Replay - Reserve&Play

Specyfikacja systemu

Paweł Frąckiewicz

Michał Gozdera

Mateusz Jastrzębiowski

Michał Klewicki

Wojciech Krawczyk

29.01.2021

Spis treści

1 Wstęp

Projekt jest realizowany w ramach przedmiotu **Inżynieria oprogramowania I** w semestrze zimowym 2020/2021. Jego celem jest przedstawienie specyfikacji systemu aplikacji **Replay - Reserve&Play**. W dużym uproszczeniu jest to aplikacja, za pomocą której można dokonać rezerwacji konsoli znajdującej się w gamebarze. W poniższej pracy pojęcie **gamebaru** jest rozumiane jako miejsce (lokal), który w swojej ofercie posiada możliwość zagrania na konsoli i do zoptymalizowania procesu rezerwacji wykorzystuje opisywaną aplikację. Szczegóły oraz dodatkowe informacje o aplikacji znajdują się w poniższych sekcjach.

2 Opis systemu

Aplikacja może być używana przez dwa niezależne podmioty: klienta, chcącego zarezerwować konsolę oraz menedżera, który przedstawia ofertę swojego lokalu. W systemie istnieje również osoba administratora, jednak jej funkcje są ściśle odseparowane i zabezpieczone od pozostałych podmiotów.

2.1 Perspektywa menedżera

Po kontakcie z administratorem systemu, menedżer może założyć konto dla swojego lokalu w systemie oraz przedstawić jego ofertę. Menedżer jest zobowiązany podać dane kontaktowe dla gamebaru: adres, numer telefonu, adres mailowy. Dane kontaktowe są widoczne dla klientów w widoku profilu lokalu. Oferta gamebaru zawiera dostępne typy konsoli, gry oraz przewidywane ramy czasowe otwarcia lokalu, a także poszczególnych slotów na każdej konsoli. Oczywiście, menedżer w czasie rzeczywistym może zmieniać aktualną ofertę np.: dodawać nowe konsole i gry w przypadku rozszerzenia biblioteki gier i urządzeń lub usuwać konsolę (być może tymczasowo) z powodu awarii. Każdy lokal może mieć własne promocje. Przewidywane są 3 rodzaje promocji:

1. darmowe wejście (wykorzystanie danego slotu)
2. kod zniżkowy (rabat naliczany do kosztu rezerwacji)
3. dodatkowy czas (darmowe wydłużenie zarezerwowanego slotu czasowego)

Promocje są wyświetlane na specjalnej tablicy w widoku profilu lokalu. Dodatkowo istnieje możliwość przydzielenia użytkownikowi statusu *stałego klienta*. W takim przypadku istnieje również możliwość dodawania konkretnych promocji wyłącznie dla stałych klientów.
We wcześniej wspomnianym widoku profilu lokalu, dostępna jest lista gier z ich opiniami oraz specjalne miejsce na opinie użytkowników dotyczące danego gamebaru. Więcej informacji o możliwości

zostawiania opinii jest przedstawione w poniższej sekcji dotyczącej użytkownika.
Menedżer posiada również możliwość pobierania raportów dotyczących analizy konkretnych okresów pod kątem dokonanych rezerwacji oraz wybieranych gier. Analizy są przygotowywane po stronie serwera pod nadzorem administratora.

2.2 Perspektywa użytkownika

Po uprzednim założeniu konta, użytkownik może przeglądać dostępne oferty gamebarów. Za pomocą podania swojej lokalizacji może sprawdzić najbliższe dostępne lokale. Oferty gamebarów mogą zostać posortowane i przefiltrowane wg następujących kryteriów:

1. aktualna odległość od lokalu
2. typ konsoli (Xbox One, Xbox X, Playstation 4, Playstation 5 itd.)
3. nazwa gry

Użytkownik może również przeglądać dostępne promocje w lokalu, a jeśli posiada status stałego klienta, ma możliwość sprawdzenia unikalnych promocji.
W celu podjęcia dobrych decyzji użytkownik może przejrzeć opinie na temat danego lokalu, a także dostępnych w nim gier. Sam również ma możliwość dodania opinii w tych dwóch kategoriach po odwiedzeniu danego lokalu. Użytkownicy są zobowiązani do dbałości o język w swoich wypowiedziach, ponieważ opinie zawierające wulgarne i niestosowane komentarze są zgłaszane do administratora, a autor podejrzanej wypowiedzi może zostać ukarany usunięciem z systemu. Każdy użytkownik może również sam zgłaszać napotkane przez siebie i wydające się niestosownymi opinie.
Po znalezieniu satysfakcjonującej opcji klient może przejść do etapu rezerwacji. Jeśli użytkownik potwierdzi rezerwację, dostępna będzie opcja opłacenia danej rezerwacji. Po dokonaniu płatności aplikacja będzie wyświetlać pozostały czas do rozpoczęcia aktywacji slotu gry. Jeśli klient zdecyduje się odwołać rezerwację, to będzie mieć taką możliwość, włącznie ze zwrotem części kosztów poniesionych w procesie wykonywania płatności. Użytkownik może również przeglądać historie swoich rezerwacji oraz ich statusy (opłacona, nieopłacona, anulowana).

2.3 Administrator

Jego zadaniem jest czuwanie nad poprawnym działaniem systemu aplikacji, zarządzanie zgłoszeniami opinii klientów, dodawanie nowych kont dla menadżerów gamebarów oraz profili dla ich lokali, a także dodawanie nowych opcji dotyczących generowania i pobierania raportów przedstawiających analizy w konkretnych lokalach, w pewnym okresie, uwzględniających ich zyski, wybierane przez klientów gry, typy konsoli, itd.

3 Architektura systemu

Cały system jest podzielony na trzy komponenty - moduły. Każdy moduł jest oddzielną aplikacją i zapewnia autonomiczne funkcjonalności:

- **Moduł gamebaru**

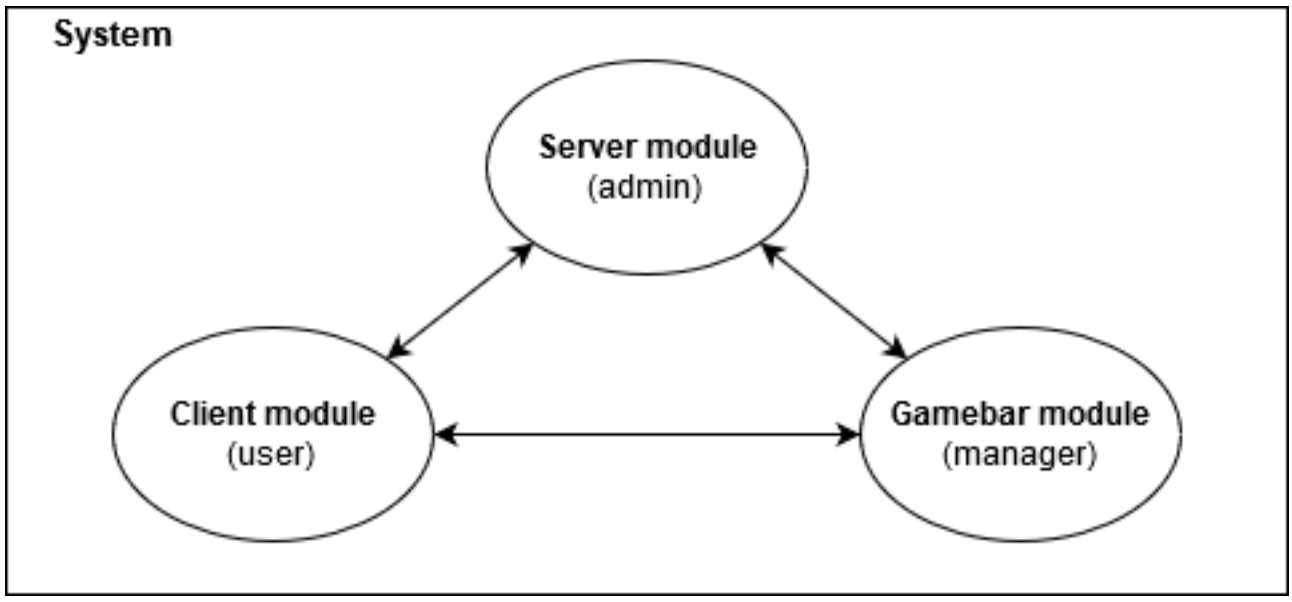
  Odpowiada za prezentowanie oferty danego lokalu oraz jej edycję. Każde działanie menadżera odbywa się w tym module, a następnie jego rezultaty wysyłane są do modułu serwera agregującego dane.

- **Moduł klienta**

  Odpowiada za przeprowadzanie akcji wykonywanych przez standardowego użytkownika. Przeglądanie ofert, dokonywanie rezerwacji itd. Następnie wyniki wykonanych akcji są wysyłane na serwer, który zapisuje rezultaty i uaktualnia właściwości gamebaru, którego one dotyczyły.

- **Moduł serwera**

  Odpowiada za odbieranie, agregowanie i przesyłanie informacji pomiędzy modułami klienta i gamebaru. Posiada funkcję do analizy danych w gamebarach. Nad jego działaniem czuwa osoba administratora.



Rysunek 1: Struktura systemu podzielona na moduły z uwzględnieniem głównych aktorów.

4 Historie użytkowników

4.1 Użytkownik

4.1.1 Historie użytkownika

Ja jako...	chcę...	żeby...
klient	mieć możliwość założenia konta w aplikacji, logowania się oraz odzyskiwania hasła	móc z niej korzystać.
klient	podać swoją lokalizację	znaleźć najbliższy gamebar.
klient	podać typ konsoli, na której chcę zagrać	sprawdzić jej dostępność.
klient	podać nazwę gry, w którą chcę zagrać	sprawdzić jej dostępność.
klient	zobaczyć dane kontaktowe gamebaru	móc się z nim skontaktować.
klient	zarezerwować konsolę w określonym czasie	móc na niej zagrać.
klient	opłacić rezerwację	nie tracić czasu na miejscu.
klient	sprawdzić opinie na temat gamebaru	podjąć właściwą decyzję.
klient	sprawdzić opinię na temat gry	podjąć właściwy wybór.
klient	dodać opinię na temat gry	podzielić się wrażeniami.
klient	dodać opinię na temat gamebaru	podzielić się wrażeniami.
klient	zgłosić opinię innego użytkownika	pomóc w moderacji sekcji opinii.
klient	zobaczyć aktualne rezerwacje	o nich nie zapomnieć.
klient	zobaczyć historię swoich rezerwacji.	
klient	mieć dostęp do ofert specjalnych	z nich skorzystać.

4.1.2 Kryteria akceptacji

Kryteria funkcjonalne

- MUST:
  - Czy mogę założyć konto?
  - Czy mogę się zalogować?
  - Czy mogę odzyskać hasło?
  - Czy mogę zarezerwować konsolę?
  - Czy mogę opłacić rezerwację?
  - Czy mogę zobaczyć listę swoich rezerwacji?
- SHOULD:

  - Czy mogę wyświetlić listę wszystkich gamebarów?
  - Czy mogę zobaczyć numer telefonu gamebaru?
  - Czy mogę przeczytać opinię o grze i gamebarze?
  - Czy mogę dodać opinię o grze i gamebarze?

Kryteria niefunkcjonalne

- COULD:
  - Czy opłacona rezerwacja zmieni kolor?
  - Czy lokalizacja najbliższego gamebaru zostanie podana jako pierwsza?
  - Czy moja rezerwacja będzie widoczna dla innych po 10 sekundach lub szybciej?

4.2 Menedżer gamebaru

4.2.1 Historie użytkownika

Ja jako...	chcę...	żeby...
menedżer	móc się zalogować, odzyskać hasło	korzystać z aplikacji.
menedżer	dodać nową konsolę/grę do gamebaru	poszerzyć ofertę.
menedżer	dodać nową promocję.	
menedżer	pisać ogłoszenia	informować klientów o bieżących sprawach.
menedżer	dodać galerię zdjęć gamebaru	klienci mogli zobaczyć, jak wygląda lokal.
menedżer	pobrać statystyki rezerwacji konsol i gier	zdobyć dane pomagające w podejmowaniu decyzji.
menedżer	zgłosić użytkownika do administratora	blokować szkodliwych użytkowników.
menedżer	zobaczyć dane kontaktowe zarejestrowanych klientów	móc się z nimi kontaktować.
menedżer	dodać rezerwację	uwzględnić rezerwacje dokonane telefonicznie.
menedżer	sprawdzić stan opłacenia rezerwacji.	
menedżer	usunąć rezerwację	zwolnić zarezerwowaną konsolę, gdyby osoba rezerwująca się nie pojawiła.
menedżer	dodać stałego klienta	oferować mu specjalne promocje.

4.2.2 Kryteria akceptacji

Kryteria funkcjonalne

- MUST:
  - Czy mogę się zalogować?
  - Czy mogę odzyskać hasło?
  - Czy mogę dodać nową konsolę/grę do oferty gamebaru?
  - Czy mogę dodać rezerwację i przypisać ją do danego klienta?
  - Czy mogę usunąć rezerwację?

  - Czy mogę sprawdzić stan opłacenia rezerwacji?

- SHOULD:
  - Czy mogę zobaczyć listę zarejestrowanych klientów wraz z ich danymi kontaktowymi?
  - Czy mogę oferować promocje stałym klientom?
  - Czy mogę dodawać ogłoszenia i galerie zdjęć?
  - Czy mogę pobrać statystyki rezerwacji konsol i gier?
  - Czy mogę dodać stałego klienta?
  - Czy mogę zgłosić użytkownika?

Kryteria niefunkcjonalne

- COULD:
  - Czy zgłoszenie użytkownika pojawi się u administratora w ciągu minuty od zgłoszenia?
  - Czy istnieją opcje filtrowania listy użytkowników?
  - Czy nowo dodane ogłoszenia będą wyróżnione?
  - Czy dodane konsole i gry będą dostępne do rezerwacji w mniej niż minutę?

4.3 Administrator

4.3.1 Historie użytkownika

Ja jako...	chcę...	żeby...
administrator	móc się zalogować, odzyskać hasło	nadzorować działanie systemu.
administrator	dodawać nowe konta gamebarów	uaktualniać listę działających lokali.
administrator	usuwać istniejące konta gamebarów	uaktualniać listę działających lokali.
administrator	usuwać/blokować zgłoszonych klientów	moderować bazę użytkowników.
administrator	usuwać zgłoszone opinie	moderować sekcję opinii.
administrator	zbierać i analizować dane klientów	udostępniać raporty managerom.
administrator	zobaczyć dane kontaktowe gamebaru	się z nim skontaktować.

4.3.2 Kryteria akceptacji

Kryteria funkcjonalne

- MUST:
  - Czy mogę się zalogować?
  - Czy mogę odzyskać hasło?
  - czy mogę dodawać oraz usuwać konta gamebarów?
  - Czy mogę zobaczyć listę gamebarów i uzyskać ich dane kontaktowe?
- SHOULD:
  - Czy mogę zobaczyć listę zgłoszonych użytkowników i opinii?
  - Czy mogę zablokować zgłoszonego użytkownika?
  - Czy mogę usunąć zgłoszoną opinię?
  - Czy mogę uzyskać dane o klientach, historii ich rezerwacji?

Kryteria niefunkcjonalne

- COULD:
  - Czy istnieje możliwość filtrowania listy gamebarów?
  - Czy dodane konto gamebaru będzie dostępne po minucie od dodania?
  - Czy lista zgłoszonych użytkowników/opinii zawiera informacje o ilości zgłoszeń?

