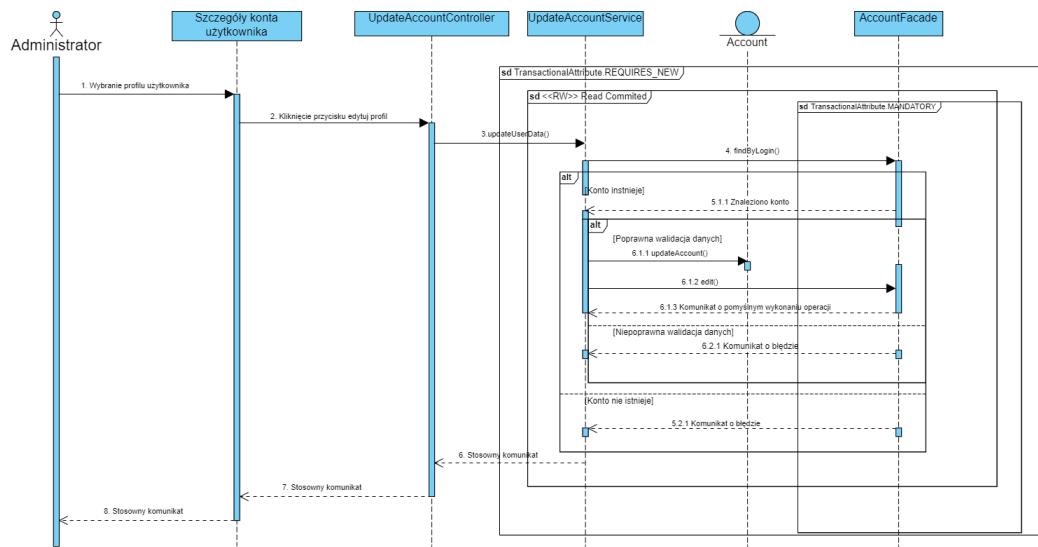


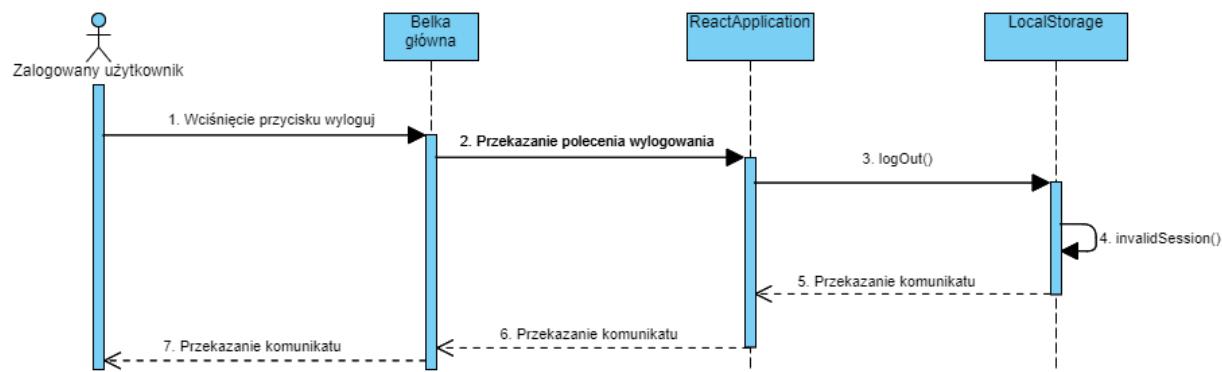
MOK.6



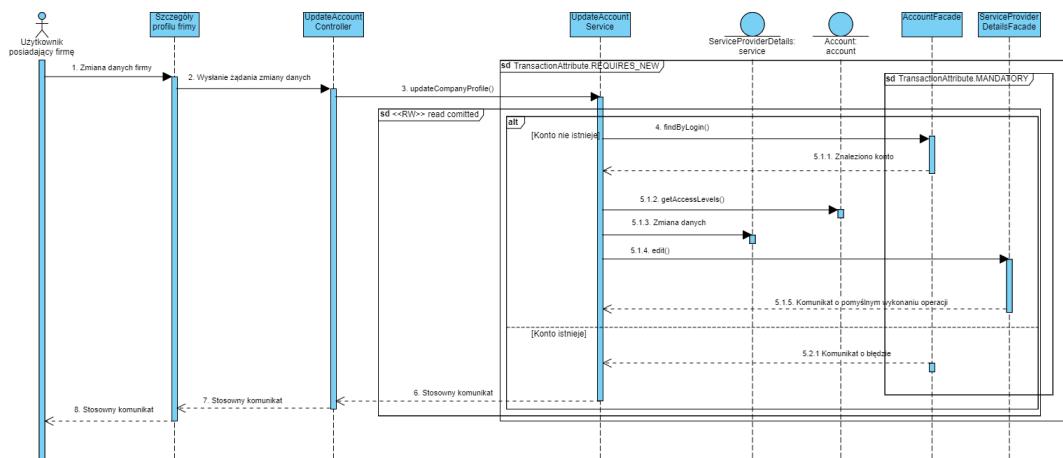
MOK.7

**MOK.8****MOK.9**

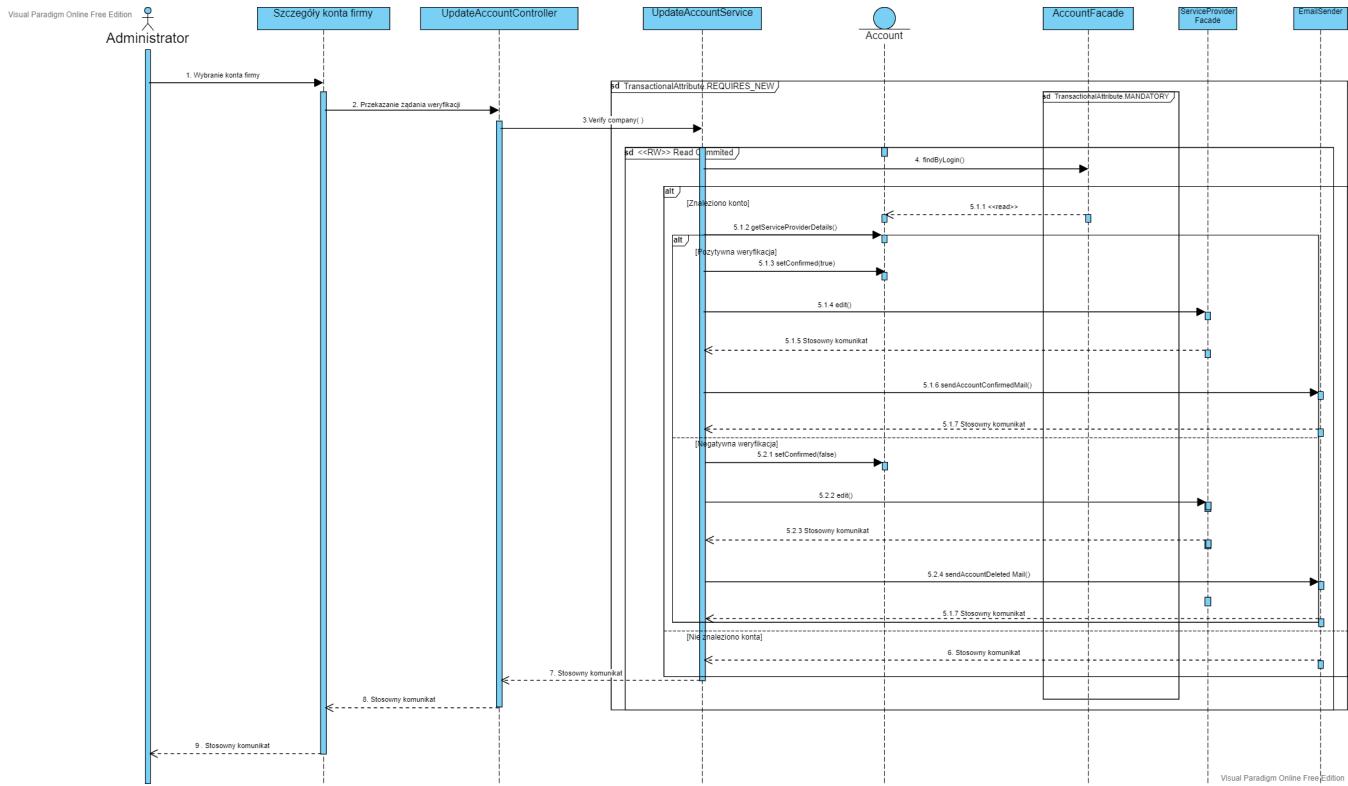
**MOK.10****MOK.11**



MOK.12

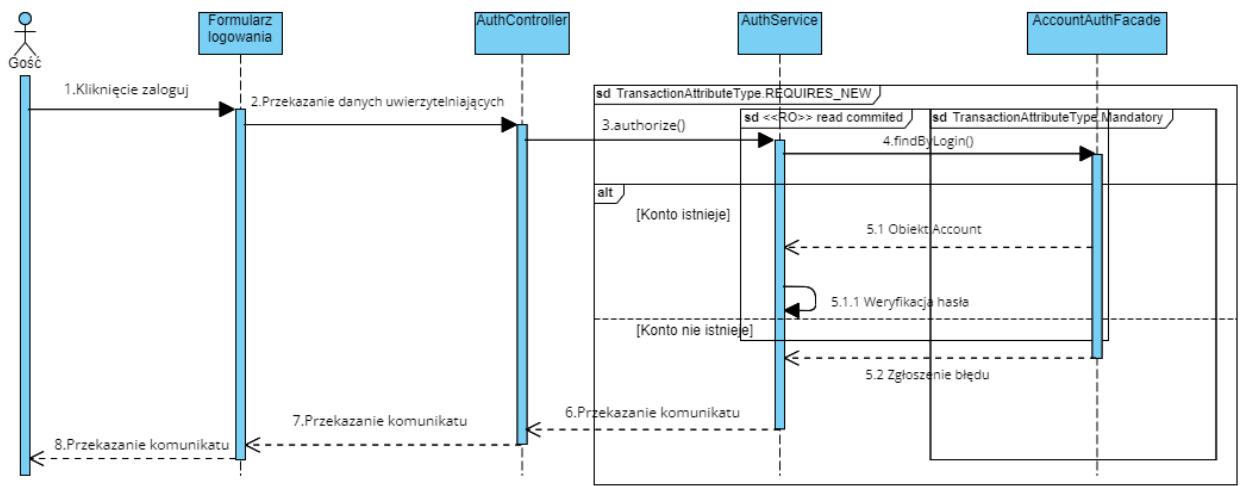


MOK.13

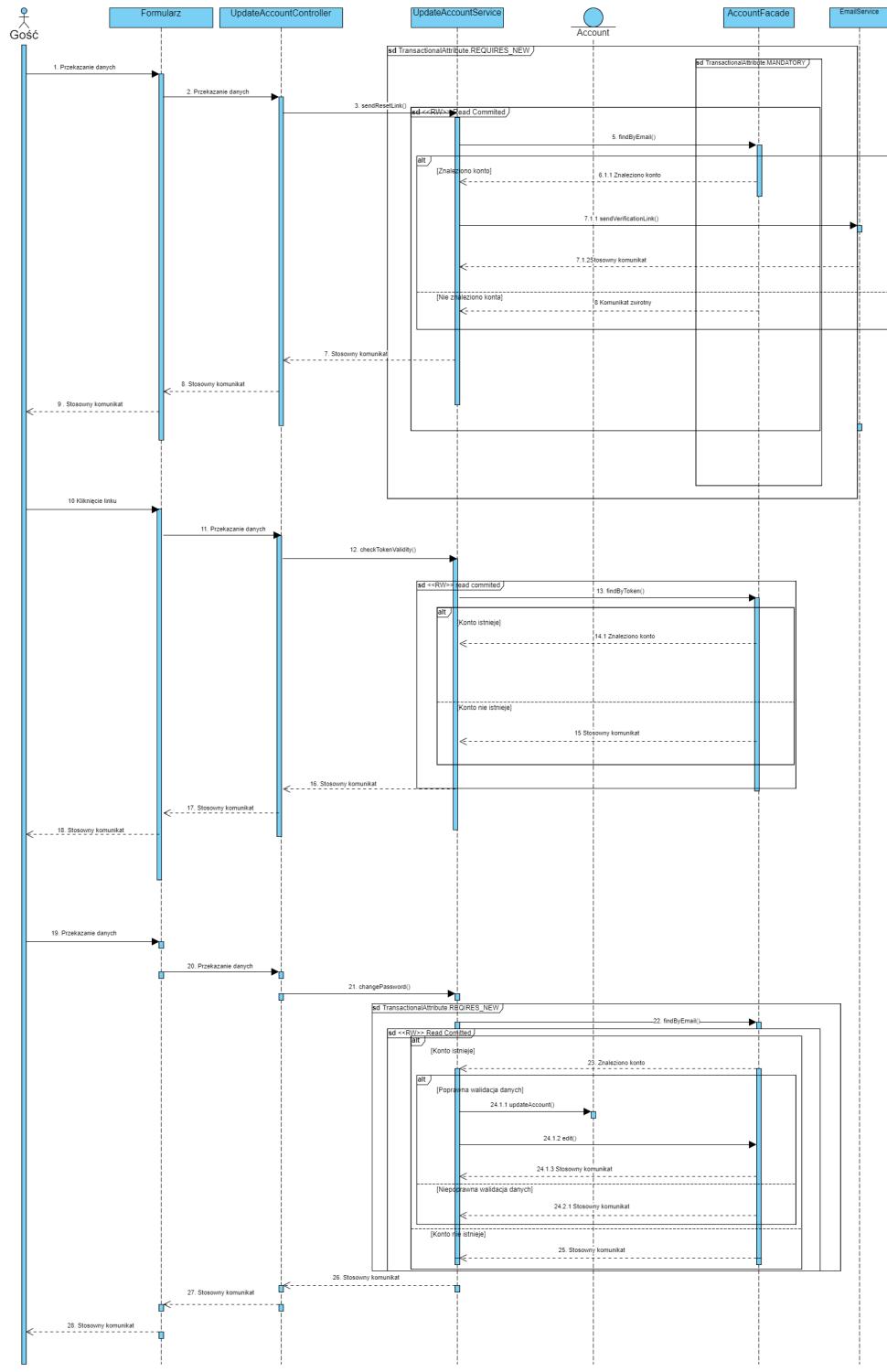


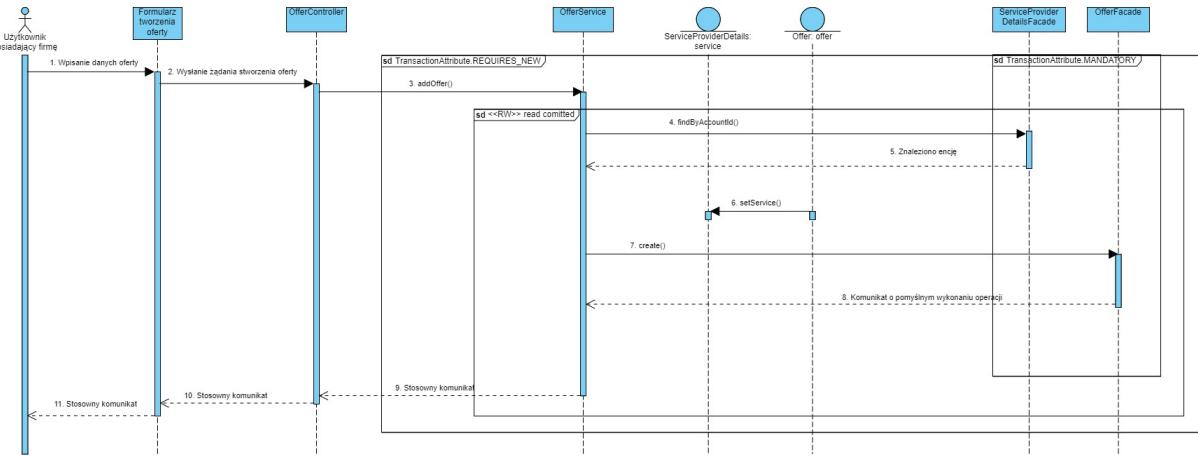
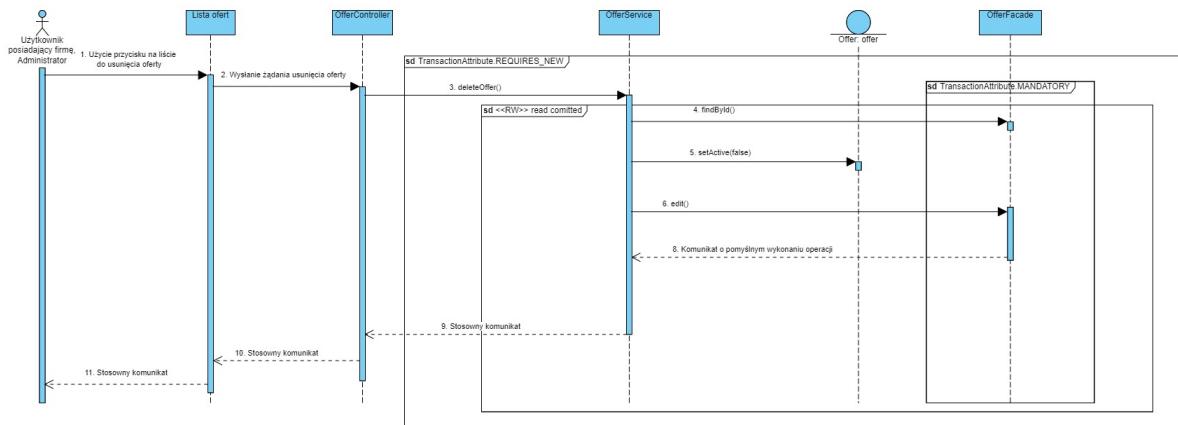
MOK.14

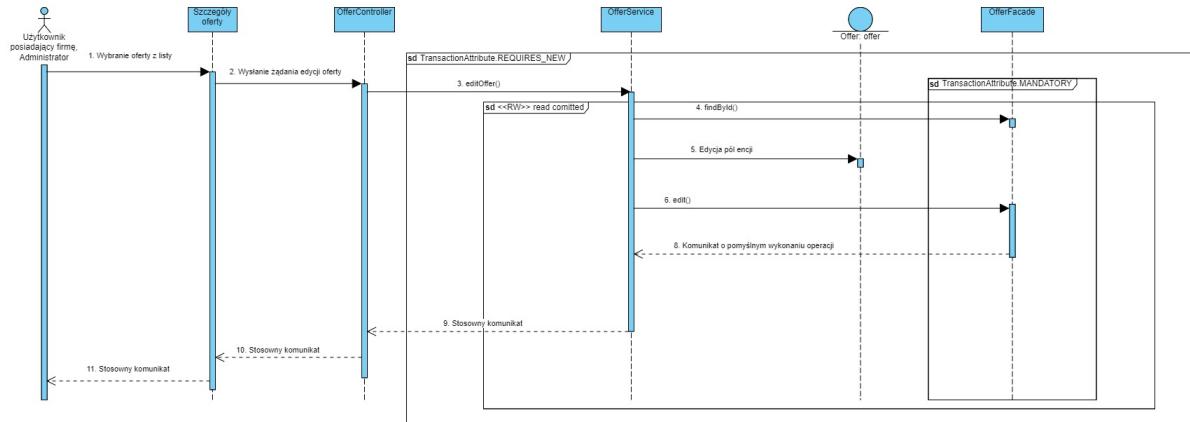
Visual Paradigm Online Free Edition



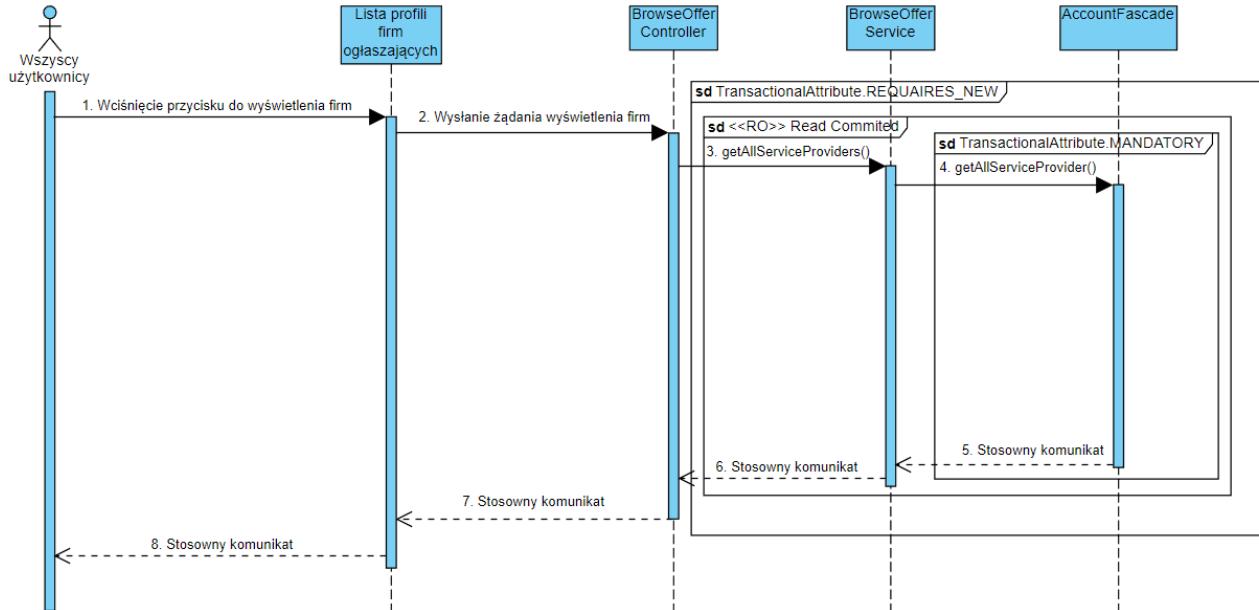
MOK. 15



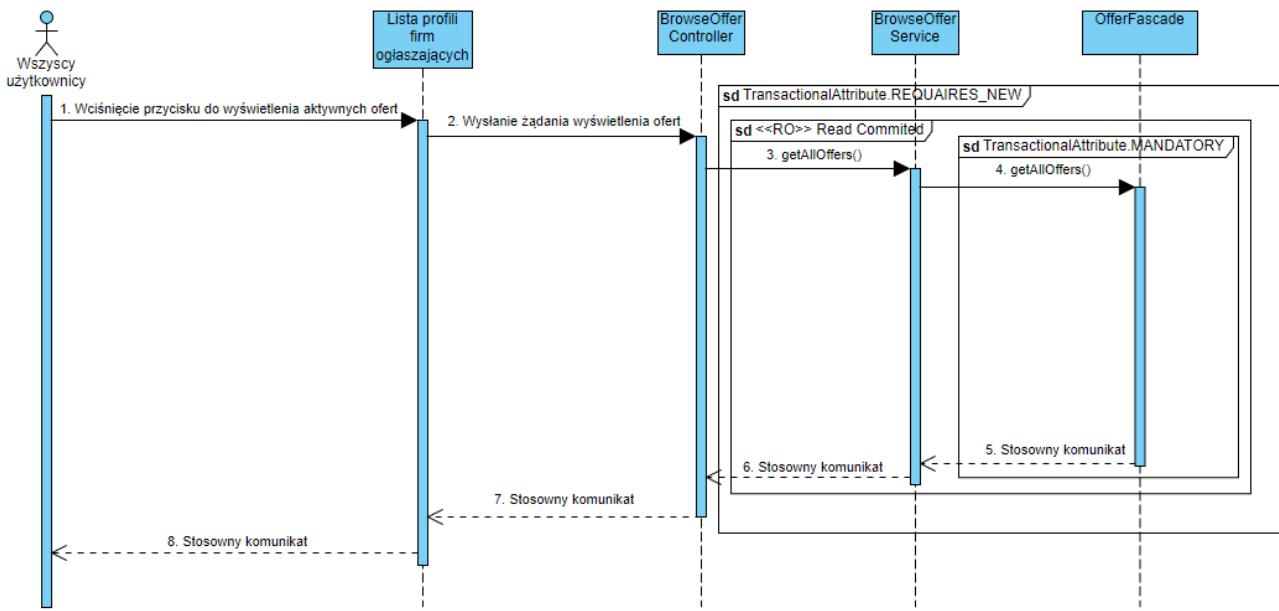
**MZ.2****MZ.3**



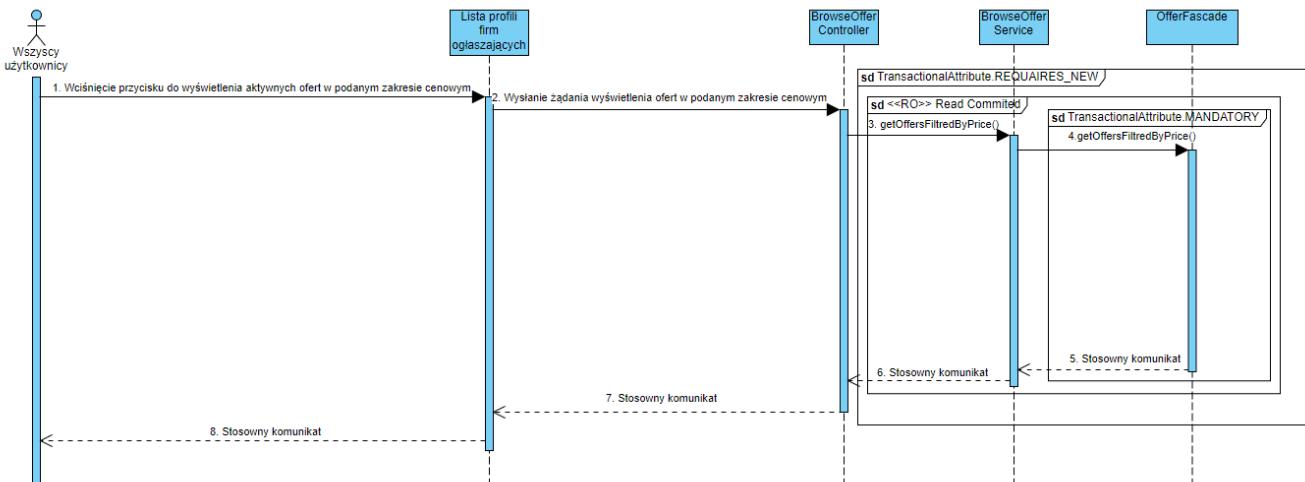
MO.1



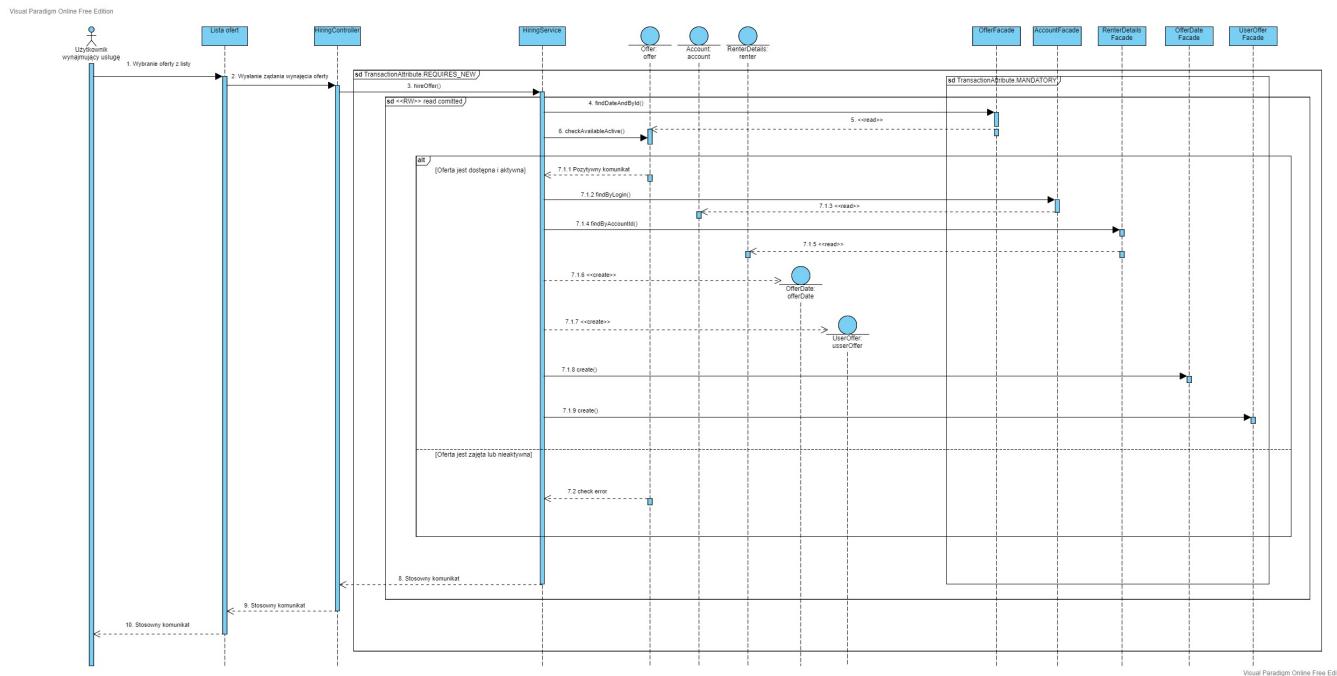
MO.2



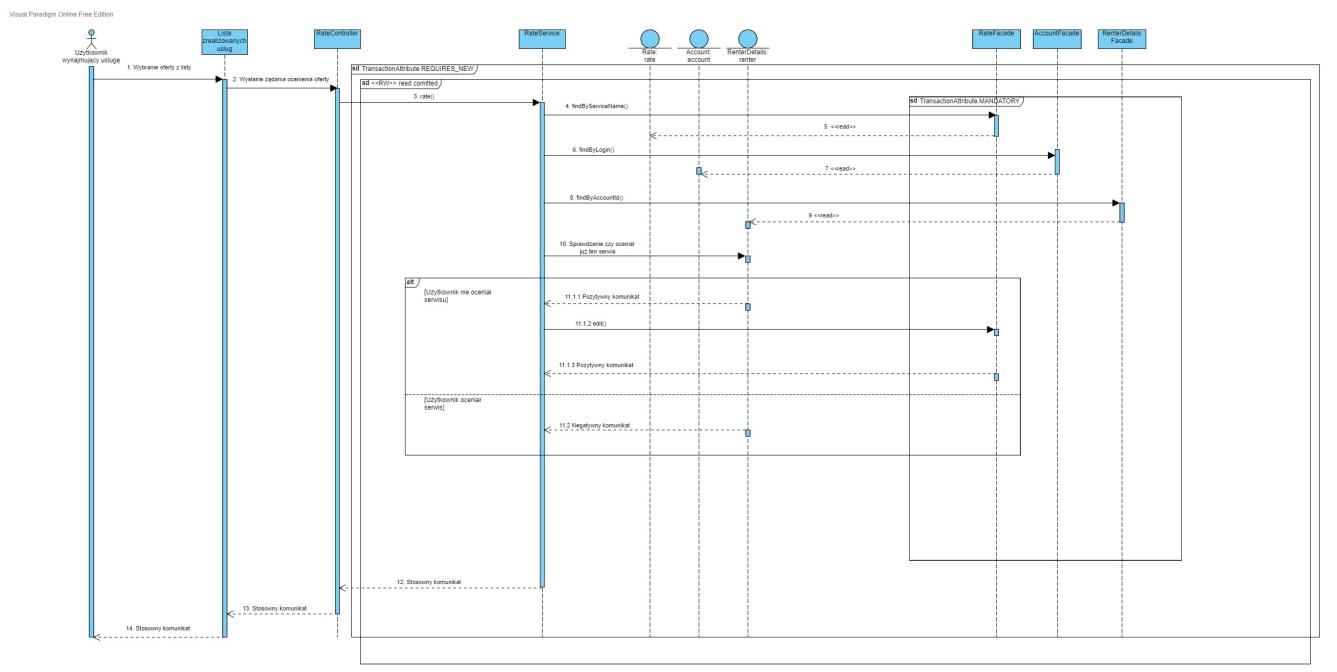
MO.3



MW.1



MW.2



8 Konfiguracja uwierzytelniania w aplikacji 1 pkt

Konfiguracja uwierzytelnienia w aplikacji - implementacja

Proces uwierzytelnienia w aplikacji implementuje klasa AuthController, która najpierw sprawdza czy login wprowadzony przez osobę próbującą się zalogować jest powiązany z istniejącym w bazie danych kontem a następnie porównuje skrót z wprowadzonego hasła ze skrótem zapisanym w bazie danych. Jeżeli operacja uwierzytelniania się powiedzie, generowany jest JWT, za co odpowiedzialna jest klasa JWTGeneratorVerifier. Okres bezczynności uwierzytelnionego użytkownika, po upływie którego nastąpi automatyczne zakończenie sesji użytkownika został ustawiony na 30 minut.

```
@Path("/auth")
@RequestScoped
public class AuthController {

    @Inject
    private IdentityStoreHandler identityStoreHandler;

    @Inject
    private ReadAccountService readAccountService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.TEXT_PLAIN)
    public Response authenticate(@Valid AuthDTO authDTO) {
        Credential credential = new UsernamePasswordCredential(authDTO.getLogin(), new Password(authDTO.getPassword()));
        CredentialValidationResult result = identityStoreHandler.validate(credential);

        if (result.getStatus() == CredentialValidationResult.Status.VALID) {
            return Response
                .accepted()
                .entity(JWTGeneratorVerifier.generateJWTString(result))
                .build();
        } else {
            return Response
                .status(Response.Status.UNAUTHORIZED)
                .build();
        }
    }

    @GET
    @Path("/{_self}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response findSelf(@Context SecurityContext securityContext) {
        return Response
            .ok()
            .entity(readAccountService.readAccountByLogin(securityContext.getUserPrincipal().getName()))
            .build();
    }
}
```

Obraz 1. Implementacja klasy AuthController

```

public class JWTGeneratorVerifier {

    private static final String SECRET =
    private static final long JWT_TIMEOUT_MS = 30 * 60 * 1000;

    public static String generateJWTString(CredentialValidationResult result) {
        //The JWT signature algorithm we will be using to sign the token
        SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;

        long nowMillis = System.currentTimeMillis();
        Date now = new Date(nowMillis);

        //We will sign our JWT with our ApiKey secret
        byte[] apiKeySecretBytes = DatatypeConverter.parseBase64Binary(SECRET);
        Key signingKey = new SecretKeySpec(apiKeySecretBytes, signatureAlgorithm.getJcaName());

        //Let's set the JWT Claims
        JwtBuilder builder = Jwts.builder()
            .setSubject(result.getCallerPrincipal().getName())
            .claim("auth", String.join(", ", result.getCallerGroups()))
            .setIssuer("SSBD01 Chads")
            .setExpiration(new Date(new Date().getTime() + JWT_TIMEOUT_MS))
            .setIssuedAt(now)
            .signWith(signatureAlgorithm, signingKey)
            .setHeaderParam("typ", "JWT");

        //Builds the JWT and serializes it to a compact, URL-safe string
        return builder.compact();
    }

    public static boolean validateJWTSignature(String token) {
        try {
            Jwts.parser().setSigningKey(SECRET).parseClaimsJws(token);
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public static Claims decodeJWT(String jwt) {
        //This line will throw an exception if it is not a signed JWS (as expected)
        return Jwts.parser()
            .setSigningKey(DatatypeConverter.parseBase64Binary(SECRET))
            .parseClaimsJws(jwt).getBody();
    }
}

```

Obraz 2. Implementacja klasy JWTGeneratorVerifier

W klasie JWTGeneratorVerifier zostały określone parametry JWT. Jest on tworzony w momencie wywołania metody generateJWTString, jego okres ważności został ustawiony na 30 minut i zawiera:

- nazwę użytkownika
- grupę/grupy użytkownika
- emitenta żetonu
- datę wygaśnięcia żetonu
- datę wystawienia żetonu
- algorytm służący do wygenerowania sygnatury żetonu

- parametr nagłówka

Konfiguracja uwierzytelnienia w aplikacji - konfiguracja

Konfiguracja uwierzytelnienia w aplikacji jest zapisana w pliku JDBCConfiguration.java, określa ona pulę połączeń na potrzeby implementacji uwierzytelnienia w aplikacji

```
@DataSourceDefinition(
    name = "java:app/jdbc/ssbd01auth",
    className = "org.postgresql.ds.PGSimpleDataSource",
    user = "ssbd01auth",
    password = "auth",
    //      serverName = "studdev.it.p.lodz.pl",
    serverName = "127.0.0.1",
    portNumber = 5432,
    databaseName = "ssbd01"
)
```

Obraz 3. Pula połączeń na potrzeby implementacji uwierzytelniania w aplikacji

Podpis

Do podpisu wykorzystany został algorytm HS256:

```
byte[] apiKeySecretBytes = DatatypeConverter.parseBase64Binary(SECRET);

SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;

Key signingKey = new SecretKeySpec(apiKeySecretBytes, signatureAlgorithm.getJcaName());

//Let's set the JWT Claims

JwtBuilder builder = Jwts.builder()

    .setSubject(username)

    .claim("auth", String.join(", ", callerGroups))

    .setIssuer("SSBD01 Chads")

    .setIssuedAt(now)

    .setExpiration(new Date(new Date().getTime() + ACCESS_JWT_TIMEOUT_MS))

    .signWith(signatureAlgorithm, signingKey)

    .setHeaderParam("typ", "JWT");
```

Procedury zarządzania kontami

- Rejestracja kont

Konto z poziomem dostępu wynajmującego lub zgłaszającego mogą zostać utworzone samodzielnie poprzez wypełnienie stosownego formularza. Konto administratora może zostać utworzone jedynie przez innego administratora.

Utworzenie nowego konta powoduje utworzeniem trzech nowych krotek w bazie danych: w tabelach: Account, Access_level oraz w zależności od poziomu dostępu nowo utworzonego konta w tabeli Admin_details, Renter_details lub Service_provider_details.

- Weryfikacji konta po samodzielnej rejestracji

Po samodzielnej rejestracji konto z poziomem dostępu wynajmujący oraz zgłaszający wymagają weryfikacji. Na podany podczas rejestracji adres email jest wysyłana wiadomość zawierająca link ważny 24 godziny po kliknięciu którego konto zostaje zweryfikowane. Ponadto konto użytkownika zgłoszającego musi zostać dodatkowo zweryfikowane przez konto administratora, do czasu zweryfikowania pozostaje ono zablokowane.

Zweryfikowanie konta użytkownika zgłoszającego i wynajmującego powoduje zmianę rekordu confirmed w tabeli Account z wartości false na true. Zweryfikowanie przez administratora konta użytkownika zgłoszającego powoduje zmianę rekordu active w tabeli Account z wartości false na true.

- Usuwanie nieaktywnych kont po upływie limitu czasu na aktywację zarejestrowanego konta

Po samodzielnej rejestracji konta jest ono początkowo niepotwierdzone. Na podany podczas rejestracji adres email jest wysyłana wiadomość zawierająca link ważny 24 godziny po kliknięciu którego konto zostaje zweryfikowane. Jeżeli po upływie 24 godzin od wysłania maila konto wciąż pozostaje niepotwierdzone to jest ono usuwane z bazy danych.

Potwierdzone konto ma zmieniany rekord confirmed w tabeli Account z wartości false na true.

- Blokowanie i odblokowanie konta

Każde z kont może zostać tymczasowo lub stale zablokowane. Operacja zablokowania i odblokowania konta jest możliwa jedynie przez konto z poziomem dostępu administratora.

Odzwierciedleniem stanu konta w bazie danych jest wartość rekordu active w tabeli Account, wartość true oznacza, że jest odblokowane a wartość false, że jest zablokowane.

- Przydzielanie i odbieranie poziomów dostępu

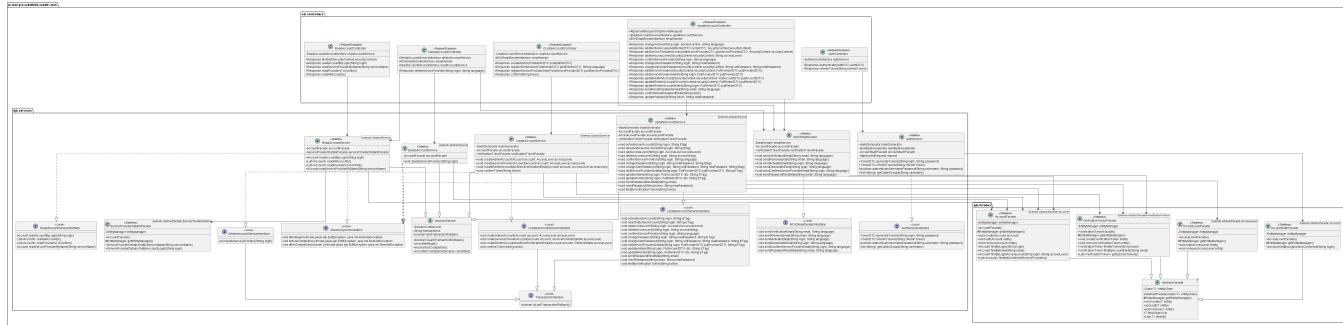
Każde z kont może mieć zmodyfikowany poziom dostępu. Jedynie administrator może przydzielać oraz odbierać poszczególne poziomy dostępu.

Przyznanie lub odebranie poziomu dostępu powoduje dodanie lub usunięcie krótki danych w tabeli Access_level.

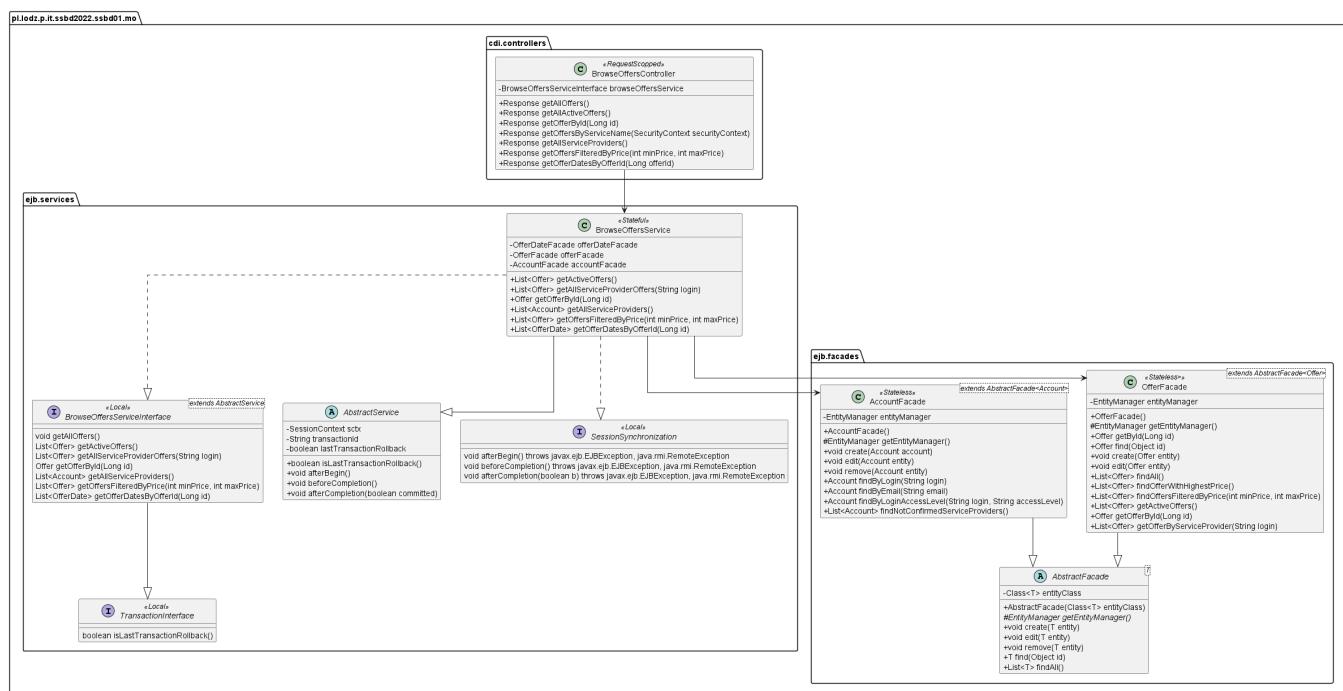
Sprawozdanie szczegółowe (25 pkt)

Sprawozdanie szczegółowe

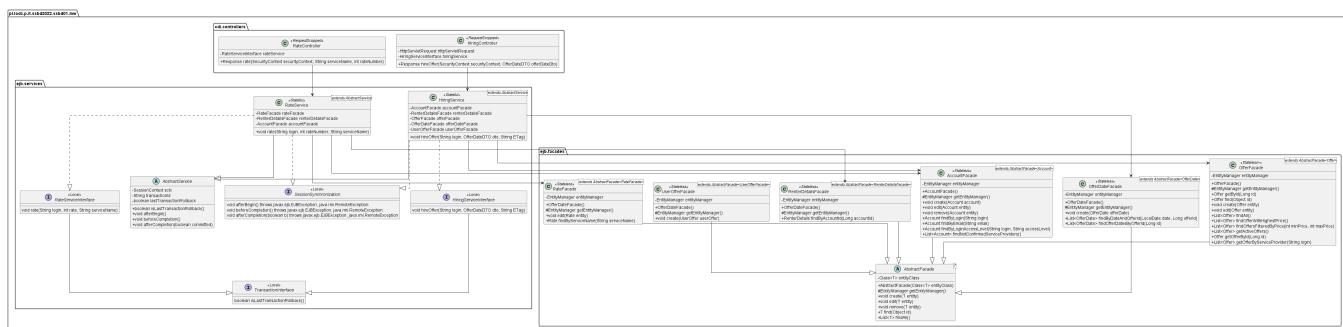
9 Diagram UML klas komponentow EJB/CDI 4 pkt



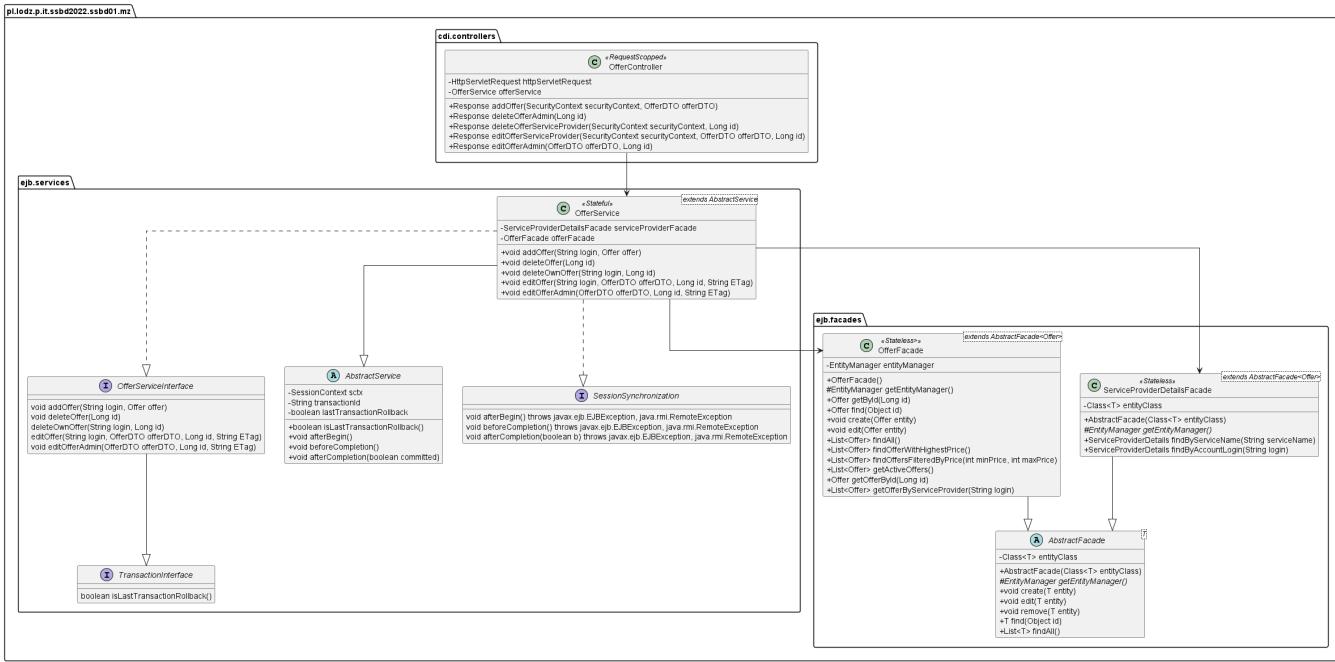
Obraz 1. Diagram klas komponentów EJB/CDI dla modułu MOK.



Obraz 2.moEJB.png Diagram klas komponentów EJB/CDI dla modułu MO.



Obraz 3. Diagram klas komponentów EJB/CDI dla modułu MW.



Obraz 4. Diagram klas komponentów EJB/CDI dla modułu MZ.

