



**Kierunek: Informatyka**

2023/2024

Marcin Król

PRACA INŻYNIERSKA

*System sprzedaży biletów*

Promotor pracy:

mgr inż. Tomasz Gryl

Tarnów, 2024



# Spis treści

Wstęp .....	3
1. Wykorzystane narzędzia i technologie informatyczne .....	4
1.1. Analiza technologiczna .....	4
1.1.1. Aplikacja webowa .....	4
1.1.2. Aplikacja mobilna .....	6
1.1.3. Aplikacja serwerowa .....	8
1.1.4. Baza danych .....	10
1.2. Zabezpieczenia aplikacji internetowych .....	11
1.3. Ochrona danych osobowych w aplikacjach internetowych .....	13
2. Dokumentacja techniczna .....	16
2.1. Zakres systemu .....	16
2.1.1. Warstwa aplikacji webowej .....	16
2.1.2. Warstwa aplikacji mobilnej .....	17
2.1.3. Warstwa aplikacji serwerowej .....	17
2.2. Perspektywa produktu .....	17
2.3. Aktorzy i charakterystyka użytkowników .....	19
2.4. Diagram przypadków użycia .....	20
2.4.1. Sekcja kliencka .....	20
2.4.2. Sekcja administracyjna .....	47
2.5. Wymagania pozafunkcjonalne .....	60
2.6. Opis API .....	62
2.7. Schemat bazy danych .....	68
2.8. Diagram klas .....	69
3. Implementacja systemu .....	71
3.1. Aplikacja webowa .....	71
3.1.1. Interfejs użytkownika .....	71
3.1.2. Interfejs organizatora .....	79
3.2. Aplikacja mobilna .....	85
3.3. Aplikacja serwerowa .....	88
3.4. Najciekawsze fragmenty kodu .....	89
4. Testy .....	101
4.1. Testy jednostkowe .....	101
4.2. Testy integracyjne .....	104
4.3. Testy manualne .....	106
Podsumowanie .....	108

Dodatek A. Szablony dokumentacji technicznej.....	110
Bibliografia .....	112
Spis tabel .....	113
Spis rysunków .....	115
Spis listingów.....	117

# Wstęp

W dzisiejszych czasach globalna cyfryzacja dotyczy niemalże każdego aspektu naszego życia. Skutkiem tego zjawiska w połączeniu z coraz większym zapotrzebowaniem na różnorodne bodźce jest stale rosnąca rola aplikacji mobilnych i platform internetowych m.in. w organizacji wydarzeń oraz dostępie do nich. W tym kontekście, zapotrzebowanie na system umożliwiający sprzedaż biletów nabiera nowego znaczenia.

Celem niniejszej pracy była implementacji oprogramowania umożliwiającego sprzedaż biletów na wydarzenia. Serwis składa się z aplikacji webowej, aplikacji mobilnej przeznaczonej na systemy Android i iOS oraz serwera WWW, komunikującego się z bazą danych. System umożliwia użytkownikom przeglądanie i wyszukiwanie dostępnych wydarzeń w okolicy, wybór biletów oraz dokonywanie płatności. Oprogramowanie umożliwia organizatorom tworzenie i zarządzanie wydarzeniami poprzez interfejs administracyjny. Ponadto system umożliwia analizę danych, dotyczących zainteresowań użytkowników oraz generuje oferty, dostosowane do ich preferencji

W pierwszym rozdziale pracy zaprezentowano koncepcję systemu oraz użyte technologie. Poruszono również kwestię bezpieczeństwa danych oraz ochronę danych osobowych.

W drugim rozdziale, skupiono się na prezentacji diagramu kontekstowego, na którym wyodrębniono kluczowych aktorów korzystających z systemu oraz przedstawiono główne przepływy informacji między nimi. Rozdział przedstawia również scenariusze użycia oraz interakcje poszczególnych aktorów w obrębie głównych modułów tworzonego systemu.

Rozdział trzeci skupia się na przedstawieniu zaimplementowanych ekranów aplikacji webowej oraz mobilnej. W rozdziale tym przedstawione zostały także najistotniejsze fragmenty kodu.

Ostatni rozdział poświęcony jest różnym rodzajom testów, które zostały przeprowadzone dla poszczególnych części systemu w celu podniesienia jego bezpieczeństwa i sprawności działania.

Całość pracy zamyka krótkie podsumowanie.

# 1. Wykorzystane narzędzia i technologie informatyczne

Pierwszy rozdział niniejszej pracy inżynierskiej poświęcony został analizie wykorzystanych narzędzi oraz technologii informatycznych. W ramach tego rozdziału, poruszone zostały także dwie kluczowe kwestie dotyczące zabezpieczeń i ochrony danych w aplikacjach internetowych.

## 1.1. Analiza technologiczna

W tym podrozdziale dokonano wnikliwej analizy technologicznej, która stanowi fundament całego projektu. Przedstawione zostały kluczowe narzędzia i technologie tworzące każdą z warstw systemu.

### 1.1.1. Aplikacja webowa

HTML (ang. *HyperText Markup Language*) to hipertekstowy język znaczników służący do definiowania struktury stron i aplikacji webowych. Język, jak sama nazwa wskazuje, wykorzystuje tzw. znaczniki, które określają elementy tworzące stronę internetową. Wyróżnić możemy dwie główne grupy znaczników – liniowe oraz blokowe. Elementy liniowe zajmują tylko tyle miejsca, ile potrzebują, domyślnie prezentowane są obok siebie, w jednym wierszu. Przykładowymi znacznikami liniowymi są np. `<span>` czy `<a>`. Elementy blokowe natomiast, domyślnie wyświetlane są od nowego wiersza [1]. Odpowiednie reguły mogą być użyte do naganiania lub modyfikowania tych zasad, w tym celu wykorzystuje się technologię kaskadowych arkuszy stylów – CSS (ang. *Cascading Style Sheets*).

Działanie kaskadowych arkuszy stylów polega na przyporządkowaniu reguł do elementów HTML. Te reguły określają sposób wyświetlania zawartości określonych elementów. Reguły CSS składają się z dwóch części: selektora i deklaracji. Zastosowanie CSS pozwala programiście odseparować strukturę od logiki aplikacji, zwiększając tym samym czytelność kodu oraz jego elastyczność. Kaskadowość języka odnosi się do sposobu w jaki stosowane są reguły CSS. Język wykorzystuje hierarchię elementów HTML do określenia, które style zostaną zastosowane. Im bardziej szczegółowy selektor, tym wyższa jego specyficzność i większy priorytet. Na przykład, `id` ma wyższą specyficzność niż klasa, a klasa wyższą niż element. Jeżeli dwie reguły mają taką samą specyficzność, to ta, która występuje później w kodzie, ma wyższy priorytet.

Kolejnym, niemniej ważnym aspektem jest wybór języka programowania. Najbardziej powszechnym w przypadku aplikacji i stron internetowych jest język JavaScript. JavaScript jest językiem wieloparadygmatowym, co przekłada się na dużą elastyczność oraz uniwersalność. Posiada zestaw reguł i protokołów przystosowanych do pracy z tekstem, tablicami, datami czy nawet wyrażeniami regularnymi [2]. Język cechuje się przede wszystkim prostotą, dużą ilością bibliotek producentów trzecich oraz możliwością wykonywania kodu po stronie klienta co znacząco pozwala na odciążenie serwerów, które nie muszą przeznaczać zasobów np. na walidację poprawności wypełnionego przez klienta formularza. To właśnie m.in. te cechy powodują, że język ten jest tak popularny, a co za tym idzie, również stale rozwijany. JavaScript posiada jeszcze jedną istotną cechę – jest językiem typowanym dynamicznie. Oznacza to, że deklaracje zmiennych mogą zostać zmienione w trakcie funkcjonowania i nie są sprawdzane przed uruchomieniem programu. Dzięki temu, że zmienne nie wymagają jawniej deklaracji typów, proces wytwarzania kodu jest sprawniejszy. Niestety takie ułatwienie może powodować trudności w śledzeniu wartości zmiennych czy późniejsze szukanie błędów. Brak jawnego typowania może również powodować przekazywanie niepoprawnych danych metodom, co może skutkować nieprawidłowym zachowaniem programu lub potencjalnymi lukami w zabezpieczeniach. Aby rozwiązać wspomniane problemy zespół inżynierów firmy Microsoft stworzył i wprowadził na rynek technologię TypeScript.

TypeScript, będący nadzbiorem JavaScript, wprowadza silne typowanie oraz wiele innych nowoczesnych funkcjonalności, znacząco podnoszących czytelność, skalowalność i bezpieczeństwo kodu. Aplikacja webowa, która stanowi jedną z trzech głównych części projektowanego w niniejszej pracy systemu, została oparta właśnie o ten język programowania oraz technologię Angular.

W książce „Angular 2 programowanie z użyciem języka TypeScript”, autorzy wspominają jak w roku 2015 firma Google ogłosiła rozwój Angular 2, a rekomendowanym językiem dla tej technologii był właśnie język TypeScript [3]. Autorzy stwierdzili, że jest to najbardziej wydajny sposób rozwijania średnich i dużych aplikacji. Dodatkowo wymieniają szereg zalet takiego połączenia. Zaletami są np. modularyzacja, wsparcie nawigacji, odpowiednia separacja logiki od wyglądu i struktury projektowanych interfejsów, cykl życia komponentów czy też dostęp do wielu dodatkowych narzędzi programistycznych.

Angular jest tak naprawdę zestawem gotowych do wykorzystania narzędzi i rozwiązań, co znacząco przyspiesza proces szybkiego tworzenia dynamicznych interfejsów użytkownika.

Angular jest stosunkowo nową technologią, pierwsza wersja trafiła na rynek w maju 2016 roku. Od tamtego czasu technologia ciągle ewoluuje i rozwija się pod nadzorem firmy Google, będącej jedną z najbardziej wpływowych firm w dziedzinie technologii informatycznej.

Jak już wspomniano technologia Angular promuje modułowość i komponentową strukturę, co ułatwia podział aplikacji na mniejsze, autonomiczne części. To przekłada się na łatwiejsze utrzymanie i rozwijanie projektu. Technologia wykorzystuje programowanie reaktywne, które pozwala na efektywne zarządzanie zmianami w interfejsie użytkownika. Dzięki temu, aplikacje stworzone w Angular są dynamiczne i szybko reagują na akcje użytkownika.

Wspomniano również o dodatkowych narzędziach, które oferuje Angular. Jest to m.in. Angular CLI (ang. *Command Line Interface*). Narzędzie, które również zostało w projekcie wykorzystane. Dostarcza ono zestaw komend, pozwalających na generowanie nowych projektów czy podstawowych struktur, takich jak komponenty, moduły czy usługi według gotowych schematów, tzw. Scaffolding. Prowadzi to do zachowania odpowiedniej konwencji nazewnictwnej i strukturalnej, co sprawia, że projekt jest bardziej przejrzysty i łatwiejszy w rozbudowie [4].

Do zalet tej technologii należy również tzw. cykl życia komponentów. Odnosi się on do sekwencji zdarzeń i metod, które są wywoływanie podczas tworzenia, aktualizacji i destrukcji komponentów. Dzięki cyklowi życia, programista ma kontrolę nad różnymi aspektami działania komponentów, takimi jak inicjalizacja, czy też manipulacja obiektowym modelem dokumentu (ang. *Document Object Model*). Najczęściej używanymi metodami cyklu życia są np. *ngOnChanges* – metoda pozwalająca na reagowanie na zmiany wejściowe przed ich przetworzeniem, *ngOnInit* – metoda cyklu życia, która zostanie wywołana po konstrukcji komponentu, wykorzystywana jest często do inicjalizacji danych, np. pobranie ich z serwera, *ngOnDestroy* – wywoływana przed zniszczeniem komponentu. Wykorzystywana do czyszczenia zasobów, takich jak subskrypcje czy nasłuchiwacze. Zarówno te jak i inne metody cyklu życia zostały wykorzystane w niniejszej pracy.

### **1.1.2. Aplikacja mobilna**

Komplementarną częścią opracowanego systemu jest aplikacja mobilna, stanowiąca niezwykle istotny element umożliwiający dostęp do elementów platformy na urządzeniach mobilnych. W celu efektywnego stworzenia tej aplikacji, zdecydowano się na wykorzystanie technologii React Native.

React Native został wydany w 2015 roku przez firmę Facebook. Głównym założeniem tej technologii było dostarczenie deweloperom narzędzia, umożliwiającego tworzenie aplikacji opartych o JavaScript i późniejszą komplikację do natywnego kodu dla danej platformy np. Android czy iOS. Takie rozwiązanie przekłada się na wysoką wydajność aplikacji, która jest porównywalna z tym, co można uzyskać przy użyciu natywnych języków programowania. React Native jest bardzo popularnym zestawem narzędzi i podobnie jak Angular posiada ogromną społeczność deweloperów, co przekłada się na dostęp do licznych bibliotek, narzędzi i dokumentacji.

Technologia oferuje także dwie istotne funkcje, które okazały się być niezwykle pomocne w procesie tworzenia aplikacji mobilnej. Są to tzw. Hot Reloading oraz Live Reloading. Pozwalają one na odświeżanie zmian dokonanych w kodzie w czasie rzeczywistym na ekranie urządzenia lub emulatora. Sprawia to, że proces tworzenia aplikacji oraz manualnego sprawdzania efektów implementacji jest niezwykle szybki.

Hot Reloading umożliwia dynamiczne wprowadzanie zmian w kodzie oraz natychmiastowe obserwowanie efektów na urządzeniu lub emulatorze, zachowując przy tym aktualny stan aplikacji. Jest to szczególnie przydatne przy obserwowaniu zmian przy wprowadzaniu zmian w interfejsie użytkownika lub wykrywaniu błędów.

Live Reloading działa na podobnej zasadzie, z tą różnicą, że automatycznie ładuje aplikację do pamięci za każdym razem, gdy zostaną zapisane jakiekolwiek zmiany w kodzie. Choć pozwala to na dokładniejsze testowanie i wykrywanie błędów, jest to jednak mniej wydajne rozwiązanie, ponieważ ładowanie aplikacji do pamięci zabiera więcej czasu [5].

W przypadku React Native, podobnie jak w zestawie narzędzi Angular występuje pojęcie cyklu życia komponentów. O cyklu życia w przypadku komponentów klasowych w React Native odnosi się on do sekwencji zdarzeń i metod, które są wywoływanie podczas tworzenia, aktualizacji czy usuwania komponentów. Dzięki cykowi życia, deweloper uzyskuje lepszą kontrolę nad różnymi aspektami tworzonych komponentów. Takimi aspektami są np. inicjalizacja, aktualizacja stanu obiektów czy obsługa zdarzeń. Przykładowymi metodami związanymi z cyklem życia są np. *componentDidMount*, *componentDidUpdate* czy *componentWillUnmount*.

Kluczowymi elementami w cyklu życia komponentu w React Native są np. Montowanie (ang. *Mounting*): *componentDidMount* – metoda wywoływana po dodaniu komponentu do drzewa DOM. Aktualizacja (ang. *Updating*): *componentDidUpdate* – metoda wywoływana po aktualizacji stanu lub właściwości komponentu. Odmontowanie (ang. *Unmounting*):

*componentWillUnmount* – metoda wywoływana przed usunięciem komponentu z drzewa DOM. Błąd (ang. *Error Handling*): *componentDidCatch* – metoda wywoływana w przypadku wystąpienia błędu w komponencie [6].

W przypadku komponentów funkcyjnych, również występuje pojęcie cyklu życia. Zastosowanie tzw. hooków pozwala na emulację niektórych aspektów cyklu życia. Hooki to funkcje, które pozwalają na korzystanie z funkcji stanu, efektów i referencji w komponentach funkcyjnych, bez potrzeby korzystania z klasowych komponentów. Powoduje to skrócenie i uproszczenie komponentów, wpływając na zwiększenie wydajności i elastyczności kodu [7].

Tak jak w przypadku aplikacji webowej, w React Native również wykorzystywane są kluczowe technologie, które pozwalają zdefiniować podstawową strukturę i wygląd interfejsu użytkownika. Takimi elementami są wspomniane już języki programowania – JavaScript oraz TypeScript, które stanowią podstawę logiki działania aplikacji. W celu określenia wyglądu i stylu komponentów, React Native korzysta z własnego systemu właściwości i wartości podobnego składnią do technologii CSS. Najpopularniejszym sposobem tworzenia wyglądu poszczególnych elementów na ekranie w React Native jest użycie obiektów JavaScript. Ponadto, React Native umożliwia korzystanie z natywnych funkcji urządzeń za pomocą tzw. Nattywnych Modułów (ang. *Native Modules*). Pozwalają one na integrację z różnymi funkcjonalnościami, takimi jak aparat, lokalizacja, GPS (ang. *Global Positioning System*) czy powiadomienia.

### **1.1.3. Aplikacja serwerowa**

Aplikacja serwerowa, oparta została na środowisku Node.js. Pełni kluczową rolę w zarządzaniu logiką biznesową oraz komunikacją między warstwą prezentacji, a bazą danych.

Node.js jest środowiskiem, które umożliwia wykonywanie kodu JavaScript. Sam został opublikowany w roku 2009. Cieszy się ogromną popularnością i wsparciem. Głównym celem tej technologii było umożliwienie łatwego tworzenia szybkich i wysoko skalowalnych aplikacji sieciowych oraz obsługi wielu równoległych połączeń sieciowych [9]. Obecnie istnieje wiele środowisk uruchomieniowych do obsługi równoległych połączeń sieciowych, takich jak Deno, które skupia się głównie na bezpieczeństwie, prostocie oraz składni TypeScript. Istnieją również języki programowania takie jak Go, który znakomicie radzi sobie z obsługą wielu równoległych połączeń. Takim językiem jest również Python, który dzięki swojej uniwersalności zyskał ogromną popularność i według zestawienia Statista, zajmuje trzecie miejsce pod względem wykorzystania [8].

Pomimo dostępności wielu konkurencyjnych rozwiązań w niniejszej pracy zdecydowano się jednak na użycie środowiska Node.js. Wynika to m.in. z jego natywnego wsparcia przetwarzania asynchronicznego, co pozwala na obsługę wielu równoległych operacji wejścia/wyjścia bez konieczności tworzenia wielu wątków. Node.js jest bardzo wydajny w obsłudze operacji, takich jak obsługa żądań HTTP (ang. *Hypertext Transfer Protocol*) czy operacji na bazach danych. Równie istotną zaletą jest wspomniana już skalowalność, co czyni go świetnym wyborem do tworzenia wysokowydajnych aplikacji, zwłaszcza w środowiskach, gdzie liczba równoległych połączeń ma kluczowe znaczenie. Podobnie jak Angular Node.js posiada rozbudowany ekosystem modułów oraz gotowych rozwiązań, co znacząco przyspiesza implementację. Dodatkowo Node.js po stronie serwera używa maszyny wirtualnej JavaScript, co przede wszystkim oznacza wsparcie dla popularnego formatu wymiany danych JSON (ang. *JavaScript Object Notation*) oraz wsparcia dla nierelacyjnych baz danych.

W niniejszej pracy Node.js wykorzystany został do stworzenia wydajnej aplikacji serwerowej oraz API (ang. *Application Programming Interface*), czyli zestawu reguł i protokołów, które umożliwiają różnym programom komunikację i współpracę ze sobą. Serwer nie tylko obsługuje różnorodne punkty końcowe (ang. *endpoints*), ale również zapewnia dostęp do danych poprzez obiekty DAO (ang. *Data Access Object*). Dzięki temu, API efektywnie łączy bazę danych z aplikacją, co odgrywa kluczową rolę w funkcjonowaniu całego systemu.

Ważnym pojęciem w kontekście bezpieczeństwa aplikacji serwerowej jest autoryzacja i uwierzytelnianie użytkowników. Do tego celu wykorzystana została autoryzacja oparta na tokenach JWT (ang. *JSON Web Tokens*). JWT to otwarty standard (RFC 7519), który definiuje kompaktowy i niezależny sposób bezpiecznego przesyłania informacji między stronami jako obiekty JSON [10]. Tokeny składają się z trzech części. Nagłówek zawierający ogólne informacje o tokenie. Zawartość – zawierająca informacje właściwe np. nazwa konta czy email. Podpis – jest to sposób na weryfikację wiarygodności informacji zawartych w tokenie. Tokeny JWT można podpisać wykorzystując jeden ze sposobów. Pierwszym z nich jest wykorzystanie klucza tajnego szyfrowanego z użyciem algorytmu HMAC (ang. *Hash-based Message Authentication Code*). Algorytm HMAC zapewnia bezpieczne i szybkie podpisywanie tokenów, korzystając z szyfrującej funkcji mieszającej i tajnego klucza współużytkowanego w celu wygenerowania wartości uwierzytelniania. Drugim sposobem jest użycie pary kluczy – publicznego i prywatnego szyfrowanych za pomocą

algorytmów mieszających RSA (ang. *Rivest-Shamir-Adleman*) lub ECDSA (ang. *Elliptic Curve Digital Signature Algorithm*). Algorytmy te oparte są na asymetrycznym szyfrowaniu, co oznacza, że używają pary kluczy do podpisywania i weryfikowania tokenów. Algorytmy te oferują wyższy poziom bezpieczeństwa w porównaniu z HMAC [11].

W przypadku aplikacji, które wymagają autoryzacji i wymiany informacji między różnymi komponentami, JWT stanowi skuteczne rozwiązanie zapewniające bezpieczny i autoryzowany dostęp do zasobów aplikacji. Dzięki zastosowaniu różnych algorytmów podpisywania, można dostosować poziom zabezpieczeń w zależności od specyfiki projektu, co czyni JWT wszechstronnym narzędziem do zapewniania bezpieczeństwa w aplikacjach internetowych.

#### 1.1.4. Baza danych

Elementem niezbędnym do poprawnego działania systemu jest baza danych, która w tym przypadku oparta została o technologię MongoDB. Jej celem jest przechowywanie kluczowych informacji o klientach, obiektach systemu, wydarzeniach oraz transakcjach dokonanych przez użytkowników w obrębie systemu. Wybór tego rozwiązania był podyktowany szeregiem zalet, które MongoDB oferuje w kontekście projektu.

MongoDB to innowacyjna, nierelacyjna baza danych, której główną cechą charakterystyczną jest model danych oparty na dokumentach [12]. W przeciwieństwie do tradycyjnych relacyjnych baz danych, które korzystają z tabel i wierszy, MongoDB gromadzi dane w elastycznych dokumentach w formacie BSON (ang. *Binary JavaScript Object Notation*). Format BSON jest to binarne kodowanie reprezentacji obiektów JSON, co sprawia, że przesyłanie danych między aplikacją, a bazą jest szybsze i bardziej wydajne. Zastosowanie takiej formy zapisu sprawia, że baza danych jest wysoce skalowalna i umożliwia efektywne przechowywanie różnorodnych danych o zróżnicowanej strukturze. Dzięki takiemu rozwiązaniu, można łatwo dostosować bazę do zmieniających się potrzeb projektu, co jest szczególnie istotne w dynamicznych środowiskach. Technologia szczególnie dobrze radzi sobie przy skalowaniu wszerz. Takie skalowanie polega przeważnie na zwiększeniu pojemności serwerów poprzez zakup dodatkowych serwerów lub dysków. Technologia doskonale radzi sobie z obsługą dużych zbiorów danych, co sprawia, że jest idealnym wyborem dla projektów, które wymagają skalowalności i wydajności w obszarze przechowywania i przetwarzania dużej ilości informacji. Systemy spodziewające się pracy na dużych zbiorach danych, np. projektowany system sprzedaży biletów, w szczególności powinny zwracać uwagę właśnie na poruszony wyżej problem związany ze skalowalnością.

Metody dokumentowe MongoDB pozwalają na hierarchizację danych, którą możemy zastosować poprzez osadzenie jednego dokumentu wewnątrz drugiego lub przekazanie referencji. Obie metody są jednakowo poprawne jednak wybór, którą z nich zastosować zależy od przeznaczenia danej kolekcji w bazie danych. Częściej stosowane jest osadzanie, umożliwiające uzyskanie wszystkich pożądanych wyników na raz. Przekłada się to na zwiększoną wydajność systemu. Referencja stosowana jest dla bardziej złożonych struktur. W przypadku MongoDB istotne jest zaprojektowanie schematów w taki sposób, aby przynosiły one jak najwięcej korzyści związanych z wydajnością. W tym kontekście często wykorzystuje się różne wzorce projektowe. Jednym z takich wzorców jest np. drzewo – używane dla danych, które mają mocną strukturę hierarchiczną. Innym przykładem są atrybuty, służące do wyodrębniania pól związkowych do tablicy klucz-wartość w celu ułatwienia sortowania lub wyszukiwania dokumentów.

Pewną nowością wprowadzoną od wersji 4.0 MongoDB są transakcje. Mechanizm ten jest niezwykle przydatny przy operacjach zapisu i odczytu zawartości wielu różnych dokumentów. Każda transakcja w obrębie bazy danych Mongo musi być zgodna ze standardem ACID (ang. *Atomicity, Consistency, Isolation, Durability*). Standard ten gwarantuje, że operacje dokonane w ramach transakcji nie spowodują żadnych błędów związanych z integralnością bazy danych. Okazuje się to niezwykle przydatne nie tylko podczas sytuacji nagłej utraty zasilania, ale przede wszystkim podczas sytuacji, w której na etapie transakcji dochodzi do błędu. Mechanizm transakcji skutecznie chroni bazę danych przed np. częściową utratą danych. Dopiero kiedy wszystkie operacje zawarte w sesji zostały wykonane zgodnie z założeniami, zmiany zostaną zaakceptowane. W razie niepowodzenia lub błędu zmiany zostaną cofnięte (ang. *rollback*).

Technologia oferuje wiele przydatnych mechanizmów. Na przykład replikację danych, która zapewnia wysoką dostępność i odporność na awarie. Przydatne jest również tworzenie różnego rodzaju indeksów, co przekłada się na szybkie wyszukiwanie i analizę danych. To z kolei wpływa na wydajność operacji odczytu i zapisu usprawniające pracę całego systemu.

## **1.2. Zabezpieczenia aplikacji internetowych**

Bezpieczeństwo aplikacji internetowych stanowi kluczowy element w procesie ich projektowania oraz eksploatacji. W niniejszym podrozdziale omówione zostaną różnorodne metody i techniki zabezpieczania aplikacji, mające na celu ochronę danych wrażliwych oraz zapewnienie poufności i integralności informacji.

Jednym z fundamentalnych aspektów ochrony danych jest właściwe zarządzanie hasłami użytkowników. Aby zminimalizować potencjalne szkody hasła nie powinny być przechowywane w sposób jawny, z uwagi na potencjalne ryzyko ich wykradnięcia w przypadku ataku na bazę danych. W tym celu stosuje się np. funkcje hashujące, które przekształcają hasło w nieodwracalny sposób. To oznacza, że nawet jeśli dane zostaną wykradzione, atakujący nie będzie w stanie odtworzyć pierwotnego hasła. Przykładowym bezpiecznym algorytmem kryptograficznym, który można wykorzystać do przekształceń haseł jest np. algorytm bcrypt.

Bardzo popularną w ostatnim czasie metodą zapewniającą dodatkowy poziom bezpieczeństwa jest autoryzacja dwuetapowa. Ta metoda wymaga od użytkownika podania dodatkowego, jednorazowego kodu weryfikacyjnego, który jest generowany i wysyłany na podany numer telefonu lub adres e-mail. Wprowadzenie tego dodatkowego kroku znacząco utrudnia nieautoryzowany dostęp do kont użytkowników nawet w przypadku, gdy hasło zostanie złamane lub wykradzione.

Nie można również zapominać o regularnych aktualizacjach oprogramowania. Choć tworzone w tych czasach systemy są coraz bezpieczniejsze to jednak wciąż zdarzają się luki w oprogramowaniu. Częste aktualizacje oprogramowania i bibliotek pozwalają uzyskać dostęp do najnowszych poprawek bezpieczeństwa, minimalizując ryzyko wykorzystania znanych podatności systemu i aplikacji.

Innym sposobem jest wdrożenie systemu monitorowania logów, który umożliwia szybkie wykrywanie nieprawidłowości oraz potencjalnych ataków. Analiza logów pozwala na szybką reakcję na ewentualne zagrożenia oraz podejrzane aktywności [13].

Ograniczanie dostępu do różnych części aplikacji na podstawie roli i uprawnień użytkowników to równie istotny element zabezpieczania współczesnych systemów. Dzięki temu, nieautoryzowani użytkownicy nie mogą uzyskać dostępu do wrażliwych danych lub funkcji.

Wykorzystanie protokołu HTTPS (ang. *Hypertext Transfer Protocol Secure*) oraz certyfikatów SSL (ang. *Secure Sockets Layer*) / TLS (ang. *Transport Layer Security*), także stanowi ważny element w ochronie danych przesyłanych przez aplikację. Takie rozwiązania pozwalają na sprawne zabezpieczenie komunikacji pomiędzy klientem, a serwerem.

Wykorzystanie rozwiązania w postaci wdrożenia zapory sieciowej (ang. *firewall*) również może pomóc chronić aplikację przed niechcianymi atakami już na etapie komunikacji przeglądarki (klienta) z serwerem. Takie rozwiązanie pozwala wyłapać np. ataki typu SQL

Injection, Cross-Site Scripting (XSS), ataki wykorzystujące niepoprawne zarządzanie sesją i uwierzytelnianiem, czy niezabezpieczony bezpośredni dostęp do obiektów. Wykorzystanie odpowiedniej zapory sieciowej pozwala na filtrację ruchu internetowego i monitorowanie żądań do aplikacji, eliminując tym samym potencjalnie niebezpieczne zapytania lub treści [14].

Inną metodą może być np. przeprowadzanie regularnych audytów bezpieczeństwa pozwalających na identyfikację potencjalnych słabych punktów w aplikacji. Specjaliści ds. bezpieczeństwa mogą przeprowadzać testy penetracyjne oraz analizy kodu, identyfikując ewentualne luki w zabezpieczeniach i sugerując sposoby ich naprawy.

Sposobów na zwiększenie bezpieczeństwa oraz ochronę danych jest wiele. Niestety, pomimo skuteczności przedstawionych wyżej sposobów, to wciąż człowiek, a w kontekście aplikacji – potencjalny użytkownik stanowi najsłabsze ognisko. Dlatego istotna jest również odpowiednia edukacja użytkowników w kwestii bezpieczeństwa i zachęcanie ich np. do wspomnianej wyżej autoryzacji dwuetapowej czy też wymuszanie odpowiedniej polityki tworzenia haseł. Silne hasła są kluczowym elementem w ochronie kont użytkowników. Dlatego też, w nowoczesnych aplikacjach stosowane są mechanizmy wymuszające na użytkownikach stosowanie złożonych haseł, które składają się z różnych typów znaków (małe i wielkie litery, cyfry, znaki specjalne). Ta praktyka ma na celu zminimalizowanie ryzyka ataków siłowych (ang. *brute force*) oraz utrudnienie potencjalnym włamywaczom próby odgadnięcia hasła.

### **1.3. Ochrona danych osobowych w aplikacjach internetowych**

W poniższym podrozdziale skupiono się na omówieniu Rozporządzenia Ogólnego o Ochronie Danych (RODO), które stanowi kluczową podstawę prawną w zakresie ochrony danych osobowych w Unii Europejskiej.

ROZPORZĄDZENIE PARLAMENTU EUROPEJSKIEGO i RADY (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych) wprowadza szereg istotnych przepisów mających na celu zabezpieczenie prywatności i praw osób fizycznych w kontekście przetwarzania ich danych osobowych [15]. Rozporządzenie narzuca szereg założeń oraz obowiązków, jakie spoczywają na podmiotach przetwarzających dane. W kontekście aplikacji internetowych, administrator systemu zobowiązany jest m.in. do

wdrożenia i utrzymania procedur oraz mechanizmów, które zapewnią zgodność z wymaganiami RODO. Obejmuje to m.in. wyznaczenie Inspektora Ochrony Danych (IOD), jeśli jest to wymagane. Administrator ma obowiązek poinformowania użytkowników o celach, dla których zbierane są ich dane osobowe, oraz o prawach, jakie przysługują im w związku z przetwarzaniem tych danych. Administrator zobowiązany jest również do zabezpieczenia danych przed nieautoryzowanym dostępem, utratą, uszkodzeniem lub ujawnieniem. W przypadku wycieku lub incydentu związanego z danymi osobowymi, administrator ma obowiązek zgłosić ten fakt właściwym organom oraz poinformować osoby, których dane mogły być naruszone.