5 Funkcjonalności

5.1 Moduł administratora

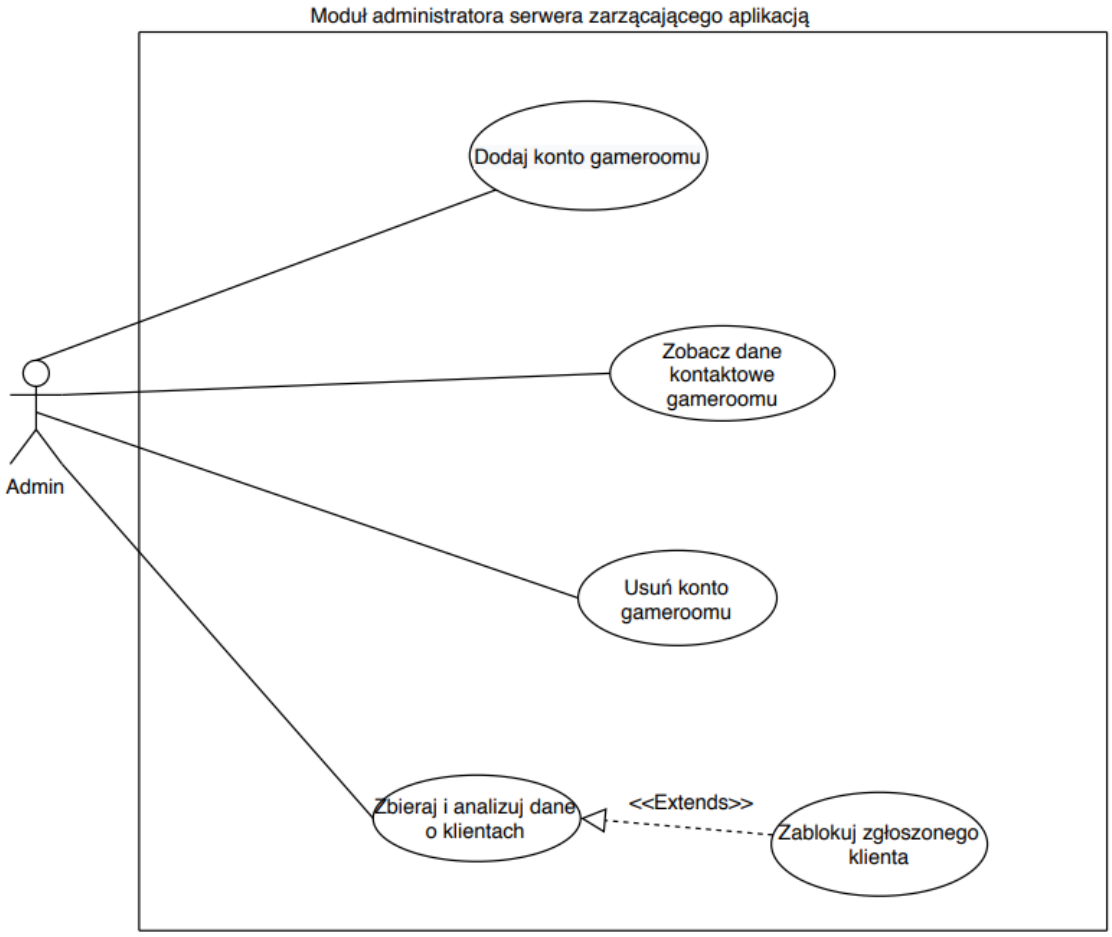
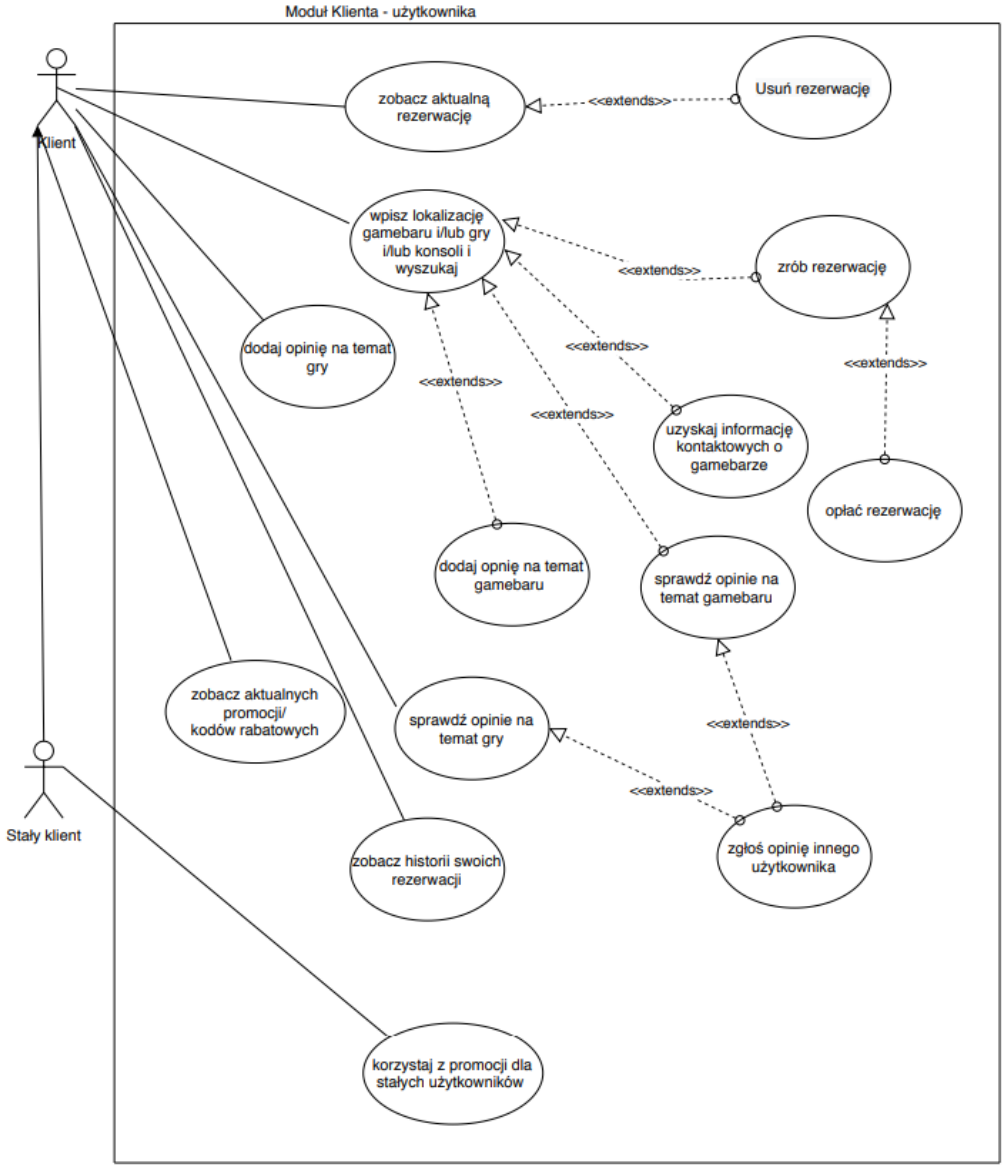


Diagram przedstawia moduł administratora. W naszym systemie administrator zarządza kontami menedżerów i klientów. Administrator odpowiada za założenie oraz usunięcie konta menedżera gamebaru. W przypadku złamania zasad przez klienta administrator ma możliwość zablokowania zgłoszonego przez menedżera użytkownika. Akcja zablokowania zgłoszonego klienta rozszerza akcję zbierania i analizowania danych o klientach.

Name:	Zablokuj zgłoszonego klienta.
Short description:	Administrator blokuje konto użytkownika zgłoszonego przez menedżera gamebaru.
Precondition:	Użytkownik posiada konto w systemie.
Postcondition:	Konto użytkownika zostało zablokowane. Nie może już z niego korzystać.
Error situations:	Został zgłoszony użytkownik, którego nie ma w systemie (np. użytkownik usunął konto).
System state in the event of an error:	Użytkownik nie jest blokowany.
Actors:	Administrator
Trigger:	Menedżer gamebaru zgłasza użytkownika do zablokowania.
Standard process:	(1) Klient gamebaru złamał zasady użytkowania (np. zniszczył konsolę). (2) Menedżer zgłasza użytkownika do administratora w celu zablokowania jego konta. (3) Administrator weryfikuje dane użytkownika i istnienie konta. (4) W systemie znajduje się konto użytkownika i dane zgłoszone przez managera są poprawne. (5) Administrator blokuje konto użytkownika.
Alternative process:	(1-3) Tak jak powyżej. (4') W systemie nie ma konta takiego użytkownika lub dane użytkownika się nie zgadzają. (5') Konto nie zostaje usunięte. (6') Administrator kontaktuje się z menedżerem w celu skorygowania danych użytkownika lub kończy proces blokowania (jeśli konto zostało usunięte przez użytkownika).

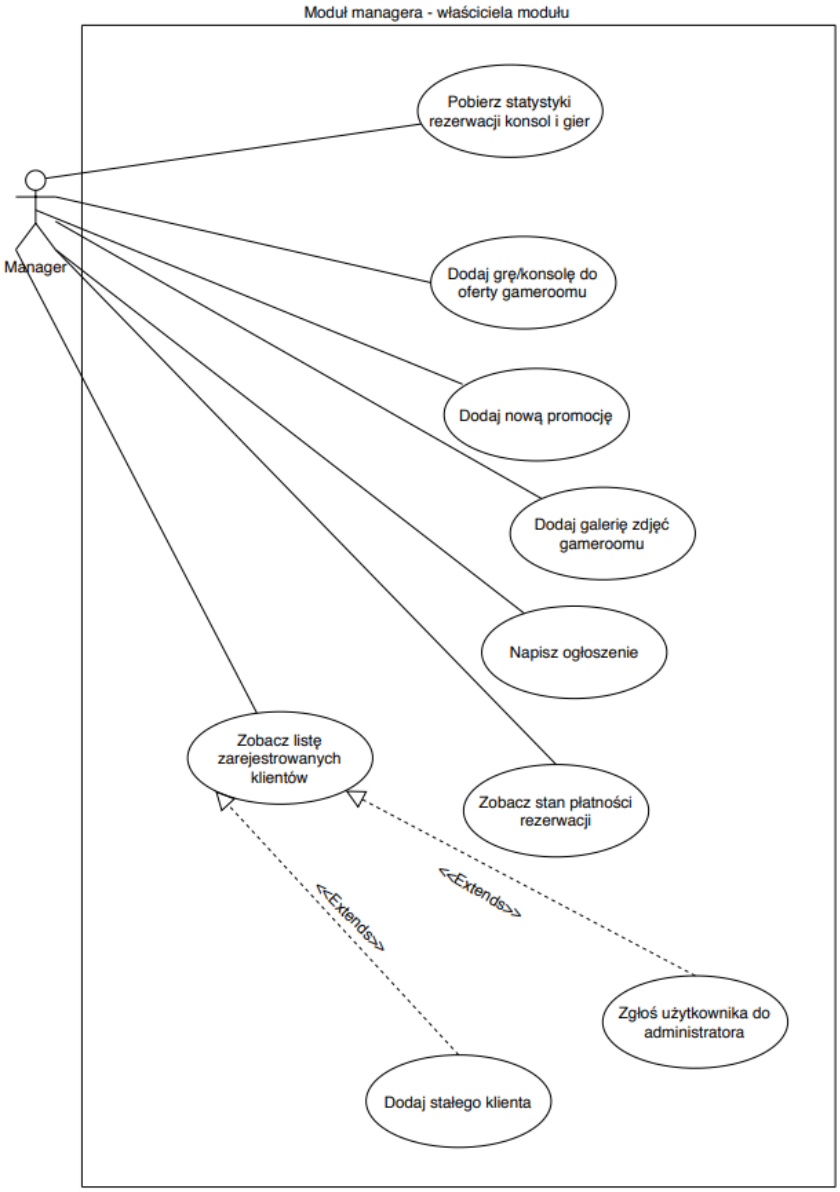
5.2 Moduł klienta



Moduł klienta zawiera akcje, które może wykonać klient. Diagram przedstawia dwa typy klientów: zwykłego i stałego. Stały klient jest uprawniony do korzystania z promocji dla stałych klientów. Oba typy klientów mogą wykonać rezerwację, opłacić rezerwację, przejrzeć swoje rezerwacje oraz je usuwać. Obaj klienci mogą również przeglądać promocje danego gamebaru, przeglądać oraz dodawać opinie na temat gamebarów oraz gier. Udostępniona została również możliwość zgłaszania opinii innych użytkowników, jeżeli zostaną uznane za niestosowne. Po wpisaniu lokalizacji gamebaru klient uzyskuje również dostęp do danych kontaktowych lokalu.

Name:	Zrób rezerwację.
Short description:	Użytkownik wykonuje rezerwację określonej konsoli w gamebarze.
Precondition:	Użytkownik posiada konto w systemie. Użytkownik wybrał interesującą go konsolę.
Postcondition:	Użytkownik zarezerwował konsolę na konkretny dzień i godzinę.
Error situations:	Wybrana przez użytkownika konsola jest już zarezerwowana w podanym przez użytkownika terminie.
System state in the event of an error:	Użytkownik zostaje powiadomiony o braku możliwość zarezerwowania konsoli na dany termin.
Actors:	Klient/Stały klient
Trigger:	Klient wybiera konsolę i termin po czym wciska przycisk "zarezerwuj".
Standard process:	(1) Klient wybiera interesujący go gamebar, konsolę oraz termin. (2) Użytkownik zostaje przekierowany do zakładki umożliwiającej dokonanie rezerwacji. (3) Użytkownik zostaje przekierowany do zakładki opłacenia rezerwacji. (4) Użytkownik ma zarezerwowaną i opłaconą konsolę w danym terminie. (5) Użytkownik może sprawdzić rezerwację w zakładce "rezerwacje".
Alternative process:	(1) Tak jak powyżej. (2') Konsola została zarezerwowana na ten termin przez innego użytkownika. (3') Użytkownik dostaje informację o braku możliwości rezerwacji konsoli w podanym terminie. (4') Użytkownik kończy rezerwację konsoli lub wybiera inny wolny termin. (5') Jeżeli użytkownik wybrał wolny termin, dalej tak jak w punktach (2-5) standardowego procesu.

5.3 Moduł menedżera



Moduł menedżera zawiera akcje jakie może wykonać menedżer gamebaru. Menedżer posiada dostęp do listy wszystkich klientów w systemie. Poprzez dostęp do tej listy może zgłosić klienta do administratora, w celu zablokowania konta klienta, który złamał przepisy oraz może dodać stałego klienta gamebaru, którym zarządza. Menedżer może również sprawdzić stan płatności rezerwacji, zamieszczać ogłoszenia i zdjęcia na profilu gamebaru oraz dodawać nowe promocje. Po zakupie nowego sprzętu lub gry do gamebaru, menedżer może również dodać informację do systemu o dostępie do nowych multimediów. W celach kontroli jakości świadczonych przez gamebar usług, menedżer może również pobrać statystyki o rezerwacjach danych konsol oraz gier.

Name:	Dodaj nową konsolę/grę do gamebaru.
Short description:	Menedżer dodaje nową konsolę/grę do oferty gamebaru.
Precondition:	Menedżer posiada konto w systemie.
Postcondition:	Nowa konsola/gra została dodana do oferty gamebaru.
Error situations:	Menedżer nie wypełnił wymaganych pól w formularzu dodawania konsoli/gry.
System state in the event of an error:	Menedżer zostaje powiadomiony o wymaganym uzupełnieniu pól formularza.
Actors:	Menedżer
Trigger:	Menedżer zakupił nową konsolę/grę do gamebaru i chce ją dodać do oferty.
Standard process:	(1) Menedżer wchodzi w zakładkę dodawania nowej konsoli/gry do oferty gamebaru. (2) Menedżer uzupełnia formularz dodawania nowej konsoli/gry. (3) Odbywa się walidacja formularza. (4) Menedżer poprawnie wypełnił formularz, który zostaje przesłany na serwer. Nowa konsola/gra zostaje dodana do bazy danych. (5) Konsola/gra została dodana do oferty gamebaru i jest dostępna dla użytkowników.
Alternative process:	(1-3) Tak jak powyżej. (3') Walidacja formularza zwraca błąd. Menedżer zostaje powiadomiony o potrzebie zmian w formularzu. (4-5) Tak jak w standardowej procedurze.

6 Struktura elementów w systemie

6.1 Użytkownik - klient

W module użytkownika - klienta gamebaru wyróżniamy dwie klasy reprezentujące typy użytkowników:

- User,
- SuperUser.

Klasa User jest reprezentacją zwykłego użytkownika, niebędącego stałym klientem żadnego gamebaru. Dziedzicząca po niej klasa SuperUser reprezentuje użytkownika, który uzyskał tytuł stałego klienta w co najmniej jednym gamebarze. W klasie User przechowujemy podstawowe informacje o użytkowniku, takie jak imię, nazwisko, email, nick, numer telefonu czy wiek. Każdy użytkownik ma również unikalne ID, pozwalające na jednoznaczne odróżnienie instancji klasy User. Metody klasy User pozwalają na:

- wykonanie rezerwacji (z wybranymi przez użytkownika parametrami),
- Dodanie oferty specjalnej do wykonywanej rezerwacji,
- napisania opinii (widocznej dla innych użytkowników) na temat gry lub gamebaru,
- opłacenia rezerwacji.