10 Model bezpieczeństwa komponentów EJB/CDI 5 pkt

Model bezpieczeństwa metod komponentów EJB

| Przypadek użycia | Kontroler | Serwis | Fasada |
|--|---|---|---|
| MOK.1 Zarejestruj | <p>@PermitAll public Response <i>CreateAccountController.registerRenterUser(@Valid PostRenterDTO postRenterDTO)</i></p> <p>@PermitAll public Response <i>CreateAccountController.registerServiceProviderUser(@Valid PostServiceProviderDTO postServiceProviderDTO)</i></p> <p>@PermitAll public Response <i>CreateAccountController.confirm(@PathParam("token") String token)</i></p> | <p>@PermitAll public void <i>CreateAccountService.createServiceProviderAccount(Account account, ServiceProviderDetails accessLevel)</i></p> <p>@PermitAll public void <i>CreateAccountService.createRenterAccountAndSendVerificationEmail (Account account, RenterDetails accessLevel)</i></p> <p>@PermitAll public void <i>CreateAccountService.confirmToken (String token)</i></p> <p>@PermitAll public void <i>MOKEmailSender.sendVerificationEmail(String email, String language)</i></p> | <p>@PermitAll public void <i>AccountFacade.create(T entity)</i></p> <p>@PermitAll public void <i>AccountFacade.edit(T entity)</i></p> <p>@PermitAll public void <i>VerificationTokenFacade.create(T entity)</i></p> <p>@PermitAll public void <i>VerificationTokenFacade.remove(T entity)</i></p> |
| MOK.2 Utwórz konto | <p>@RolesAllowed("Admin") public Response <i>CreateAccountController.createAdmin(@Valid PostAdminDTO postAdminDTO)</i></p> | <p>@RolesAllowed("Admin") public void <i>CreateAccountService.createAdminAccount(Account account, AdminDetails accessLevel)</i></p> | <p>@PermitAll public void <i>AccountFacade.create(T entity)</i></p> |
| MOK.3 Zablokuj konto | <p>@RolesAllowed("Admin") public Response <i>UpdateAccountController.changeActive(@QueryParam("login") String login, @QueryParam("active") boolean active, @QueryParam("versionHash") String versionHash)</i></p> | <p>@RolesAllowed("Admin") public void <i>UpdateAccountService.activateUserAccount(String login, String ETag)</i></p> <p>@PermitAll public void <i>MOKEmailSender.sendDeactivateEmail(String login, String language)</i></p> | <p>@PermitAll public Account <i>AccountFacade.findByLogin(String login)</i></p> <p>@PermitAll public void <i>AccountFacade.edit(T entity)</i></p> |
| MOK.4 Odblokuj konto | <p>@RolesAllowed("Admin") public Response <i>UpdateAccountController.changeActive(@QueryParam("login") String login, @QueryParam("active") boolean active, @QueryParam("versionHash") String versionHash)</i></p> | <p>@RolesAllowed("Admin") public void <i>UpdateAccountService.deactivateUserAccount(String login, String ETag)</i></p> <p>@PermitAll public void <i>MOKEmailSender.sendDeactivateEmail(String login, String language)</i></p> | <p>@PermitAll public Account <i>AccountFacade.findByLogin(String login)</i></p> <p>@PermitAll public void <i>AccountFacade.edit(T entity)</i></p> |
| MOK.5 Dolóż poziom dostępu do konta | <p>@RolesAllowed("Admin") public Response <i>UpdateAccountController.addRenterAccess(AddRenterDTO renterDTO, @Context SecurityContext securityContext)</i></p> <p>@RolesAllowed("Admin") public Response <i>UpdateAccountController.addServiceProviderAccess(AddServiceProviderDTO addServiceProviderDTO, @Context SecurityContext securityContext)</i></p> | <p>@RolesAllowed("Admin") public void <i>UpdateAccountService.addAccessLevel(String login, AccessLevel accessLevel)</i></p> | <p>@PermitAll public Account <i>AccountFacade.findByLogin(String login)</i></p> <p>@PermitAll public void <i>AccountFacade.edit(T entity)</i></p> |

| | | | |
|--------|--|---|---|
| MOK.6 | <p>Odlacz poziom dostepu od konta</p> <pre>@RolesAllowed("Admin") public Response UpdateAccountController.deleteAccessLevel(@Context SecurityContext securityContext, @QueryParam("accessLevel") String accessLevel)</pre> | <pre>@RolesAllowed("Admin") public void UpdateAccountService.deleteAccessLevel(String login, String accessString)</pre> | <pre>@RolesAllowed("Admin") public Account AccountFacade.findByLoginAccessLevel(String login, String accessLevel)</pre> <pre>@RolesAllowed("Admin") public void AccessLevelFacade.remove(T entity)</pre> |
| MOK.7 | <p>Zmień własne hasło</p> <pre>@RolesAllowed("ALL") public Response UpdateAccountController.changeUserOwnPassword(@Context SecurityContext securityContext, @QueryParam("oldPassword") String oldPassword, @QueryParam("newPassword") String newPassword, @QueryParam("versionHash") String versionHash)</pre> | <pre>@RolesAllowed("ALL") public void UpdateAccountService.changeOwnPassword(String login, String oldPassword, String newPassword, String ETag)</pre> | <pre>@PermitAll public Account AccountFacade.findByLogin(String login)</pre> <pre>@PermitAll public void AccountFacade.edit(T entity)</pre> |
| MOK.8 | <p>Zmień hasło innego użytkownika</p> <pre>@RolesAllowed("Admin") public Response UpdateAccountController.changeUserPassword(@QueryParam("login") String login, @QueryParam("newPassword") String newPassword)</pre> | <pre>@RolesAllowed("Admin") public void UpdateAccountService.changePassword(String login, String newPassword, String ETag)</pre> | <pre>@PermitAll public Account AccountFacade.findByLogin(String login)</pre> <pre>@PermitAll public void AccountFacade.edit(T entity)</pre> |
| MOK.9 | <p>Edytuj dane własnego konta</p> <pre>@RolesAllowed("ServiceProvider") public Response UpdateAccountController.editServiceProvider(@Context SecurityContext securityContext, @Valid PutProviderDTO putProviderDTO)</pre> <pre>@RolesAllowed("Admin") public Response UpdateAccountController.updateAdminAccount(@Context SecurityContext securityContext, PutAccountDTO putAccountDTO)</pre> <pre>@RolesAllowed("Renter") public Response UpdateAccountController.updateRenterAccount(@Context SecurityContext securityContext, PutRenterDTO putRenterDTO)</pre> | <pre>@RolesAllowed("AdminService") public void UpdateAccountService.editServiceProviderDetails(String login, PutProviderDTO putProviderDTO, String ETag)</pre> <pre>@RolesAllowed("Admin") public void UpdateAccountService.updateAdmin(String login, PutAccountDTO dto, String ETag)</pre> <pre>@RolesAllowed("AdminRenter") public void UpdateAccountService.updateRenter(String login, PutRenterDTO dto, String ETag)</pre> | <pre>@PermitAll public Account AccountFacade.findByLogin(String login)</pre> <pre>@PermitAll public void AccountFacade.edit(T entity)</pre> <pre>@PermitAll public void AccessLevelFacade.edit(T entity)</pre> |
| MOK.10 | <p>Edytuj dane konta innego użytkownika</p> <pre>@RolesAllowed("Admin") public Response UpdateAccountController.editServiceProviderAdmin(@PathParam("login") String login, @Valid PutProviderDTO putProviderDTO)</pre> <pre>@RolesAllowed("Admin") public Response UpdateAccountController.updateRenterAccountAdmin(@PathParam("login") String login, PutRenterDTO putRenterDTO)</pre> | <pre>@RolesAllowed("AdminService") public void UpdateAccountService.editServiceProviderDetails(String login, PutProviderDTO putProviderDTO, String ETag)</pre> <pre>@RolesAllowed("AdminRenter") public void UpdateAccountService.updateRenter(String login, PutRenterDTO dto, String ETag)</pre> | <pre>@PermitAll public Account AccountFacade.findByLogin(String login)</pre> <pre>@PermitAll public void AccountFacade.edit(Account entity)</pre> <pre>@RolesAllowed("ALL") public void AccessLevelFacade.edit(AccessLevel entity)</pre> |
| MOK.11 | Wyloguj | - | - |

| | | | |
|--------|--|---|---|
| MOK.12 | <p>@RolesAllowed("ServiceProvider")</p> <pre>public Response UpdateAccountController.editServiceProvider(@Context SecurityContext securityContext, @Valid PutProviderDTO putProviderDTO)</pre> | <p>@RolesAllowed("AdminService")</p> <pre>public void UpdateAccountService.editSe rviceProviderDetails(String login, PutProviderDTO putProviderDTO, String ETag)</pre> | <p>@PermitAll</p> <pre>public Account AccountFacade .findByLogin(String login)</pre> <p>@PermitAll</p> <pre>public void AccountFacade. edit(Account entity)</pre> <p>@RolesAllowed("ALL")</p> <pre>public void AccessLevelFacade .edit(AccessLevel entity)</pre> |
| MOK.13 | <p>@RolesAllowed("Admin")</p> <pre>public Response DeleteAccountController.deleteServiceProvider(@QueryParam("login") String login, @QueryParam("language") String language)</pre> <p>@RolesAllowed("Admin")</p> <pre>public Response ReadAccountController.readProvidersToConfirm()</pre> <p>@RolesAllowed("Admin")</p> <pre>public Response UpdateAccountController.confirmServiceProvider(@QueryParam ("login") String login,@QueryParam("language") String language)</pre> | <p>@PermitAll</p> <pre>public Account ReadAccountService.read AccountByLogin(String login)</pre> <p>@RolesAllowed("Admin")</p> <pre>public void DeleteAccountService.deleteS erviceProvider(String login)</pre> <p>@RolesAllowed("Admin")</p> <pre>public List<Account> ReadAccountServ ice.readProvidersToConfirm()</pre> <p>@RolesAllowed("Admin")</p> <pre>public void UpdateAccountService.confir mServiceProvider(String login)</pre> <p>@PermitAll</p> <pre>public void MOKEmailSender.sendRemo veEmail(String email, String language)</pre> <p>@PermitAll</p> <pre>public void MOKEmailSender.sendConfir mServiceProviderEmail(String login, String language)</pre> | <p>@PermitAll</p> <pre>public Account AccountFacade.findByLogin (String login)</pre> <p>@PermitAll</p> <pre>public void AccountFacade.remove(T entity)</pre> <p>@RolesAllowed("Admin")</p> <pre>public List<Account> Acco untFacade.findNotConfirmedS erviceProviders()</pre> <p>@PermitAll</p> <pre>public void AccountFacade.Ac countFacade.edit(T entity)</pre> |
| MOK.14 | <p>@PermitAll</p> <pre>public Response AuthController.authenticate(@Valid AuthDTO authDTO)</pre> <p>@RolesAllowed("ALL")</p> <pre>public Response AuthControllerlr.findSelf(@Context SecurityContext securityContext)</pre> | <p>@PermitAll</p> <pre>public TokenDTO AuthService.generateT okens(String login, String password)</pre> <p>@RolesAllowed("ALL")</p> <pre>public Account ReadAccountService.read AccountByLogin(String login)</pre> | <p>@PermitAll</p> <pre>public Account AccountFacade .findByLogin(String login)</pre> <p>@PermitAll</p> <pre>public void VerificationTokenFacade. create(VerificationToken entity)</pre> |

| | | | |
|--|--|--|--|
| MOK.15 Reset hasła | <p>@PermitAll public Response <i>UpdateAccountController.sendResetPasswordEmail(@PathParam("email") String email, @QueryParam("language") String language)</i></p> <p>@PermitAll public Response <i>UpdateAccountController.confirmResetPasswordEmail(@PathParam("token") String token)</i></p> <p>@PermitAll public Response <i>UpdateAccountController.updatePassword(@QueryParam("token") String token, @QueryParam("newPassword") String newPassword)</i></p> | <p>@PermitAll public void <i>UpdateAccountService.sendPpasswordResetMail(String email)</i></p> <p>@PermitAll public void <i>UpdateAccountService.findByVerificationToken(String token)</i></p> <p>@PermitAll public void <i>UpdateAccountService.resetPpassword(String token, String newPassword)</i></p> <p>@PermitAll public void <i>MOKEmailSender.sendPasswordResetMail(String email, String language)</i></p> | <p>@PermitAll public Account <i>AccountFacade.findByEmail(String login)</i></p> <p>@PermitAll public void <i>VerificationTokenFacade.create(VerificationToken entity)</i></p> <p>@PermitAll public VerificationToken <i>VerificationTokenFacade.findByToken(String token)</i></p> <p>@PermitAll public void <i>VerificationTokenFacade.remove(T entity)</i></p> <p>@PermitAll public void <i>AccountFacade.edit(Account entity)</i></p> |
| Moduły dodatkowe | Kontroler | Serwis | Fasada |
| MZ.1 Dodawanie ofert | <p>@RolesAllowed("ServiceProvider") public Response <i>OfferController.addOffer(@Context SecurityContext securityContext, OfferDTO offerDTO)</i></p> | <p>@RolesAllowed("ServiceProvider") public void <i>OfferService.OfferService.addOffer(String login, Offer offer)</i></p> | <p>@RolesAllowed("ServiceProvider") public void <i>OfferFacade.create(Offer entity)</i></p> |
| MZ.2 Usuwanie ofert | <p>@RolesAllowed("Admin") public Response <i>OfferController.deleteOfferAdmin(@PathParam("id") Long id)</i></p> <p>@Path("/deleteByService/{id}") public Response <i>OfferController.deleteOfferServiceProvider(@Context SecurityContext securityContext, @PathParam("id") Long id)</i></p> | <p>@RolesAllowed("Admin") public void <i>OfferService.deleteOffer(Long id)</i></p> <p>@RolesAllowed("ServiceProvider") public void <i>OfferService.deleteOwnOffer(String login, Long id)</i></p> | <p>@PermitAll public Offer <i>OfferFacade.findById(Object id)</i></p> <p>@RolesAllowed("AdminService") public void <i>OfferFacade.edit(Offer entity)</i></p> <p>@PermitAll public Offer <i>OfferFacade.getB yId(Long id)</i></p> |
| MZ.3 Edycja ofert | <p>@RolesAllowed("ServiceProvider") public Response <i>OfferController.editOfferServiceProvider(@Context SecurityContext securityContext, @Valid OfferDTO offerDTO, @PathParam("id") Long id)</i></p> <p>@RolesAllowed("Admin") public Response <i>OfferController.editOfferAdmin(@Valid OfferDTO offerDTO, @PathParam("id") Long id)</i></p> | <p>@RolesAllowed("ServiceProvider") public void <i>OfferService.editOffer(String login, OfferDTO offerDTO, Long id, String ETag)</i></p> <p>@RolesAllowed("Admin") public void <i>OfferService.editOfferAdmin(OfferDTO offerDTO, Long id, String ETag)</i></p> | <p>@PermitAll public Offer <i>OfferFacade.getB yId(Long id)</i></p> <p>@RolesAllowed("AdminService") public void <i>OfferFacade.edit(Offer entity)</i></p> |
| MO.1 Przeglądanie profili firm ogłasza jacych | <p>@PermitAll public Response <i>BrowseOfferController.getAllServiceProviders()</i></p> | <p>@PermitAll public List<Account> <i>BrowseOfferService.getAllServiceProviders()</i></p> | <p>@PermitAll public List<Account> <i>getAccountList()</i></p> |

| | | | |
|---|--|--|---|
| MO.2 Przeglądanie aktualnych ofert | <p>@PermitAll public Response <i>BrowseOfferController.getAllOffers()</i></p> <p>@PermitAll public Response <i>BrowseOfferController.getAllActiveOffers()</i></p> <p>@PermitAll public Response <i>BrowseOfferController.getOfferById(@QueryParam("id") Long id)</i></p> <p>@PermitAll public Response <i>BrowseOfferController.getOfferDatesByOfferId(@QueryParam("id") Long offerId)</i></p> | <p>@PermitAll public List<Offer> <i>BrowseOfferService.getAllOffers()</i></p> <p>@PermitAll public List<Offer> <i>BrowseOfferService.getActiveOffers()</i></p> <p>@PermitAll public Offer <i>BrowseOfferService.getOfferById(Long id)</i></p> <p>@PermitAll public List<OfferDate> <i>BrowseOfferService.getOfferDatesByOfferId(Long id)</i></p> | <p>@PermitAll public List<Offer> <i>OfferFacade.findAll()</i></p> <p>@PermitAll public List<Offer> <i>OfferFacade.getActiveOffers()</i></p> <p>@PermitAll public Offer <i>OfferFacade.getOfferById(Long id)</i></p> <p>@PermitAll public List<OfferDate> <i>OfferFacade.findOfferDatesByOfferId(Long id)</i></p> |
| MO.3 Filtrowanie przeglądanych ofert | <p>@PermitAll public Response <i>BrowseOfferController.getOffersFilteredByPrice(@QueryParam("minPrice") int minPrice, @QueryParam("maxPrice") int maxPrice)</i></p> | <p>@PermitAll public List<Offer> <i>BrowseOfferService.getOffersFilteredByPrice(int minPrice, int maxPrice)</i></p> | <p>@PermitAll public List<Offer> <i>OfferFacade.findOfferWithHighestPrice()</i></p> <p>@PermitAll public List<Offer> <i>OfferFacade.findOffersFilteredByPrice(int minPrice, int maxPrice)</i></p> |
| MW.1 Możliwość wynajęcia techniki | <p>@RolesAllowed("Renter") public Response <i>HiringController.hireOffer(@Context SecurityContext securityContext, OfferDateDTO offerDateDto)</i></p> | <p>@RolesAllowed("Renter") public void <i>HiringService.hireOffer(String login, OfferDateDTO dto, String ETag)</i></p> | <p>@RolesAllowed("Renter") public List<OfferDate> <i>OfferDateFacade.findByDateAndOfferId(LocalDate date, Long offerId)</i></p> <p>@PermitAll public Offer <i>OfferFacade.find(Object id)</i></p> <p>@PermitAll public Account <i>AccountFacade.findByLogin(String login)</i></p> <p>@RolesAllowed("Renter") public RenterDetails <i>RenterDetailsFacade.findById(Long accountId)</i></p> <p>@RolesAllowed("Renter") public void <i>OfferDateFacade.create(OfferDate offerDate)</i></p> <p>@RolesAllowed("Renter") public void <i>UserOfferFacade.create(UserOffer userOffer)</i></p> |

| | | | |
|---------------------------|--|--|--|
| MW.2 Ocenianie technik | <pre>@RolesAllowed("Renter") public Response RateController.rate(@Context SecurityContext securityContext, @PathParam("serviceName") String serviceName, @PathParam("rateNumber") int rateNumber)</pre> | <pre>@RolesAllowed("Renter") public void RateService.rate(String login, int rateNumber, String serviceName)</pre> | <pre>@RolesAllowed("Renter") public Rate RateFacade.findByNameServiceName(String serviceName)</pre> |
| | | <pre>@PermitAll public Account AccountFacade.findByLogin(String login)</pre> | <pre>@RolesAllowed("Renter") public RenterDetails RenterDetailsFacade.findByAccountId(Long accountId)</pre> |
| | | | <pre>@RolesAllowed("Renter") public void RateFacade.edit(Rate entity)</pre> |

Tabela przedstawiająca mapowanie grup oraz tożsamości użytkowników na odpowiadające im role aplikacji

Po lewej stronie role aplikacji na które mapowane są grupy użytkowników wypisane po prawej stronie

| Role aplikacji | Grupy oraz tożsamości użytkowników |
|-----------------|------------------------------------|
| Admin | Admin |
| Renter | Renter |
| ServiceProvider | ServiceProvider |
| RenterService | Renter ServiceProvider |
| AdminService | Admin ServiceProvider |
| AdminRenter | Admin Renter |
| ALL | Admin ServiceProvider Renter |

Implementacja LoggerInterceptor oraz jego użycie na przykładzie klasy CreateAccountService

| |
|---------------------------------|
| Implementacja LoggerInterceptor |
|---------------------------------|

```

public class LoggerInterceptor {

    @Resource
    private SessionContext sctx;
    private static final Logger logger = Logger.getLogger(LoggerInterceptor.class.getName());

    @AroundInvoke
    public Object traceInvoke(InvocationContext ictx) throws Exception {
        StringBuilder message = new StringBuilder("Method invoked: ");
        Object result;
        try {
            try {
                message.append(ictx.getMethod().toString());

                message.append(", Caller: ").append(sctx.getCallerPrincipal().getName());
                message.append(", Method parameters: ");
                if (null != ictx.getParameters()) {
                    for (Object param : ictx.getParameters()) {
                        message.append(String.valueOf(param));
                    }
                }
            } catch (Exception e) {
                logger.log(Level.SEVERE, "Unexpected logging exception: ", e.getMessage());
                throw e;
            }
            result = ictx.proceed();
        } catch (Exception e) {
            message.append(" Method invocation ended with exception: ").append(e.getMessage());
            logger.log(Level.SEVERE, message.toString(), e);
            throw e;
        }

        message.append("Returned value: ").append(String.valueOf(result)).append(" ");
        logger.info(message.toString());
    }
}

```

Przykładowe zastosowanie LoggerInterceptor

```

@Stateful
@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)

@Interceptors({
    GenericServiceInterceptor.class,
    LoggerInterceptor.class})
public class CreateAccountService extends AbstractService implements CreateAccountServiceInterface, SessionSynchronization {

```

11 Identyfikacja transakcji aplikacyjnych 6 pkt

Transakcje aplikacyjne z udziałem metod komponentów EJB/CDI.

MOK. 1.1 (Rejestracja użytkownika wynajmującego usługi):

komponent CDI @RequestScoped CreateAccountController, metoda registerRenterUser(PostRenterDTO postRenterDTO, String language) →

komponent EJB @Stateful CreateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) createRenterAccountAndSendVerificationEmail(JPA: Account account, JPA: AccessLevel accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: Account account),

komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: VerificationToken token)

MOK. 1.2 (Rejestracja użytkownika zapewniającego usługi):

komponent CDI @RequestScoped CreateAccountController, metoda registerServiceProviderUser(PostServiceProviderDTO postServiceProviderDTO) →

komponent EJB @Stateful CreateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) createServiceProviderAccount(JPA: Account account, JPA: AccessLevel accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: Account account)

MOK. 2

komponent CDI @RequestScoped CreateAccountController, metoda createAdmin(PostAdminDTO postAdminDTO) →

komponent EJB @Stateful CreateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) createAdminAccount(JPA: Account account, JPA: AccessLevel accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: Account account)

MOK. 3/4

komponent CDI @RequestScoped UpdateAccountController, metoda changeActive(String login, boolean active, String versionHash, String language) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) activate/deactivateUserAccount(JPA: Account account, JPA: AccessLevel accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

MOK. 5

komponent CDI @RequestScoped UpdateAccountController, metoda addRenterAccess/ServiceProviderAccess(AddRenterDTO/AddServiceProviderDTO dto, SecurityContext securityContext) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) addAccessLevel(String login, JPA: AccessLevel accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

MOK. 6

komponent CDI @RequestScoped UpdateAccountController, metoda deleteAccessLevel(SecurityContext securityContext, String accessLevel) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) deleteAccessLevel(String login, String accessLevel) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLoginAccessLevel(String login, String accessLevel),

komponent EJB @Stateless AccessLevelFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) remove(JPA: AccessLevel accessLevel)

MOK. 7

komponent CDI @RequestScoped UpdateAccountController, metoda changeUserOwnPassword(SecurityContext securityContext, String oldPassword, String newPassword, String versionHash) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) changeOwnPassword(SecurityContext securityContext, String oldPassword, String newPassword, String versionHash) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

MOK. 8

komponent CDI @RequestScoped UpdateAccountController, metoda changeUserPassword(String login, String newPassword, String versionHash) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) changePassword(String login, String newPassword, String versionHash) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

MOK. 9/12 (edytowanie własnego konta jako użytkownik zapewniający usługi)

komponent CDI @RequestScoped UpdateAccountController, metoda editAdmin/ServiceProvider/Renter(SecurityContext securityContext, PutDTO dto) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) updateAdmin/ServiceProvider/Renter(String login, PutDTO dto) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

komponent EJB @Stateless AccessLevelFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: AccessLevel accessLevel)

MOK. 10

komponent CDI @RequestScoped UpdateAccountController, metoda editAdmin/ServiceProvider/Renter(String login, PutDTO dto) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) updateAdmin/ServiceProvider/Renter(String login, PutDTO dto) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

komponent EJB @Stateless AccessLevelFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: AccessLevel accessLevel)

MOK. 13

Przypadek 1. Firma została zatwierdzona przez administratora.

komponent CDI @RequestScoped UpdateAccountController, metoda confirmServiceProvider(String login, String language) →

komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) confirmServiceProvider(String login, String language) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account)

Przypadek 2. Firma nie została zatwierdzona przez administratora.

komponent CDI @RequestScoped DeleteAccountController, metoda deleteServiceProvider(String login, String language) →
komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) deleteServiceProvider(String login) →
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) remove(JPA: Account account)

MOK. 15

Krok 1. Wysłanie maila z linkiem do zmiany hasła na podany mail.

komponent CDI @RequestScoped UpdateAccountController, metoda sendResetPasswordEmail(String email, String language) →
komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) sendPasswordResetEmail(String email) →
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByEmail(String email),
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: VerificationToken token)

Krok 2. Sprawdzenie czy token do zmiany hasła wygasł, przeniesienie na stronę z formularzem do zmiany hasła lub przeniesienie na stronę informującą, że token wygasł.

komponent CDI @RequestScoped UpdateAccountController, metoda confirmPasswordEmail(String token) →
komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) sendPasswordResetEmail(String email) →
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByToken(String token)

Krok 3. Reset hasła z wykorzystaniem wyżej wymienionego tokenu.

komponent CDI @RequestScoped UpdateAccountController, metoda resetPassword(String token, String newPassword) →
komponent EJB @Stateful UpdateAccountService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) resetPassword(String token, String newPassword) →
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByToken(String token),
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Account account),
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) remove(JPA: VerificationToken verificationToken)

MZ.1

komponent CDI @RequestScoped OfferController, metoda addOffer(SecurityContext securityContext, OfferDTO offerDTO) →
komponent EJB @Stateful OfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) addOffer(String login, JPA: Offer offer) →
komponent EJB @Stateless ServiceProviderDetailsFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByAccountLogin(String login),
komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: Offer offer)

MZ.2

komponent CDI @RequestScoped OfferController, metoda deleteOfferAdmin(Long id)/deleteOfferServiceProvider(SecurityContext securityContext, Long id) →
komponent EJB @Stateful OfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) deleteOffer(Long id)/deleteOwnOffer(String login, Long id) →
komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getById(Long id),

komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Offer offer)

MZ.3

komponent CDI @RequestScoped OfferController, metoda editOfferAdmin(OfferDTO offerDTO)/editOfferServiceProvider(SecurityContext, OfferDTO offerDTO) →

komponent EJB @Stateful OfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) editOfferAdmin(OfferDTO offerDTO, String ETag)/editOffer(String login, OfferDTO offerDTO, String ETag) →

komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getById(Long id),

komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Offer offer)

MO.1

komponent CDI @RequestScoped BrowseOfferController, metoda getAllServiceProviders() →

komponent EJB @Stateful BrowseOfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) getAllServiceProviders() →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getAllServiceProviders()

MO.2

komponent CDI @RequestScoped BrowseOfferController, metoda getAllActiveOffers() →

komponent EJB @Stateful BrowseOfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) getAllActiveOffers() →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getAllActiveOffers()

MO.3

komponent CDI @RequestScoped BrowseOfferController, metoda getOffersFilteredByPrice(int minPrice, int maxPrice) →

komponent EJB @Stateful BrowseOfferService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) getOffersFilteredByPrice(int minPrice, int maxPrice) →

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getOffersFilteredByPrice(int minPrice, int maxPrice)

MW.1

komponent CDI @RequestScoped HiringController, metoda hireOffer(SecurityContext securityContext, OfferDateDTO offerDateDTO) →

komponent EJB @Stateful HiringService, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) hireOffer(String login, OfferDateDTO offerDateDTO, String Etag) →

komponent EJB @Stateless OfferDateFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByDateAndOfferId(LocalDate date, Long id),

komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) find(Long id),

komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),

komponent EJB @Stateless RenterDetailsFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByAccountId(Long id),

komponent EJB @Stateless OfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) find(Long id),

komponent EJB @Stateless OfferDateFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: OfferDate offerDate),

komponent EJB @Stateless UserOfferFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) create(JPA: UserOffer useroffer)

MW.2

komponent CDI @RequestScoped RateController, metoda rate(SecurityContext securityContext, String serviceName, int rateNumber) →
komponent EJB @Stateful RateService metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) rate(String login, int rateNumber, String serviceName) →
komponent EJB @Stateless RateFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByName(String login),
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByLogin(String login),
komponent EJB @Stateless RenterDetailsFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByAccountId(Long id),
komponent EJB @Stateless RateFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) edit(JPA: Rate rate)

SYSTEM 1

komponent EJB @Stateful MOKEmailSender, metoda @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW) sendEmail(String emial, String language) →
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) findByAccountEmail(String email)

SYSTEM 2

komponent EJB @Singleton Scheduler, metoda removeInactiveTokensAndUsers() →
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) getExpiredTokens(),
komponent EJB @Stateless VerificationTokenFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) remove(JPA: VerificationToken verificationToken),
komponent EJB @Stateless AccountFacade, metoda @TransactionAttribute(TransactionAttributeType.MANDATORY) remove(JPA: Account account)

* → - oznacza wywołanie metody wstrzykniętego komponentu EJB/CDI.

Mechanizm ponawiania odwołanych transakcji aplikacyjnych.

W mechanizmie ponawiania odwołanych transakcji aplikacyjnych została zastosowana pętla "do while", w której warunkiem sprawdzanym jest zmienna określająca czy transakcja została odwołana oraz licznik ponowień transakcji, który wynosi 3. Za pętlą "do while" został zastosowany warunek sprawdzający, który zadziała tylko wtedy, gdy transakcja ponowi się 3 razy, a jej statusem będzie nadal stan odwołany. Na końcu metody znajduje się odpowiedź zwrotna, która ma zostać wysłana do klienta wykonującego żądanie.

```

public Response someResponse() {
    int retryTXCounter = 3;
    boolean rollbackTX;
    do{
        try {
            rollbackTX =
updateAccountService.
isLastTransactionRollback();
            if (rollbackTX) LOGGER
.info("**** Transaction rollback");

        } catch (
ApplicationBaseException exc) {
            throw exc;
        } catch (
EJBAccessException |
AccessLocalException exc) {
            throw
ApplicationBaseException.
getAccessDeniedException(exc);
        } catch (Exception exc) {
            throw
ApplicationBaseException.
getGeneralErrorException(exc);
        }
    } while (rollbackTX && --
retryTXCounter > 0);

    if (rollbackTX) {
        throw
ApplicationBaseException.
getGeneralErrorException();
    }

    return Response
        .ok()
        .build();
}

```

Przypadki wykorzystywania mechanizmu blokad optymistycznych zastosowanych dla encji JPA w scenariuszach wymagających realizacji więcej niż jednej transakcji aplikacyjnej.

| Nazwa przypadku | Metoda wczytująca dane | Metoda modyfikująca dane | Mechanizm zapewniający wiarygodność wersji encji JPA |
|--|---|---|--|
| 1. Zablokowanie /odblokowanie konta użytkownika. | public Account findByLogin (String login) | public void activateUserAccount(String login, String versionHash) / public void deactivateUserAccount(String login, String versionHash) | if (versionHash.equals(hashGenerator.generateHash(String.valueOf (account.getVersion())))) { account.setActive(false); accountFacade.edit(account); } else { throw new OptimisticLockException(); } |
| 2. Zmiana hasła innego użytkownika. | public Account findByLogin (String login) | public void changePassword(String login, String newPassword, String versionHash) | if (versionHash.equals(hashGenerator.generateHash(String.valueOf (account.getVersion())))) { if (account.getPassword().equals(hashGenerator.generateHash (newPassword))) { throw new PasswordAlreadyUsedException(); } account.setPassword(hashGenerator.generateHash (newPassword)); accountFacade.edit(account); } else { throw new OptimisticLockException(); } |

| | | | |
|--|---|---|--|
| 3. Zmiana własnego hasła. | public Account findByLogin (String login) | public void changeOwnPassword(String login, String oldPassword, String newPassword, String versionHash) | <pre>if (versionHash.equals(hashGenerator.generateHash(String.valueOf(account.getVersion())))) { if (account.getPassword().equals(hashGenerator.generateHash(oldPassword))) { if (account.getPassword().equals(hashGenerator.generateHash(newPassword))) { throw new PasswordAlreadyUsedException(); } account.setPassword(hashGenerator.generateHash(newPassword)); accountFacade.edit(account); } else { throw new AuthorizationException(); } } else { throw new OptimisticLockException(); }</pre> |
| 4. Edycja konta firmy. | public Account findByLogin (String login) | public void editServiceProviderDetails(String login, PutProviderDTO putProviderDTO) | <pre>if (putProviderDTO.getAccountVersionHash().equals(hashGenerator.generateHash(String.valueOf(dbAccount.getVersion())))) && putProviderDTO.getAccessLevelVersionHash().equals(hashGenerator.generateHash(String.valueOf(accessLevel.getVersion())))) { dbAccount.setName(putProviderDTO.getName()); dbAccount.setSurname(putProviderDTO.getSurname()); dbAccount.setPhoneNumber(putProviderDTO.getPhoneNumber()); accessLevel.setAddress(putProviderDTO.getAddress()); accessLevel.setDescription(putProviderDTO.getDescription()); accessLevel.setLogoUrl(putProviderDTO.getLogoUrl()); accountFacade.edit(dbAccount); accessLevelFacade.edit(accessLevel); } else { throw new OptimisticLockException(); }</pre> |
| 5. Edycja konta firmy przez administratora. | public Account findByLogin (String login) | public void editServiceProviderDetails(String login, PutProviderDTO putProviderDTO) | <pre>if (putProviderDTO.getAccountVersionHash().equals(hashGenerator.generateHash(String.valueOf(dbAccount.getVersion())))) && putProviderDTO.getAccessLevelVersionHash().equals(hashGenerator.generateHash(String.valueOf(accessLevel.getVersion())))) { dbAccount.setName(putProviderDTO.getName()); dbAccount.setSurname(putProviderDTO.getSurname()); dbAccount.setPhoneNumber(putProviderDTO.getPhoneNumber()); accessLevel.setAddress(putProviderDTO.getAddress()); accessLevel.setDescription(putProviderDTO.getDescription()); accessLevel.setLogoUrl(putProviderDTO.getLogoUrl()); accountFacade.edit(dbAccount); accessLevelFacade.edit(accessLevel); } else { throw new OptimisticLockException(); }</pre> |
| 6. Edycja konta administratora. | public Account findByLogin (String login) | public void updateAdmin(String login, PutAccountDTO dto) | <pre>if (dto.getAccountVersionHash().equals(hashGenerator.generateHash(String.valueOf(dbAccount.getVersion())))){ dbAccount.setName(dto.getName()); dbAccount.setSurname(dto.getSurname()); dbAccount.setPhoneNumber(dto.getPhoneNumber()); accountFacade.edit(dbAccount); } else { throw new OptimisticLockException(); }</pre> |

| | | | |
|---|---|--|---|
| 7. Edycja konta klienta. | public Account findByLogin (String login) | public void updateRenter(String login, PutRenterDTO dto) | <pre>if (dto.getAccountVersionHash().equals(hashGenerator.generateHash (String.valueOf(dbAccount.getVersion())))) && dto.getAccessLevelVersionHash().equals (hashGenerator.generateHash(String.valueOf(accessLevel. getVersion())))) { dbAccount.setName(dto.getName()); dbAccount.setSurname(dto.getSurname()); dbAccount.setPhoneNumber(dto.getPhoneNumber()); accessLevel.setUserName(dto.getUserName()); accountFacade.edit(dbAccount); accessLevelFacade.edit(accessLevel); } else { throw new OptimisticLockException(); }</pre> |
| 8. Edycja konta klienta przez administratora. | public Account findByLogin (String login) | public void updateRenter(String login, PutRenterDTO dto) | <pre>if (dto.getAccountVersionHash().equals(hashGenerator.generateHash (String.valueOf(dbAccount.getVersion()))) && dto.getAccessLevelVersionHash().equals (hashGenerator.generateHash(String.valueOf(accessLevel. getVersion())))) { dbAccount.setName(dto.getName()); dbAccount.setSurname(dto.getSurname()); dbAccount.setPhoneNumber(dto.getPhoneNumber()); accessLevel.setUserName(dto.getUserName()); accountFacade.edit(dbAccount); accessLevelFacade.edit(accessLevel); } else { throw new OptimisticLockException(); }</pre> |

Przykładowy scenariusz dla przypadku 4.

Zaprezentowanie metody kontrolera, która jest odpowiedzialna za edycję konta firmy.

UpdateAccountController.java

```

@PUT
@RolesAllowed("ServiceProvider")
@Path("/editServiceProvider")
@Consumes(MediaType.APPLICATION_JSON)
public Response editServiceProvider(
    @Context SecurityContext securityContext,
    @Valid PutProviderDTO putProviderDTO) {
    int retryTXCounter = 3;
    boolean rollbackTX;

    do {
        try {
            updateAccountService.editServiceProviderDetails(
                securityContext.getUserPrincipal().getName(),
                putProviderDTO
            );
        }

        rollbackTX = updateAccountService.isLastTransactionRollback();
        if (rollbackTX) LOGGER.info("*** Transaction rollback");

        } catch (ApplicationBaseException exc) {
            throw exc;
        } catch (EJBAccessException exc) {
            throw ApplicationBaseException.getAccessDeniedException(exc);
        } catch (Exception exc) {
            throw ApplicationBaseException.getGeneralErrorException(exc);
        }
    } while (rollbackTX && --retryTXCounter > 0);

    if (rollbackTX) {
        throw ApplicationBaseException.getGeneralErrorException();
    }

    return Response
        .ok()
        .build();
}

```

Zaprezentowanie metody serwisu odpowiedzialną za odczyt encji, sprawdzenie wiarygodności jej wersji, edycję encji oraz za jej utrwalenie. Jeśli wersje się nie zgadzają, zostaje wyrzucony wyjątek OptimisticLockException.

UpdateAccountService.java

```

@RolesAllowed("AdminService")

@Override
public void editServiceProviderDetails(String login, PutProviderDTO putProviderDTO)
{
    Account dbAccount = accountFacade.findByLogin(login);

    for (AccessLevel ac : dbAccount.getAccessLevelCollection()){
        if (ac instanceof ServiceProviderDetails){
            ServiceProviderDetails accessLevel = (ServiceProviderDetails) ac;

            if (putProviderDTO.getAccountVersionHash().equals(hashGenerator.generateHash(String.valueOf(dbAccount.getVersion()))))
                && putProviderDTO.getAccessLevelVersionHash().equals(hashGenerator.generateHash(String.valueOf(accessLevel.getVersion())))
        } {
            dbAccount.setName(putProviderDTO.getName());
            dbAccount.setSurname(putProviderDTO.getSurName());
            dbAccount.setPhoneNumber(putProviderDTO.getPhoneNumber());

            accessLevel.setAddress(putProviderDTO.getAddress());
            accessLevel.setDescription(putProviderDTO.getDescription());
            accessLevel.setLogoUrl(putProviderDTO.getLogoUrl());
            accountFacade.edit(dbAccount);
            accessLevelFacade.edit(accessLevel);

        } else {
            throw new OptimisticLockException();
        }
    }
}

```

Zaprezentowanie metody fasady odpowiedzialnej za odczyt konta użytkownika po podanym loginie.

AccountFacade.java

```

@PermitAll
public Account findByLogin(String login) {
    TypedQuery<Account> typedQuery = entity
        Manager.createNamedQuery("Account.
        findByLogin", Account.
        class);
    typedQuery.setParameter("login", login);
    return typedQuery.getSingleResult();
}

```

Zaprezentowanie metody abstrakcyjnej fasady służącej do zaktualizowania encji w bazie danych.

AbstractFacade.java

```

public void edit(T entity) {
    getEntityManager().merge(entity);
    getEntityManager().flush();
}

```

Rozwiązanie zapewniające rejestrowanie w dzienniku zdarzeń wykonywanych transakcji aplikacyjnych.

W naszym systemie wielodostępowym każdy "Service" dziedziczy po klasie "AbstractService" oraz implementuje interfejs SessionSynchronization. Metody interfejsu SessionSynchronization są nadpisywane w klasie AbstractService. Zostało to tak zaimplementowane, ponieważ kontener EJB nie widzi implementacji interfejsów w klasach nadzędnych komponentów EJB. W trakcie transakcji, za pomocą loggera, nasza aplikacja rejestruje jej 3 stany:

- stan po rozpoczęciu - metoda afterBegin(),
- stan przed zakończeniem - metoda beforeCompletion(),
- stan po zakończeniu - metoda afterCompletion(boolean committed).

Każda transakcja posiada swój identyfikator generowany w następujący sposób:

```

transactionId =
    Long.toString(
        System.
        currentTimeMillis()
    )
    +
    ThreadLocalRandom.
    current().
    nextLong(Long.
    MAX_VALUE);

```

Tożsamość użytkownika, na rzecz którego wykonywana jest metoda jest pobierana z kontekstu sesji komponentu:

```

sctx.
getCallerPrincipal
().getName()

```

Nadpisana metoda afterCompletion(boolean committed) rejestruje nam wynik zakończenia transakcji: "APPROVAL" - zatwierdzenie, "CANCELLATION" - wycofanie.

Klasa AbstractService z nadpisanymi metodami interfejsu SessionSynchronized:

```

public abstract class AbstractService {
    protected static final Logger LOGGER = Logger.getGlobal
();

    @Resource
    private SessionContext sctx;

    private String transactionId;

    private boolean lastTransactionRollback;

    @TransactionalAttribute(NOT_SUPPORTED)
    public boolean isLastTransactionRollback() {
        return lastTransactionRollback;
    }

    public void afterBegin() {
        transactionId = Long.toString(System.
currentTimeMillis())
            + ThreadLocalRandom.current().nextLong(Long.
MAX_VALUE);

        LOGGER.log(Level.INFO, "Transaction TX_id={0} begin
in {1}, identity: {2}",
            new Object[]{transactionId, this.getClass().
getName(), sctx.getCallerPrincipal().getName()});
    }

    public void beforeCompletion() {
        LOGGER.log(Level.INFO, "Transaction TX_id={0} before
approval in {1}, identity: {2}",
            new Object[]{transactionId, this.getClass().
getName(), sctx.getCallerPrincipal().getName()});
    }

    public void afterCompletion(boolean committed) {
        lastTransactionRollback = !committed;

        LOGGER.log(Level.INFO, "Transaction TX_id={0} end in
{1}, by {3}, identity {2}",
            new Object[]{transactionId, this.getClass().
getName(), sctx.getCallerPrincipal().getName(),
            committed ? "APPROVAL" : "CANCELLATION"});
    }
}

```

Klasa CreateAccountService dziedzicząca po AbstractService oraz implementująca interfejs SessionSynchronization:

```

public class
CreateAccountService extends
AbstractService implements
CreateAccountServiceInterface
, SessionSynchronization {
    ...
}

```

12 Zgłoszane wyjątki 4 pkt

Wyjątki biznesowe:

| Nazwa wyjątku | Miejsce zgłoszenia wyjątku | Przyczyna wystąpienia wyjątku | Miejsce obsługi wyjątku | Klucz internacjonalizacji | Czy prowadzi do odwołania transakcji |
|-------------------------------------|------------------------------------|--|--|--|--------------------------------------|
| AccessLevelAlreadyExistsException | Serwis Interceptor | Próba dodania poziomu dostępu, który już jest przypisany do konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do UpdateAccountController i obsługiwany w metodach: addRenterAccess() i addServiceProviderAccess(). | access.level.already.exists.exception | Tak |
| AccessLevelCannotBeDeletedException | Serwis Interceptor | Próba usunięcia poziomu dostępu, który nie może zostać usunięty | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do UpdateAccountController i obsługiwany w metodzie deleteAccessLevel(). | access.level.cannot.be.deleted.exception | Tak |
| AccessLevelNotFoundException | Serwis Interceptor | Próba usunięcia poziomu dostępu z konta, które go nie posiada | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do UpdateAccountController i obsługiwany w metodzie deleteAccessLevel(). | access.level.not.found.exception | Tak |
| ApplicationBaseException | Kontroler Serwis Interceptor | Służy do zamiiany wyjątków kontenera / aplikacji na własny wyjątek aplikacyjny | Wyjątek po których dziedziczą wszystkie pozostałe wyjątki, które są później obsługiwane w kontrolerach. | brak | Tak |
| AuthorizationException | Serwis Interceptor | Brak autoryzacji potrzebnej do wykonania akcji | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do AuthController i obsługiwany w metodach: generateTokens() i refreshTokens() oraz do UpdateAccountService i obsługiwany w metodzie changeOwnPassword(). | authorization.exception | Tak |
| AuthUserNotFoundException | Interceptor | Podanie błędного loginu i/lub hasła podczas uwierzytelnienia | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do AuthController i obsługiwany w metodzie authenticate(). | auth.user.not.found.exception | Tak |
| EmailTakenException | Interceptor | Próba utworzenia konta z adresem email już przypisanym do innego konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do CreateAccountController i obsługiwany w metodach: createAdmin(), registerRenterUser() i registerServiceProviderUser(). | email.taken.exception | Tak |
| InvalidOfferDateException | Serwis Interceptor | Próba podania błędnej daty podczas wynajmowania oferty | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do HiringController i obsługiwany w metodzie: hireOffer() | invalid.offer.date.exception | Tak |
| LoginTakenException | Interceptor | Próba utworzenia konta z loginem już przypisanym do innego konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do CreateAccountController i obsługiwany w metodach: createAdmin(), registerRenterUser() i registerServiceProviderUser(). | login.taken.exception | Tak |

| | | | | | |
|------------------------------|--------------------------|---|--|---------------------------------|-----|
| NIPTakenException | Interceptor | Próba utworzenia konta firmy z numerem NIP już przypisany do innego konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do CreateAccountController i obsługiwany w metodzie: registerServiceProviderUser(). | nip.taken.exception | Tak |
| OfferAlreadyTakenException | Serwis Interceptor | Próba wynajęcia już wynajętej oferty | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do HiringController i obsługiwany w metodzie: hireOffer(). | offer.already.taken.exception | Tak |
| OfferInactiveException | Serwis Interceptor | Próba wynajęcia nieaktywnej oferty | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do HiringController i obsługiwany w metodzie: hireOffer(). | offer.inactive.exception | Tak |
| OfferNotFoundException | Interceptor | Próba ocenienia nieistniejącej oferty | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i obsługiwany w fasadzie OfferFacade. | offer.not.found.exception | Tak |
| OfferOwnerException | Serwis Interceptor | Próba usunięcia lub zmodyfikowania oferty dla błędnie podanego loginu zgłaszającego | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do OfferController i obsługiwany w metodach: deleteOfferServiceProvider() i editOfferServiceProvider(). | offer.owner.exception | Tak |
| PasswordAlreadyUsedException | Serwis Interceptor | Próba zmiany hasła na to, które obecnie jest używane przez użytkownika | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do updateAccountController i obsługiwany w metodzie changeUserOwnPassword(). | password.already.used.exception | Tak |
| RateOutOfRangeException | Serwis Interceptor | Próba podania oceny spoza zakresu 1-5 | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do RateController i obsługiwany w metodzie rate(). | rate.out.of.range.exception | Tak |
| ServiceAddressTakenException | Interceptor | Próba utworzenia konta firmy z adresem już przypisanym do innego konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do CreateAccountController i obsługiwany w metodzie: registerServiceProviderUser(). | service.address.taken.exception | Tak |
| ServiceAlreadyRatedException | Serwis Interceptor | Próba wielokrotnego ocenienia tego samego konta zgłaszającego | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do RateController i obsługiwany w metodzie: rate(). | service.already.rated.exception | Tak |
| ServiceNameTakenException | Interceptor | Próba utworzenia konta firmy z nazwą już przypisaną do innego konta | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do CreateAccountController i obsługiwany w metodzie: registerServiceProviderUser(). | service.name.taken.exception | Tak |
| TokenExpiredException | Kontroler Interceptor | Próba wykonania akcji, do której potrzebny jest żeton | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do UpdateAccountController i obsługiwany w metodzie: confirmResetPasswordEmail() oraz do CreateAccountController i obsługiwany w metodzie: confirm(). | token.expired.exception | Tak |
| UserNotFoundExeption | Interceptor | Niemogość znalezienia danego użytkownika | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i propagowany do odpowiedniego kontrolera gdzie jest także obsługiwany. Podany wyjątek jest propagowany do ReadAccountController i obsługiwany w metodzie: readAccountByLogin(). | user.not.found.exception | Tak |
| UnknownErrorExeption | Interceptor | Pozostałe wyjątki | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | unknown.error.exception | Tak |
| PersistanceErrorExeption | Interceptor | Problem z utrwalaniem encji w bazie danych lub w przypadku błędnego odczytu z bazy danych | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | persistence.error.exception | Tak |

| | | | | | |
|--------------------------|-------------|---|---|---------------------------|-----|
| Optimistic LockException | Interceptor | Wystąpienie blokady optymistycznej | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | optimistic.lock.exception | Tak |
| AccessDeniedException | Interceptor | Brak autoryzacji do wykonania metody | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | access.denied.exception | Tak |
| DataNotValidException | Interceptor | Dane wprowadzone przez użytkownika nie przeszły validacji | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | data.not.valid.exception | Tak |
| EmailSenderException | Interceptor | Niemogość wysłania maila na podany adres | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | email.sender.exception | Tak |

Wyjątki systemowe:

| Nazwa wyjątku | Miejsce zgłoszenia wyjątku | Przyczyna wystąpienia wyjątku | Miejsce obsługi wyjątku | Czy prowadzi do odwołania transakcji |
|-----------------------------------|----------------------------|---|--|--------------------------------------|
| EJBAccessException | Serwis Fasada | Brak autoryzacji do wykonania metody | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i zamieniany na AccessDeniedException. Po zamianie na AccessDeniedException, może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | Tak |
| EJBException | Serwis Fasada | Zgłaszany, gdy klient nie propagował transakcji do wywołanej metody biznesowej komponentu EJB, w ramach wykonania której został zgłoszony wyjątek nieaplikacyjny lub kiedy klient wywołał metodę z atrybutem transakcyjnym TransactionAttributeType.NEVER | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | Tak |
| EJBTransactionRequiredException | Serwis Fasada | Zgłaszany, jeżeli klient nie posiada kontekstu transakcji, który był wymagany przez metodę biznesową. | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | Tak |
| EJBTransactionRolledbackException | Serwis Fasada | Zgłaszany, jeżeli klient propagował odwołaną transakcję do wywoływanej metody biznesowej komponentu EJB. | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | Tak |
| NoSuchEJBException | Serwis Fasada | Zgłaszany, jeżeli odwołania klienta odnoszą się do usuniętego komponentu EJB. | Wyjątek jest obsługiwany w warstwie logiki biznesowej. Jest łapany w interceptorze i może być propagowany do każdego z kontrolerów gdzie może zostać obsłużony przez każdą z metod z tego kontrolera. | Tak |

13 Interfejs użytkownika 3 pkt

Należy omówić sposób tworzenia szablonu stron, przedstawiając stosowny kod szablonu.

Należy też przedstawić mechanizm doboru strony głównej i/lub konstruowania menu po poprawnym zakończeniu procesu uwierzytelniania w aplikacji, dla konta z kilkoma poziomami dostępu. Forma - zacytowane fragmenty kodu stanowiące rozwiązanie.

Interfejs użytkownika tworzy aplikacja internetowa SPA stworzona w języku TypeScript z wykorzystaniem biblioteki React.js zbudowany w oparciu o pakiet create-react-app. Do dynamicznego routingu wykorzystano pakiet react-router-dom.