Warto również poruszyć kwestię prawa do zapomnienia, które zgodnie z artykułem 17. RODO, umożliwia osobie, której dane dotyczą, żądanie usunięcia swoich danych osobowych przez administratora. Oznacza to, że jeśli dane nie są już potrzebne do celów, dla których zostały zebrane lub przetwarzane, lub jeśli osoba wycofa zgodę na przetwarzanie, administrator ma obowiązek trwale usunąć te dane. Prawo to ma na celu pozwolić umożliwić jednostkę większą kontrolę nad swoimi danymi osobowymi i ochronę ich prywatności.

Stosowanie się do zasad zawartych w RODO, pozwala znaczco zminimalizować ryzyko wycieku lub pozyskania danych wrażliwych, niestety z punktu widzenia biznesowego może sprawić pewne trudności. Takim przypadkiem jest agregacja danych, która często jest niezbędna do skutecznego prowadzenia analiz i podejmowania kluczowych decyzji. W kontekście ochrony danych osobowych, proces ten staje się wyzwaniem, zwłaszcza w przypadku big data, gdzie istnieje konieczność łączenia danych z wielu różnych źródeł. Taki przypadek może być niezgodny z wymogami prawnymi dotyczącymi ochrony danych osobowych. Dlatego też, kluczowym aspektem jest zastosowanie odpowiednich technik np. anonimizacji, które pozwalają na zachowanie poufności danych, nie łamiąc przy tym przepisów RODO. Anonimizacja danych to proces usuwania lub zmieniania informacji identyfikujących dane osobowe, w taki sposób, aby nie można było ich powiązać z konkretną osobą. Zanonimizowane dane nie są już traktowane jako dane osobowe i nie podlegają ochronie RODO. Jednakże, aby dane mogły być uznane za zanonimizowane, muszą być one poddane takiej obróbce, która uniemożliwia identyfikację osób, do których się odnoszą.

Naruszenie przepisów dotyczących ochrony danych osobowych wiąże się z surowymi karami finansowymi. Przepisy przewidują dwie kategorie kar, zależnych od rodzaju i szkodliwości przewinienia. UODO może nałożyć na podmiot karę nawet 20 mln euro lub 4% całkowitego rocznego obrotu firmy z poprzedniego roku. Wysokość kary może być

zależna od rzeczy takich jak charakter i waga czynu, czas trwania naruszenia, liczba poszkodowanych osób i rozmiar poniesionej przez nich szkody czy kategorie danych osobowych, których dotyczyło naruszenie przepisów RODO. Prezes UODO może również zdecydować, że kara nie będzie miała charakteru finansowego [16].

## 2. Dokumentacja techniczna

W niniejszym rozdziale przedstawione zostało działanie poszczególnych warstw systemu oraz zakres jego funkcjonowania. Rozdział skupia się także na szczegółowej analizie oraz charakterystyce aktorów korzystających z systemu. Przedstawiony również został moduł kliencki i administracyjny wraz z wymaganiami funkcjonalnymi systemu.

### 2.1. Zakres systemu

Projektowany system składa się z trzech kluczowych warstw, z których każda pełni istotną rolę w zapewnieniu sprawnego funkcjonowania oraz komunikacji między różnymi komponentami systemu.

#### 2.1.1. Warstwa aplikacji webowej

Warstwa aplikacji webowej stanowi alternatywny sposób dostępu do systemu, szczególnie dla użytkowników preferujących przeglądanie na większych ekranach, takich jak komputery stacjonarne czy laptopy. Aplikacja webowa stanowi największą i jednocześnie najbardziej rozbudowaną część systemu, podyktowane jest to ilością możliwości, które oferuje.

Możliwe jest wyróżnienie dwóch zasadniczych funkcji tej warstwy. Pierwsza z nich to dostarczenie użytkownikom systemu, z poziomu przeglądarki internetowej, intuicyjnego i wygodnego interfejsu, który umożliwia im przeglądanie i wyszukiwanie interesujących ich wydarzeń. Użytkownicy systemu zostali podzieleni na dwie główne grupy – użytkowników zalogowanych oraz użytkowników niezalogowanych. Użytkownicy niezalogowani posiadają ograniczony dostęp do możliwości systemu i nie posiadają takich funkcjonalności jak dodawanie wydarzeń do ulubionych czy dodawanie wydarzeń do koszyka. Posiadają oni jednak swobodny dostęp do przeglądania wydarzeń, dostępnych biletów czy innych udostępnionych użytkownikom zasobów systemu. Użytkownicy zalogowani posiadają natomiast szereg dodatkowych możliwości tj. dodawanie wydarzeń do ulubionych lub obserwowanych, dodawania biletów do koszyka, kupowanie biletów na dane wydarzenia czy ustawienia preferencji proponowanych wydarzeń.

Drugą niemniej istotną funkcjonalnością aplikacji webowej jest udostępnianie organizatorom panelu administracyjnego, z którego w wygodny sposób mogą tworzyć i dodawać nowe wydarzenia. Panel umożliwia również podgląd oraz edycję utworzonych

wydarzeń, a także dostarcza szereg statystyk i wykresów umożliwiających organizatorom analizę przychodów czy popularności utworzonych wydarzeń.

### **2.1.2. Warstwa aplikacji mobilnej**

Warstwa aplikacji mobilnej, w odróżnieniu od warstwy aplikacji webowej, została przeznaczona innej grupie odbiorców końcowych. Do grupy tej należą osoby, które korzystają z urządzeń przenośnych takich jak smartfony czy tablety. Głównym celem tej warstwy jest dostarczenie intuicyjnego i przyjaznego interfejsu z którego to poziomu użytkownik uzyska dostęp do przeprowadzonych przez siebie transakcji oraz kupionych biletów. Oprócz szczegółów wydarzenia do każdego biletu dołączony będzie kod QR, który umożliwiły szybką weryfikację ważności biletu przez organizatora wydarzenia np. poprzez zeskanowanie kodu przez pracownika kina. Dodatkowo użytkownik może przejść do szczegółów na temat wydarzenia z poziomu kupionego biletu. Aplikacja mobilna skupia się wyłącznie na dostarczeniu najbardziej istotnych możliwości systemu użytkownikom mobilnym i w odróżnieniu od aplikacji webowej nie udostępnia organizatorom panelu administracyjnego.

### **2.1.3. Warstwa aplikacji serwerowej**

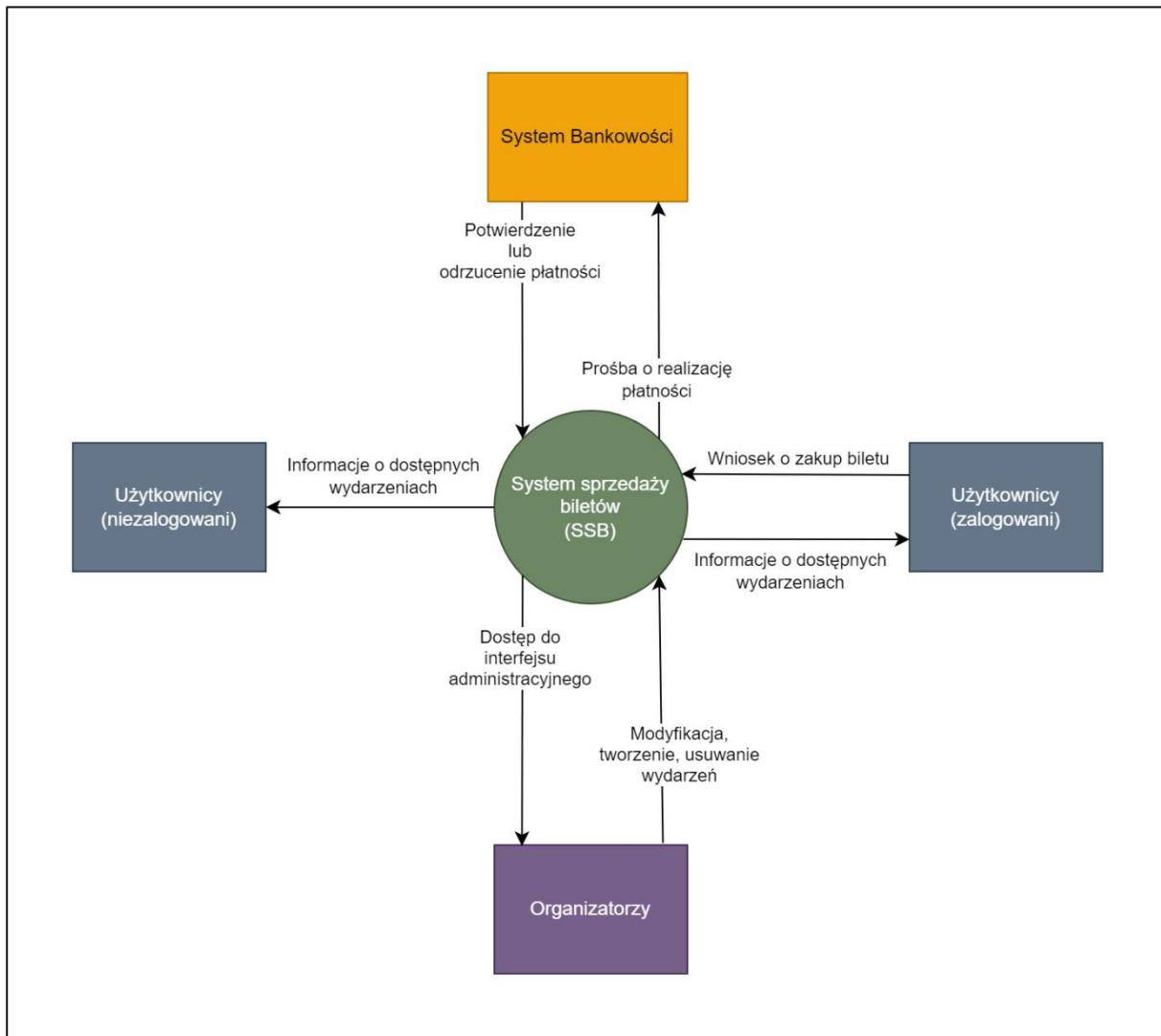
Warstwa aplikacji serwerowej pełni kluczową rolę w przetwarzaniu i zarządzaniu danymi. To tutaj odbywa się komunikacja z bazą danych oraz obsługa żądań klientów. Ponadto, warstwa ta zarządza logiką biznesową, w tym procesem sprzedaży biletów czy dodawaniem wydarzeń. Aplikacja serwerowa obsługuje również autoryzację i uwierzytelnianie użytkowników, w tym celu wykorzystano autoryzację opartą o tokeny JWT, która szczegółowo omówiona została w rozdziale 1.

Aplikacja serwerowa wykorzystuje standardowe protokoły obsługi żądań HTTP, które umożliwiają aplikacji mobilnej oraz webowej swobodną komunikację z bazą danych. Dostęp do zasobów oraz komunikacja odbywają się dzięki zastosowaniu punktów końcowych (ang. *endpoints*). Zastosowana architektura aplikacji serwerowej umożliwia szybkie przekazywanie informacji oraz synchronizację danych pomiędzy różnymi platformami.

## **2.2. Perspektywa produktu**

System SSB służy jako oprogramowanie zarządzające informacjami o dostępnych wydarzeniach, umożliwia aktorom wyszukiwanie, przeglądanie i kupno biletów na wybrane wydarzenia, pozwala również na tworzenie, edycję i analizę wydarzeń specjalnemu typowi aktorów zwanych w systemie organizatorami. System SSB, złożony jest z trzech warstw

opisanych w rozdziale 2. Warstwy współpracują w ramach jednego systemu oferując aktorom szereg funkcjonalności. Diagram kontekstowy, przedstawiony na rysunku 2.1. stanowi kluczowy element analizy systemu, pozwalając na identyfikację aktorów korzystających z systemu oraz relacji jakie zachodzą między nimi, a systemem. System SSB zakłada współpracę z czterema typami aktorów wyszczególnionych na diagramie kontekstowym i szczegółowo opisanych w podrozdziale 2.3.



Rys. 2.1. Diagram kontekstowy systemu SSB. (źródło: Opracowanie własne)

Głównymi obiektami z którymi system SSB się komunikuje są użytkownicy zalogowani oraz niezalogowani. Docelowo system komunikuje się z użytkownikami niezalogowanymi jedynie w celu udostępniania informacji o wydarzeniach, ilość interakcji pomiędzy systemem, a użytkownikami niezalogowanymi jest ograniczona jedynie do podstawowych funkcjonalności tj. wyświetlania i wyszukiwania wydarzeń.

Komunikacja między systemem SSB, a użytkownikami zalogowanymi jest bardziej złożona i sprowadza się nie tylko do wyświetlania i wyszukiwania zasobów, ale również umożliwia dodawanie wydarzeń do ulubionych, obserwowanych czy koszyka, a także zakup biletów na wybrane przez użytkowników zalogowanych wydarzenia. System umożliwia również, użytkownikom zalogowanym zmianę preferencji w celu dostosowania proponowanych wydarzeń.

Z systemu korzystać będą również aktorzy zwani w projekcie organizatorami. System umożliwia organizatorom tworzenie, edycję oraz usuwanie wydarzeń. Oferuje również wykres przedstawiający sprzedaż biletów w czasie czy szereg statystyk dotyczących stworzonych przez nich wydarzeń.

Na diagramie kontekstowym wyróżnić możemy jeszcze jednego aktora – system bankowości. System bankowości jest obiektem zewnętrznym z którym system SSB zakłada komunikację jedynie w celu przeprowadzania płatności i otrzymania od niego informacji o potwierdzeniu bądź odrzuceniu operacji płatności.

### **2.3. Aktorzy i charakterystyka użytkowników**

W tabelach 2.1. – 2.4. zostały przedstawione charakterystyki aktorów korzystających z systemu SSB. Każdy z przedstawionych aktorów posiada określone uprawnienia i możliwości w ramach systemu.

*Tab. 2.1. Charakterystyka użytkownika niezalogowanego.*

<b>ID: USER_GUEST</b>
<b>Nazwa: Użytkownik niezalogowany</b>
<b>Opis:</b>
Użytkownik niezalogowany to domyślnie klient, który odwiedza aplikację webową bądź mobilną po raz pierwszy. Użytkownik niezalogowany posiada możliwość przeglądania i wyszukiwania dostępnych wydarzeń, może utworzyć nowe konto w systemie SSB wypełniając poprawnie odpowiedni formularz dostępny z pozycji interfejsu użytkownika. Użytkownik niezalogowany może, wypełniając poprawnie formularz logowania, zalogować się do systemu, tym samym zyskując dostęp do nowych funkcjonalności systemu.

Tab. 2.2. Charakterystyka użytkownika zalogowanego.

<b>ID: USER_LOGGED</b>
<b>Nazwa: Użytkownik zalogowany</b>
<b>Opis:</b> Użytkownik zalogowany to klient posiadający już konto w systemie SSB. Użytkownik zalogowany posiada dostęp do podstawowych funkcji systemu tj. wyszukiwanie i przeglądanie dostępnych wydarzeń, ale również do funkcjonalności tj. dodawanie wydarzeń do ulubionych, obserwowanych czy koszyka, a także zakup biletów.

Tab. 2.3. Charakterystyka organizatora.

<b>ID: ORGANIZER</b>
<b>Nazwa: Organizator</b>
<b>Opis:</b> Organizator to firma lub organizacja zewnętrzna, która pomyślnie przeszła etap rejestracji konta organizatora dostępny z poziomu aplikacji webowej oraz pomyślnie zalogowała się do systemu danymi konta organizatora. Organizator po zalogowaniu zyskuje dostęp do specjalnego panelu administracyjnego w którym może dodawać, edytować usuwać bądź analizować utworzone przez siebie wydarzenia.

Tab. 2.4. Charakterystyka systemu bankowości.

<b>ID: BANK</b>
<b>Nazwa: System bankowości</b>
<b>Opis:</b> System bankowości to firma lub organizacja zewnętrzna realizująca płatność użytkowników zalogowanych w systemie. System bankowości dodatkowo powinien przesyłać informację zwrotną potwierdzającą bądź odrzucającą płatność systemowi SSB w celu finalizacji bądź odrzuceniu zamówienia po stronie SSB. Jako, że system bankowości jest aktorem zewnętrznym (terminator), niniejsza praca nie implementuje jego działania.

## 2.4. Diagram przypadków użycia

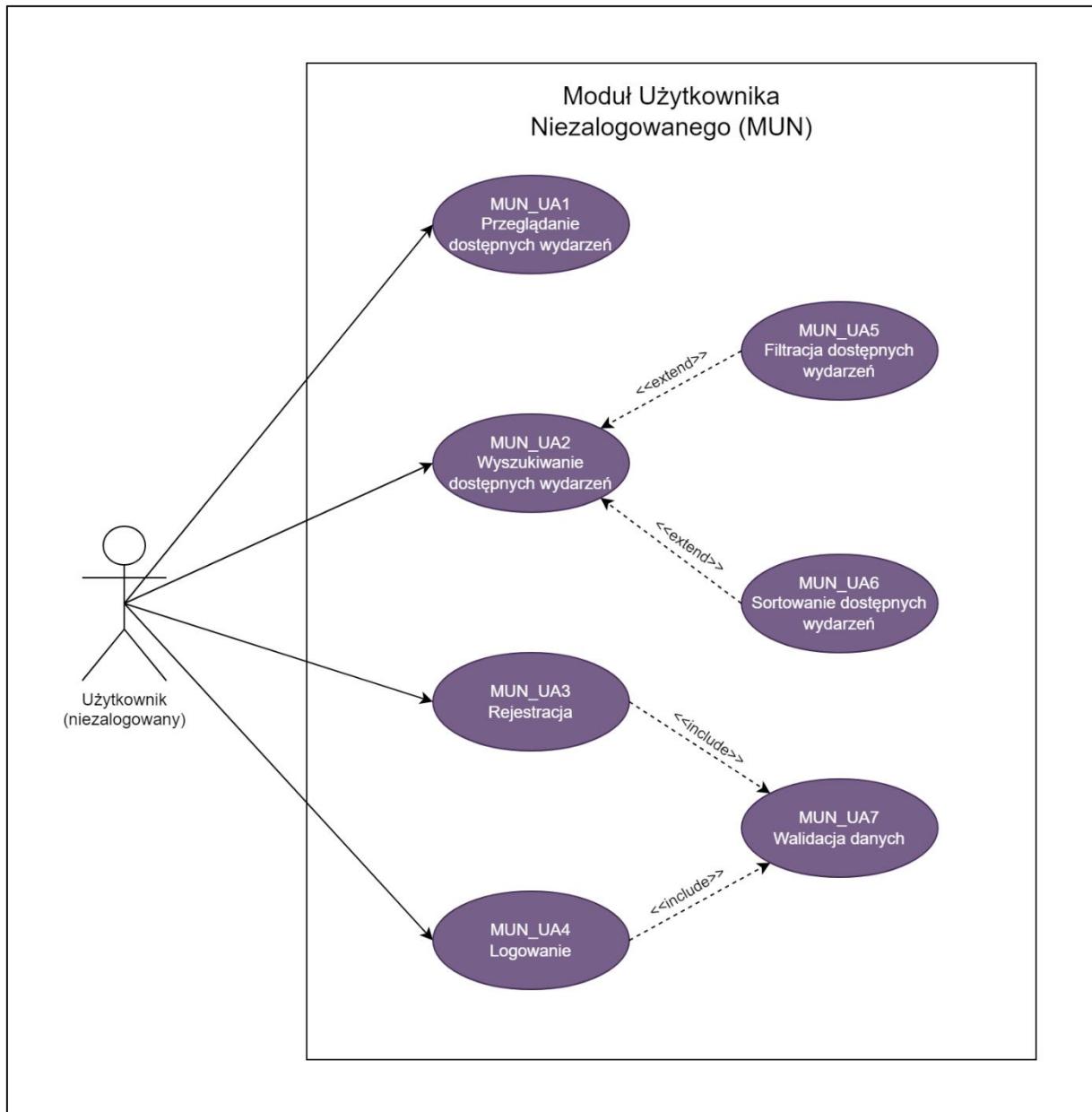
W tym podrozdziale skupiono się na przedstawieniu i opisaniu diagramów przypadków użycia. W celu lepszego zobrazowania pracy systemu postanowiono podzielić ten rozdział na dwa mniejsze podrozdziały opisujące kolejno sekcję kliencką, na którą składają się moduł użytkownika zalogowanego i moduł użytkownika niezalogowanego oraz sekcję administracyjną na którą składa się moduł organizatora.

### 2.4.1. Sekcja kliencka

W tym podrozdziale skupiono się na przedstawieniu i opisaniu diagramów przypadków użycia związanych z sekcją kliencką. Sekcja składa się z dwóch głównych modułów: modułu użytkownika zalogowanego (MUZ) oraz modułu użytkownika niezalogowanego (MUN).

Moduł użytkownika niezalogowanego przedstawiony został na rysunku 2.2. reprezentuje on interakcje zachodzące między użytkownikami niezalogowanymi, a systemem SSB. Moduł użytkownika zalogowanego przedstawiony został na rysunku 2.3. reprezentuje on interakcje zachodzące między użytkownikami zalogowanymi, systemem bankowości, a systemem SSB.

Moduł organizatora przedstawiony został na rysunku 2.4. reprezentuje on interakcje zachodzące między organizatorami, a systemem SSB. Poszczególne przypadki użycia zostały szczegółowo scharakteryzowane w tabelach od 2.5. – 2.11. dla modułu MUN oraz w tabelach 2.12. – 2.27. dla modułu MUZ.



Rys. 2.2. Diagram przypadków użycia MUN. (źródło: Opracowanie własne)

Tab. 2.5. Przypadek użycia – Przeglądanie dostępnych wydarzeń.

<b>ID: MUN_UA1</b>
<b>Nazwa: Przeglądanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik niezalogowany wchodzi na stronę główną aplikacji webowej lub mobilnej i chce przeglądać dostępne wydarzenia.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>Użytkownik chce przeglądać dostępne wydarzenia.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik nie jest zalogowany w systemie.</li><li>Użytkownik znajduje się na jednej ze stron prezentujących dostępne wydarzenia.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest w stanie przeglądać dostępne wydarzenia.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik otwiera stronę główną aplikacji webowej lub mobilnej.</li><li>System prezentuje użytkownikowi dostępne wydarzenia.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Użytkownik otwiera inną stronę / podstronę przeznaczoną do prezentacji wydarzeń w aplikacji webowej lub mobilnej.</li><li>System prezentuje użytkownikowi dostępne wydarzenia.</li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada bądź nie ma dostępu do bazy danych.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.6. Przypadek użycia – Wyszukiwanie dostępnych wydarzeń.

<b>ID: MUN_UA2</b>
<b>Nazwa: Wyszukiwanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik niezalogowany chce wyszukać dostępne wydarzenia. W tym celu wchodzi na odpowiednią stronę. System prezentuje użytkownikowi odpowiednie wyniki.
<b>Wyzwalacze:</b> 1. Użytkownik chce skorzystać z funkcji wyszukiwania dostępnych wydarzeń.
<b>Warunki początkowe:</b> 1. Użytkownik nie jest zalogowany w systemie. 2. Użytkownik znajduje się na stronie umożliwiającej wyszukiwanie wydarzeń.
<b>Warunki końcowe:</b> 1. Użytkownik otrzymuje domyślne wyniki wyszukiwania wydarzeń.
<b>Scenariusz główny:</b> 1. Użytkownik odwiedza stronę umożliwiającą wyszukiwanie i filtrację wydarzeń. 2. Użytkownik nie używa filtracji ani sortowania. 3. System domyślnie prezentuje użytkownikowi wszystkie dostępne wydarzenia w kolejności od najnowszego.
<b>Scenariusz alternatywny i rozszerzenia:</b> 2.A. Użytkownik używa jednego filtra lub kombinacji filtrów [MUN_UA5]. 2.A.1. System prezentuje spersonalizowane wyniki wyszukiwania wg. zastosowanej filtracji. 2.B. Użytkownik używa jednej z dostępnych opcji sortowania [MUN_UA6]. 2.B.1. System prezentuje wyniki wg. wybranej opcji sortowania.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. System nie może przetworzyć zapytania wyszukiwania.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.7. Przypadek użycia – Filtracja dostępnych wydarzeń.

<b>ID: MUN_UA5</b>
<b>Nazwa: Filtracja dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik niezalogowany chce wyszukać tylko konkretne wydarzenia spełniające warunki filtracji. W tym celu wybiera pożądane filtry z dostępnych i kliką przycisk "Filtruj".
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik niezalogowany pragnie skorzystać z filtracji.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik jest niezalogowany.</li><li>2. Użytkownik znajduje się na stronie oferującą możliwość filtracji wyszukiwanych wydarzeń.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System prezentuje użytkownikowi wydarzenia spełniające kryteria filtracji.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik wybiera jedną opcję filtracji.</li><li>2. System prezentuje wyniki spełniające kryteria.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>1.A. Użytkownik używa kombinacji filtrów.<ol style="list-style-type: none"><li>1.A.1. System prezentuje spersonalizowane wyniki wyszukiwania wg. zastosowanej filtracji.</li></ol></li><li>1.B. Użytkownik usuwa lub zmienia filtry do wartości domyślnych.<ol style="list-style-type: none"><li>1.B.1. System prezentuje domyślne (wszystkie wydarzenia) wyniki wyszukiwania.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. System nie może przetworzyć zapytania dotyczącego filtrowania.</li><li>3. Brak wyników spełniających kryteria.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.8. Przypadek użycia – Sortowanie dostępnych wydarzeń.

<b>ID: MUN_UA6</b>
<b>Nazwa: Sortowanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik niezalogowany chce posortować już wyszukane wyniki. W tym celu wybiera pożądaną opcję sortowania z wysuwanego menu zawierającego dostępne opcje sortowania.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik niezalogowany pragnie posortować wyniki wyszukiwania.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest niezalogowany.</li><li>Użytkownik znajduje się na stronie oferującą możliwość sortowania dostępnych wyników.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System prezentuje użytkownikowi posortowane wyniki.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik wybiera jedną z dostępnych opcji sortowania.</li><li>System prezentuje wyniki posortowane wg. kryterium sortowania.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>System nie może przetworzyć zapytania dotyczącego filtrowania.</li><li>Brak wyników spełniających kryteria.</li></ol>
Dodatkowe wymagania: <ul style="list-style-type: none"><li>dostęp do sieci</li></ul>

Tab. 2.9. Przypadek użycia – Rejestracja.

<b>ID: MUN_UA3</b>
<b>Nazwa: Rejestracja</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik niezalogowany chce utworzyć konto w systemie, aby móc korzystać z dodatkowych funkcji i możliwości. Użytkownik wypełnia formularz rejestracyjny. Jeżeli formularz został wypełniony poprawnie, a aplikacja serwerowa nie zgłosi żadnego błędu użytkownik pomyślnie utworzy nowe konto w systemie SSB.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik niezalogowany pragnie utworzyć nowe konto w systemie.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik nie jest zalogowany w systemie.</li><li>2. Użytkownik znajduje się na stronie logowania lub innej dostępnej dla niezalogowanych użytkowników.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. Nowy obiekt (użytkownik) zostaje utworzony w systemie.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik niezalogowany klika na przycisk "Zarejestruj się" znajdującym się w menu na stronie aplikacji mobilnej lub webowej.</li><li>2. Użytkownik zostaje przekierowany do strony z formularzem rejestracyjnym.</li><li>3. Użytkownik poprawnie wypełnia wszystkie wymagane pola.</li><li>4. Użytkownik klika na przycisk "Zarejestruj się" znajdującym się pod formularzem.</li><li>5. System weryfikuje dane wprowadzone w pola po stronie aplikacji webowej lub mobilnej.</li><li>6. System waliduje dane po stronie aplikacji serwerowej [MUN_UA7].</li><li>7. System tworzy nowy obiekt (użytkownik) w bazie danych.</li><li>8. System informuje użytkownika o poprawnym utworzeniu konta.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <p>5.A. System wykrywa, że użytkownik wprowadził niepoprawny adres e-mail.</p> <p>5.A.1. System wyświetla komunikat o niepoprawnym e-mailu i prosi o jego poprawienie.</p> <p>5.A.2. Należy powtórzyć kroki począwszy od kroku 3.</p> <p>5.B. System wykrywa, że hasło użytkownika nie zgadza się z powtórzonym hasłem lub nie spełnia polityki silnych haseł.</p> <p>5.B.1. System wyświetla komunikat o zbyt słabym haśle lub o tym, że hasło i powtórzone hasło się nie zgadzają. System prosi o zmianę.</p> <p>5.B.2. Należy powtórzyć kroki począwszy od kroku 3.</p> <p>5.C. System wykrywa, że przynajmniej jedno z wymaganych pól jest puste.</p> <p>5.C.1. System wyświetla komunikat o konieczności wypełnienia brakujących wymaganych pól.</p>

5.C.2. Należy powtórzyć kroki począwszy od kroku 3.

6.A. System wykrywa, że podany adres e-mail jest już zarejestrowany w systemie.

6.A.1. System wyświetla komunikat informujący o konieczności wyboru innego adresu e-mail.

6.A.2. Należy powtórzyć kroki począwszy od kroku 3.

6.B. System wykrywa, że podana nazwa użytkownika jest już zajęta w systemie.

6.B.1. System wyświetla komunikat informujący o konieczności wyboru innej nazwy użytkownika.

6.B.2. Należy powtórzyć kroki począwszy od kroku 3.

#### **Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego rejestracji.
3. Użytkownik o podanej nazwie już istnieje.
4. Podany email już istnieje.
5. Przynajmniej jedno z wymaganych pól formularza jest puste.

#### **Dodatkowe wymagania:**

- podczas tworzenia nowego obiektu (użytkownik) w bazie danych automatycznie tworzony jest jego id
- podczas tworzenia nowego obiektu (użytkownik) w bazie danych równolegle tworzony jest obiekt "password" przechowujący w postaci nie jawnej, zaszyfrowane funkcją kryptograficzną, hasło użytkownika.

*Tab. 2.10. Przypadek użycia – Logowanie.*

**ID: MUN\_UA4**

**Nazwa: Logowanie**

**Aktorzy główni:** Użytkownik niezalogowany

**Aktorzy pomocniczy:** brak

**Poziom:** Użytkownik

**Priorytet:** Niski

#### **Opis:**

Użytkownik niezalogowany chce zalogować się na swoje konto w systemie, aby uzyskać dostęp do dodatkowych funkcji. W tym celu musi wybrać opcję "Zaloguj się" dostępnej w menu w aplikacji webowej lub mobilnej.

#### **Wyzwalacze:**

1. Użytkownik pragnie zalogować się na swoje konto.

#### **Warunki początkowe:**

1. Użytkownik jest niezalogowany
2. Użytkownik posiada aktywne konto w systemie.
3. Użytkownik znajduje się na stronie z formularzem logowania.

**Warunki końcowe:**

1. Użytkownik niezalogowany loguje się do systemu stając się jednocześnie użytkownikiem zalogowanym.

**Scenariusz główny:**

1. Użytkownik niezalogowany klikna na przycisk "Zaloguj się" znajdujący się w menu na stronie aplikacji mobilnej lub webowej.
2. Użytkownik zostaje przekierowany do strony z formularzem do logowania.
3. Użytkownik poprawnie wypełnia wszystkie wymagane pola.
4. Użytkownik klikna na przycisk "Zaloguj się" znajdującym się pod formularzem.
5. System weryfikuje dane wprowadzone w pola po stronie aplikacji webowej lub mobilnej.
6. System waliduje dane po stronie aplikacji serwerowej [MUN\_UA7].
7. System tworzy nową sesję użytkownika nadając mu aktywny token JWT.
8. Użytkownik niezalogowany staje się użytkownikiem zalogowanym i otrzymuje od teraz może używać dodatkowych funkcji systemu.

**Scenariusz alternatywny i rozszerzenia:**

- 5.A. System wykrywa, że przynajmniej jedno z wymaganych pól jest puste.
  - 5.A.1. System wyświetla komunikat o konieczności wypełnienia brakujących wymaganych pól.
  - 5.A.2. Należy powtórzyć kroki począwszy od kroku 3.
- 6.A. System wykrywa, że użytkownik wprowadził niepoprawne hasło bądź login
  - 6.A.1. System wyświetla komunikat o błędnej autoryzacji.
  - 6.A.2. Należy powtórzyć kroki począwszy od kroku 3.

**Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego logowania.
3. Dane logowania są niepoprawne.
4. Użytkownik nie istnieje.
5. Obiekt użytkownik jest uszkodzony bądź niekompletny.

**Dodatkowe wymagania:**

- dostęp do sieci

*Tab. 2.11. Przypadek użycia – Walidacja danych.*

<b>ID: MUN_UA7</b>
<b>Nazwa: Walidacja danych</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> System
<b>Priorytet:</b> Wysoki
<b>Opis:</b> Użytkownik niezalogowany wypełnia formularz i przesyła go dalej, system sprawdza poprawność danych formularza.

**Wyzwalacze:**

1. Użytkownik chce przesłać wypełniony formularz.

**Warunki początkowe:**

1. Użytkownik nie jest zalogowany w systemie.
2. Użytkownik znajduje się na stronie rejestracji lub logowania.
3. Użytkownik wyzwała funkcję przesyłającą formularz.

**Warunki końcowe:**

1. Dane formularza zostają zwalidowane i zweryfikowane, a system przesyła odpowiedź do klienta.

**Scenariusz główny:**

1. Użytkownik wypełnia formularz i przesyła go do weryfikacji.
2. System waliduje i weryfikuje dane.
3. System tworzy nowy obiekt użytkownik w systemie lub tworzy nową sesję użytkownika i loguje go do systemu.
4. System zwraca informację o wyniku walidacji.

**Scenariusz alternatywny i rozszerzenia:**

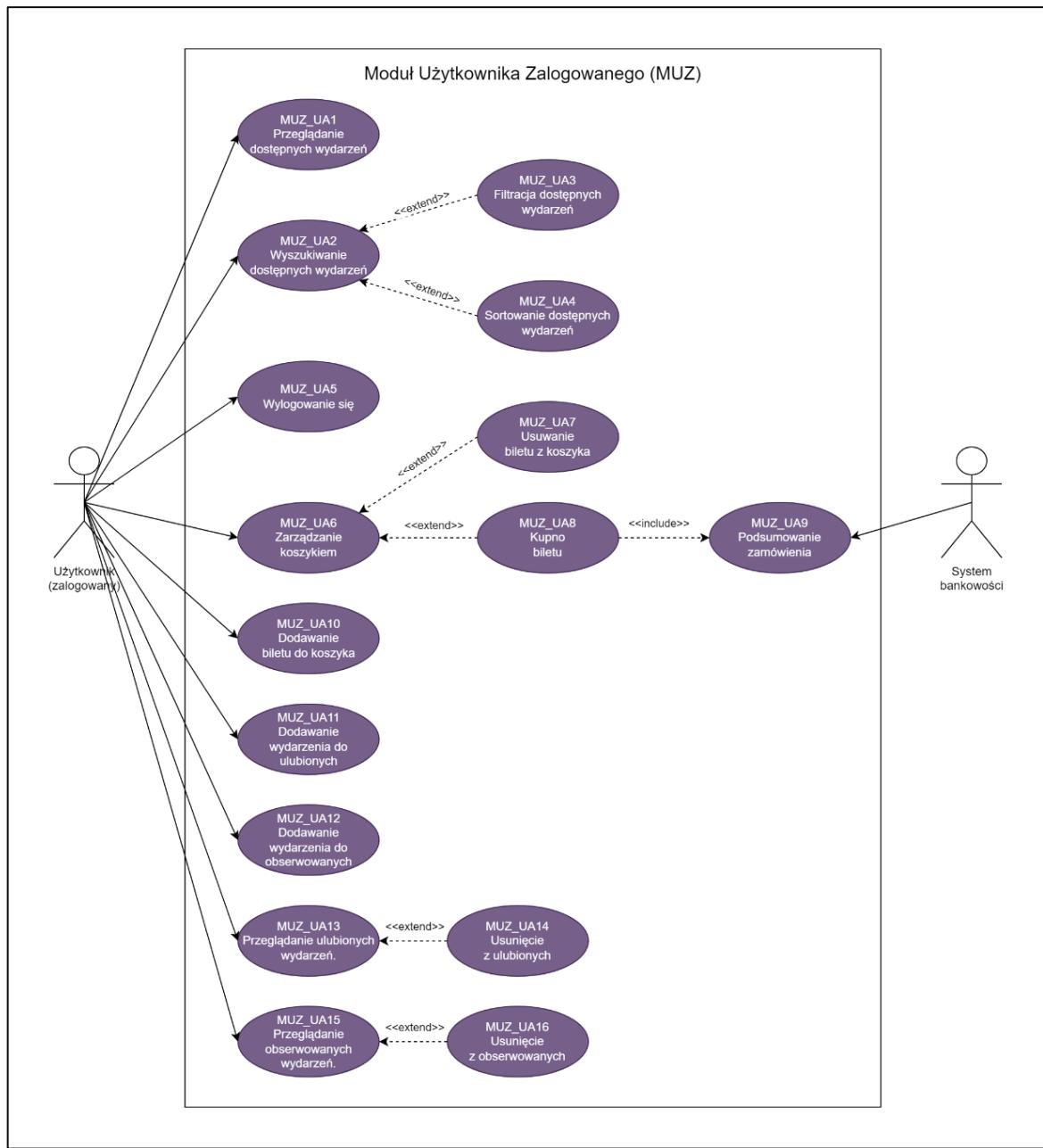
- 3.A. Dane rejestracji nie zostały poprawnie zwalidowane lub zweryfikowane.
  - 3.A.1. System nie tworzy nowego obiektu.
  - 3.A.2. System zwraca komunikat o błędzie.
  - 3.A.3. Koniec przypadku użycia.
- 3.B. Dane logowania nie zostały poprawnie zwalidowane lub zweryfikowane.
  - 3.B.1. System nie tworzy nowej sesji użytkownika i nie loguje go do systemu.
  - 3.B.2. System zwraca komunikat o błędzie.
  - 3.B.3 Koniec przypadku użycia.

**Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego walidacji danych.
3. Dane logowania są błędne.
4. Dane rejestracji są błędne.
5. Wymagane pola są puste.
6. Hasło nie spełnia polityki silnych haseł.
7. Hasło i powtórz hasło nie są identyczne.
8. Próba rejestracji adresu e-mail, który już istnieje.
9. Próba rejestracji nazwy użytkownika, który już istnieje.
10. Hasło nie zgadza się z hasłem w bazie danych.
11. Login nie zgadza się z loginem w bazie danych.

**Dodatkowe wymagania:**

- dostęp do sieci



Rys. 2.3. Diagram przypadków użycia MUZ. (źródło: Opracowanie własne)

Tab. 2.12. Przypadek użycia – Przeglądanie dostępnych wydarzeń (MUZ).

<b>ID: MUZ_UA1</b>
<b>Nazwa: Przeglądanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany wchodzi na stronę główną aplikacji webowej lub mobilnej i chce przeglądać dostępne wydarzenia.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik chce przeglądać dostępne wydarzenia.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany w systemie.</li><li>Użytkownik znajduje się na jednej ze stron prezentujących dostępne wydarzenia.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest w stanie przeglądać dostępne wydarzenia.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik otwiera stronę główną aplikacji webowej lub mobilnej.</li><li>System prezentuje użytkownikowi dostępne wydarzenia.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Użytkownik otwiera inną stronę / podstronę przeznaczoną do prezentacji wydarzeń w aplikacji webowej lub mobilnej.</li><li>System prezentuje użytkownikowi dostępne wydarzenia.</li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada bądź nie ma dostępu do bazy danych.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.13. Przypadek użycia – Wyszukiwanie dostępnych wydarzeń (MUZ).

<b>ID: MUZ_UA2</b>
<b>Nazwa: Wyszukiwanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany chce wyszukać dostępne wydarzenia. W tym celu wchodzi na odpowiednią stronę. System prezentuje użytkownikowi odpowiednie wyniki.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>Użytkownik chce skorzystać z funkcji wyszukiwania dostępnych wydarzeń.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany w systemie.</li><li>Użytkownik znajduje się na stronie umożliwiającej wyszukiwanie wydarzeń.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>Użytkownik otrzymuje domyślne wyniki wyszukiwania wydarzeń.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik odwiedza stronę umożliwiającą wyszukiwanie i filtrację wydarzeń.</li><li>Użytkownik nie używa filtracji ani sortowania.</li><li>System domyślnie prezentuje użytkownikowi wszystkie dostępne wydarzenia w kolejności od najnowszego.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Użytkownik używa jednego filtra lub kombinacji filtrów [MUZ_UA3].<ol style="list-style-type: none"><li>System prezentuje spersonalizowane wyniki wyszukiwania wg. zastosowanej filtracji.</li></ol></li><li>Użytkownik używa jednej z dostępnych opcji sortowania [MUZ_UA4].<ol style="list-style-type: none"><li>System prezentuje wyniki wg. wybranej opcji sortowania.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>System nie może przetworzyć zapytania wyszukiwania.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.14. Przypadek użycia – Filtracja dostępnych wydarzeń (MUZ).

<b>ID: MUZ_UA3</b>
<b>Nazwa: Filtracja dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany chce wyszukać tylko konkretne wydarzenia spełniające warunki filtracji. W tym celu wybiera pożądane filtry z dostępnych i kliką przycisk "Filtruj".
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik zalogowany pragnie skorzystać z filtracji.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany.</li><li>Użytkownik znajduje się na stronie oferującą możliwość filtracji wyszukiwanych wydarzeń.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System prezentuje użytkownikowi wydarzenia spełniające kryteria filtracji.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik wybiera jedną opcję filtracji.</li><li>System prezentuje wyniki spełniające kryteria.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Użytkownik używa kombinacji filtrów.<ol style="list-style-type: none"><li>System prezentuje spersonalizowane wyniki wyszukiwania wg. zastosowanej filtracji.</li></ol></li><li>Użytkownik usuwa lub zmienia filtry do wartości domyślnych.<ol style="list-style-type: none"><li>System prezentuje domyślne (wszystkie wydarzenia) wyniki wyszukiwania.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>System nie może przetworzyć zapytania dotyczącego filtrowania.</li><li>Brak wyników spełniających kryteria.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.15. Przypadek użycia – Sortowanie dostępnych wydarzeń (MUZ).

<b>ID: MUZ_UA4</b>
<b>Nazwa: Sortowanie dostępnych wydarzeń</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany chce posortować już wyszukane wyniki. W tym celu wybiera pożądaną opcję sortowania z wysuwanego menu zawierającego dostępne opcje sortowania.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik zalogowany pragnie posortować wyniki wyszukiwania.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik jest zalogowany.</li><li>2. Użytkownik znajduje się na stronie oferującą możliwość sortowania dostępnych wyników.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System prezentuje użytkownikowi posortowane wyniki.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik wybiera jedną z dostępnych opcji sortowania.</li><li>2. System prezentuje wyniki posortowane wg. kryterium sortowania.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>• brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. System nie może przetworzyć zapytania dotyczącego filtrowania.</li><li>3. Brak wyników spełniających kryteria.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.16. Przypadek użycia – Wylogowanie się (MUZ).

<b>ID: MUZ_UA5</b>
<b>Nazwa: Wylogowanie się</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany chce się wylogować z systemu. W tym celu naciska odpowiedni przycisk w menu.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik zalogowany pragnie się wylogować.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany.</li><li>Użytkownik znajduje się na stronie oferującą możliwość wylogowania się.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>Sesja użytkownika zostaje zakończona, użytkownik zostaje wylogowany z systemu.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik kliką przycisk dostępny w menu na jednej ze stron oferujących dostęp do menu.</li><li>System wylogowuje użytkownika z systemu.</li><li>System przekierowuje użytkownika na stronę główną.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>System nie może przetworzyć zapytania dotyczącego wylogowania go z systemu.</li><li>Sesja użytkownika nie zostaje poprawnie usunięta.</li><li>Użytkownik traci dostęp do dodatkowych funkcji strony, jednak jego sesja wciąż jest otwarta lub wciąż dostępny jest jego token.</li><li>System nie jest w stanie usunąć sesji użytkownika.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>dostęp do sieci</li></ul>

Tab. 2.17. Przypadek użycia – Zarządzanie koszykiem (MUZ).

<b>ID: MUZ_UA6</b>
<b>Nazwa: Zarządzanie koszykiem</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany <b>Aktorzy pomocniczy:</b> System bankowości <b>Poziom:</b> Użytkownik <b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany pragnie przeglądać swój koszyk lub wykonać inne interakcje dostępne z jego poziomu. W tym celu klika ikonę lub przycisk kierujący go do strony z jego koszykiem.
<b>Wyzwacze:</b> 1. Użytkownik zalogowany pragnie zarządzać swoim koszykiem.
<b>Warunki początkowe:</b> 1. Użytkownik jest zalogowany. 2. Użytkownik znajduje się na stronie umożliwiającej zarządzanie koszykiem.
<b>Warunki końcowe:</b> 1. System prezentuje koszyk użytkownika.
<b>Scenariusz główny:</b> 1. Użytkownik klika przycisk lub ikonę pozwalającą przejść do koszyka. 2. System przekierowuje użytkownika do koszyka. 3. System prezentuje użytkownikowi zawartość jego koszyka.
<b>Scenariusz alternatywny i rozszerzenia:</b> 4.A. Użytkownik klika czerwony przycisk znajdujący się obok biletu w jego koszyku [MUZ_UA7]. 4.A.1. System usuwa obiekt (bilet) z koszyka użytkownika. 4.A.2. Koniec przypadku użycia. 4.B. Użytkownik klika przycisk "Kup bilety" [MUZ_UA8]. 4.B.1. System przekierowuje użytkownika do strony z podsumowaniem zamówienia. 4.B.2. System prezentuje podsumowanie zamówienia [MUZ_UA9].
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Aplikacja serwerowa nie jest w stanie usunąć biletu. 3. Użytkownik nie posiada żadnych biletów. 4. Użytkownik próbuje usunąć nieistniejący obiekt. 5. Użytkownik próbuje kupić nieistniejący bilet.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.18. Przypadek użycia – Usuwanie biletu z koszyka (MUZ).

<b>ID: MUZ_UA7</b>
<b>Nazwa: Usuwanie biletu z koszyka</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany pragnie usunąć bilet ze swojego koszyka. W tym celu klikna na przycisk symbolizujący kosz. System usuwa obiekt z koszyka użytkownika.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik zalogowany pragnie usunąć wybrany bilet ze swojego koszyka.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany.</li><li>Użytkownik znajduje się na stronie swojego koszyka.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System usuwa wybrany bilet z koszyka użytkownika.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik klikna przycisk lub ikonę pozwalającą przejść do koszyka.</li><li>System usuwa referencję do obiektu w koszyku użytkownika.</li><li>System prezentuje użytkownikowi zaktualizowaną zawartość koszyka.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>Aplikacja serwerowa nie jest w stanie usunąć biletu.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>dostęp do sieci</li></ul>

Tab. 2.19. Przypadek użycia – Kupno biletu (MUZ).

<b>ID: MUZ_UA8</b>
<b>Nazwa: Kupno biletu</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany <b>Aktorzy pomocniczy:</b> System bankowości <b>Poziom:</b> Użytkownik <b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany pragnie kupić bilety znajdujące się w jego koszyku. W tym celu klikna na przycisk "Kup bilety". Użytkownik zostaje przekierowany do podsumowania zamówienia.
<b>Wyzwacze:</b> 1. Użytkownik zalogowany pragnie zakupić bilety na wydarzenia.
<b>Warunki początkowe:</b> 1. Użytkownik jest zalogowany. 2. Użytkownik znajduje się na stronie swojego koszyka.
<b>Warunki końcowe:</b> 1. System prezentuje podsumowanie zamówienia.
<b>Scenariusz główny:</b> 1. Użytkownik klikna przycisk "Kup bilety". 2. System przekierowuje użytkownika do strony z podsumowaniem zamówienia [MUZ_UA9]. 3. System prezentuje podsumowanie zamówienia.
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>• brak</li></ul>
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.20. Przypadek użycia – Podsumowanie zamówienia (MUZ).

<b>ID: MUZ_UA9</b>
<b>Nazwa: Podsumowanie zamówienia</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> System bankowości
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Wysoki
<b>Opis:</b> Użytkownik zalogowany pragnie złożyć zamówienie. W tym celu wypełnia formularz podsumowania zamówienia i naciska odpowiedni przycisk.
<b>Wyzwalacze:</b> 1. Użytkownik zalogowany pragnie zakupić bilety na wydarzenia.
<b>Warunki początkowe:</b> 1. Użytkownik jest zalogowany. 2. Użytkownik znajduje się na stronie podsumowania zamówienia.
<b>Warunki końcowe:</b> 1. System wysyła potwierdzenie zakupu na email użytkownika.
<b>Scenariusz główny:</b> 1. Użytkownik wypełnia formularz podsumowania zamówienia. 2. System przekierowuje użytkownika do strony z płatnością (system bankowości). 3. System bankowości realizuje płatność. 4. System bankowości zwraca potwierdzenie realizacji płatności systemowi SSB. 5. System prezentuje komunikat potwierdzający płatność i wysyła potwierdzenie na email użytkownika.
<b>Scenariusz alternatywny i rozszerzenia:</b> 1.A. Użytkownik kliką przycisk "Powrót". 1.A.1 System przekierowuje użytkownika z powrotem do strony z koszykiem. 1.A.2. Koniec przypadku użycia. 4.A. System bankowości odrzuca płatność użytkownika. 4.A.1. System bankowości wysyła komunikat o odrzuceniu płatności systemowi SSB. 4.A.2. System SSB informuje użytkownika o niepowodzeniu transakcji. 4.A.3. System SSB przekierowuje użytkownika do strony z koszykiem. 4.A.4. Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania. 3. System bankowości nie jest w stanie przetworzyć płatności. 4. System bankowości odrzucił płatność użytkownika.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.21. Przypadek użycia – Dodawanie biletu do koszyka (MUZ).

<b>ID: MUZ_UA10</b>
<b>Nazwa: Dodawanie biletu do koszyka</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik zalogowany pragnie dodać wybrane wydarzenie do swojego koszyka. W tym celu klika przycisk "Dodaj do koszyka" znajdujący się obok wybranego biletu.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik zalogowany pragnie dodać bilet do koszyka.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik jest zalogowany.</li><li>2. Użytkownik znajduje się na stronie wydarzenia lub innej oferującej dodawanie biletów do koszyka.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System dodaje nową referencję obiektu (bilet) do obiektu koszyk użytkownika.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik wchodzi na stronę wydarzenia.</li><li>2. Użytkownik klika przycisk "Dodaj do koszyka" obok pożdanego biletu.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>• brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>3. Dodawany bilet wygasł lub został usunięty.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.22. Przypadek użycia – Dodawanie wydarzenia do ulubionych (MUZ).

<b>ID: MUZ_UA11</b>
<b>Nazwa: Dodanie wydarzenia do ulubionych</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie dodać wybrane wydarzenie do ulubionych. W tym celu naciska ikonę symbolizującą serce znajdująca się na stronie wydarzenia.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik zalogowany pragnie polubić wydarzenie.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany.</li><li>Użytkownik znajduje się na stronie wybranego wydarzenia.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System dodaje id obiektu polubionego wydarzenia do listy w obiekcie użytkownika.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik wchodzi na stronę wydarzenia.</li><li>Użytkownik klika na niewypełnioną kolorem ikonę symbolizującą serce.</li><li>System dodaje id obiektu wydarzenia do listy w obiekcie użytkownik.</li><li>System prezentuje użytkownikowi zmienioną (wypełnioną kolorem czerwonym) ikonę serca.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>2.A. Użytkownik klika na już wypełnioną kolorem ikonę serca.<ol style="list-style-type: none"><li>2.A.1. System usuwa id obiektu wydarzenia z listy w obiekcie użytkownik.</li><li>2.A.2. System prezentuje użytkownikowi zmienioną (niewypełnioną kolorem czerwonym) ikonę serca.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>Wydarzenie wygasło bądź nie istnieje podczas próby polubienia / odznaczenia polubienia.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.23. Przypadek użycia – Dodawanie wydarzenia do obserwowanych (MUZ).

<b>ID: MUZ_UA12</b>
<b>Nazwa:</b> <b>Dodanie wydarzenia do obserwowanych.</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie dodać wybrane wydarzenie do obserwowanych. W tym celu naciska ikonę symbolizującą dzwonek znajdującą się na stronie wydarzenia.
<b>Wyzwacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik zalogowany pragnie obserwować wydarzenie.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik jest zalogowany.</li><li>2. Użytkownik znajduje się na stronie wybranego wydarzenia.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System dodaje id obiektu obserwowanego wydarzenia do listy w obiekcie użytkownika.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik wchodzi na stronę wydarzenia.</li><li>2. Użytkownik klikna na niewypełnioną kolorem ikonę symbolizującą dzwonek.</li><li>3. System dodaje id obiektu wydarzenia do listy w obiekcie użytkownika.</li><li>4. System prezentuje użytkownikowi zmienioną (wypełnioną kolorem) ikonę dzwonka.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>2.A. Użytkownik klikna na już wypełnioną kolorem ikonę dzwonka.<ol style="list-style-type: none"><li>2.A.1. System usuwa id obiektu wydarzenia z listy w obiekcie użytkownika.</li><li>2.A.2. System prezentuje użytkownikowi zmienioną (niewypełnioną kolorem) ikonę dzwonka.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>3. Wydarzenie wygasło bądź nie istnieje podczas próby obserwowania / odznaczenia obserwowania.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.24. Przypadek użycia – Przeglądanie ulubionych wydarzeń (MUZ).

<b>ID: MUZ_UA13</b>
<b>Nazwa: Przeglądanie ulubionych wydarzeń.</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie przeglądać ulubione wydarzenia. W tym celu klikna przycisk "Ulubione" w menu dostępnym z poziomu strony aplikacji webowej lub mobilnej.
<b>Wyzwalacze:</b> 1. Użytkownik zalogowany pragnie przeglądać ulubione wydarzenie.
<b>Warunki początkowe:</b> 1. Użytkownik jest zalogowany. 2. Użytkownik znajduje się na stronie oferującej dostęp do menu.
<b>Warunki końcowe:</b> 1. System prezentuje użytkownikowi ulubione wydarzenia w postaci tabelki.
<b>Scenariusz główny:</b> 1. Użytkownik znajduje się na stronie oferującej dostęp do menu. 2. Użytkownik klikna przycisk "Ulubione" dostępny w menu. 3. System przekierowuje użytkownika na podstronę ulubione. 4. System prezentuje użytkownikowi wydarzenia dodane przez niego do ulubionych.
<b>Scenariusz alternatywny i rozszerzenia:</b> 4.A. Użytkownik nie posiada żadnych wydarzeń dodanych do ulubionych. 4.A.1. System prezentuje monit o braku wydarzeń dodanych do ulubionych. 5.A. Użytkownik klikna ikonę serca w rzędzie opatrzonym nazwą Usuń z ulubionych. 5.A.1. System usuwa wydarzenie z ulubionych [MUZ_UA14]. 5.A.2. System prezentuje zaktualizowaną listę ulubionych.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania. 3. Wydarzenie wygasło bądź nie istnieje podczas próby polubienia / odznaczenia polubienia.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.25. Przypadek użycia – Usunięcie z ulubionych(MUZ).

<b>ID: MUZ_UA14</b>
<b>Nazwa: Usunięcie z ulubionych.</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie usunąć wydarzenie z ulubionych wydarzeń. W tym celu klikna na ikonę symbolizującą serce znajdująca się obok polubionego wydarzenia na stronie ulubionych wydarzeń.
<b>Wyzwalače:</b> <ol style="list-style-type: none"><li>1. Użytkownik zalogowany pragnie usunąć wydarzenie z ulubionych wydarzeń.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>2. Użytkownik jest zalogowany.</li><li>3. Użytkownik znajduje się na stronie z ulubionymi wydarzeniami.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System usuwa wydarzenie z ulubionych.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik znajduje się na stronie z ulubionymi wydarzeniami.</li><li>2. Użytkownik klikna ikonę symbolizującą serce obok wydarzenia, które chce usunąć z ulubionych.</li><li>3. System usuwa referencję do obiektu wydarzenie z listy ulubionych wydarzeń użytkownika.</li><li>4. System prezentuje użytkownikowi zaktualizowaną listę ulubionych wydarzeń.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>• brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>3. Wydarzenie wygasło bądź nie istnieje podczas próby usunięcia z ulubionych.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.26. Przypadek użycia – Przeglądanie obserwowanych wydarzeń (MUZ).

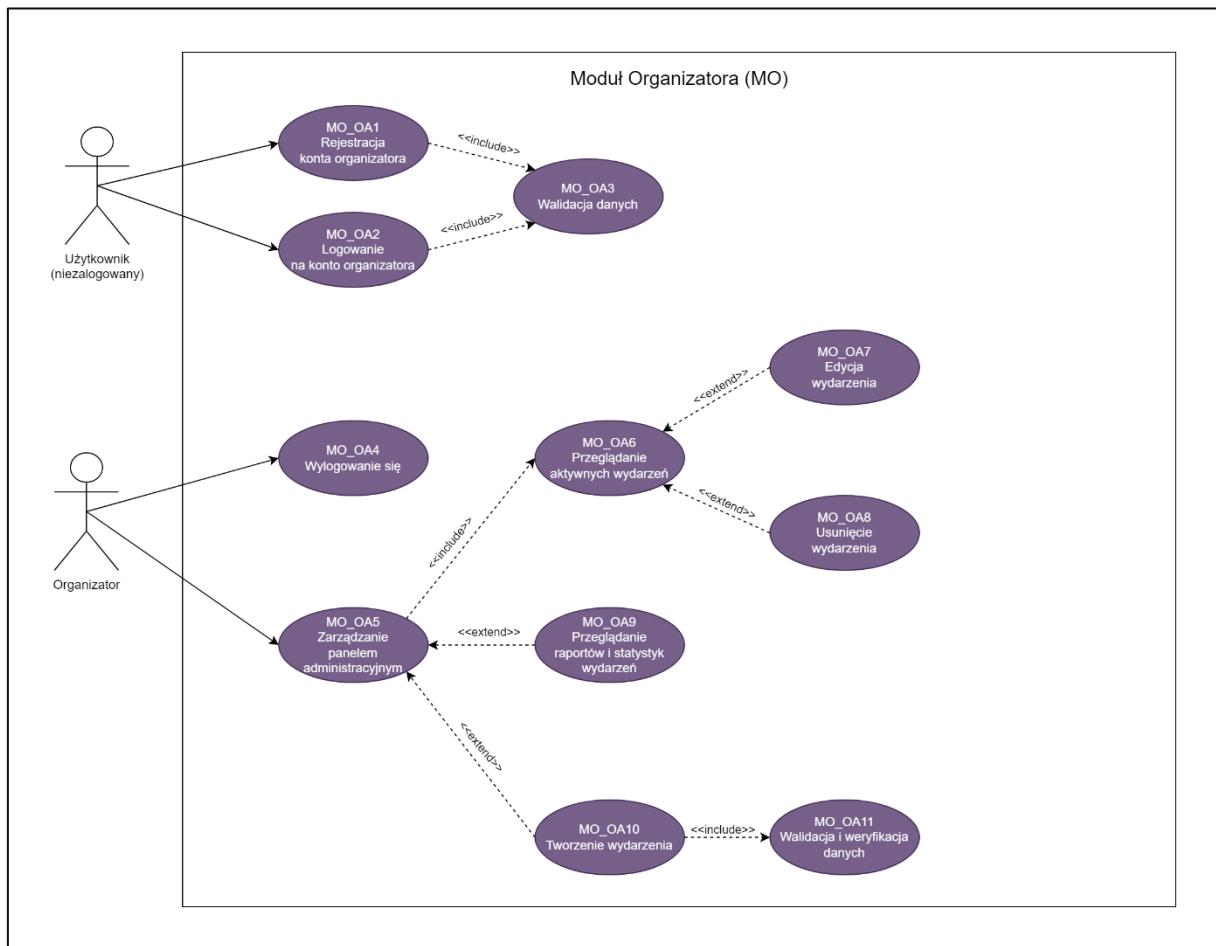
<b>ID: MUZ_UA15</b>
<b>Nazwa: Przeglądanie obserwowanych wydarzeń.</b>
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie przeglądać obserwowane wydarzenia. W tym celu klikna na przycisk "Obserwowane" w menu dostępnym z poziomu strony aplikacji webowej lub mobilnej.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Użytkownik zalogowany pragnie przeglądać obserwowane wydarzenie.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Użytkownik jest zalogowany.</li><li>Użytkownik znajduje się na stronie oferującej dostęp do menu.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System prezentuje użytkownikowi obserwowane wydarzenia w postaci tabelki.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Użytkownik znajduje się na stronie oferującej dostęp do menu.</li><li>Użytkownik klikna na przycisk "Obserwowane" dostępny w menu.</li><li>System przekierowuje użytkownika na podstronę obserwowanych wydarzeń.</li><li>System prezentuje użytkownikowi wydarzenia dodane przez niego do obserwowanych.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Użytkownik nie posiada żadnych wydarzeń dodanych do obserwowanych.<ol style="list-style-type: none"><li>System prezentuje monit o braku wydarzeń dodanych do obserwowanych.</li></ol></li><li>Użytkownik klikna ikonę dzwonka w rzędzie opatrzonym nazwą Usuń z obserwowanych.<ol style="list-style-type: none"><li>System usuwa wydarzenie z obserwowanych [MUZ_UA16].</li><li>System prezentuje zaktualizowaną listę ulubionych.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>Wydarzenie wygasło bądź nie istnieje podczas próby obserwowania / odznaczenia obserwowania.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.27. Przypadek użycia – Usunięcie z obserwowanych. (MUZ).

<b>ID: MUZ_UA16</b>
<b>Nazwa:</b> Usunięcie z obserwowanych.
<b>Aktorzy główni:</b> Użytkownik zalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik zalogowany pragnie usunąć wydarzenie z obserwowanych wydarzeń. W tym celu klikna ikonę symbolizującą dzwonek znajdującą się obok polubionego wydarzenia na stronie obserwowanych wydarzeń.
<b>Wyzwalače:</b> <ol style="list-style-type: none"><li>1. Użytkownik zalogowany pragnie usunąć wydarzenie z obserwowanych wydarzeń.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>1. Użytkownik jest zalogowany.</li><li>2. Użytkownik znajduje się na stronie z obserwowanymi wydarzeniami.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>1. System usuwa wydarzenie z obserwowanych.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>1. Użytkownik znajduje się na stronie z obserwowanymi wydarzeniami.</li><li>2. Użytkownik klikna ikonę symbolizującą dzwonek obok wydarzenia, które chce usunąć z obserwowanych.</li><li>3. System usuwa referencję do obiektu wydarzenie z listy obserwowanych wydarzeń użytkownika.</li><li>4. System prezentuje użytkownikowi zaktualizowaną listę obserwowanych wydarzeń.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>• brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>2. Aplikacja serwerowa nie jest w stanie przetworzyć zapytania.</li><li>3. Wydarzenie wygasło bądź nie istnieje podczas próby usunięcia z obserwowanych.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

#### 2.4.2. Sekcja administracyjna

W tym podrozdziale skupiono się na przedstawieniu i opisaniu diagramu przypadków użycia związanych z sekcją administracyjną. Sekcja administracyjna składa się z jednego głównego modułu: modułu organizatora (MO) przedstawionego na rysunku 2.4. Diagram w głównej mierze reprezentuje interakcje zachodzące między organizatorami, a systemem SSB. Diagram uwzględnia również interakcje organizatorów jako użytkowników niezalogowanych jeszcze do systemu jako organizatorzy, a systemem SSB. Poszczególne przypadki użycia należące do tej sekcji zostały przedstawione w tabelach od 2.28. – 2.38.



Rys. 2.4. Diagram przypadków użycia MO. (źródło: Opracowanie własne)

Tab. 2.28. Przypadek użycia – Rejestracja konta organizatora (MO).