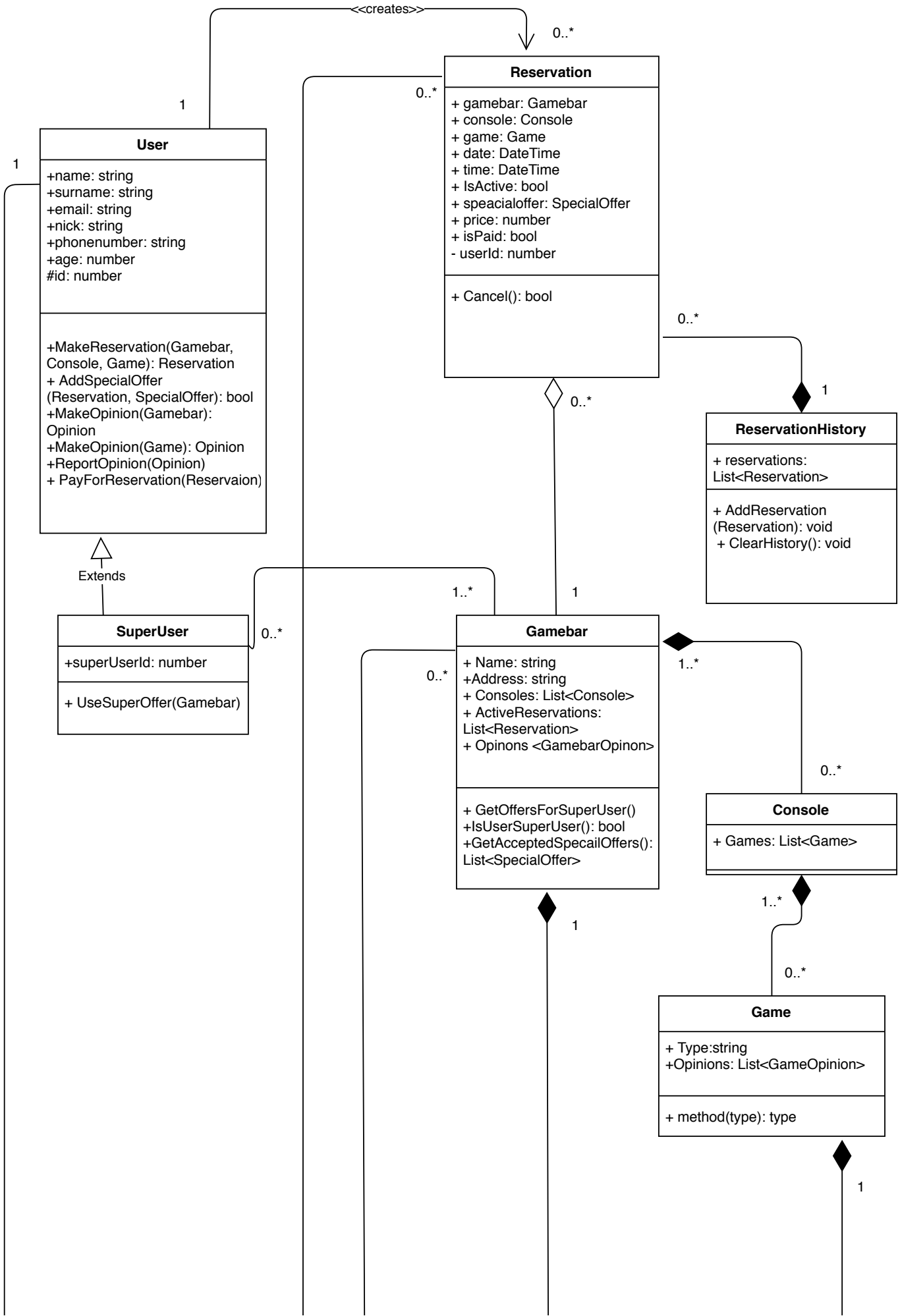
Ponadto, obiekty klasy SuperUser mają możliwość korzystania z ofert dla stałych klientów.

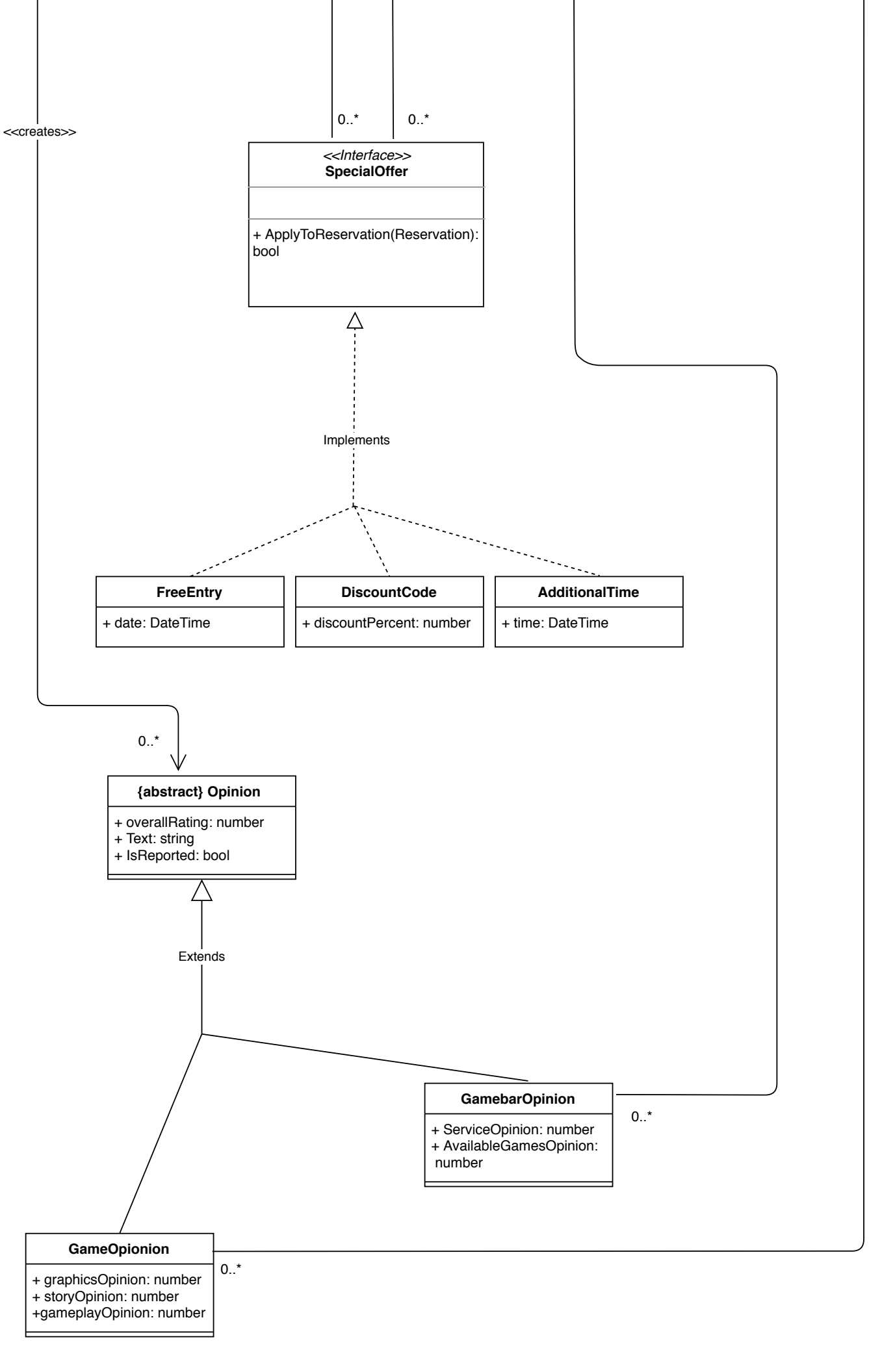
Obiekty klasy Reservation, tworzone przez instancję klasy User przechowują parametry rezerwacji (gra, konsola, data itp.). Są one agregowane przez klasę ReservationHistory, dzięki czemu użytkownik ma dostęp do historii swoich rezerwacji. Klasa Gamebar w module użytkownika zawiera informacje do wyświetlenia dla klienta gamebaru (nazwa, adres, lista dostępnych konsol itp.). Agreguje obiekty klasy Console, które z kolei agregują obiekty klasy Game (Gamebar ma konsole, z których każda posiada zestaw dostępnych gier).

Wyróżniamy 3 typy ofert specjalnych (oferty są przypisane do konkretnego gamebaru - jego menedżer może zaakceptować globalnie dostępne promocje, co przekazywane jest potem do modułu użytkownika): FreeEntry, DicountCode, AdditionalTime (klasy implementujące interfejs SpecialOffer). Nazwa każdej z klas odzwierciedla sposób działania oferty.

Możliwe jest też dodawanie opinii. Klasy GameOpinion oraz GamebarOpinion dziedziczą po klasie Opinion i mogą być tworzone przez obiekty klasy User. W opinii o grze możliwe do oceny są parametry takie jak grafika czy sposób rozgrywki. Opinia o gamebarze zawiera np. opinię obsługi czy dostępnych gier.

Szczegółowe informacje zawarte są na diagramie poniżej.


<<creates>>
0..*
1
Reservation
0..*
+ gamebar: Gamebar
+ console: Console
+ game: Game
+ date: DateTime
+ time: DateTime
+ IsActive: bool
+ speacialoffer: SpecialOffer
+ price: number
+ isPaid: bool
- userId: number
+ Cancel(): bool
User
1
+name: string
+surname: string
+email: string
+nick: string
+phonenumber: string
+age: number
#id: number
+MakeReservation(Gamebar, Console, Game): Reservation
+ AddSpecialOffer (Reservation, SpecialOffer): bool
+MakeOpinion(Gamebar): Opinion
+MakeOpinion(Game): Opinion
+ReportOpinion(Opinion)
+ PayForReservation(Reservaion)
0..*
1
ReservationHistory
+ reservations: List<Reservation>
+ AddReservation (Reservation): void
+ ClearHistory(): void
0..*
1
Extends
SuperUser
+superUserId: number
+ UseSuperOffer(Gamebar)
0..*
1..*
1
Gamebar
+ Name: string
+Address: string
+ Consoles: List<Console>
+ ActiveReservations: List<Reservation>
+ Opinons <GamebarOpinon>
+ GetOffersForSuperUser()
+IsUserSuperUser(): bool
+GetAcceptedSpecailOffers(): List<SpecialOffer>
0..*
1..*
0..*
Console
+ Games: List<Game>
1..*
0..*
1
Game
+ Type:string
+Opinions: List<GameOpinion>
+ method(type): type
1


<<creates>>
0..*
0..*
<<Interface>>
SpecialOffer
+ ApplyToReservation(Reservation): bool
Implements
FreeEntry
+ date: DateTime
DiscountCode
+ discountPercent: number
AdditionalTime
+ time: DateTime
0..*
{abstract} Opinion
+ overallRating: number
+ Text: string
+ IsReported: bool
Extends
GamebarOpinion
+ ServiceOpinion: number
+ AvailableGamesOpinion: number
0..*
GameOpionion
+ graphicsOpinion: number
+ storyOpinion: number
+gameplayOpinion: number
0..*

6.2 Administrator systemu

Moduł administratora systemu to najprostszy architektonicznie moduł w aplikacji. Głównymi zadaniami administratora systemu są:

- zarządzanie użytkownikami,
- zarządzanie gamebarami,
- wykonywanie analiz dotyczących działalności gamebarów i aktywności użytkowników.

W tym module istotne są role poszczególnych członków systemu. Abstrakcyjna klasa Person zawiera podstawowe informacje gromadzone przez system na temat użytkowników, niezależnie od pełnionej roli (imię, nazwisko, nr telefonu, email, data urodzenia) oraz pozwala na zarządzanie hasłami i loginami. Po klasie Person dziedziczą:

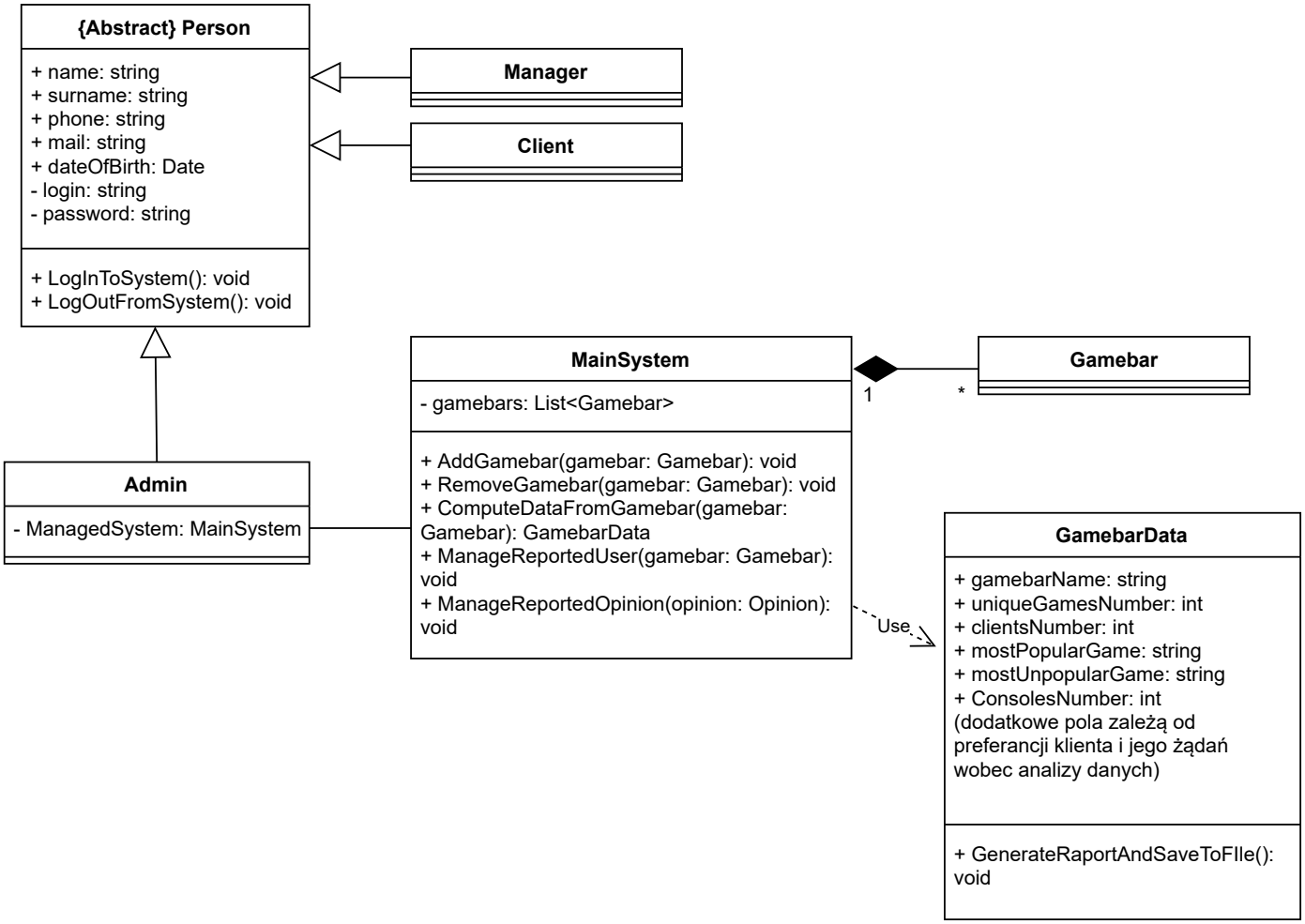
- klasa Manager - reprezentuje menedżera gamebaru,
- klasa Client - reprezentuje użytkownika - klienta gamebaru,
- klasa Admin - reprezentuje administratora zarządzającego systemem.

Każdy Admin zarządza dokładnie jednym systemem, przy czym system może być zarządzany przez wielu Adminów (co najmniej jednego). Klasa MainSystem reprezentuje system zarządzania gamebarami, agregując instancje klasy Gamebar. Metody klasy MainSystem umożliwiają:

- dodawanie gamebaru do systemu,
- usuwanie gamebaru z systemu,
- przetwarzanie danych z gamebaru - klasy GamebarData,
- zarządzanie zgłoszonymi użytkownikami - usuwanie, blokowanie itp.

System zbiera informacje o gamebarach - klasa GamebarData. Poza podstawowymi danymi (najbardziej popularne gry, najmniej popularne gry wśród danych klientów itp.), możliwe jest dostosowanie zbieranych danych do wymagań klientów systemu. Istnieje możliwość wygenerowania raportu z posiadanych informacji.

Szczegółowe informacje zawarte są na diagramie poniżej.


{Abstract} Person
+ name: string
+ surname: string
+ phone: string
+ mail: string
+ dateOfBirth: Date
- login: string
- password: string
+ LogInToSystem(): void
+ LogOutFromSystem(): void
Manager
Client
Admin
- ManagedSystem: MainSystem
MainSystem
- gamebars: List<Gamebar>
+ AddGamebar(gamebar: Gamebar): void
+ RemoveGamebar(gamebar: Gamebar): void
+ ComputeDataFromGamebar(gamebar: Gamebar): GamebarData
+ ManageReportedUser(gamebar: Gamebar): void
+ ManageReportedOpinion(opinion: Opinion): void
1
*
Gamebar
Use
GamebarData
+ gamebarName: string
+ uniqueGamesNumber: int
+ clientsNumber: int
+ mostPopularGame: string
+ mostUnpopularGame: string
+ ConsolesNumber: int
(dodatkowe pola zależą od preferancji klienta i jego żądań wobec analizy danych)
+ GenerateRaportAndSaveToFIle(): void

6.3 Menedżer gamebaru

Najważniejsza klasą w module menedżera gamebaru jest klasa Gamebar, wokół której skupia się cała architektura modułu. Przechowuje ona podstawowe informacje o lokalu (nazwa, adres, email, numer telefonu itp.). Jest powiązana z klasą Manager (dla jednego gamebaru wyróżniamy dokładnie jednego menedżera). W klasie Gamebar przechowujemy zdjęcia, które mogą być udostępniane użytkownikom chętnym do wykonania rezerwacji.