Konfiguracja dynamicznego routingu:

```
{...}

<Routes>
  <Route path="/" element={<Layout />}>
    <Route path="/" element={<Home />} />
    <Route
      path="register"
      element={
        <Protected permitGuestsOnly>
          <Register />
        </Protected>
      }
    />
    <Route
      path="login"
      element={
        <Protected permitGuestsOnly>
          <Login />
        </Protected>
      }
    />
    <Route
      path="passwordReset"
      element={
        <Protected permitGuestsOnly>
          <PasswordReset />
        </Protected>
      }
    />
    <Route
      path="passwordReset/:token"
      element={
        <Protected permitGuestsOnly>
          <PasswordReset />
        </Protected>
      }
    />
    <Route
      path="admin"
      element={
        <Protected permitRoles={"Admin"}>
          <AdminPanel />
        </Protected>
      }
    />
    <Route
      path="/admin/changePassword"
      element={
        <Protected permitRoles={"Admin"}>
          <ChangePassword />
        </Protected>
      }
    />
```

```
</Protected>
}
/>
<Route
path="/admin/changeUser"
element={
  <Protected permitRoles={"Admin"}>
    <ChangeUserDetails />
  </Protected>
}
/>

<Route
path="myaccount/edit"
element={
  <Protected permitLoggedInOnly>
    <MyAccountEdit />
  </Protected>
}
/>
<Route
path="myaccount/changePassword"
element={
  <Protected permitLoggedInOnly>
    <ChangeUserPasswordForm />
  </Protected>
}
/>

<Route
path="myoffers"
element={
  <Protected permitRoles={"ServiceProvider"}>
    <MyOffers />
  </Protected>
}
/>

<Route
path="serviceProviders"
element={
  <Protected permitLoggedInOnly>
    <ServiceProviders />
  </Protected>
}
/>

<Route
path="serviceProviders/:serviceName"
element={<ServiceProviderDetails />}
/>

<Route
path="myRents"
element={
  <Protected permitRoles={"Renter"}>
    <MyRents />
  </Protected>
}
/>

<Route path="offers" element={<AvailableOffers />} />
<Route path="tokenExpired" element={<TokenExpiredPage />} />
<Route
path="tokenConfirmed"
element={<TokenConfirmationPage />}
/>
<Route
path="/unauthorized"
element={<UnauthorizedErrorPage />}
/>
<Route path="/" element={<NotFoundPage />} />
```

```
</Route>
</Routes>
```

{...}

Frontend/src/App.tsx

Niektóre komponenty są zagnieżdżone w komponencie Protected, który został omówiony w [14 Zabezpieczenia interfejsu użytkownika 3 pkt.](#)

Wszystkie widoki są zagnieżdżone w ścieżce "", która renderuje komponent Layout. Komponent Layout renderuje warunkowo odpowiednie menu w zależności od stanu zalogowania użytkownika, a także - korzystając ze specyfiki zagnieżdżania ścieżek w react-router-dom w wersji v6 - komponent zagnieźdzony w zależności od aktualnego adresu URL za pomocą komponentu Outlet z wspomnianego pakietu.

```
export default function Layout() {
  const { isLoggedIn } = useAppSelector((state) => state);

  return <>{isLoggedIn ? <LoggedInNavbar /> : <LoggedOutNavbar />}</>;
}
```

Frontend/src/common/components/Layout.tsx

W komponencie LoggedInNavbar zawarto mechanizm konstruowania menu po poprawnym zakończeniu procesu uwierzytelniania w aplikacji oparty na renderowaniu warunkowym przycisków menu na podstawie bieżącego poziomu dostępu użytkownika, który jest przechowywany w stanie aplikacji. Aktualizacja bieżącego poziomu dostępu użytkownika w stanie aplikacji została przedstawiona w [14 Zabezpieczenia interfejsu użytkownika 3 pkt.](#)

```
const LoggedInNavbar = () => {
  const theme = useMantineTheme();
  const [opened, setOpened] = useState(false);
  const dispatch = useAppDispatch();
  const accessLevel = useAppSelector((state) => state?.user);
  const activeView = useAppSelector((state) => state?.activeView);

  let bgColor = "";
  if (activeView === "Admin") {
    bgColor = "red";
  } else if (activeView === "Renter") {
    bgColor = "yellow";
  } else if (activeView === "ServiceProvider") {
    bgColor = "blueviolet";
  } else {
    bgColor = "";
  }

  const { t } = useTranslation();

  return (
    <AppShell
      padding="md"
      header={
        <Header
          height={70}
          p="md"
          sx={{ backgroundColor: bgColor, position: "relative", zIndex: 10 }}
        >
          <div
            style={{
              display: "flex",
              gap: "10px",
              padding: "10px 0",
            }}
          >
            {accessLevel === "Admin" ? (
              <UserIcon
                user={user}
                onClick={() => dispatch(openModal("Profile"))}
              />
            ) : null}
            {accessLevel === "Renter" ? (
              <UserIcon
                user={user}
                onClick={() => dispatch(openModal("Profile"))}
              />
            ) : null}
            {accessLevel === "ServiceProvider" ? (
              <UserIcon
                user={user}
                onClick={() => dispatch(openModal("Profile"))}
              />
            ) : null}
            {accessLevel === "Guest" ? (
              <UserIcon
                user={user}
                onClick={() => dispatch(openModal("Profile"))}
              />
            ) : null}
            {accessLevel === "Anonymous" ? (
              <UserIcon
                user={user}
                onClick={() => dispatch(openModal("Profile"))}
              />
            ) : null}
          </div>
        </Header>
      }
    </AppShell>
  );
}
```

```

alignItems: "center",
height: "100%",
justifyContent: "space-between"
)}
>
<MediaQuery largerThan="sm" styles={{ display: "none" }}>
<Burger
opened={opened}
onClick={() => setOpened((o) => !o)}
size="sm"
color={theme.colors.gray[6]}
mr="xl"
/>
</MediaQuery>

<Text weight={700}>EStrady</Text>

<Center sx={{ display: "flex", gap: "10px" }}>
{accessLevel ? (
<>
<Button
component={Link}
to="/"
variant="filled"
color="green"
radius="md"
size="md"
>
{t("layout-Navbar-MainView")}
</Button>
<Button
component={Link}
to="offers"
variant="filled"
color="green"
radius="md"
size="md"
>
{t("layout-Navbar-Offers")}
</Button>
<Button
component={Link}
to="/serviceProviders"
variant="filled"
color="green"
radius="md"
size="md"
>
{t("layout-Navbar-Companies")}
</Button>

{activeView === "Admin" ? (
<Button
component={Link}
to="admin"
variant="filled"
color="green"
radius="md"
size="md"
>
{t("layout-Navbar-AdminPanel")}
</Button>
) : null}
{activeView === "ServiceProvider" ? (
<Button
component={Link}
to="myoffers"
variant="filled"
color="green"
radius="md"
size="md"
>
{t("layout-Navbar-ServiceProviderPanel")}
</Button>
) : null}

```

```
    {t("layout-Navbar-MyOffers")}  
    <Button>  
    : null  
    {activeView === "Renter" ? (  
      <Button  
        component={Link}  
        to="myRents"  
        variant="filled"  
        color="green"  
        radius="md"  
        size="md"  
      >  
      {t("layout-Navbar-MyRents")}  
      </Button>  
    ) : null}  
    <Button  
      component={Link}  
      to="/"  
      variant="filled"  
      color="green"  
      radius="md"  
      size="md"  
      onClick={() => dispatch(logout())}  
    >  
    {t("layout-Navbar-Logout")}  
    </Button>  
    <Button  
      variant="filled"  
      color="green"  
      radius="md"  
      size="md"  
      onClick={() => setOpened(true)}  
    >  
    {t("layout-Navbar-MyAccount")}  
    </Button>  
    </>  
  ) : null}  
  
  <DarkModeSwitch />  
  </Center>  
  </div>  
  <Header>  
}<br/>  
footer={  
  <>  
    <Box sx={{ position: "relative", zIndex: 1, height: "60px" }}></Box>  
    <Footer  
      height={60}  
      p="md"  
      sx={{ position: "fixed", bottom: "0", "z-index": "5" }}  
    >  
      footer  
    </Footer>  
    </>  
  </>  
}  
styles={({theme}) => ({  
  main: {  
    backgroundColor:  
      theme.colorScheme === "dark"  
        ? theme.colors.dark[8]  
        : theme.colors.gray[0],  
    padding: 0,  
    zIndex: 10,  
    position: "relative"  
  },  
  minHeight: "100vh",  
  body: { minHeight: "calc(100vh - 130px)" }  
})}  
>  
<Drawer  
  opened={opened}  
  onClose={() => setOpened(false)}>
```

```
title={t("myAccount-modifyAccount-Title")}  
padding="xl"  
size="xl"  
position={"right"}  
>  
<MyAccount setOpened={setOpened} />  
</Drawer>  
<Outlet />  
</AppShell>  
);  
};  
  
export default LoggedInNavbar;
```

Frontend/src/common/components/LoggedInNavbar.tsx

Zmiana bieżącego poziomu dostępu jest możliwa tylko dla kont z poziomem dostępu administratora, pozostałe konta mogą posiadać tylko jeden poziom dostępu. Funkcjonalność zmiany poziomu dostępu znajduje się w widoku MyAccount:

```
{...}  
  
{user?.accessLevelDTOs[0]?.accessLevel == "Admin" && (  
<Select  
label={t("myAccount-changeView-inputLabel")}  
onChange={(value: string) => changeAccessView(value)}  
defaultValue={activeView}  
data={user?.accessLevelDTOs.map((accessLevel) => {  
  return {  
    value: accessLevel.accessLevel,  
    label: accessLevel.accessLevel  
  };  
});  
)  
>  
)  
  
{...}
```

Frontend/src/pages/MyAccount.tsx

Z widoków wyodrębniono komponenty dostarczające poszczególne funkcjonalności takie jak obsługa logowania użytkownika za pomocą formularza.

```

function Login() {
  const user = useSelector((state: RootState) => state?.user);

  return !user ? (
    <Center sx={{ width: "100%", minHeight: "100%" }}>
      <LoginForm sx={{ width: "min(50%, 500px)" }} />
    </Center>
  ) : (
    <div>
      <p>
        Login:
        {user.login}
      </p>
      <p>
        Email:
        {user.email}
      </p>
    </div>
  );
}

export default Login;

```

Frontend/src/pages/Login.tsx

```
interface LoginFormProps extends MantineStyleSystemProps, DefaultProps {}
```

```

function LoginForm({ sx, ...rest }: LoginFormProps) {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [isRequestOnGoing, setIsRequestOnGoing] = useState<boolean>(false);
  const [response, setResponse] = useState<string>("");
  const dispatch = useAppDispatch();
  const navigate = useNavigate();

  const { t } = useTranslation();

  const handleSubmit = async (e: React.ChangeEvent<HTMLFormElement>) => {
    e.preventDefault();
    setIsRequestOnGoing(true);

    const requestResponse = await login(username, password);
    if (requestResponse === "Accepted") {
      dispatch(updateUser());
      navigate("/");
    }
    setResponse(requestResponse);
    setIsRequestOnGoing(false);
  };

  const notificationsHandler = () => {
    if (isRequestOnGoing && response === "") {
      return <Loader variant="bars" />;
    } else if (response === "Accepted") {
      return (
        <Notification
          disallowClose={true}
          icon={<Check size={18} />}
          color="teal"
          title="Poprawnie zalogowano"
        ></Notification>
      );
    } else if (response !== "Accepted" && response !== "") {
      return (
        <Notification
          icon={<X size={18} />}
        ></Notification>
      );
    }
  };
}
```

```

        color="red"
        onClose={() => {
          setResponse("");
        }}
      >
    {response}
  </Notification>
);
} else {
  return (
    <Button type="submit" mt="md" fullWidth>
      {t("LoginForm-body-button-login")}
    </Button>
  );
}
};

return (
<Box sx={sx} component="section" {...rest}>
  <Title order={1} pb="md">
    {t("LoginForm-title")}
  </Title>
  <Box
    component="form"
    sx={{
      display: "flex",
      flexDirection: "column",
      rowGap: "8px"
    }}
    onSubmit={handleSubmit}
  >
    <TextInput
      required
      label={t("LoginForm-body-mail-label")}
      placeholder={t("LoginForm-body-mail-placeholder")}
      onChange={(e) => setUsername(e.target.value)}
    />
    <PasswordInput
      required
      label={t("LoginForm-body-password-label")}
      placeholder={t("LoginForm-body-password-placeholder")}
      onChange={(e) => setPassword(e.target.value)}
    />
    <UnstyledButton type="button" component={Link} to="/passwordReset">
      {t("LoginForm-body-button-passwordReset")}
    </UnstyledButton>
    <Group position="center" mt="md">
      {notificationsHandler()}
    </Group>
  </Box>
</Box>
);
};

export default LoginForm;

```

Frontend/src/features/Login/LoginForm.tsx

W aplikacji wykorzystano pakiet @mantine/core zawierający ponad 120 konfigurowalnych komponentów zwracających ostylowane elementy HTML, a także @mantine/modals do wyświetlania okien dialogowych.

14 Zabezpieczenia interfejsu użytkownika 3 pkt

Konfiguracja kontroli dostępu użytkownika do widoków z interfejsem użytkownika:

Klasa Protected

```
export default function Protected({
  permitRoles,
  children,
  permitLoggedInOnly,
  permitGuestsOnly
}: ProtectedType) {
  const activeView: string | null = useAppSelector(
    (state) => state?.activeView
  );
  const { isLoggedIn } = useAppSelector((state) => state);

  if (
    (permitGuestsOnly && !isLoggedIn) ||
    (permitLoggedInOnly && isLoggedIn) ||
    (typeof permitRoles === "string" &&
      permitRoles.localeCompare(activeView) === 0) ||
    (typeof permitRoles === "object" && permitRoles.includes(activeView))
  ) {
    return children;
  }

  return <Navigate to="/unauthorized" replace />;
}
```

Wykorzystanie klasy Protected

```
<Route
  path="admin"
  element={
    <Protected permitRoles={"Admin"}>
      <AdminPanel />
    </Protected>
  }
/>
```

Warunkowe wyświetlanie formularza

```
const MyAccountEdit = () => {
  const activeView: string = useAppSelector((state) => state?.activeView);

  useEffect(() => {
    store.dispatch(updateUser());
  }, []);

  return (
    <>
      {activeView.localeCompare("Admin") === 0 && <EditAdminAccountForm />}
      {activeView.localeCompare("Renter") === 0 && <EditRenterAccountForm />}
      {activeView.localeCompare("ServiceProvider") === 0 && (
        <EditServiceProviderAccountForm />
      )}
    </>
  );
};
```

Tożsamość użytkowników na role aplikacji w warstwie widoku:

Ustawienie pola activeView - Redux

```
export const authSlice = createSlice({
  name: "auth",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(logout.fulfilled, () => {
        return initialState;
      })
      .addCase(updateUser.fulfilled, (state, action) => {
        state.user = <UserDataType>action.payload;
        if (!state.activeView) {
          state.activeView = state.user.accessLevelDTOs[0].accessLevel;
        }
        state.isUserLoggedIn = true;
      })
      .addCase(setActiveView.fulfilled, (state, action) => {
        state.activeView = action.payload;
      });
  }
});
```

Ustawienie pola activeView - LocalStorage

```
export const setActiveView = createAsyncThunk(
  "auth/setActiveView",
  (view: string) => {
    localStorage.setItem("activeView", view);
    return view;
  }
);
```

Wyliczanie i składowanie haseł przypisanych do kont użytkowników w bazie danych w postaci niejawniej:

Klasa SHA256HashGenerator

```
public class SHA256HashGenerator implements HashGenerator {

    @Override
    public String generateHash(String input) {
        try {
            return bytesToHex(
                MessageDigest.getInstance("SHA-256").digest(
                    input.getBytes(StandardCharsets.UTF_8)));
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(SHA256HashGenerator.class.getName()).log(Level.SEVERE, null, ex);
            throw new IllegalStateException(ex);
        }
    }

    private static String bytesToHex(byte[] hash) {
        StringBuilder hexString = new StringBuilder(2 * hash.length);
        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

Przykład wykorzystania:

Przykład wykorzystania generatora

```
public void createAdminAccount(Account account, AccessLevel accessLevel) {
    account.setPassword(hashGenerator.generateHash(account.getPassword()));
    accessLevel.setAccount(account);
    account.getAccessLevelCollection().add(accessLevel);
    account.setConfirmed(true);
    accountFacade.create(account);
}
```

15 Zmiany - projekt szczegółowy

Dodano przypadek użycia MOK.15 w [Rozdziale 2](#).

Poprawiono przypadek użycia SYSTEM.2 w [Rozdziale 2](#).

Dodano tabelę Verification_token w [Rozdziale 3](#) , [Rozdziale 4](#) , [Rozdziale 5](#) oraz w [Rozdziale 6](#).

Dodano diagram sekwencji dla przypadku użycia MOK.15 w [Rozdziale 7](#).

Zmieniono diagramy sekwencji (1.1, 3, 4, 13) w [Rozdziale 7](#).

Zmieniono skrypt inicjujący bazę danych (dodanie tabeli Verification_token oraz zmiana danych inicjujących) - [Rozdział 3](#).

Wprowadzono poprawki błędów wymienionych podczas etapu wstępnego (powtórzenia pojęć w słownikach, odwrócona interpretacja właściciela związku w tabeli, nie wymieniono podpisu, który potwierdza autentyczność tokenu) w [Rozdziale 1](#) , [Rozdziale 5](#) oraz w [Rozdziale 8](#).

Sprawozdanie końcowe (10 pkt)

Sprawozdanie końcowe

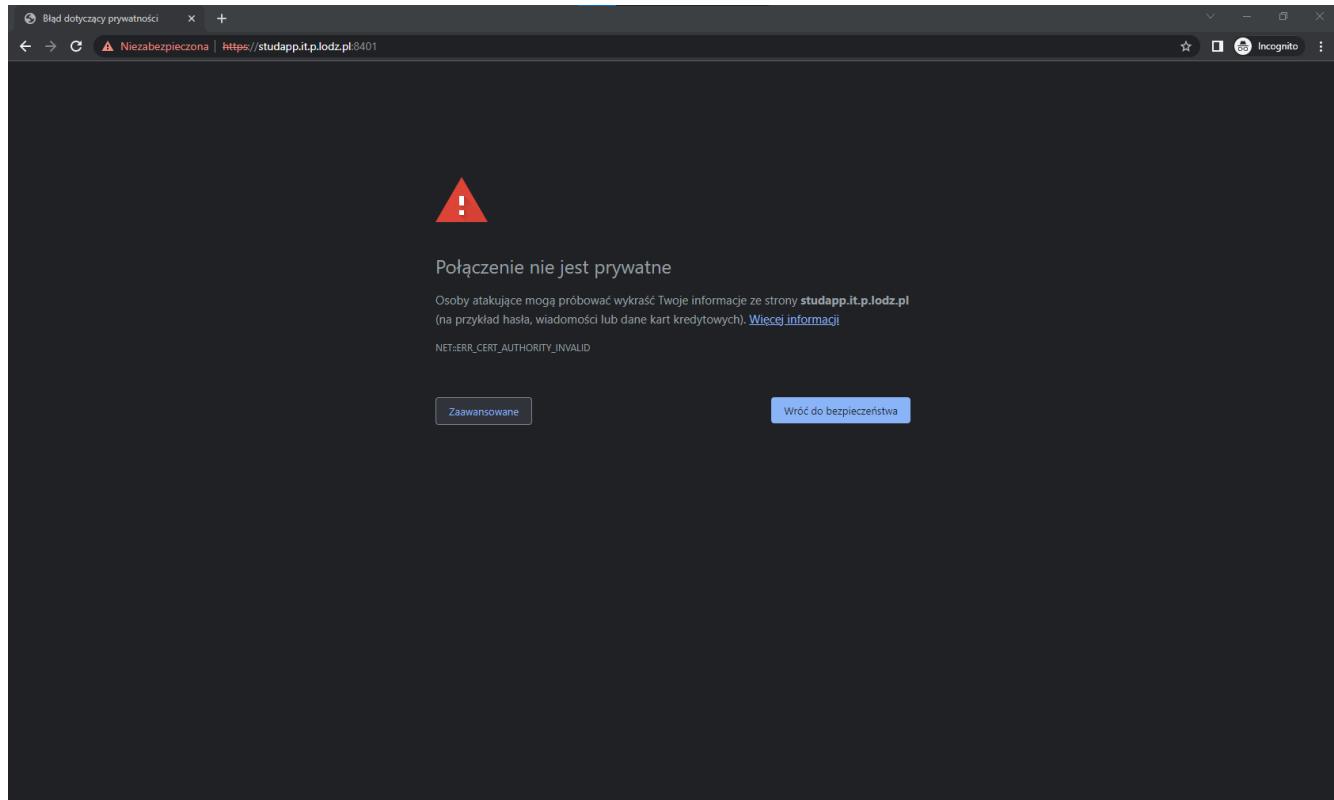
16 Zabezpieczenie transmisji HTTP 1 pkt

W celu zabezpieczenia transmisji HTTP jest wymuszenie wykorzystania zaszyfrowanego protokołu HTTPS podczas komunikacji między aplikacją SPA uruchomionej w przeglądarce a serwerem aplikacji. Wymuszenie to zostało skonfigurowane z deskryptorem web.xml (patrz [Obraz 1.](#)), opisuje to element <transport-guarantee>CONFIDENTIAL</transport-guarantee> na wszystkich możliwych ścieżkach co pokazuje element <url-pattern>/*</url-pattern>.

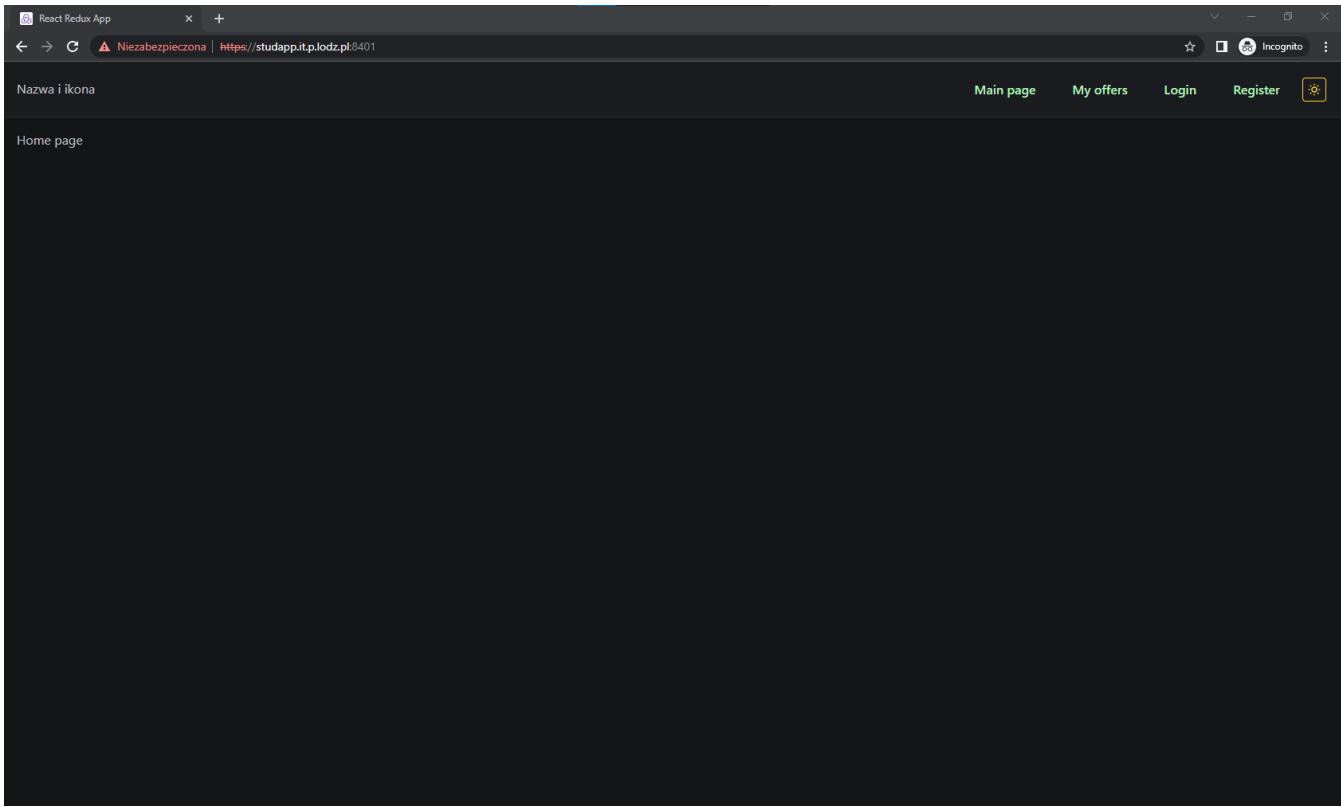
```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>All</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

[Obraz 1.](#) Wycinek kodu z pliku web.xml

Powodem ostrzeżenia mówiącego iż połaczenie nie jest prywatne jest certyfikat wystawiany przez domenę studapp.it.p.lodz.pl który jest certyfikatem samopodpisany, co za tym idzie urząd wystawiający ten certyfikat jest nie zaufany ponieważ nie znajduje się w magazynie zaufanych głównych urzędów certyfikacji. Jednakże można wymusić w przeglądarce połaczenie poprzez akceptację ryzyka (w przypadku przeglądarki Chrome należy wcisnąć przycisk zaawansowane a następnie kliknąć link prowadzący do aplikacji)



[Obraz 2.](#) Pierwsze nawiązanie połączenia z serwerem aplikacji



//TODO zmienić zrzut main page-a

17 Weryfikacja poprawnosci danych 2 pkt

Konwerter klas encji

W aplikacji został użyty wzorzec DTO w celu zabezpieczenia encji przed wyjściem do warstwy widoku. W celu mapowania obiektów encji na obiekty DTO zostały stworzone mapery pozwalające na taką operację OBRAZ 1 oraz jawne jego wykorzystanie OBRAZ 2

```
/**  
 * Mapper odpowiedzialny za mapowanie encji na odpowiednie DTO dla żądań GET  
 */  
  
3 usages  
public class GetMoMapper {  
  
    1 usage  
    public static OfferViewDTO getOfferDTO(Offer offer) {  
        return new OfferViewDTO(  
            offer.getId(),  
            offer.getDescription(),  
            offer.getPrice(),  
            offer.getServiceProvider().getServiceProviderName()  
        );  
    }  
  
    1 usage  
    public static List<OfferViewDTO> getOfferDTOList(Collection<Offer> offers) {  
        return offers.stream().filter(Objects::nonNull).  
            .map(GetMoMapper::getOfferDTO).  
            .collect(Collectors.toUnmodifiableList());  
    }  
}
```

Obraz 1. Implementacja mapera

```
@GET @  
@PermitAll  
@Produces(MediaType.APPLICATION_JSON)  
public Response getAllOffers() {  
    try {  
        return Response.ok().entity(GetMoMapper.getOfferDTOList(browseOfferService.getAllOffers())).build();  
    } catch (ApplicationBaseException exc) {  
        throw exc;  
    } catch (EJBAccessException | AccessLocalException exc) {  
        throw ApplicationBaseException.getAccessDeniedException(exc);  
    } catch (Exception exc) {  
        throw ApplicationBaseException.getGeneralErrorException(exc);  
    }  
}
```

Obraz 2. Jawne wykorzystanie mapera przez metodę getAllOffers()

Walidacja danych w warstwie prezentacji

W warstwie prezentacji podczas wprowadzania danych zostały wykorzystane formularze wraz z metodami walidującymi w których np. w tym przypadku sprawdzane jest czy podane hasło oraz jego potwierdzenie są takie same oraz czy pokrywają się z wyrażeniem regex (patrz [Obraz 3.](#)).

```
const registrationForm = useForm( {initialValues, initialErrors, validate, rules, schema}: {  
    initialValues: {  
        login: "",  
        password: "",  
        repeatedPassword: "",  
        name: "",  
        surname: "",  
        email: "",  
        phoneNumber: "",  
        renterName: "",  
        providerName: "",  
        nip: "",  
        address: ""  
    },  
  
    validate: {  
        password: (value) => passwordValidation(value),  
        repeatedPassword: (value, values) =>  
            value === values.password  
                ? null  
                : t("registerForm-Validation-RepeatedPassword"),  
        email: (value) =>  
            /\S+@\S+/.test(value) ? null : t("registerForm-Validation-Email"),  
        phoneNumber: (value) =>  
            /\d{9}$/.test(value) ? null : t("registerForm-Validation-PhoneNumber"),  
        nip: (value) =>  
            accountType === "renter" || /\d{10}$/.test(value)  
                ? null  
                : t("registerForm-Validation-NIP")  
    }  
});
```

[Obraz 3.](#) Formularz rejestracji wraz z walidacją odpowiednich danych

Walidacja przez adnotację BeanValidation

Walidacja poprzez adnotację BeanValidation znajduje się w projekcie zarówno w obiektach DTO (patrz [Obraz 4](#)) jak i w klasach encyjnych (patrz [Obraz 5](#)). Ograniczenia te pokrywają się z ograniczeniami z warstwy prezentacji oraz z tymi umieszczonymi w bazie danych. Niespełnienie wymogów ograniczeń, w przypadku DTO podczas dostarczania do metod kontrolerów (patrz [Obraz 6.](#)) a w przypadku encji próby zapisaniem niepoprawnych danych do bazy danych (patrz [Obraz 7.](#)) skutkuje wyrzuceniem błędu ValidationException który obsługiwany jest w każdej z warstw aplikacji backendowej.

```

@Getter
@Setter
@NoArgsConstructor
public class AddServiceProviderDTO {

    @Size(min = 2, max = 32)
    @NotBlank
    private String serviceName;

    @Pattern(regexp = "^\d{10}$")
    @NotBlank
    private String nip;

    @Size(min = 4, max = 100)
    @NotBlank
    private String address;

    @NotEmpty
    private String description;

    @NotEmpty
    private String logoUrl;

}

```

Obraz 4. Przykładowe ograniczenia dla obiektu DTO

```

public class ServiceProviderDetails extends AccessLevel implements Serializable {

    2 usages
    @Basic(optional = false)
    @Size(min = 2, max = 32)
    @Column(name = "service_name", unique = true)
    @Getter
    @Setter
    private String serviceName;

    2 usages
    @Basic(optional = false)
    @Size(min = 10, max = 10)
    @Pattern(regexp = "^\d{10}$")
    @Column(name = "nip", updatable = false, unique = true)
    @Getter
    @Setter
    private String nip;

    2 usages
    @Basic(optional = false)
    @Column(name = "address", unique = true)
    @Size(min = 4, max = 100)
    @Getter
    @Setter
    private String address;

    1 usage
    @Basic(optional = false)
    @Column(name = "description")
    @Getter
    @Setter
    private String description;
}

```

Obraz 5. Przykładowe ograniczenia dla obiektu encji

```
@POST  
@PermitAll  
@Path("/register-provider")  
@Consumes(MediaType.APPLICATION_JSON)  
public Response registerServiceProviderUser(@Valid PostServiceProviderDTO postServiceProviderDTO) {  
    int retryTXCounter = 3;  
    boolean rollbackTX;  
    do {  
        try {  
            createAccountService.createServiceProviderAccount(  
                PostMokMapper.convertToAccount(postServiceProviderDTO),  
                PostMokMapper.convertToServiceProviderDetails(postServiceProviderDTO)  
        );  
    } catch (Exception e) {  
        if (retryTXCounter > 0) {  
            rollbackTX = true;  
            retryTXCounter--;  
        } else {  
            throw e;  
        }  
    }  
}
```

Obraz 6. Weryfikacja obiektu DTO dostarczonego do metody kontrolera

```
@RolesAllowed("Admin")  
@Override  
public void createAdminAccount(Account account, AccessLevel accessLevel) {  
    account.setPassword(hashGenerator.generateHash(account.getPassword()));  
    accessLevel.setAccount(account);  
    account.getAccessLevelCollection().add(accessLevel);  
    account.setConfirmed(true);  
    accountFacade.create(account);  
}
```

Obraz 7. W przypadku wywołania metody create na fasadzie accountFacade zostanie wyrzucony błąd ValidationException który zostanie przechwycony w interceptorze i zmieniony na błąd aplikacyjny.

18 Obsługa błędów 2 pkt

Obsługa błędów w aplikacji odbywa się w interceptorach. Łapane są tam wyjątki aplikacyjne oraz wyjątki systemowe a następnie propagowane są one do wyjątków ApplicationBaseException oraz łapane w poszczególnych kontrolerach.

Informowanie użytkownika o błędach pojawiających się w warstwie logiki aplikacji obsługiwane jest poprzez przesłanie w ciele odpowiedzi HTTP wiadomości błędu ze złapanego wyjątku.

Obsługa przykładowego wyjątku aplikacyjnego:

```
@RolesAllowed("Admin")
@Override
public void changePassword(String login, String newPassword, String ETag) {
    Account account = accountFacade.findByLogin(login);
    if (ETag.equals(HMAC.calculateHMAC(account.getLogin(), account.getVersion()))) {
        if (account.getPassword().equals(hashGenerator.generateHash(newPassword))) {
            throw new PasswordAlreadyUsedException();
        }
        account.setPassword(hashGenerator.generateHash(newPassword));
        accountFacade.edit(account);
    } else {
        throw new OptimisticLockException();
    }
}
```

Rzucenie wyjątku PasswordAlreadyUsedException rozszerzającego ApplicationBaseException w metodzie serwisowej

```
public class PasswordAlreadyUsedException extends ApplicationBaseException {

    private static final Response.Status DEFAULT_STATUS = Response.Status.CONFLICT;

    private static final String DEFAULT_MESSAGE = "password.already.used.exception";

    public PasswordAlreadyUsedException() {
        super(DEFAULT_STATUS, DEFAULT_MESSAGE);
    }

    public PasswordAlreadyUsedException(Throwable cause) { super(DEFAULT_STATUS, DEFAULT_MESSAGE, cause); }
}
```

```
public class GenericServiceInterceptor {

    @AroundInvoke
    public Object interceptor(InvocationContext ictx) throws Exception {
        try {
            return ictx.proceed();
        } catch (ApplicationBaseException exc) {
            throw exc;
        } catch (ValidationException exc) {
            throw ApplicationBaseException.getValidationException(exc);
        } catch (OptimisticLockException exc) {
            throw ApplicationBaseException.getOptimisticLockException(exc);
        } catch (EJBAccessException | AccessLocalException exc) {
            throw ApplicationBaseException.getAccessDeniedException(exc);
        } catch (Exception exc) {
            throw ApplicationBaseException.getGeneralErrorException(exc);
        }
    }
}
```

Wyjątki ApplicationBaseException łapane są w interceptorze a następnie przerzucane dalej by mogły być złapane w warstwie kontrolerów.

```
@PUT
@RolesAllowed("Admin")
@Path("/changeUserPassword")
public Response changeUserPassword(@QueryParam("login") String login,
    int retryTXCounter = 3;
    boolean rollbackTX;
    do {
        try {
            updateAccountService.changePassword(login, newPassword, httpMethod);
            rollbackTX = updateAccountService.isLastTransactionRollback();
            if (rollbackTX) LOGGER.info("*** Transaction rollback");
        } catch (ApplicationBaseException exc) {
            throw exc;
        } catch (EJBAccessException | AccessLocalException exc) {
            throw ApplicationBaseException.getAccessDeniedException(exc);
        } catch (Exception exc) {
            throw ApplicationBaseException.getGeneralErrorException(exc);
        }
    } while (rollbackTX && --retryTXCounter > 0);

    if (rollbackTX) {
        throw ApplicationBaseException.getGeneralErrorException();
    }

    return Response
        .ok()
        .build();
}
```

Wyjątek jest następnie łapany w kontrolerze a jego wiadomość wysyłana jest w ciele odpowiedzi HTTP.

W warstwie prezentacji w zależności od otrzymanej wiadomości wyświetlany jest zinternacjonalizowany tekst z informacją o błędzie.

```
.then((res: AxiosResponse) => {
  return "OK";
  // return res.statusText;
})
.catch((err: AxiosError) => {
  if (err.response?.data.includes("password.already.used.exception")) {
    return i18next.t(
      "adminPanel-UserList-ChangePasswordView-passwordInput-errorNotification-passwordExists"
    );
  }
  return i18next.t(
    "adminPanel-UserList-ChangePasswordView-passwordInput-errorNotification-unknown"
  );
});
```

Wyjątki systemowe są obsługiwane w taki sam sposób. W interceptorze łapany jest dany wyjątek a następnie propagowany jest na wyjątek ApplicationBaseException

W interceptorach łapany jest wyjątek klasy Exception przez co wszystkie nie przewidziane błędy są również obsłużone. Nieprzewidziany wyjątek jest propagowany do wyjątku ApplicationBaseException i rzucony dalej. Kontroler łapie rzucony wyjątek a wiadomość o niespodziewanym błędzie przesłana zostaje do warstwy aplikacji i odpowiednia wiadomość zostaje wyświetlona dla użytkownika.

```
public static ApplicationBaseException getGeneralErrorException(Exception cause) {
  return new ApplicationBaseException(Response.Status.INTERNAL_SERVER_ERROR, UNKNOWN_ERROR_EXCEPTION, cause);
}
```

W warstwie aplikacji jest również zapewnione przekierowywanie użytkowników na odpowiednie podstrony w momencie gdy użytkownik próbuje wejść na podstronę o adresie który nie istnieje, lub gdy użytkownik próbuje dostać się na podstronę do której nie ma dostępu.

```
<Route
  path="/unauthorized"
  element={<UnauthorizedErrorPage />}
/>
<Route path="/" element={<NotFoundPage />} />
</Route>
```



Brak dostępu do danego zasobu

19 Wersje językowe interfejsu użytkownika 1 pkt

W warstwie prezentacji została wykorzystana biblioteka react-i18next która udostępnia mechanizm internacjonalizacji polegający na wyświetlaniu ze słownika komunikatów zgodnych z aktualnymi preferencjami przeglądarki użytkownika.

Stworzone zostały dwa słowniki komunikatów w wersji językowej polskiej oraz angielskiej.

Na poniższych zrzutach ekranów przedstawiono obie wersje słownika dla części interfejsu użytkownika związanego z rejestracją użytkownika w tym walidacja danych oraz informacje o wyjątkach rzuconych w warstwie logiki .

```
"registerForm-PasswordHint-Hint-1": "Conajmniej 8 znaków",
"registerForm-PasswordHint-Hint-2": "Conajmniej 1 duża oraz 1 mała litera",
"registerForm-PasswordHint-Hint-3": "Conajmniej 1 cyfra",
"registerForm-Validation-Password": "Hasło nie spełnia wymagań",
"registerForm-Validation-RepeatedPassword": "Hasło różni się od siebie",
"registerForm-Validation-Email": "Nieprawidłowy mail",
"registerForm-Validation-PhoneNumber": "Numer telefonu powinien składać się z 9 cyfr",
"registerForm-Validation-NIP": "Numer NIP powinien składać się z 10 cyfr",
"registerForm-SuccessfulRegistrationNotification-Title": "Konto zostało stworzone",
"registerForm-SuccessfulRegistrationNotification-RenterInfo": "Mail weryfikacyjny został wysłany na podany adres e-mail.",
"registerForm-SuccessfulRegistrationNotification-ProviderInfo": "Konto oczekuje na weryfikację przez administrację serwisu.",
"registerForm-RequestError-LoginTaken": "Podany login jest zajęty",
"registerForm-RequestError-EmailTaken": "Podany email jest zajęty",
"registerForm-RequestError-NIPTaken": "Firma o podanym numerze NIP jest już zarejestrowana w serwisie",
"registerForm-RequestError-UnexpectedError": "Wystąpił błąd. Proszę spróbować ponownie",
```

```
"registerForm-PasswordHint-Hint-1": "At least 8 characters",
"registerForm-PasswordHint-Hint-2": "At least 1 upper-case and lower-case letter",
"registerForm-PasswordHint-Hint-3": "At least 1 digit",
"registerForm-Validation-Password": "Password does not meet the requirements",
"registerForm-Validation-RepeatedPassword": "Passwords do not match",
"registerForm-Validation-Email": "Invalid mail",
"registerForm-Validation-PhoneNumber": "Phone number must contains 9 digits",
"registerForm-Validation-NIP": "NIP number must contains 10 digits",
"registerForm-SuccessfulRegistrationNotification-Title": "Account has been created",
"registerForm-SuccessfulRegistrationNotification-RenterInfo": "Verification mail has been sent to the given email address.",
"registerForm-SuccessfulRegistrationNotification-ProviderInfo": "Your account is now waiting for administration approval.",
"registerForm-RequestError-LoginTaken": "Given login is already taken",
"registerForm-RequestError-EmailTaken": "Given email is already taken",
"registerForm-RequestError-NIPTaken": "Company with given NIP number is already registered in our service",
"registerForm-RequestError-UnexpectedError": "Unexpected error occurred. Please try again",
```

Poniższe dwa zrzuty ekranu ukazują, wyświetlanie wiadomości przypisanej do danego klucza kolejno dla błędów otrzymanych z logiki aplikacji oraz walidacji danych w formularzu rejestracyjnym

```
export const registerServiceProvider = async (provider: IRegisterData) =>
  axios
    .post("mok/create/register-provider", provider)
    .then((res: AxiosResponse) => {
      // return res.statusText;
      return "Created";
    })
    .catch((err: AxiosError) => {
      if (err.response?.data.includes("email.taken.exception")) {
        return i18next.t("registerForm-RequestError-EmailTaken");
      }
      if (err.response?.data.includes("login.taken.exception")) {
        return i18next.t("registerForm-RequestError-LoginTaken");
      }
      if (err.response?.data.includes("nip.taken.exception")) {
        return i18next.t("registerForm-RequestError-NIPTaken");
      }
      return i18next.t("registerForm-RequestError-UnexpectedError");
    });
}
```

```

const registrationForm = useForm( {initialValues, initialErrors, validate: rules, schema}: {
  initialValues: {
    login: "",
    password: "",
    repeatedPassword: "",
    name: "",
    surname: "",
    email: "",
    phoneNumber: "",
    renterName: "",
    providerName: "",
    nip: "",
    address: ""
  },
  validate: {
    password: (value) => passwordValidation(value),
    repeatedPassword: (value, values) =>
      value === values.password
        ? null
        : t("registerForm-Validation-RepeatedPassword"),
    email: (value) =>
      /^[\S+@\S+$/.test(value) ? null : t("registerForm-Validation-Email"),
    phoneNumber: (value) =>
      /^[\d{9}$/.test(value) ? null : t("registerForm-Validation-PhoneNumber"),
    nip: (value) =>
      accountType === "renter" || /^[\d{10}$/.test(value)
        ? null
        : t("registerForm-Validation-NIP")
  }
});

```

Dla wyświetlanego poziomu dostępu działa taki sam mechanizm na przykład dla paska nawigacji w zależności od poziomu dostępu wyświetlane są komunikaty o różnych kluczach

```
{activeView === "Admin" ? (
  <Button
    component={Link}
    to="admin"
    variant="filled"
    color="green"
    radius="md"
    size="md"
  >
    {t("layout-Navbar-AdminPanel")}
  </Button>
) : null}

{activeView === "ServiceProvider" ? (
  <Button
    component={Link}
    to="myoffers"
    variant="filled"
    color="green"
    radius="md"
    size="md"
  >
    {t("layout-Navbar-MyOffers")}
  </Button>
) : null}

{activeView === "Renter" ? (
  <Button
    variant="filled"
    color="green"
    radius="md"
    size="md"
  >
    {t("layout-Navbar-MyRents")}
  </Button>
```

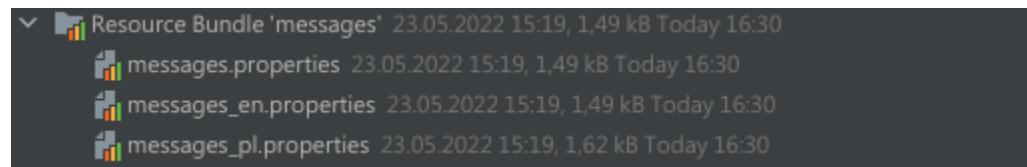
```
"Layout-Navbar-Register": "Register",
"Layout-Navbar-MainView": "Main page",
"Layout-Navbar-Offers": "Offers",
"Layout-Navbar-Contact": "Contact",
"Layout-Navbar-AdminPanel": "Admin Panel",
"Layout-Navbar-MyOffers": "My offers",
"Layout-Navbar-MyRents": "My rent",
"Layout-Navbar-Logout": "Logout",
"Layout-Navbar-Login": "Login",
"Layout-Navbar-MyAccount": "My account",
```

Natomiast mechanizm internacjonalizacji wysyłanych mail-i jest bardzo podobny jednakże podczas tego procesu język przesyłany jest w formie parametru language podczas żądania HTTP przychodzącego z warstwy prezentacji. W tym przypadku tekst również pobierany jest ze słowników.

W zależności od przysłanego parametru powstaje mail w danej wersji językowej , według szablonu oraz z odpowiednimi komponentami takimi jak na przykład przycisk.

Słowniki po stronie aplikacji logiki umieszczone są w pakiecie zasobów umieszczonym w folderze resources

Oprócz dwóch wersji językowych słowników został stworzony 3 słownik tzw. słownik domyślny który użyty zostałby w przypadku gdy kod językowy otrzymany w parametrze z warstwy prezentacji nie byłby poprawny.



```
@POST
@PermitAll
@Path("/register-renter")
@Consumes(MediaType.APPLICATION_JSON)
public Response registerRenterUser(@Valid PostRenterDTO postRenterDTO, @QueryParam("language") String language) {
    int retryTXCounter = 3;
    boolean rollbackTX;
    do {
        try {
            createAccountService.createRenterAccountAndSendVerificationEmail(
                PostMokMapper.convertToAccount(postRenterDTO),
                PostMokMapper.convertToRenterDetails(postRenterDTO)
            );
            rollbackTX = createAccountService.isLastTransactionRollback();
            if (rollbackTX) LOGGER.info( msg: "*** Transaction rollback");
        } catch (ApplicationBaseException exc) {
            throw exc;
        } catch (EJBAccessException | AccessLocalException exc) {
            throw ApplicationBaseException.getAccessDeniedException(exc);
        } catch (Exception exc) {
            throw ApplicationBaseException.getGeneralErrorException(exc);
        }
    } while (rollbackTX && --retryTXCounter > 0);

    if (rollbackTX) {
        throw ApplicationBaseException.getGeneralErrorException();
    }

    emailSender.sendVerificationEmail(postRenterDTO.getEmail(), language);
}

return Response
    .status(Response.Status.CREATED)
    .build();
}
```

```
public void sendEmail(String to, String token, MailTypeEnum mailType, String language) {  
  
    Properties props = new Properties();  
    props.put("mail.smtp.auth", "true");  
    props.put("mail.smtp.starttls.enable", "true");  
    props.put("mail.smtp.host", "smtp.gmail.com");  
    props.put("mail.smtp.port", "587");  
  
    Session session = Session.getInstance(props,  
        getPasswordAuthentication() -> {  
            return new PasswordAuthentication(SENDER_EMAIL, SENDER_PASSWORD);  
        } );  
  
    ResourceBundle rb;  
  
    if (language.equals("pl")) {  
        Locale.setDefault(new Locale(language: "pl"));  
        rb = ResourceBundle.getBundle(baseName: "messages", Locale.forLanguageTag("pl-PL"));  
    } else {  
        ;  
        Locale.setDefault(new Locale(language: "en"));  
        rb = ResourceBundle.getBundle(baseName: "messages", Locale.forLanguageTag("en"));  
    }  
}
```

20 Wykaz akcji dostępnych z interfejsu użytkownika 4 pkt

| Lp. | Oznaczenie przypadku. | Opis przypadku użycia. | Role mające prawo do wykonania akcji. | Sekwencja nazw wywoływanych metod oraz obiektów. |
|-----|-----------------------|--|---------------------------------------|--|
| 1. | MOK.1 | Rejestracja użytkownika wynajmującego usługi. | Guest | @RequestScoped CreateAccountController.registerRenterUser() → @Stateful CreateAccountService.createRenterAccountAndSendVerificationEmail() → @ApplicationScoped SHA256HashGenerator.generateHash() → @Stateless AccountFacade.create() → @Stateless VerificationTokenFacade.create() → @Stateful MOKEmailSender.sendVerificationEmail() → @Stateless VerificationTokenFacade.findByAccountEmail() → @ApplicationScoped EmailService.sendEmail() |
| 2. | MOK.1 | Rejestracja użytkownika zapewniającego usługi. | Guest | @RequestScoped CreateAccountController.registerServiceProviderUser() → @Stateful CreateAccountService.createServiceProviderAccount() → @ApplicationScoped SHA256HashGenerator.generateHash() → @Stateless AccountFacade.create() |
| 3. | MOK.2 | Utworzenie konta administratora przez innego administratora. | Admin | @RequestScoped CreateAccountController, metoda createAdmin() → @Stateful CreateAccountService.createAdminAccount() → @ApplicationScoped SHA256HashGenerator.generateHash() → @Stateless AccountFacade.create() |
| 4. | MOK.3 | Zablokowanie konta. | Admin | @RequestScoped UpdateAccountController.changeActive() → @Stateful UpdateAccountService.deactivateUserAccount() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() @Stateful MOKEmailSender.sendDactivateEmail() → @Stateless VerificationTokenFacade.findByLogin() → @ApplicationScoped EmailService.sendEmail() |
| 5. | MOK.4 | Odblokowanie konta. | Admin | @RequestScoped UpdateAccountController.changeActive() → @Stateful UpdateAccountService.activateUserAccount() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() @Stateful MOKEmailSender.sendActivateEmail() → @Stateless VerificationTokenFacade.findByLogin() → @ApplicationScoped EmailService.sendEmail() |
| 6. | MOK.5 | Doliczenie poziomu dostępu do konta. | Admin | @RequestScoped UpdateAccountController.deleteAccessLevel() → @Stateful UpdateAccountService.deleteAccessLevel() → @Stateless AccountFacade.findByLoginAccessLevel() → @Stateless AccessLevelFacade.remove() |