<b>ID: MO_OA1</b>
<b>Nazwa: Rejestracja konta organizatora</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik/Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Użytkownik niezalogowany chce utworzyć konto organizatora w systemie, aby móc korzystać z funkcji i panelu organizatora. Użytkownik wypełnia formularz rejestracyjny. Jeżeli formularz został wypełniony poprawnie, a aplikacja serwerowa nie zgłosi żadnego błędu użytkownik pomyślnie utworzy nowe konto organizatora w systemie SSB.
<b>Wyzwacze:</b> 1. Użytkownik niezalogowany pragnie utworzyć nowe konto organizatora w systemie.
<b>Warunki początkowe:</b> 1. Użytkownik nie jest zalogowany w systemie. 2. Użytkownik znajduje się na stronie do rejestracji konta organizatora.
<b>Warunki końcowe:</b> 1. Nowy obiekt (organizator) zostaje utworzony w systemie.
<b>Scenariusz główny:</b> 1. Użytkownik niezalogowany klika na przycisk "Jesteś organizatorem?" znajdującym się w menu na stronie aplikacji mobilnej lub webowej. 2. Użytkownik zostaje przekierowany do strony z formularzem rejestracyjnym organizatora. 3. Użytkownik poprawnie wypełnia wszystkie wymagane pola. 4. Użytkownik klika na przycisk "Utwórz konto organizatora" znajdującym się pod formularzem. 5. System weryfikuje dane wprowadzone w pola po stronie aplikacji webowej. 6. System waliduje dane po stronie aplikacji serwerowej [MO_OA3]. 7. System tworzy nowy obiekt (organizator) w bazie danych. 8. System informuje użytkownika o poprawnym utworzeniu konta.
<b>Scenariusz alternatywny i rozszerzenia:</b> 5.A. System wykrywa, że użytkownik wprowadził niepoprawny adres e-mail. 5.A.1. System wyświetla komunikat o niepoprawnym e-mailu i prosi o jego poprawienie. 5.A.2. Należy powtórzyć kroki począwszy od kroku 3. 5.B. System wykrywa, że hasło organizatora nie zgadza się z powtórzonym hasłem lub nie spełnia polityki silnych haseł. 5.B.1. System wyświetla komunikat o zbyt słabym haśle lub o tym, że hasło i powtórzone hasło się nie zgadzają. System prosi o zmianę. 5.B.2. Należy powtórzyć kroki począwszy od kroku 3. 5.C. System wykrywa, że przynajmniej jedno z wymaganych pól jest puste. 5.C.1. System wyświetla komunikat o konieczności wypełnienia brakujących wymaganych pól.

5.C.2. Należy powtórzyć kroki począwszy od kroku 3.

- 6.A. System wykrywa, że podany adres e-mail jest już zarejestrowany w systemie.
  - 6.A.1. System wyświetla komunikat informujący o konieczności wyboru innego adresu e-mail.
  - 6.A.2. Należy powtórzyć kroki począwszy od kroku 3.
- 6.B. System wykrywa, że podana nazwa użytkownika jest już zajęta w systemie.
  - 6.B.1. System wyświetla komunikat informujący o konieczności wyboru innej nazwy użytkownika.
  - 6.B.2. Należy powtórzyć kroki począwszy od kroku 3.
- 6.C. System wykrywa, że podana nazwa organizatora jest już zajęta.
  - 6.C.1. System wyświetla komunikat informujący o konieczności wyboru innej nazwy organizatora.
  - 6.C.2. Należy powtórzyć kroki począwszy od kroku 3.

**Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego rejestracji.
3. Organizator o podanej nazwie już istnieje.
4. Podany email już istnieje.
5. Przynajmniej jedno z wymaganych pól formularza jest puste.

**Dodatkowe wymagania:**

- podczas tworzenia nowego obiektu (organizator) w bazie danych automatycznie tworzone jest jego id
- podczas tworzenia nowego obiektu (organizator) w bazie danych równolegle tworzony jest obiekt "password" przechowujący w postaci nie jawniej, zaszyfrowane funkcją kryptograficzną, hasło organizatora.

Tab. 2.29. Przypadek użycia – Logowanie na konto organizatora (MO).

<b>ID: MO_OA2</b>
<b>Nazwa: Logowanie na konto organizatora</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Użytkownik
<b>Priorytet:</b> Niski
<b>Opis:</b> Użytkownik niezalogowany chce zalogować się na konto organizatora w systemie, aby uzyskać dostęp do dodatkowych funkcji i panelu organizatora. W tym celu musi wybrać opcję "Zaloguj się" dostępną w menu w aplikacji webowej.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>1. Użytkownik pragnie zalogować się na swoje konto.</li></ol>

**Warunki początkowe:**

1. Użytkownik jest niezalogowany
2. Użytkownik posiada aktywne konto organizatora w systemie.
3. Użytkownik znajduje się na stronie z formularzem logowania na konto organizatora.

**Warunki końcowe:**

1. Użytkownik niezalogowany loguje się do systemu stając się jednocześnie organizatorem.

**Scenariusz główny:**

1. Użytkownik niezalogowany klika na przycisk "Zaloguj się" znajdujący się w menu na stronie aplikacji webowej.
2. Użytkownik zostaje przekierowany do strony z formularzem do logowania.
3. Użytkownik poprawnie wypełnia wszystkie wymagane pola.
4. Użytkownik klika na przycisk "Zaloguj się" znajdującym się pod formularzem.
5. System weryfikuje dane wprowadzone w pola po stronie aplikacji webowej.
6. System waliduje dane po stronie aplikacji serwerowej [MO\_OA3].
7. System tworzy nową sesję organizatora nadając mu aktywny token JWT.
8. Użytkownik niezalogowany staje się organizatorem i od teraz może korzystać z panelu organizatora oraz dodatkowych funkcji organizatora.
9. System prezentuje organizatorowi stronę z panelem organizatora.

**Scenariusz alternatywny i rozszerzenia:**

- 5.A. System wykrywa, że przynajmniej jedno z wymaganych pól jest puste.
  - 5.A.1. System wyświetla komunikat o konieczności wypełnienia brakujących wymaganych pól.
  - 5.A.2. Należy powtórzyć kroki począwszy od kroku 3.
- 6.A. System wykrywa, że użytkownik wprowadził niepoprawne hasło bądź nazwę organizatora.
  - 6.A.1. System wyświetla komunikat o błędnej autoryzacji.
  - 6.A.2. Należy powtórzyć kroki począwszy od kroku 3.

**Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego logowania.
3. Dane logowania są niepoprawne.
4. Organizator nie istnieje.
5. Obiekt organizator jest uszkodzony bądź niekompletny.

**Dodatkowe wymagania:**

- dostęp do sieci

Tab. 2.30. Przypadek użycia – Walidacja danych (MO).

<b>ID: MO_OA3</b>
<b>Nazwa:</b> <b>Walidacja danych</b>
<b>Aktorzy główni:</b> Użytkownik niezalogowany
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> System
<b>Priorytet:</b> Wysoki
<b>Opis:</b> Użytkownik niezalogowany wypełnia formularz i przesyła go dalej, system sprawdza poprawność danych formularza.
<b>Wyzwalacze:</b> 1. Użytkownik chce przesłać wypełniony formularz.
<b>Warunki początkowe:</b> 1. Użytkownik nie jest zalogowany w systemie. 2. Użytkownik znajduje się na stronie rejestracji lub logowania. 3. Użytkownik wzywa funkcję przesyłającą formularz.
<b>Warunki końcowe:</b> 1. Dane formularza zostają zwalidowane i zweryfikowane, a system przesyła odpowiedź do klienta.
<b>Scenariusz główny:</b> 2. Użytkownik wypełnia formularz i przesyła go do weryfikacji. 3. System waliduje i weryfikuje dane. 4. System tworzy nowy obiekt organizator w systemie lub tworzy nową sesję organizatora i loguje go do systemu. 5. System zwraca informację o wyniku walidacji.
<b>Scenariusz alternatywny i rozszerzenia:</b> 3.A. Dane rejestracji nie zostały poprawnie zwalidowane lub zweryfikowane. 3.A.1. System nie tworzy nowego obiektu. 3.A.2. System zwraca komunikat o błędzie. 3.A.3. Koniec przypadku użycia. 3.B. Dane logowania nie zostały poprawnie zwalidowane lub zweryfikowane. 3.B.1. System nie tworzy nowej sesji organizatora i nie loguje go do systemu. 3.B.2. System zwraca komunikat o błędzie. 3.B.3 Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. System nie może przetworzyć zapytania dotyczącego walidacji danych. 3. Dane logowania są błędne. 4. Dane rejestracji są błędne. 5. Wymagane pola są puste. 6. Hasło nie spełnia polityki silnych haseł. 7. Hasło i powtórz hasło nie są identyczne.

- |  |
|--|
| <p>8. Próba rejestracji adresu e-mail lub nazwy organizatora, który już istnieje.</p> <p>9. Hasło nie zgadza się z hasłem w bazie danych.</p> <p>10. Nazwa organizatora nie zgadza się z nazwą w bazie danych.</p> |
|--|

**Dodatkowe wymagania:**

- dostęp do sieci

*Tab. 2.31. Przypadek użycia – Wylogowanie się (MO).*

**ID: MO\_OA4**

**Nazwa:** Wylogowanie się.

**Aktorzy główni:** Organizator

**Aktorzy pomocniczy:** brak

**Poziom:** Organizator

**Priorytet:** Średni

**Opis:**

Organizator chce się wylogować z systemu. W tym celu naciska odpowiedni przycisk w menu.

**Wyzwalače:**

1. Organizator pragnie się wylogować.

**Warunki początkowe:**

1. Organizator jest zalogowany.
2. Organizator znajduje się na stronie oferującą możliwość wylogowania się.

**Warunki końcowe:**

1. Sesja organizatora zostaje zakończona
2. Organizator zostaje wylogowany z systemu.

**Scenariusz główny:**

1. Organizator kliką przycisk dostępny w menu na jednej ze stron oferujących dostęp do menu.
2. System wylogowuje organizatora z systemu.
3. System przekierowuje organizatora na stronę główną.

**Scenariusz alternatywny i rozszerzenia:**

- brak

**Wyjątki:**

1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
2. System nie może przetworzyć zapytania dotyczącego wylogowania organizatora z systemu.
3. Sesja organizatora nie zostaje poprawnie usunięta.
4. Organizator traci dostęp do dodatkowych funkcji strony, jednak jego sesja wciąż jest otwarta lub wciąż dostępny jest jego token.
5. System nie jest w stanie usunąć sesji organizatora.

**Dodatkowe wymagania:**

- dostęp do sieci

Tab. 2.32. Przypadek użycia – Zarządzanie panelem administracyjnym (MO).

<b>ID: MO_OA5</b>
<b>Nazwa:</b> Zarządzanie panelem administracyjnym.
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce zarządzać wydarzeniami. W tym celu system po zalogowaniu organizatora do systemu prezentuje mu panel administracyjny w którym może przeglądać utworzone wydarzenia, tworzyć nowe wydarzenia, edytować wydarzenia lub przeglądać raporty i statystyki wydarzeń.
<b>Wyzwalače:</b> 1. Organizator chce zarządzać swoimi wydarzeniami.
<b>Warunki początkowe:</b> 1. Organizator jest zalogowany.
<b>Warunki końcowe:</b> 1. System udostępnia organizatorowi szereg funkcji do zarządzania wydarzeniami.
<b>Scenariusz główny:</b> 1. Organizator przechodzi na stronę główną panelu administracyjnego lub zostaje przekierowany automatycznie po zalogowaniu do systemu. 2. Domyślnie system prezentuje organizatorowi pierwszy panel administracyjny – "Aktywne wydarzenia". 3. Organizator przegląda dostępne wydarzenia [MO_OA6].
<b>Scenariusz alternatywny i rozszerzenia:</b> 3.A. Organizator klika przycisk / etykietę "Raporty i statystyki". 3.A.1. System prezentuje zawartość panelu "Raporty i statystyki" [MO_OA9]. 3.A.2. Koniec przypadku użycia. 3.B. Organizator klika przycisk / etykietę "Stwórz nowe wydarzenie" [MO_OA10]. 3.B.1. System prezentuje zawartość panelu "Stwórz nowe wydarzenie". 3.B.2. Koniec przypadku użycia. 3.C. Organizator klika przycisk / etykietę "Aktywne wydarzenia" [MO_OA6]. 3.C.1. System prezentuje zawartość panelu "Aktywne wydarzenia". 3.C.2. Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.33. Przypadek użycia – Przeglądanie aktywnych wydarzeń (MO).

<b>ID: MO_OA6</b>
<b>Nazwa: Przeglądanie aktywnych wydarzeń</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce przeglądać aktywne wydarzenia. W tym celu klika na przycisk / etykietę "Aktywne wydarzenia". System prezentuje zawartość panelu.
<b>Wyzwalače:</b> 1. Organizator chce przeglądać aktywne wydarzenia.
<b>Warunki początkowe:</b> 1. Organizator jest zalogowany.
<b>Warunki końcowe:</b> 1. System prezentuje organizatorowi zawartość panelu "Aktywne wydarzenia".
<b>Scenariusz główny:</b> 1. Organizator klika przycisk / etykietę "Aktywne wydarzenia". 2. System prezentuje organizatorowi zawartość panelu "Aktywne wydarzenia". 3. Organizator przegląda dostępne wydarzenia.
<b>Scenariusz alternatywny i rozszerzenia:</b> 2.A. System wykrywa, że organizator nie posiada żadnych utworzonych wydarzeń. 2.A.1. System prezentuje informację o braku utworzonych wydarzeń. 2.A.2. Koniec przypadku użycia. 3.A. Organizator klika przycisk symbolizujący pióro. 3.A.1. System prezentuje organizatorowi formularz edycji wydarzenia [MO_OA7]. 3.A.2. Koniec przypadku użycia. 3.B. Organizator klika przycisk symbolizujący kosz. 3.B.1. System prezentuje monit o usunięcie wydarzenia [MO_OA8]. 3.B.2. Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Organizator nie posiada żadnych wydarzeń.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.34. Przypadek użycia – Edycja wydarzeń (MO).

<b>ID: MO_OA7</b>
<b>Nazwa:</b> <b>Edycja wydarzenia</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce edytować istniejące wydarzenie. W tym celu klikna na ikonę symbolizującą pióro. System prezentuje zawartość strony umożliwiającą edycję wydarzenia.
<b>Wyzwalacze:</b> 1. Organizator chce edytować istniejące wydarzenie.
<b>Warunki początkowe:</b> 1. Organizator jest zalogowany.
<b>Warunki końcowe:</b> 1. System zmienia wartości pól obiektu (wydarzenie).
<b>Scenariusz główny:</b> 1. Organizator klikna ikonę symbolizującą pióro. 2. System prezentuje zawartość strony umożliwiającą edycję wydarzenia. 3. Organizator edytuje interesujące go pola / wartości. 4. Organizator potwierdza chęć edycji. 5. System weryfikuje i waliduje poprawność formularza edycji wydarzenia. 6. System zmienia wartości pól obiektu w bazie danych. 7. System informuje organizatora o pomyślnym przeprowadzeniu edycji.
<b>Scenariusz alternatywny i rozszerzenia:</b> 3.A. Organizator klikna przycisk "Anuluj". 3.A.1. System przekierowuje organizatora z powrotem do panelu administracyjnego. 3.A.2. Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Walidacja lub weryfikacja nie zostaje przeprowadzona pomyślnie.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.35. Przypadek użycia – Archiwizacja/Usunięcie wydarzenia (MO).

<b>ID: MO_OA8</b>
<b>Nazwa: Usunięcie (archiwizacja? Zmiana statusu zamiast usuwania?) wydarzenia</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce usunąć istniejące wydarzenie. W tym celu klikna na ikonę symbolizującą kosz. System pyta organizatora czy na pewno chce usunąć wydarzenie. Organizator potwierdza tym samym usuwając wydarzenie.
<b>Wyzwacze:</b> 1. Organizator chce usunąć istniejące wydarzenie.
<b>Warunki początkowe:</b> 1. Organizator jest zalogowany.
<b>Warunki końcowe:</b> 1. System usuwa obiekt (wydarzenie).
<b>Scenariusz główny:</b> 1. Organizator klikna ikonę symbolizującą kosz. 2. System pyta organizatora czy na pewno chce usunąć wydarzenie. 3. System usuwa obiekt (wydarzenie) z bazy danych. 4. System informuje organizatora o pomyślnym usunięciu wydarzenia.
<b>Scenariusz alternatywny i rozszerzenia:</b> 2.A. Organizator klikna przycisk "Anuluj". 2.A.1. System zamknie okno z monitem. 2.A.2. Koniec przypadku użycia.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych. 2. Organizator próbuje usunąć wydarzenie, na które jakiś użytkownik kupił już bilet, dodał je do ulubionych lub obserwowanych.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.36. Przypadek użycia – Przeglądanie raportów i statystyk (MO).

<b>ID: MO_OA9</b>
<b>Nazwa: Przeglądanie raportów i statystyk</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce przeglądać statystyki i raporty dotyczące wydarzeń, które utworzył. W tym celu organizator loguje się do systemu i naciska przycisk / etykietę "Raporty i statystyki". System prezentuje organizatorowi panel ze statystykami i rapportami.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Organizator chce przeglądać dostępne raporty i statystyki.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Organizator jest zalogowany.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>System prezentuje panel raportów i statystyk.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Organizator kliką przycisk / etykietę "Raporty i statystyki".</li><li>System prezentuje organizatorowi zawartość panelu "Raporty i statystyki".</li><li>Organizator przegląda dostępne raporty i statystyki.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ul style="list-style-type: none"><li>brak</li></ul>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>dostęp do sieci</li></ul>

Tab. 2.37. Przypadek użycia – Tworzenie wydarzeń (MO).

<b>ID: MO_OA10</b>
<b>Nazwa: Tworzenie wydarzenia</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> Organizator
<b>Priorytet:</b> Średni
<b>Opis:</b> Organizator chce utworzyć nowe wydarzenie. W tym celu loguje się do systemu i kliką przycisk / etykietę "Utwórz nowe wydarzenie". System prezentuje organizatorowi panel z formularzem tworzenia nowego wydarzenia. Organizator wypełnia formularz, a system po poprawnej weryfikacji i walidacji wprowadzonych danych tworzy nowe wydarzenie w systemie dodając do bazy danych nowe obiekty.
<b>Wyzwacze:</b> 1. Organizator chce utworzyć nowe wydarzenie.
<b>Warunki początkowe:</b> 1. Organizator jest zalogowany.
<b>Warunki końcowe:</b> 1. System tworzy nowe wydarzenie.
<b>Scenariusz główny:</b> 1. Organizator kliką przycisk / etykietę "Utwórz nowe wydarzenie". 2. System prezentuje organizatorowi formularz w panelu "Utwórz nowe wydarzenie". 3. Organizator wypełnia formularz. 4. System weryfikuje i waliduje dane formularza [MO_OA11]. 5. System dodaje nowy obiekt/obiekty typu bilet do bazy danych. 6. System dodaje nowy obiekt wydarzenie do bazy danych. 7. System informuje organizatora o utworzeniu nowego wydarzenia.
<b>Scenariusz alternatywny i rozszerzenia:</b> 4.A. Weryfikacja lub walidacja danych nie powodzi się. 4.A.1. System wyświetla komunikat, że weryfikacja lub walidacja się nie powiodła. 4.A.2. Należy powtórzyć kroki od kroku 3.
<b>Wyjątki:</b> 1. Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

Tab. 2.38. Przypadek użycia – Walidacja i weryfikacja danych (MO).

<b>ID: MO_OA11</b>
<b>Nazwa: Walidacja i weryfikacja danych</b>
<b>Aktorzy główni:</b> Organizator
<b>Aktorzy pomocniczy:</b> brak
<b>Poziom:</b> System
<b>Priorytet:</b> Wysoki
<b>Opis:</b> Organizator wypełnia formularz tworzenia wydarzenia i przesyła go dalej, system sprawdza poprawność danych formularza.
<b>Wyzwalacze:</b> <ol style="list-style-type: none"><li>Organizator chce przesłać wypełniony formularz.</li></ol>
<b>Warunki początkowe:</b> <ol style="list-style-type: none"><li>Organizator jest zalogowany w systemie.</li><li>Organizator znajduje się na stronie z formularzem tworzenia wydarzenia.</li><li>Organizator wyzwała funkcję przesyłając formularz.</li></ol>
<b>Warunki końcowe:</b> <ol style="list-style-type: none"><li>Dane formularza zostają zwalidowane i zweryfikowane, a system przesyła odpowiedź do klienta.</li></ol>
<b>Scenariusz główny:</b> <ol style="list-style-type: none"><li>Organizator wypełnia formularz i przesyła go do weryfikacji.</li><li>System waliduje i weryfikuje dane.</li><li>System tworzy nowy obiekt lub obiekty typu bilet.</li><li>System tworzy nowy obiekt typu wydarzenie.</li><li>System zwraca informację o wyniku walidacji.</li></ol>
<b>Scenariusz alternatywny i rozszerzenia:</b> <ol style="list-style-type: none"><li>Dane formularza nie zostały poprawnie zwalidowane lub zweryfikowane.<ol style="list-style-type: none"><li>System nie tworzy nowych obiektów.</li><li>System zwraca komunikat o błędzie.</li><li>Koniec przypadku użycia.</li></ol></li></ol>
<b>Wyjątki:</b> <ol style="list-style-type: none"><li>Aplikacja serwerowa nie odpowiada lub nie ma dostępu do bazy danych.</li><li>System nie może przetworzyć zapytania dotyczącego walidacji danych.</li><li>Dane formularza są błędne.</li><li>Wymagane pola są puste.</li><li>Próba utworzenia wydarzenia bez utworzonych biletów.</li></ol>
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• dostęp do sieci</li></ul>

## 2.5. Wymagania pozafunkcjonalne

W tym podrozdziale skoncentrowano się na zdefiniowaniu cech i parametrów oprogramowania, które wykraczają poza samo działanie funkcjonalne systemu. Wymagania pozafunkcjonalne przedstawione w tabelach 2.39. – 2.47 w tym podrozdziale dotyczą wszelkich rzeczy związanych z tworzonym systemem m.in. od strony jakościowej, wydajnościowej, kompatybilności oraz bezpieczeństwa. Wykorzystane kategorie zgodne są z normą ISO 25010 [17].

Tab. 2.39. Wymagania NFR – NFR\_SSB\_01

<b>Identyfikator</b>	NFR_SSB_01
<b>Kategoria</b>	Poufność
<b>Priorytet</b>	Wysoki
<b>Opis</b>	Wszystkie hasła w systemie muszą być przechowywane w formie nie jawniej, tzn. muszą być zaszyfrowane funkcją mieszającą np. bcrypt.

Tab. 2.40. Wymagania NFR – NFR\_SSB\_02

<b>Identyfikator</b>	NFR_SSB_02
<b>Kategoria</b>	Poufność / Bezpieczeństwo danych
<b>Priorytet</b>	Wysoki
<b>Opis</b>	System powinien być zgodny z obowiązującymi przepisami dotyczącymi ochrony danych osobowych, w tym RODO.

Tab. 2.41. Wymagania NFR – NFR\_SSB\_03

<b>Identyfikator</b>	NFR_SSB_03
<b>Kategoria</b>	Skalowalność
<b>Priorytet</b>	Średni
<b>Opis</b>	System powinien umożliwiać łatwe skalowanie, aby sprostać rosnącemu obciążeniu w miarę wzrostu liczby użytkowników.

Tab. 2.42. Wymagania NFR – NFR\_SSB\_04

<b>Identyfikator</b>	NFR_SSB_04
<b>Kategoria</b>	Dostępność
<b>Priorytet</b>	Średni
<b>Opis</b>	System powinien być dostępny dla użytkowników przez co najmniej 95% czasu w ciągu miesiąca kalendarzowego.

Tab. 2.43. Wymagania NFR – NFR\_SSB\_05

<b>Identyfikator</b>	NFR_SSB_05
<b>Kategoria</b>	Wydajność
<b>Priorytet</b>	Średni
<b>Opis</b>	System powinien być w stanie obsłużyć równoczesne zapytania od tysięcy użytkowników bez znaczącego spadku wydajności.

Tab. 2.44. Wymagania NFR – NFR\_SSB\_06

<b>Identyfikator</b>	NFR_SSB_06
<b>Kategoria</b>	Bezpieczeństwo danych
<b>Priorytet</b>	Wysoki
<b>Opis</b>	System powinien wykorzystywać mechanizm zarządzania sesją użytkownika i autoryzacji dostępu do zasobów np. JWT.

Tab. 2.45. Wymagania NFR – NFR\_SSB\_07

<b>Identyfikator</b>	NFR_SSB_07
<b>Kategoria</b>	Bezpieczeństwo danych
<b>Priorytet</b>	Wysoki
<b>Opis</b>	Wszystkie obiekty w systemie muszą być jednoznacznie identyfikowalne za pomocą unikalnych identyfikatorów.

Tab. 2.46. Wymagania NFR – NFR\_SSB\_08

<b>Identyfikator</b>	NFR_SSB_08
<b>Kategoria</b>	Łatwość utrzymania
<b>Priorytet</b>	Niski
<b>Opis</b>	Komentarze oraz nazwy funkcji / zmiennych powinny być pisane w języku angielskim. Jest to powszechna praktyka w programowaniu, która wspiera czytelność i współpracę w zespołach o różnych językach ojczystych

Tab. 2.47. Wymagania NFR – NFR\_SSB\_09

<b>Identyfikator</b>	NFR_SSB_09
<b>Kategoria</b>	Łatwość utrzymania
<b>Priorytet</b>	Średni
<b>Opis</b>	Kod źródłowy musi być odpowiednio skomentowany, zgodnie z ustalonymi standardami w projekcie, aby ułatwić zrozumienie i modyfikację kodu.

## 2.6. Opis API

W ramach tego podrozdziału przedstawiono szczegółowy opis interfejsu API wraz z jego punktami końcowymi udostępnionymi warstwą aplikacji webowej i mobilnej przez aplikację serwerową. Do stworzenia dokumentacji API użyto narzędzia swagger.io (rys. 2.6. – 2.8.), które ułatwia projektowanie, budowanie, dokumentowanie i korzystanie z interfejsów opartych na rozwiązaniu architektonicznym REST [18]. API zostało podzielone na cztery podgrupy, z których każda zawiera punkty końcowe przeznaczone do obsługi określonych zasobów. Podgrupy te obejmują artystów, użytkowników, bilety oraz wydarzenia. Dodatkowo, wydzielono schematy, które zawierają struktury danych używane w API (rys. 2.5.). Te schematy obejmują wszystkie definicje, takie jak *ticketSchema*, *artistSchema*, *passwordSchema*, itd., które określają format i strukturę danych używanych w operacjach API. Dzięki nim, możliwe jest jednolite i spójne zarządzanie danymi w całym systemie.



Rys. 2.5. Wykaz schematów. (źródło: Opracowanie własne)

## Artists API for managing artists.

GET /api/artists Get all artists

GET /api/artists/{id} Get a single artist by ID

POST /api/artist Create a single artist

GET /api/events/{id}/artists Returns artists objects that participate in given event

## Events API for managing events.

GET /api/events Get all events

GET /api/events/{id} Get a single event by ID

POST /events/transaction Create a single event using transactions

POST /api/event/likes-follows/{eventId}/{userId}/{actionType} Add like or follower to an event

GET /api/event/likes-follows/{eventId}/{actionType} Get likes or followers count for an event

## Tickets API for managing tickets.

GET /api/tickets Get all tickets

GET /api/events/tickets/{id} Get a single ticket by ID

POST /api/events/ticket/{id} Create a single ticket and return object ID

GET /api/events/{id}/tickets Get tickets for a specific event

## Transactions API for managing transactions.

GET /api/transactions Get all transactions

GET /api/transactions/transaction/{id} Get a single transaction by ID

POST /api/transactions/transaction Create a new transaction

GET /api/transactions/all/{userId} Get all transactions for a given user

Rys. 2.6. Wykaz punktów końcowych systemu SSB. (źródło: Opracowanie własne)

Users API for managing users.	
POST	/api/user/auth Authenticate a user
POST	/api/user/create Create a new user
POST	/api/user/update Update an existing user
GET	/api/user/preferences/{userId} Get user preferences by ID
PUT	/api/user/{userId}/preferences/onetimeMonit Update oneTimeMonitChecked flag for a user
DELETE	/api/user/logout/{userId} Logout a user
POST	/api/user/{userId}/cart/add-tickets/{eventId}/{ticketId} Add ticket(s) to user's cart
POST	/api/user/{userId}/cart/remove-ticket/{eventId}/{ticketId} Remove ticket(s) from user's cart
GET	/api/user/{userId}/cart Get user's cart
GET	/api/user/{userId}/preferences Get user's preferences by ID
POST	/api/profile/like-follow/{userId}/{eventId}/{actionType} Like or follow an event
GET	/api/profile/likes-follows/{userId}/{actionType} Get liked or followed events by user
GET	/api/profile/likes-follows/{userId} Get counts of liked and followed events by user
POST	/api/profile/check-if-event-liked/{userId}/{eventId}/{actionType} Check if user liked or followed an event

Rys. 2.7. Wykaz punktów końcowych systemu SSB. (źródło: Opracowanie własne)

Organiser	
GET	/api/organiser/{userId} Get organiser's owned events by ID
POST	/api/organiser/{userId}/add-event/{eventId} Add event to organiser's ownedEvents
GET	/api/organiser/stats/tickets-sold-by-event/{eventId} Get sold tickets for a specific event
GET	/api/organiser/stats/tickets-sold-by-organiser/{organiserName} Get sold tickets for a specific organizer
GET	/api/organiser/stats/total-earnings-by-organiser/{organiserName} Get total earnings for a specific organizer
GET	/api/organiser/stats/total-earnings-by-event/{eventId} Get total earnings for a specific event
GET	/api/organiser/stats/total-views-by-organiser/{organiserName} Get total views earned by all events for a given organiser
GET	/api/organiser/sale-data/{organiserName} Get sale data for charts for all sales by organiser

Rys. 2.8. Wykaz punktów końcowych systemu SSB. (źródło: Opracowanie własne)

Tab. 2.48. Charakterystyka punktów końcowych służących do zarządzania artystami.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
GET	/api/artists	-	brak	Pobiera listę artystów.
GET	/api/artists/{id}	-	brak	Pobiera pojedynczego artystę o określonym ID.
POST	/api/artists	{           "name": "string",           "image": "string",           "shortDescription": "string",           "career": "string"         }	organiser	Tworzy nowego artystę.
GET	/api/artists/{id}/events	-	brak	Pobiera wydarzenia związane z danym artystą.

Tab. 2.49. Charakterystyka punktów końcowych służących do zarządzania wydarzeniami.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
GET	/api/events	-	brak	Pobiera listę wydarzeń.
GET	/api/events/{id}	-	brak	Pobiera pojedyncze wydarzenie o określonym ID.
POST	/events/transaction	Dane wydarzenia.	organiser	Tworzy nowe wydarzenie.
POST	/api/event/likes-follows/{eventId}/{userId}/{actionType}	-	user	Zarządza polubieniem lub obserwowaniem wydarzenia przez użytkownika.
GET	/api/event/likes-follows/{eventId}/{actionType}	-	brak	Pobiera liczbę polubień lub obserwujących dla danego wydarzenia.

Tab. 2.50. Charakterystyka punktów końcowych służących do zarządzania biletami.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
GET	/api/tickets	-	brak	Pobiera listę wszystkich biletów.
GET	/api/events/tickets/{id}	-	brak	Pobiera pojedynczy bilet o określonym ID.
POST	/api/events/ticket/{id}	{           "type": "string",           "price": 0,           "dayOfWeek": "string",           "date": "string",           "color": "string",           "availableTickets": "string"         }	organiser	Tworzy nowy bilet.
GET	/api/events/{id}/tickets	-	brak	Pobiera bilety, które uczestniczą w danym wydarzeniu.

Tab. 2.51. Charakterystyka punktów końcowych służących do zarządzania transakcjami.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
GET	/api/transactions	-	organiser	Pobiera listę wszystkich transakcji dokonanych w systemie.
GET	/api/transactions/transaction/{id}	-	user	Pobiera pojedynczą transakcję o danym ID.
POST	/api/transactions/transaction	Dane transakcji.	user	Tworzy nową transakcję.
GET	/api/transactions/all/{userId}	-	user	Pobiera wszystkie transakcje dokonane przez danego użytkownika.