Metody klasy Gamebar umożliwiają:

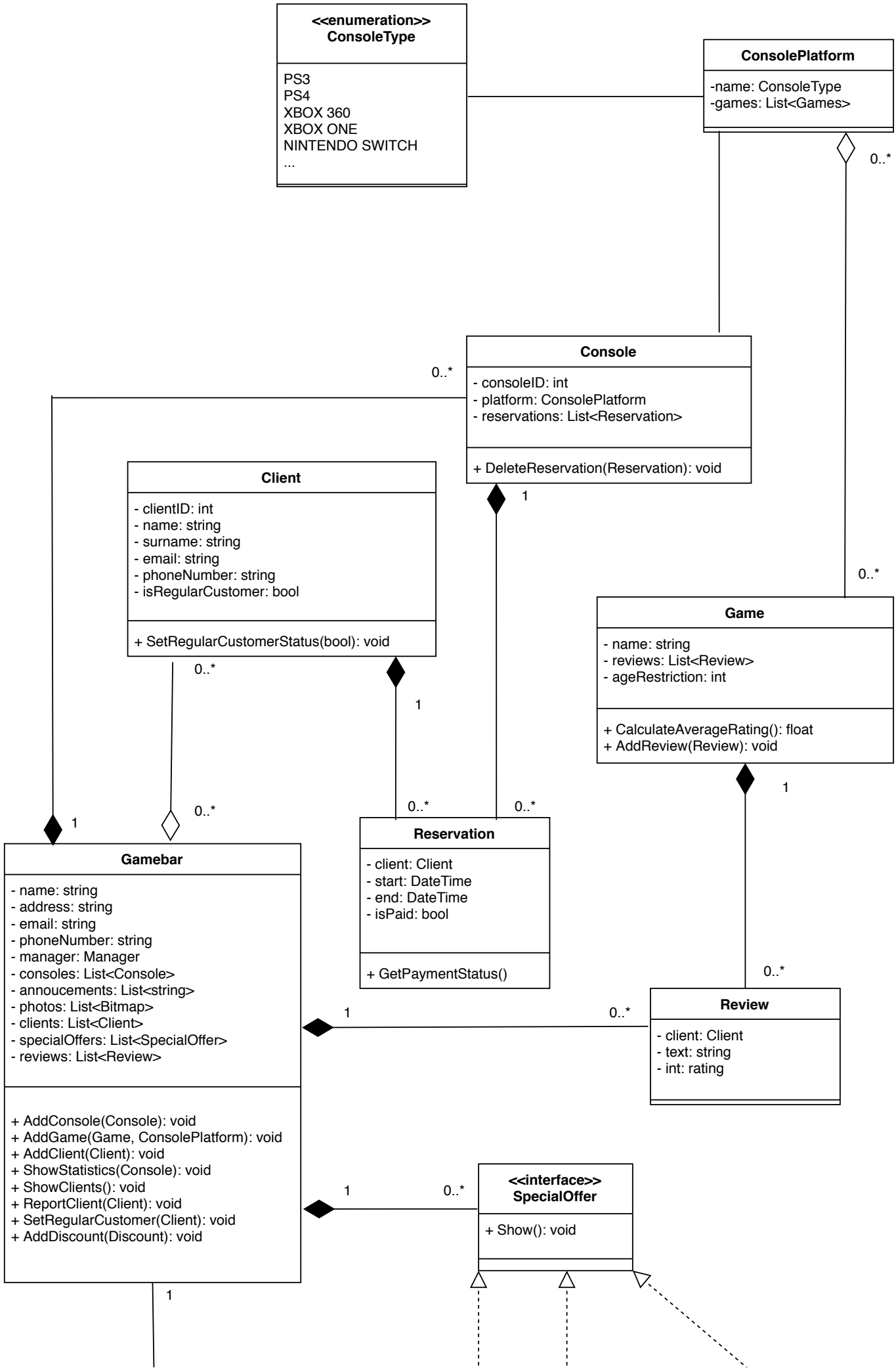
- dodanie konsoli, gry, klienta do gamebaru,
- analizę statystyk,
- wyświetlanie klientów lokalu,
- zgłaszanie klientów do administratora,
- dodawanie stałych klientów,
- dodawanie ofert specjalnych.

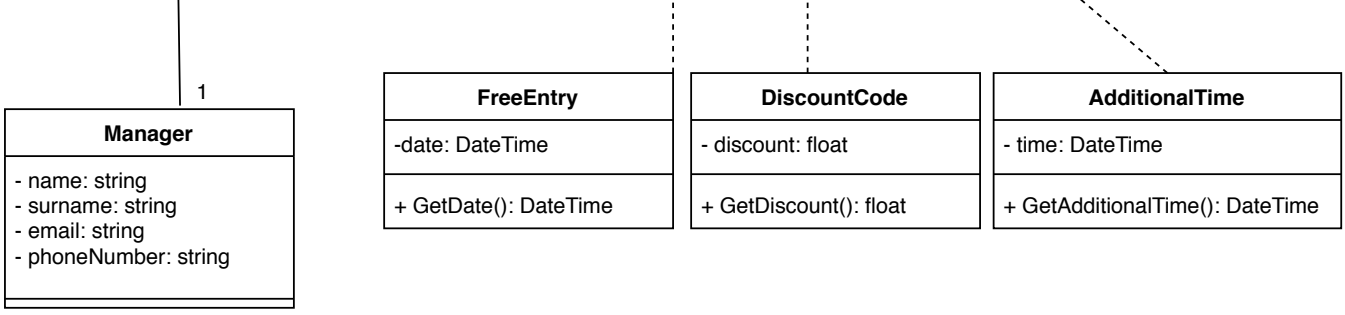
Ponadto, klasa Gamebar agreguje konsole, które są w danej lokacji (klasa Console) oraz klientów (klasa Client), którzy dokonali rezerwacji. Obiekty klasy Reservation agregowane są zarówno przez obiekty klasy Client, jak i klasy Console. Pozwala to na kojarzenie rezerwacji z konkretnymi użytkownikami oraz przechowywanie stanu dostępności konsoli. Klasa Console powiązana jest z klasą ConsolePlatform, która agreguje gry (klasa Game) dostępne na danej konsoli (Enum ConsoleType określa typ konsoli).

Klasa Review zawiera informacje o opinii klienta i dotyczy (jest agregowana przez) zarówno Gamebaru, jak i konkretnej gry.

Gamebar przechowuje również informacje o specjalnych ofertach, które są w nim dostępne. Interfejs SpecialOffer pozwala na wyświetlenie konkretnej oferty oraz udostępnienie jej użytkownikowi. Dostępne są 3 rodzaje ofert specjalnych (odpowiadające ofertom w module użytkownika): FreeEntry, DiscountCode, AdditionalTime.

Szczegółowe informacje zawarte są na diagramie poniżej.

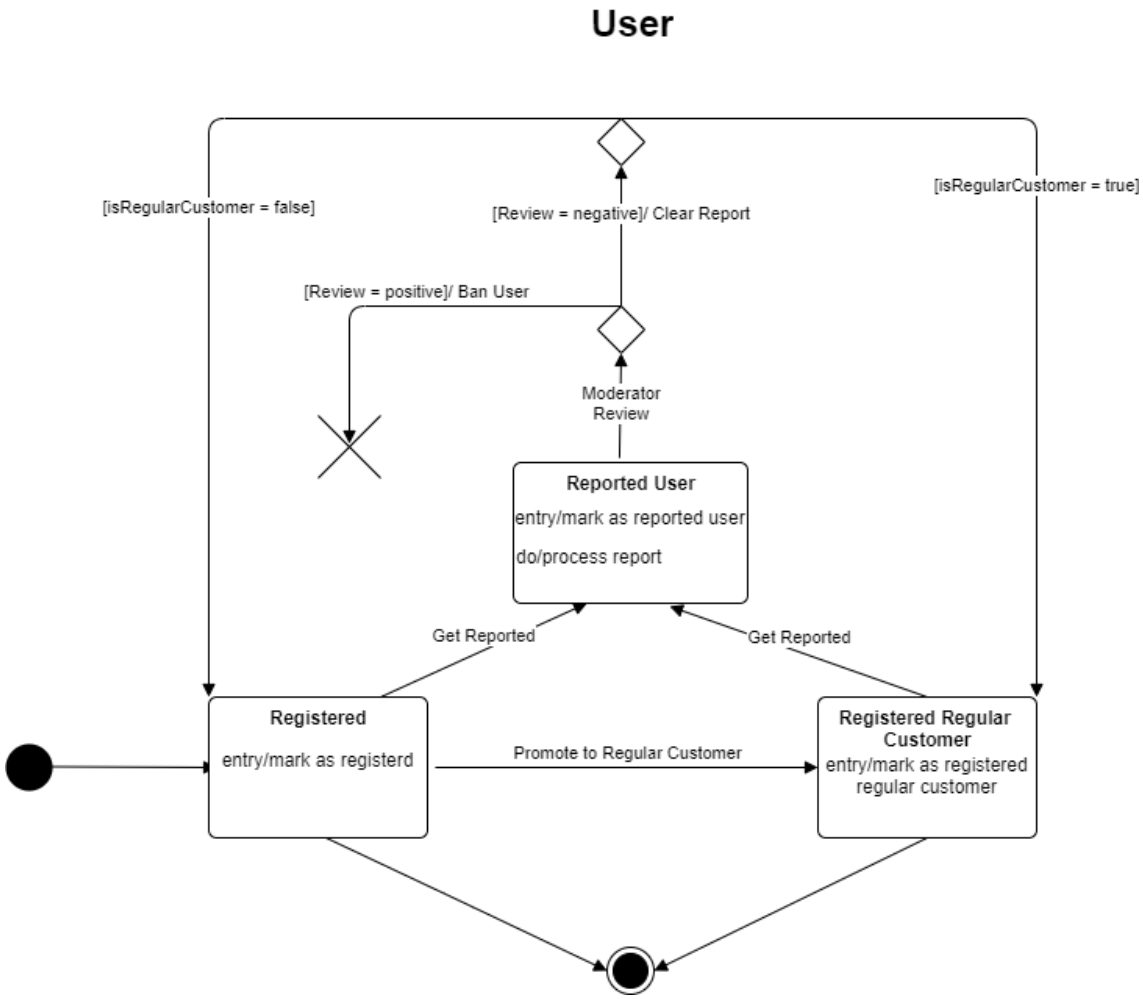

<<enumeration>>
ConsoleType
PS3
PS4
XBOX 360
XBOX ONE
NINTENDO SWITCH
...
ConsolePlatform
-name: ConsoleType
-games: List<Games>
0..*
Console
- consoleID: int
- platform: ConsolePlatform
- reservations: List<Reservation>
+ DeleteReservation(Reservation): void
0..*
1
Client
- clientID: int
- name: string
- surname: string
- email: string
- phoneNumber: string
- isRegularCustomer: bool
+ SetRegularCustomerStatus(bool): void
0..*
1
Game
- name: string
- reviews: List<Review>
- ageRestriction: int
+ CalculateAverageRating(): float
+ AddReview(Review): void
0..*
1
Gamebar
- name: string
- address: string
- email: string
- phoneNumber: string
- manager: Manager
- consoles: List<Console>
- annoucements: List<string>
- photos: List<Bitmap>
- clients: List<Client>
- specialOffers: List<SpecialOffer>
- reviews: List<Review>
+ AddConsole(Console): void
+ AddGame(Game, ConsolePlatform): void
+ AddClient(Client): void
+ ShowStatistics(Console): void
+ ShowClients(): void
+ ReportClient(Client): void
+ SetRegularCustomer(Client): void
+ AddDiscount(Discount): void
1
0..*
Reservation
- client: Client
- start: DateTime
- end: DateTime
- isPaid: bool
+ GetPaymentStatus()
0..*
1
Review
- client: Client
- text: string
- int: rating
0..*
1
<<interface>>
SpecialOffer
+ Show(): void
0..*
1
1


1
Manager
- name: string
- surname: string
- email: string
- phoneNumber: string
FreeEntry
-date: DateTime
+ GetDate(): DateTime
DiscountCode
- discount: float
+ GetDiscount(): float
AdditionalTime
- time: DateTime
+ GetAdditionalTime(): DateTime

7 Stany systemu i obiektów

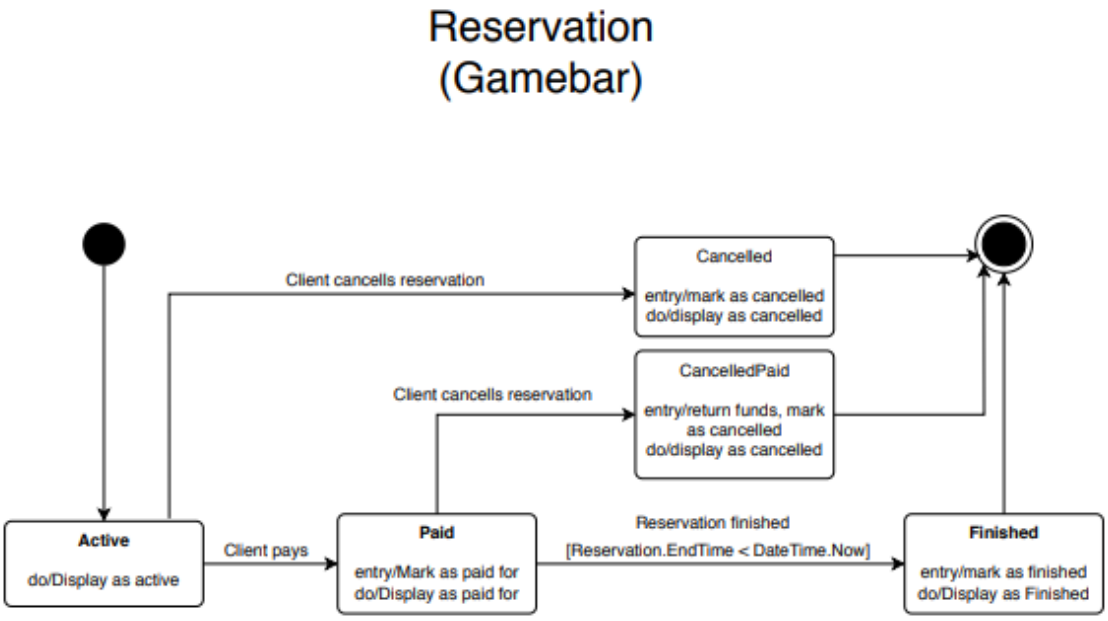
7.1 User

Użytkownik po zarejestrowaniu przechodzi w domyślny stan Registered. Jeśli regularnie uczęszcza do gamebaru to może uzyskać status stałego klienta. W takiej sytuacji obiekt przechodzi w stan Registered Regular Customer. Jeśli użytkownik zostanie zgłoszony z powodu swojej opinii lub zachowania w gamebarze, przechodzi w tymczasowy stan Reported, a jego sytuacja zostaje rozpatrzona przez moderatorów. Jeśli zgłoszenie okazuje się nieuzasadnione lub niewystarczająco poważne, użytkownik wraca do stanu sprzed zgłoszenia, w przeciwnym wypadku zostaje zablokowany.

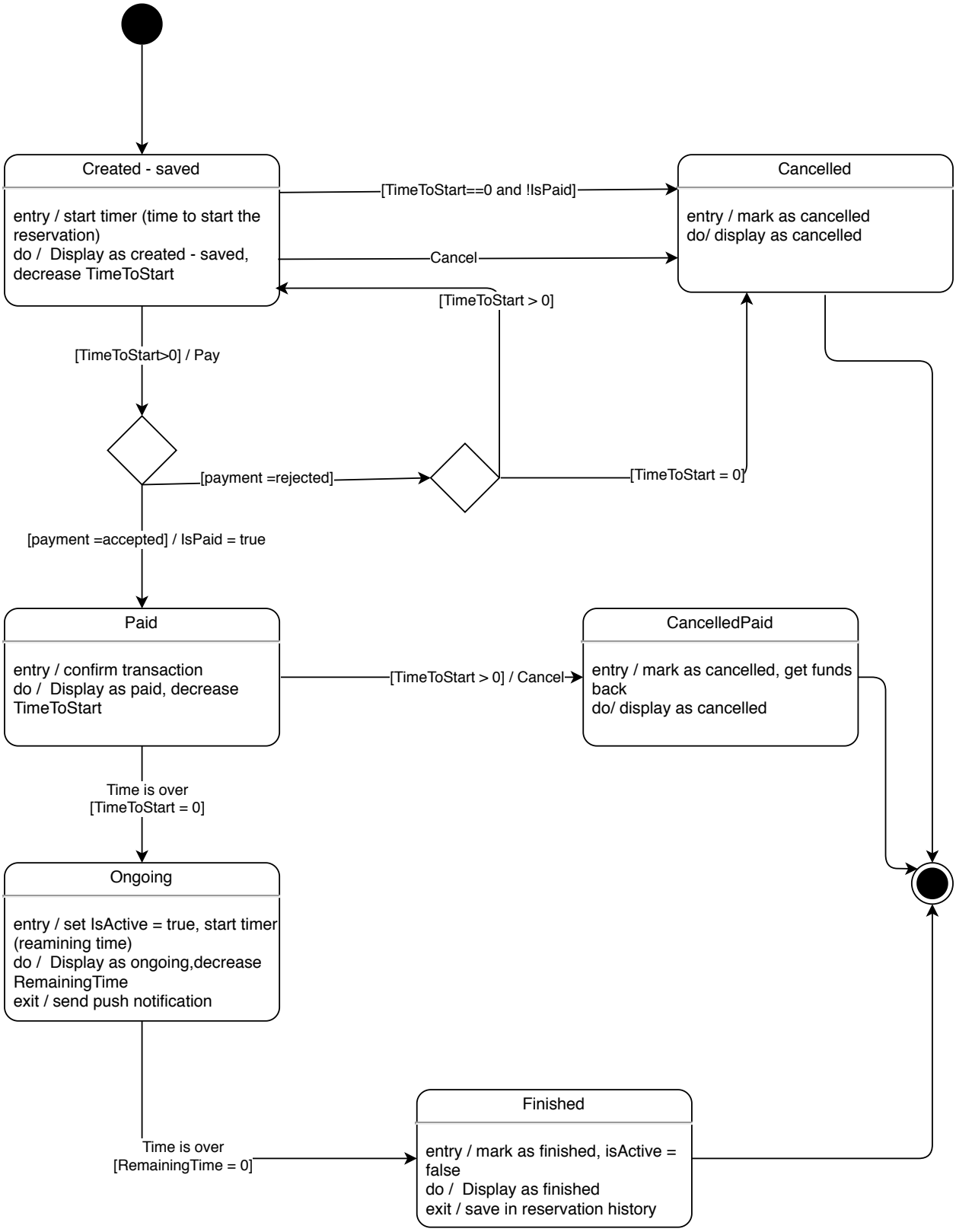


7.2 Reservation

Rezerwacja jest kluczowym obiektem w naszym systemie i ma ona odpowiadające sobie obiekty w modelu Klienta i Gamebaru. Na początku zostaje ona stworzona i przechodzi w stan aktywny. Stąd użytkownik może ją anulować i przechodzi w stan Cancelled, lub opłacić w odpowiednim terminie. Opłacona rezerwacja przechodzi w stan Paid, może ona nadal zostać anulowana i przejść w stan CancelledPaid lub dojść do skutku. Będąc w wykonaniu jest w stanie Ongoing i po zakończeniu sesji przez klienta przechodzi w stan Finished.