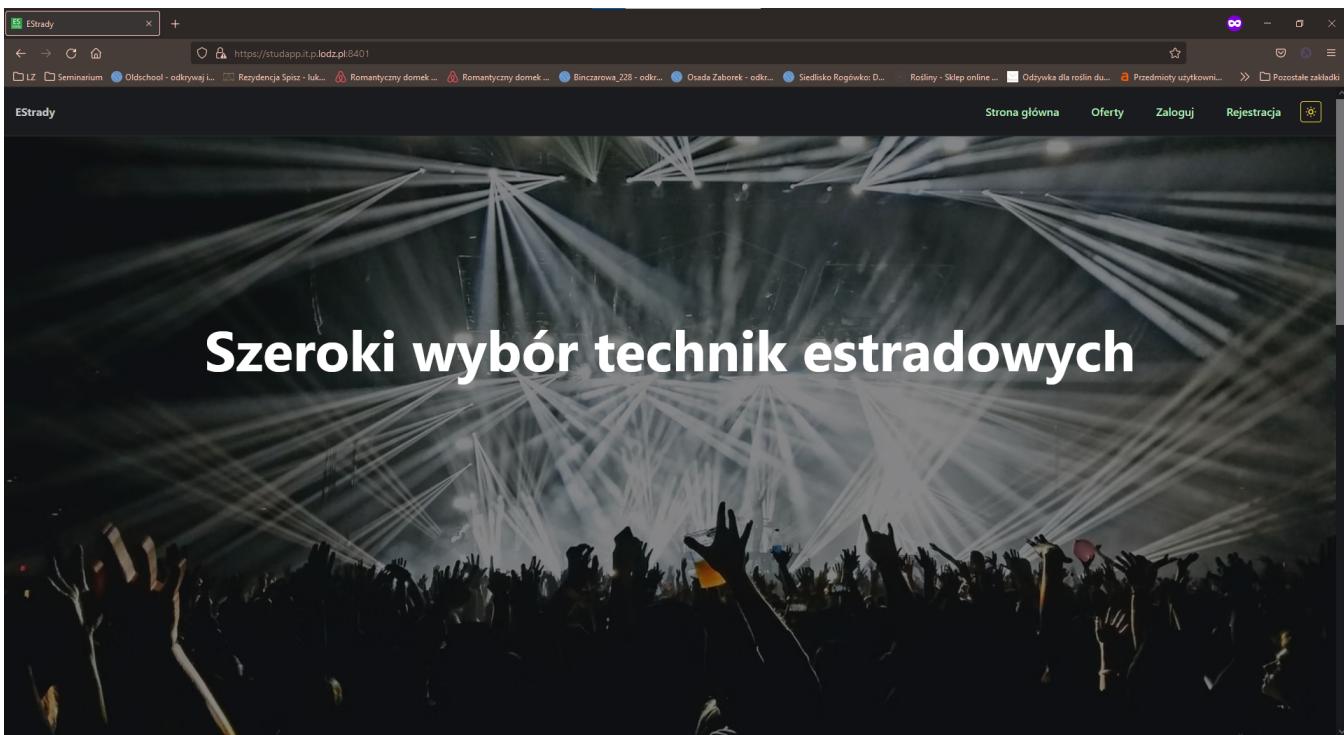
| | | | | |
|-----|--------|---|------------------------------------|--|
| 7. | MOK.6 | Odlączanie poziomu dostępu do konta. | Admin | @RequestScoped UpdateAccountController.deleteAccessLevel() → @Stateful UpdateAccountService.addAccessLevel() → @Stateless AccountFacade.findByLoginAccessLevel() → @Stateless AccessLevelFacade.edit() |
| 8. | MOK.7 | Zmiana własnego hasła. | Admin | @RequestScoped UpdateAccountController.changeUserOwnPassword() → |
| | | | ServiceProvider | @Stateful UpdateAccountService.changeOwnPassword() → |
| | | | Renter | @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() |
| 9. | MOK.8 | Zmiana hasła przez administratora. | Admin | @RequestScoped UpdateAccountController.changeUserPassword() → @Stateful UpdateAccountService.changePassword() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() |
| 10. | MOK.9 | Edycja danych własnego konta. | Admin ServiceProvider Renter | @RequestScoped UpdateAccountController.editAdmin() / UpdateAccountController.editServiceProvider() / UpdateAccountController.editRenter() → @Stateful UpdateAccountService.updateAdmin() / UpdateAccountService.updateServiceProvider() / UpdateAccountService.updateRenter() → komponent EJB @Stateless AccountFacade.findByLogin() → komponent EJB @Stateless AccountFacade.edit() → komponent EJB @Stateless AccessLevelFacade.edit() |
| 11. | MOK.10 | Edycja danych konta innego użytkownika. | Admin | @RequestScoped UpdateAccountController.editAdmin() / UpdateAccountController.editServiceProvider() / UpdateAccountController.editRenter() → @Stateful UpdateAccountService.updateAdmin() / UpdateAccountService.updateServiceProvider() / UpdateAccountService.updateRenter() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() → @Stateless AccessLevelFacade.edit() |
| 12. | MOK.11 | Wylogowanie się użytkownika. | Admin ServiceProvider Renter | Wylogowanie nie odbywa się po stronie warstwy logiki biznesowej, ale po stronie warstwy prezentacji i polega na usunięciu danych z lokalnego obiektu "storage" w przeglądarce internetowej użytkownika. const logout = () => { localStorage.removeItem("authToken"); localStorage.removeItem("refreshToken"); delete axios.defaults.headers.common.Authorization; }; |
| 13. | MOK.12 | Aktualizacja profilu firmy. | ServiceProvider | @RequestScoped UpdateAccountController.editServiceProvider() → @Stateful UpdateAccountService.updateServiceProvider() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() → @Stateless AccessLevelFacade.edit() |

| | | | | |
|-----|--------|---|-----------------|---|
| 14. | MOK.13 | Firma zostaje zatwierdzona przez administratora. | Admin | <pre>@RequestScoped UpdateAccountController.confirmServiceProvider() → @Stateful UpdateAccountService.confirmServiceProvider() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.edit() → @Stateful MOKEmailSender.sendConfirmServiceProviderEmail() → @Stateless AccountFacade.findByLogin() → @ApplicationScoped EmailService.sendEmail()</pre> |
| 15. | MOK.13 | Firma zostaje odrzucona przez administratora. | Admin | <pre>@RequestScoped DeleteAccountController.deleteServiceProvider() → @Stateful ReadAccountService.readAccountByLogin() → @Stateful UpdateAccountService.deleteServiceProvider() → @Stateless AccountFacade.findByLogin() → @Stateless AccountFacade.remove() → @Stateful MOKEmailSender.sendRemoveEmail() → @ApplicationScoped EmailService.sendEmail()</pre> |
| 16. | MOK.14 | Zalogowanie się użytkownika do serwisu. | Guest | <pre>@RequestScoped AuthController.authenticate() → @Stateful AuthService.generateTokens() → @ApplicationScoped DatabaseIdentityStore.validate() → @Stateful AuthService.authenticateUsernamePassword() → @Stateless AccountAuthFacade.findByLoginActiveConfirmed() → @ApplicationScoped SHA256HashGenerator.generateHash() → @Stateful AuthService.getCallerGroups()</pre> |
| 17. | MOK.15 | Resetowania hasła, krok 1 - wysłanie maila z linkiem do zmiany hasła na podany mail. | Guest | <pre>@RequestScoped UpdateAccountController.sendResetPasswordEmail() → @Stateful UpdateAccountService.sendPasswordResetEmail(l) → @Stateless AccountFacade.findByEmail() → @Stateless VerificationTokenFacade.create()</pre> |
| 18. | MOK.15 | Resetowania hasła, krok 2 - sprawdzenie czy token do zmiany hasła wygasł, przeniesienie na stronę z formularzem do zmiany hasła lub przeniesienie na stronę informującą, że token wygasł. | Guest | <pre>@RequestScoped UpdateAccountController. confirmResetPasswordEmail() → @Stateful UpdateAccountService.sendPasswordResetEmail() → @Stateless VerificationTokenFacade.findByToken()</pre> |
| 19. | MOK.15 | Resetowania hasła, krok 3 - reset hasła z wykorzystaniem wyżej wymienionego tokenu. | Guest | <pre>@RequestScoped UpdateAccountController.resetPassword() → @Stateful UpdateAccountService.resetPassword() → @Stateless VerificationTokenFacade.findByToken() → @Stateless AccountFacade.edit() → @Stateless VerificationTokenFacade.remove()</pre> |
| 20. | MZ.1 | Dodanie oferty usługi przez użytkownika. | ServiceProvider | <pre>@RequestScoped OfferController.addOffer() → @Stateful OfferService.addOffer() → @Stateless ServiceProviderDetailsFacade.findByAccountLogin() → @Stateless OfferFacade.create()</pre> |

| | | | | |
|------------|----------|--|---|---|
| 21. | MZ.2 | Usunięcie wybranej oferty (ustawienie jako niedostępnej do wynajęcia). | Admin ServiceProvider | @RequestScoped OfferController.deleteOfferAdmin() / OfferController.deleteOfferServiceProvider() → @Stateful OfferService.deleteOffer() / OfferService.deleteOwnOffer() → @Stateless OfferFacade.getId() → @Stateless OfferFacade.edit() |
| 22. | MZ.3 | Edycja wybranej oferty. | Admin ServiceProvider | @RequestScoped OfferController.editOfferAdmin() / OfferController.editOfferServiceProvider() → @Stateful OfferService.editOfferAdmin() / OfferService.editOffer() → @Stateless OfferFacade.getId() → @Stateless OfferFacade.edit() |
| 23. | MO.1 | Przeglądanie profili firm wynajmujących usługi. | Guest Admin ServiceProvider Renter | @RequestScoped BrowseOfferController.getAllServiceProviders() → @Stateful BrowseOfferService.getAllServiceProviders() → @Stateless AccountFacade.getAllServiceProviders() |
| 24. | MO.2 | Przeglądanie aktualnych ofert firm wynajmujących usługi. | Guest Admin ServiceProvider Renter | @RequestScoped BrowseOfferController.getAllActiveOffers() → @Stateful BrowseOfferService.getAllActiveOffers() → @Stateless OfferFacade.getAllActiveOffers() |
| 25. | MO.3 | Filtrowanie przeglądanych ofert poprzez podanie ceny minimalnej i maksymalnej. | Guest Admin ServiceProvider Renter | @RequestScoped BrowseOfferController.getOffersFilteredByPrice() → @Stateful BrowseOfferService.getOffersFilteredByPrice() → @Stateless OfferFacade.findOffersFilteredByPrice() |
| 26. | MW.1 | Wynajęcie konkretnej oferty firmy wynajmującej usługi. | Renter | @RequestScoped HiringController.hireOffer() → @Stateful HiringService.hireOffer() → @Stateless OfferDateFacade.findByDateAndOfferId() → @Stateless OfferFacade.find() → @Stateless AccountFacade.findByLogin() → @Stateless RenterDetailsFacade.findByIdAccountId() → @Stateless OfferFacade.find() → @Stateless OfferDateFacade.create() → @Stateless UserOfferFacade.create() |
| 27. | MW.2 | Ocena firmy wynajmującej usługi. | Renter | @RequestScoped RateController.rate() → @Stateful RateService.rate() → @Stateless RateFacade.findByServiceName() → @Stateless AccountFacade.findByLogin() → @Stateless RenterDetailsFacade.findByIdAccountId() → @Stateless RateFacade.edit() |
| 28. | SYSTEM 1 | Wysyłanie maili do weryfikacji konta użytkownika wynajmującego usługi. | - | @Stateful MOKEmailSender.sendVerificationEmail() → @Stateless VerificationTokenFacade.findByAccountEmail() → @ApplicationScoped EmailService.sendEmail() |

| | | | | |
|-----|----------|--|---|---|
| 29. | SYSTEM 2 | Usuwanie niepotwierdzonych kont użytkowników wynajmujących usługi. | - | @Singleton Scheduler.removeInactiveTokensAndUsers() → @Stateless VerificationTokenFacade.getExpiredTokens() → @Stateless VerificationTokenFacade.remove() → @Stateless AccountFacade.remove() → @Stateful MOKEmailSender.sendRemoveEmail() → @ApplicationScoped EmailService.sendEmail() |
|-----|----------|--|---|---|

1. Rejestracja użytkownika wynajmującego usługę (MOK.1)



Obraz 1.1

Zarejestruj się

Strona główna Oferty Zaloguj Rejestracja

Footer

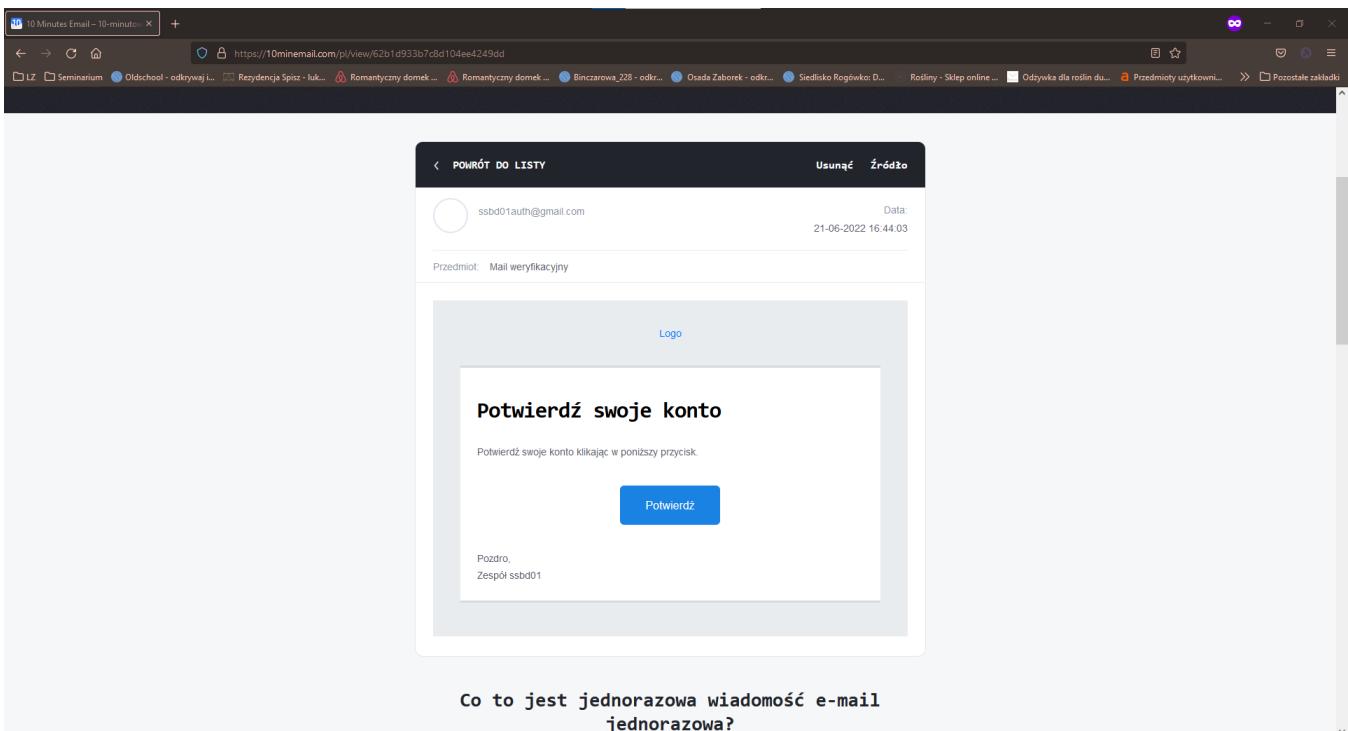
Form Fields:

- Login *: testSpraw
- Hasło *: (show/hide)
- Powtóż hasło *: (show/hide)
- Imię *: Ssbd
- Nazwisko *: Ssbd
- Email *: komos32194@runqx.com
- Numer telefonu *: 123987345
- Rodzaj konta: Klient (selected), Usługodawca
- Nazwa użytkownika: testSpraw
- reCAPTCHA: Nie jestem robotem (checked)

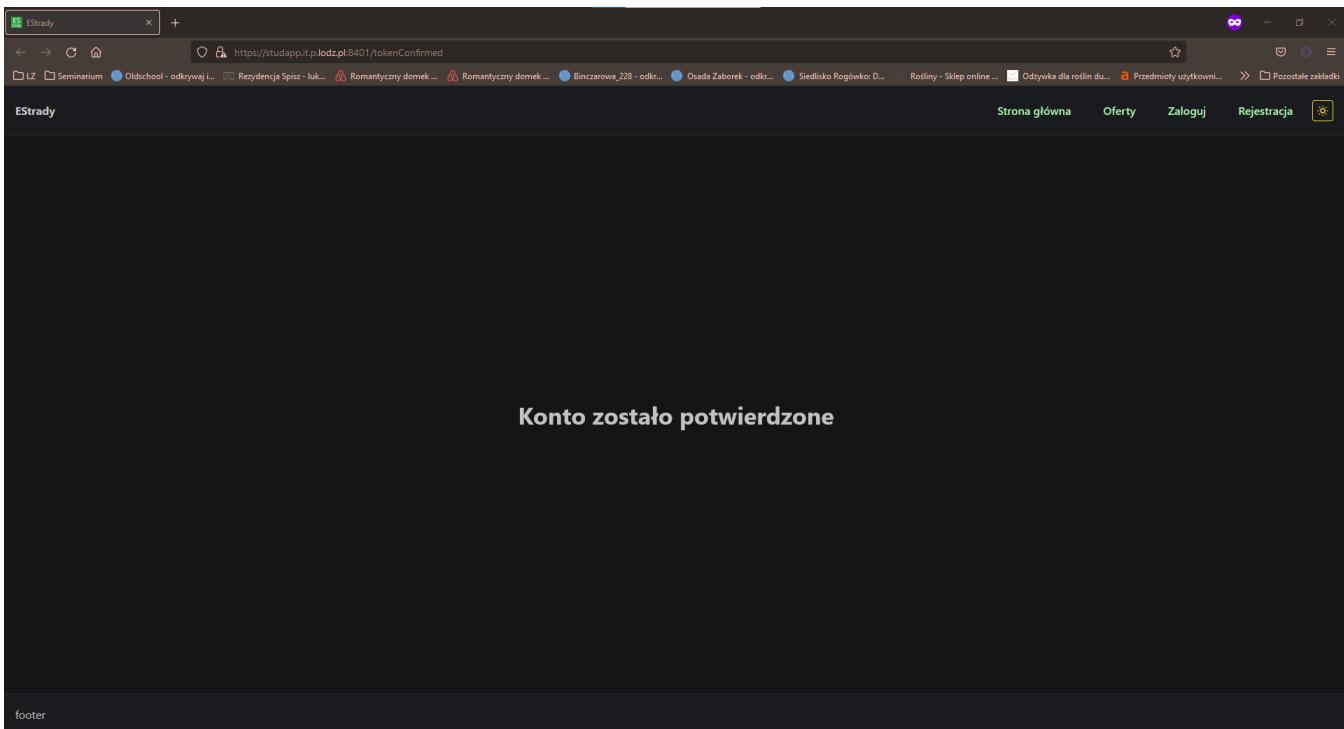
Buttons:

- Zarejestruj (Green button)

Obraz 1.2

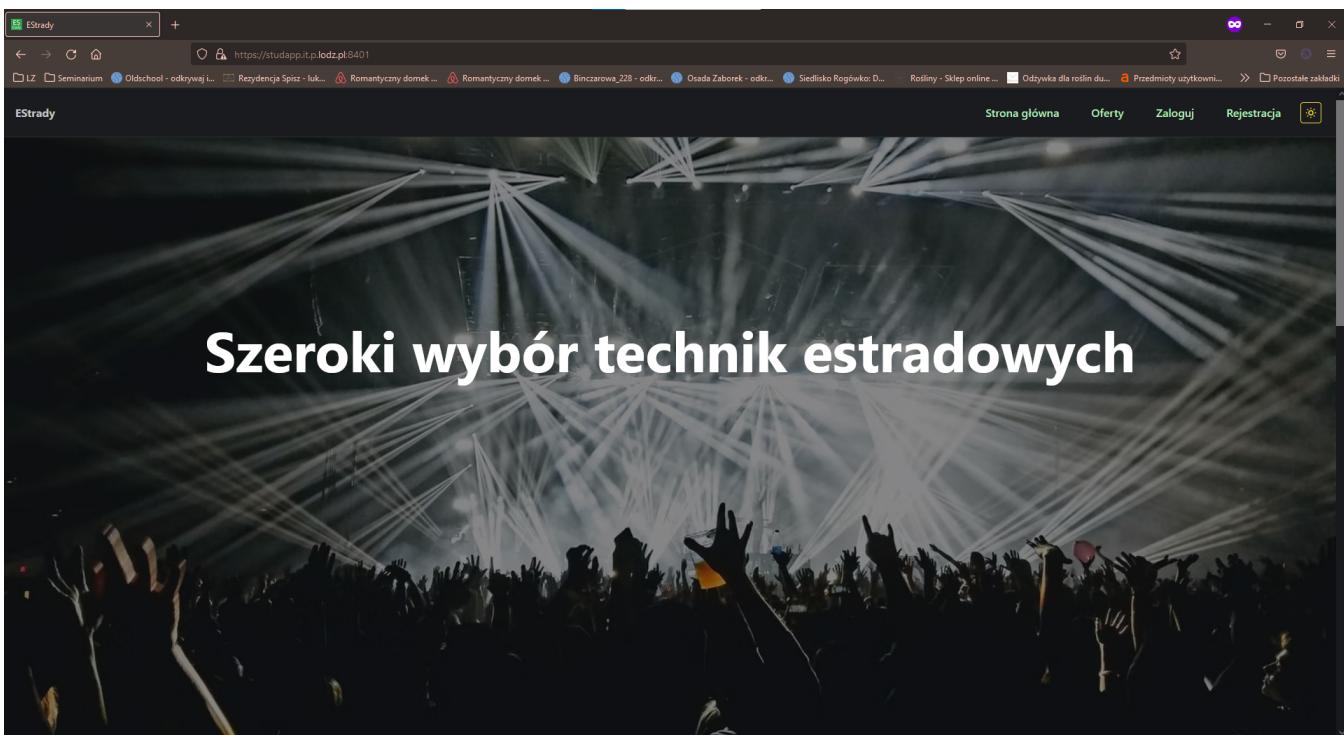


Obraz 1.3



Obraz 1.4

2. Rejestracja użytkownika zapewniającego usługi (MOK.1)



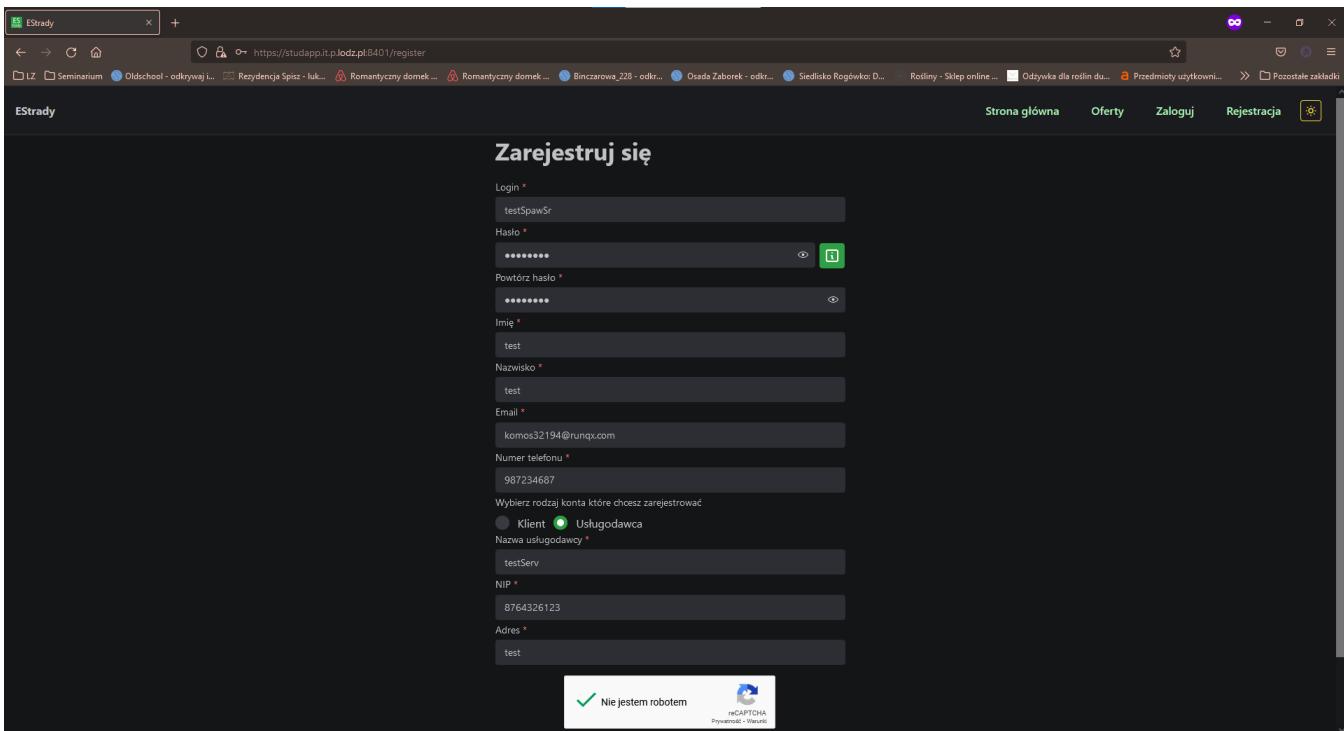
Obraz 2.1

Zarejestruj się

Strona główna Oferty Zaloguj Rejestracja

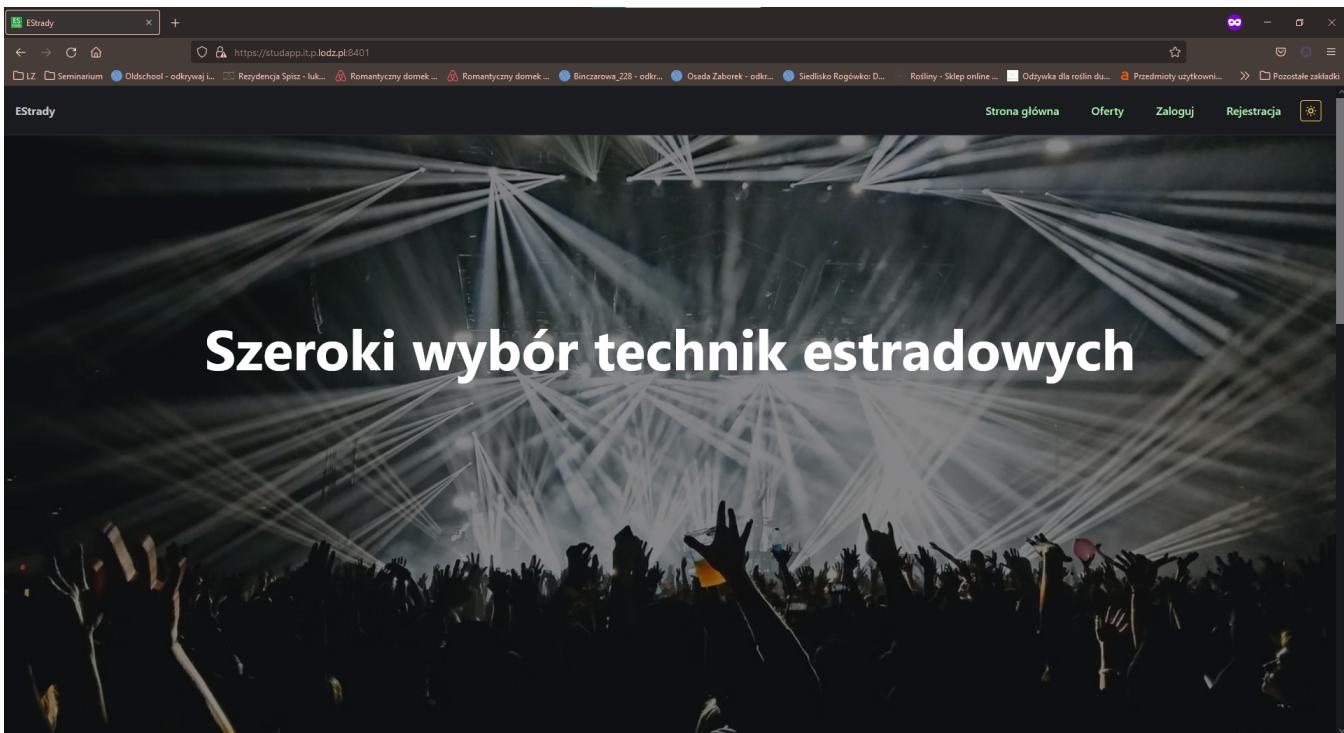
Login * testSpawSr
 Hasło * *****
 Powtóż hasło * *****
 Imię * test
 Nazwisko * test
 Email * komos32194@runqa.com
 Numer telefonu * 987234687
 Wybierz rodzaj konta które chcesz zarejestrować
 Klient Uslugodawca
 Nazwa usługodawcy * testServ
 NIP * 8764326123
 Adres * test

Nie jestem robotem 
 reCAPTCHA
 Prawo autorskie - Wszelkie

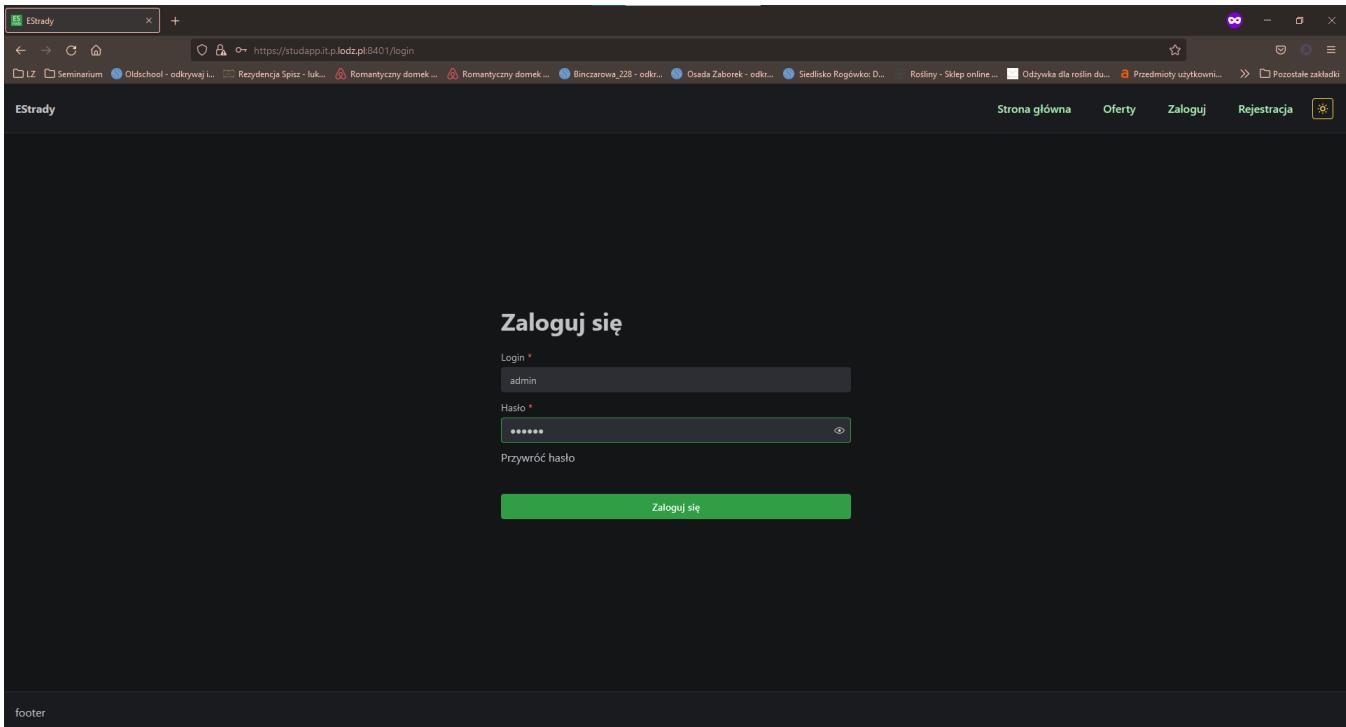


Obraz 2.2

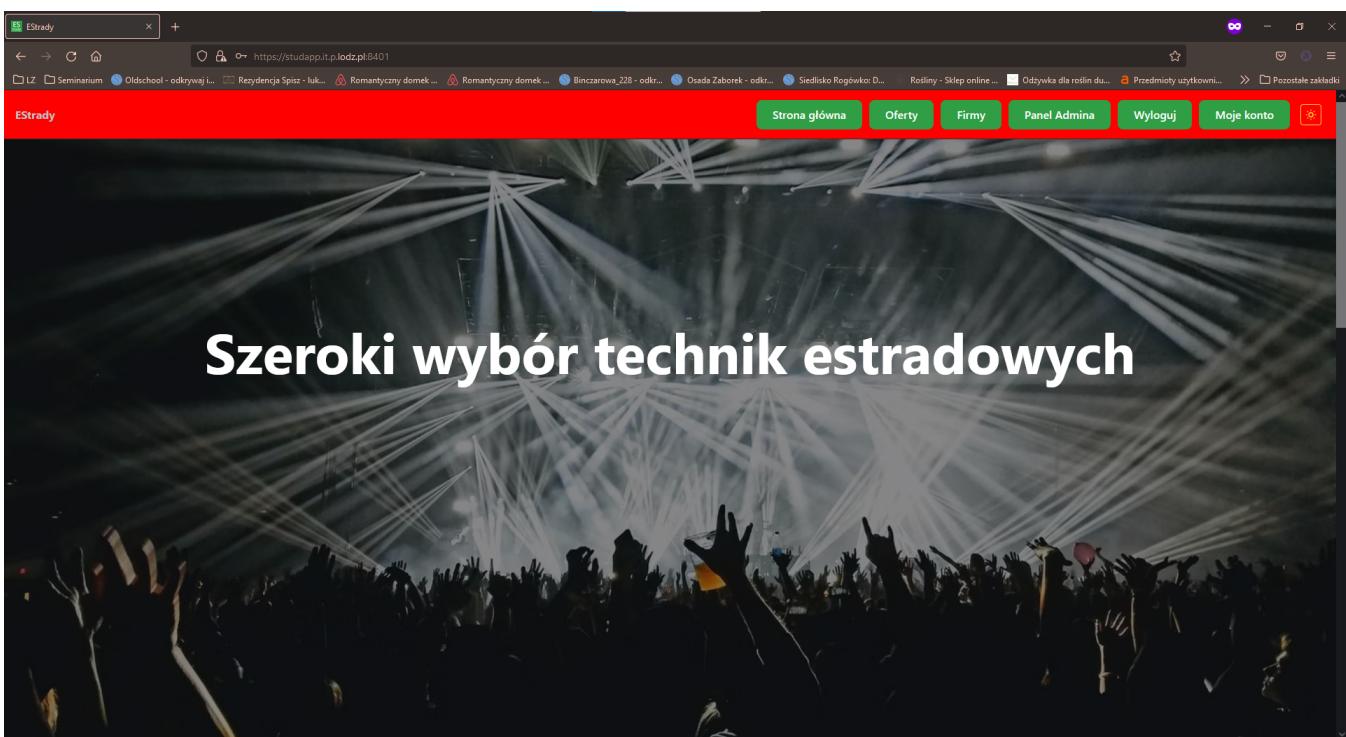
3. Utworzenie konta administratora przez innego administratora (MOK.2)



Obraz 3.1



Obraz 3.2



Obraz 3.3

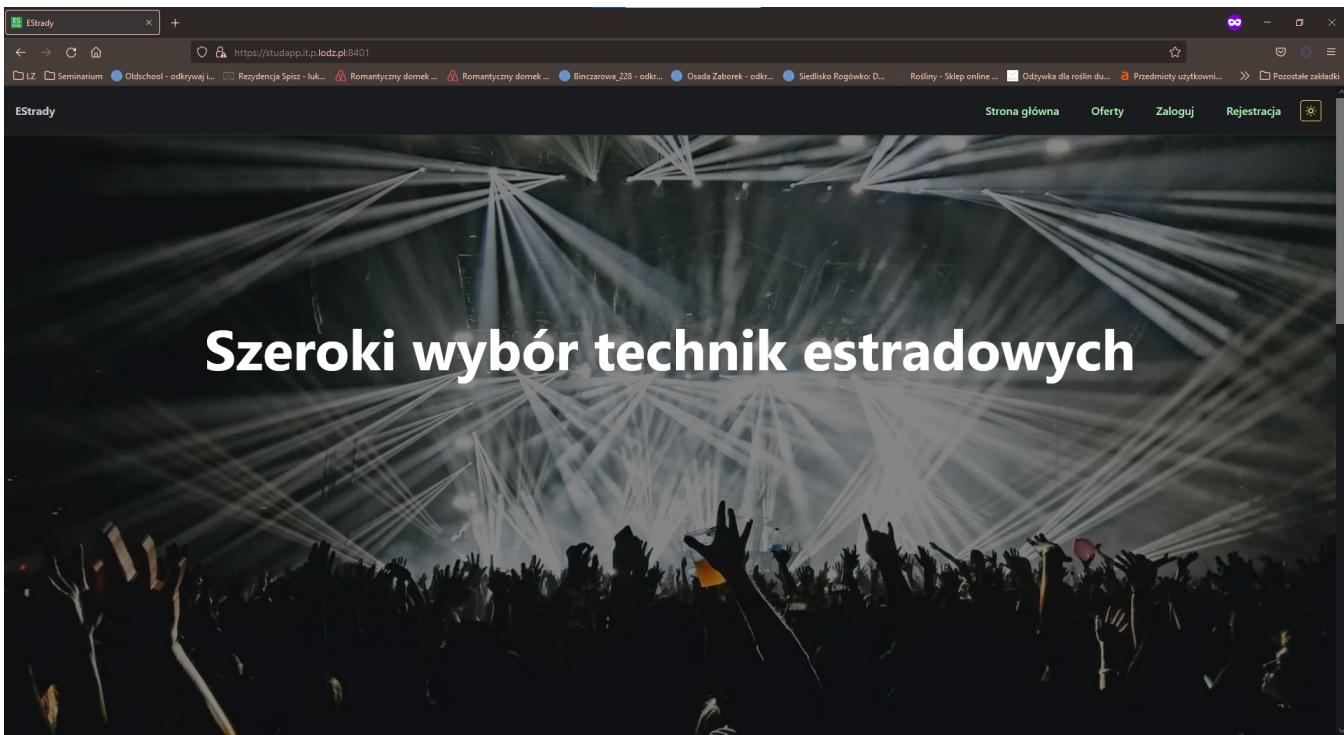
| EStrady | | | |
|--|----------|-----------------|---|
| Strona główna Oferty Firmy Panel Admina Wyloguj Moje konto ? | | | |
| Lista użytkowników Wiadomości Ustawienia Konta do potwierdzenia Dodaj poziom dostępu Stwórz konto administratora | | | |
| Znajdź użytkownika <input type="text" value="Szukaj..."/> | | | |
| Imię | Nazwisko | Login | Status |
| Ssbadmin | Ssbd | admin | Aktywne Zmień hasło |
| Ssbdrenter | Ssbd | renter | Aktywne Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter2 | Aktywne Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter3 | Aktywne Zmień hasło Modyfikuj konto |
| Ssbbserviceprovider | Ssbd | serviceProvider | Aktywne Zmień hasło Modyfikuj konto |
| Test | Test | testSpawaw | Aktywne Zmień hasło Modyfikuj konto |
| Ssbd | Ssbd | testSpraw | Aktywne Zmień hasło Modyfikuj konto |

Obraz 3.4

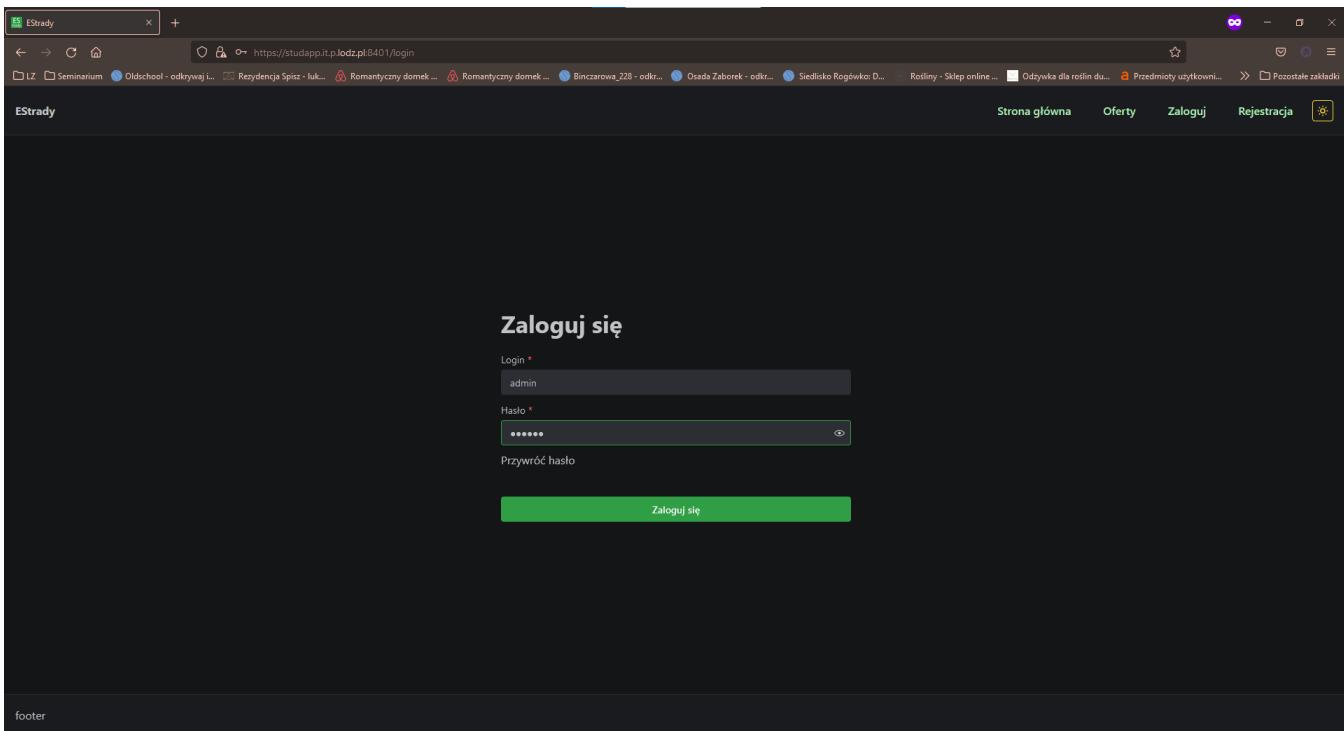
| EStrady | | | |
|--|--|--|--|
| Strona główna Oferty Firmy Panel Admina Wyloguj ? | | | |
| Lista użytkowników Wiadomości Ustawienia Konta do potwierdzenia Dodaj poziom dostępu Stwórz konto administratora | | | |
| Tworzenie konta administratora | | | |
| Login * <input type="text" value="adminSpaw"/> | | | |
| Hasło * <input type="password" value="*****"/> Zmień hasło | | | |
| Powtórz hasło * <input type="password" value="*****"/> Zmień hasło | | | |
| Imię * <input type="text" value="Test"/> | | | |
| Nazwisko * <input type="text" value="Test"/> | | | |
| Email * <input type="text" value="email@email.com"/> | | | |
| Numer telefonu * <input type="text" value="127346432"/> | | | |
| Konto zostało stworzone Konto zostało stworzone | | | |

Obraz 3.5

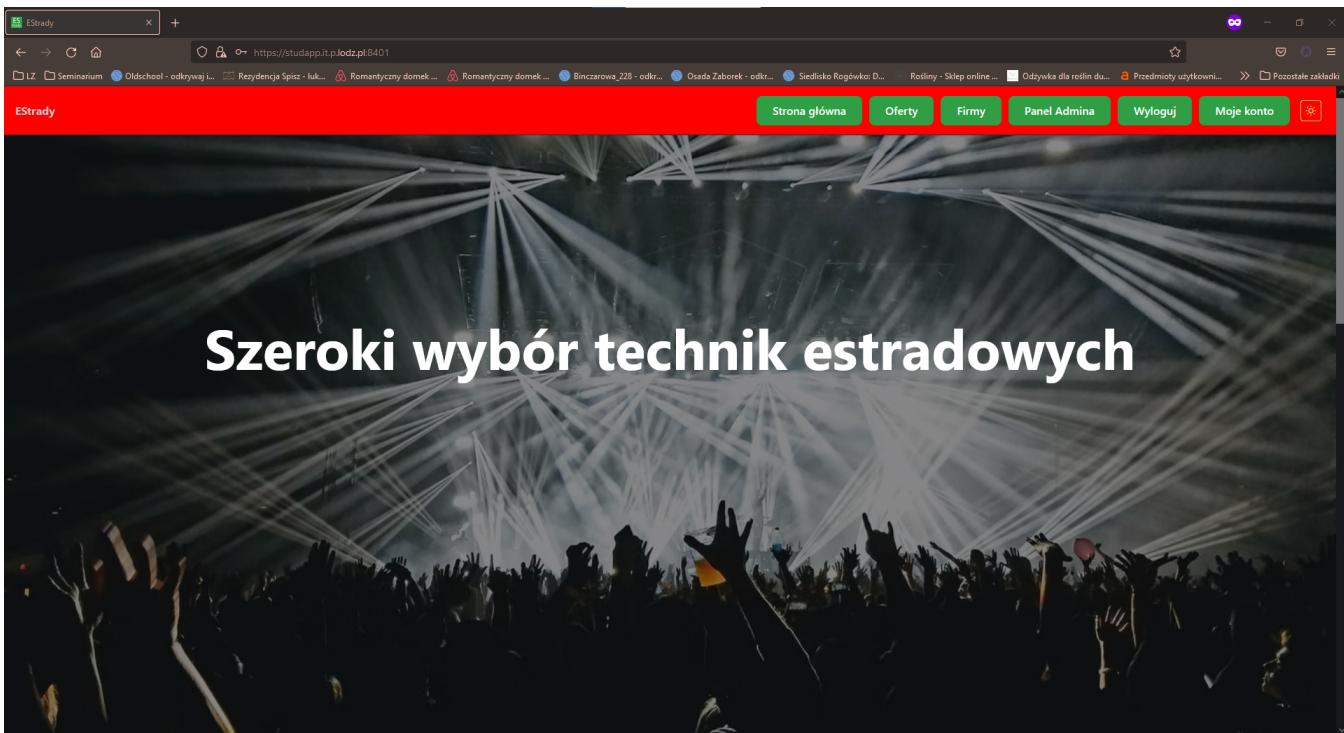
4. Zablokowanie konta (MOK.3)



Obraz 4.1



Obraz 4.2

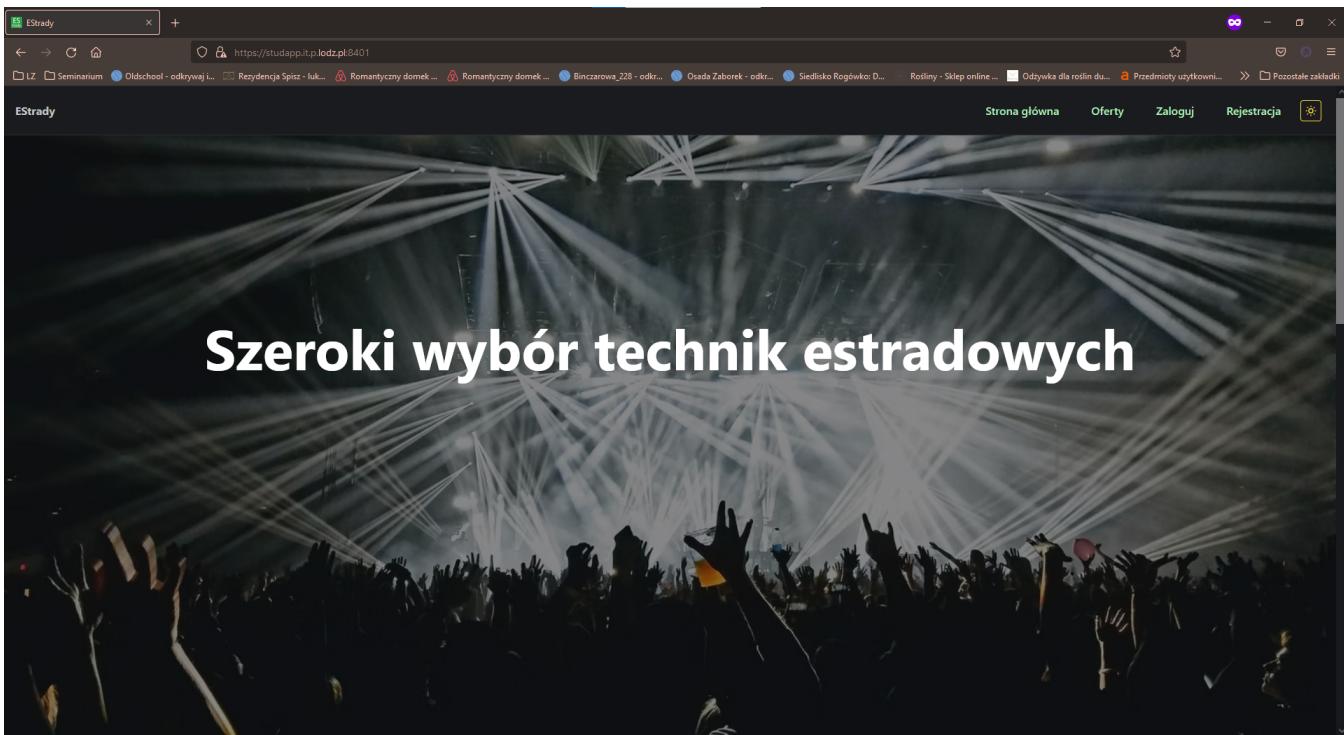


Obraz 4.3

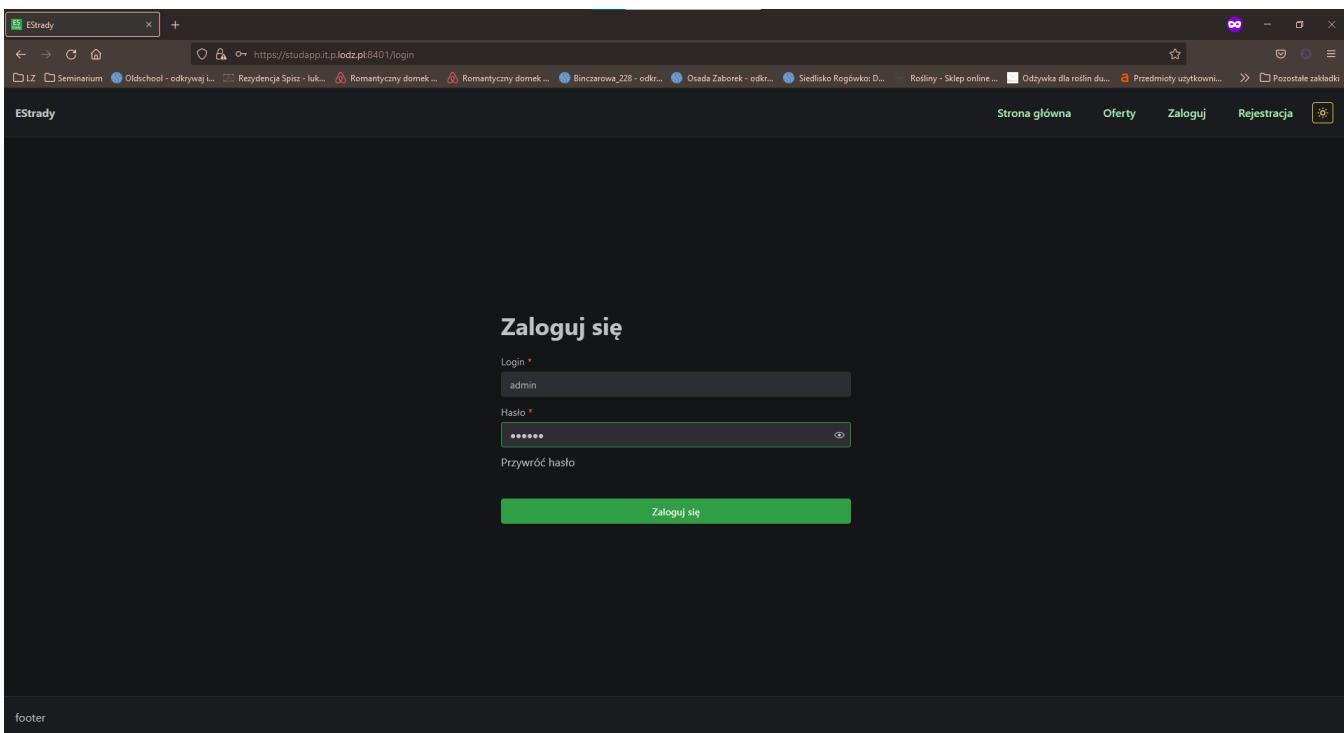
A screenshot of the 'Panel Admina' section of the Estrady website, specifically the 'Lista użytkowników' (User List) page. The page has a red header with the same navigation links as the homepage. The main content area shows a table of user accounts. Each account row includes columns for 'Imię' (Name), 'Nazwisko' (Surname), 'Login', and 'Status'. Buttons for 'Aktywne' (Active), 'Zmień hasło' (Change password), and 'Modyfikuj konto' (Edit account) are present in the status column. A search bar labeled 'Znajdź użytkownika' (Find user) is located above the table. The table lists several users, including 'Ssbdadmin', 'Test', 'Ssbdrenter', 'Ssbdrenter', 'Ssbdrenter', 'Ssbdserviceprovider', 'Test', and 'Ssbd', each with their respective details and status buttons.