Tab. 2.52. Charakterystyka punktów końcowych służących do zarządzania użytkownikiem.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
POST	/api/user/auth	{           "login": "string",           "password": "string"         }	brak	Autoryzuje użytkownika i zwraca token dostępowy.
POST	/api/user/create	{           "email": "string",           "name": "string",           "password": hashString         }	brak	Tworzy nowego użytkownika.
POST	/api/user/update	Dane do zmiany.	user	Aktualizuje dane użytkownika.
GET	/api/user/preferences/{userId}	-	user	Pobiera preferencje onetimemonit użytkownika.
PUT	/api/user/{userId}/preferences/onetimemonit	-	user	Aktualizuje flagę jednorazowego monitu dla danego użytkownika.
DELETE	/api/user/logout/{userId}	-	user	Wylogowuje użytkownika na podstawie ID użytkownika.
POST	/api/user/{userId}/cart/add-ticket/{eventId}/{ticketId}	{           "quantity": 0         }	user	Dodaje bilet(y) do koszyka użytkownika.
POST	/api/user/{userId}/cart/remove-ticket/{eventId}/{ticketId}	{           "quantity": 0         }	user	Usuwa bilet(y) z koszyka użytkownika.
GET	/api/user/{userId}/cart	-	user	Pobiera zawartość koszyka użytkownika.
GET	/api/user/{userId}/preferences	-	user	Pobiera wszystkie preferencje użytkownika.
POST	/api/profile/like-follow/{userId}/{eventId}/{actionType}	-	user	Zarządza polubieniem lub obserwowaniem wydarzenia przez użytkownika.

GET	/api/profile/likes-follows/{userId}/{actionType}	-	user	Pobiera listę wydarzeń, które użytkownik polubił lub obserwuje.
GET	/api/profile/likes-follows/{userId}	-	user	Pobiera liczbę wydarzeń, które użytkownik polubił lub obserwuje.
POST	/api/profile/check-if-event-liked/{userId}/{eventId}/{actionType}	{           "isLiked": true         }	user	Sprawdza, czy użytkownik polubił lub obserwuje dane wydarzenie.

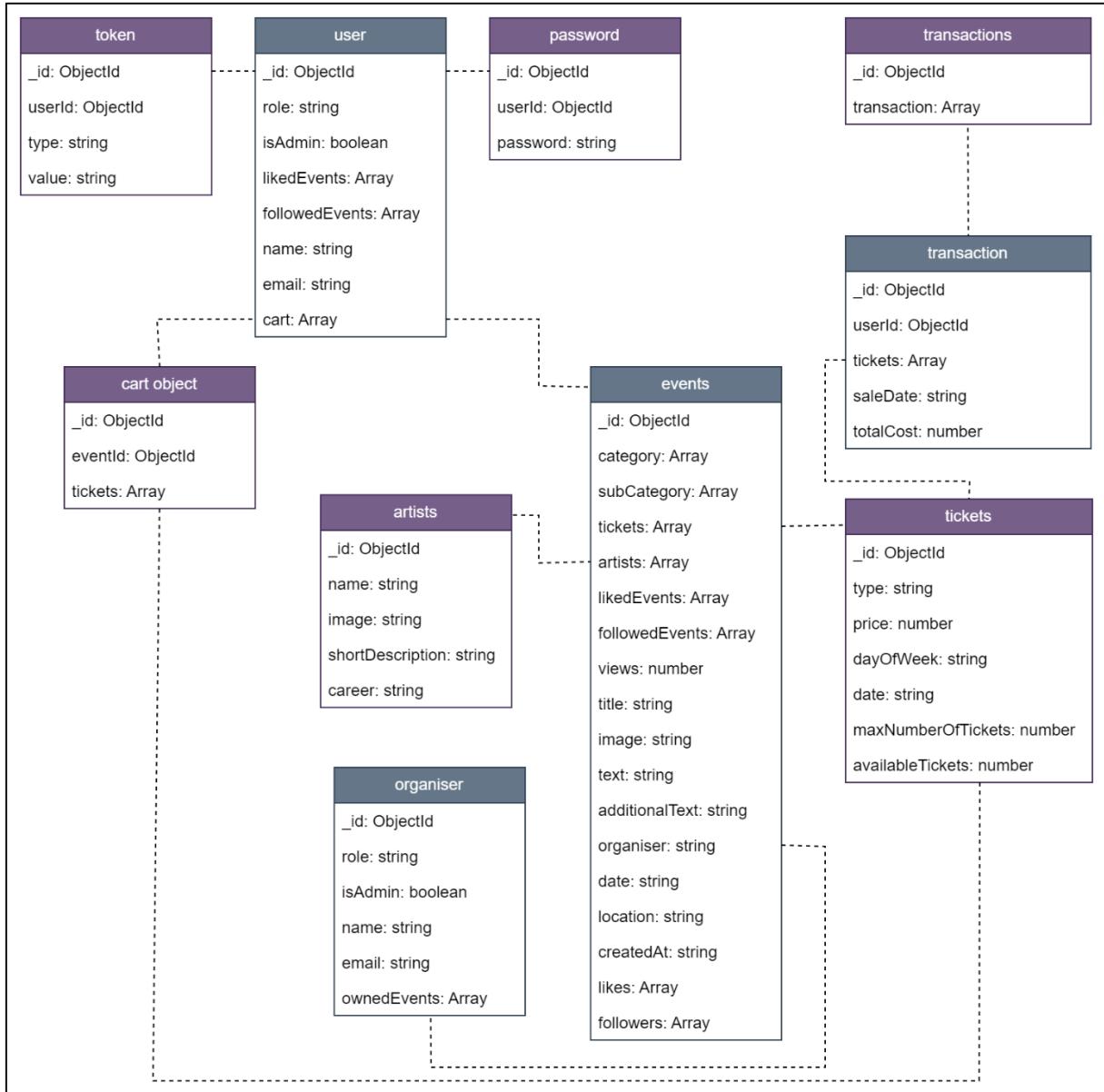
Tab. 2.53. Charakterystyka punktów końcowych służących do zarządzania organizatorem.

Metoda	Ścieżka	Przekazywane dane	Autoryzacja	Opis
GET	/api/organiser/{userId}	-	organiser	Pobiera listę wszystkich posiadanych wydarzeń.
POST	/api/organiser/{userId}/add-event/{eventId}	-	organiser	Dodaje wydarzenie do listy wydarzeń organizatora, których jest właścicielem.
GET	/api/organiser/stats/tickets-sold-by-event/{eventId}	-	organiser	Pobiera liczbę sprzedanych biletów dla danego wydarzenia.
GET	/api/organiser/stats/tickets-sold-by-organiser/{organiserName}	-	organiser	Pobiera liczbę sprzedanych biletów danego organizatora.
GET	/api/organiser/stats/total-earnings-by-organiser/{organiserName}	-	organiser	Pobiera całkowite zarobki danego organizatora.
GET	/api/organiser/stats/total-earnings-by-event/{eventId}	-	organiser	Pobiera całkowite zarobki danego wydarzenia.
GET	/api/organiser/stats/total-views-by-organiser/{organiserName}	-	organiser	Pobiera całkowite wyświetlenia wydarzeń danego organizatora.
GET	/api/organiser/sale-data/{organiserName}	-	organiser	Pobiera dane do wykresu sprzedaży biletów.

## 2.7. Schemat bazy danych

Baza danych systemu SSB została stworzona w oparciu o technologię MongoDB, co umożliwia przechowywanie danych w formie elastycznych dokumentów BSON. Baza ta służy do zarządzania danymi związanymi z wydarzeniami, artystami, biletami oraz użytkownikami serwisu. Baza została podzielona na kolekcje przechowujące dokumenty o określonej strukturze. Na rysunku 2.7. przedstawiono diagram ERD na którym

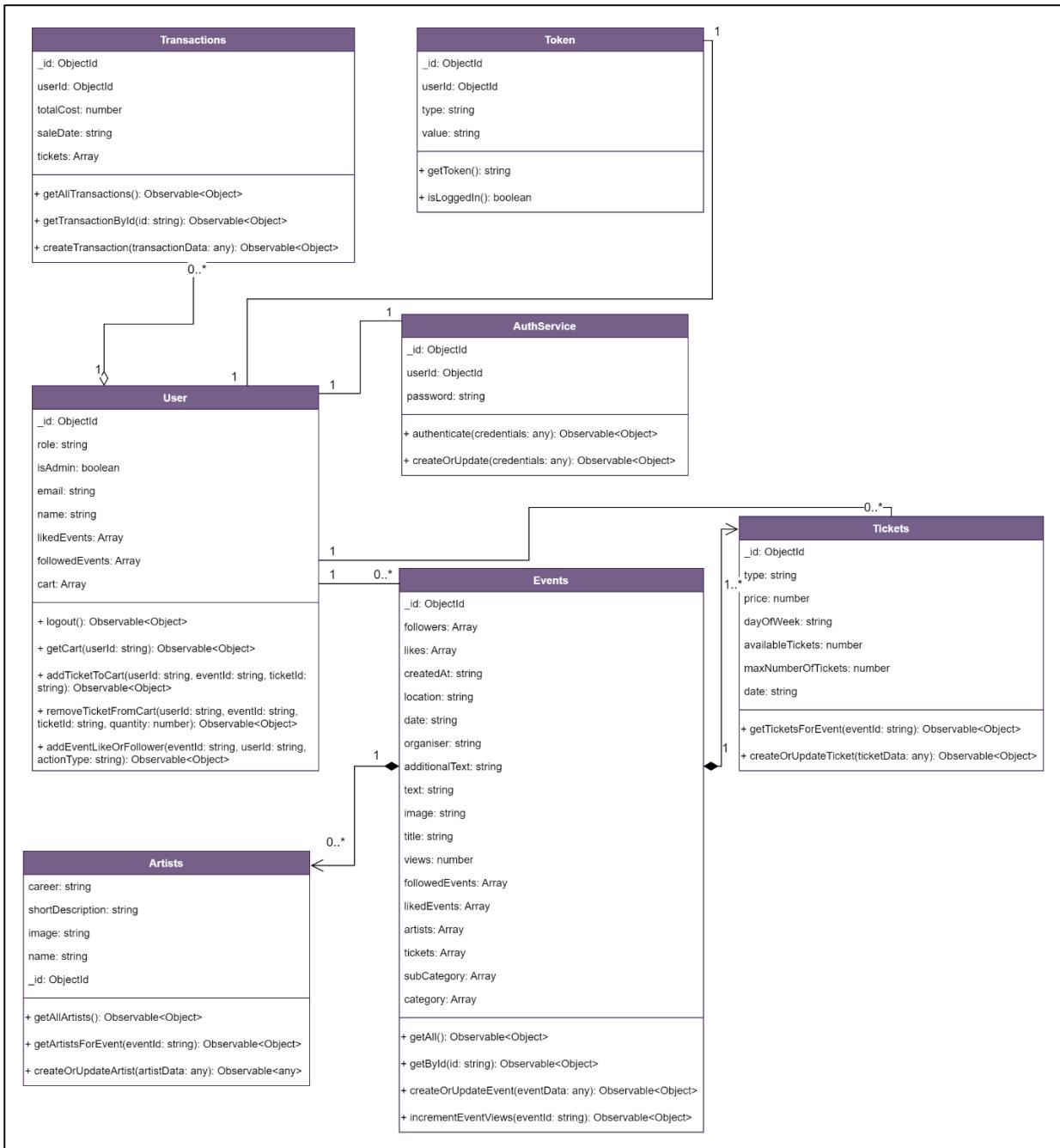
wyszczególniono najważniejsze kolekcje. Baza MongoDB jest bazą nierelacyjną, połączenia między kolekcjami symbolizują jedynie referencje lub zagnieżdżenia dokumentów, które umożliwiają powiązanie danych między sobą.



Rys. 2.7. Schemat ERD bazy danych systemu SSB. (źródło: Opracowanie własne)

## 2.8. Diagram klas

W tym podrozdziale przedstawiony, na rysunku 2.8., został koncepcyjny diagram klas systemu SSB.



Rys. 2.8. Schemat klas systemu SSB. (źródło: Opracowanie własne)

### 3. Implementacja systemu

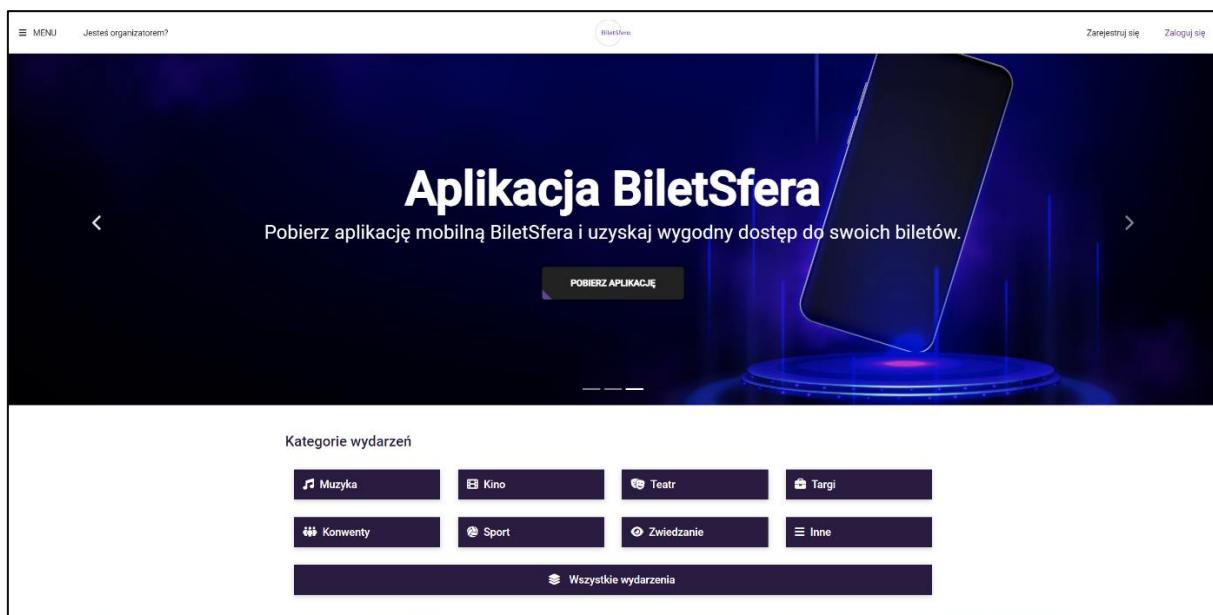
W rozdziale trzecim niniejszej pracy przedstawione zostały interfejsy graficzne aplikacji webowej oraz ekran aplikacji mobilnej. W ramach tego rozdziału, przedstawione zostały również najważniejsze fragmenty kodu.

#### 3.1. Aplikacja webowa

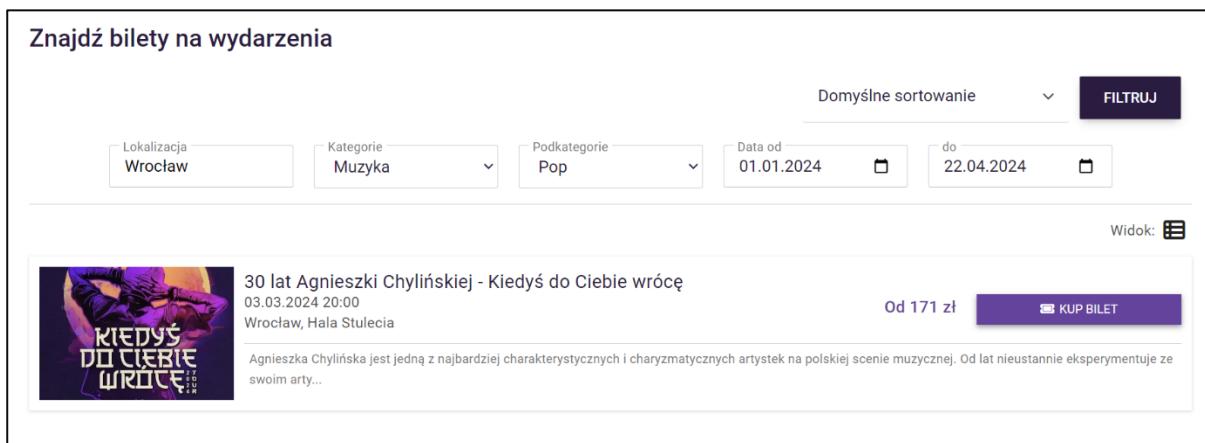
W tym podrozdziale przedstawione zostały interfejsy użytkownika oraz organizatora dostępne z poziomu przeglądarki. Dodatkowo przedstawiona i opisana została zaimplementowana architektura aplikacji serwerowej.

##### 3.1.1. Interfejs użytkownika

Użytkownikowi odwiedzającemu aplikację webową po raz pierwszy prezentowana jest strona główna (rys. 3.1.), która oferuje mu szereg różnych możliwości. Korzystając z przycisków dostępnych w menu nawigacyjnym użytkownik ma możliwość utworzenia konta lub zalogowania się do systemu. W menu nawigacyjnym dostępne jest również menu, które po rozwinięciu oferuje dostęp do innych punktów nawigacyjnych. Poniżej menu nawigacyjnego znajdują się kategorie wydarzeń, dzięki którym użytkownik w łatwiejszy i szybszy sposób może przejść do bardziej zaawansowanego wyszukiwania wydarzeń (rys. 3.2.). Zaawansowane wyszukiwanie oferuje różne opcje filtracji oraz sortowania ułatwiając wyszukiwanie konkretnych wyników.

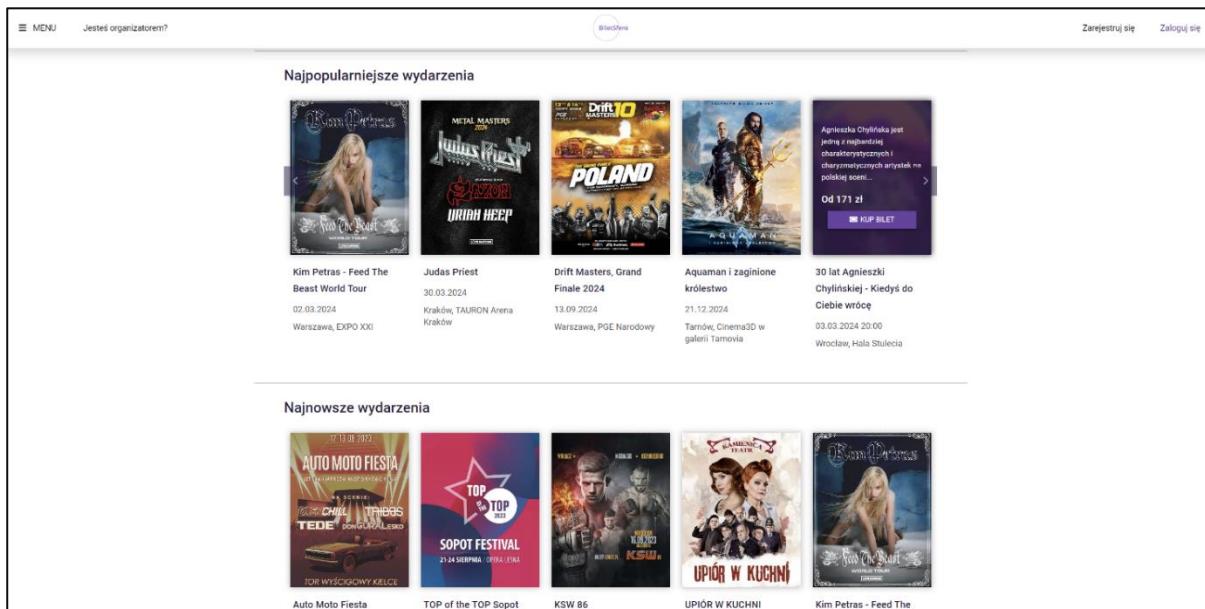


Rys. 3.1. Strona główna. (źródło: Opracowanie własne)



Rys. 3.2. Zaawansowane wyszukiwanie wydarzeń. (źródło: Opracowanie własne)

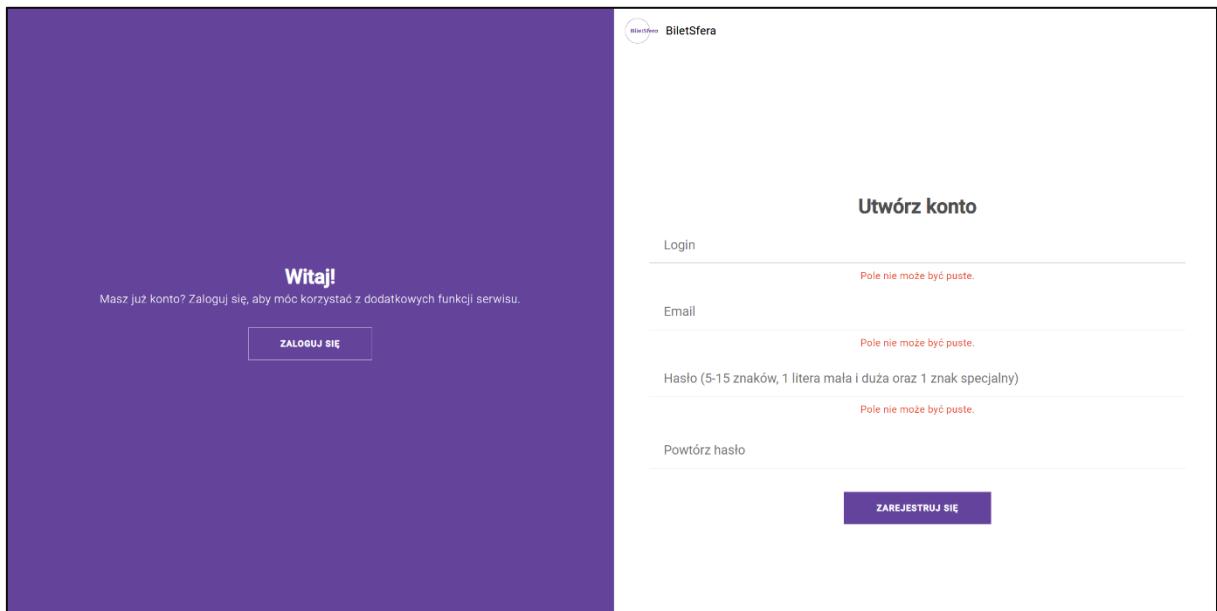
Na stronie głównej aplikacji użytkownikowi prezentowane są również wydarzenia (rys. 3.3.). Użytkownik niezalogowany ma m.in. dostęp do karuzeli najpopularniejszych wydarzeń w której prezentowane są wydarzenia posiadające największą liczbę wyświetleń. Poniżej w następnej sekcji znajdują się najnowsze wydarzenia.



Rys. 3.3. Strona główna – wydarzenia. (źródło: Opracowanie własne)

W celu odblokowania dodatkowych możliwości systemu zalecane jest utworzenie konta. Na rysunku 3.4., przedstawiony został ekran zawierający formularz umożliwiający utworzenie konta w systemie. Formularz został zabezpieczony zarówno po stronie aplikacji webowej jak i po stronie aplikacji serwerowej. Zabezpieczenia obejmują m.in. puste pola, unikalność nazw użytkowników czy politykę silnych haseł. Logowanie do systemu odbywa się analogicznie poprzez wypełnienie stosownego formularza, który również został

odpowiednio zabezpieczony. Aby przejść do logowania z poziomu tworzenia konta wystarczy nacisnąć przycisk "Zaloguj się". Po zakończeniu animacji odsłonięty zostaje formularz logowania.



**Utwórz konto**

Login Pole nie może być puste.

Email Pole nie może być puste.

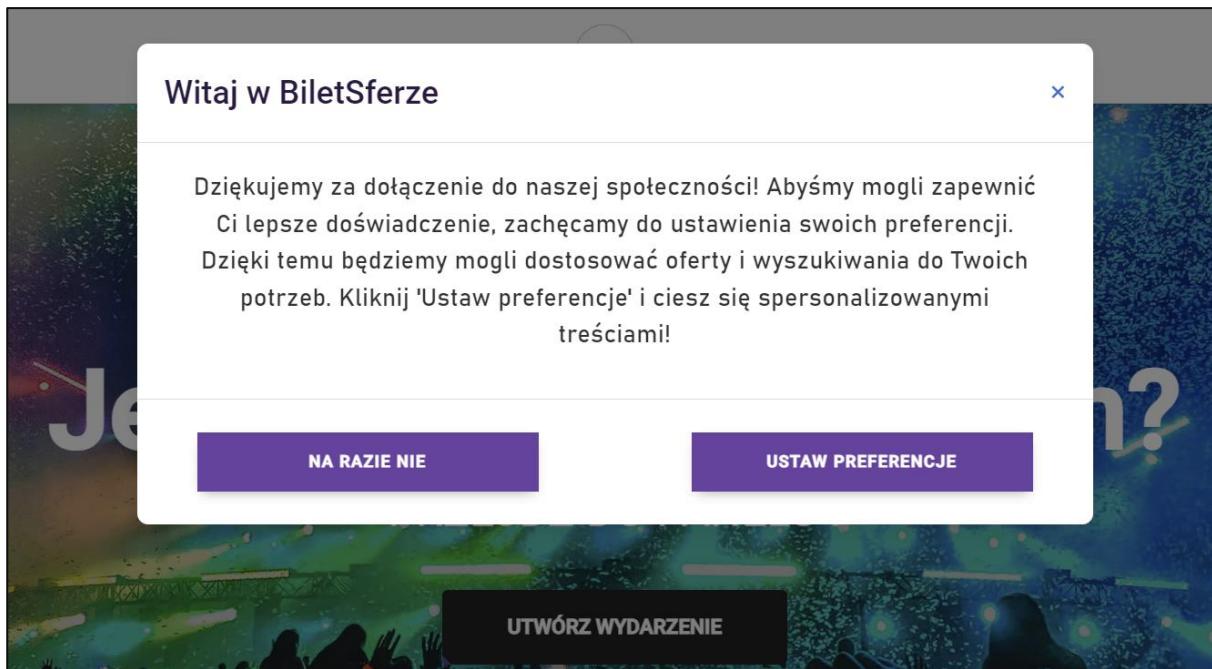
Hasło (5-15 znaków, 1 litera mała i duża oraz 1 znak specjalny) Pole nie może być puste.

Powtóż hasło

**ZAREJESTRUJ SIĘ**

Rys. 3.4. Ekran tworzenia konta. (źródło: Opracowanie własne)

Po poprawnym zalogowaniu się na istniejące konto, użytkownik zostaje przekierowany na stronę główną i od tej pory może w pełni korzystać z korzyści oferowanych przez system. W przypadku pierwszego logowania, system prezentuje użytkownikowi monit (rys. 3.5.) zachęcający do ustawienia preferencji konta w celu dostarczania użytkownikowi spersonalizowanego doświadczenia. Wiąże się to m.in. ze specjalną sekcją na stronie głównej w której wyświetlane są użytkownikowi wydarzenia pasujące do jego preferencji.

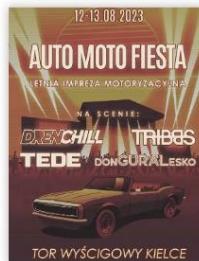


Rys. 3.5. Monit przy pierwszym logowaniu. (źródło: Opracowanie własne)

Ekran wyboru preferencji został przedstawiony na rysunku 3.6. Po ustawieniu preferencji użytkownikowi prezentowane są dodatkowe sekcje na stronie głównej zawierające wydarzenia, którymi może być zainteresowany (rys. 3.7.)

Rys. 3.6. Wybór preferencji (źródło: Opracowanie własne)

### Wybrane dla Ciebie



Auto Moto Fiesta

12.08.2024 - 13.08.2024

Kielce



TOP of the TOP Sopot Festival

21.08.2024 - 24.08.2024

Sopot



Kim Petras - Feed The Beast World Tour

02.03.2024

Warszawa, EXPO XXI

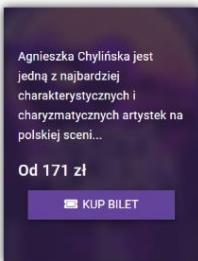


Judas Priest

30.03.2024

Kraków, TAURON Arena

Kraków



Agnieszka Chylińska jest jedną z najbardziej charakterystycznych i charizmatycznych artystek na polskiej scenie...

Od 171 zł

[KUP BILET](#)

30 lat Agnieszki Chylińskiej - Kiedyś do

Ciebie wróćę

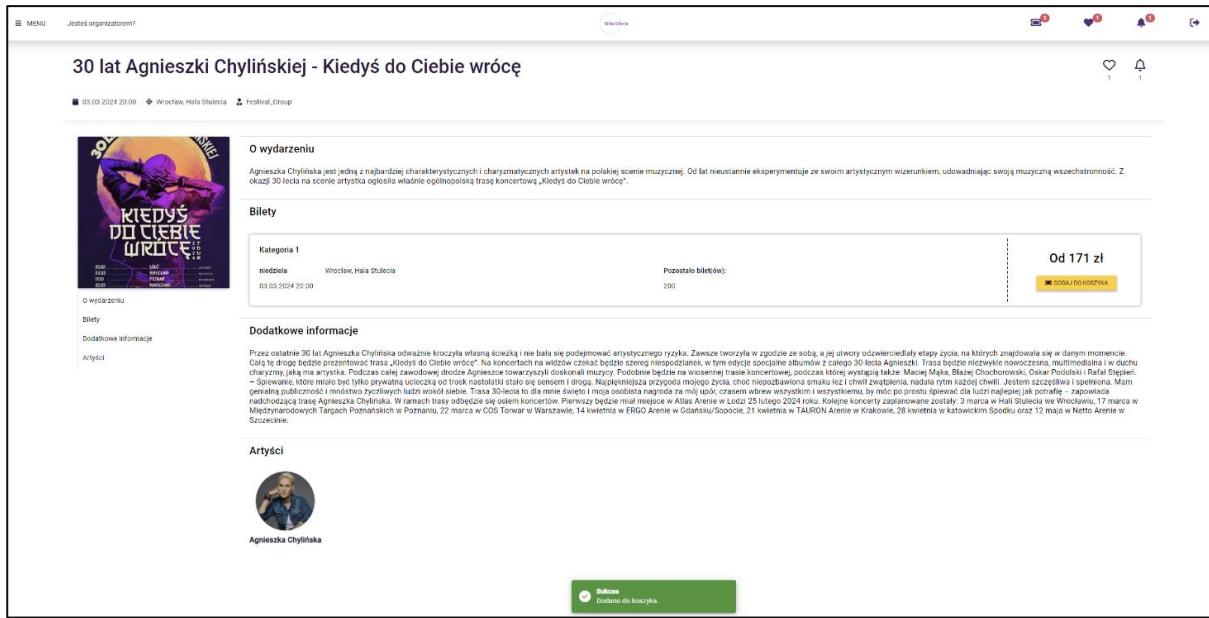
03.03.2024 20:00

Wrocław, Hala Stulecia

Rys. 3.7. Sekcja związana z preferencjami użytkownika (źródło: Opracowanie własne)

Po najechaniu kursem na dowolną z kart wydarzenia (rys. 3.7.), wyświetlany jest krótki opis. Po kliknięciu w kartę użytkownik zostaje przeniesiony do strony zawierającej szczegóły wydarzenia (rys. 3.8.).

Na ekranie przedstawionym na rysunku 3.8. użytkownik, oprócz dostępu do informacji ma również możliwość dodania interesujących go biletów. Aby dodać bilet do koszyka użytkownik musi być zalogowany w systemie, w przeciwnym razie wyświetlony zostanie monit błędu. Z poziomu ekranu szczegółu wydarzenia zalogowany użytkownik posiada również możliwość dodania wydarzenia do listy polubionych lub obserwowanych wydarzeń. Dodanie wydarzenia do ulubionych lub obserwowanych wydarzeń odbywa się poprzez kliknięcie ikon symbolizujących serce (polubienie wydarzenia) lub dzwonek (zaobserwowanie wydarzenia). Stan strony po dodaniu wydarzenia do ulubionych, obserwowanych oraz dodania biletu został przedstawiony na rysunku 3.8.



Rys. 3.8. Detal wydarzenia. (źródło: Opracowanie własne)

Użytkownik może przeglądać dodane do koszyka bilety przechodząc na stronę koszyka (rys. 3.9.). Użytkownik może się tam dostać np. poprzez rozwijane menu w lewym górnym rogu lub poprzez kliknięcie w ikonę symbolizującą bilet. Na stronie koszyka użytkownik może zarządzać dodanymi biletami. Może usuwać bilety, zwiększać lub zmniejszać ilość biletów, które chce zakupić, dodawać kody rabatowe lub przejść do płatności. Jeżeli bilet na wydarzenie wygaśnie lub upłynie jego termin bilet wciąż będzie znajdować się w koszyku użytkownika, jednak nie będzie on brany pod uwagę przy płatności. Bilety, które wygasły można jedynie usunąć (rys. 3.9.).