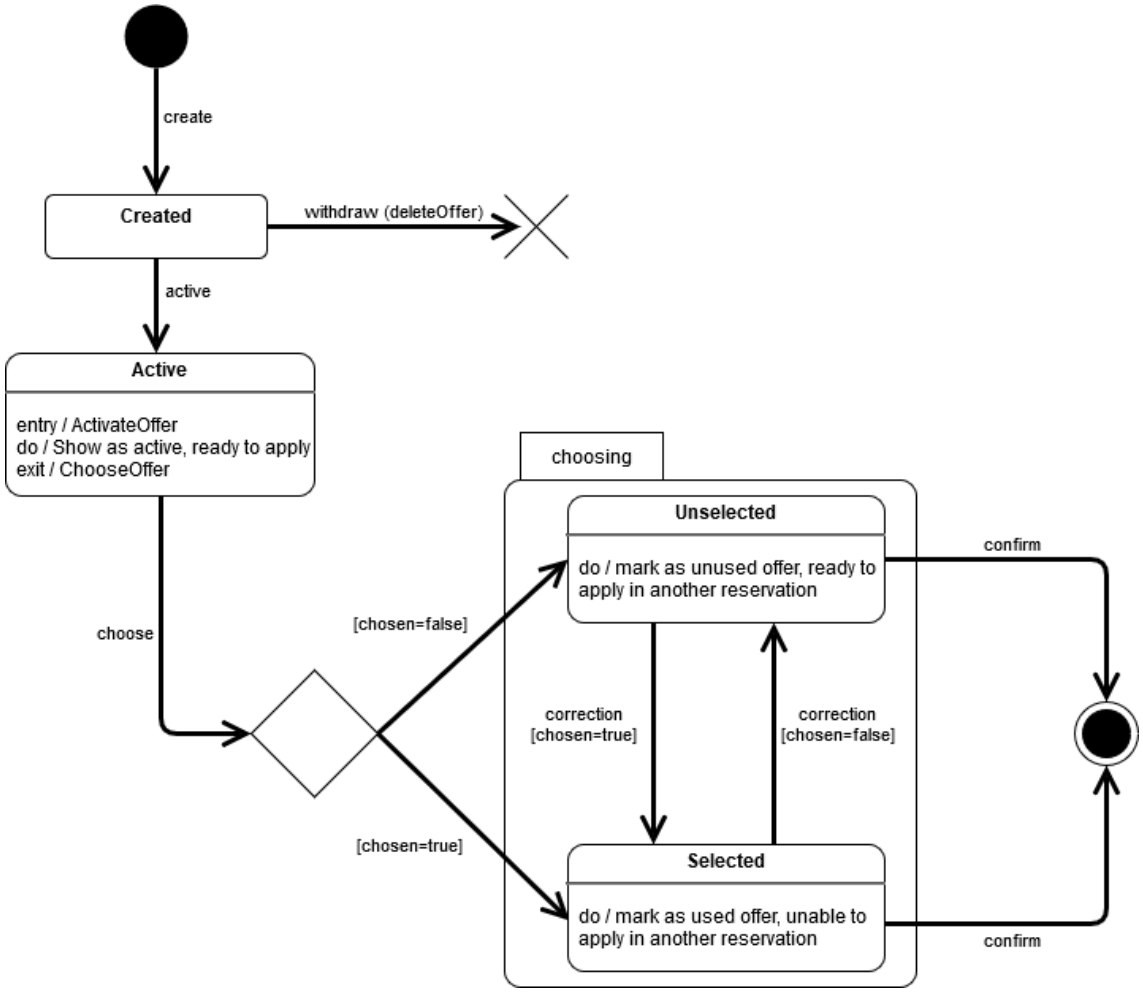


Reservation (Cilent Module)



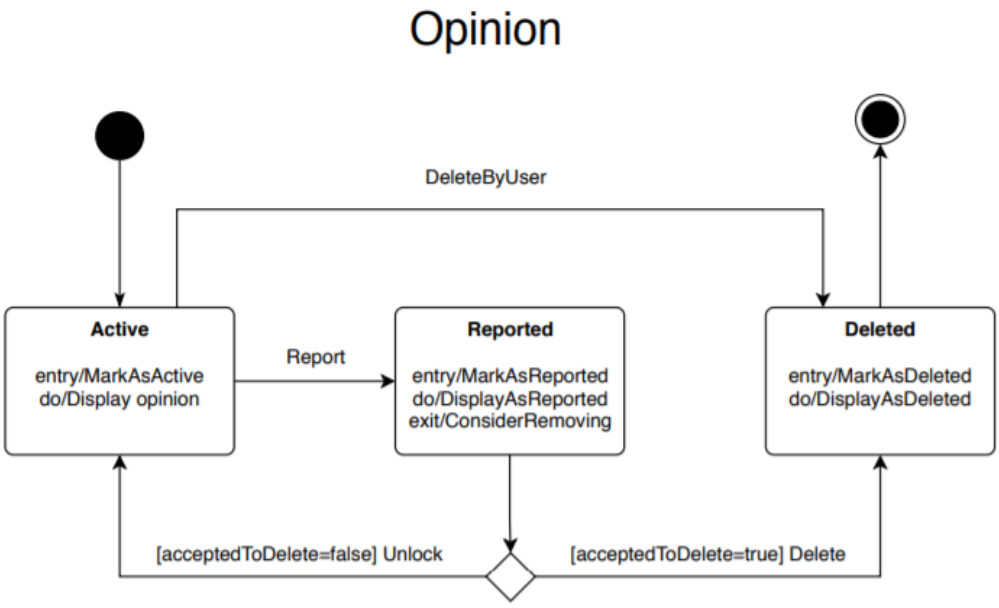
7.3 Offer

Oferta zostaje stworzona i wchodzi w stan Active. Następnie może zostać wybrana i wchodzi w stan choosing, w którym można ją zaznaczyć lub odznaczyć, po decyzji i akceptacji oferta wchodzi w życie.



7.4 Opinion

Opinia jest dosyć prostym obiektem. Po stworzeniu jest w stanie aktywnym i póki nie zostanie zgłoszona lub usunięta to w tym stanie pozostaje. Po zgłoszeniu przechodzi w stan Reported i jeśli zgłoszenie było uzasadnione to przechodzi w stan Deleted i zostaje usunięta. W przeciwnym wypadku wraca do stanu Active.

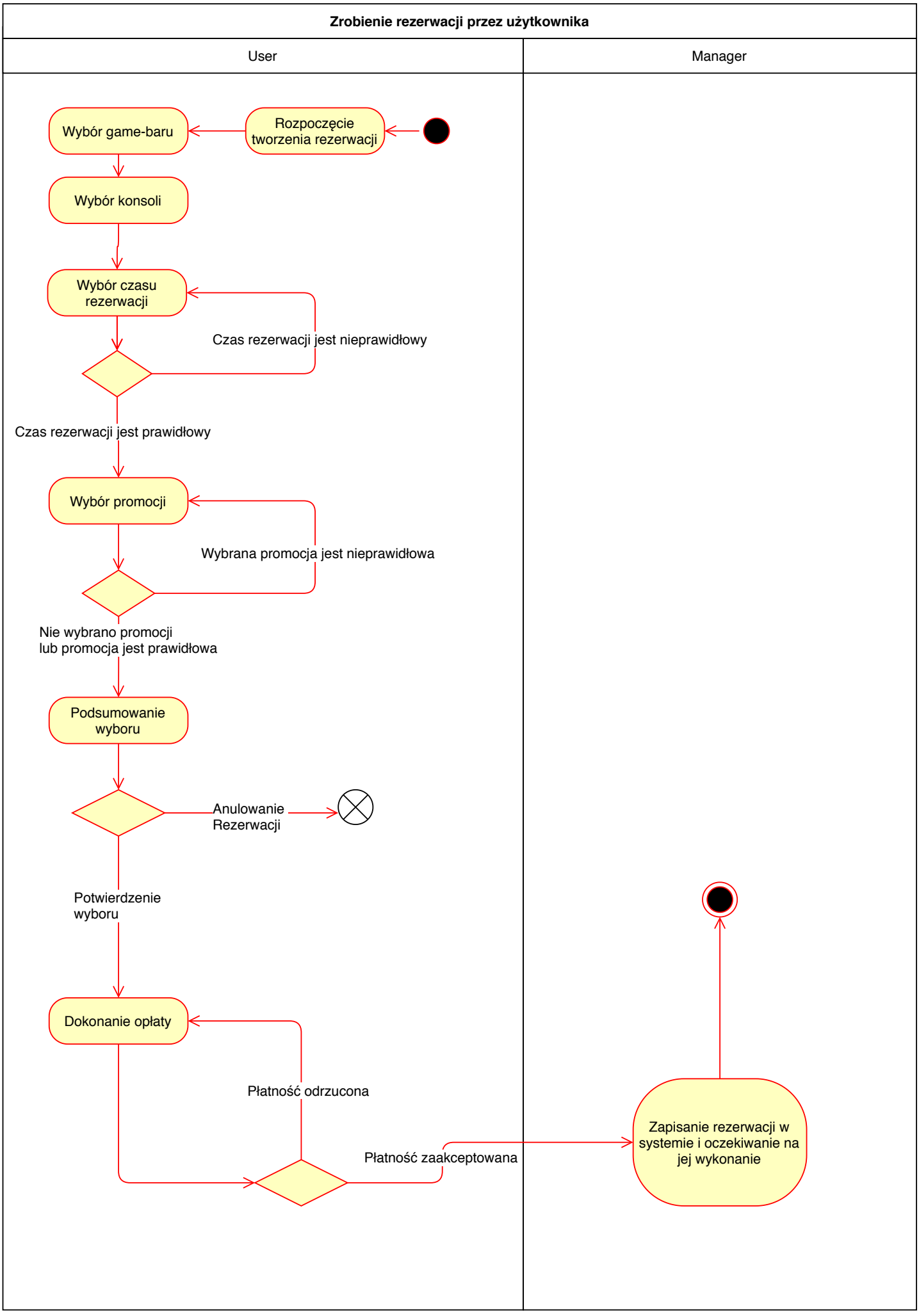


8 Przepływ kontroli i danych

Diagramy aktywności służą do formalnego modelowania aspektów przetwarzania w systemie. Określają przepływ kontroli oraz danych pomiędzy komponentami. Poniżej przedstawione są opisy kluczowych funkcjonalności w systemie, które obejmują standardowe ścieżki wykonania oraz ich alternatywy. Poniżej każdego opisu znajduje się odpowiadający mu diagram.

8.1 Dokonanie rezerwacji przez użytkownika

W operacji dokonania rezerwacji przez użytkownika biorą udział dwa moduły: moduł użytkownika (User) oraz moduł danego gamebaru, określanego również jako moduł menedżera (Manager). Oczywistym jest, że akcja rozpoczyna się w module User poprzez rozpoczęcie procesu tworzenia rezerwacji przez użytkownika. Użytkownik dokonuje wyboru interesującego go gamebaru oraz konsoli. Następnie dokonywany jest wybór czasu rezerwacji. Jest on powtarzany do momentu zaznaczenia/wpisania przez użytkownika poprawnego czasu, w którym chcę odwiedzić dany gamebar. (system sprawdza, czy o danej godzinie gamebar jest otwarty oraz czy są dostępne wolne miejsca). Kolejnym krokiem jest opcjonalne zastosowanie kodu promocyjnego. Użytkownik może pominąć ten krok, jeżeli nie chce zastosować kodu promocyjnego. Natomiast w przypadku podjęcie próby użycia kodu, walidacja będzie powtarzana do momentu wpisania *poprawnego* kodu (system sprawdza, czy dany kod pochodzi z danego gamebaru, jest wciąż aktywny oraz czy użytkownik ma do niego uprawnienia). Jest to proces analogiczny do procesu sprawdzania poprawności czasu rezerwacji. Jeśli wszystkie operacje zakończył się sukcesem, użytkownik może potwierdzić swój wybór i przejść do dokonania płatności. Oczywiście istnieje również opcja przerwania procesu tworzenia rezerwacji. Płatność jest akceptowana, jeżeli zewnętrzny proces płatności przebiegł poprawnie lub użytkownik podjął decyzję, że opłaci rezerwację w terminie późniejszym. Następnie kontrola przechodzi do modułu menedżera, gdzie odpowiednie informacje są uaktualnianie i zapisywane w bazie danych.


Zrobienie rezerwacji przez użytkownika
User
Manager
Rozpoczęcie tworzenia rezerwacji
Wybór game-baru
Wybór konsoli
Wybór czasu rezerwacji
Czas rezerwacji jest nieprawidłowy
Czas rezerwacji jest prawidłowy
Wybór promocji
Wybrana promocja jest nieprawidłowa
Nie wybrano promocji lub promocja jest prawidłowa
Podsumowanie wyboru
Anulowanie Rezerwacji
Potwierdzenie wyboru
Dokonanie opłaty
Płatność odrzucona
Płatność zaakceptowana
Zapisanie rezerwacji w systemie i oczekiwanie na jej wykonanie

8.2 Dodanie promocji przez Menedżera

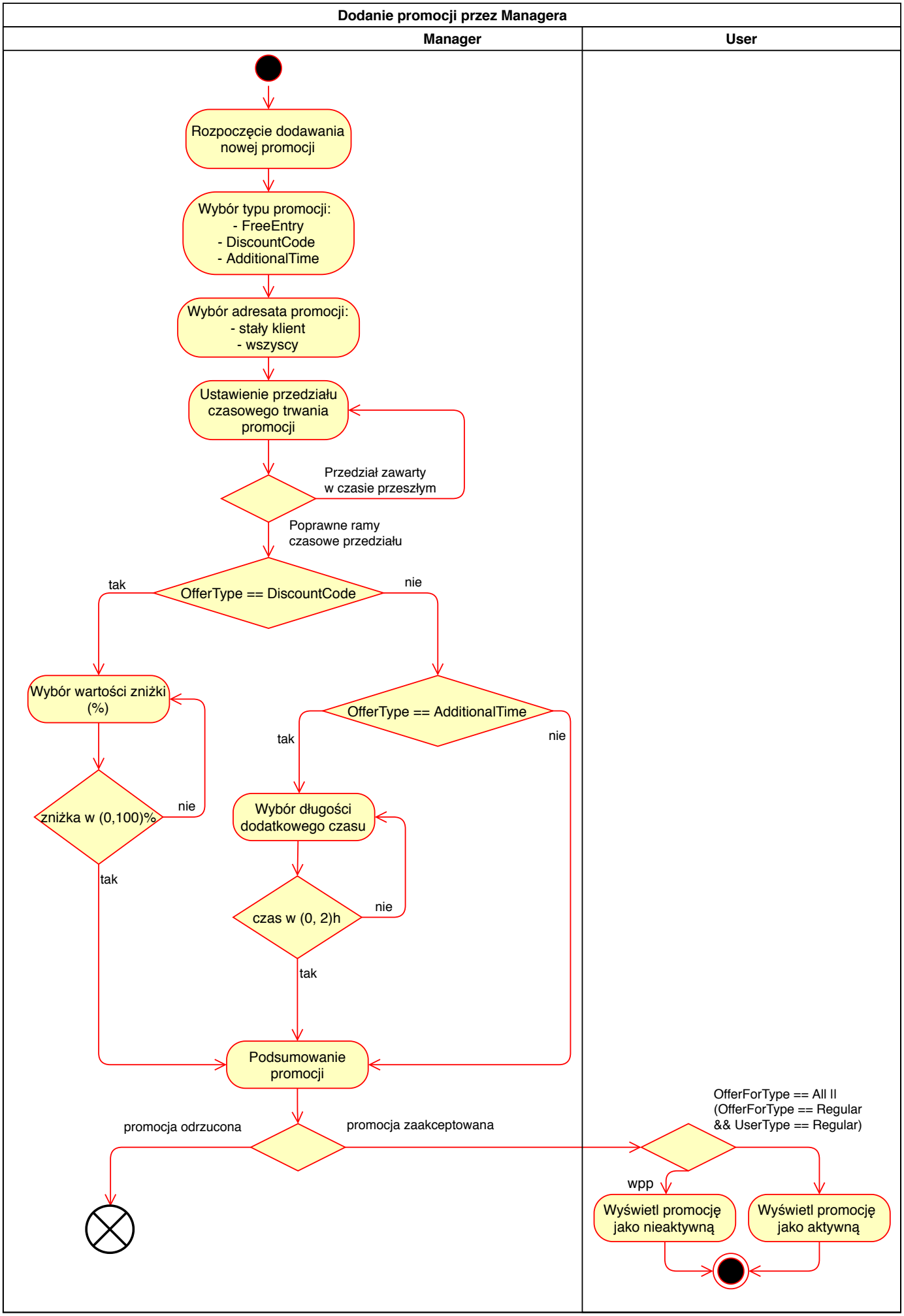
Operacja dodania promocji przez menedżera odbywa się pomiędzy dwoma modułami: modułem menedżera (Manager) oraz modułem użytkownika (User). Punktem startowym operacji jest przejście do trybu tworzenia i dodawania nowej promocji przez menedżera danego gamebaru. W początkowej fazie procesu menedżer dokonuje wyboru typu promocji. Przewiduje się następujące typy promocji w systemie:

- FreeEntry (darmowe wejście)
- DiscountCode (kod rabatowy)
- AdditionalTime (dodatkowy czas, wydłużenie rezerwacji)

Dalszym krokiem jest wybór adresata promocji. Adresatem może być każdy użytkownik lub tylko użytkownicy o statusie stałego klienta. Menedżer ustala również przedział czasu, w którym dana promocja będzie aktywna. Kontrola w systemie sprawdza poprawność wyboru menedżera (być może wybrany przedział jest częściowo zawarty w czasie przeszłym, taki wybór jest nieakceptowany) i pozostaje w tym stanie do momentu podanie poprawnego przedziału czasowego. W zależności od wybranego typu promocji system wymaga podania dodatkowych informacji.