Obraz 4.4

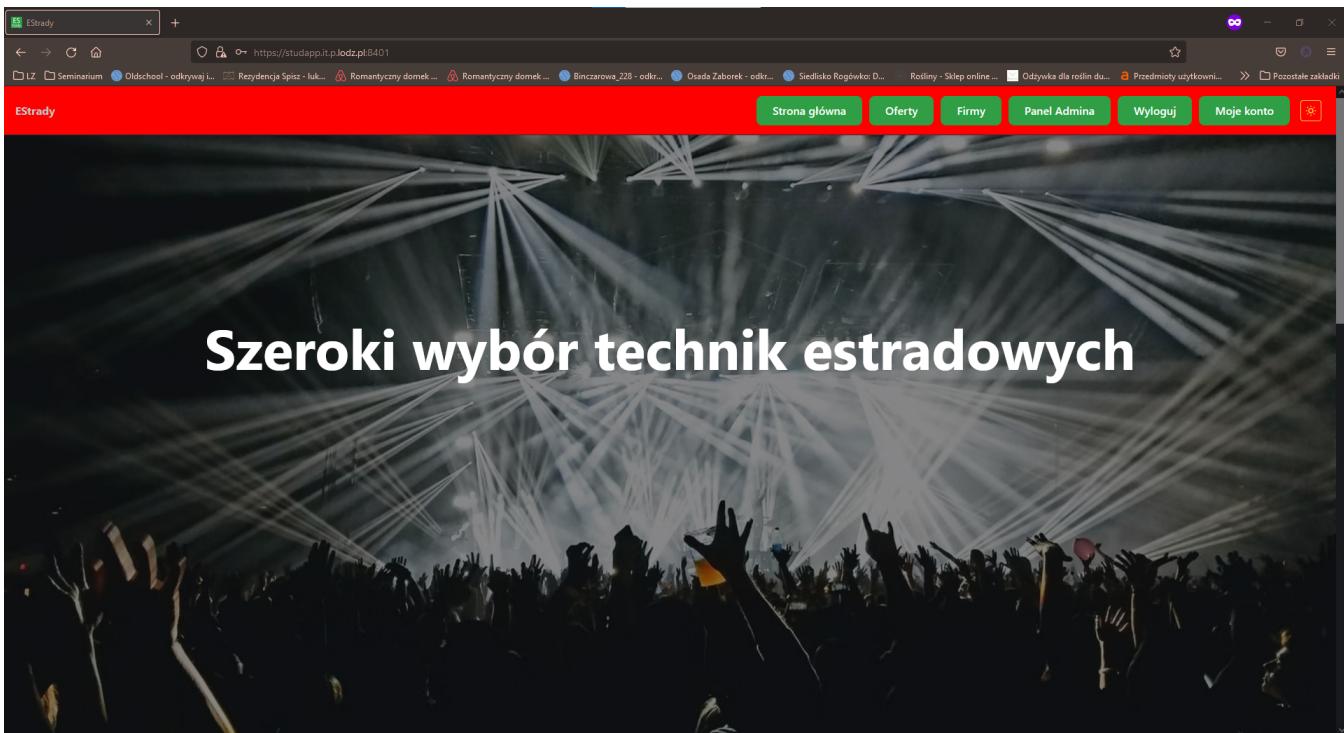
5. Odblokowanie konta (MOK.5)



Obraz 5.1



Obraz 5.2

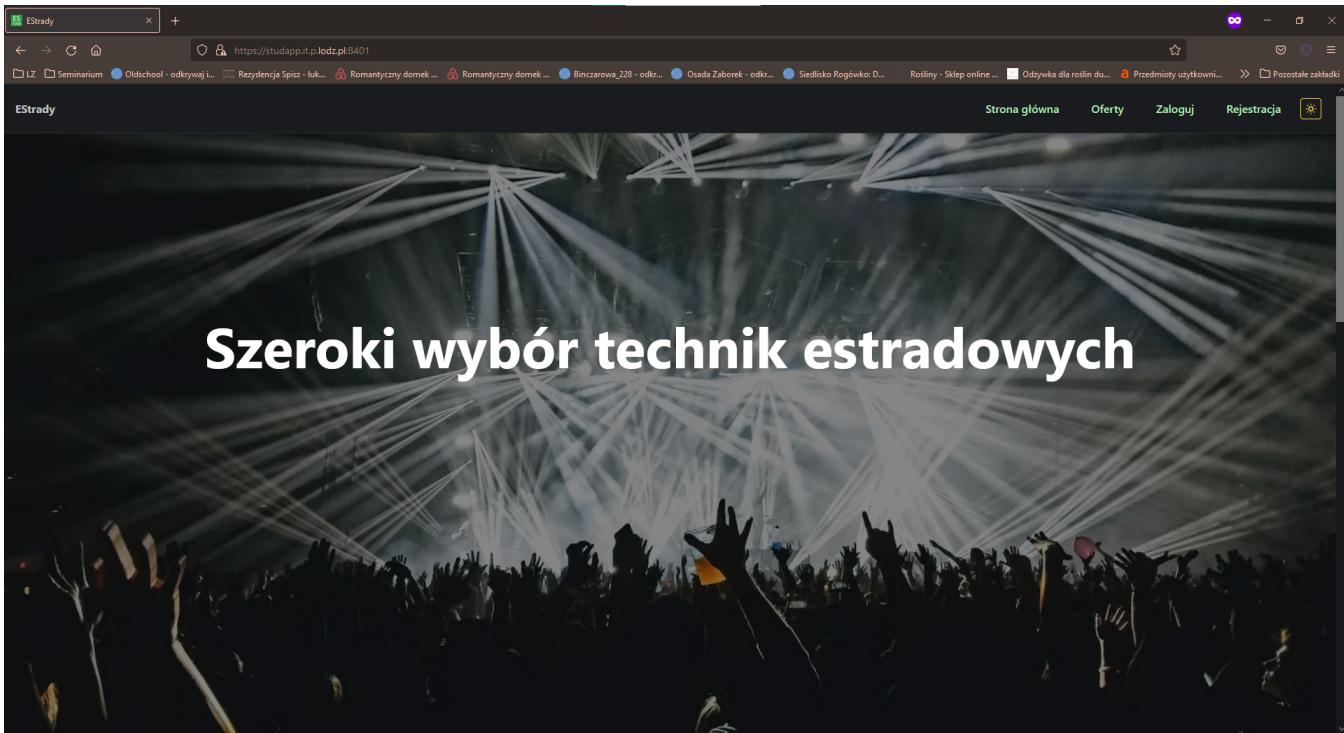


Obraz 5.3

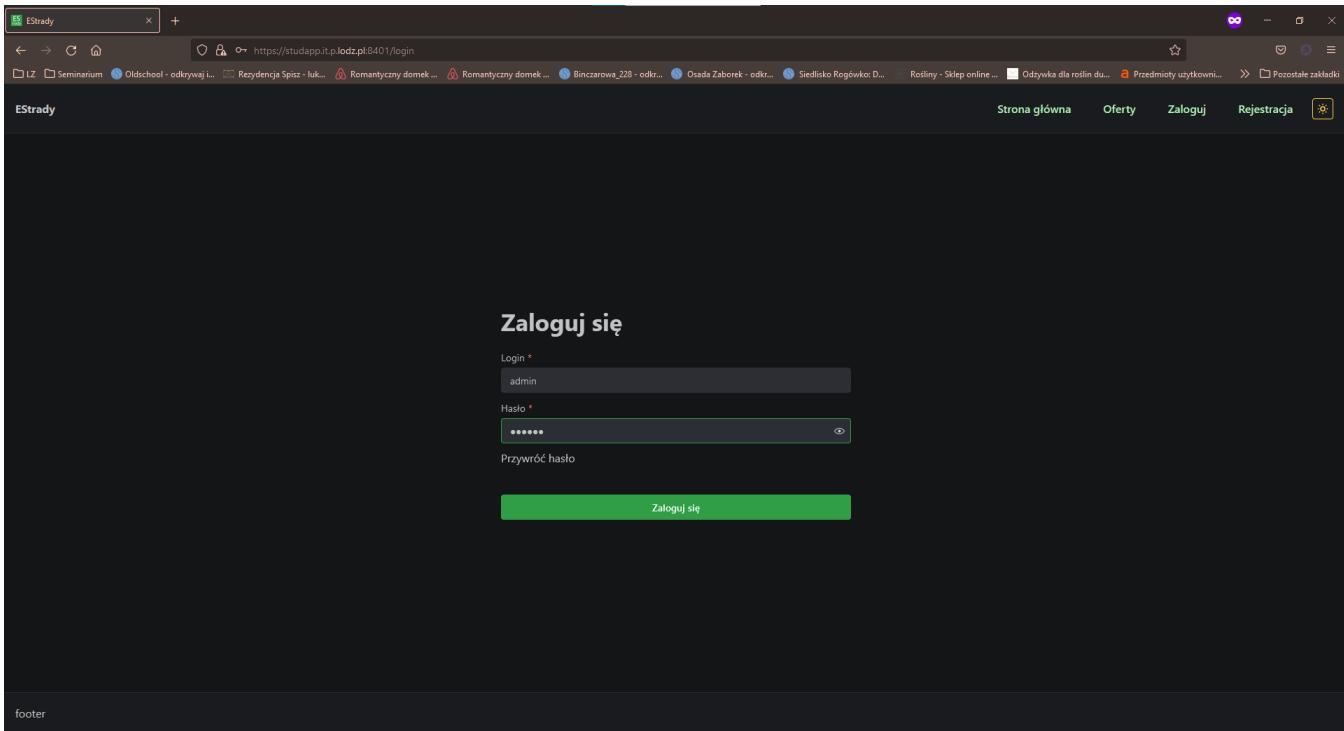
A screenshot of the Estrady website's administrative panel, specifically the user management section. The title bar says 'Estrady' and the address bar shows 'https://studapp.it.p.lodz.pl:8401/admin'. The page has a red header with the Estrady logo and navigation links: 'Strona główna', 'Oferty', 'Firmy', 'Panel Admina', 'Wyloguj', and 'Moje konto'. Below the header is a search bar labeled 'Znajdź użytkownika' with a placeholder 'Szukaj...'. A table lists users with columns: 'Imię', 'Nazwisko', 'Login', and 'Status'. Each row contains a user's name, surname, login, and status (either 'Aktywne' or 'Zmien hasło'). There are also buttons for 'Zmień hasło' and 'Modyfikuj konto'. At the bottom of the table, there are buttons for 'Dodaj poziom dostępu' and 'Stwórz konto administratora'. The footer of the page says 'Footer'.

Obraz 5.4

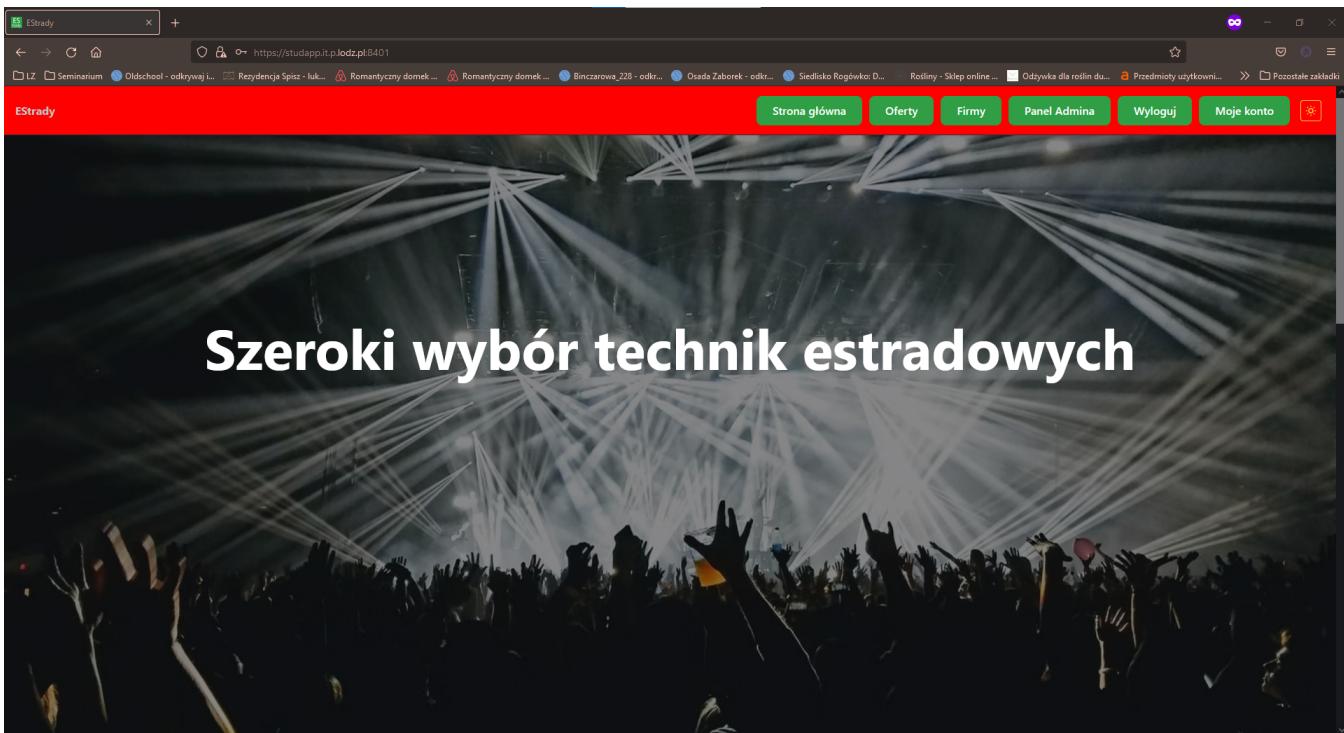
6. Dołączenie poziomu dostępu do konta (MOK.5)



Obraz 6.1



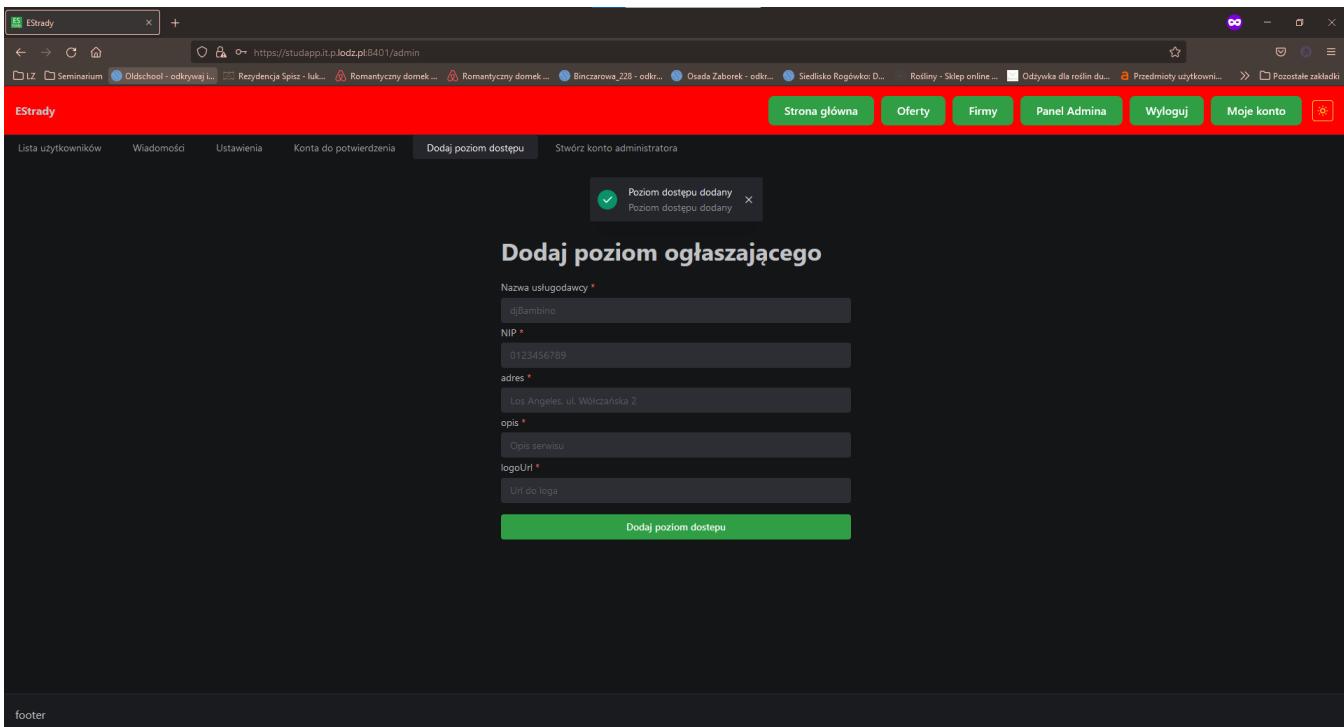
Obraz 6.2



Obraz 6.3

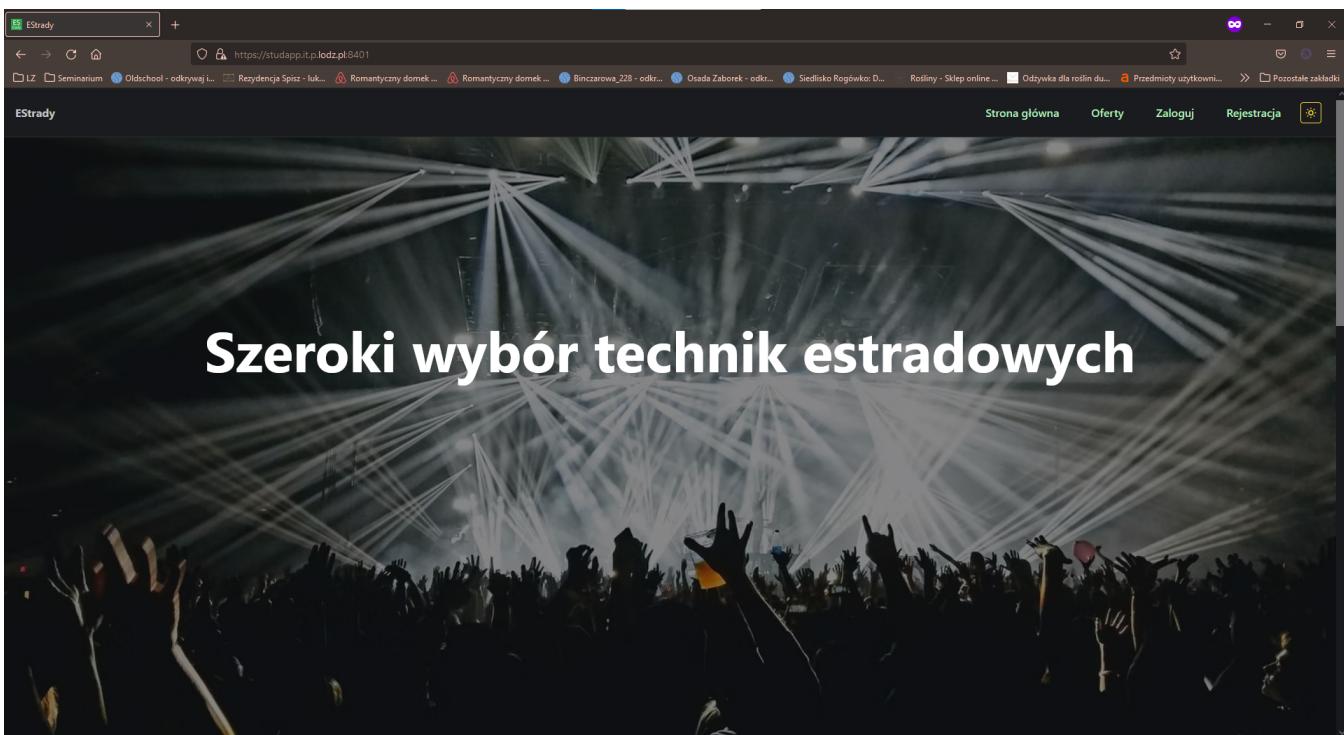
A screenshot of the Estrady website's admin panel, specifically the "Dodaj poziom wynajmującego" (Add rental level) and "Dodaj poziom ogłaszającego" (Add advertising level) sections. The top navigation bar includes links for "Lista użytkowników", "Wiadomości", "Ustawienia", "Konta do potwierdzenia", "Dodaj poziom dostępu", and "Stwórz konto administratora". The "Dodaj poziom dostępu" button is highlighted in green. The first section, "Dodaj poziom wynajmującego", has a field for "Nazwa wynajmującego" containing "djBambino" and a "Dodaj poziom dostępu" button. The second section, "Dodaj poziom ogłaszającego", contains fields for "Nazwa usługodawcy" (djBambino), "NIP" (0123456789), "adres" (Los Angeles, ul. Wielicka 2), "opis" (Opis serwisu), and "logoUrl" (Url do loga), each with its own "Dodaj poziom dostępu" button.

Obraz 6.4

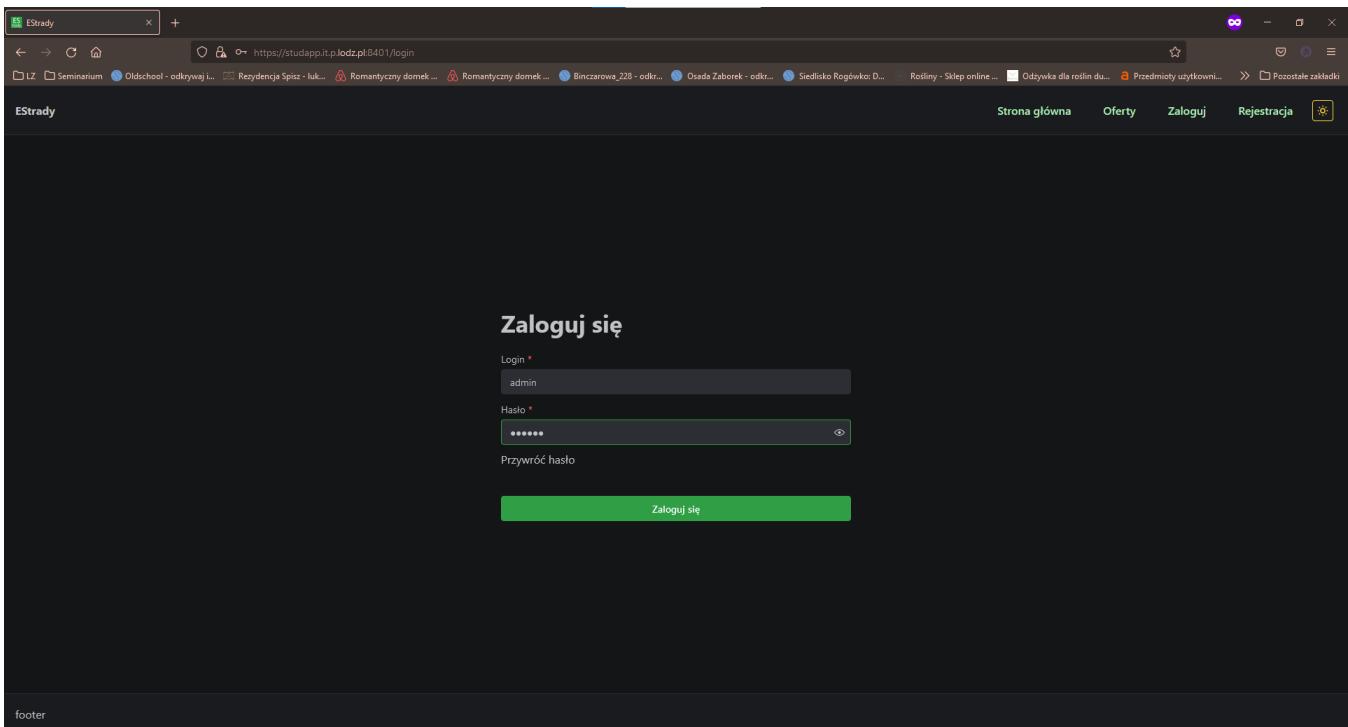


Obraz 6.5

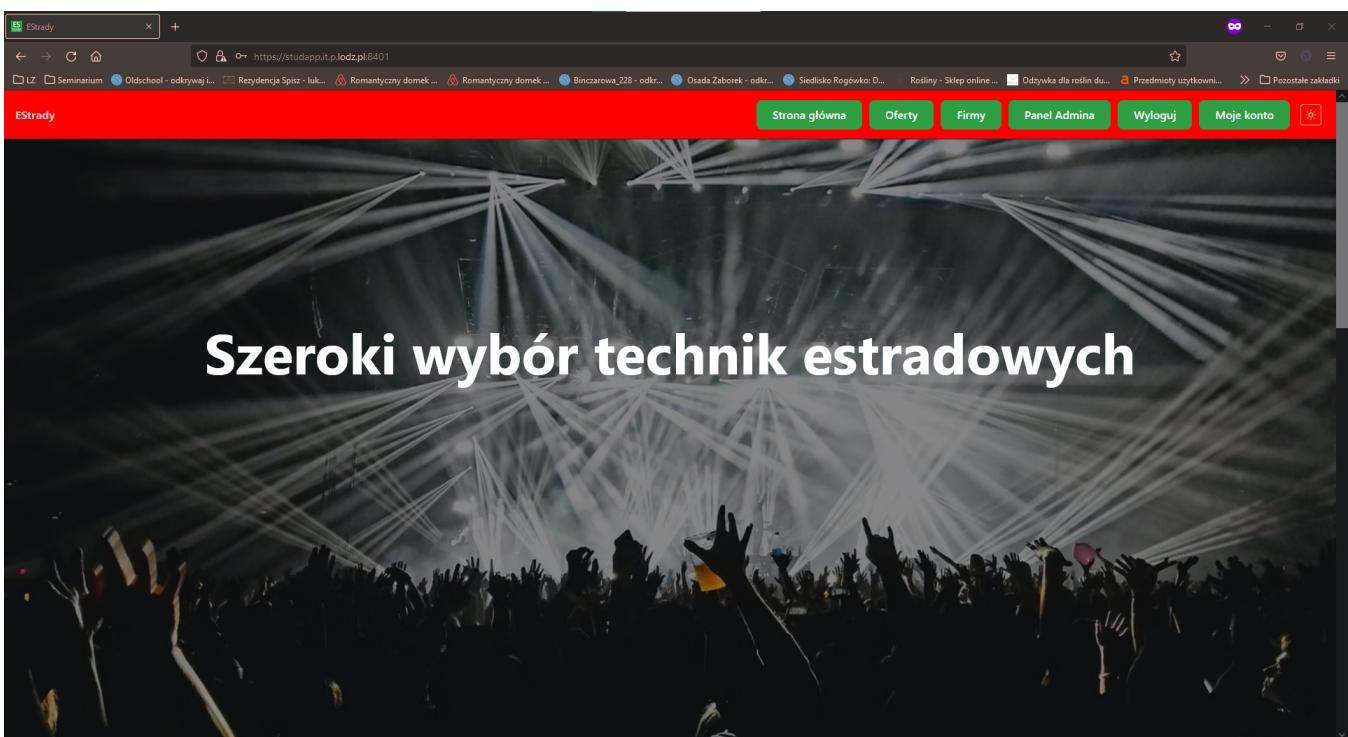
7. Odłączenie poziomu dostępu do konta (MOK.6)



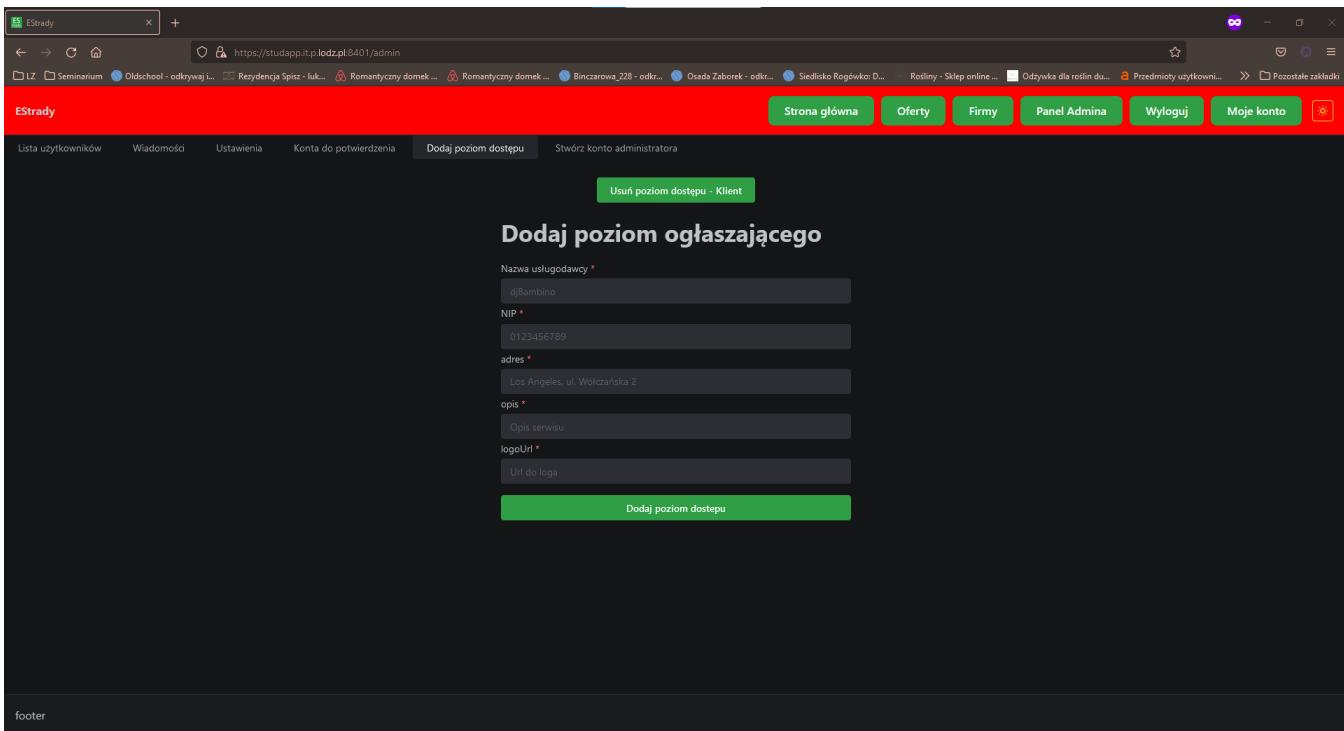
Obraz 7.1



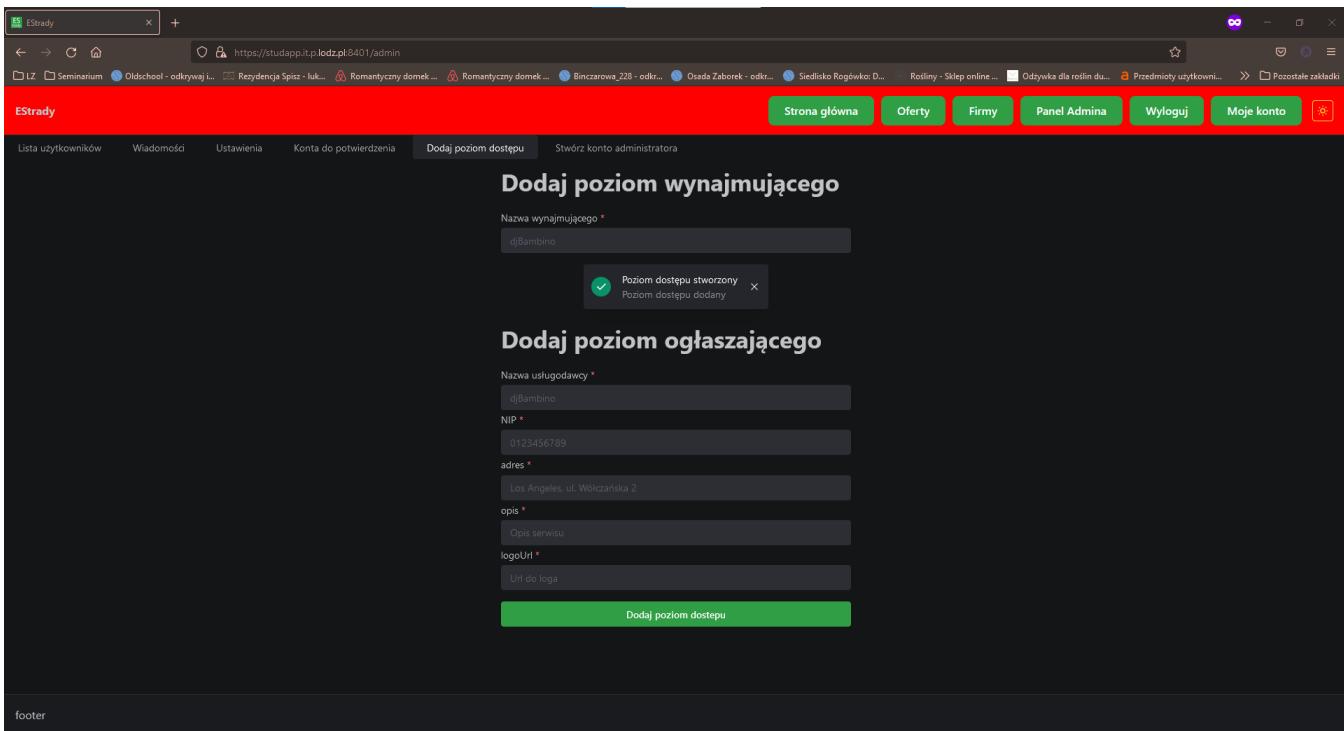
Obraz 7.2



Obraz 7.3

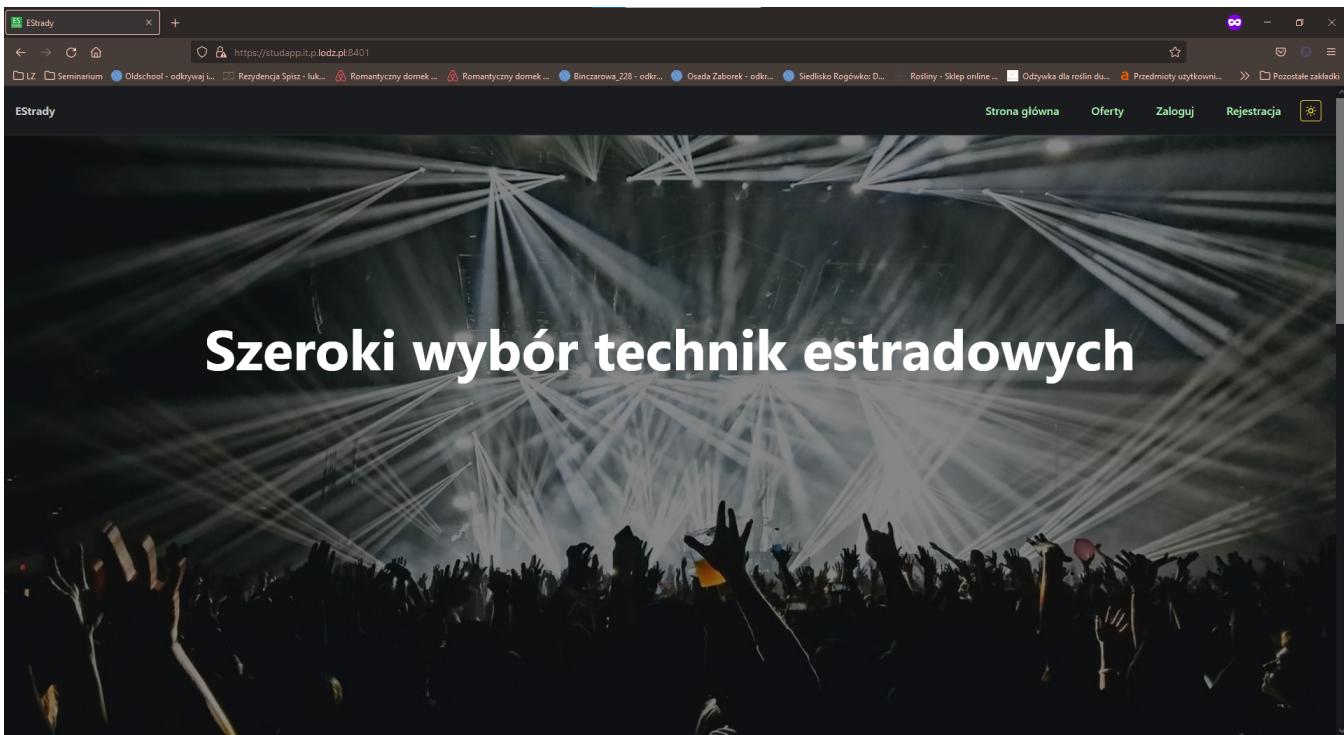


Obraz 7.4

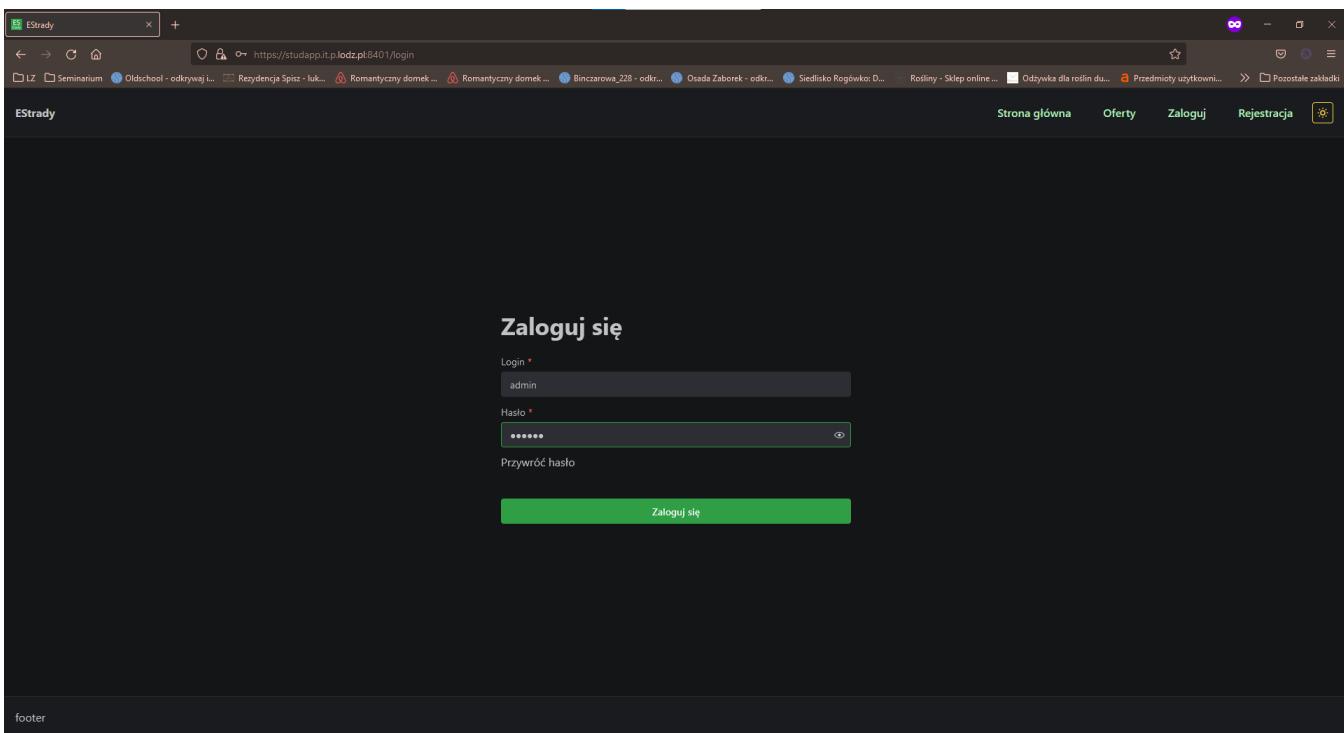


Obraz 7.5

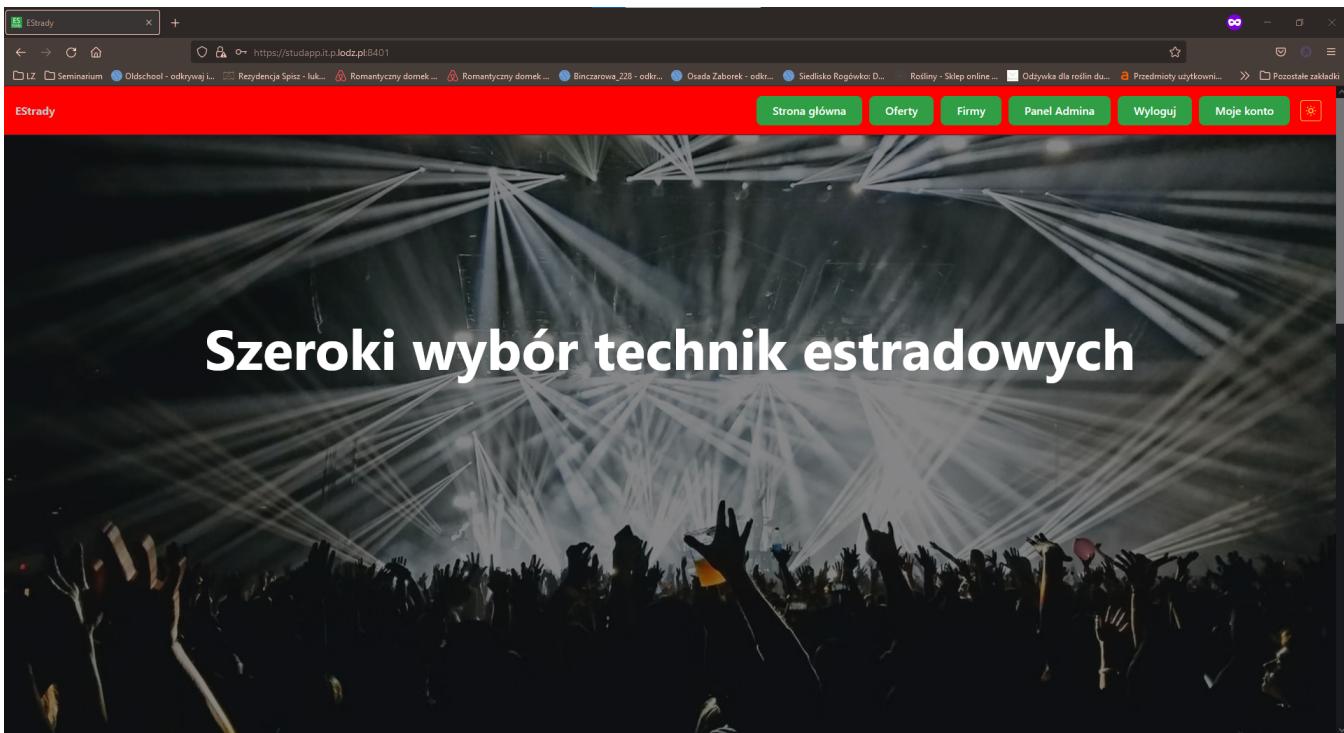
8 Zmiana własnego hasła (MOK. 7)



Obraz 8.1



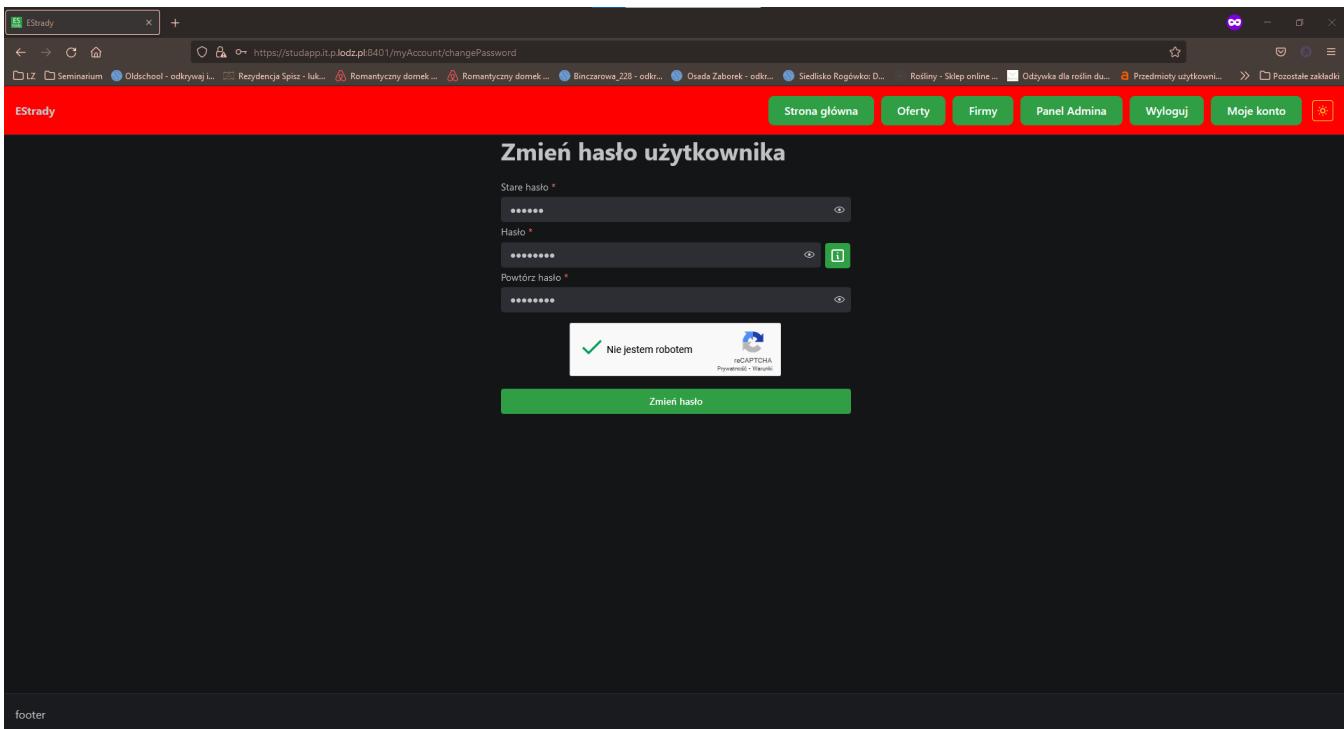
Obraz 8.2



Obraz 8.3

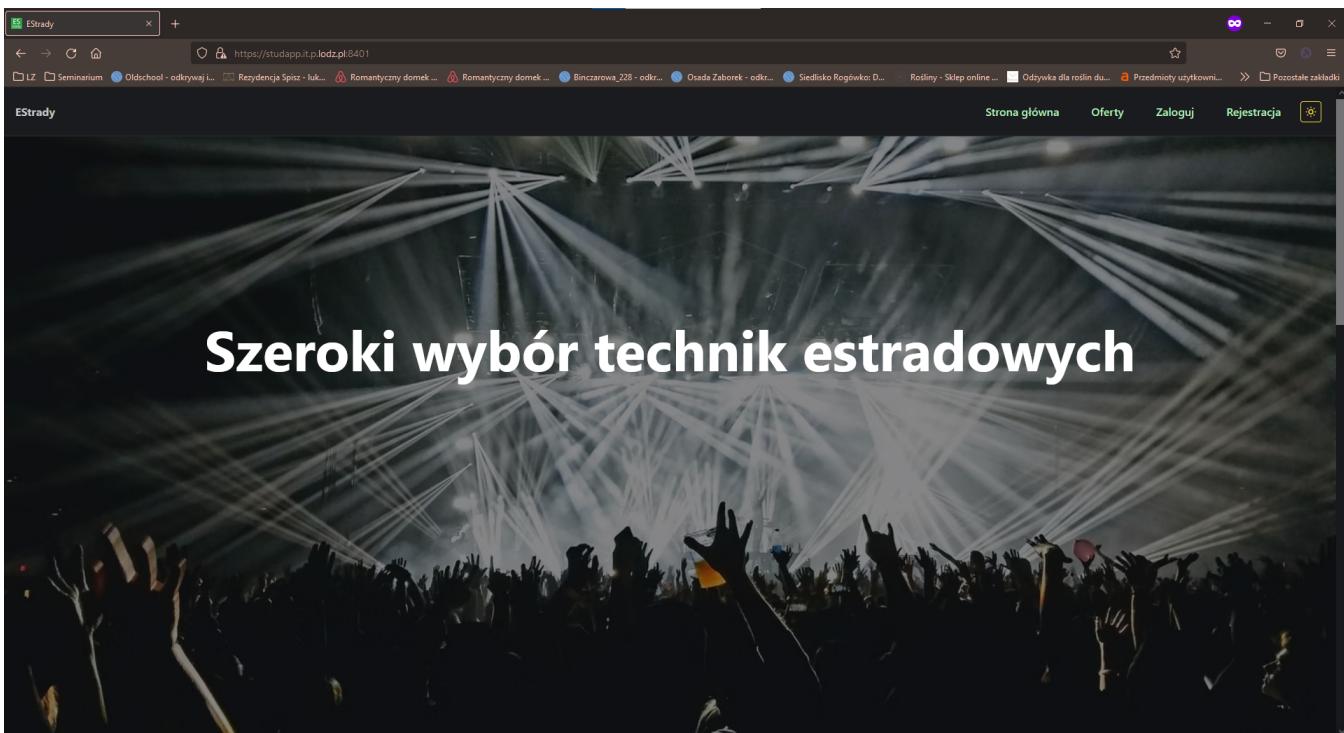
A screenshot of the Estrady website's admin panel, accessible via the 'Panel Admina' link in the top navigation. The page shows a list of users and options for managing accounts. A modal window is open, prompting the user to 'Dodaj poziom wynajmującego' (Add rental level). It includes fields for 'Nazwa wynajmującego' (Name of the tenant) and a dropdown for 'Poziomy dostępu' (Access levels), which is set to 'Admin'. Another modal window is visible in the background, showing a success message: 'Poziom dostępu stworzony' (Access level created). Below these, another form is shown for 'Dodaj poziom ogłoszającego' (Add advertising level), with fields for 'Nazwa usługodawcy' (Supplier name), 'NIP' (Tax ID), 'adres' (Address), 'opis' (Description), and 'logoURL' (Logo URL). A green button at the bottom right of this form says 'Dodaj poziom dostępu' (Add access level).