Koszyk					
NAZWA BILETU	CENA	ILOŚĆ	W SUMIE RAZEM	USUŃ Z KOSZYKA	
	171 zł	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	171 zł		
	135 zł	<input type="button" value="-"/> <input type="button" value="1"/> <input type="button" value="+"/>	135 zł		
<input type="button" value="ZASTOSUJ KOD PROMOCYJNY"/> <input type="text" value="Wprowadź kod promocyjny"/> <input type="button" value="ZASTOSUJ"/>					
<b>ŁĄCZNA KWOTA:</b> <b>171 zł</b>					
<input type="button" value="KUP BILETY"/>					

Rys. 3.9. Koszyk użytkownika. (źródło: Opracowanie własne)

Płatność następuje poprzez naciśnięcie przycisku "Kup Bilet" oraz poprawnym przeprowadzeniu płatności (rys. 3.10.).

Podsumowanie zamówienia

Zamówione bilety				Łączna kwota	
NAZWA BILETU	CENA	ILOŚĆ	W SUMIE RAZEM	Rabat	0 zł
30 lat Agnieszki Chylińskiej - Kiedyś do Ciebie wróćę	171 zł	1	171 zł	Razem	171 zł

Twoje dane

MK  
marcinkrol1@onet.pl

ZAMAWIAM I PŁACĘ

MasterCard PayU iBank Przelewy24

Kupione bilety będą dostępne w zakładce Transakcje.

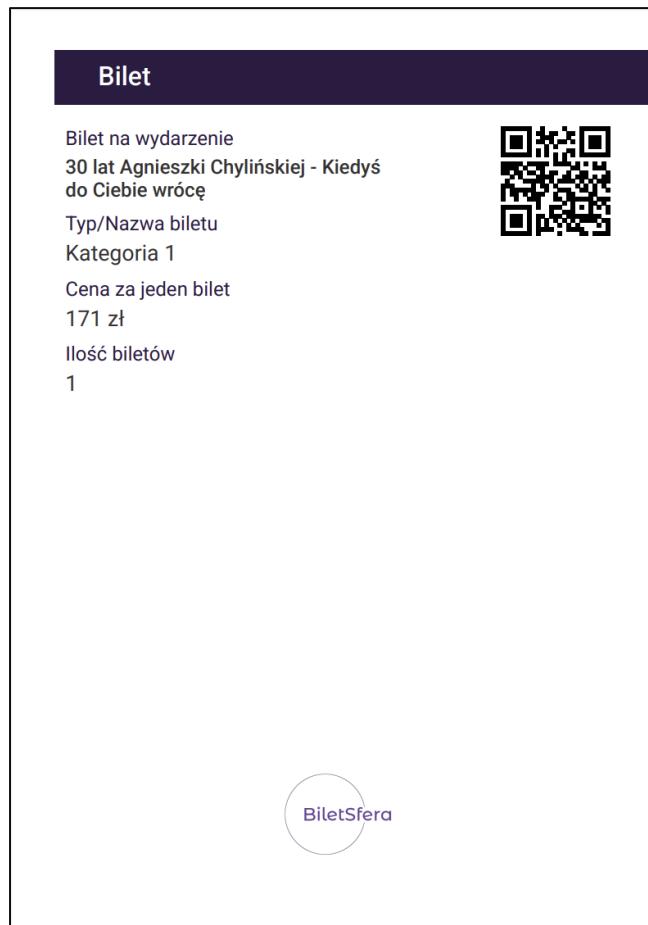
Rys. 3.10. Proces płatności. (źródło: Opracowanie własne)

Po dokonaniu płatności zakupione bilety dostępne są w zakładce "Transakcje" dostępnej z poziomu rozwijanego menu (rys. 3.11.). Użytkownik oprócz danych na temat przeprowadzonych transakcji ma możliwość wydrukowania biletu w formacie pdf (rys. 3.12.).

Historia zakupów

Data	Koszt zamówienia	Zakupione bilety	POBIERZ JAKO PDF
28.12.2023 18:39	49.8 zł	2 bilet(ów) Bilet Standard - 24.9 zł	POBIERZ JAKO PDF
29.12.2023 11:45	49.8 zł	2 bilet(ów) Bilet Standard - 24.9 zł	POBIERZ JAKO PDF
30.12.2023 16:52	171 zł	1 bilet(ów) Kategoria 1 - 171 zł	POBIERZ JAKO PDF

Rys. 3.11. Historia transakcji. (źródło: Opracowanie własne)



Rys. 3.12. Bilet w formacie pdf. (źródło: Opracowanie własne)

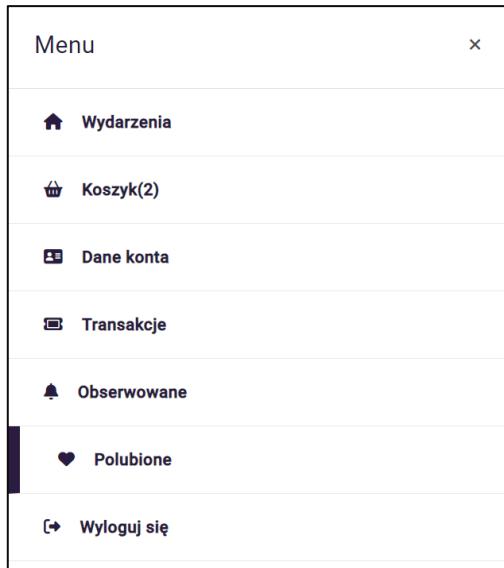
Użytkownik może zobaczyć polubione (rys. 3.13.) lub zaobserwowane (rys. 3.14.) wydarzenia przechodząc do odpowiednich stron używając rozwijanego menu nawigacyjnego (rys. 3.15.).

Polubione			
Nazwa	Data wydarzenia	Usuń z ulubionych	
30 lat Agnieszki Chylińskiej - Kiedyś do Ciebie wróć	03.03.2024 20:00		

Rys. 3.13. Tabela polubionych wydarzeń. (źródło: Opracowanie własne)

Obserwowane			
Nazwa	Ceny od	Dostępność	Kup bilety / Usuń
30 lat Agnieszki Chylińskiej - Kiedyś do Ciebie wróć	171 zł	Dostępny	

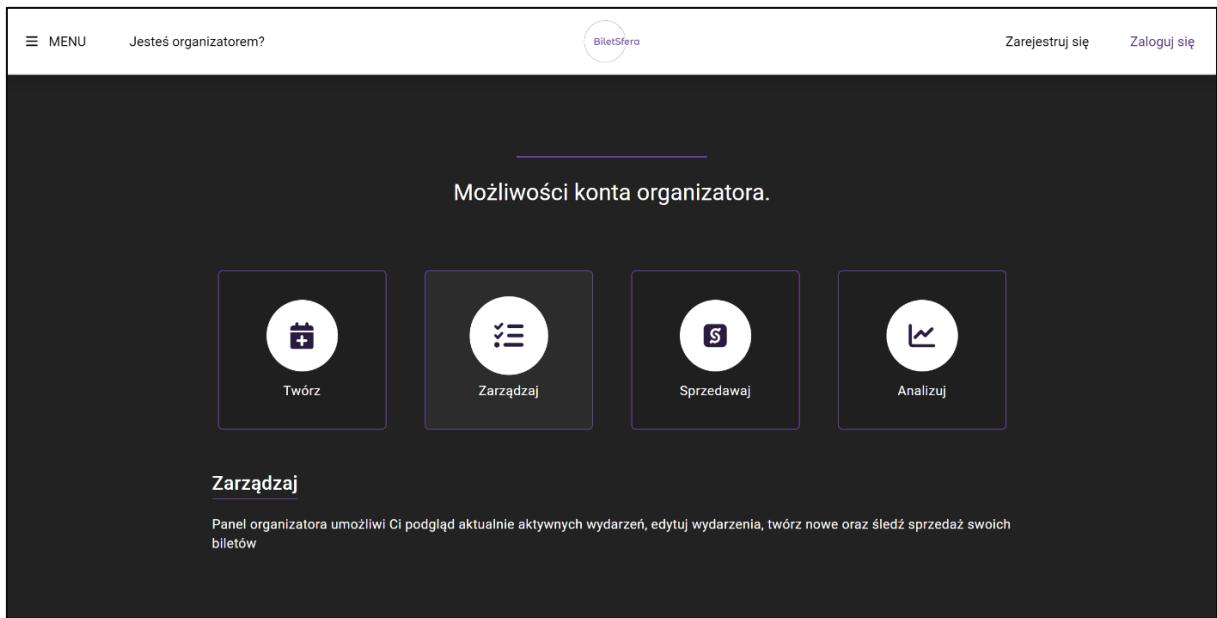
Rys. 3.14. Tabela obserwowanych wydarzeń. (źródło: Opracowanie własne)



Rys. 3.15. Rozwijane menu nawigacyjne. (źródło: Opracowanie własne)

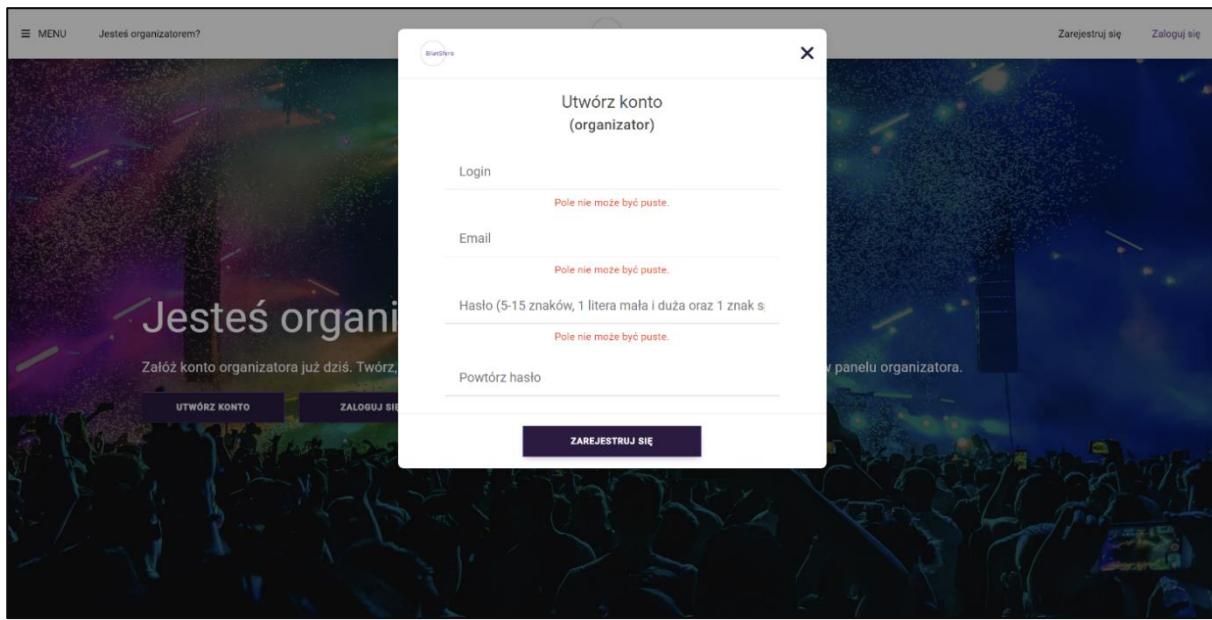
### 3.1.2. Interfejs organizatora

Organizatorzy posiadają dedykowaną stronę, na której mogą znaleźć informacje o możliwościach oferowanych przez panel organizatora dostępny po zalogowaniu na konto organizatora (rys. 3.16.). Aby dostać się na stronę należy kliknąć etykietę "Jesteś organizatorem?" dostępną z poziomu menu nawigacyjnego.



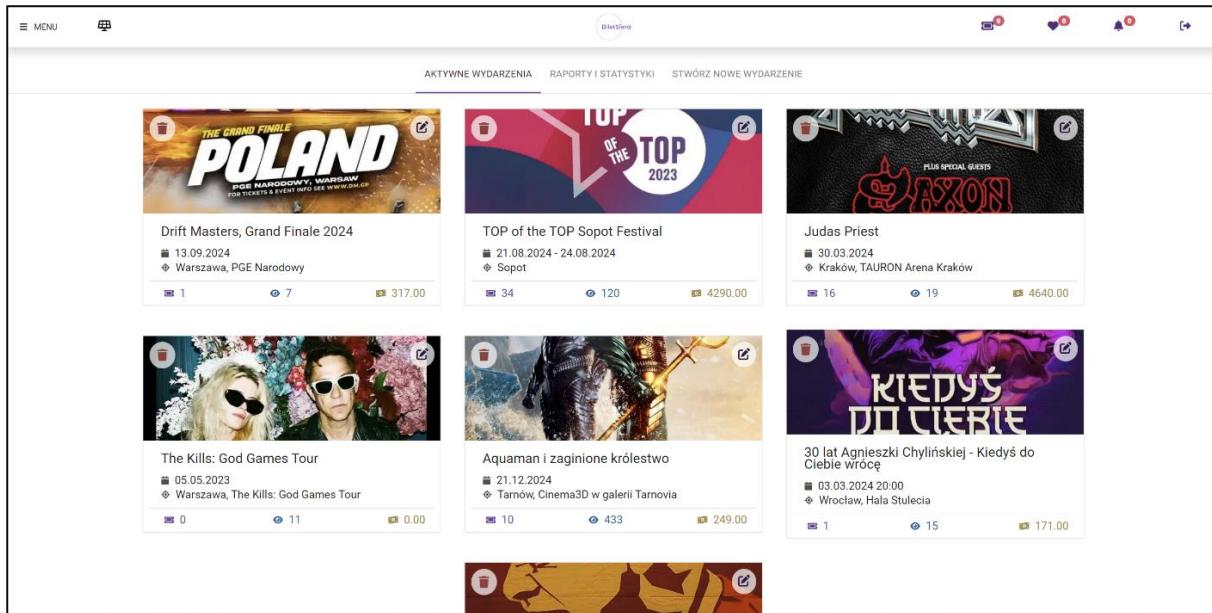
Rys. 3.16. Strona organizatora. (źródło: Opracowanie własne)

Strona umożliwia również szybkie utworzenie konta organizatora oraz zalogowanie się na konto organizatora (rys. 3.17.). Formularze podobnie jak w przypadku rejestracji i logowania użytkownika zostały zabezpieczone po stronie aplikacji webowej oraz aplikacji serwerowej.



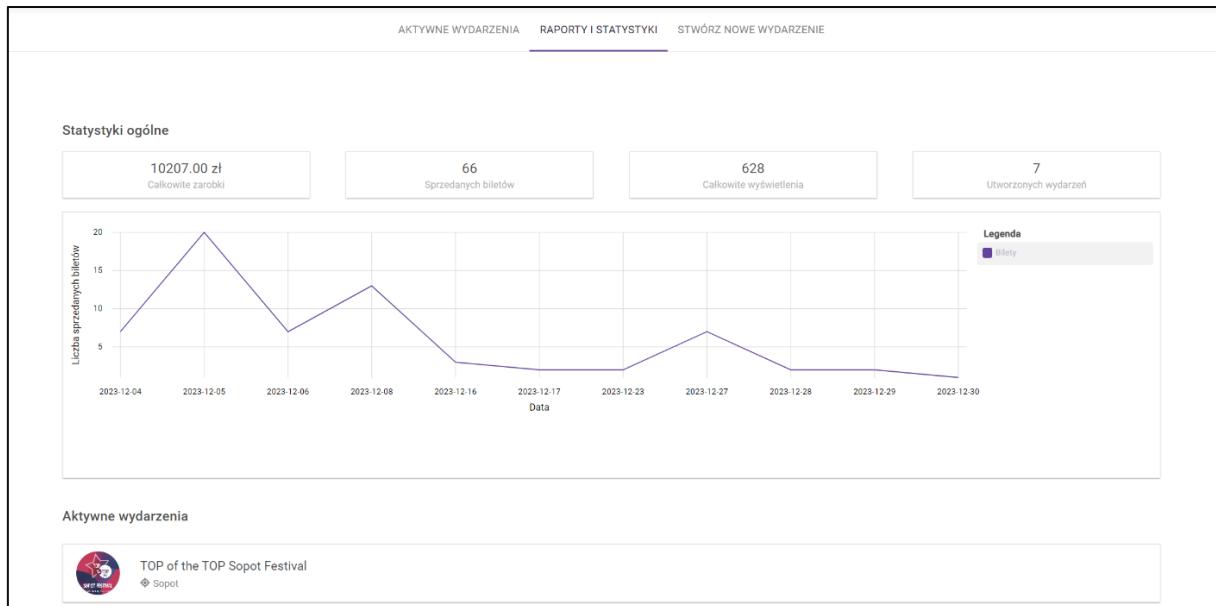
Rys. 3.17. Formularz rejestracji organizatora. (źródło: Opracowanie własne)

Po poprawnym zalogowaniu na konto, organizator zostaje przekierowany na stronę panelu aktywnych wydarzeń (rys. 3.18.). W zakładce aktywne wydarzenia znajdują się aktywne wydarzenia wraz z podstawowymi statystykami tj. liczba sprzedanych biletów, liczba wyświetleń czy zarobki wydarzenia. W tym miejscu można również usuwać lub edytować wybrane wydarzenia.



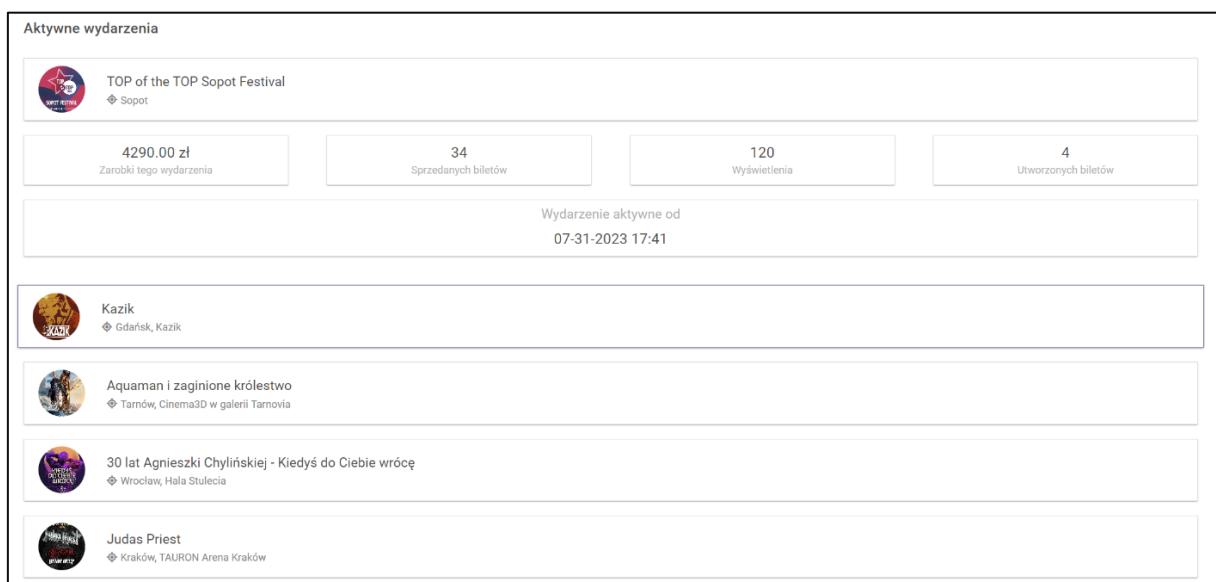
Rys. 3.18. Panel organizatora – aktywne wydarzenia. (źródło: Opracowanie własne)

W zakładce raporty i statystyki znajdują się statystyki ogólne oraz wykres przedstawiający sprzedaż biletów w czasie (rys. 3.19.). Statystyki, które są dostępne z tego poziomu to m.in.: całkowite zarobki, liczba sprzedanych biletów, całkowite wyświetlenia czy liczba utworzonych wydarzeń.



Rys. 3.19. Panel organizatora – raporty i statystyki. (źródło: Opracowanie własne)

Szczegółowe informacje i statystyki na temat pojedynczego wydarzenia prezentowane są po kliknięciu w wybrane wydarzenie w sekcji aktywne wydarzenia w zakładce raporty co zostało przedstawione na rysunku 3.20.

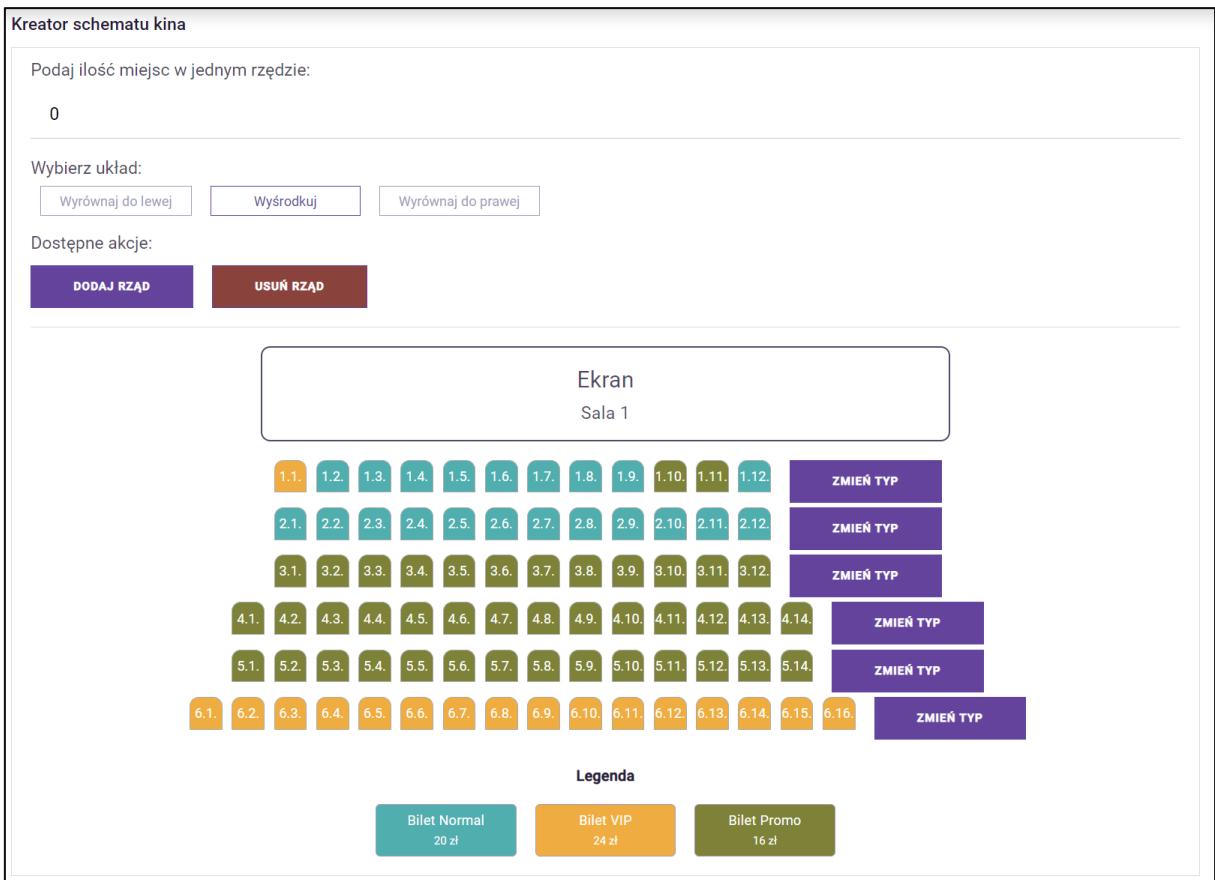


Rys. 3.20. Raporty i statystyki – statystyki pojedynczego wydarzenia. (źródło: Opracowanie własne)

Ostatnia zakładka umożliwia organizatorom tworzenie nowych wydarzeń. Organizator w celu utworzenia nowego wydarzenia musi uzupełnić formularz, który dla zachowania lepszej przejrzystości został podzielony na 6 etapów (rys. 3.21.).

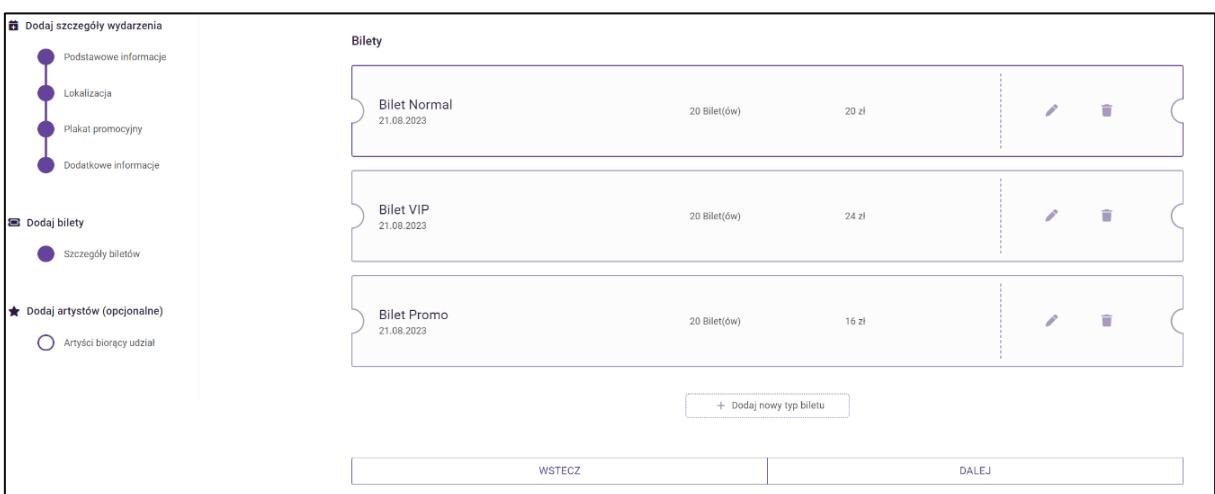
Rys. 3.21. Tworzenie nowego wydarzenia. (źródło: Opracowanie własne)

Po wybraniu kategorii "Kino" organizatorowi prezentowany jest kreator schematu kina (rys. 3.22.). Kreator umożliwia tworzenie podstawowych schematów kina, a po utworzeniu biletów w sekcji "Dodaj bilety" umożliwia zmianę typów siedzeń. Po kliknięciu w jeden z typów biletów dostępnych pod legendą organizator klikając w przycisk "Zmień typ" może zmienić typ wszystkim siedzeniom w danym rzędzie. Aby zmienić typ pojedynczego siedzenia należy kliknąć wybrane siedzenie.



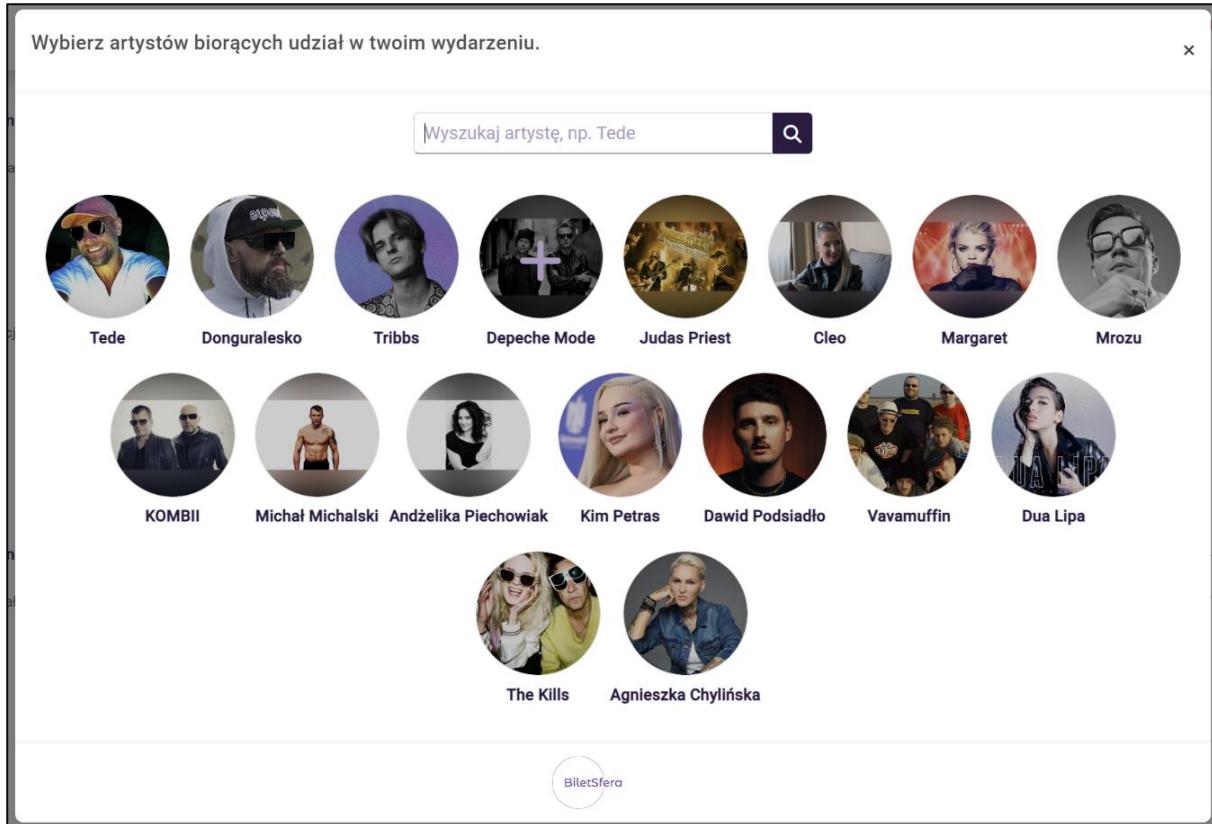
Rys. 3.22. Kreator schematu kina. (źródło: Opracowanie własne)

Istotnym etapem tworzenia wydarzenia jest utworzenie biletów. Po kliknięciu przycisku "Dodaj nowy typ biletu" prezentowany jest formularz tworzenia biletu, gdzie organizator uzupełnia m.in. kolor biletu, cenę czy nazwę biletu. Po wypełnieniu wymaganych danych bilet zostaje utworzony (rys. 3.23.).



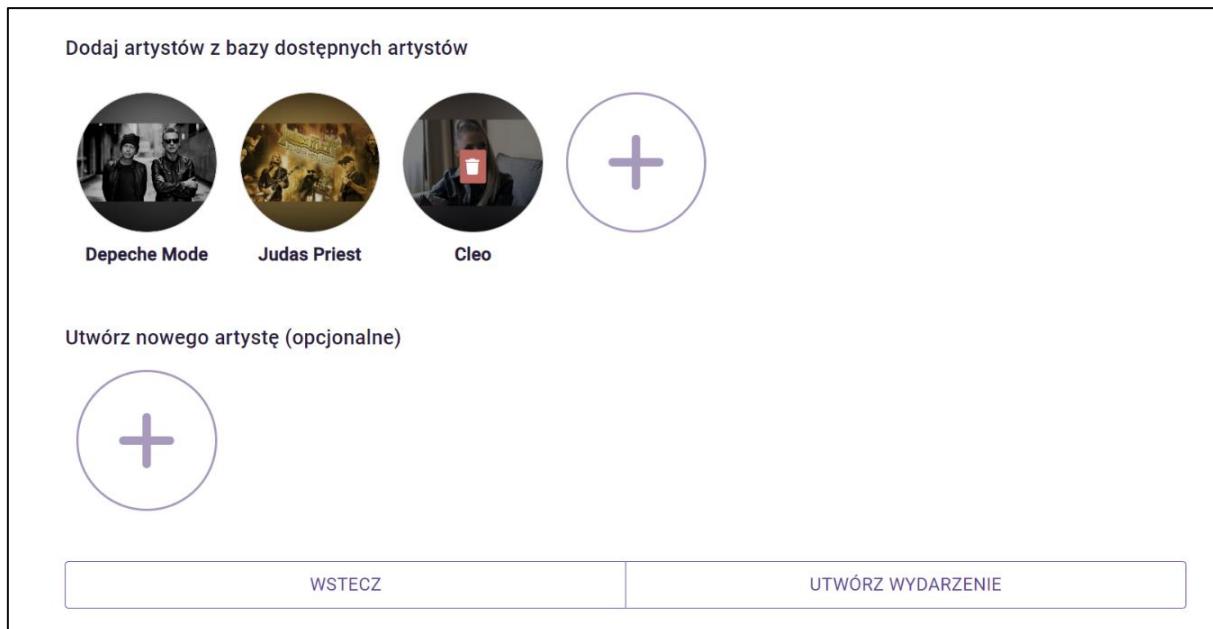
Rys. 3.23. Sekcja z utworzonymi biletami. (źródło: Opracowanie własne)

Etap dodania artystów biorących udział w wydarzeniu jest opcjonalny, jednak został on odpowiednio rozbudowany, aby umożliwić intuicyjny i szybki sposób dodawania artystów. Po kliknięciu w okrąg z plusem system prezentuje okienko wyboru artystów (rys. 3.24.). Pole wyszukiwania umożliwia szybkie wyszukiwanie artystów, po kliknięciu na portret artysta zostaje dodany.