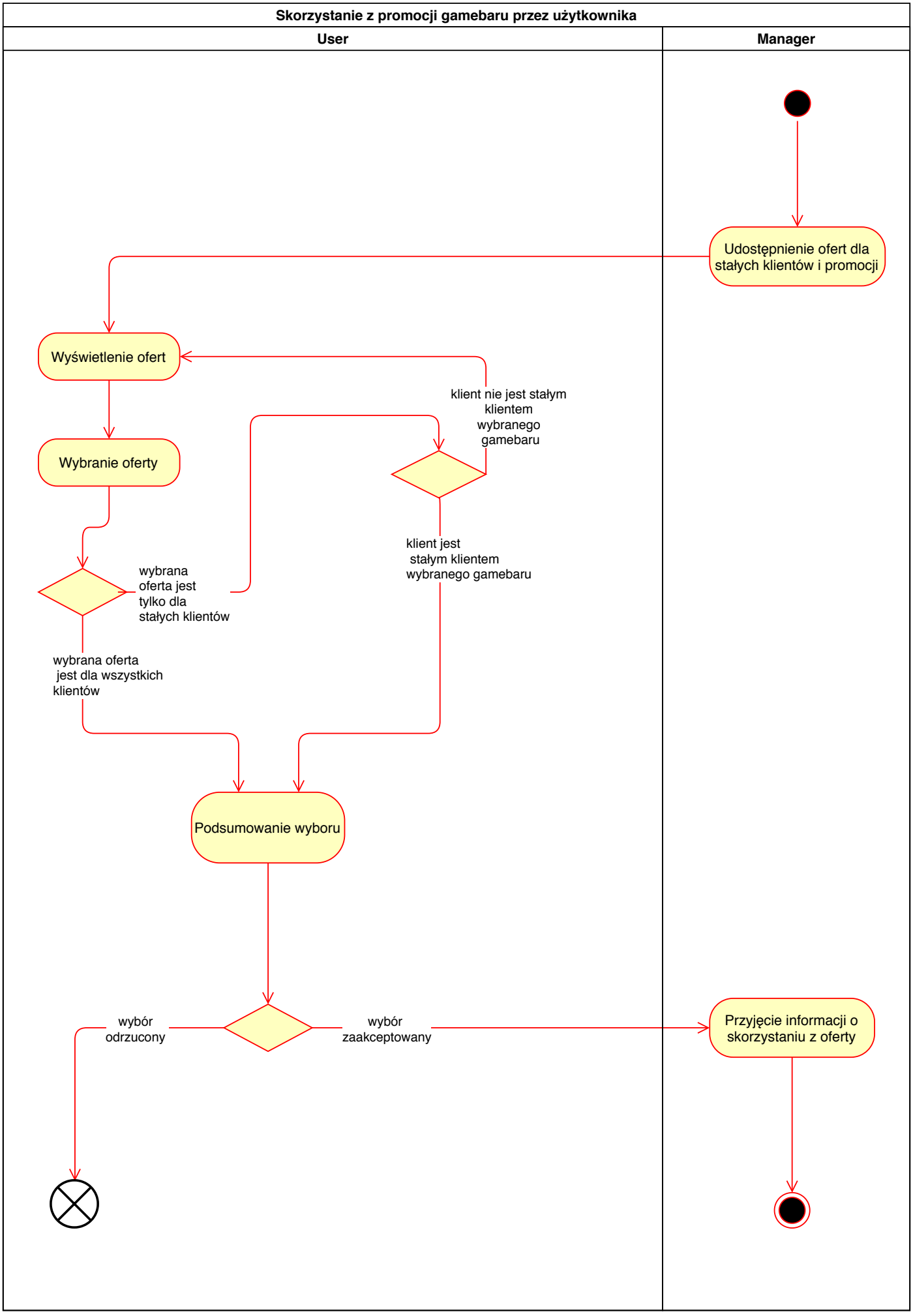
- Jeżeli wybrano kod rabatowy, system wymusza podanie przez menedżera odpowiedniej wartości zniżki wyrażonej w procentach (%). Wybór podlega walidacji do momentu uzyskania poprawnej wartości z przedziału (0, 100)%.
- Jeżeli wybrano dodatkowy czas wydłużający podstawowy czas rezerwacji, system wymusza podanie odpowiedniej wartości wyrażonej w godzinach. Zakłada się, że wartość powinna być zawarta w przedziale (0, 2)h.
- Jeżeli wybrano dodatkowe wejście dodatkowe informacje nie są wymagane.

Po sukcesywnym przebyciu procesu uzupełniania dodatkowych informacji, menedżer decyduje o akceptacji nowej promocji. W przypadku jej zatwierdzenia, kontrola systemu przechodzi do modułu użytkownika (User) i w zależności od statusu użytkownika oraz adresata promocji wyświetla nową promocję jako aktywną lub nieaktywną dla danego użytkownika. Menedżer może również podjąć decyzję o anulowaniu promocji. W takim przypadku operacja kończy się i żadne dane nie są zapisywane w module użytkownika.


Dodanie promocji przez Managera
Manager
User
Rozpoczęcie dodawania nowej promocji
Wybór typu promocji:
- FreeEntry
- DiscountCode
- AdditionalTime
Wybór adresata promocji:
- stały klient
- wszyscy
Ustawienie przedziału czasowego trwania promocji
Przedział zawarty w czasie przeszłym
Poprawne ramy czasowe przedziału
OfferType == DiscountCode
tak
nie
Wybór wartości zniżki (%)
zniżka w (0,100)%
nie
tak
OfferType == AdditionalTime
tak
nie
Wybór długości dodatkowego czasu
czas w (0, 2)h
nie
tak
Podsumowanie promocji
promocja odrzucona
promocja zaakceptowana
OfferForType == All II
(OfferForType == Regular
&& UserType == Regular)
wpp
Wyświetl promocję jako nieaktywną
Wyświetl promocję jako aktywną

8.3 Skorzystanie z promocji przez użytkownika

W operacji skorzystania z promocji przez użytkowania biorą udział dwa moduły: moduł menedżera (Manager) oraz moduł użytkownika (User). Zakładamy, że początek opcji ma swoje miejsce w module menedżera poprzez udostępnienie oferta dla klientów. Następnie kontrola systemu przechodzi do modułu użytkownika. Udostępnione oferty są wyświetlane w przeznaczonym do tego panelu. W przypadku wybrania konkretnej oferty przez użytkownika system sprawdza status użytkownika oraz typ adresata oferty. Na tej podstawie zapada decyzja czy dany użytkownik jest uprawniony do skorzystania z konkretnej promocji. W przypadku pozytywnego wybrania oferty, użytkownik przechodzi do podsumowania wyboru. Możliwe jest odrzucenie skorzystania z wybranej poprzednio promocji lub akceptacja dokonanego wyboru. W przypadku zatwierdzenia wyboru, kontrola systemu przechodzi ponownie do moduły menedżera i informuje o fakcie, iż dany użytkownik skorzystał z udostępnionej oferty.


Skorzystanie z promocji gamebaru przez użytkownika
User
Manager
Udostępnienie ofert dla stałych klientów i promocji
Wyświetlenie ofert
Wybranie oferty
wybrana oferta jest tylko dla stałych klientów
wybrana oferta jest dla wszystkich klientów
klient nie jest stałym klientem wybranego gamebaru
klient jest stałym klientem wybranego gamebaru
Podsumowanie wyboru
wybór odrzucony
wybór zaakceptowany
Przyjęcie informacji o skorzystaniu z oferty

8.4 Dodanie/zgłoszenie opinii

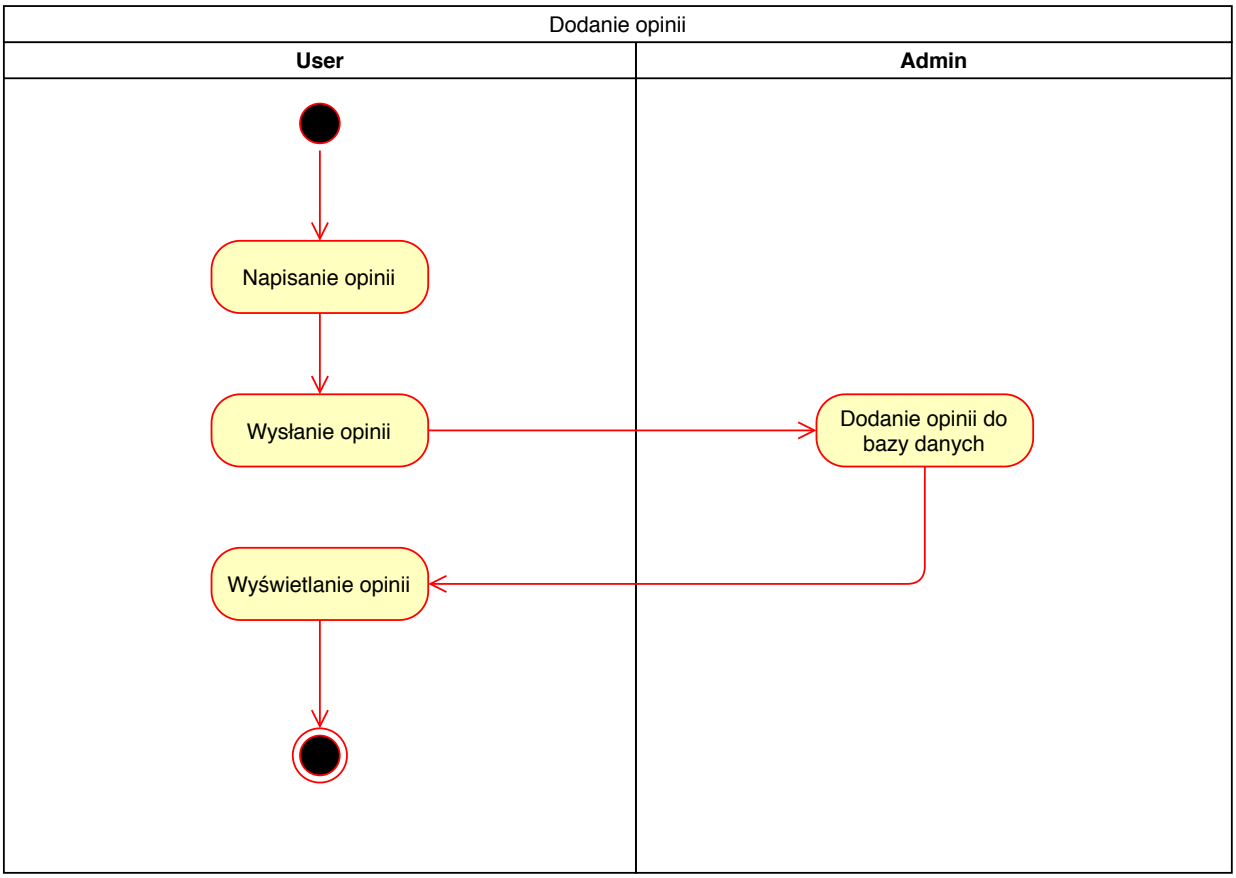
Zarówno w operacji dodania jak i zgłoszenia opinii uczestniczą dwa modułu: moduł użytkownika (User) oraz moduł admina (Admin).

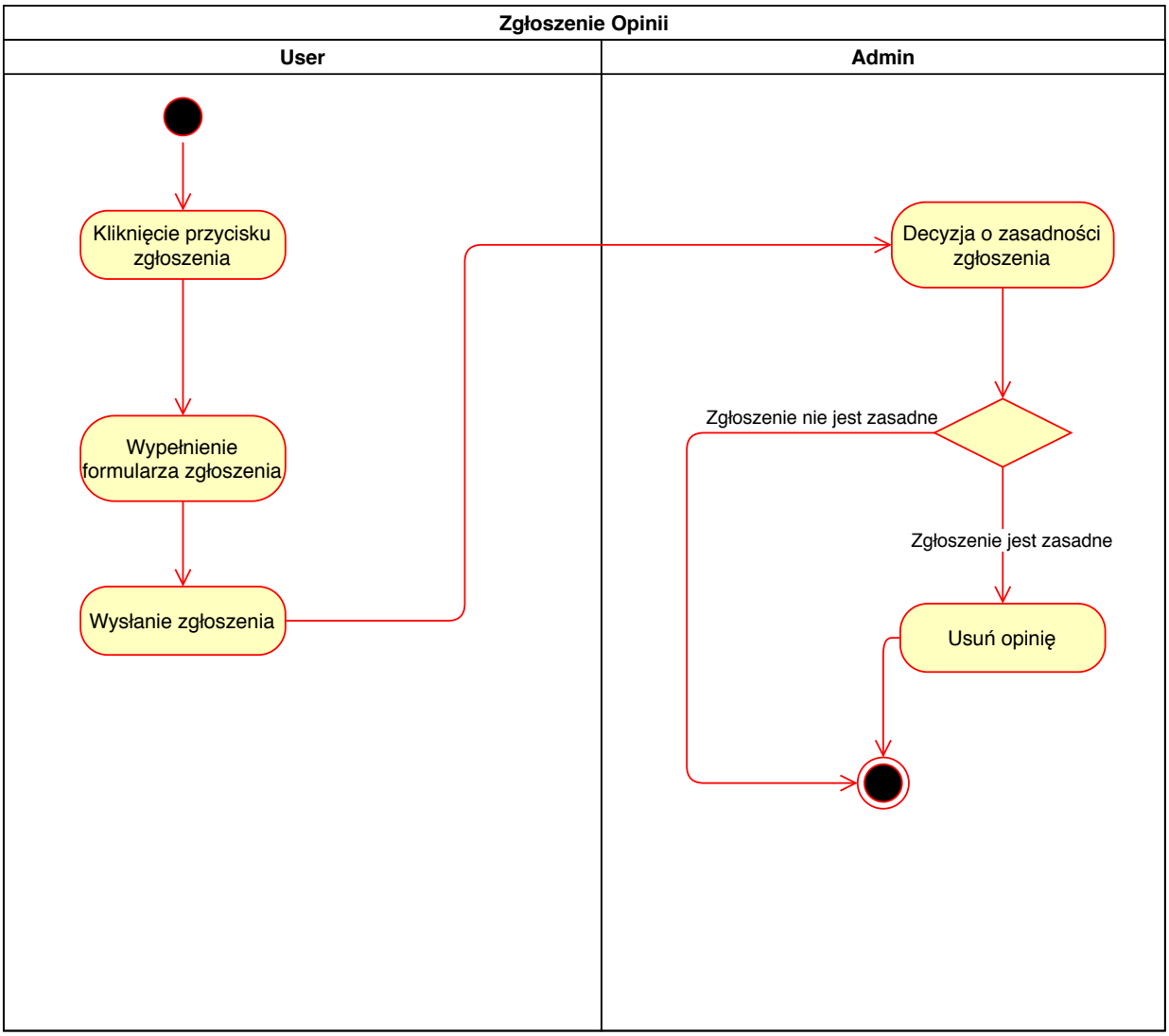
- Dodanie opinii

  Operacja rozpoczyna się w module użytkownika, który wpisuję treść swojej opinii i potwierdza chęć jej wysłania. Następnie akcja odbywa się po stronie serwera, który dodaję opinię do bazy danych i kieruję kontrolę systemu znów do modułu użytkownika, gdzie dana opinia jak i pozostałe są wyświetlane.