Obraz 8.4

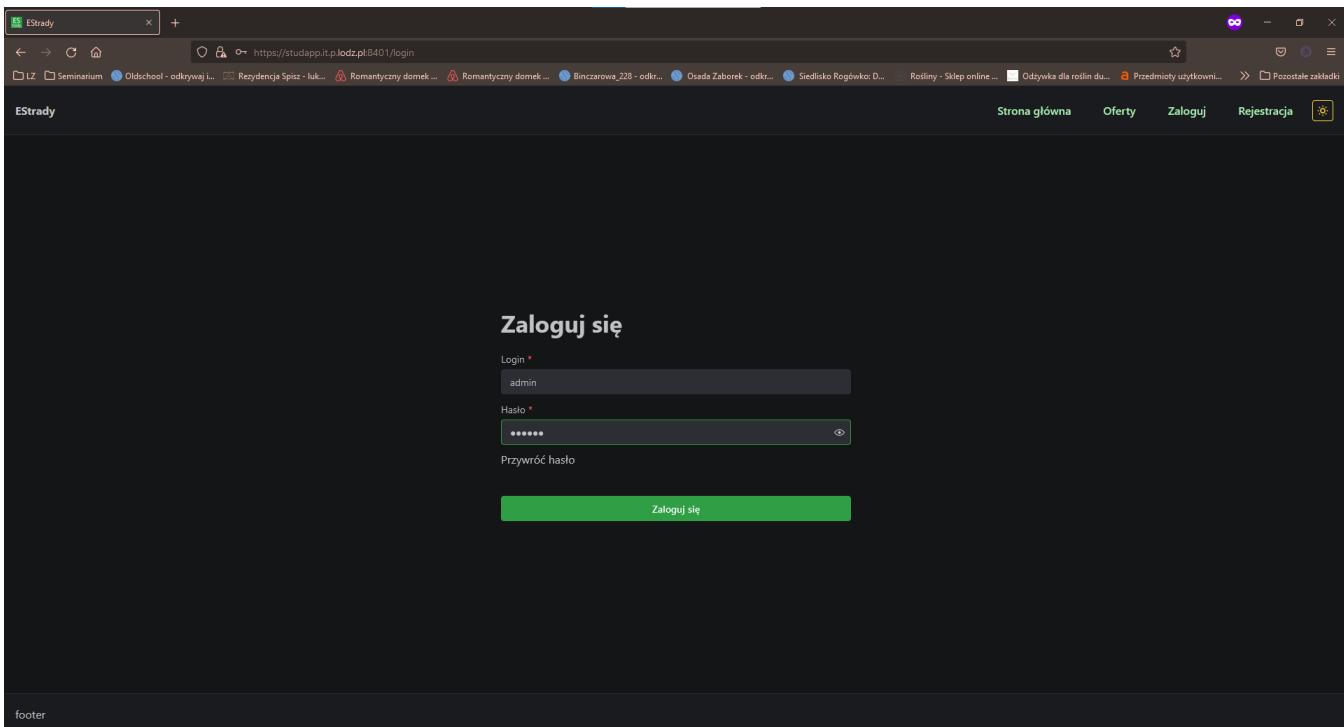


Obraz 8.5

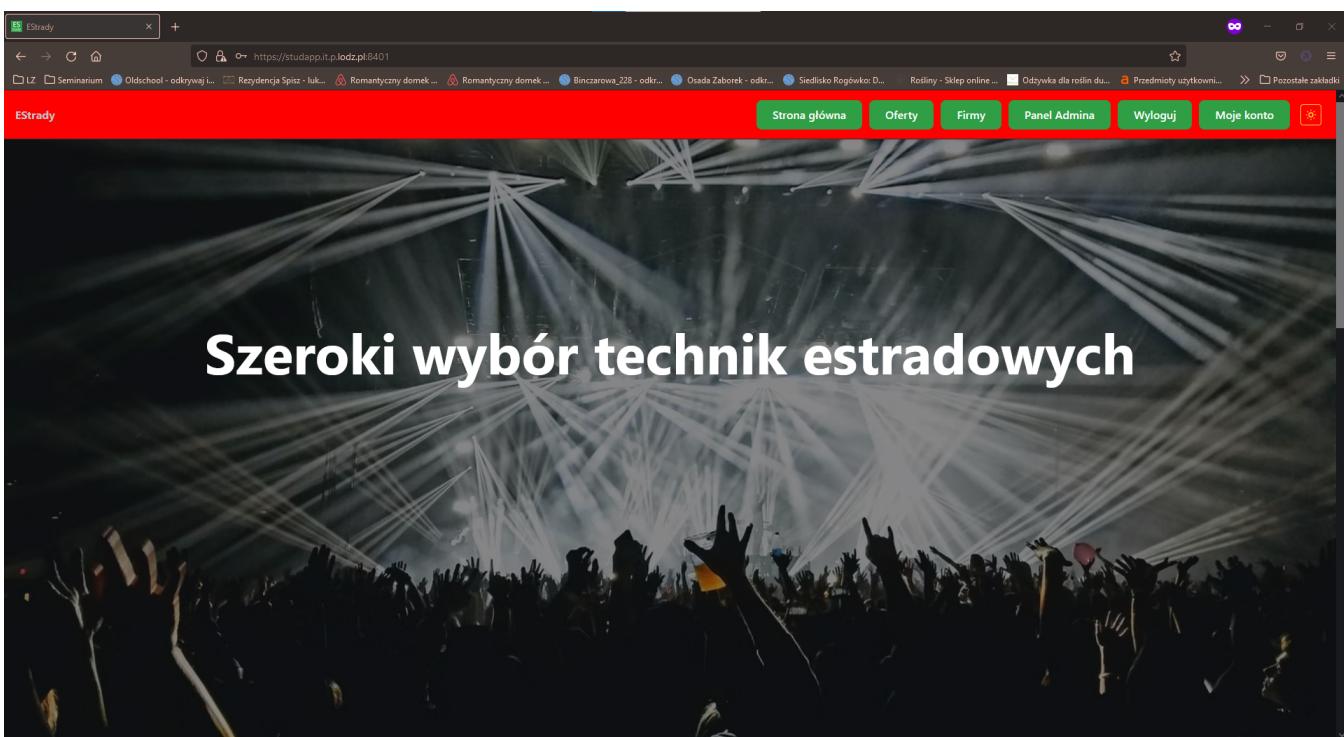
9. Zmiana hasła przez administratora (MOK.8)



Obraz 9.1



Obraz 9.2



Obraz 9.3

Estrady

Strona główna Oferty Firmy Panel Admina Wyloguj

Znajdź użytkownika Szukaj...

| Imię | Nazwisko | Login | Status |
|---------------------|----------|-----------------|-------------------------------------|
| Ssbadmin | Ssbd | admin | Aktywne Zmień hasło |
| Test | Test | adminSpaw | Aktywne Zmień hasło |
| Ssbdrenter | Ssbd | renter | Aktywne Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter2 | Aktywne Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter3 | Aktywne Zmień hasło Modyfikuj konto |
| Ssbbserviceprovider | Ssbd | serviceProvider | Aktywne Zmień hasło Modyfikuj konto |
| Test | Test | testSpawaw | Aktywne Zmień hasło Modyfikuj konto |
| Ssbd | Ssbd | testSpraw | Aktywne Zmień hasło Modyfikuj konto |

footer

Obraz 9.4

Estrady

Strona główna Oferty Firmy Panel Admina Wyloguj

Zmień hasło użytkownika

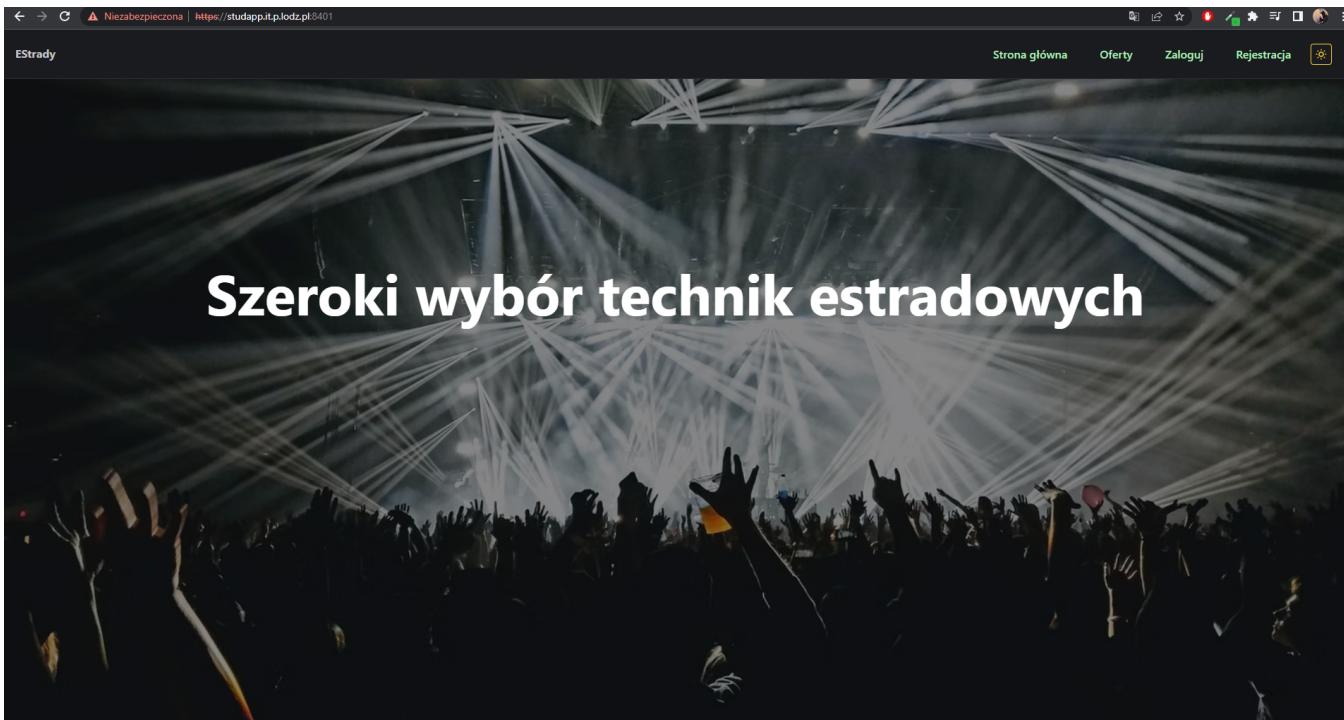
Hasło * Hasło do konta zostało zmienione
Powtóż hasło *

Hasło do konta zostało zmienione
Użytkownik może od teraz logować się nowym hasłem

footer

Obraz 9.5

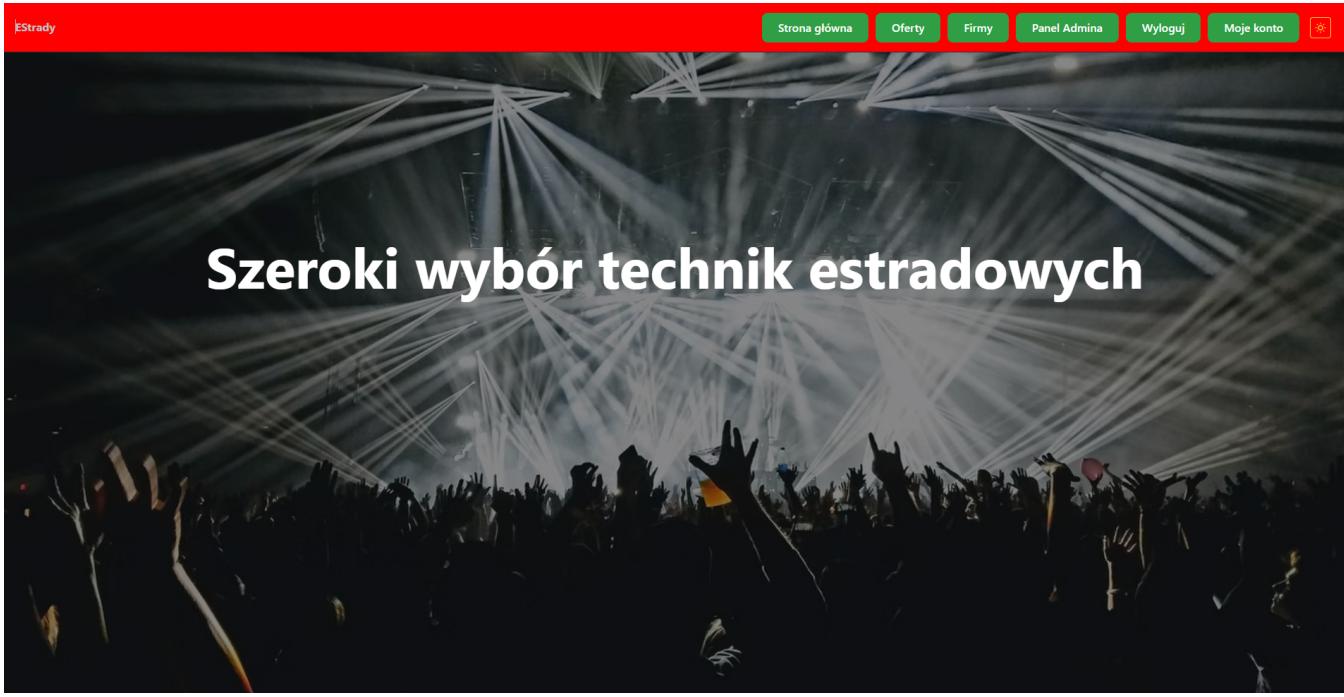
10. Edycja danych własnego konta (MOK.9)



Obraz 10.1

The screenshot shows a login page for the website. At the top, there is a navigation bar with links for 'Strona główna', 'Oferty', 'Zaloguj', 'Rejestracja', and a user icon. Below the navigation, the main content area has a heading 'Zaloguj się'. It contains two input fields: 'Login *' with the value 'admin' and 'Hasło *' with a masked value. There is also a link 'Przywróć hasło' (Reset password). A large green button at the bottom right is labeled 'Zaloguj się' (Log in).

Obraz 10.2



Obraz 10.3

Nazwa: Ssbdadmin
Poziomy dostępu: Admin

Edytuj konto
Zmień hasło

Zmień widok
Admin

Obraz 10.4

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/myAccount/edit

EStrady

Strona główna Oferty Firmy Panel Admina Wyloguj Moje konto

Edycja konta

Imię
Słbdadmin
Nazwisko
Słbd
Numer telefonu
111111111

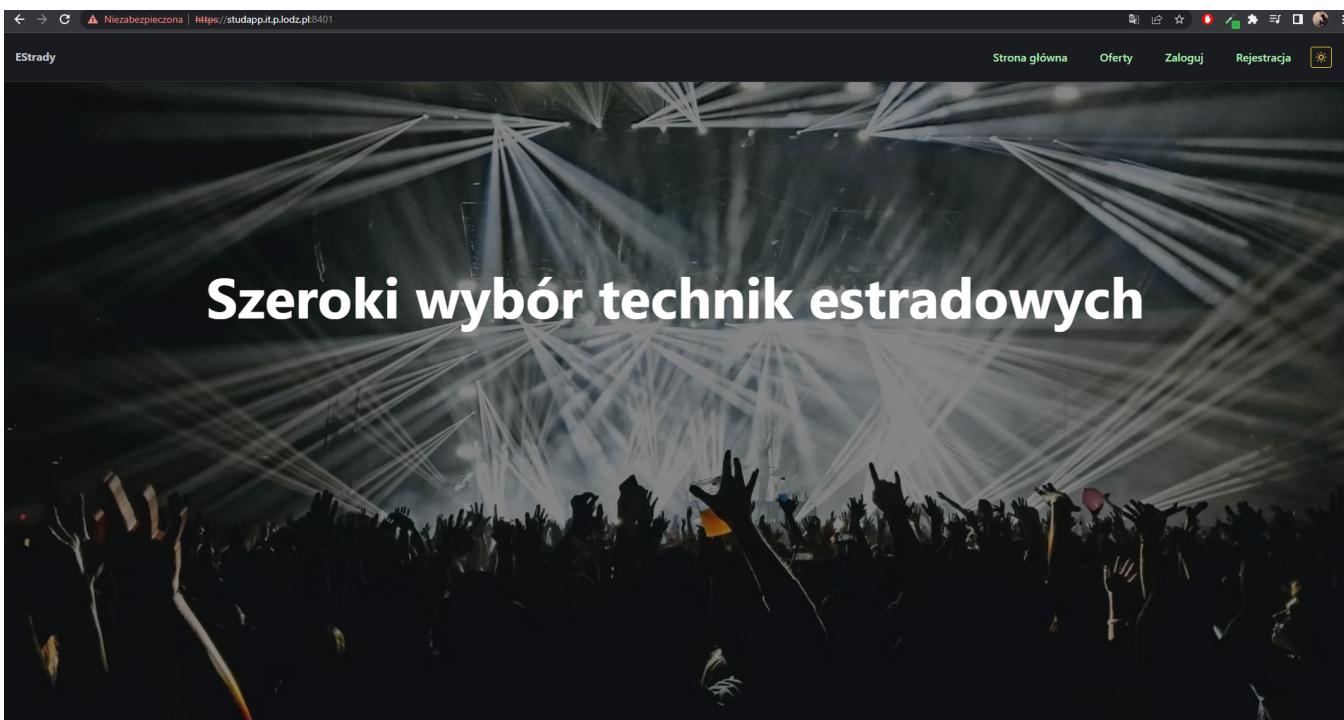
Nie jestem robotem reCAPTCHA

Zapisz

footer

Obraz 10.5

11. Edycja danych konta innego użytkownika (MOK.10)



Obraz 11.1

Niezabezpieczona | <https://studapp.it.p.lodz.pl:8401/login>

EStrady

Strona główna Oferty Zaloguj Rejestracja

Zaloguj się

Login *

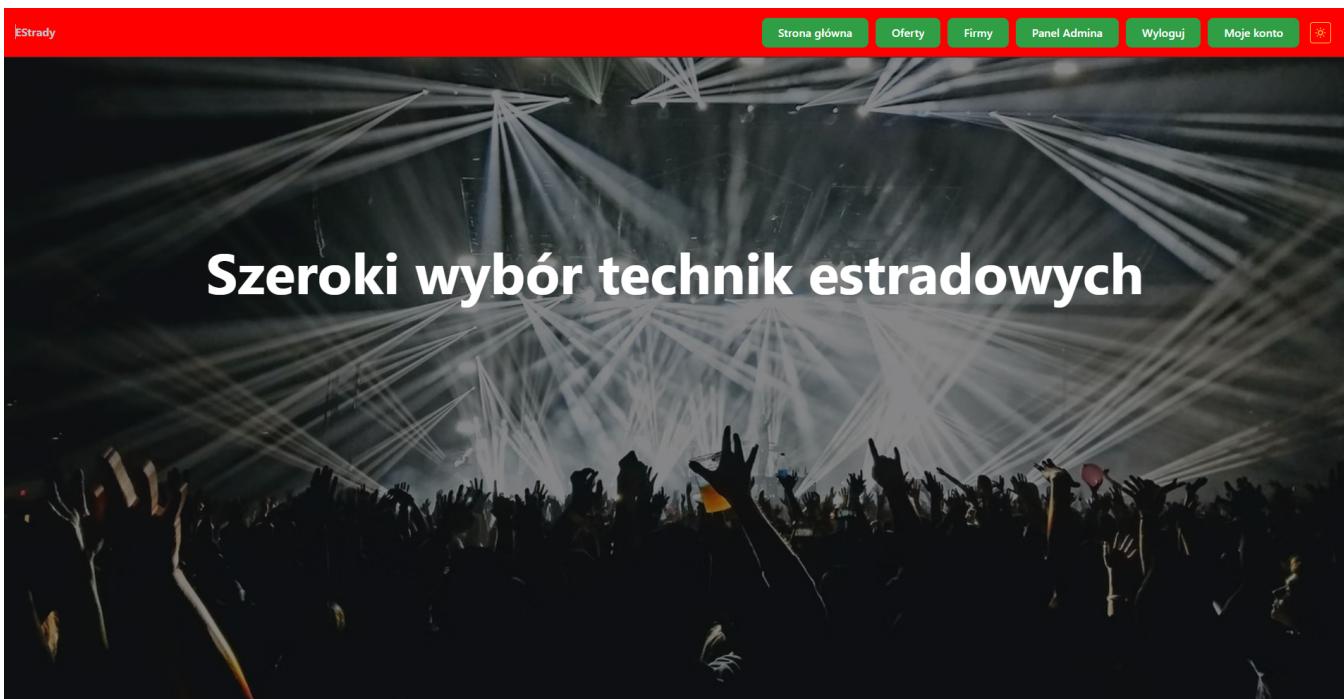
Hasło *

Przywróć hasło

Zaloguj się

footer

Obraz 11.2



Obraz 11.3

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/admin

EStrady

Strona główna Oferty Firmy Panel Admina Wyloguj Moje konto

[Lista użytkowników](#) Wiadomości Ustawienia Konta do potwierdzenia Dodaj poziom dostępu Stwórz konto administratora

Znajdź użytkownika Szukaj...

| Imię | Nazwisko | Login | Status | |
|---------------------|----------|-----------------|---------|-----------------------------|
| Ssbdadmin | Ssbd | admin | Aktywne | Zmień hasło |
| Test | Test | adminSpaw | Aktywne | Zmień hasło |
| Ssbdrenter | Ssbd | renter | Aktywne | Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter2 | Aktywne | Zmień hasło Modyfikuj konto |
| Ssbdrenter | Ssbd | renter3 | Aktywne | Zmień hasło Modyfikuj konto |
| Ssbbserviceprovider | Ssbd | serviceProvider | Aktywne | Zmień hasło Modyfikuj konto |
| Test | Test | testSpawaw | Aktywne | Zmień hasło Modyfikuj konto |
| Ssbd | Ssbd | testSpraw | Aktywne | Zmień hasło Modyfikuj konto |

footer

Obraz 11.4

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/admin/changeUser

EStrady

Strona główna Oferty Firmy Panel Admina Wyloguj Moje konto

Edycja konta

Imię
Ssbdrenter

Nazwisko
Ssbd

Numer telefonu
333333333

Nazwa użytkownika
renter1

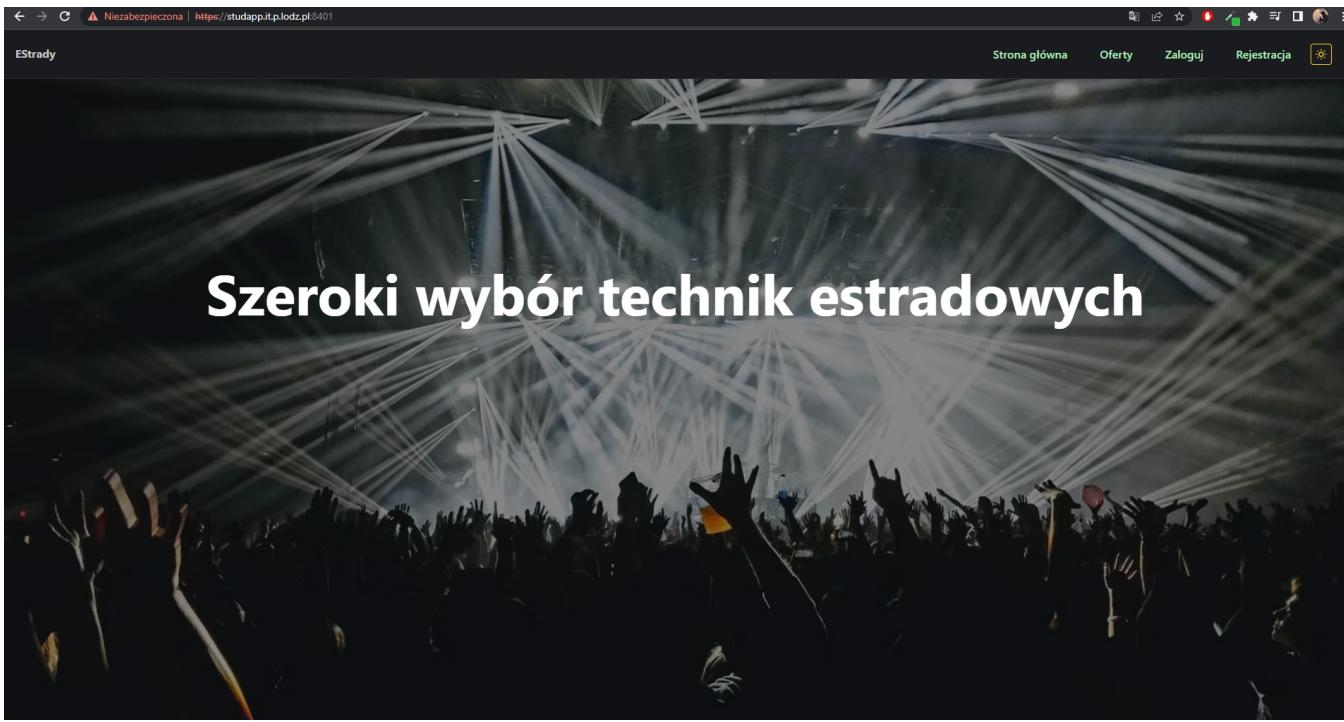
Nie jestem robotem reCAPTCHA Przyciąć - Wariunki

Zapisz

footer

Obraz 11.5

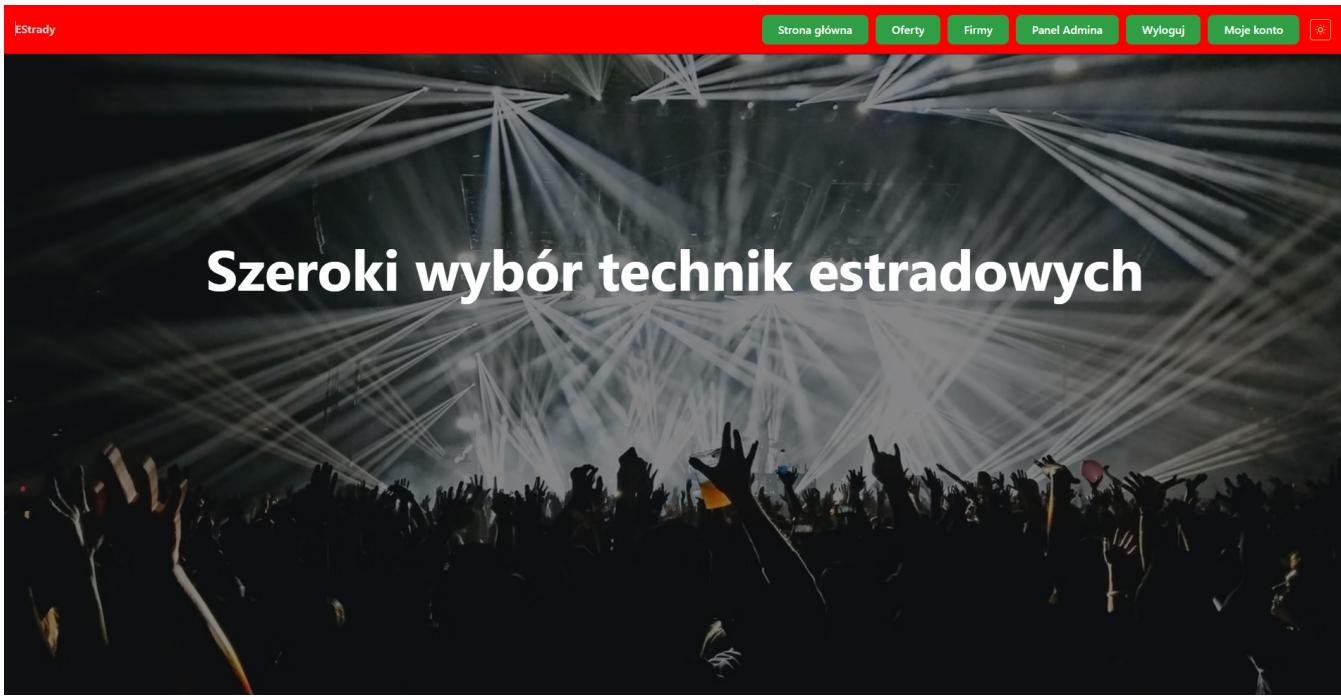
12. Wylogowanie się użytkownika. (MOK.11)



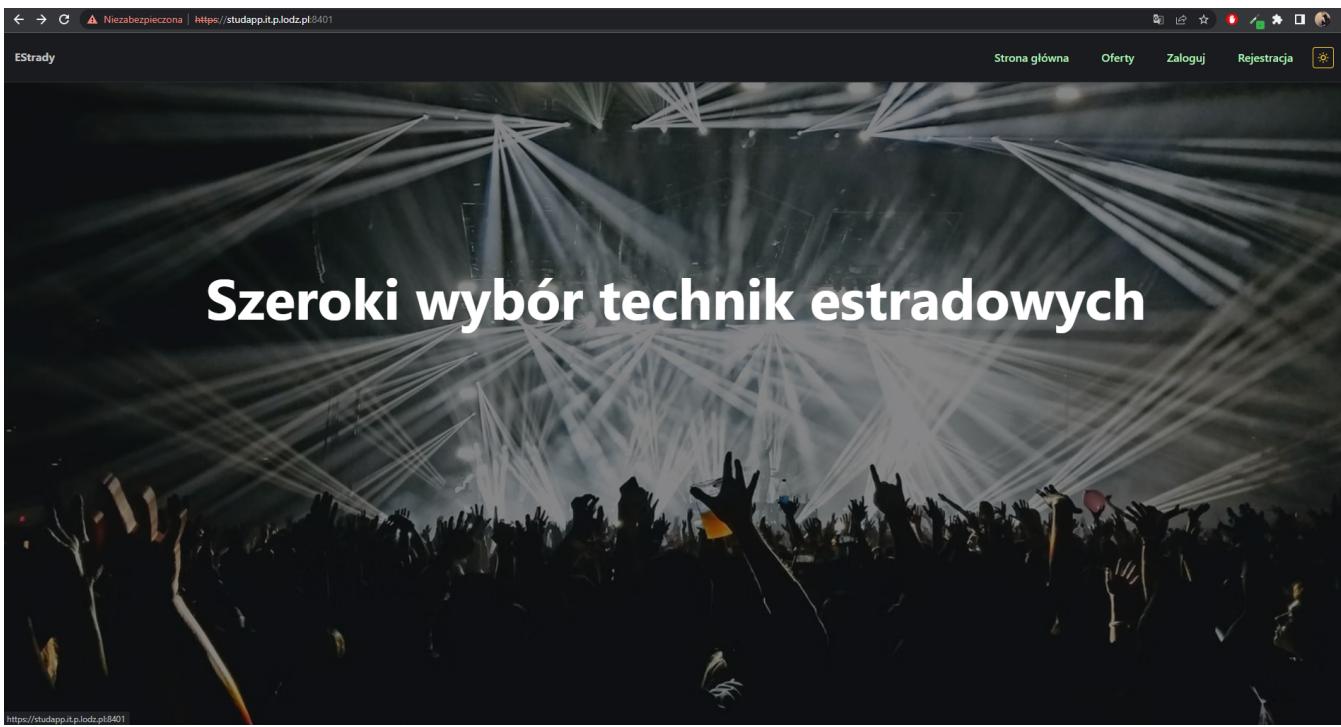
Obraz 12.1

The image shows the login page of the EStrady website. The page has a dark background. At the top, there is a header with the EStrady logo and navigation links for "Strona główna", "Oferty", "Zaloguj", and "Rejestracja". Below the header, the main content area features a heading "Zaloguj się". It contains two input fields: one for "Login" with the value "admin" and another for "Hasło" with the value "*****". There is also a link "Przywróć hasło" and a large green "Zaloguj się" button. At the bottom of the page, there is a footer section with the word "footer". The address bar at the top left indicates the URL as https://studapp.it.p.lodz.pl:8401/login.

Obraz 12.2

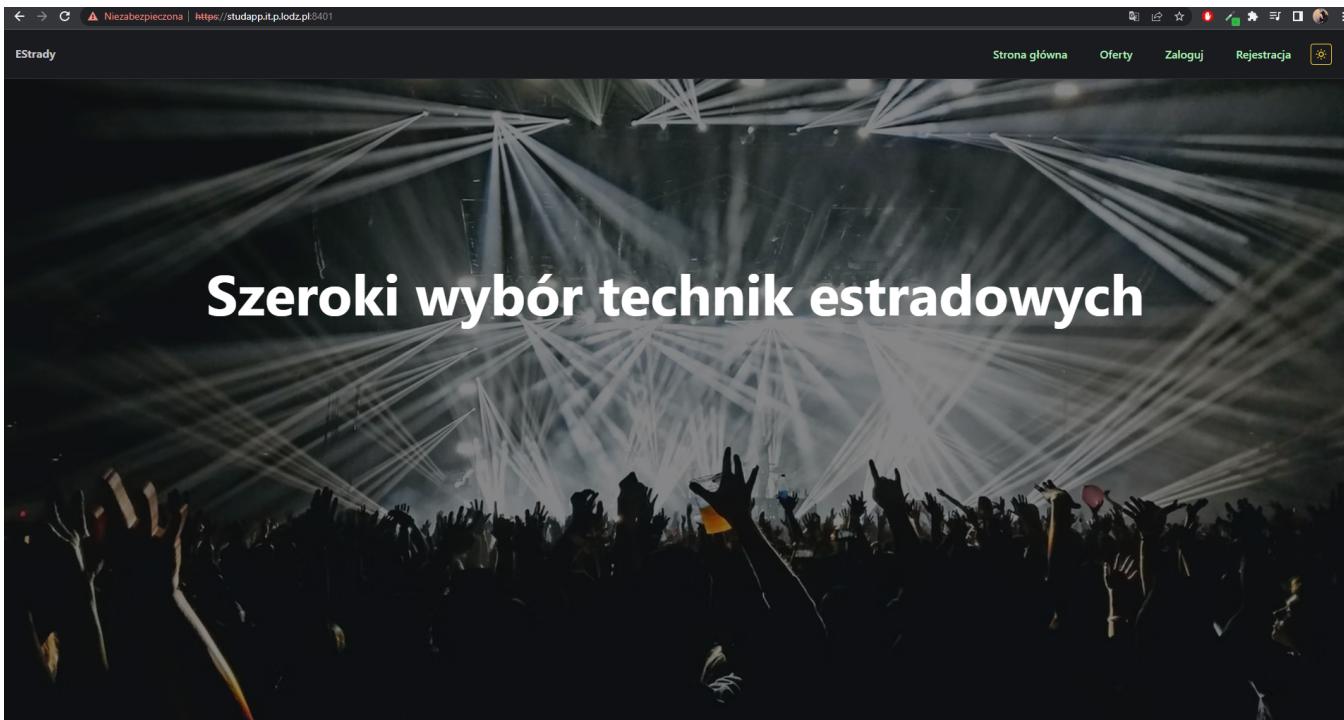


Obraz 12.3



Obraz 12.4

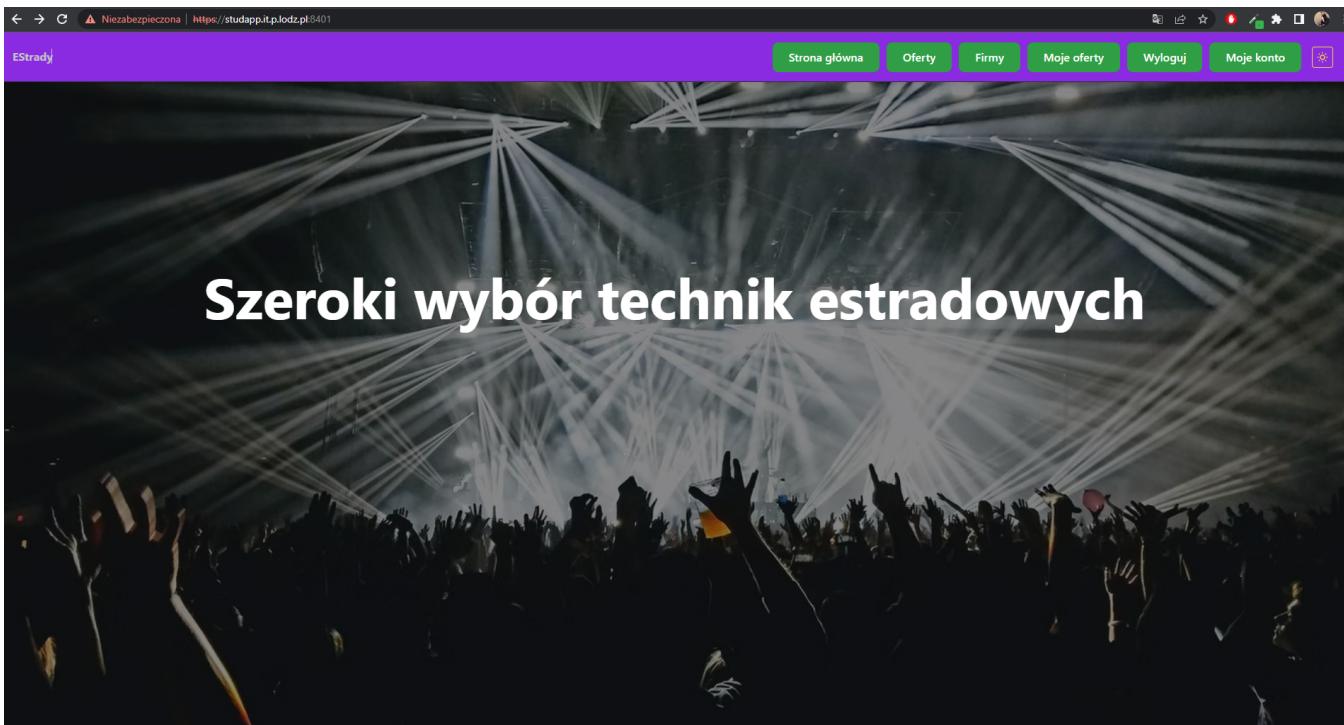
13. Aktualizacja profilu firmy (MOK.12)



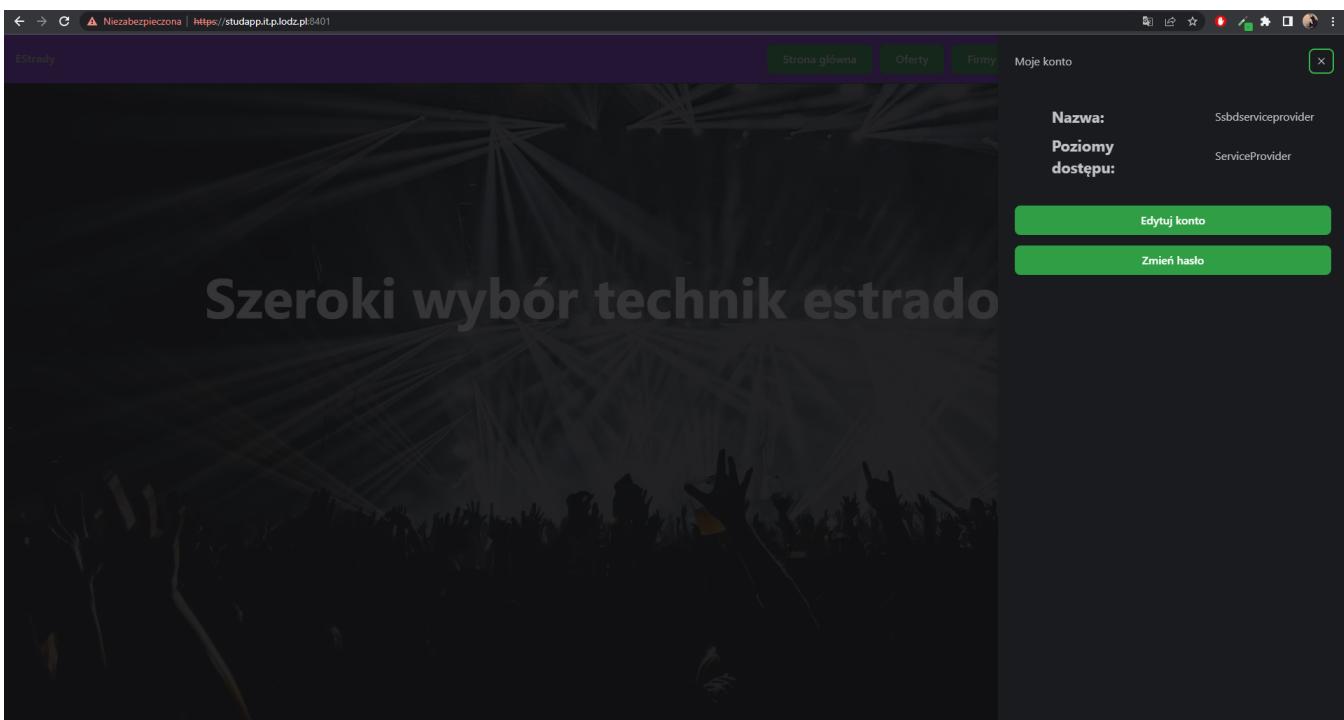
Obraz 13.1

The image shows the login page of the EStrady website. The URL in the address bar is <https://studapp.it.p.lodz.pl:8401/login>. The page has a dark header with the EStrady logo and navigation links for "Strona główna", "Oferty", "Zaloguj", and "Rejestracja". The main content area is titled "Zaloguj się" and contains two input fields: "Login" (with placeholder "serviceProvider") and "Hasło" (with placeholder "....."). Below the fields is a link "Przywrócić hasło". A green "Zaloguj się" button is at the bottom. The footer of the page is visible at the very bottom.

Obraz 13.2



Obraz 13.3



Obraz 13.4

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/myAccount/edit

EStrady

Strona główna Oferty Firmy Moje oferty Wyloguj Moje konto

Edycja konta

Imię
Szbbserviceprovider

Nazwisko
Szb

Numer telefonu
22222222

Description
Best services

Adres
New York

Logo URL
url123

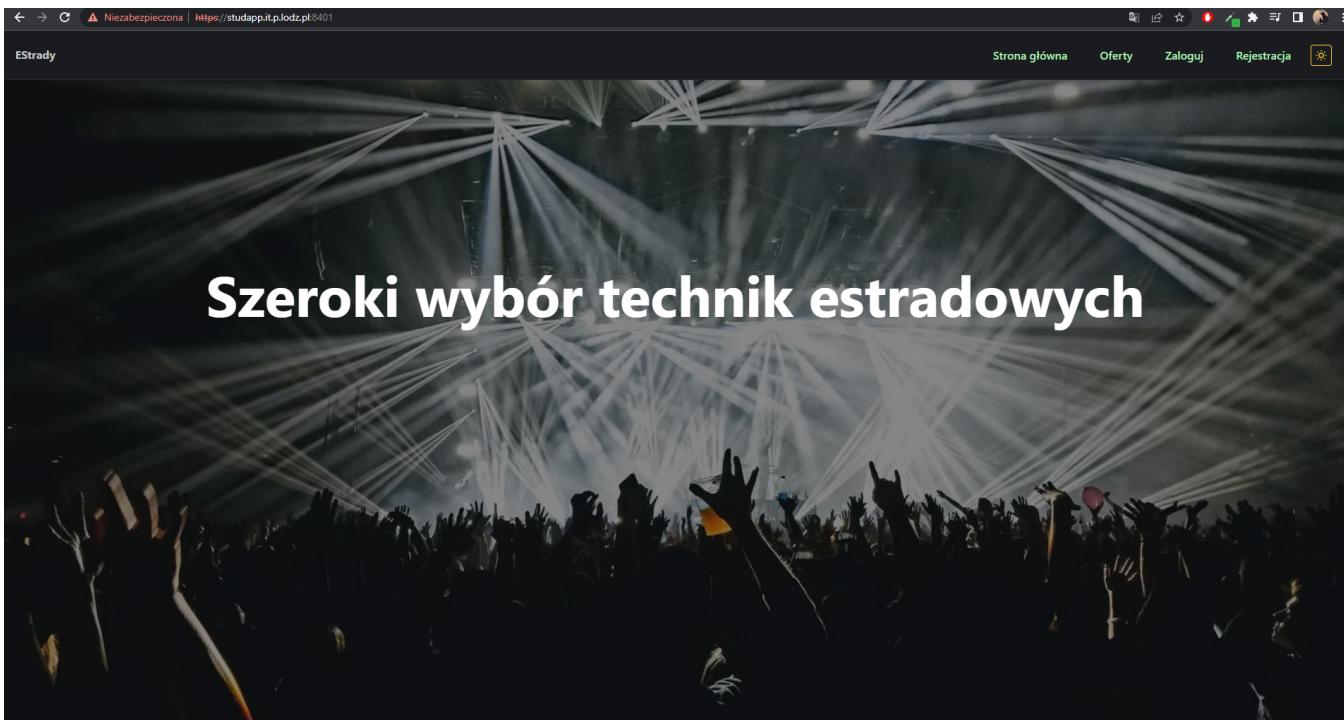
Nie jestem robotem 
reCAPTCHA
Prywatność • Regulamin

Zapisz

footer

Obraz 13.5

14. Firma zostaje zatwierdzona przez administratora (MOK.13)



Obraz 14.1

Niezabezpieczona | <https://studapp.it.p.lodz.pl:8401/login>

EStrady

Strona główna Oferty Zaloguj Rejestracja

Zaloguj się

Login *

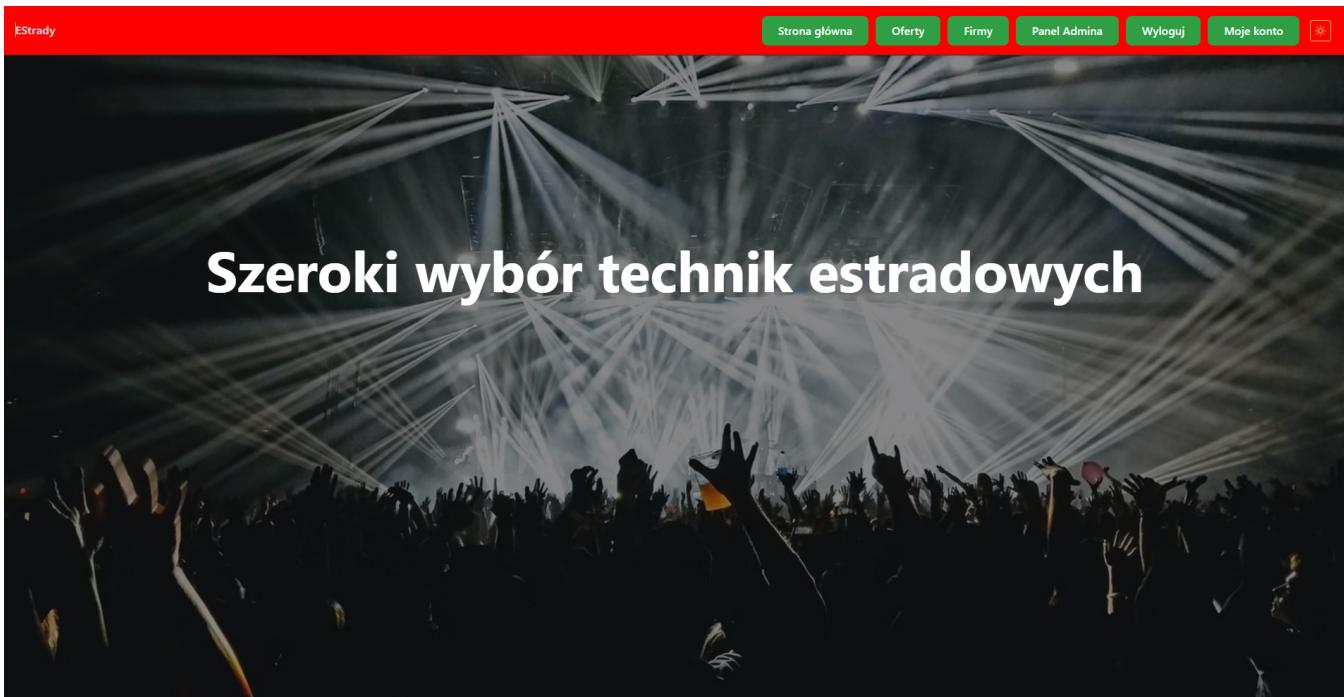
Hasło *

Przywróć hasło

Zaloguj się

footer

Obraz 14.2



Obraz 14.3

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/admin

EStrady

Strona główna Oferty Firmy Panel Admina Wyloguj

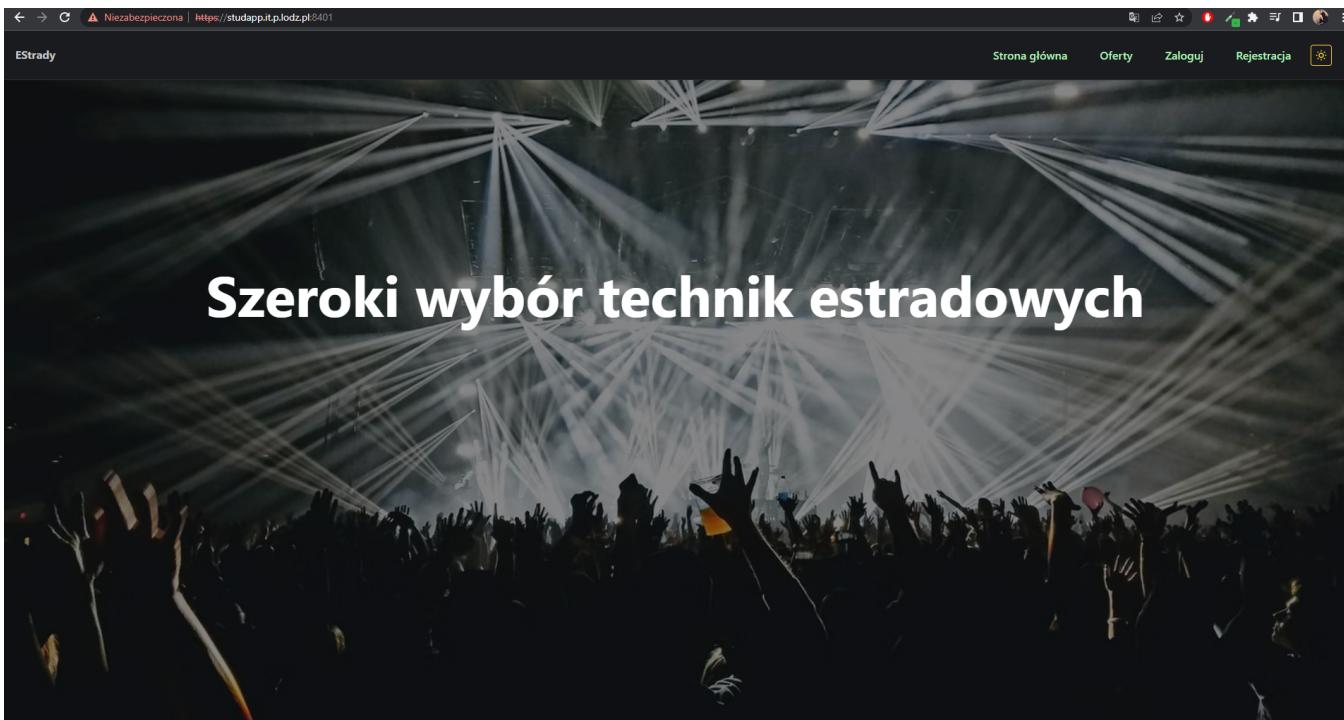
Lista użytkowników Wiadomości Ustawienia Konta do potwierdzenia Dodaj poziom dostępu Stwórz konto administratora

| Nazwa usługodawcy | NIP | Adres | Szczegóły | Potwierdź / Usuń |
|-------------------|------------|-------|-----------|------------------|
| Testtt | 8764326123 | test | | |

footer

Obraz 14.4

15.Firma zostaje odrzucona przez administratora (MOK.13)



Obraz 15.1

Niezabezpieczona | <https://studapp.it.p.lodz.pl:8401/login>

EStrady

Strona główna Oferty Zaloguj Rejestracja

Zaloguj się

Login *

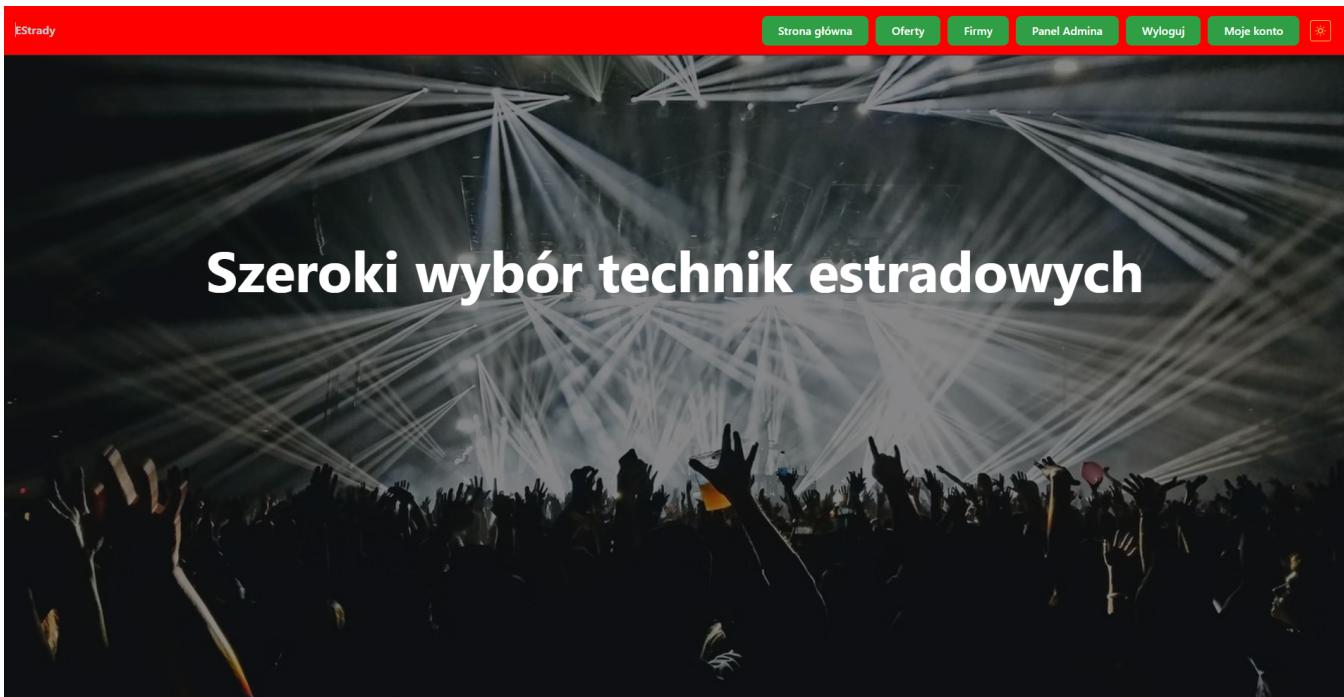
Hasło *

Przywróć hasło

Zaloguj się

footer

Obraz 15.2



Obraz 15.3

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/admin

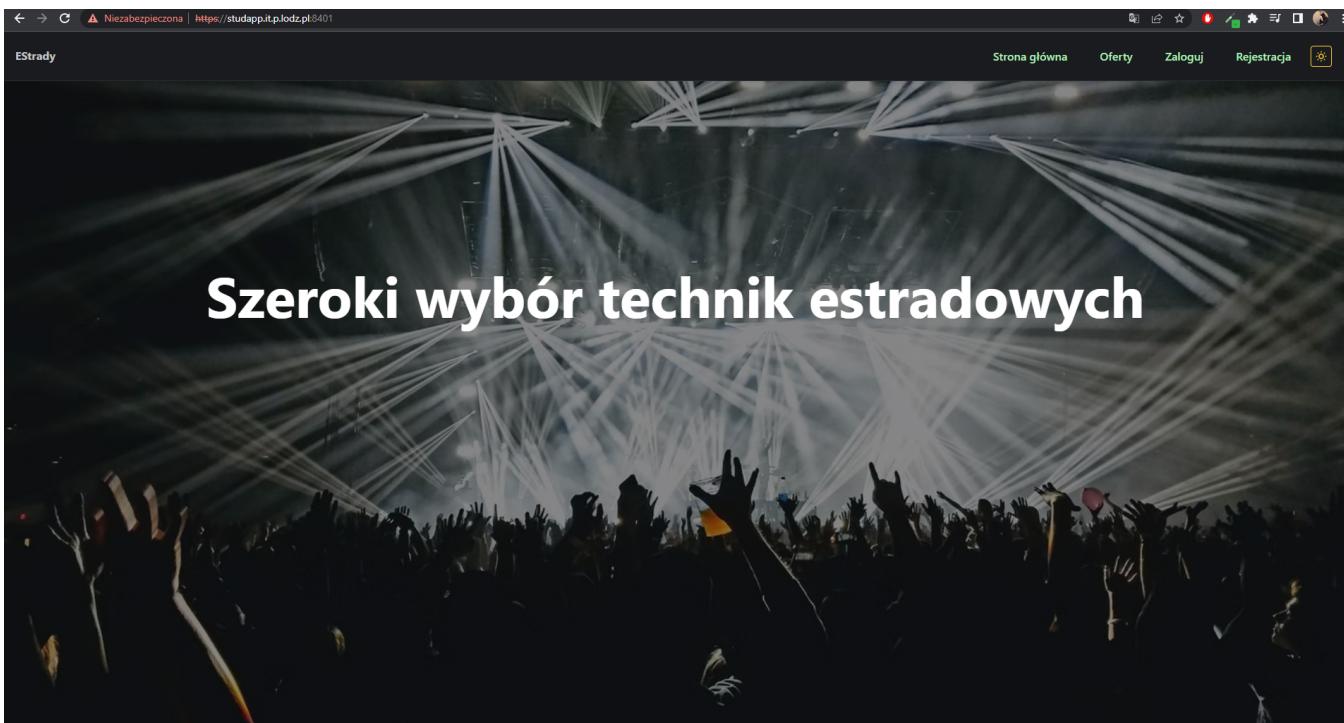
EStrady

Lista użytkowników Wiadomości Ustawienia Konta do potwierdzenia Dodaj poziom dostępu Stwórz konto administratora