Rys. 3.24. Okno wyboru artystów. (źródło: Opracowanie własne)

Po najechaniu kursem na portret artysty prezentowany jest przycisk umożliwiający usunięcie go z sekcji artystów biorących udział w wydarzeniu (rys. 3.25.). System oferuje również możliwość utworzenia nowego artysty, jeżeli ten nie znajduje się jeszcze w bazie danych. Po poprawnym wypełnieniu formularza artysta zostaje dodany do bazy danych. Poprawne wypełnienie wszystkich etapów wydarzenia oraz kliknięcie w przycisk "Utwórz wydarzenie" inicjuje proces tworzenia obiektów po stronie aplikacji serwerowej. Dodatkowo formularz tworzenia wydarzeń został zabezpieczony zarówno po stronie aplikacji webowej jak i aplikacji serwerowej, co szczegółowo zostało przedstawione w podrozdziale 3.4.



Rys. 3.25. Artyści biorący udział w wydarzeniu. (źródło: Opracowanie własne)

### 3.2. Aplikacja mobilna

W tym podrozdziale przedstawiony i opisany został interfejs użytkownika dostępny w aplikacji mobilnej.

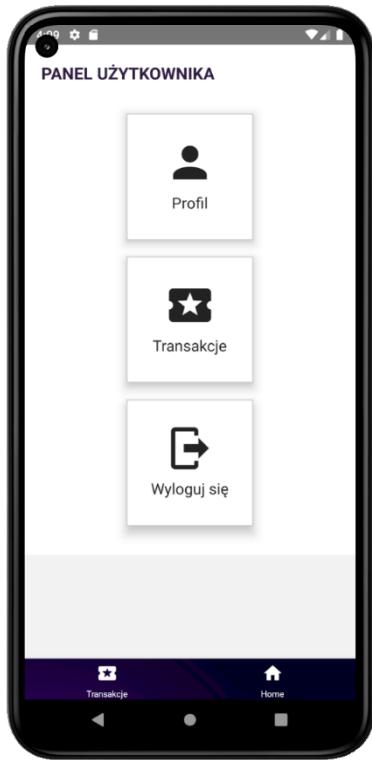
Po pierwszym uruchomieniu użytkownikowi prezentowany jest ekran z formularzem logowania (rys. 3.26.). Pola zostały zabezpieczone, po prowadzeniu błędnych danych lub niewprowadzeniu jednego z wymaganych pól system wyświetla odpowiedni komunikat błędu (rys. 3.26.).



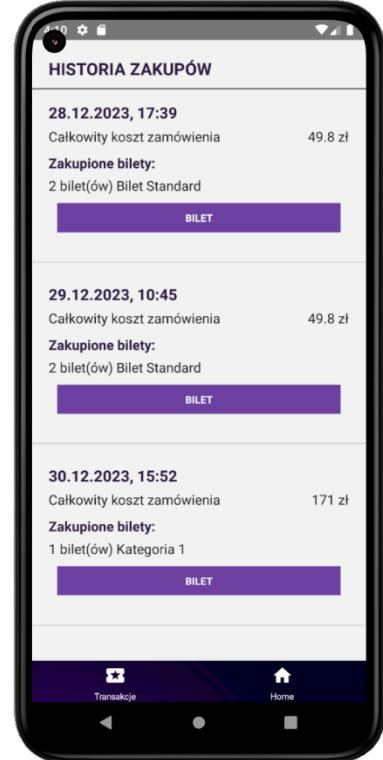
Rys. 3.26. Ekran logowania z monitem błędu. (źródło: Opracowanie własne)

Jeżeli użytkownik poprawnie zaloguje się na istniejące konto, system przekierowuje użytkownika na ekran główny przedstawiony na rysunku 3.27. W panelu użytkownika znajdują się trzy kafelki. Kafelka "Profil" po naciśnięciu przekierowuje użytkownika do ekranu w którym znajdują się podstawowe informacje o koncie tj. email czy nazwa. Kafelka "Transakcje" przekierowuje użytkownika do ekranu w którym znajdują się wszystkie dokonane przez użytkownika transakcje (rys. 3.28.). Kafelka "Wyloguj się" służy do wylogowania użytkownika z systemu. Użytkownik ma też dostęp do nawigacji dolnej, która pozwala na szybkie przechodzenie do najważniejszych ekranów.

Ekran przedstawiony na rysunku 3.28. zawiera informacje o dokonanych transakcjach oraz przycisk "Bilet", który po kliknięciu przekierowuje użytkownika do szczegółów biletu.



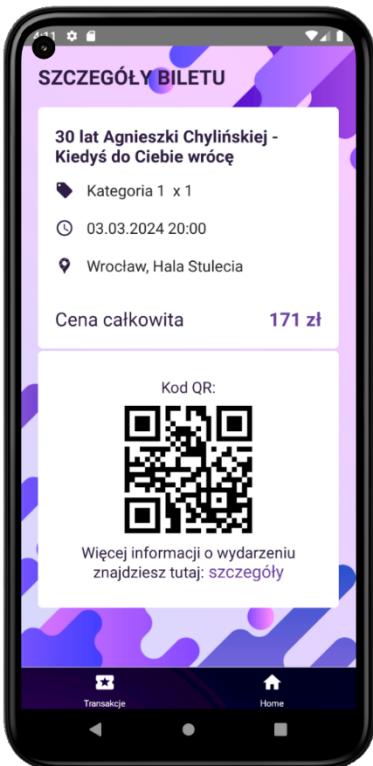
Rys. 3.27. Panel użytkownika. (źródło:  
Opracowanie własne)



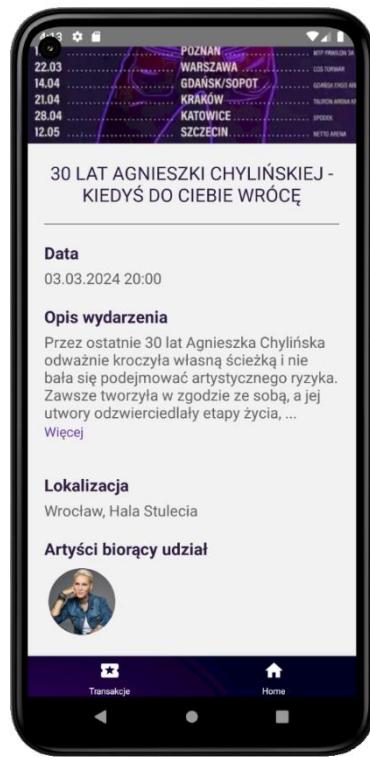
Rys. 3.28. Historia zakupów. (źródło:  
Opracowanie własne)

Ekran przedstawiony na rysunku 3.29. zawiera szczegóły dotyczące zakupionego biletu. Poniżej podstawowych informacji znajduje się kod QR, który umożliwia szybką weryfikację ważności biletu np. poprzez pracownika kina czy wydarzenia współpracującego z systemem SSB. Użytkownik ma też możliwość przejścia do szczegółów wydarzenia klikając w tekst "szczegóły" znajdujący się pod kodem QR.

Ekran szczegółów wydarzenia zawiera grafikę promocyjną wydarzenia. Informacje o wydarzeniu znajdują się poniżej grafiki promocyjnej (rys. 3.30.).



Rys. 3.29. Szczegóły biletu. (źródło: Opracowanie własne)



Rys. 3.30. Szczegóły wydarzenia. (źródło: Opracowanie własne)

### 3.3. Aplikacja serwerowa

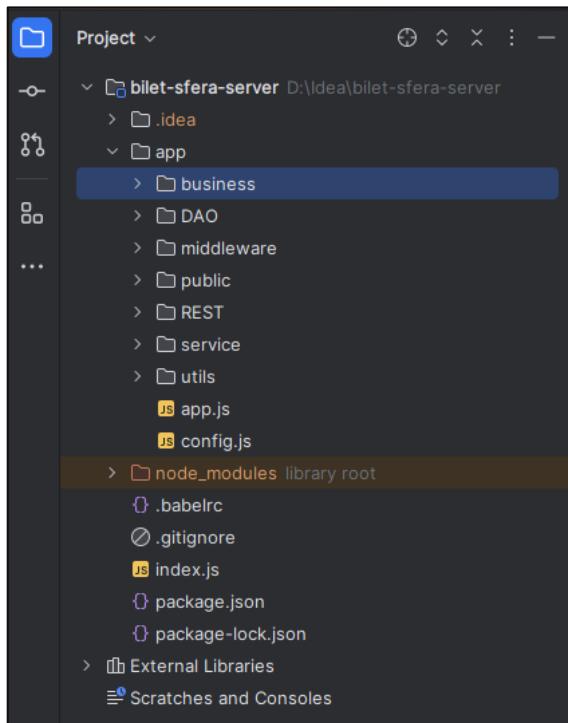
W tym podrozdziale przedstawiona i opisana została architektura aplikacji serwerowej.

Aplikacja serwerowa oparta jest na technologiach Node.js i Express.js. Aplikacja serwerowa skupia się na realizacji logiki biznesowej, komunikacji z bazą danych MongoDB oraz obsłudze zapytań HTTP.

Na rysunku 3.31. została przedstawiona struktura aplikacji serwerowej. Poniżej opisano poszczególne komponenty komponenty

- /business: W tym module zawarta jest główna logika aplikacji, odpowiedzialna za przetwarzanie danych oraz implementację funkcjonalności biznesowych.
- /DAO: Ten moduł obsługuje interakcje z bazą danych MongoDB. Zapewnia dostęp do danych, umożliwiając odczyt, zapis oraz modyfikację informacji w bazie.
- /middleware: Moduł zawiera komponenty pośredniczące, które przetwarzają żądania HTTP przed ich przekazaniem do obsługi w warstwie REST.
- /REST: W tej części znajdują się punkty końcowe udostępniane na zewnątrz w celu zapewnienia komunikacji pomiędzy aplikacjami klienckimi, a bazą danych.

- /service: Ten moduł zawiera specjalizowane usługi, takie jak momentWrapper oraz mongoConverter, które wspierają różne aspekty aplikacji, takie jak manipulacja datami czy konwersja danych do formatu zrozumiałego dla bazy danych.
- /utils: Tutaj umieszczone są różnorodne narzędzia pomocnicze, takie jak konfiguracja Swaggera, style dla interfejsu Swagger UI oraz schematy dla Swagger.io.



Rys. 3.31. Architektura aplikacji serwerowej. (źródło: Opracowanie własne)

### 3.4. Najciekawsze fragmenty kodu

W podrozdziale 3.4. zaprezentowane zostały najciekawsze i najistotniejsze fragmenty kodu związanego z implementacją aplikacji serwerowej, webowej oraz mobilnej.

Aplikacja serwerowa stanowi największa i najważniejszą część całego projektu, dlatego to jej poświęcona zostanie większość tego podrozdziału.

Na listingu 3.1. przedstawiony został punkt końcowy służący do utworzenia nowego konta w systemie. Funkcja obsługująca żądanie sprawdza czy przesłane z aplikacji klienckiej hasło spełnia warunki silnego hasła. Następnie wykonywana jest metoda *createNewOrUpdate()*, która tworzy nowe konto. W przypadku podania słabego hasła lub wystąpienia błędu metoda zwraca stosowny komunikat aplikacji klienckiej.

```

1 // Create user
2   router.post('/api/user/create', async (request, response, next) => {
3     try {
4       // Validate the password using the regex
5       const strongRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-
6       z\d@$!%*?&]{5,15}$/;
7       if (!strongRegex.test(request.body.password)) {
8         response.status(400).json({ error: 'Password does not meet the strength
9         criteria.' });
10        return;
11      }
12
13      // Proceed with user creation if the password is strong
14      const result = await
15      business.getUserManager(request).createNewOrUpdate(request.body);
16      response.status(200).json(result);
17    } catch (error) {
18      applicationException.errorHandler(error, response);
19    }
20  });

```

*Listing. 3.1. Tworzenie nowego konta.*

Poniżej, na listingu 3.2. przedstawione zostały dwie metody używane do uwierzytelnienia użytkownika. Funkcja *authorize* ma na celu weryfikację czy hasło podane dla danego identyfikatora użytkownika znajduje się w kolekcji zawierającej hasła użytkowników. Jeśli rekord zostanie znaleziony, funkcja zwraca zahasowane hasło. W razie niepowodzenia zgłasza błąd. Funkcja *authenticate* na samym początku sprawdza czy użytkownik o danej nazwie istnieje w bazie danych. Jeżeli tak wywołuje metodę *authorize* opisaną wyżej, a następnie porównuje hasła wykorzystując w tym celu dedykowaną funkcję *bcrypt.compare* dostępną w bibliotece *bcrypt*. Jeśli hasło jest niepoprawne lub nie istnieje, funkcja zgłasza błąd uwierzytelnienia. W przypadku poprawnego uwierzytelnienia tworzy i zwraca token JWT dla danego użytkownika za pomocą metody *TokenDAO.create*.

Do przekształceń haseł w bazie danych na postać nie jawną wykorzystany został algorytm kryptograficzny *bcrypt*. Algorytm ten cechuje się dużym bezpieczeństwem i jest wysoce odporny na ataki typu brute-force. Zastosowanie specjalnej funkcji mieszającej oraz tzw. soli pozwala na generowanie unikalnych hashy. Hash nawet dwóch tych samych haseł nigdy nie będzie taki sam, co gwarantuje dodatkowe bezpieczeństwo. Algorytm jest używany już od wielu lat i do tej pory nie znaleziono w nim żadnych luk lub podatności.

```

1  async function authorize(userId, password) {
2    const result = await PasswordModel.findOne({ userId });
3    if (result && mongoConverter(result)) {
4      return result ? result.password : null;
5    }
6    throw applicationException.new(applicationException.UNAUTHORIZED, 'User
7 and password does not match');
8
9  async function authenticate(name, password) {
10    const user = await UserDAO.getByEmailOrName(name);
11    if (!user) {
12      throw applicationException.new(applicationException.UNAUTHORIZED,
13      'User with that email does not exist');
14    }
15    const hashedPassword = await PasswordDAO.authorize(user.id);
16    if (!hashedPassword || !(await bcrypt.compare(password, hashedPassword))) {
17      throw applicationException.new(applicationException.UNAUTHORIZED,
18      'Incorrect password');
19    }
20    const token = await TokenDAO.create(user);
21    return getToken(token);
22  }

```

*Listing. 3.2. Uwierzytelnianie użytkownika.*

Na listingu 3.3. przedstawiono proces dodawania biletu do koszyka użytkownika. Metoda najpierw znajduje użytkownika o podanym *id*, jeżeli użytkownik nie istnieje, zostaje zgłoszony wyjątek. W linijce 7. znajdowany jest przedmiot w koszyku. Jeżeli przedmiot nie istnieje wykonywana jest metoda *push*, która dodaje nowy przedmiot z podanym biletom i ilością do koszyka użytkownika. Jeśli bilet istnieje, to w bloku *else* (13. linijka) zaktualizowana zostaje jego ilość. Na koniec funkcja zapisuje zaktualizowanego użytkownika ze zmodyfikowanym koszykiem i zwraca zaktualizowane dane użytkownika.

```

1  async function addToCart(userId, eventId, ticketId, quantity) {
2    try {
3      const user = await UserModel.findOne({ _id: userId });
4      if (!user) {
5        throw applicationException.new(applicationException.NOT_FOUND, 'User
not found');
6      }
7
8      const cartItem = user.cart.find(item => item.event.toString() === eventId);
9      if (!cartItem) {
10        user.cart.push({
11          event: eventId,
12          tickets: [{ ticket: ticketId, quantity }]
13        });
14      } else {
15        const ticket = cartItem.tickets.find(t => t.ticket.toString() === ticketId);
16        if (ticket) {
17          ticket.quantity += quantity || 1;
18        } else {
19          cartItem.tickets.push({ ticket: ticketId, quantity });
20        }
21      }
22      // Save the updated user with the modified cart
23      const updatedUser = await user.save();
24      return mongoConverter(updatedUser);
25    } catch (error) {
26      throw error;
27    }
28  }

```

*Listing. 3.3. Dodawanie biletu do koszyka.*

Aplikacja serwerowa udostępnia wiele punktów końcowych typu *get*, które służą do uzyskania przez aplikacje klienckie potrzebnych danych. W przypadku pobierania wydarzeń zaistniała potrzeba dodatkowego przetwarzania daty. Przykład jednego z ciekawszych zapytań *get* przedstawiono na listingu 3.4. Zapytanie, które wykonuje klient inicjalizuje metodę, która następnie pozyskuje parametr *userId*. W linijce 5 następuje próba znalezienia użytkownika o podanym *id* oraz pobranie pola *preferences*. Jeżeli użytkownik i pole zostanie znalezione tworzona jest tablica zawierająca wszystkie preferowane przez użytkownika kategorie i podkategorie wydarzeń. Dalsza część kodu wykonuje agregację znajdując tylko wydarzenia należące do wybranych kategorii i podkategorii. Kod zaczynający się od linijki 23. Służy do przefiltrowania otrzymanych wydarzeń i zwróceniu tylko tych, których termin jeszcze nie upłynął.

```

1 // Get events based on user preferences
2 router.get('/api/events/preferences/:userId', auth, async (req, res) => {
3     const { userId } = req.params;
4     try {
5         const user = await UserDAO.model.findOne({ _id: userId
6         }).select('preferences');
7         if (!user) {
8             return res.status(404).json({ message: 'User not found' });
9         }
10        const userPreferences = [
11            ...user.preferences.selectedCategories,
12            ...user.preferences.selectedSubCategories
13        ];
14        const allEvents = await EventDAO.model.aggregate([
15            {
16                $match: {
17                    $or: [
18                        { category: { $in: userPreferences } },
19                        { subCategory: { $in: userPreferences } }
20                    ]
21                }
22            }
23        ]);
24
25        const currentDate = new Date();
26        const activeEvents = allEvents.filter(event => {
27            const parsedDate = parseDate(event.date);
28            return parsedDate >= currentDate;
29        });
30
31        res.status(200).json({ matchedEvents: activeEvents });
32    } catch (error) {
33        console.error('Error:', error);
34        res.status(500).json({ message: 'Internal server error' });
35    }
36});

```

*Listing. 3.4. Pobranie wydarzeń na podstawie preferencji.*

Najistotniejszą aspektem systemu w kontekście użytkownika jest umożliwienie przeprowadzenia transakcji i zakupu biletów na wydarzenia. Listing 3.5. przedstawia fragment odpowiedzialny właśnie za ten aspekt. W celu zapewnienia integralności danych oraz wykluczenia sytuacji, w której jakiś etap transakcji nie zostaje wykonany poprawnie, a mimo to zostają zmienione stany obiektów lub utworzone nowe obiekty, wykorzystano mechanizm sesji oraz transakcji MongoDB. Transakcje umożliwiają wykonanie wielu operacji w izolacji i cofnięcie zmian, jeśli jedna z nich zakończy się niepowodzeniem. Natomiast mechanizm sesji służy do śledzenia stanu komunikacji między bazą danych,

a serwerem. Mechanizm ten gwarantuje, że wszystkie operacje w ramach jednej sesji są odpowiednio koordynowane i zachowują spójność danych między sobą.

Gdy aplikacja kliencka wysyła żądanie na punkt końcowy przedstawiony na listingu 3.5. funkcja auth sprawdza uwierzytelnienie użytkownika, następnie pobiera informacje o identyfikatorze użytkownika i zakupionych biletach z ciała żądania HTTP. W ramach transakcji MongoDB, dla każdego biletu przeprowadzany jest szereg działań. W przypadku wydarzenia z kategorii Kino, wykonywana jest operacja rezerwująca miejsca w Sali, widoczna później na schemacie w aplikacji klienckiej. Po przeprowadzeniu operacji na biletach, wykonywana jest metoda `createNewOrUpdate()` tworząca nową obiekt transakcji. Gdy wszystkie operacje zakończą się pomyślnie, transakcja jest zatwierdzana (`commit`) i sesja jest zamknięta.

Warto dodać, iż w kontekście ochrony danych oraz procesu uwierzytelniania, wszystkie kluczowe punkty końcowe zostały zabezpieczone przez metodę uwierzytelniającą (`auth`). Ta procedura wymusza na każdym żądaniu uwierzytelnienie się poprawnym tokenem JWT. Innymi słowy, żądania, które nie zawierają odpowiedniego uwierzytelnienia, nie doprowadzą do wykonania operacji na serwerze. Ta metoda zabezpieczenia jest kluczowa dla zapewnienia integralności danych oraz ochrony systemu przed nieuprawnionym dostępem czy manipulacją.

```

1 // Create a single transaction
2 router.post('/api/transactions/transaction', auth, async (request, response, next) => {
3   const { userId, tickets } = request.body;
4
5   try {
6     const session = await mongoose.startSession();
7     session.startTransaction();
8
9     try {
10       // Iterate through each ticket in the request and update seats if required
11       for (const ticket of tickets) {
12         const { eventId, seatNumbers } = ticket;
13
14         // Check if seat management is required for the event
15         const event = await EventDAO.model.findById(eventId).session(session);
16         const requiresSeatManagement = event.category.includes('Kino');
17
18         if (requiresSeatManagement) {
19           await transactionDAO.updateIsAvailableForEventSeats(eventId,
20             seatNumbers, session);
21         }
22       }
23
24       // Transaction processing
25       const result = await
26         business.getTransactionManager().createNewOrUpdate(request.body, session);
27
28       await session.commitTransaction();
29       session.endSession();
30       response.status(200).send(result);
31     } catch (error) {
32       await session.abortTransaction();
33       session.endSession();
34       throw error;
35     }
36   } catch (error) {
37     console.error(error);
38     response.status(500).send({ error: 'Server Error' });
39   }
40 });

```

*Listing. 3.5. Proces dokonywania transakcji.*

Istota przeprowadzania operacji w obrębie sesji Mongo jest szczególnie widoczna na przykładzie przedstawionym na listingu 3.6. Metoda startEventTransaction() odpowiedzialna jest za utworzenie nowego wydarzenia. Ze względu na to, że na wydarzenie w bazie danych składa się wiele innych odrębnych dokumentów, zaistniała konieczność utworzenia niektórych z nich przed utworzeniem samego wydarzenia. Gdyby nie zastosowano sesji, to jeżeli na jakimś etapie tworzenia dokumentów pojawiłby się błąd, mogłoby dojść do sytuacji,

gdzie tworzona jest tylko część dokumentów. Po utworzeniu sesji i rozpoczęciu transakcji tworzone są dokumenty reprezentujące bilety wydarzenia. Następnie bilety dodawane są do wydarzenia i tworzony jest dokument wydarzenia. Ostatecznie wydarzenie dodawane jest do tablicy – pola w bazie danych znajdującego się u danego organizatora, zawierającego id wydarzeń, których jest właścicielem. W linijce 12. oraz 13. jeżeli wszystko przebiegło pomyślnie transakcja jest akceptowana, a sesja się kończy. Jeżeli na którymś etapie dojdzie do błędu wszystkie operacje zostają cofnięte (ang. *rollback*).

```
1  async function startEventTransaction(newEventDetails) {
2    let session;
3    try {
4      session = await mongoose.startSession();
5      session.startTransaction();
6
6      const ticketIds = await
7      TicketDAO.createTicketsAndGetIds(newEventDetails.tickets, session);
8
8      const createdEvent = await createEventWithTickets(newEventDetails, ticketIds,
9      session);
10
10     await addEventToOrganizerOwnedEvents(newEventDetails, createdEvent.id,
11     session);
12
12     await session.commitTransaction();
13     session.endSession();
14
14     return createdEvent;
15   } catch (error) {
16     if (session) {
17       await session.abortTransaction();
18       session.endSession();
19     }
20     throw new Error('Transaction aborted: ' + error.message);
21   }
22 }
```

*Listing. 3.6. Tworzenie nowego wydarzenia w obrębie jednej transakcji.*

Głównym zadaniem aplikacji webowej jest dostarczenie interfejsów graficznych. Dlatego jej kod skupia się głównie na prezentacji danych oraz zasobów bazy danych. Interfejsy użytkownika oraz organizatora zostały odpowiednio rozdzielone. Dodatkowo użytkownik w zależności od tego czy jest zalogowany czy nie posiada inny zakres możliwości oferowanych przez system. Takie rozwiązanie spowodowało, że zaszła potrzeba zabezpieczenia poszczególnych ekranów oraz funkcjonalności, nie tylko po stronie serwera, ale przede wszystkim po stronie aplikacji klienckich. W tym celu wykorzystane zostały różne

mechanizmy, jednym z nich jest mechanizm AuthGuard. Służy on do ochrony tras w aplikacji przed nieautoryzowanym dostępem. Zabezpieczenie tras wymaga utworzenia klasy AuthGuard, przedstawionej na listingu 3.7. Klasa ta implementuje interfejs CanActivate, który definiuje metodę `canActivate()`, wywoływaną za każdym razem gdy użytkownik próbuje uzyskać dostęp do trasy chronionej. Listing 3.8. przedstawia natomiast fragment w którym zdefiniowane są trasy prowadzące do ekranów przeznaczonych tylko dla użytkownika zalogowanego, zabezpieczonych mechanizmem AuthGuard.

```

1  @Injectable({
2    providedIn: 'root'
3  })
4  export class AuthGuard implements CanActivate {
5
5  constructor(private authService: AuthService, public router: Router) {
6    }
7
7  canActivate(
8    route: ActivatedRouteSnapshot,
9    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
10  if (!this.authService.isLoggedIn()) {
11    this.router.navigateByUrl('/');
12    return false;
13  } else {
14    return true;
15  }
16}
17}
18}

```

Listing. 3.7. Klasa AuthGuard.

```

1  {
2    path: 'transactions',
3    component: TransactionListComponent,
4    canActivate: [AuthGuard]
5  },
6  {
7    path: 'cart/order',
8    component: OrderComponent,
9    canActivate: [AuthGuard]
10 }

```

Listing. 3.8. Fragment definicji tras aplikacji webowej.

Jak wspomniano wcześniej, aplikacja webowa w głównej mierze służy do prezentacji zasobów. Na listingu 3.9. przedstawiono fragment odpowiedzialny za pobranie danych z wykorzystaniem serwisu komunikującego się z aplikacją serwerową. Po pobraniu danych

wykonywana jest metoda `fetchTicketsForEachEvent()` która iteruje po elementach `items$`. Dla każdego wydarzenia pobierane są dane o biletach, które zapisywane są w obiekcie `ticketsMap` gdzie kluczem jest `id` wydarzenia. Listing 3.10. przedstawia sposób wyświetlenia pobranych danych użytkownikowi. Szablon HTML wykorzystuje w tym celu dyrektywę strukturalną `*ngFor` iterującą po elementach `items$` i przekazującą odpowiednie dane do komponentu `<event-card />` odpowiedzialnego za wyświetlenie pojedynczej karty wydarzenia.

```

1  getAll() {
2    this.service.getAll().subscribe((response) => {
3      this.items$ = response;
4      this.dataLoaded = true;
5      this.fetchTicketsForEachEvent();
6    });
7  }
8  fetchTicketsForEachEvent() {
9    this.items$.forEach((event: any) => {
10      this.service.getTicketsForEvent(event.id).subscribe((res: any) => {
11        this.ticketsMap[event.id] = res;
12      });
13    });
14  }

```

Listing. 3.9. Fragment logiki odpowiedzialnej za pobranie danych.

```

1  <div *ngFor="let item of items$">
2    <event-card
3      [id]="item.id"
4      [title]="item.title"
5      [image]="item.image"
6      [text]="item.text"
7      [tickets]="ticketsMap[item.id]"
8      [date]="item.date"
9      [location]="item.location"
10   ></event-card>
11 </div>

```

Listing. 3.10. Fragment struktury HMTL.

Istotnym elementem aplikacji webowej jest jej komunikacja z aplikacją serwerową. Komunikacja realizowana jest za pośrednictwem serwisów. Fragment serwisu odpowiedzialnego za zapytania związane z pobieraniem danych przedstawiony został na listingu 3.11. Metoda `getAll()` wykonuje zapytanie http pobierające dane wydarzeń. Metoda `getTicketsForEvent()` odpowiada za wykonanie zapytania zwracającego wszystkie bilety konkretnego wydarzenia. Metoda `getCart()` zwraca koszyk użytkownika, w odróżnieniu od poprzednich ta metoda została dodatkowo zabezpieczona i wymaga uwierzytelnienia. Dlatego w nagłówku zapytania przesyłany jest token.

```

1  getAll() {
2    return this.http.get(this.url + '/api/events');
3  }
4  getTicketsForEvent(eventId: string) {
5    return this.http.get(this.url + '/api/events/' + eventId + '/tickets');
6  }
7  getCart(userId: string) {
8    let headers = new HttpHeaders({'Authorization': 'Bearer ' + this.token,
9      'Content-Type': 'application/json'})
10   return this.http.get(this.url + '/api/user/' + userId + '/cart', {headers: headers});
11 }

```

*Listing. 3.11. Fragment serwisu DataService.*

Aplikacja mobilna i aplikacja webowa mają wspólny cel – dostarczenie interfejsu graficznego dla użytkowników. Poniższy kod skupia się głównie na implementacji funkcjonalności prezentacji danych oraz interfejsu użytkownika, które stanowią największą część logiki aplikacji mobilnej.

Na listingu 3.12. przedstawiony został fragment odpowiedzialny za pobranie danych dotyczących szczegółów wydarzenia. We fragmencie wykorzystane zostały hooki do utworzenia dwóch stanów przechowujących dane oraz wywołanie asynchronicznych funkcji dostępnych w serwisie (listing 3.14.). Funkcje dostępne w serwisie wykonują zapytania do aplikacji serwerowej podobnie jak jest to w przypadku aplikacji webowej. Do wykonywania zapytań HTTP wykorzystana została popularna biblioteka Axios.

Prezentacja danych w komponentach funkcyjnych polega na zwracaniu elementów interfejsu użytkownika za pomocą funkcji *return()*, co zostało zobrazowane w kodzie na listingu 3.13. W tym fragmencie wykorzystano metodę *map()*, służącą do iteracji po elementach tablicy oraz tworzenia nowej tablicy na podstawie przekształceń każdego elementu oryginalnej tablicy. W kodzie w omawianym fragmencie, dla każdego artysty z tablicy *artists*, generowany jest odpowiedni widok zawierający grafikę przedstawiającą danego artystę, umieszczony wewnętrz komponentów *<View>* i *<Image>*, co umożliwia dynamiczne wyświetlanie danych.

```

1  useEffect(() => {
2      let eventId = props.route.params.eventId;
3      const fetchData = async () => {
4          const eventDetails = await DataService.getEventDetailsById(eventId);
5          setEventDetails(eventDetails);
6          const artists = await DataService.getArtistDetailsById(eventId);
7          setArtists(artists);
8      };
9      fetchData();
10 }, [props.route.params.eventId]);

```

*Listing. 3.12. Fragment komponentu funkcyjnego EventDetailsScreen.*

```

1  return (
2      <ScrollView horizontal>
3          {artists.map((artist, index) => (
4              <View key={index} style={styles.artistCircle}>
5                  {artist && artist.image && (
6                      <Image source={{ uri: artist.image }} style={styles.artistImage}>
7                      </Image>
8                  )}
9                  </View>
10             )));
11     </ScrollView>
12 );

```

*Listing. 3.13. Fragment funkcji return.*

```

1  async getEventDetailsById(eventId) {
2      try {
3          const response = await axios.get(this.url+ '/events/' +eventId);
4          return response.data;
5      } catch (error) {
6          console.error('Error fetching event details:', error);
7          throw error;
8      }
9  }
10
11  async getArtistDetailsById(eventId) {
12      try {
13          const response = await axios.get(this.url+ '/events/' +eventId + '/artists');
14          return response.data;
15      } catch (error) {
16          console.error('Error fetching artist details:', error);
17          throw error;
18      }

```

*Listing. 3.14. Fragment serwisu DataService.*

## 4. Testy

System SSB to obszerny system, zapewniający klientom szeroki zakres możliwości. Przetwarza on dane wrażliwe oraz umożliwia realizację różnorodnych operacji biznesowych, takich jak zakup biletów czy dokonywanie płatności. Zachodzi więc konieczność solidnego zabezpieczenia i przetestowania newralgicznych części systemu. Rozdział 4. skupia się właśnie na tym zagadnieniu.