- Zgłoszenie opinii

  Operacja rozpoczyna się w module użytkownika, który widząc nieodpowiednią treść opinii chce ją zgłosić. Po wypełnieniu odpowiedniego formularza (jakie zasady regulaminu narusza opinia itd.) użytkownik zatwierdza zgłoszenie i potwierdza jego wysłanie. Następnie kontrola systemu przechodzi do modułu admina, gdzie odpowiednia osoba sprawdza, czy opinia faktycznie naruszyła regulamin aplikacji/gamebarów. W przypadku, gdy zgłoszenie opinii było bezzasadne opinia nie zostaje usunięta. W przeciwnym przypadku opinia zostaje usunięta.


Dodanie opinii
User
Admin
Napisanie opinii
Wysłanie opinii
Dodanie opinii do bazy danych
Wyświetlanie opinii


Zgłoszenie Opinii
User
Admin
Kliknięcie przycisku zgłoszenia
Wypełnienie formularza zgłoszenia
Wysłanie zgłoszenia
Decyzja o zasadności zgłoszenia
Zgłoszenie nie jest zasadne
Zgłoszenie jest zasadne
Usuń opinię

9 Opis REST API

Moduły w aplikacji porozumiewają się z bazą danych za pomocą REST API, które oddziela część frontendową od bazy. API przetwarza requesty i następnie wysyła odpowiednie komunikaty do bazy bazy. Poniżej znajdują się opisy wiadomości wysyłanych między modułami, z uwzględnieniem ich typów.

9.1 HTTP GET

Wspólne	
getReservations(clientId,gamebarID)	pobiera rezerwacje klienta lub gamebaru do wyświetlenia
getGameOpinions(gameID)	pobiera opinie o grze do wyświetlenia
getGamebarOpinions(gamebarID)	pobiera opinie o gamebarze do wyświetlenia
getDiscounts	mieć możliwość założenia konta w aplikacji, logowania się oraz odzyskiwania hasła
Klient	
getReservations(clientID)	pobiera rezerwacje danego klienta do wyświetlenia
Menadżer	
getReservations(gamebarID)	pobiera rezerwacje danego gamebaru do wyświetlenia
getReport(gamebarID)	pobiera raport dla danego gamebaru
getClients(gamebarID)	pobiera liste klientów danego gamebaru

9.2 HTTP POST

Wspólne	
Klient	
makeReservation(gamebarID,datetime,userID)	utworzenie rezerwacji
applyCode(userID,codeID,reservationID)	zaaplikowanie kodu rabatowego
makeGameOpinion(gameID,clientID,text)	utworzenie opinii do danej gry
makeGamebarOpinion(gamebarID,clientID,text)	utworzenie opinii do danego gamebaru
Menadżer	
reportUser(userID)	zgłoszenie użytkownika
tryAddGame(gamebarID,gameName,platform)	dodanie gry do oferty gamebaru
addDiscount(discountType, discountData)	dodanie promocji

9.3 API Messages

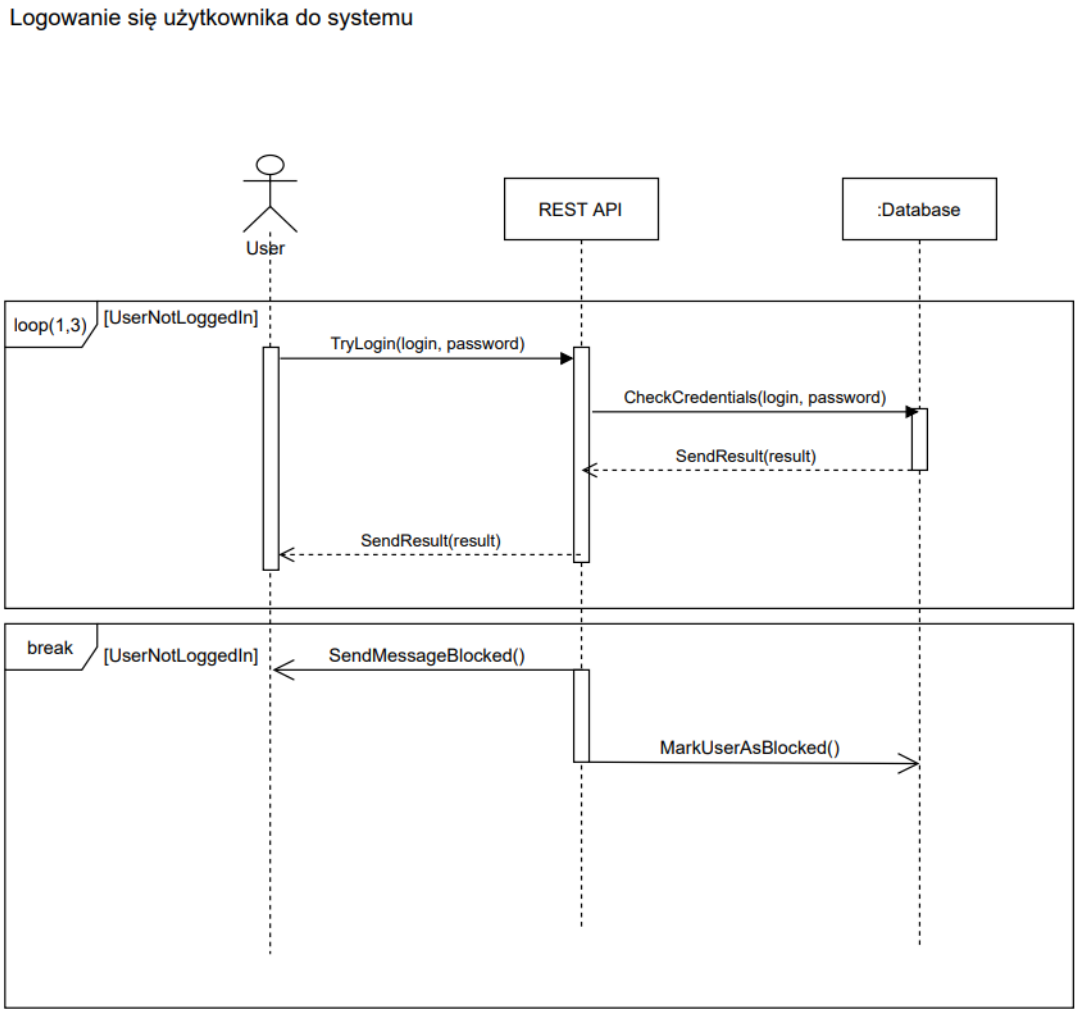
API będzie również wysyłać dodatkowe komunikaty lub żądać podania dodatkowych informacji np. podczas zapytania o sposób płatności.

Klient	
NotifySuccessApplayingCode(ClientId, reservationId)	wysłanie notyfikacji o sukcesie w aplikowaniu kodu
RequestPaymentMethod(clientId, reservationId)	zapytanie o sposób płatności za rezerwacje, teraz bądź później
RequestPayment(clientId, reservationID)	request o zapłatę
Menadżer	
NotifyAboutReportedUser(userId, idDeted)	wysłanie notyfikacji o pomyślnym zgłoszeniu użytkownika
NorifyAboutAddingGame(gameId)	wysłanie notyfikacji o pomyślym dodaniu gry do listy gier danego gamebaru

10 Komunikacja w systemie

Uwaga: We wszystkich diagramach zakładamy, że *REST API* jest częścią serwera administratora, który pełni również rolę serwera komunikacji. Komunikacja między pozostałymi modułami odbywa się za pomocą *HTTP*.

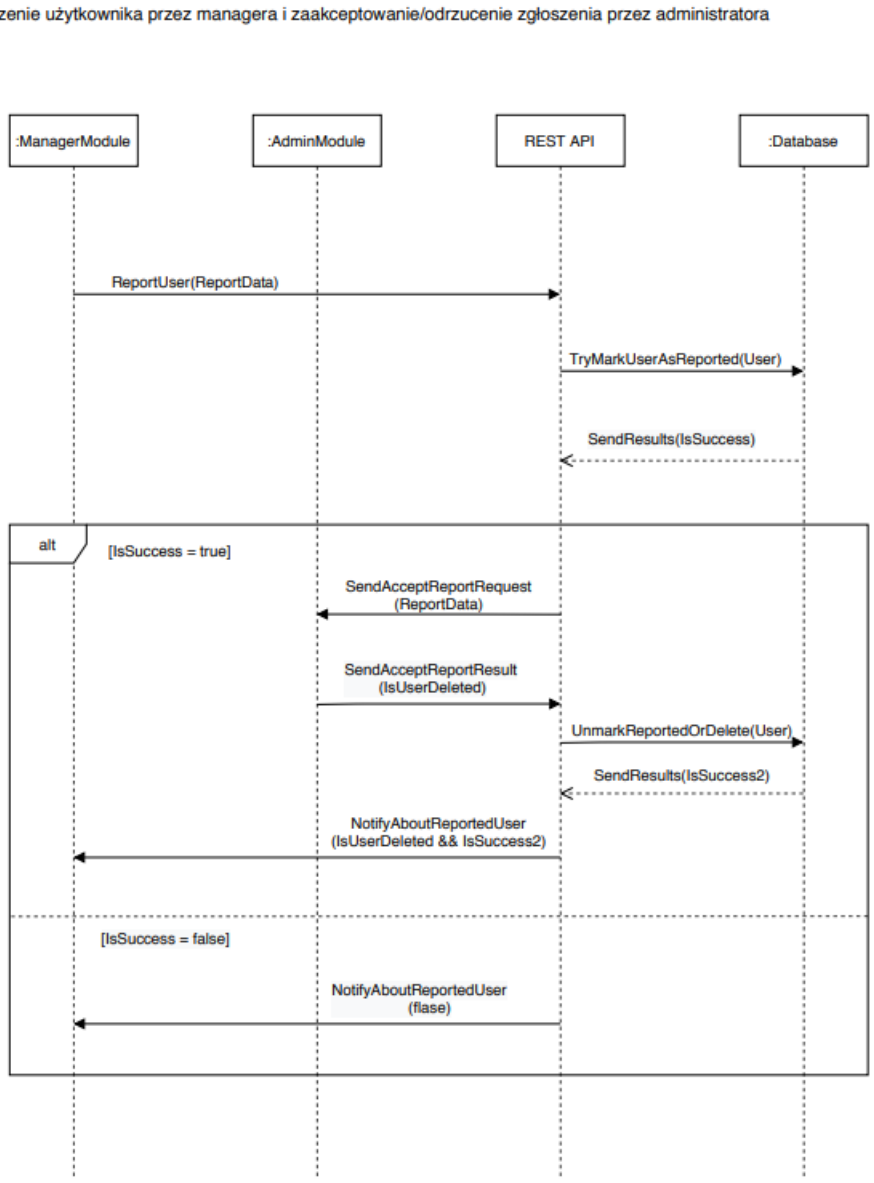
10.1 Logowanie użytkownika do systemu



Klient wpisuje dane logowania w aplikacji klienckiej, które pobierane są przez moduł użytkownika. Następnie moduł użytkownika za pomocą REST API, które jest częścią serwera administratora, łączy się z bazą danych w celu sprawdzenia podanych danych logowania. REST API wysyła do bazy danych dane logowania w celu sprawdzenia ich poprawności (zakładamy uproszczony model - jeżeli podane przez REST API dane logowania znajdują się w bazie baza zwróci *true*, wpp. zwróci *false*.). Użyt-

kownik może podać dane logowania maksymalnie 3 razy (loop(1,3)). Po trzykrotnym wprowadzeniu niepoprawnych danych (baza danych zwróciła *false*) użytkownik zostaje zablokowany (zakładamy, że nastąpiła próba włamania na konto użytkownika). REST API wysyła do bazy polecenie oznaczenia użytkownika jako zablokowanego, a do użytkownika powiadomienie o jego zablokowaniu.

10.2 Zgłoszenie użytkownika do usunięcia przez managera



Powyższy diagram przedstawia proces zgłaszania, przez managera, użytkownika do usunięcia/zablokowania w wyniku jego niewłaściwego zachowania. Moduł managera wysyła request zgłoszenia użytkownika zawierający powód zablokowania oraz dane użytkownika. REST API wysyła request do bazy danych, który ma na celu podjęcie próby oznaczenia użytkownika jako zablokowanego. Po wykonaniu operacji baza danych zwraca wynik, który następnie jest przesyłany do modułu administratora. Zgodnie z naszym systemem administrator podejmuje decyzję czy wysłana przez managera prośba o usunięcie użytkownika jest dobrze uzasadniona.

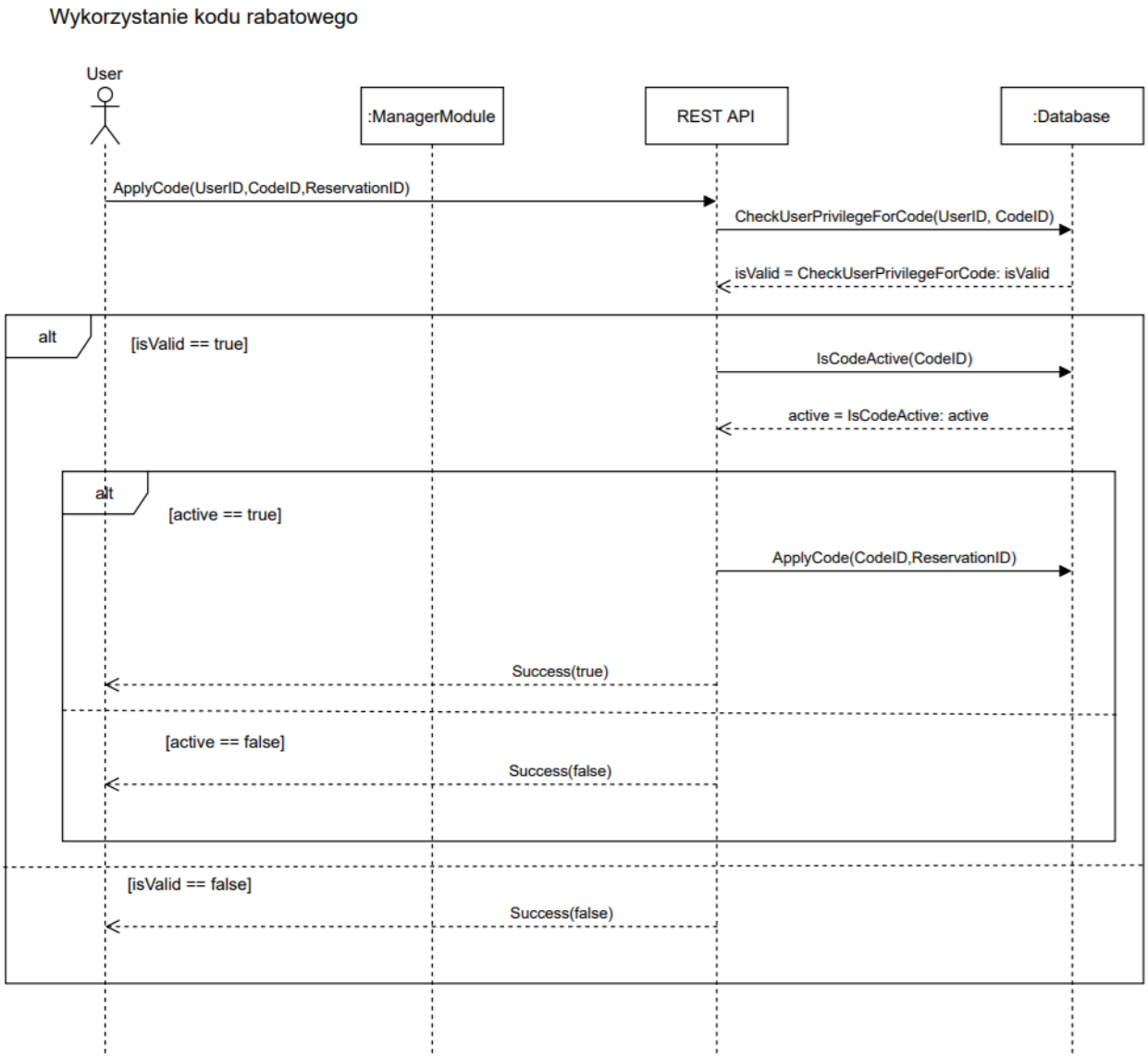
Jeżeli tak, przez REST API wysyłane jest zapytanie do bazy danych, które usuwa użytkownika z systemu.

Jeżeli prośba o usunięcie użytkownika nie jest wystarczająco dobrze umotywowana, do bazy danych wysyłany jest request aby odblokować użytkownika.

W podstawowej wersji naszego systemu nie ma powiadamiania managera o wyniku operacji blokowania użytkownika. Można dodać taką operację jako COULD HAVE.

Dokładnie analogiczny schemat komunikacji występuje w przypadku zgłaszania i usuwania opinii.