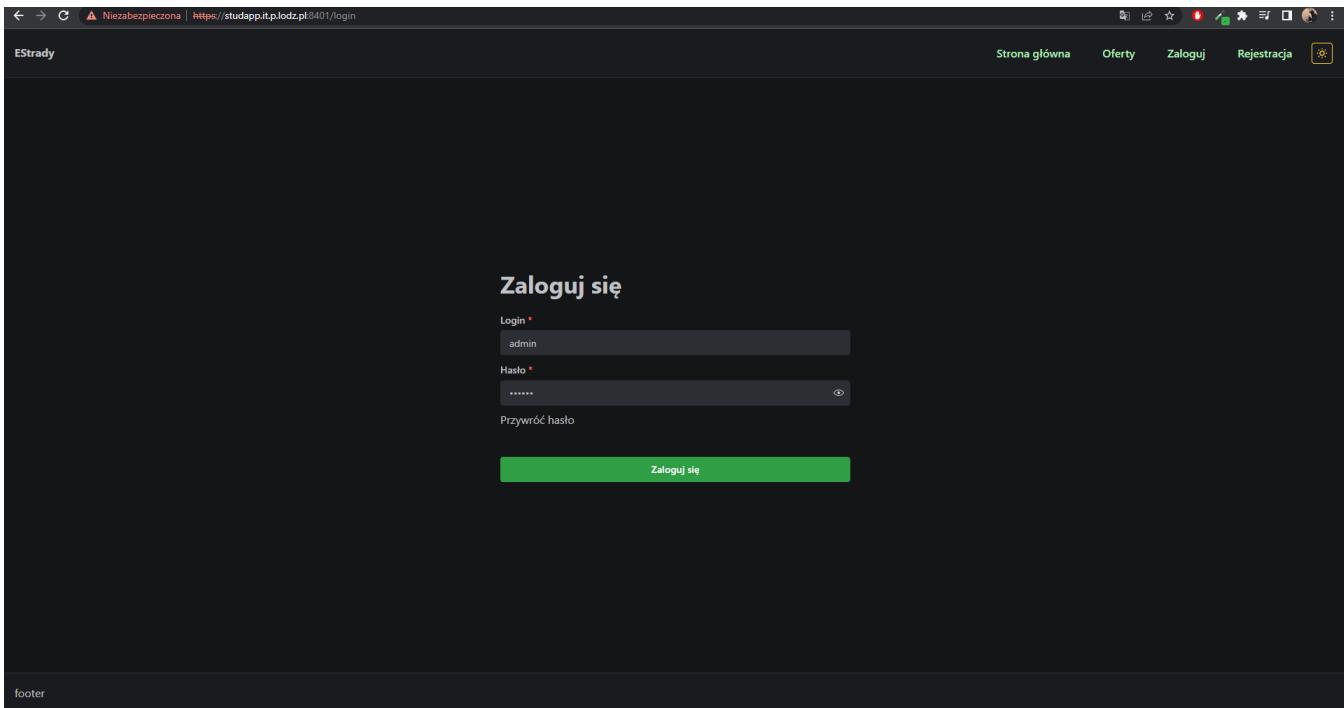
| Nazwa usługodawcy | NIP | Adres | Szczegóły | Potwierdź / Usuń |
|-------------------|------------|-------|-----------|------------------|
| Testtt | 8764326123 | test | | |

footer

16. Zalogowanie się do serwisu (MOK.14)

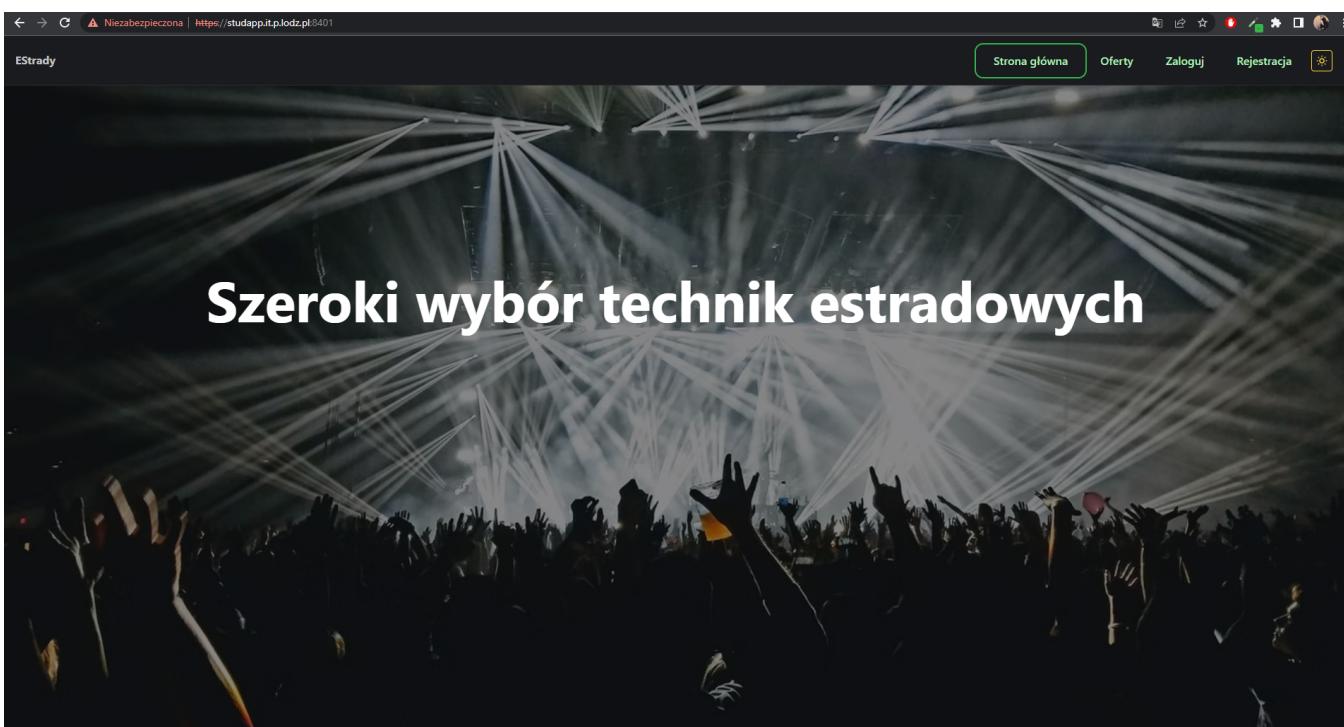


Obraz 16.1

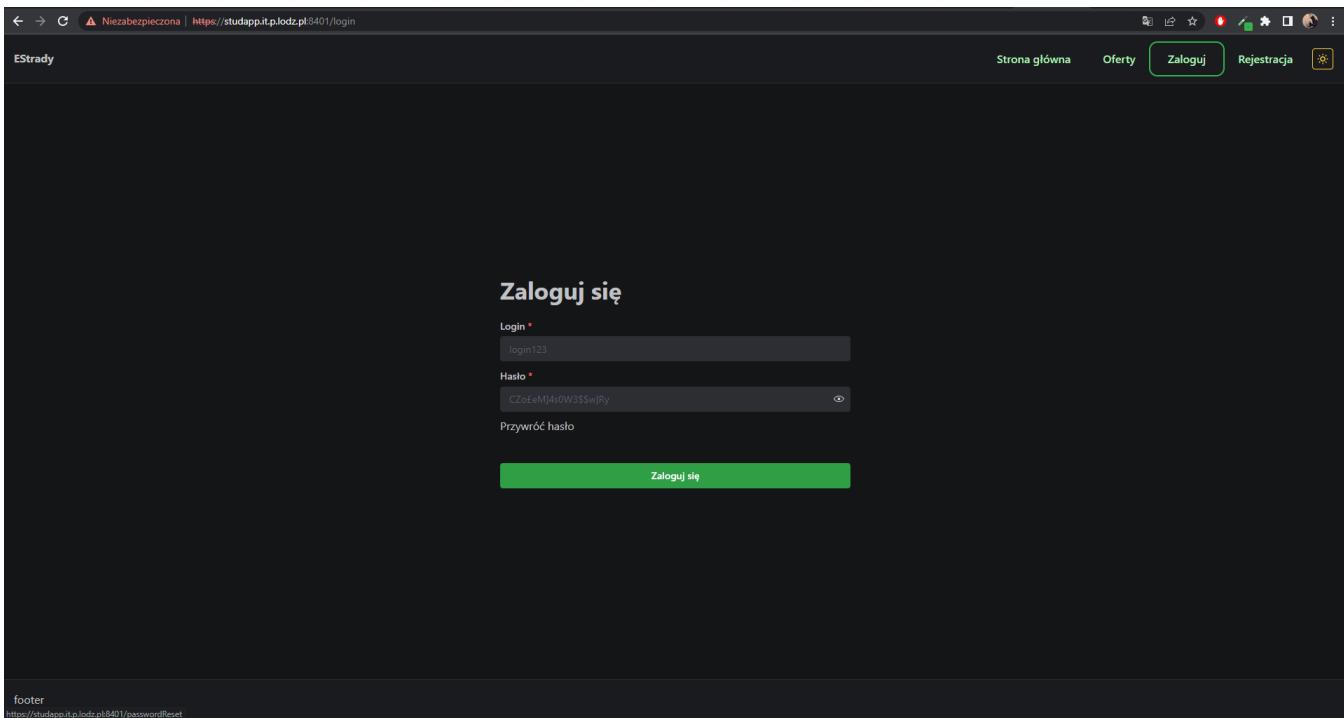


Obraz 16.2

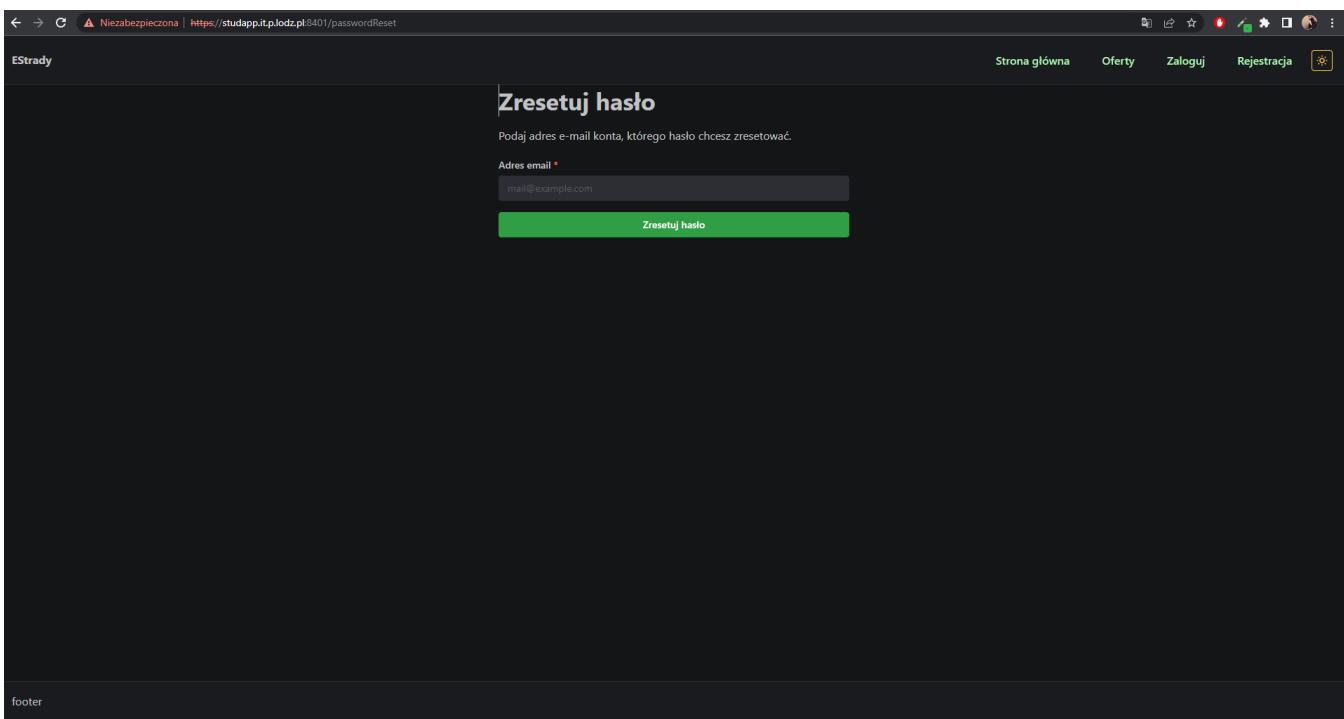
17. Resetowanie hasła, krok 1 - wysłanie maila z linkiem do zmiany hasła na podany email. (MOK.15)



Obraz 17.1

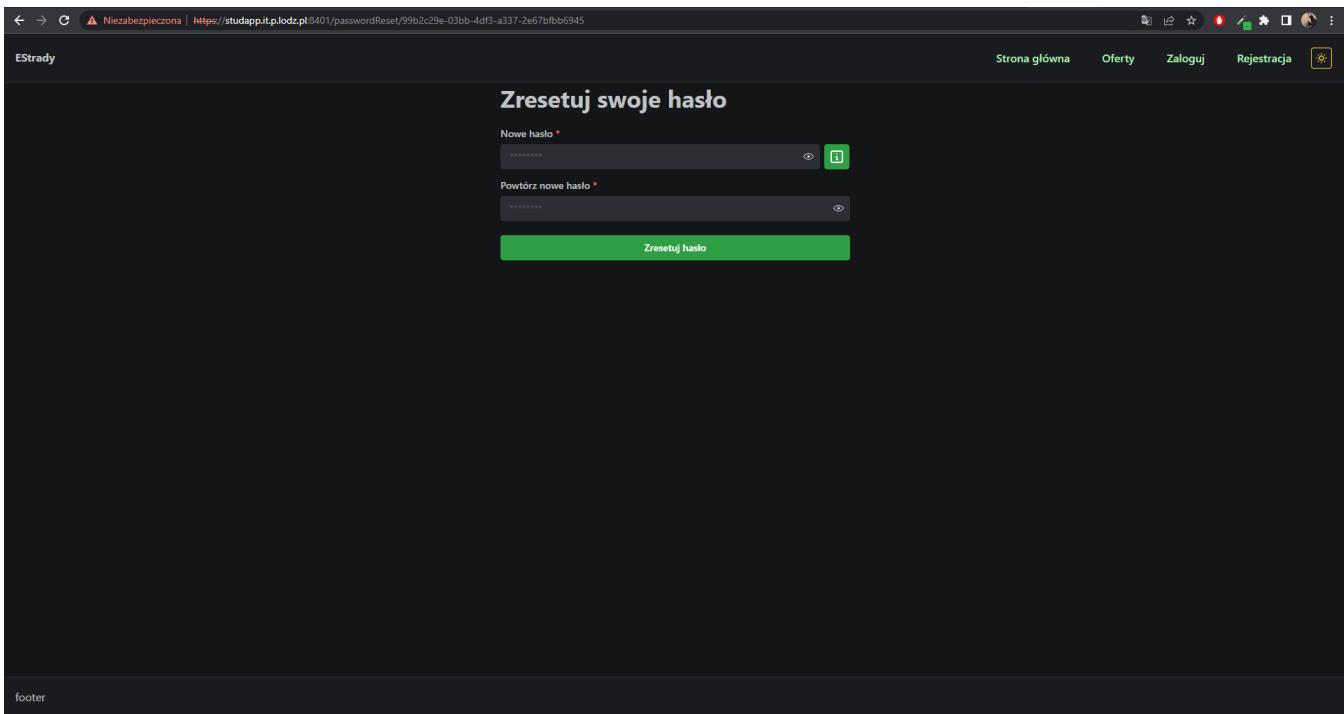


Obraz 17.2



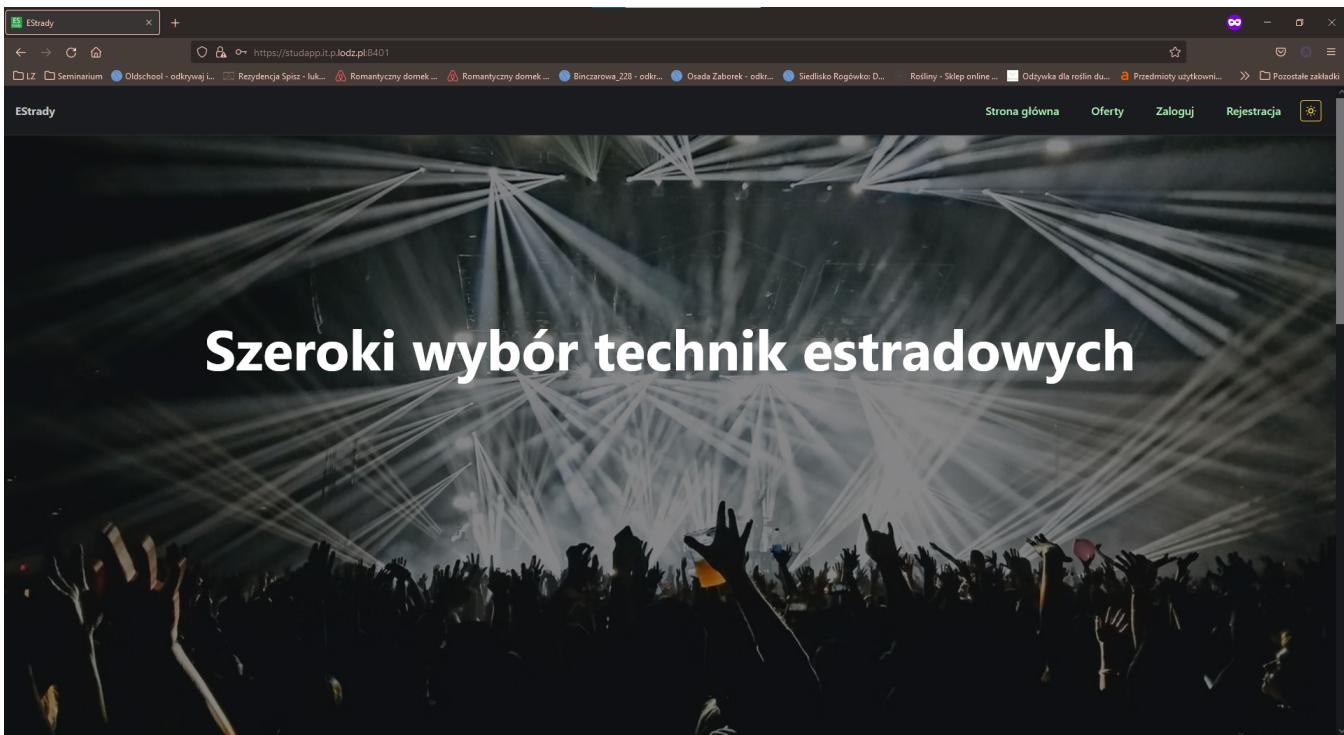
Obraz 17.3

19. Resetowanie hasła, krok 3 - reset hasła z wykorzystaniem wyżej wymienionego tokenu. (MOK.15)

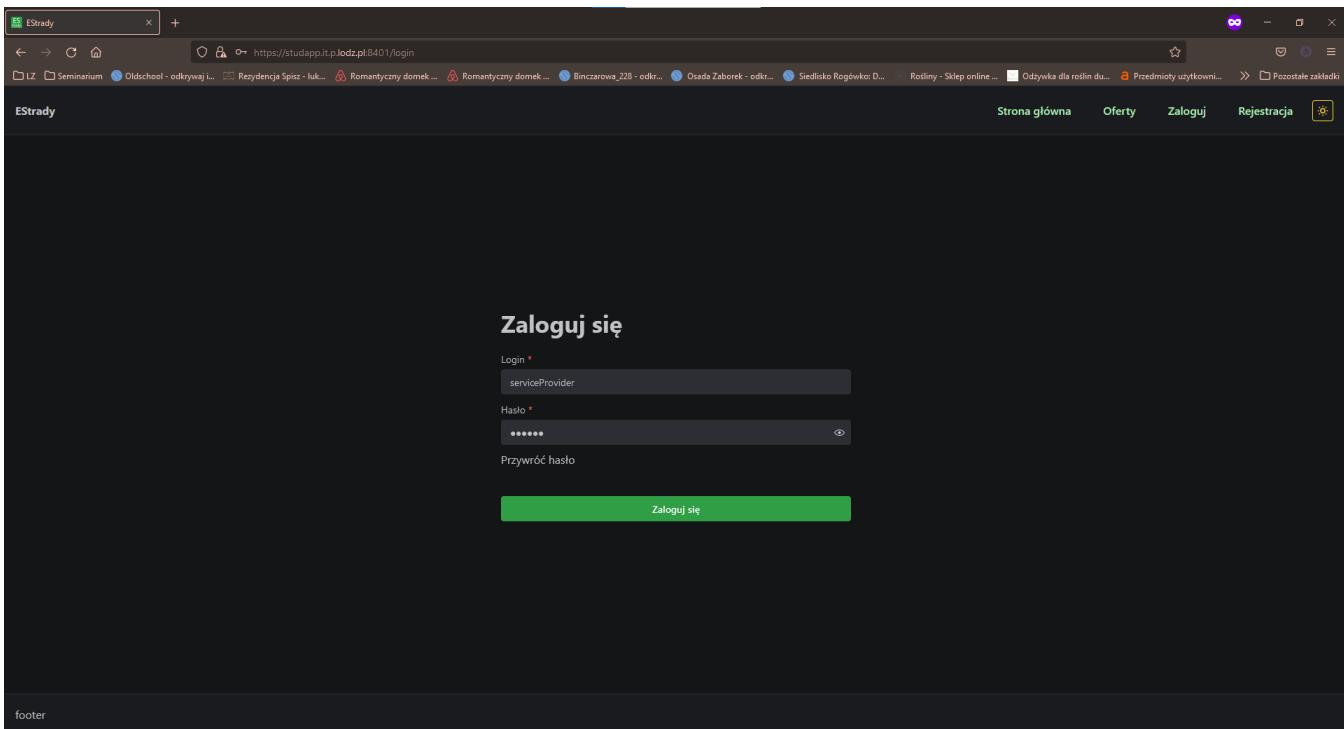


Obraz 19.1

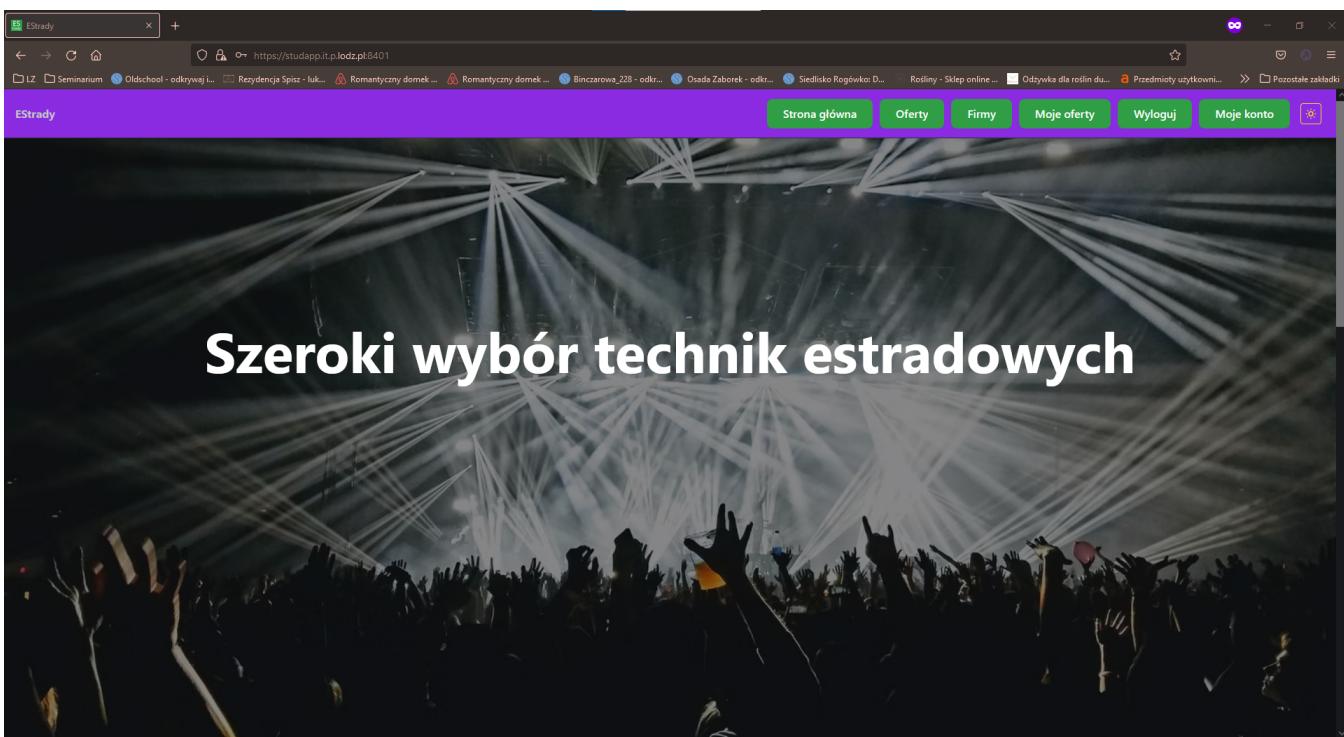
20. Dodanie oferty usługi przez użytkownika (MZ.1)



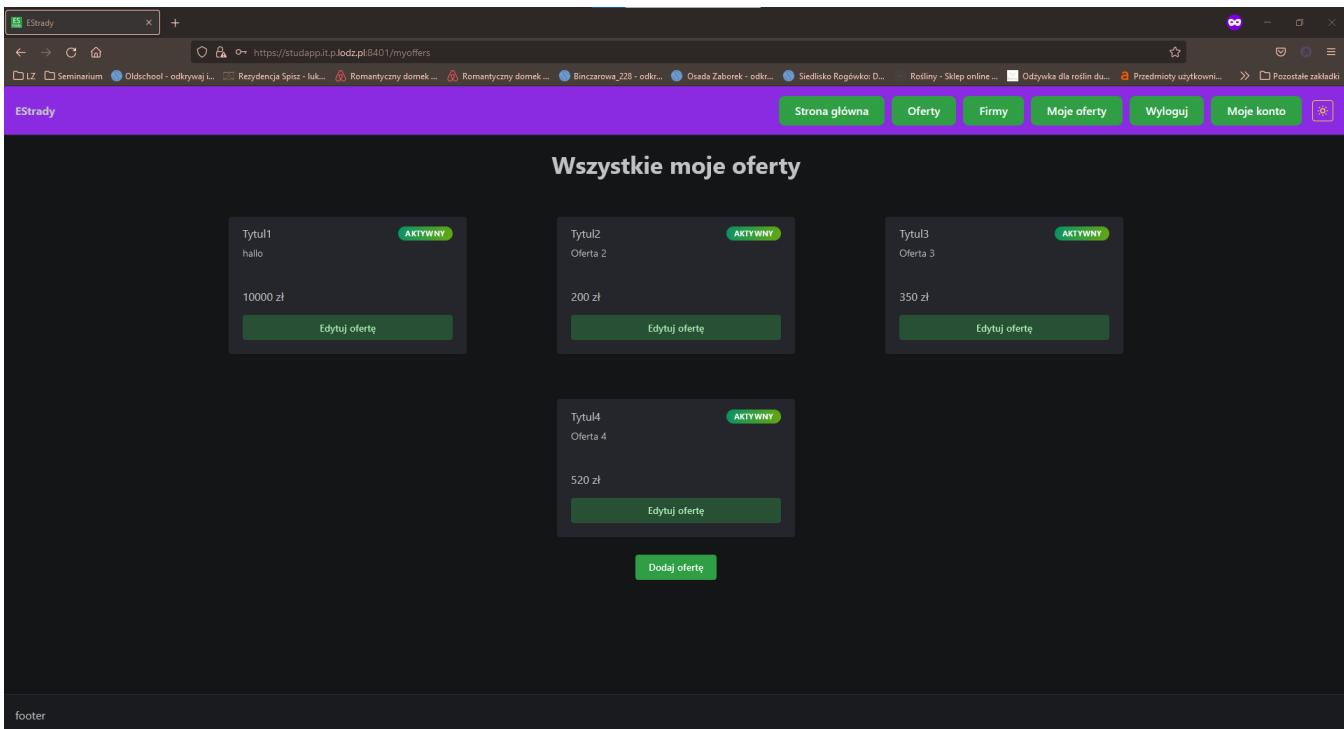
Obraz 20.1



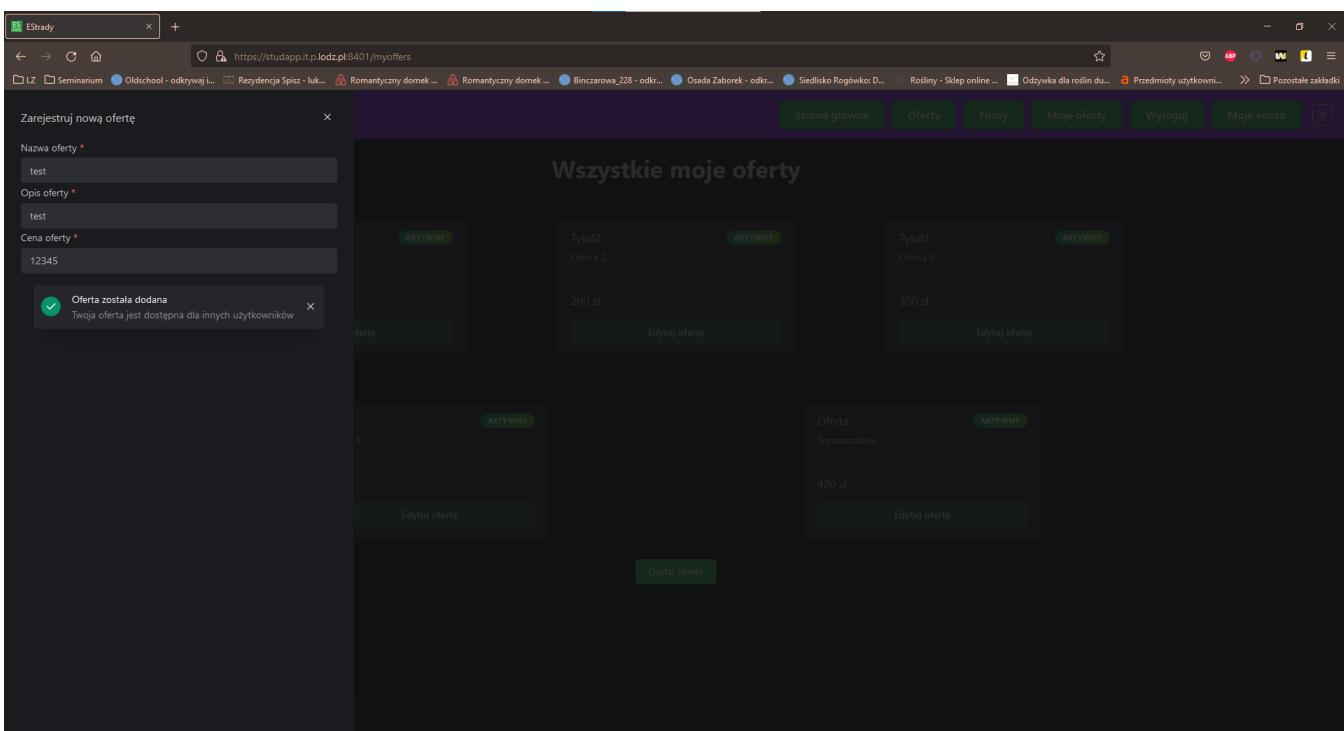
Obraz 20.2



Obraz 20.3

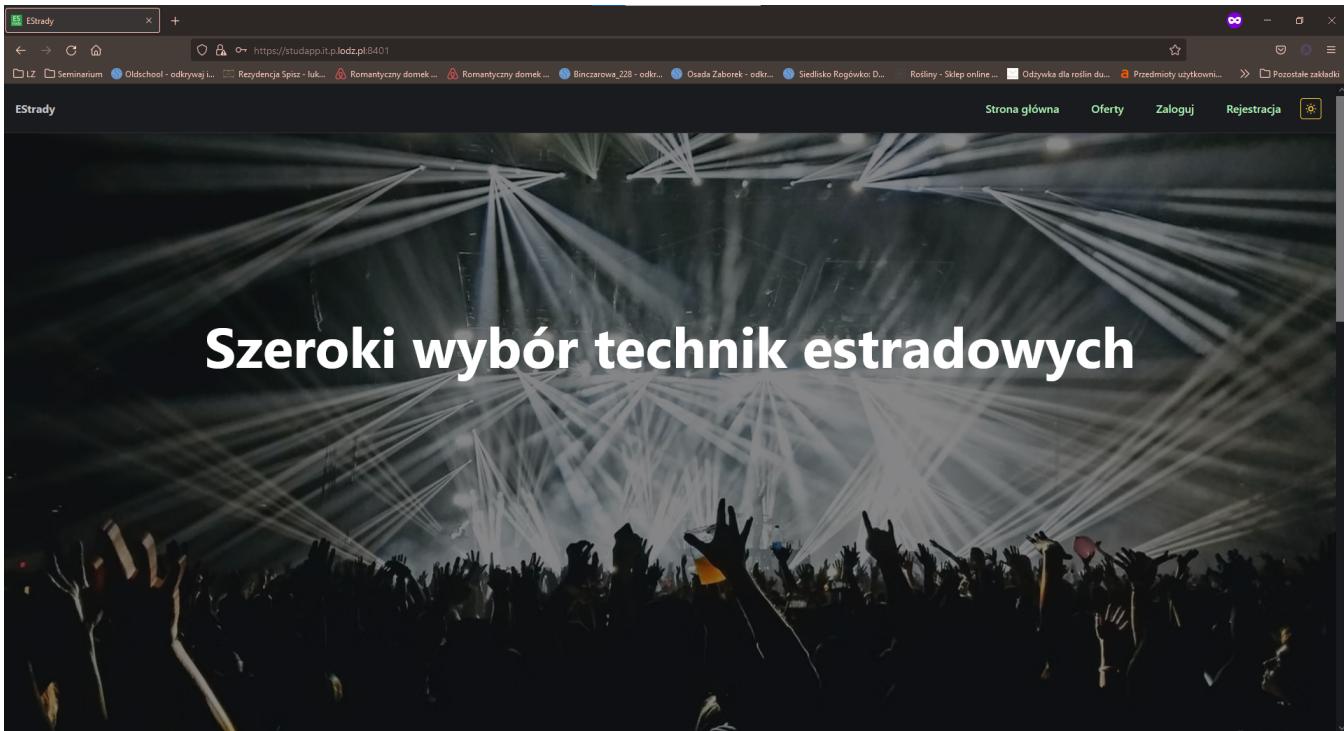


Obraz 20.4

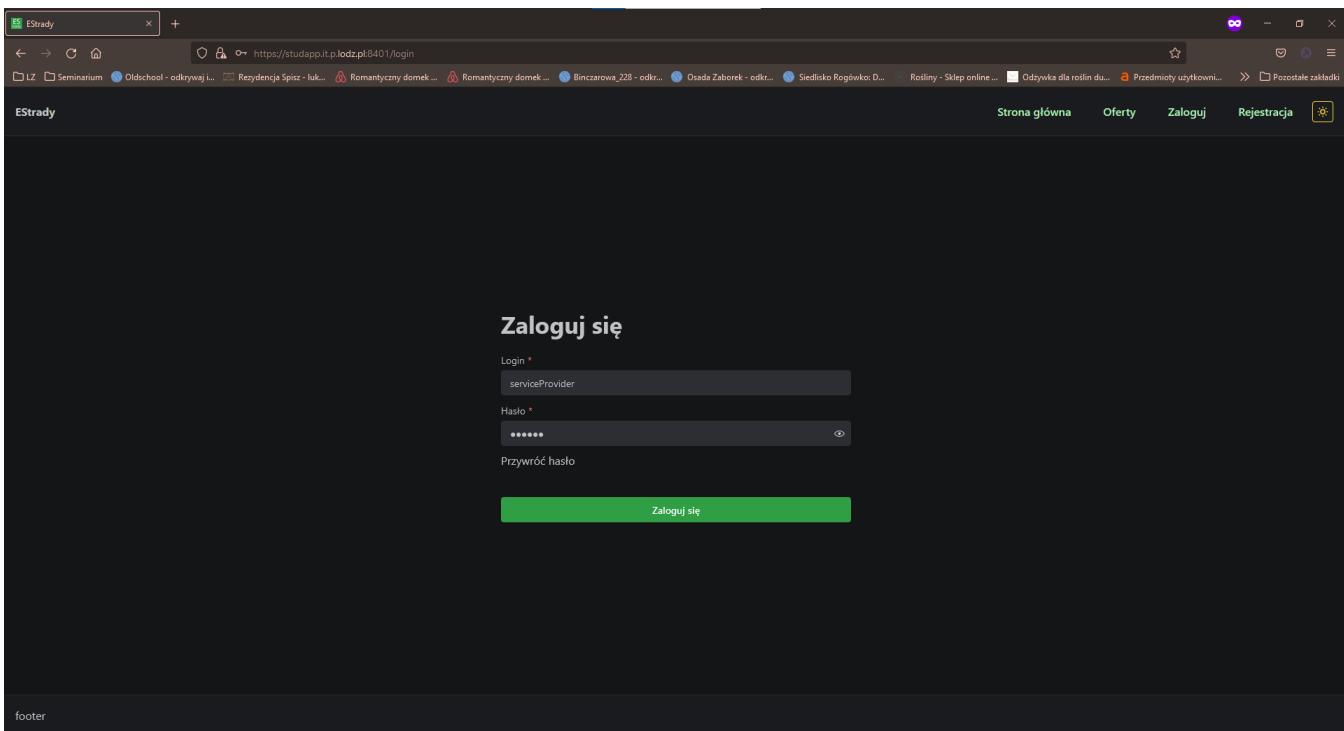


Obraz 20.5

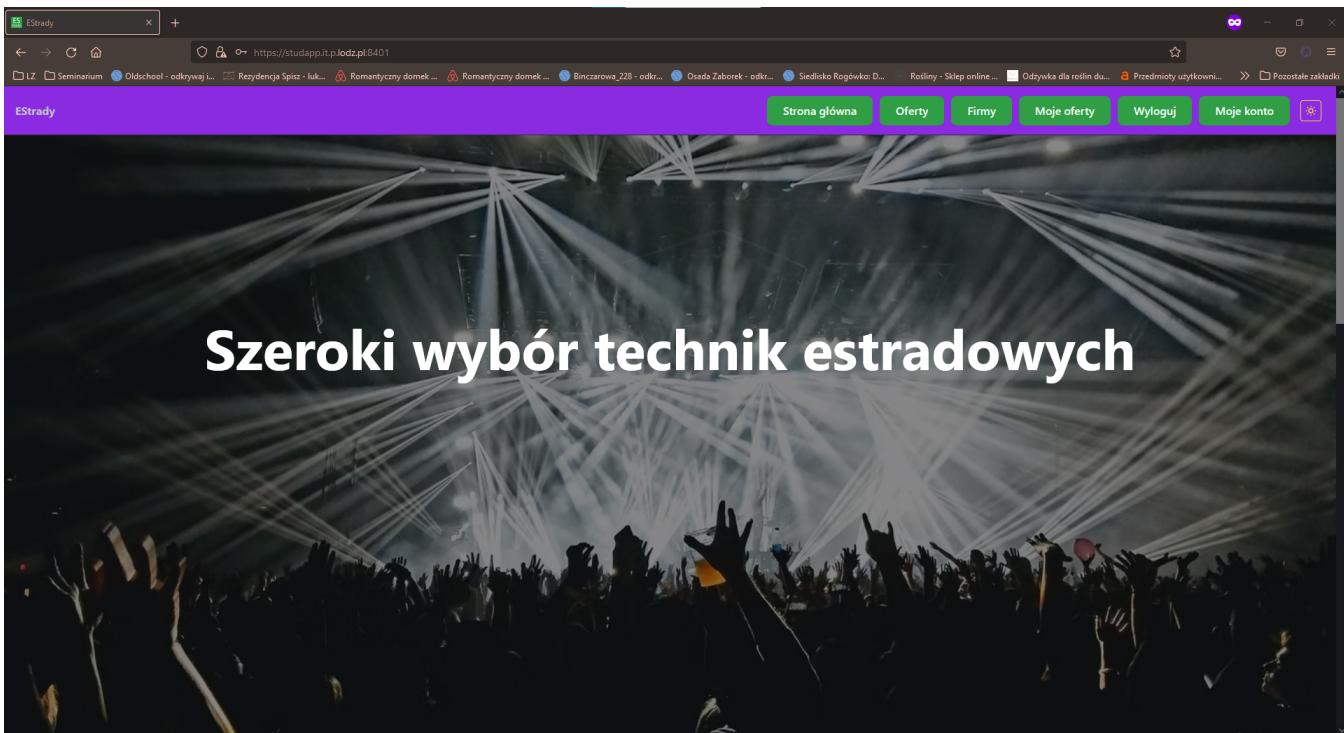
21. Usunięcie wybranej oferty (ustawienie jako niedostępnej do wynajęcia) (MZ.2)



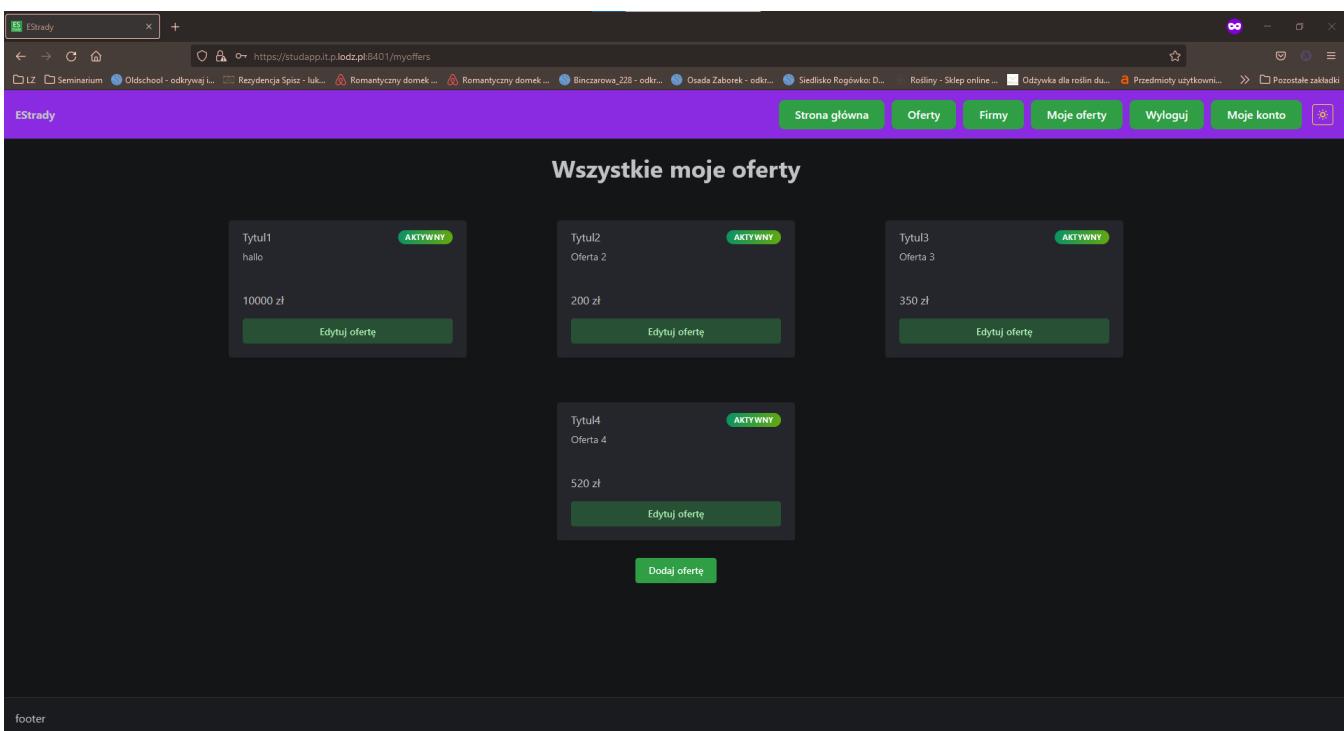
Obraz 21.1



Obraz 21.2

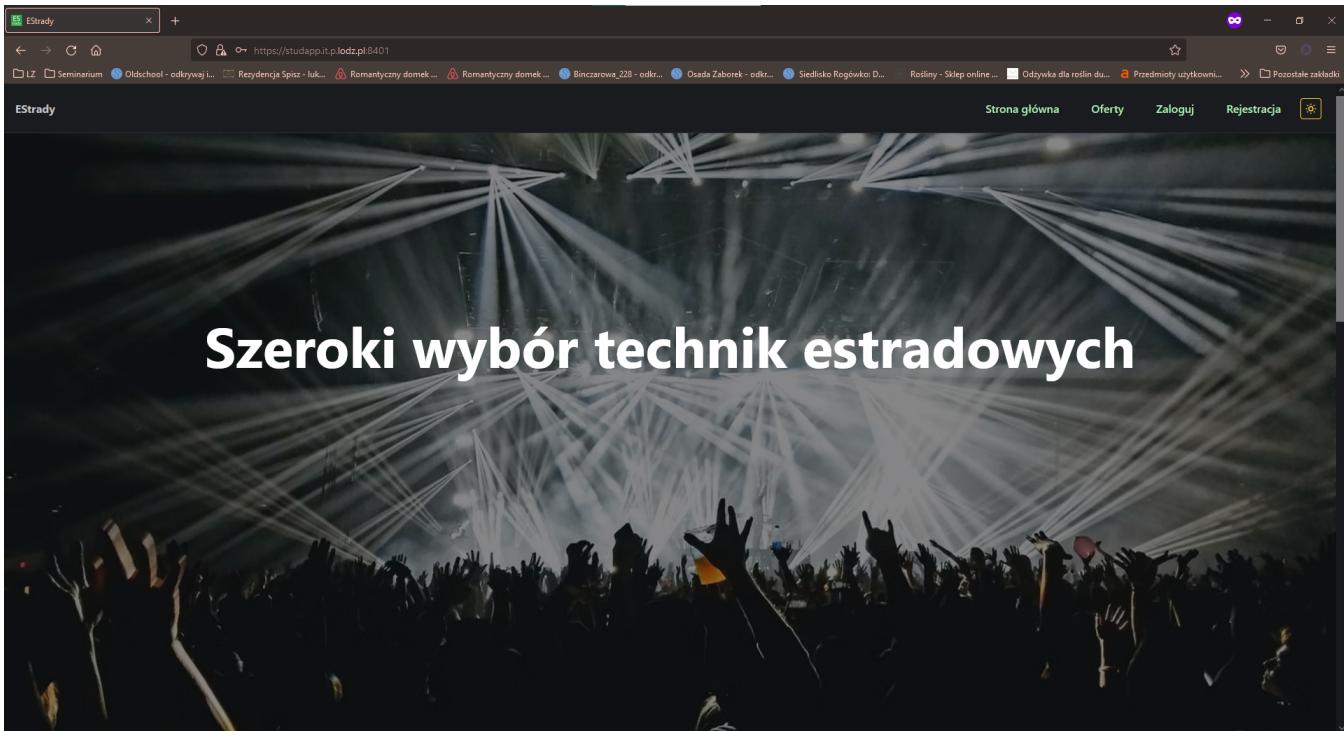


Obraz 21.3

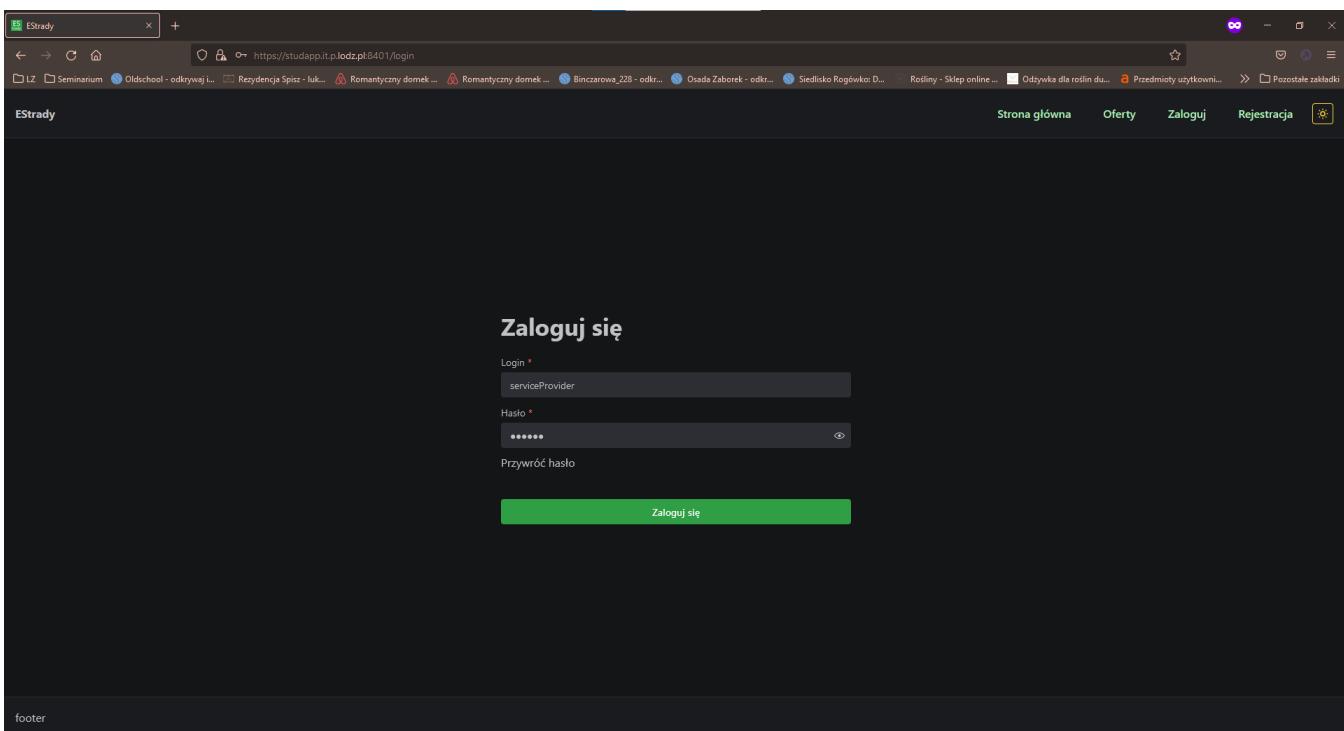


Obraz 21.4

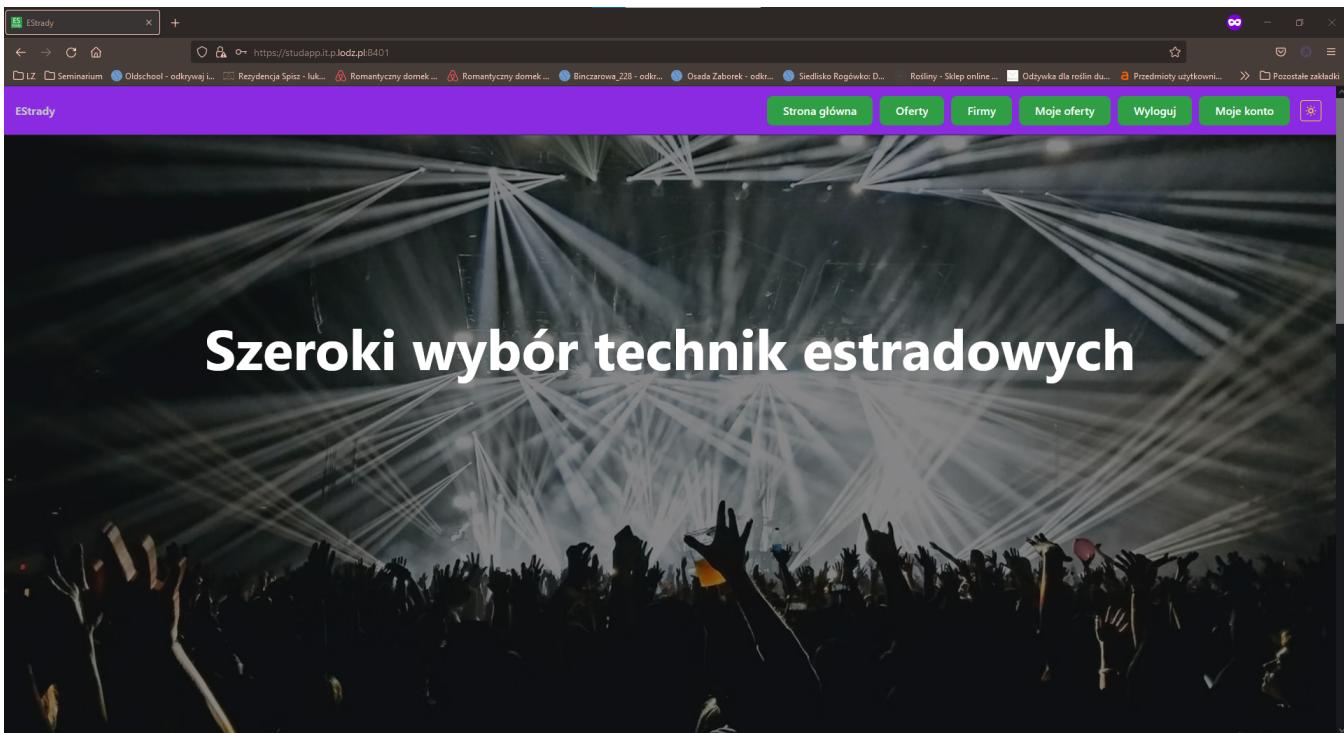
22. Edycja wybranej oferty (MZ.3)