### 4.1. Testy jednostkowe

W celu przetestowania poszczególnych części systemu wykorzystane zostały trzy rodzaje testów. Niniejszy podrozdział skupia się na przedstawieniu pierwszego rodzaju testów – testów jednostkowych.

Testy jednostkowe zwane również modułowymi skupiają się wyłącznie na małych i odizolowanych częściach kodu. Ich celem jest zweryfikowanie, czy dany fragment lub instrukcja działa zgodnie z zamierzeniami [19]. Z wykorzystaniem testów jednostkowych sprawdzone zostały kluczowe metody po stronie aplikacji klienckich.

W przypadku aplikacji webowej do testowania wykorzystany został zestaw narzędzi Jasmine. Na listingu 4.1. przedstawiony został przykładowy test jednostkowy, sprawdzający poprawność walidacji pól formularza. Test rozpoczyna się od deklaracji nazwy grupy testów zawartych w bloku `describe()`. Następnie tworzony jest blok `beforeEach()`, który wykonywany jest przed każdym testem. Blok ten służy do zainicjowania serwisów komponentów z użyciem symulowanych obiektów (ang. *mock*). Dzięki wykorzystaniu takich imitacji serwisów czy komponentów można odizolować daną metodę lub fragment kodu od całego systemu. Następnie pojawia się kolejny blok `describe()` w którym deklarowana jest grupa testów dotyczących metody walidacji pól formularza. Pierwszy test deklarowany jest w bloku `it()`. Ciało testu została zaimplementowana z wykorzystaniem wzorca AAA (ang. *Arrange Act Assert*), który narzuca odpowiednią organizację struktury testów. Zastosowanie takiego wzorca pozwala utrzymać porządek i przejrzystość testów. Linijki 18. oraz 19. to pierwsza faza testowania w której przygotowywane jest środowisko testowe. Następna faza – działanie (ang. *Act*), rozpoczyna się w linijce 21. gdzie wykonywane jest np. wywołanie metody walidującej formularz. Ostatnią fazą testu jest asercja. W tej fazie sprawdzane jest czy otrzymane wyniki zgodne są z oczekiwaniami.

Listing. 4.1. Test weryfikujacy działanie metody walidujacej pola formularza.

W celu przetestowania aplikacji webowej w sumie przeprowadzonych zostało 24. testy jednostkowe (rys. 4.1.).

## Karma v 6.4.2 - connected; test: complete;

Chrome 120.0.0.0 (Windows 10) is idle

 Jasmine 4.6.0  
• • • • • • • • • • • • • • • • •

24 specs, 0 failures, randomized with seed 09723

Rys. 4.1. Przebieg testów jednostkowych aplikacji webowej. (źródło: Opracowanie własne)

Aplikacja mobilna została przetestowana z wykorzystaniem zestawu narzędzi Jest. Jako, że głównym zadaniem aplikacji mobilnej jest prezentacja zasobów i komunikacja z aplikacją serwerową, głównie testowana była poprawność wyświetlania poszczególnych ekranów oraz mniejsze funkcje odpowiedzialne za wyświetlanie lub zmianę wyglądu na ekranie aplikacji. Przykładowy test jednostkowy przeprowadzony dla jednej z metod używanej w aplikacji mobilnej przedstawiony został na listingu 4.2.

```
1  describe('renderText method', () => {
2    test('returns additionalText truncated to 200 characters followed by ellipsis if length
3      // Arrange
4      const eventDetails = {
5        additionalText: 'This is a longer text that exceeds 200 characters for testing
6        'This is a longer text that exceeds 200 characters for testing purposes.' +
7        'This is a longer text that exceeds 200 characters for testing purposes.' +
8        'This is a longer text that exceeds 200 characters for testing purposes.' +
9        'This is a longer text that exceeds 200 characters for testing purposes.' +
10       'This is a longer text that exceeds 200 characters for testing purposes.' +
11       'This is a longer text that exceeds 200 characters for testing purposes.', +
12     };
13     const showFullText = false;
14
15     //Act
16     const expected = 'This is a longer text that exceeds 200 characters for testing
17     purposes.This is a longer text that exceeds 200 characters for testing purposes.This is
18     a longer text that exceeds 200 characters for test...';
19
20     // Assert
21     expect(renderText(eventDetails, showFullText)).toEqual(expected);
22   });
23});
```

Listing. 4.2. Test poprawności działania funkcji skracającej wyświetlany tekst.

Przebieg testów aplikacji mobilnej przedstawiony został na rysunku 4.2.

```
Snapshot Summary
  › 1 snapshot written from 1 test suite.

Test Suites: 6 passed, 6 total
Tests:       9 passed, 9 total
Schemas:    1 written, 4 passed, 5 total
Time:        3.454 s

Ran all test suites.
```

Rys. 4.2. Przebieg testów jednostkowych aplikacji mobilnej.

## 4.2. Testy integracyjne

Testy integracyjne tworzone są w celu zweryfikowania współpracy pomiędzy różnymi częściami systemu. Taki rodzaj testów pozwala upewnić się, czy wszystkie elementy realizują założone procesy biznesowe. A także, czy infrastruktura wspiera kompleksowy system oraz obsługę wyjątków. Z tego względu testy integracyjne wykorzystane zostały do solidnego przetestowania istotnych części aplikacji serwerowej, której głównym zadaniem jest współpraca z wieloma modułami oraz obsługa błędów krytycznych.

Testy integracyjne w odróżnieniu od testów jednostkowych operują na rzeczywistych modułach, serwisach oraz bazie danych. Dlatego w celu poprawnego przetestowania współpracy pomiędzy modułami konieczne było przygotowanie środowiska testowego. Testowanie wymagało, aby baza danych była czyszczona przed każdym testem. W celu zapewnienia integralności bazy danych ze środowiska produkcyjnego, środowisko testowe korzysta z bazy danych umieszczonej w pamięci hosta (ang. *in-memory mongo database*).

Testy integracyjne zostały napisane z wykorzystaniem SuperTest, będącą biblioteką Node.js wspomagającą testowanie interfejsów API. Wykorzystano, również zestaw narzędzi Jest pozwalający na proste testowanie poprawności działania kodu JavaScript.

Przykładowy test integracyjny napisany z użyciem SuperTest został przedstawiony na listingu 4.3. Testy integracyjne zostały oparte o ten sam wzorzec co testy jednostkowe. Test przedstawiony na listingu 4.3. zaczyna się od zadeklarowania nazwy grupy testów w bloku `describe()`, następnie w ciele bloku `it()` test rozpoczyna się od fazy przygotowania środowiska

testowego. W tym przypadku w bazie danych tworzony jest użytkownik. Następnie symulowane jest jego logowanie do systemu oraz pobierany jest token potrzebny do uwierzytelnienia. W fazie działania wykonywane jest zapytanie typu get, jako, że zapytanie wymaga uwierzytelnienia przesyłany jest również pobrany wcześniej token. W ostatniej fazie wykonywana jest asercja sprawdzająca czy zapytanie zostało wykonane poprawnie. W tym przypadku zwrócenie przez aplikację serwerową kodu 200 symbolizuje poprawne wykonanie zapytania.

```

1  describe('Users cart – Get users cart', () => {
2    it('should get users cart and respond with 200 status code', async () => {
3      // Arrange
4      const userData = {
5        name: 'TEST',
6        email: 'email@gmail.com',
7        password: 'zaq123!@K'
8      };
9
10     const loginCredentials = {
11       login: 'TEST',
12       password: 'zaq123!@K'
13     };
14
15     const createdUserResponse = await request(app)
16       .post('/api/user/create')
17       .send(userData);
18
19     const { userId: userId } = createdUserResponse.body;
20
21     const loginResponse = await request(app)
22       .post('/api/user/auth')
23       .send(loginCredentials);
24     const token = loginResponse.body.token;
25
26     // Act
27     const getCartResponse = await
28     request(app).get(`/api/user/${userId}/cart`).set('Authorization', "Bearer "+token)
29
30     // Assert
31     expect(getCartResponse.status).toBe(200);
32   });
33 });

```

*Listing. 4.3. Test poprawności pobierania koszyka użytkownika.*

W celu przetestowania najważniejszych części i funkcjonalności aplikacji serwerowej wykonanych zostało 21 testów integracyjnych. Przebieg testów przedstawiony został na rysunku 4.3.

```
Tests:          21 passed, 21 total
S snapshots: 0 total
Time:          2.834 s, estimated 3 s
Ran all test suites.
```

*Rys. 4.3. Przebieg testów integracyjnych aplikacji serwerowej.*

### 4.3. Testy manualne

Testy manualne są formą testów oprogramowania, w której to dedykowany do tego tester sprawdza działanie poszczególnych możliwości systemu ręcznie. Główną przewagą tego typu testów jest możliwość znalezienie błędów lub awarii trudnych do wykrycia przez testy automatyczne. Przykładem takich błędów są np. problemy wizualne. Testy manualne są więc dobrym sposobem na testowanie interfejsów graficznych. Z tego powodu testy manualne zostały wykorzystane do przetestowania aplikacji klienckich. Proces tworzenia i przeprowadzenia testu manualnego rozpoczyna się od przygotowania scenariuszy testowych. Scenariusze zawierają kroki do wykonania, oczekiwane rezultaty i warunki testu. Kolejnym krokiem jest wykonanie testu w oparciu o wytworzzone scenariusze testowe. W przypadku wykrycia błędów, tworzone są raporty zawierające szczegóły znalezionych problemów, w tym kroki do ich odtworzenia i informacje na temat środowiska testowego. Przykładowe scenariusze testowe zawarte zostały w tabeli 4.1.

Tab. 4.1. Scenariusze testowe.

Test Case ID	Opis	Warunki wstępne	Kroki testowe	Oczekiwany wynik
TC_01	Otwarcie strony logowania.	Dostęp do strony logowania.	1. Kliknij w ikonę awatara w menu nawigacyjnym. 2. Kliknij w rozsuwanym menu przycisk "Zaloguj".	Strona logowania poprawnie się otwiera.
TC_02	Otwarcie strony rejestracji.	Dostęp do strony rejestracji.	1. Kliknij w ikonę awatara w menu nawigacyjnym. 2. Kliknij w rozsuwanym menu przycisk "Zarejestruj się".	Strona rejestracji poprawnie się otwiera.
TC_03	Logowanie poprzez wprowadzenie poprawnych danych.	Strona logowania jest już otwarta.	1. Wypełnij prawidłowo wszystkie pola formularza. 2. Kliknij przycisk "Zaloguj się".	Użytkownik zostaje poprawnie zalogowany.
TC_04	Przeglądanie koszyka.	Użytkownik jest zalogowany.	1. Kliknij w ikonę awatara w menu nawigacyjnym. 2. Kliknij w rozsuwanym menu przycisk "Koszyk"	Użytkownik może zobaczyć zawartość koszyka z wybranymi biletami lub informację o braku jakichkolwiek biletów.
TC_05	Sprawdzenie reakcji formularza na puste pole	Formularz logowania lub rejestracji posiada puste pola.	1. Pozostaw jedno z pól formularza pustym, np. pole loginu lub hasła. 2. Kliknij przycisk "Zaloguj się".	Formularz informuje użytkownika o konieczności wypełnienia wszystkich pól lub blokuje wysłanie danych.

W projekcie napisanych zostało w sumie 19. scenariuszy dla testera manualnego.

## Podsumowanie

W niniejszej pracy inżynierskiej zaprojektowany oraz zaimplementowany został system umożliwiający sprzedaż biletów na różne wydarzenia. System wyróżnia się kompleksowym podejściem do obsługi klientów – zarówno osób kupujących bilety, jak i organizatorów wydarzeń. Dostarcza różnorodne narzędzia wspierające proces tworzenia wydarzeń oraz analizy sprzedaży.

W pierwszym rozdziale niniejszej pracy przeprowadzona została analiza technologiczna, która jest niezbędnym etapem poprzedzającym implementację systemu. Drugi rozdział skupił się na analizie dotyczącej zakresu funkcjonowania systemu. W rozdziale tym dokonano również szczegółowej analizy możliwości poszczególnych aktorów w obrębie zaprojektowanego systemu. Przedstawione zostały także wymagania funkcjonalne oraz poza funkcjonalne systemu SSB. W rozdziale trzecim przedstawione zostały ekran aplikacji mobilnej oraz webowej. Przedstawiona również została struktura aplikacji serwerowej oraz najciekawsze i najistotniejsze fragmenty kodu poszczególnych modułów. Ostatni rozdział skupił się wyłącznie na przeprowadzonych testach oraz ich rodzajach.

W pracy skupiono się na stworzeniu kompleksowego systemu składającego się z trzech głównych modułów: aplikacji webowej w technologii Angular, aplikacji mobilnej w React Native oraz aplikacji serwerowej opartej na Node.js i Express.js. Pomimo przewagi języka TypeScript nad językiem JavaScript, na jego użycie zdecydowano się jedynie w przypadku aplikacji webowej. Decyzja ta podyktowana była kompatybilnością oraz dostępnością różnych bibliotek i narzędzi wykorzystanych w przypadku aplikacji serwerowej oraz mobilnej. Dlatego aplikacje mobilna oraz serwerowa w całości oparte zostały o język JavaScript. Takie rozwiązanie stanowi pewną zaletę, ponieważ jeden programista znający język JavaScript, będący wciąż popularniejszym językiem niż TypeScript, może zarządzać zarówno aplikacją serwerową, jak i mobilną. Użyty w przypadku aplikacji webowej język TypeScript nie powinien stanowić problemu dla programisty posługującego się językiem JavaScript. Jako, że TypeScript jest nadzbiorem języka JavaScript, każdy prawidłowy kod JavaScript jest również prawidłowym kodem TypeScript, co czyni ten język łatwym w zrozumieniu dla programisty JavaScript.

Końcowo, udało się osiągnąć założone cele pracy. Udało się zaimplementować trzy główne moduły systemu, co pozwoliło na kompleksowe pokrycie różnych platform dzięki

modułowej strukturze systemu. Opracowano panel organizatora wydarzeń, który umożliwia podstawowe zadania, takie jak tworzenie wydarzeń, przeglądanie aktywnych wydarzeń oraz analizę statystyk sprzedaży biletów. Zaimplementowano interfejs użytkownika, który zapewnia dostęp do różnorodnych funkcji, takich jak przeglądanie wydarzeń, ich filtrowanie na podstawie kryteriów (np. lokalizacji), dodawanie biletów do koszyka oraz zakup biletów. Dodatkowo użytkownik może wybierać miejsca na podstawie schematów sal czy pomieszczeń, dodawać wydarzenia do ulubionych oraz obserwowanych. System prezentuje oferty dostosowane do preferencji. Wykonano łącznie 54. testów sprawdzających poprawność zaimplementowanego kodu.

Podsumowując, udało się zbudować system, który oferuje wiele możliwości oraz dostarcza wygodne narzędzia zarówno dla użytkowników, jak i organizatorów wydarzeń.

System pomimo oferowania wielu możliwości, wciąż ma duży potencjał na rozwój. Potencjalnymi opcjami rozwoju są np. rozbudowa filtracji na podstawie dodatkowych preferencji użytkownika. Bardziej zaawansowany system tworzenia schematów pokoi i pomieszczeń. Umożliwiający organizatorom wydarzeń tworzenia bardziej skomplikowanych schematów, np. dla kin czy teatrów, co pozwoli na lepszą prezentację układu przestrzeni dla potencjalnych uczestników. Dodanie możliwości prezentacji w czasie rzeczywistym potencjalnego wyglądu strony wydarzenia, poprzez lepsze formatowanie tekstów i opisów, co może przyciągnąć uwagę użytkowników. Wprowadzenie integracji z usługami Google, takimi jak Google OAuth i Google Maps, dla bardziej precyzyjnego wyboru lokalizacji oraz zwiększenia komfortu użytkowników korzystających z aplikacji. Dodanie bardziej zaawansowanej analizy danych biznesowych i zainteresowań użytkowników dla organizatorów. Rozbudowane raporty i statystyki w panelu organizatorów mogłyby pomóc w lepszym zrozumieniu i wykorzystaniu danych.

Rozwój tych elementów systemu mógłby jeszcze bardziej zwiększyć jego funkcjonalność i użyteczność zarówno dla użytkowników, jak i organizatorów wydarzeń.

## Dodatek A. Szablony dokumentacji technicznej.

W Tabelach 5.1. – 5.5. zostały przedstawione szablony dokumentacji technicznej. Na podstawie szablonów Tabel 5.1. – 5.4. wykonane zostały tabele w rozdziale 2. Na podstawie szablonu tabeli 5.5. wykonane zostały listingi w rozdziale 3.

*Tab. 5.1. Tytuł tabeli.*

<b>ID: identyfikator</b>
<b>Nazwa: nazwa</b>
<b>Opis:</b> Opis w tabeli.

*Tab. 5.2. Nazwa przypadku użycia.*

<b>ID: identyfikator</b>
<b>Nazwa: nazwa</b>
<b>Aktorzy główni:</b> aktor główny
<b>Aktorzy pomocniczy:</b> aktor pomocniczy
<b>Poziom:</b> poziom
<b>Priorytet:</b> priorytet
<b>Opis:</b> Opis przypadku użycia.
<b>Wyzwalacze:</b> 1. Nazwa wyzwalacza.
<b>Warunki początkowe:</b> 1. Warunek początkowy.
<b>Warunki końcowe:</b> 1. Warunek końcowy.
<b>Scenariusz główny:</b> 1. Kroki scenariusza.
<b>Scenariusz alternatywny i rozszerzenia:</b> 1. Kroki scenariusza alternatywnego.
<b>Wyjątki:</b> 1. Wyjątki.
<b>Dodatkowe wymagania:</b> <ul style="list-style-type: none"><li>• Wymagania dodatkowe.</li></ul>

Tab. 5.3. Nazwa wymagania NFR.

<b>Identyfikator</b>	Identyfikator NFR.
<b>Kategoria</b>	Kategoria
<b>Priorytet</b>	Priorytet
<b>Opis</b>	Opis wymagania NFR.

Tab. 5.4. Nazwa tabeli.

<b>Metoda</b>	<b>Ścieżka</b>	<b>Przekazywane dane</b>	<b>Autoryzacja</b>	<b>Opis</b>
Metoda	Ścieżka	Dane	Autoryzacja	Opis

Linijka kodu	Kod
--------------	-----

Listing. 5.5. Nazwa listingu.

## Bibliografia

- [1] Jon Duckett, *HTML i CSS. Zaprojektuj i zbuduj witrynę WWW*, Gliwice, Helion, 2017, ISBN: 978-83-283-4480-8
- [2] Opis technologii JavaScript (dostęp 18.10.2023) [Online] <https://vavatech.pl/technologie/jazyki-programowania/javascript>
- [3] Yakov Fain & Anton Moiseev, *Angular 2 Programowanie z użyciem TypeScript*, Gliwice, Helion, 2017, ISBN: 978-83-283-3639-1
- [4] Opis narzędzia Angular CLI oraz pojęcie Scaffolding'u (dostęp 18.10.2023) [Online] <https://reintech.io/blog/how-to-use-angular-cli-scaffolding-applications>
- [5] Opis pojęcia hot-reloading'u (dostęp 18.10.2023) [Online] <https://dev.to/jagroop2000/how-are-hot-reloading-and-live-reloading-in-react-native-different-3nj7>
- [6] Cykl życia komponentów w React Native (dostęp 19.10.2023) [Online] <https://www.netguru.com/blog/react-native-lifecycle>
- [7] Pojęcie hooków w React Native / React (dostęp 19.10.2023) [Online] <https://boringowl.io/blog/wprowadzenie-do-react-hooks-skroc-swoje-komponenty-i-zwieksz-wydajnosc>
- [8] Zestawienie najpopularniejszych języków programowania 2023 <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/>
- [9] Mike Cantelon, Marc Harter, TJ Holowaychuk, Nathan Rajlich, *Node.js w Akcji*, Gliwice, Helion, 2014, ISBN: 978-83-246-9681-9
- [10] Opis standardu JSON Web Token (dostęp 20.10.2023) [Online] <https://jwt.io/introduction>
- [11] Algorytmy szyfrujące i ochrona danych (dostęp 20.10.2023) [Online] <https://www.ibm.com/docs/pl/i/7.5?topic=cryptography-concepts>
- [12] Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, *Przewodnik po MongoDB, Wydajna i skalowalna baza danych, Wydanie III*, Gliwice, Helion SA, 2021, ISBN: 978-83-283-6534-6
- [13] Sposoby zabezpieczania aplikacji webowej (dostęp 20.10.2023) [Online] <https://www.politykabezpieczenia.pl/pl/a/5-sposobow-na-zapewnienie-bezpieczenia-aplikacji-webowej>
- [14] Zabezpieczenie aplikacji poprzez użycie Firewallu (dostęp 20.10.2023) [Online] <https://apius.pl/obszar-dzialania/bezpieczenstwo-informacji/bezpieczenstwo-aplikacji-i-danych>
- [15] ROZPORZĄDZENIE PARLAMENTU EUROPEJSKIEGO i RADY (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych). Dziennik Urzędowy Unii Europejskiej, L 119/1, 4.5.2016.
- [16] Kary finansowe przewidziane przez naruszenie RODO (dostęp 21.10.2023) [Online] <https://resilia.pl/blog/rejestr-kar-rodo-polskie-firmy-od-momentu-wprowadzenia>
- [17] Wymagania poza-funkcjonalne (dostęp 28.10.2023) [Online] [https://www.cs.put.poznan.pl/jrojek/files/io1/requirements/NFR\\_opis.pdf](https://www.cs.put.poznan.pl/jrojek/files/io1/requirements/NFR_opis.pdf)
- [18] Dokumentacja narzędzia swagger.io (dostęp 29.10.2023) [Online] <https://swagger.io/docs/specification/about/>
- [19] Rafał Pawlak, *Testowanie oprogramowania podręcznik dla początkujących*, Gliwice, Helion, 2014, ISBN: 978-83-246-9308-5

## Spis tabel

Tab. 2.1. Charakterystyka użytkownika niezalogowanego. ....	19
Tab. 2.2. Charakterystyka użytkownika zalogowanego. ....	20
Tab. 2.3. Charakterystyka organizatora. ....	20
Tab. 2.4. Charakterystyka systemu bankowości. ....	20
Tab. 2.5. Przypadek użycia – Przeglądanie dostępnych wydarzeń. ....	22
Tab. 2.6. Przypadek użycia – Wyszukiwanie dostępnych wydarzeń. ....	23
Tab. 2.7. Przypadek użycia – Filtracja dostępnych wydarzeń. ....	24
Tab. 2.8. Przypadek użycia – Sortowanie dostępnych wydarzeń. ....	25
Tab. 2.9. Przypadek użycia – Rejestracja. ....	26
Tab. 2.10. Przypadek użycia – Logowanie. ....	27
Tab. 2.11. Przypadek użycia – Walidacja danych. ....	28
Tab. 2.12. Przypadek użycia – Przeglądanie dostępnych wydarzeń (MUZ). ....	31
Tab. 2.13. Przypadek użycia – Wyszukiwanie dostępnych wydarzeń (MUZ). ....	32
Tab. 2.14. Przypadek użycia – Filtracja dostępnych wydarzeń (MUZ). ....	33
Tab. 2.15. Przypadek użycia – Sortowanie dostępnych wydarzeń (MUZ). ....	34
Tab. 2.16. Przypadek użycia – Wylogowanie się (MUZ). ....	35
Tab. 2.17. Przypadek użycia – Zarządzanie koszykiem (MUZ). ....	36
Tab. 2.18. Przypadek użycia – Usuwanie biletu z koszyka (MUZ). ....	37
Tab. 2.19. Przypadek użycia – Kupno biletu (MUZ). ....	38
Tab. 2.20. Przypadek użycia – Podsumowanie zamówienia (MUZ). ....	39
Tab. 2.21. Przypadek użycia – Dodawanie biletu do koszyka (MUZ). ....	40
Tab. 2.22. Przypadek użycia – Dodawanie wydarzenia do ulubionych (MUZ). ....	41
Tab. 2.23. Przypadek użycia – Dodawanie wydarzenia do obserwowanych (MUZ). ....	42
Tab. 2.24. Przypadek użycia – Przeglądanie ulubionych wydarzeń (MUZ). ....	43
Tab. 2.25. Przypadek użycia – Usunięcie z ulubionych (MUZ). ....	44
Tab. 2.26. Przypadek użycia – Przeglądanie obserwowanych wydarzeń (MUZ). ....	45
Tab. 2.27. Przypadek użycia – Usunięcie z obserwowanych. (MUZ). ....	46
Tab. 2.28. Przypadek użycia – Rejestracja konta organizatora (MO). ....	48
Tab. 2.29. Przypadek użycia – Logowanie na konto organizatora (MO). ....	49
Tab. 2.30. Przypadek użycia – Walidacja danych (MO). ....	51
Tab. 2.31. Przypadek użycia – Wylogowanie się (MO). ....	52

Tab. 2.32. Przypadek użycia – Zarządzanie panelem administracyjnym (MO).....	53
Tab. 2.33. Przypadek użycia – Przeglądanie aktywnych wydarzeń (MO) .....	54
Tab. 2.34. Przypadek użycia – Edycja wydarzeń (MO) .....	55
Tab. 2.35. Przypadek użycia – Archiwizacja/Usunięcie wydarzenia (MO) .....	56
Tab. 2.36. Przypadek użycia – Przeglądanie raportów i statystyk (MO).....	57
Tab. 2.37. Przypadek użycia – Tworzenie wydarzeń (MO) .....	58
Tab. 2.38. Przypadek użycia – Walidacja i weryfikacja danych (MO) .....	59
Tab. 2.39. Wymagania NFR – NFR_SSB_01 .....	60
Tab. 2.40. Wymagania NFR – NFR_SSB_02 .....	60
Tab. 2.41. Wymagania NFR – NFR_SSB_03 .....	60
Tab. 2.42. Wymagania NFR – NFR_SSB_04 .....	60
Tab. 2.43. Wymagania NFR – NFR_SSB_05 .....	61
Tab. 2.44. Wymagania NFR – NFR_SSB_06 .....	61
Tab. 2.45. Wymagania NFR – NFR_SSB_07 .....	61
Tab. 2.46. Wymagania NFR – NFR_SSB_08 .....	61
Tab. 2.47. Wymagania NFR – NFR_SSB_09 .....	61
Tab. 2.48. Charakterystyka punktów końcowych służących do zarządzania artystami. ....	65
Tab. 2.49. Charakterystyka punktów końcowych służących do zarządzania wydarzeniami...	65
Tab. 2.50. Charakterystyka punktów końcowych służących do zarządzania biletami. ....	66
Tab. 2.51. Charakterystyka punktów końcowych służących do zarządzania transakcjami....	66
Tab. 2.52. Charakterystyka punktów końcowych służących do zarządzania użytkownikiem.	67
Tab. 2.53. Charakterystyka punktów końcowych służących do zarządzania organizatorem..	68
Tab. 4.1. Scenariusze testowe. ....	107
Tab. 5.1. Tytuł tabeli. ....	110
Tab. 5.2. Nazwa przypadku użycia. ....	110
Tab. 5.3. Nazwa wymagania NFR. ....	111
Tab. 5.4. Nazwa tabeli. ....	111

# Spis rysunków

Rys. 2.1. Diagram kontekstowy systemu SSB. ....	18
Rys. 2.2. Diagram przypadków użycia MUN. ....	21
Rys. 2.3. Diagram przypadków użycia MUZ. ....	30
Rys. 2.4. Diagram przypadków użycia MO. ....	47
Rys. 2.5. Wykaz schematów. ....	62
Rys. 2.6. Wykaz punktów końcowych systemu SSB. ....	63
Rys. 2.7. Wykaz punktów końcowych systemu SSB. ....	64
Rys. 2.8. Wykaz punktów końcowych systemu SSB. ....	64
Rys. 2.7. Schemat ERD bazy danych systemu SSB. ....	69
Rys. 2.8. Schemat klas systemu SSB. ....	70
Rys. 3.1. Strona główna. ....	71
Rys. 3.2. Zaawansowane wyszukiwanie wydarzeń. ....	72
Rys. 3.3. Strona główna – wydarzenia. ....	72
Rys. 3.4. Ekran tworzenia konta. ....	73
Rys. 3.5. Monit przy pierwszym logowaniu. ....	74
Rys. 3.6. Wybór preferencji. ....	74
Rys. 3.7. Sekcja związana z preferencjami użytkownika. ....	75
Rys. 3.8. Detal wydarzenia. ....	76
Rys. 3.9. Koszyk użytkownika. ....	76
Rys. 3.10. Proces płatności. ....	77
Rys. 3.11. Historia transakcji. ....	77
Rys. 3.12. Bilet w formacie pdf. ....	78
Rys. 3.13. Tabela polubionych wydarzeń. ....	78
Rys. 3.14. Tabela obserwowanych wydarzeń. ....	78
Rys. 3.15. Rozwijane menu nawigacyjne. ....	79
Rys. 3.16. Strona organizatora. ....	79
Rys. 3.17. Formularz rejestracji organizatora. ....	80
Rys. 3.18. Panel organizatora – aktywne wydarzenia. ....	80
Rys. 3.19. Panel organizatora – raporty i statystyki. ....	81
Rys. 3.20. Raporty i statystyki – statystyki pojedynczego wydarzenia. ....	81
Rys. 3.21. Tworzenie nowego wydarzenia. ....	82

Rys. 3.22. Kreator schematu kina.....	83
Rys. 3.23. Sekcja z utworzonymi biletami.....	83
Rys. 3.24. Okno wyboru artystów.....	84
Rys. 3.25. Artyści biorący udział w wydarzeniu.....	85
Rys. 3.26. Ekran logowania z monitem błędu.....	86
Rys. 3.31. Architektura aplikacji serwerowej.....	89
Rys. 4.1. Przebieg testów jednostkowych aplikacji webowej.....	103
Rys. 4.2. Przebieg testów jednostkowych aplikacji mobilnej.....	104
Rys. 4.3. Przebieg testów integracyjnych aplikacji serwerowej.....	106

# Spis listingów

Listing. 3.1. Tworzenie nowego konta. ....	90
Listing. 3.2. Uwierzytelnianie użytkownika.....	91
Listing. 3.3. Dodawanie biletu do koszyka. ....	92
Listing. 3.4. Pobranie wydarzeń na podstawie preferencji.....	93
Listing. 3.5. Proces dokonywania transakcji. ....	95
Listing. 3.6. Tworzenie nowego wydarzenia w obrębie jednej transakcji. ....	96
Listing. 3.7. Klasa AuthGuard.....	97
Listing. 3.8. Fragment definicji tras aplikacji webowej. ....	97
Listing. 3.9. Fragment logiki odpowiedzialnej za pobranie danych.....	98
Listing. 3.10. Fragment struktury HMTL.....	98
Listing. 3.11. Fragment serwisu DataService.....	99
Listing. 3.12. Fragment komponentu funkcyjnego EventDetailsScreen. ....	100
Listing. 3.13. Fragment funkcji return.....	100
Listing. 3.14. Fragment serwisu DataService.....	100
Listing. 4.1. Test weryfikujący działanie metody walidującej pola formularza.....	102
Listing. 4.2. Test poprawności działania funkcji skracającej wyświetlany tekst .....	103
Listing. 4.3. Test poprawności pobierania koszyka użytkownika. ....	105
Listing. 5.5. Nazwa listingu.....	111