Obraz 22.1



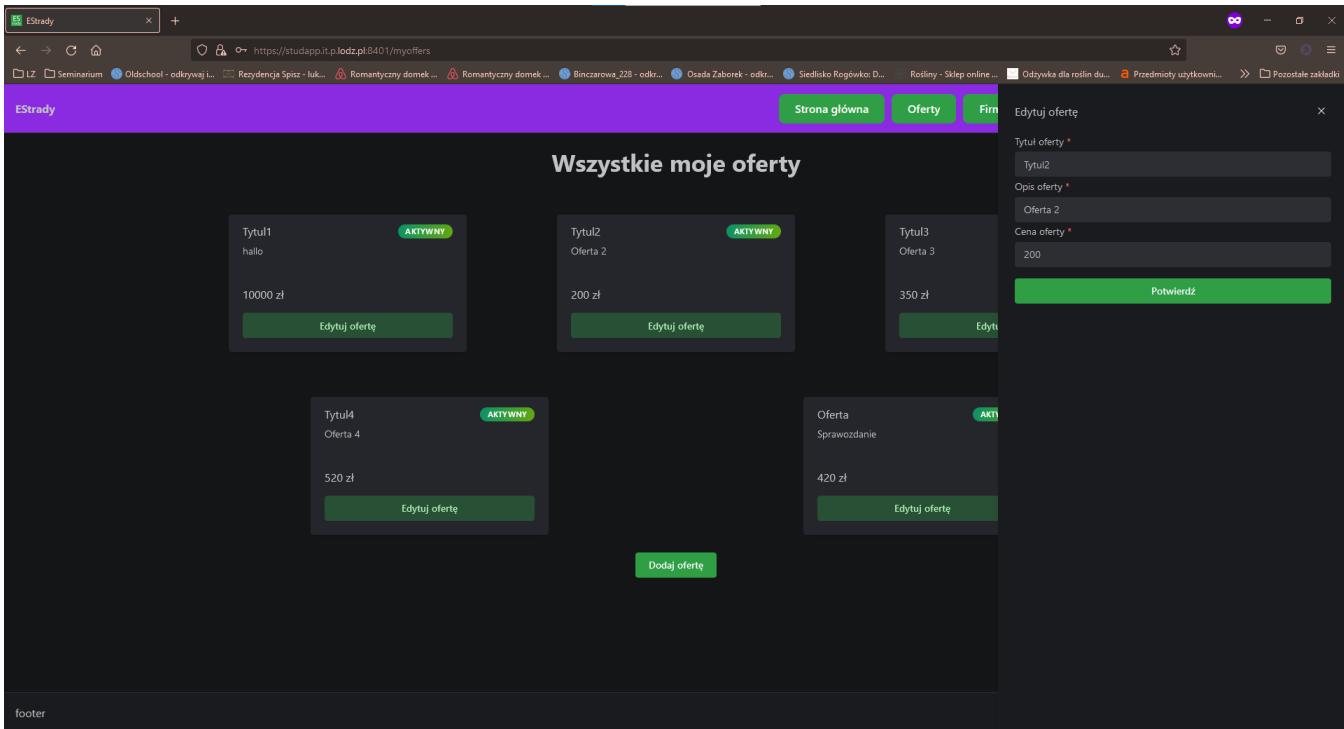
Obraz 22.2



Obraz 22.3

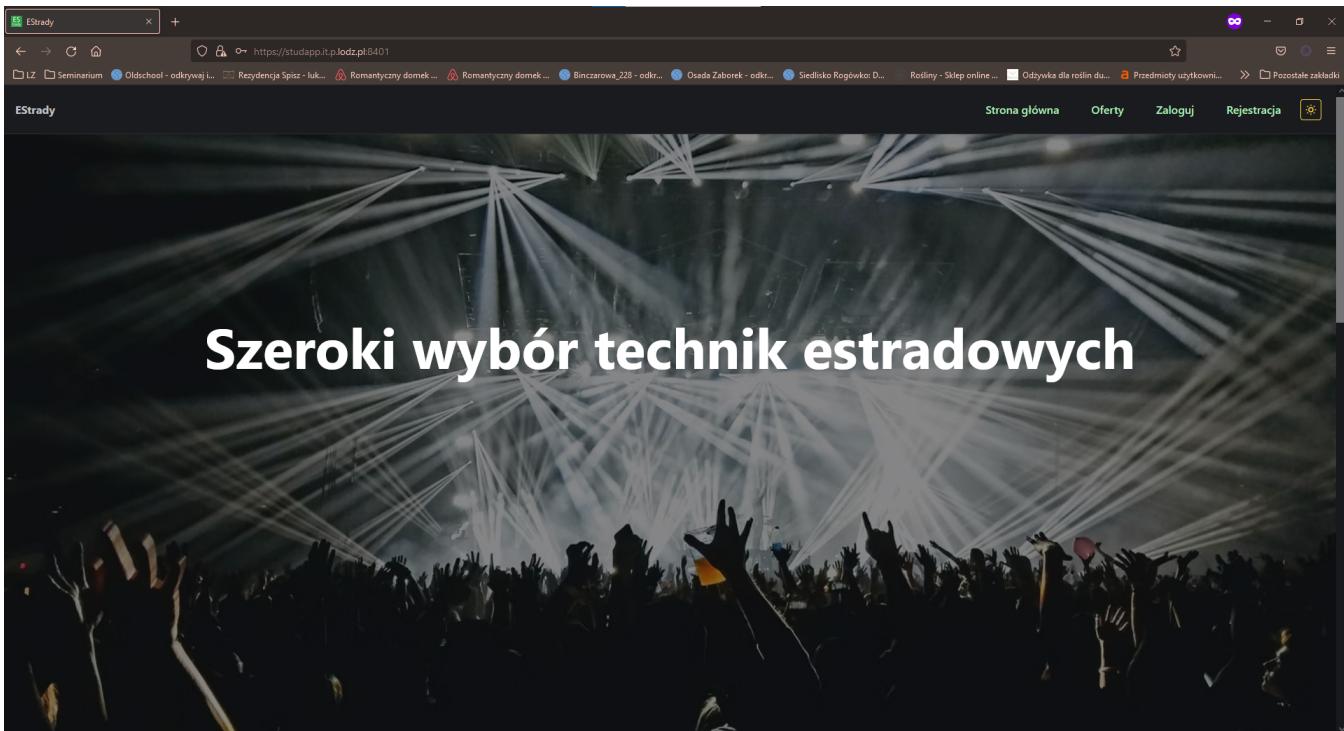
A screenshot of the Estrady website showing the 'Moje oferty' (My offers) section. The page has a purple header with the logo 'Estrady' and the same set of navigation links as the homepage. The main content area is titled 'Wszystkie moje oferty' (All my offers). It displays four offer cards, each with a green 'AKTYWNY' (Active) button. The first card: 'Tytuł1' (Title 1), 'hallo', '10000 zł', 'Edytuj ofertę' (Edit offer). The second card: 'Tytuł2' (Title 2), 'Oferta 2', '200 zł', 'Edytuj ofertę' (Edit offer). The third card: 'Tytuł3' (Title 3), 'Oferta 3', '350 zł', 'Edytuj ofertę' (Edit offer). The fourth card: 'Tytuł4' (Title 4), 'Oferta 4', '520 zł', 'Edytuj ofertę' (Edit offer). At the bottom of the list is a green 'Dodać ofertę' (Add offer) button. A small 'Footer' link is visible at the very bottom left of the page.

Obraz 22.4

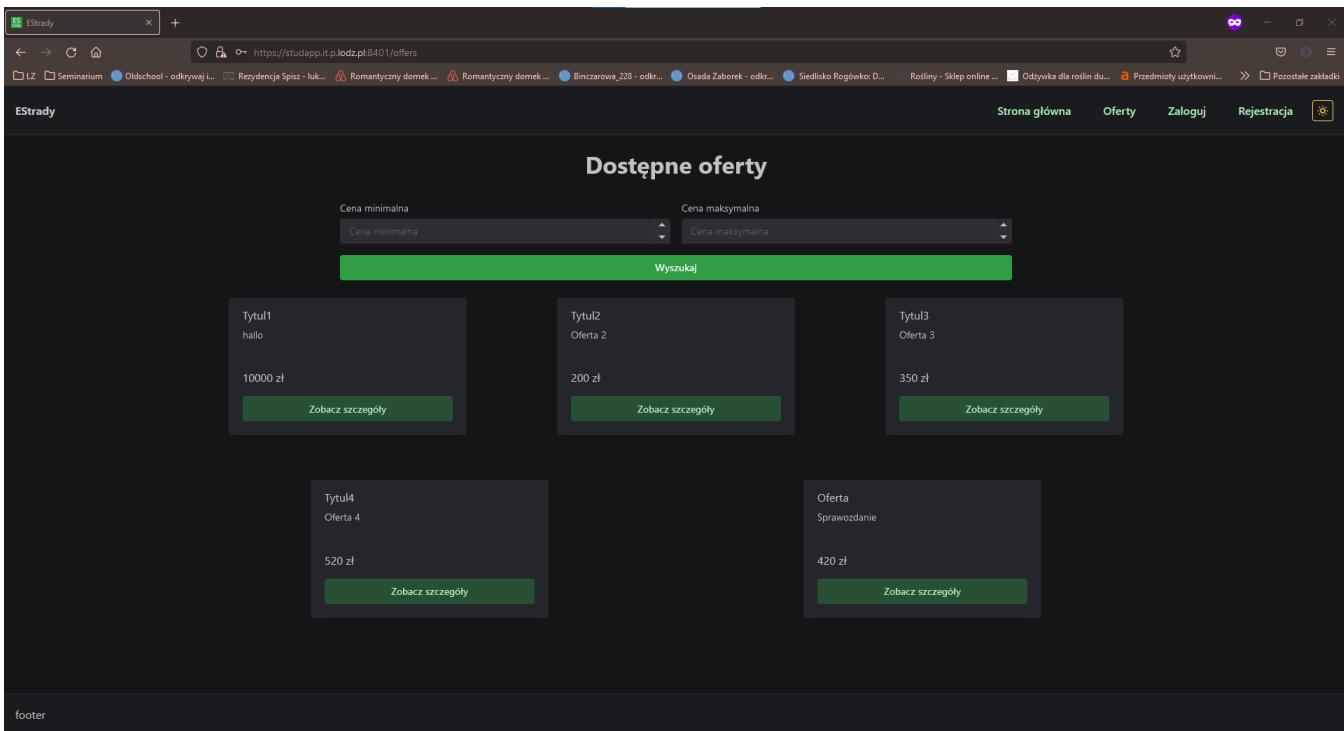


Obraz 22.5

23. Przeglądanie profili firm wynajmujących usługi (MO.1)

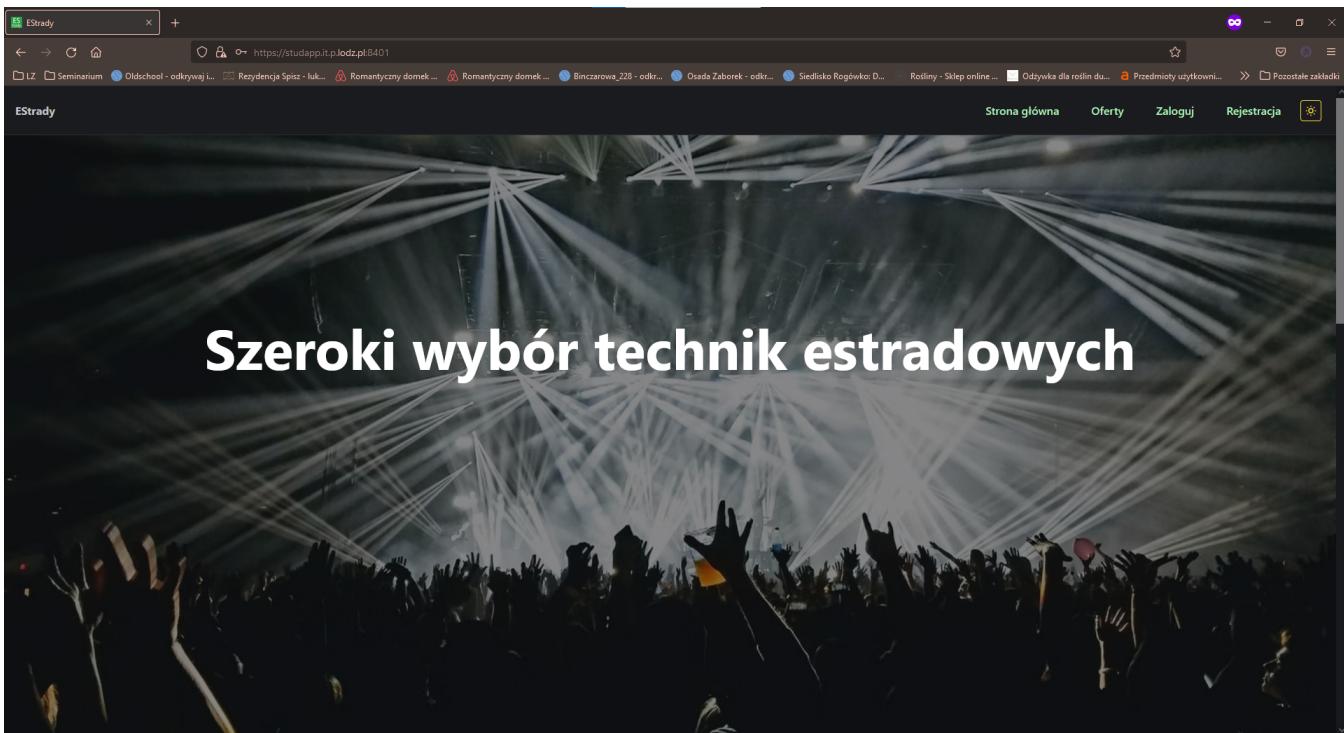


Obraz 23.1



Obraz 23.2

24. Przeglądanie aktualnych ofert firm wynajmujących usługi (MO.2)



Obraz 24.1

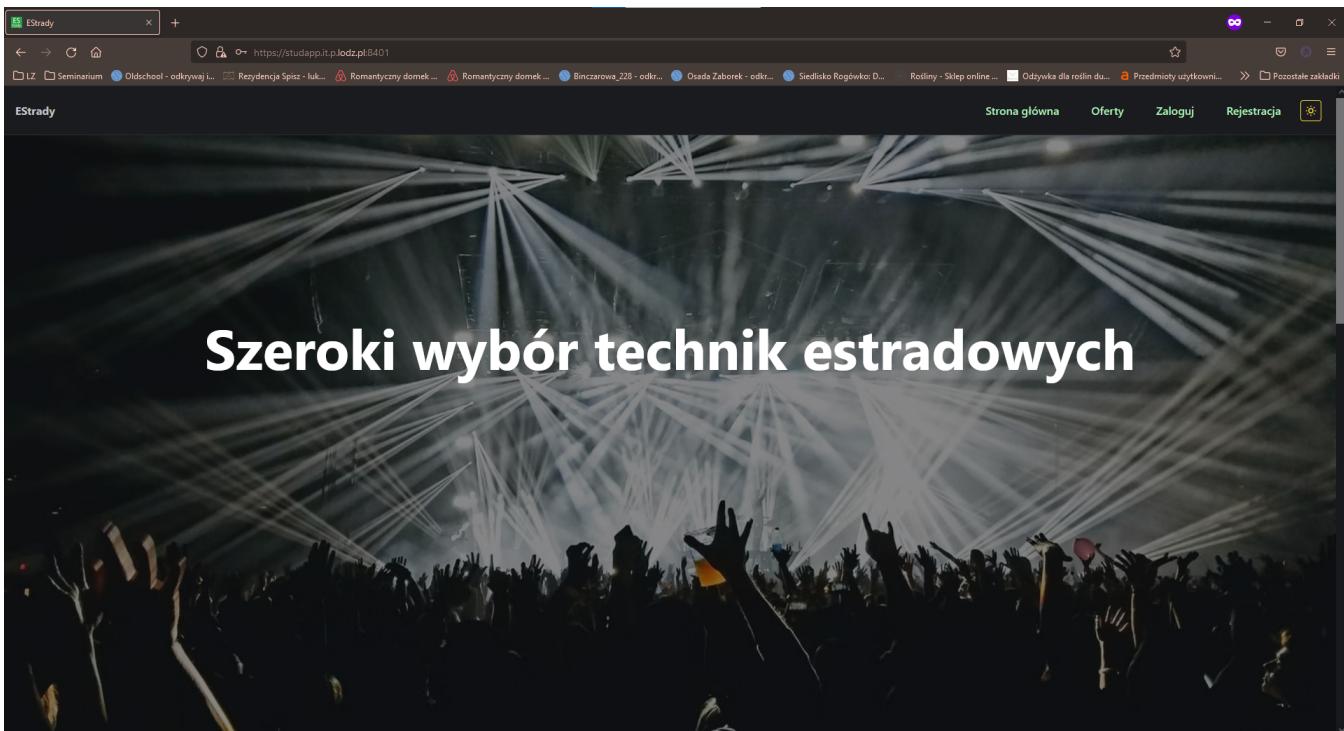
The screenshot shows a web browser window for the Estrady website. At the top, there is a navigation bar with links like 'Strona główna', 'Oferty', 'Zaloguj', and 'Rejestracja'. Below the navigation, a search bar is displayed with two dropdown menus for 'Cena minimalna' and 'Cena maksymalna', both currently set to 'Cena minimalna'. A green 'Wyszukaj' button is positioned below the dropdowns. Below the search bar, there are four cards representing different offers:

- Tytuł1**
hallo
10000 zł
[Zobacz szczegóły](#)
- Tytuł2**
Oferta 2
200 zł
[Zobacz szczegóły](#)
- Tytuł3**
Oferta 3
350 zł
[Zobacz szczegóły](#)
- Tytuł4**
Oferta 4
520 zł
[Zobacz szczegóły](#)
- Oferta**
Sprawozdanie
420 zł
[Zobacz szczegóły](#)

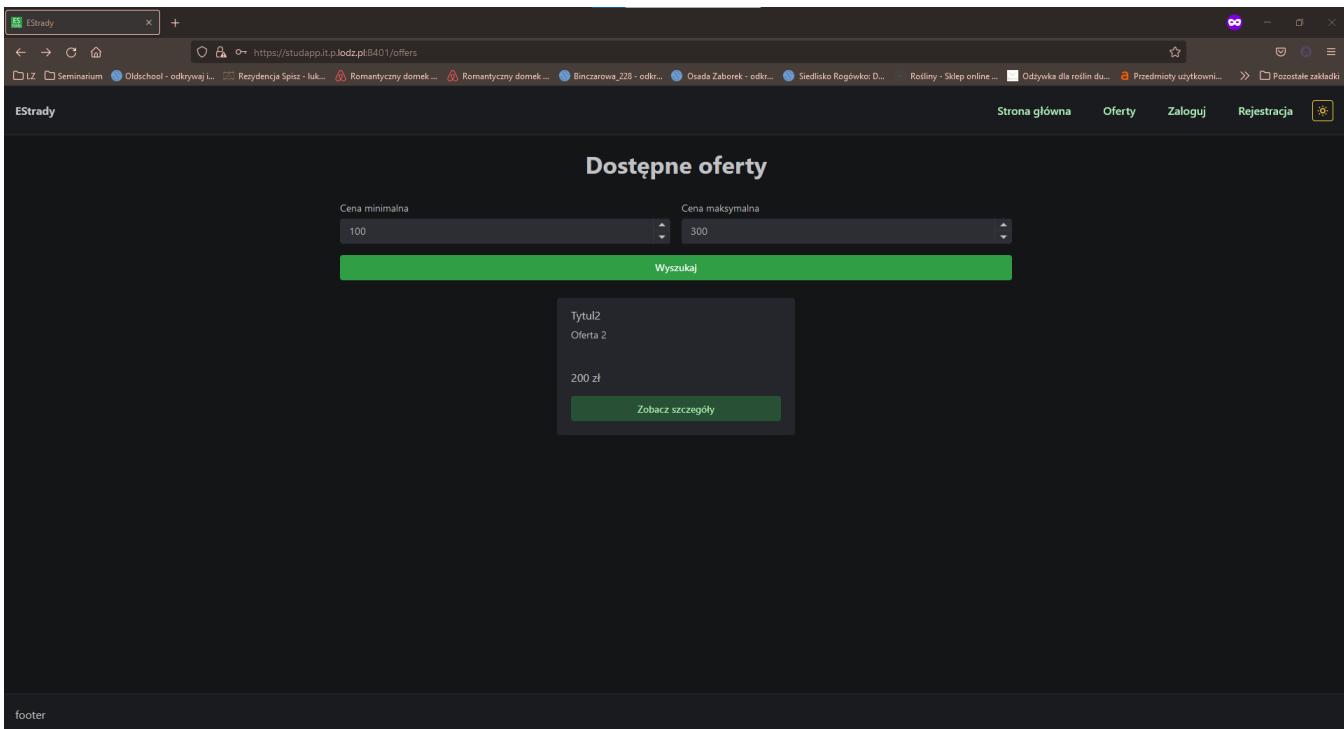
At the bottom left of the page, the word 'footer' is visible.

Obraz 24.2

25. Filtrowanie przeglądanych ofert poprzez podanie ceny minimalnej i maksymalnej (MO.3)

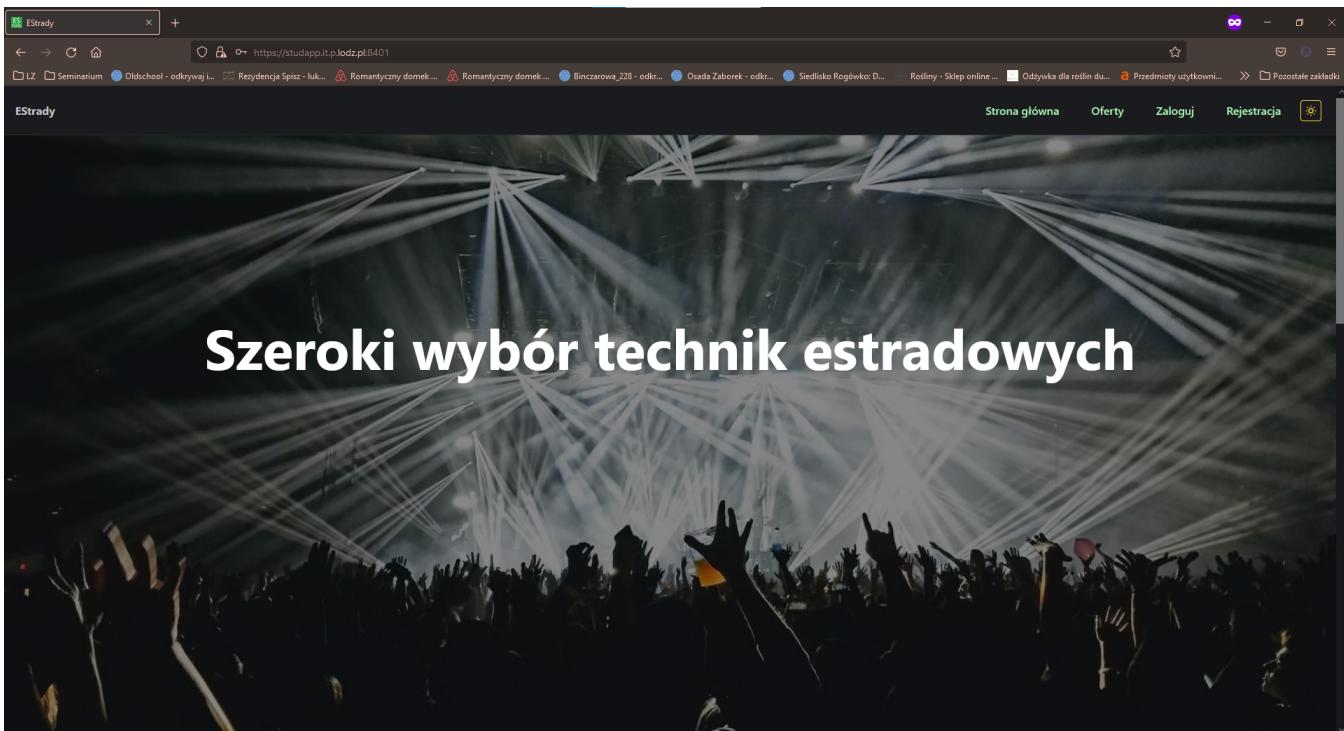


Obraz 25.1

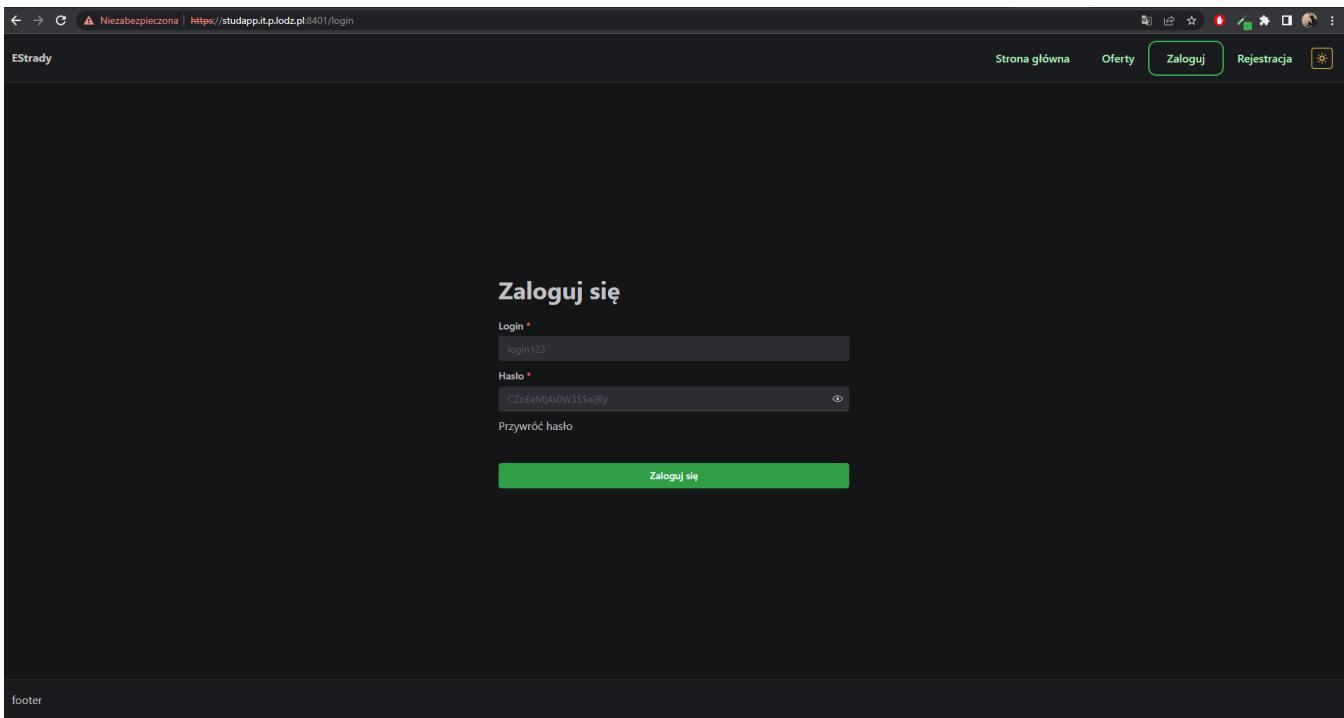


Obraz 25.2

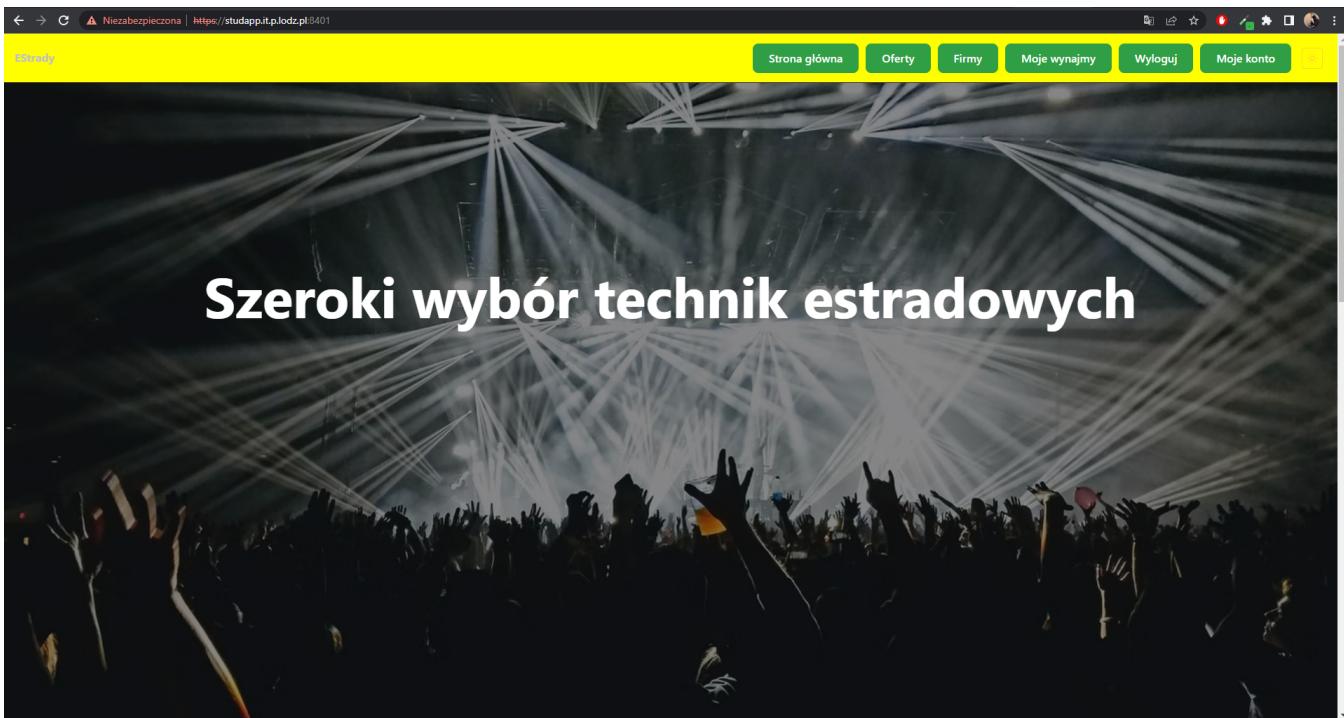
26. Wynajęcie konkretnej oferty firmy wynajmującej usługi (MW.1)



Obraz 26.1



Obraz 26.2



Obraz 26.3

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/offers

Estrady

Strona główna Oferty Firmy Moje wynajmy Wyloguj Moje konto

Dostępne oferty

Cena minimalna Cena maksymalna

Wyszukaj

Tytuł1
hallo
10000 zł [Zobacz szczegóły](#)

Tytuł2
Oferta 2
200 zł [Zobacz szczegóły](#)

Tytuł3
Oferta 3
350 zł [Zobacz szczegóły](#)

Tytuł4
Oferta 4
520 zł [Zobacz szczegóły](#)

Oferta
Sprawozdanie
420 zł [Zobacz szczegóły](#)

footer
https://studapp.it.p.lodz.pl:8401/offers

Obraz 26.4

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/offers

Informacje o estradzie

Tytuł1
hallo
Cena: 10000zł
Firma oferująca usługę: service
[Zobacz szczegóły firmy](#)

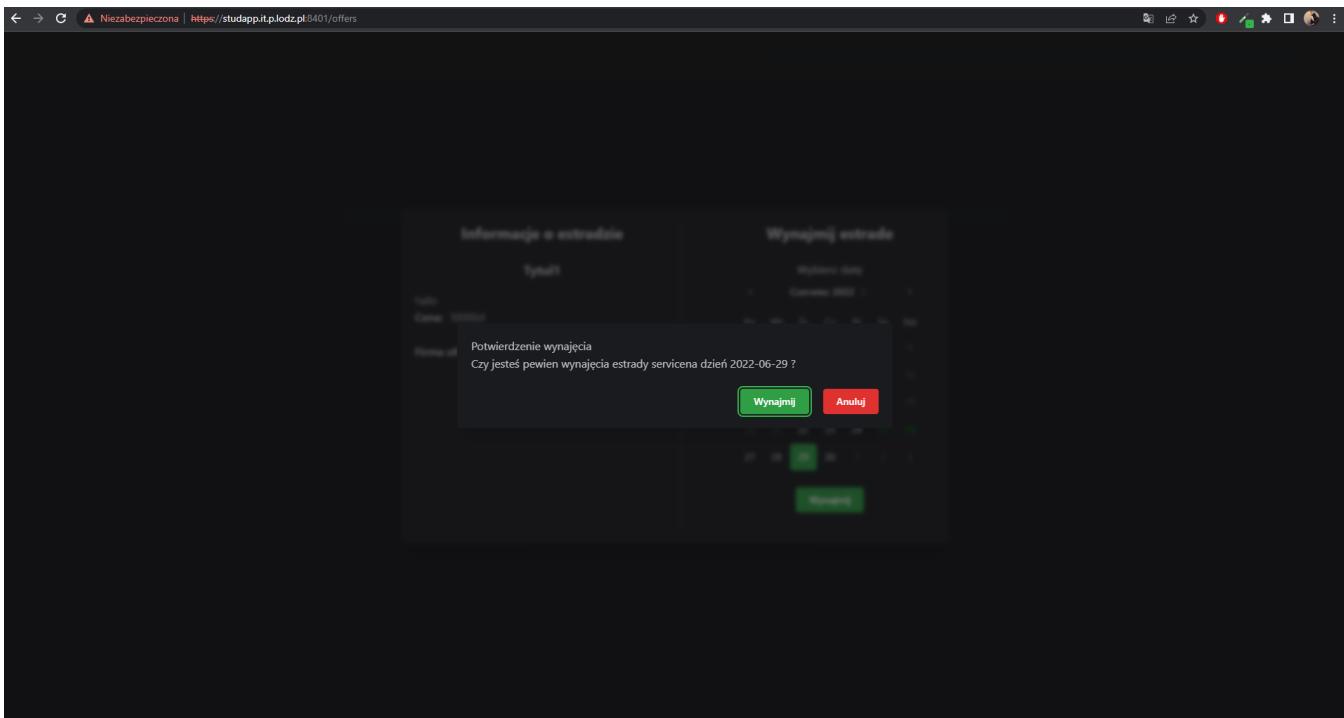
Wynajmij estrade

Wybierz datę
Czerwiec 2022

| Pn | Wt | Śr | Cz | Pt | So | Nd |
|----|----|----|----|----|----|----|
| 30 | 31 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 1 | 2 | 3 |

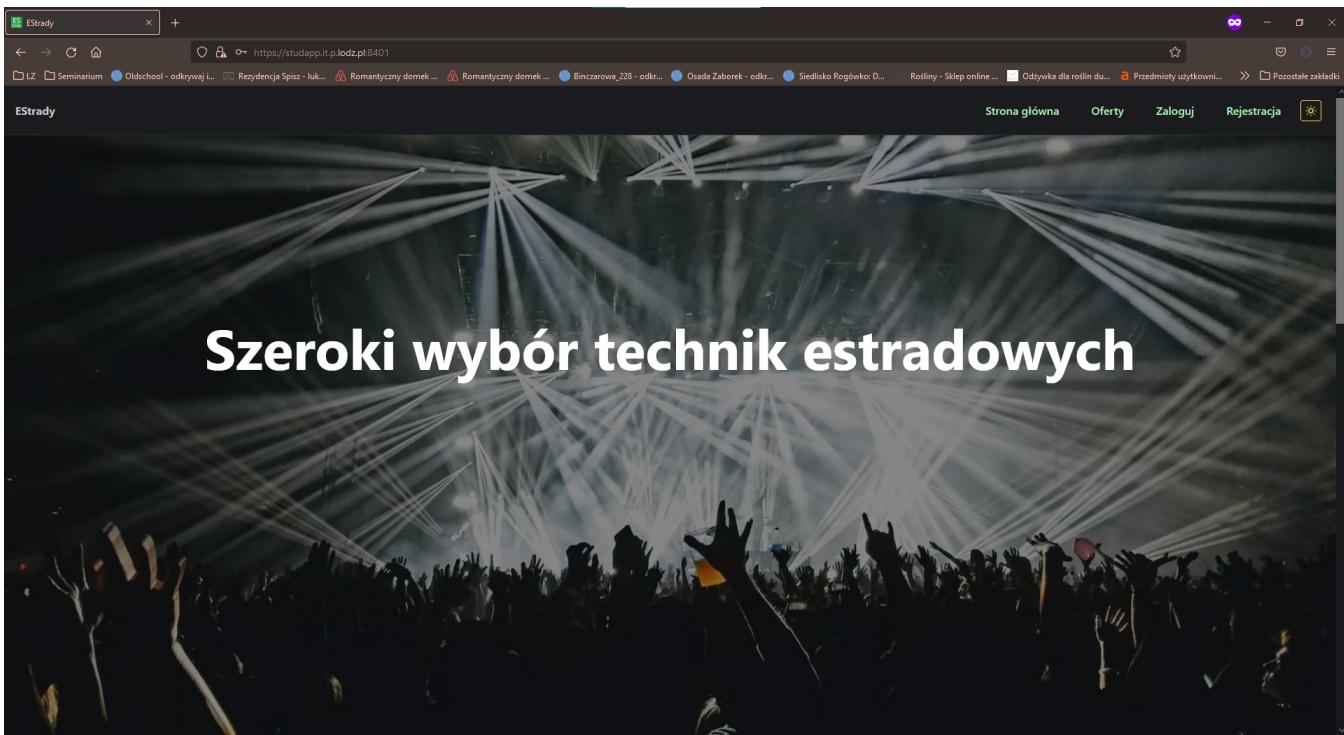
[Wynajmi](#)

Obraz 26.5

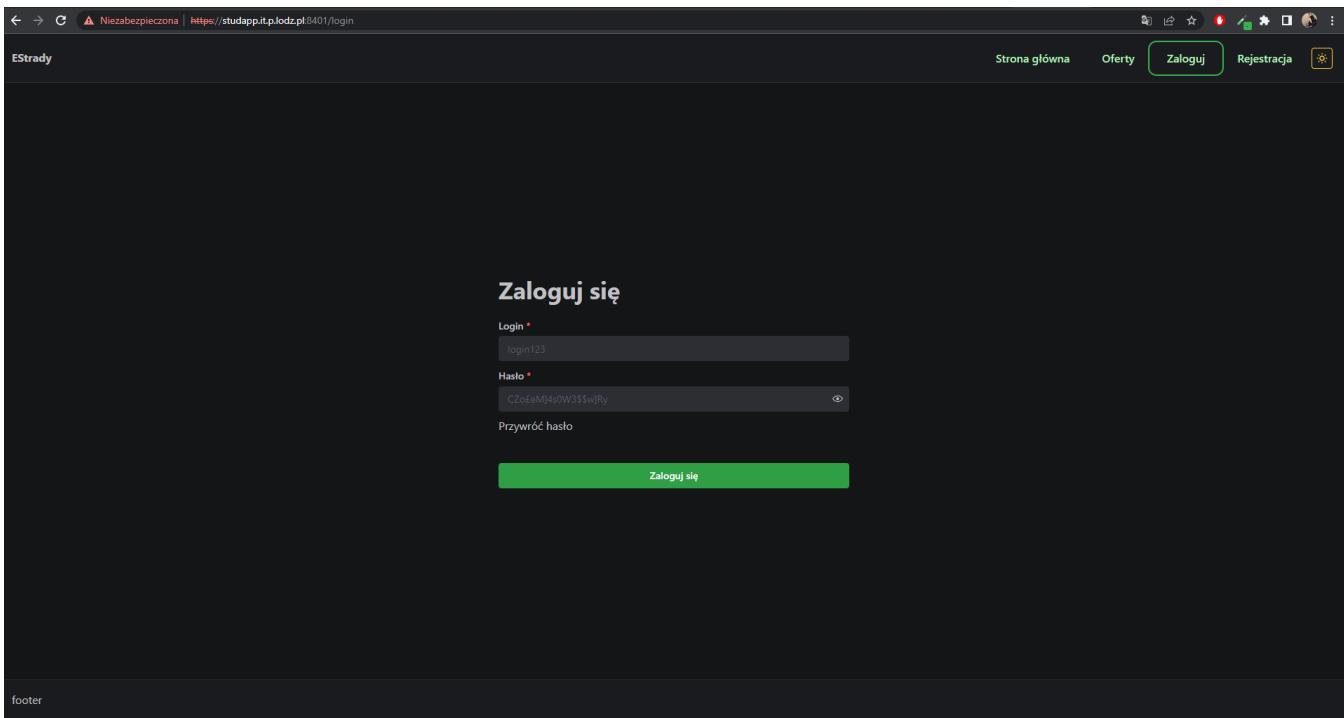


Obraz 26.7

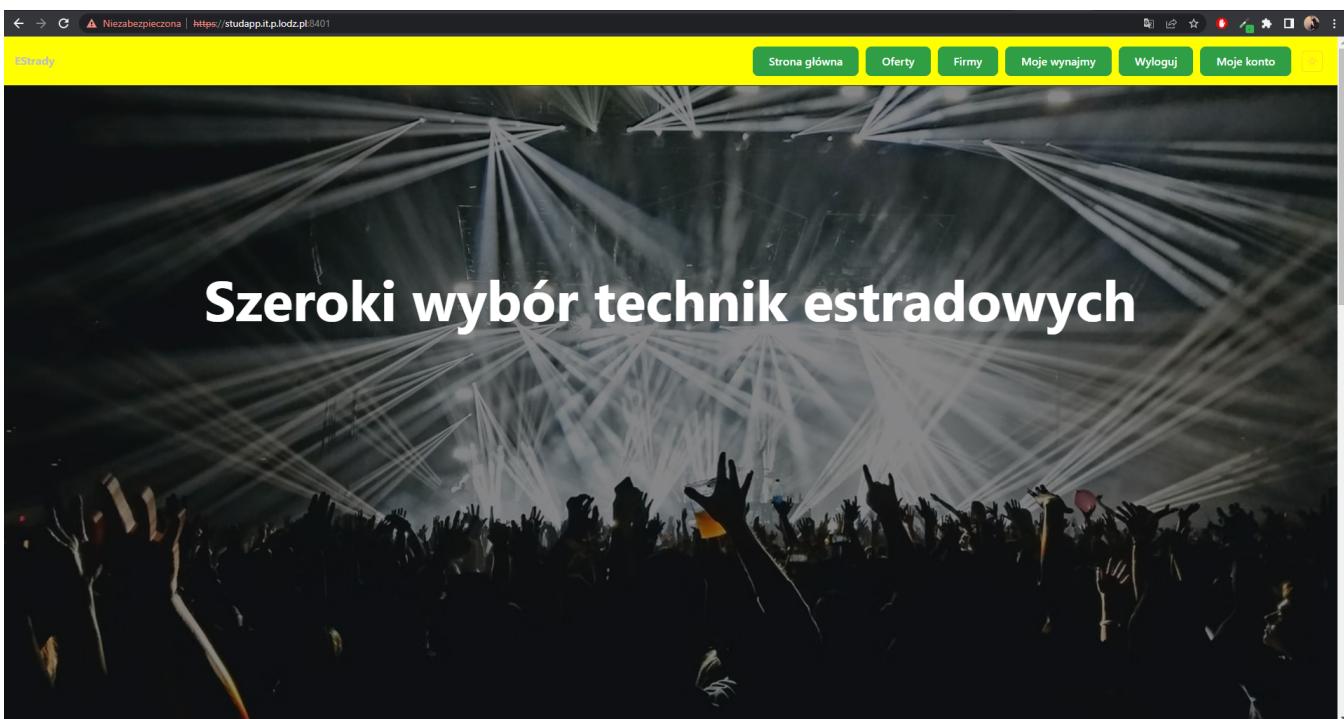
27. Ocena firmy wynajmującej usługi (MW.2)



Obraz 27.1



Obraz 27.2



Obraz 27.3

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/myRents

Strony Strona główna Oferty Firmy Moje wynajmy Wyloguj Moje konto

Twoje wypożyczenia

| Nazwa oferty | Cena | Data | Nazwa usługodawcy | Oceń |
|--------------|-------|------------|-------------------|------|
| Tytuł1 | 10000 | 2020-03-21 | service | Oceń |
| Tytuł2 | 200 | 2022-10-21 | service | Oceń |
| Tytuł1 | 10000 | 2022-06-29 | service | Oceń |

footer https://studapp.it.p.lodz.pl:8401/myRents

Obraz 27.4

Niezabezpieczona | https://studapp.it.p.lodz.pl:8401/myRents

Strona główna Oferty Firmy Moje wynajmy Wyloguj Moje konto

Wybierz ocenę

Submit

| Nazwa oferty | Cena | Data | Nazwa usługodawcy | Oceń |
|--------------|-------|------------|-------------------|------|
| Tytuł1 | 10000 | 2020-03-21 | service | Oceń |
| Tytuł2 | 200 | 2022-10-21 | service | Oceń |
| Tytuł1 | 10000 | 2022-06-29 | service | Oceń |

footer https://studapp.it.p.lodz.pl:8401/myRents

Obraz 27.5

21 Zmiany - projekt końcowy

W rozdziale tym należy wymienić zmiany wprowadzone do sprawozdania wstępniego i szczegółowego, zwięzłe je charakteryzując. Dla każdej zmiany wymagane jest zamieszczenie odwołania do strony poddanej modyfikacjom.

Poprawione oraz dodane uprawnienia do tabel związanych z modułami MO, MZ oraz MW w rozdziale [4 Użytkownicy bazodanowi](#)

Poprawione parametry w niektórych metodach serwisowych dla modułu MOK oraz dodanie modelu bezpieczeństwa metod komponentów EJB dla modułów MO, MZ oraz MW w Rozdziale [10 Model bezpieczeństwa komponentów EJB/CDI 5 pkt](#)

Poprawione diagramy UML klas komponentów EJB/CDI w rozdziale [9 Diagram UML klas komponentow EJB/CDI](#)

Poprawiony diagram UML obiektów encji w rozdziale [6 Diagram klas encji 1 pkt](#)

Zaktualizowane diagramy sekwencji oraz identyfikacje transakcji baz w rozdziale [7 Diagramy sekwencji oraz identyfikacja transakcji bazodanowych 3 pkt](#)

Zaktualizowanie wykazania transakcji aplikacyjnych w rozdziale [11 Identyfikacja transakcji aplikacyjnych 6 pkt](#)

Wprowadzono poprawki błędów wymienionych podczas etapu szczegółowego rozdziału 12 [zgłasiane wyjątki 4pkt](#) (brak klucznej internacjalizacji w wykazanych wyjątkach biznesowych, brak kompletu wyjątków biznesowych (w szczególności dla wyścigów występujących w modułach funkcjonalnych innych niż MOK, brak kompletu wymaganych wyjątków systemowych zgłaszanych przez kontener EJB)

Zaktualizowanie tabel w bazie danych, skryptu z danymi inicjującymi oraz dodanie tabeli rate_renter_details (potrzebna do relacji "wiele do wielu" między tabelami rate oraz renter_details): [3 Struktury relacyjnej bazy danych 2 pkt](#)

Zaktualizowanie listingów w rozdziale [13 Interfejs użytkownika 3 pkt](#)

Zaktualizowano diagramy użycia oraz scenariusze przypadków użycia [2 Moduły funkcjonalne, aktorzy, przypadki użycia 3 pkt](#)

Zaktualizowano konfigurację kontroli dostępu użytkownika do widoków z interfejsem użytkownika oraz tożsamość użytkowników na role aplikacji w warstwie widoku [14 Zabezpieczenia interfejsu użytkownika 3 pkt](#)