10.3 Wykorzystanie kodu rabatowego



Powyższy diagram przedstawia proces wykorzystania kodu rabatowego przez klienta gamebaru. Moduł użytkownika wysyła do serwera ID użytkownika, kodu oraz rezerwacji, której ma zostać zastosowany kod rabatowy. Na podstawie otrzymanych danych serwer wysyła bazie danych request aby potwierdziła poprawność danych - czy użytkownik o danym ID ma faktycznie uprawnienia do użycia podanego przez niego kodu (czy istnieje taki rekord w bazie danych). Następnie na podstawie informacji zwrotnej od bazy danych serwer wybiera jedną z dwóch możliwości: 1. Jeżeli użytkownik ma przywileje, aby użyć kodu rabatowego, ponownie do bazy danych wysyłane jest zapytanie. Tym razem

ma ona sprawdzić czy podanych przez użytkownika kod jest nadal aktywny (promocja nie wygasła). Jeżeli kod jest aktywny wysyłany jest request do bazy danych mający na celu przypisanie kodu rabatowego do wybranej przez użytkownika rezerwacji. 2. Jeżeli użytkownik nie może skorzystać z tego kodu zwracana jest informacja o błędzie do modułu użytkownika.

10.4 Wykonanie rezerwacji

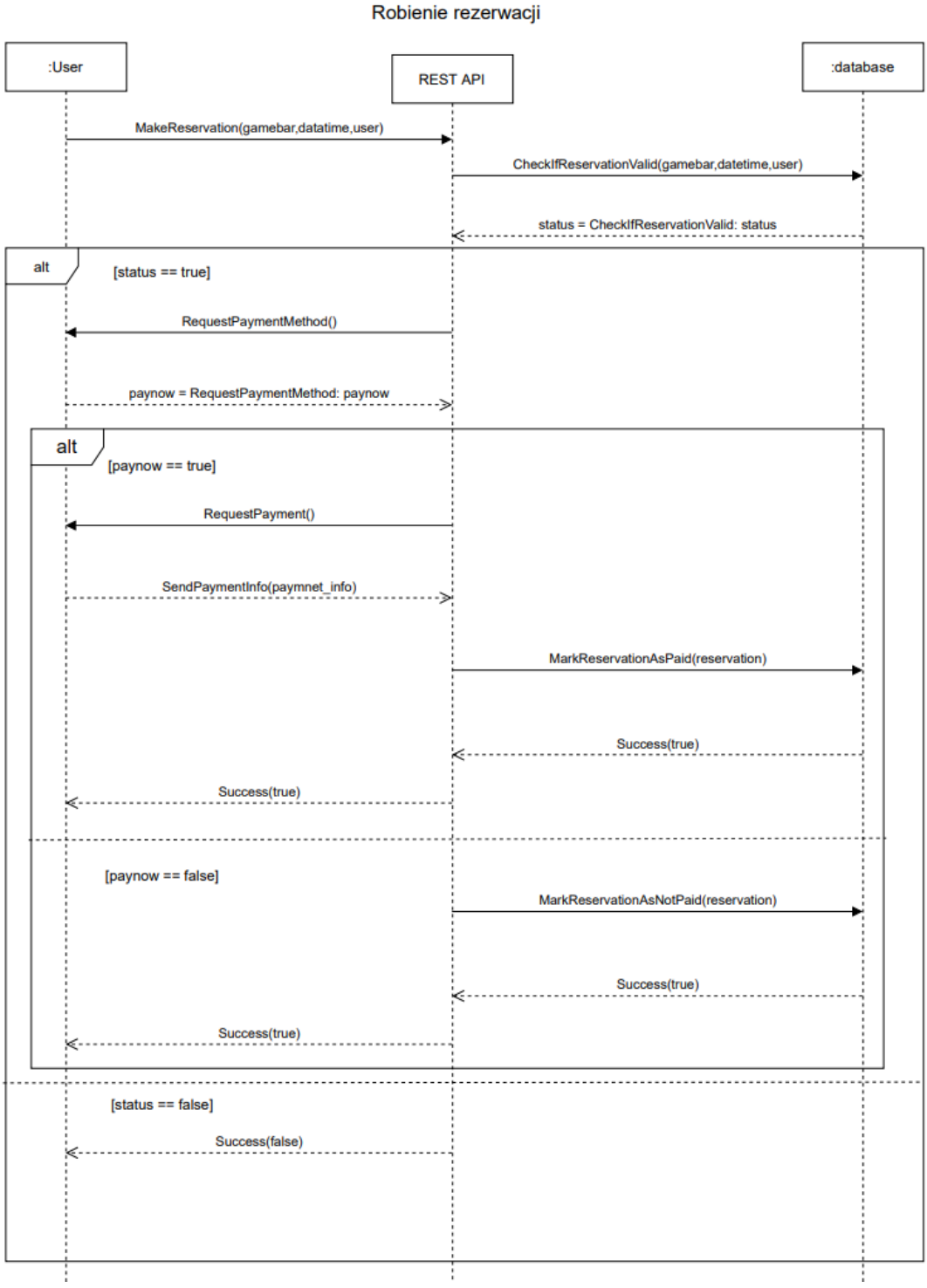


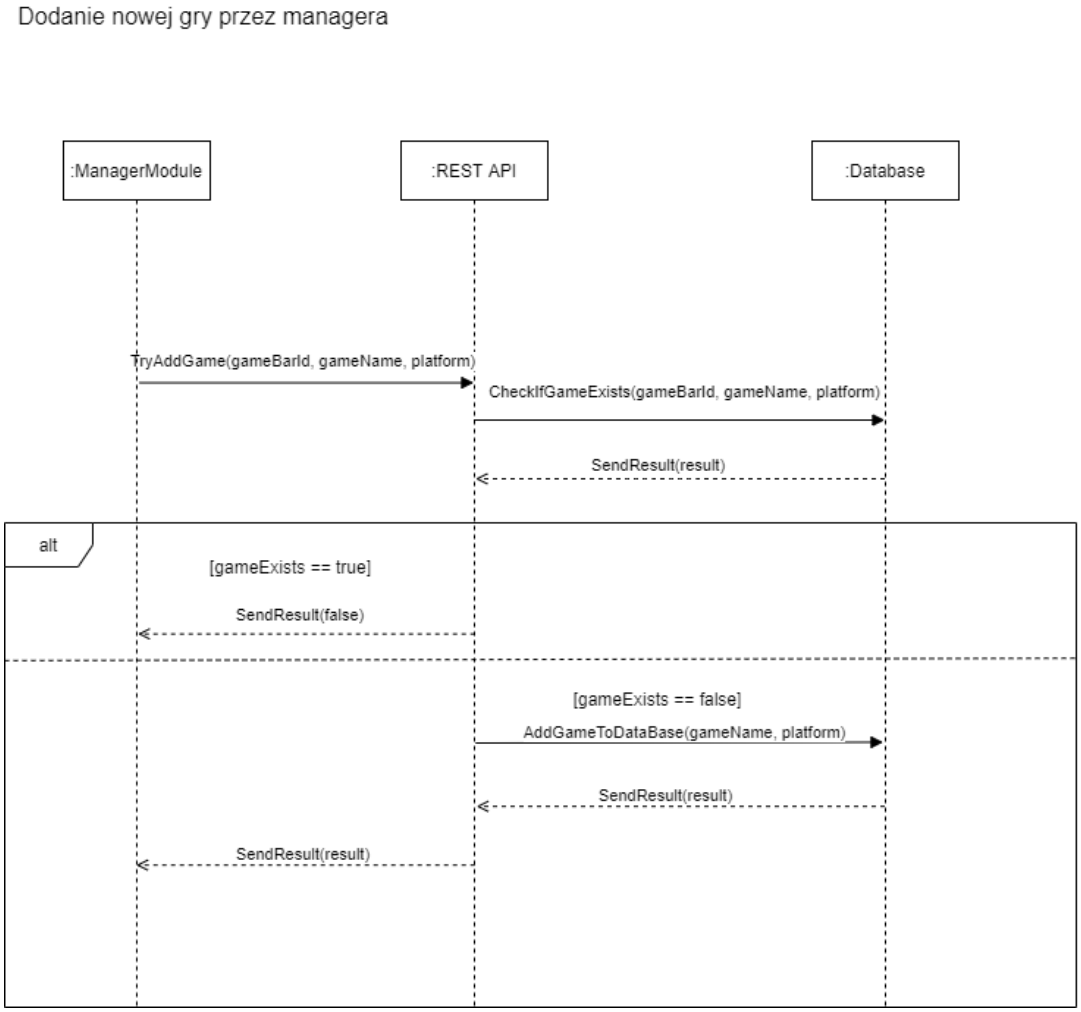
Diagram przedstawia sekwencję dodawania rezerwacji przez użytkownika. Moduł użytkownika za pomocą REST API wyraża chęć dokonania rezerwacji w określonym gamebarze, o określonej godzinie. Następnie serwer wykonuje request do bazy danych aby sprawdzić czy dokonanie takiej rezerwacji jest możliwe (czy podany termin nie jest zajęty). Jeśli wykonanie rezerwacji jest możliwe, serwer żąda wybrania metody płatności od modułu użytkownika. Jeśli dokonanie rezerwacji w określonym terminie nie jest możliwe do modułu użytkownika zwraca jest informacja o niepowodzeniu operacji.

W przypadku gdy do modułu użytkownika dotarło żądanie wyboru metody uiszczenia opłaty, użytkownik wybiera czy chce dokonać opłaty teraz czy w późniejszym terminie po czym moduł użytkownika przekazuje wybór do serwera.

Jeśli użytkownik chce dokonać opłaty od razu, serwer wysyła żądanie zapłaty. Odpowiedzią modułu użytkownika jest request z danymi potrzebnymi do przeprowadzenia płatności. Po ich otrzymaniu i autoryzacji płatności, serwer oznacza, za pomocą requestu *MarkReservationAsPaid*, w bazie danych rezerwację jako opłaconą i zwraca do modelu użytkownika informację o zakończeniu płatności.

Jeśli użytkownik chce dokonać płatności w późniejszym terminie, serwer wysyła do bazy danych request *MartkReservationAsNotPaid* aby oznaczyć rezerwację jako nieopłaconą, po czym zwraca informację do modułu użytkownika o przełożeniu płatności na przyszłość.

10.5 Dodanie gry przez managera

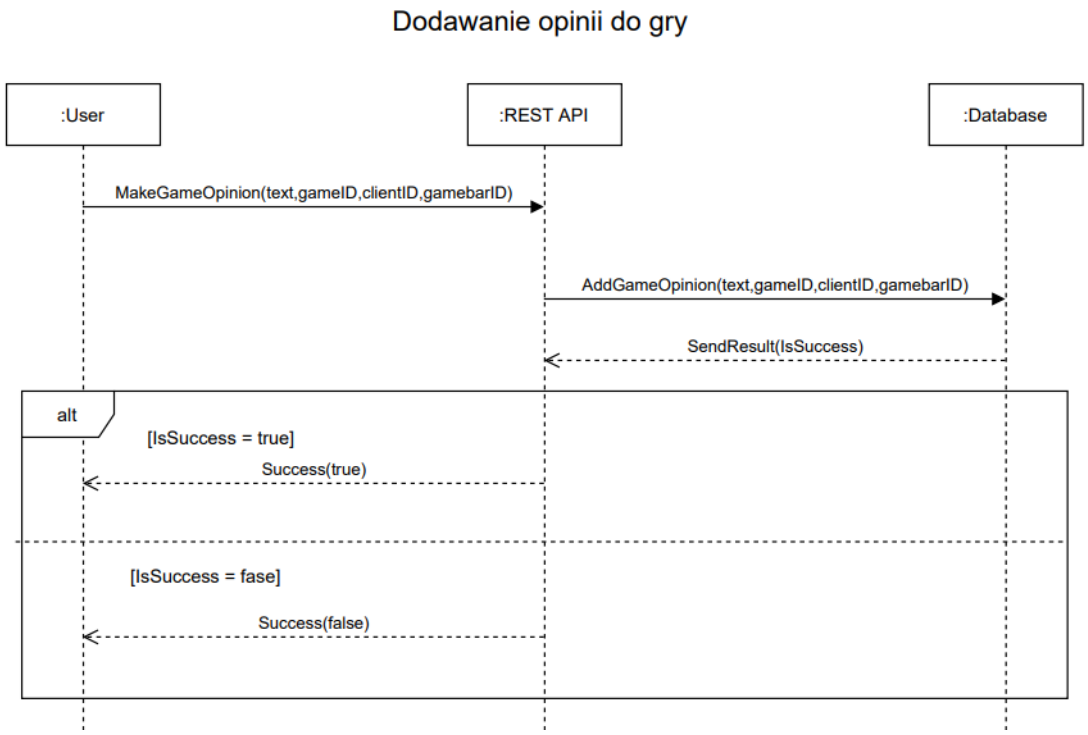


Powyższy diagram przedstawia sekwencję mającą na celu dodanie nowej gry do oferty gamebaru. Moduł managera wykonuje request podając ID baru, nazwę gry oraz nazwę platformy, na której ta gra działa (PC, PS4 itp). Następnie serwer za pomocą sprawdza w bazie danych czy gra podana przez managera nie znajduje się już w ofercie danego gamebaru.

Jeśli gra jest już w ofercie, do modułu managera zwracana jest informacja o zaistniałym problemie.

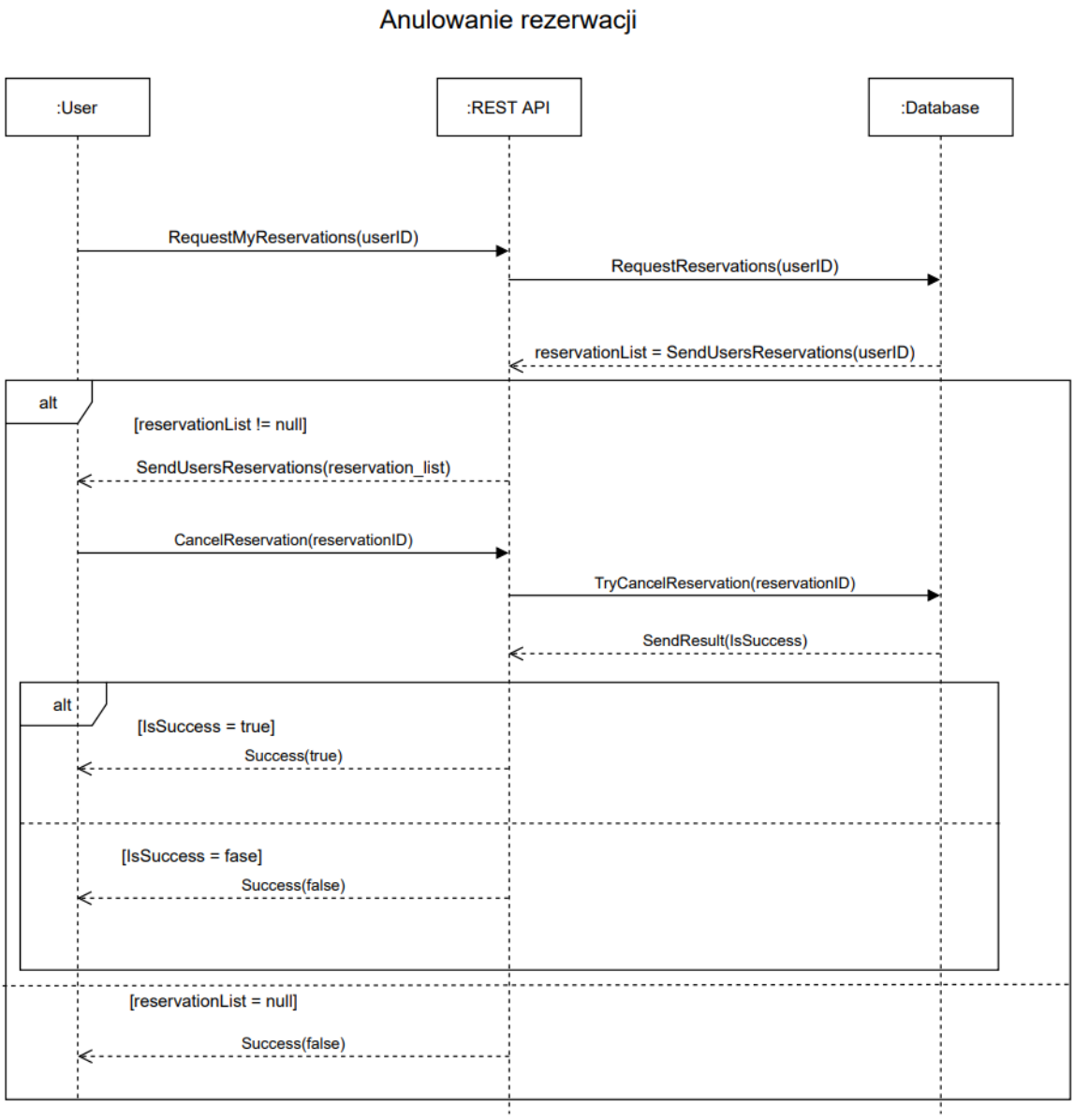
Jeśli gry nie ma jeszcze w ofercie gamebaru, wykonywany jest kolejny request do bazy danych (*AddGameToDataBase(...)*) w celu dodania gry do oferty. Do serwera a następnie do modułu managera zwracana jest informacja o tym czy operacja została zakończona sukcesem.

10.6 Dodawanie opinii przez użytkownika



Powyższy diagram przedstawia proces dodawania opinii przez użytkownika. Moduł użytkownika wykonuje request *MakeGameOpinion*, w którym podane są: tekst opinii, id klienta, który chce dodać opinię oraz gamebar, w którym ta opinia ma zostać dodana. Następnie serwer wykonuje request do bazy danych aby dodała ona do swoich rekordów nową opinię, podając treść opinii, ID gry, o której jest ta opinia, oraz ID gamebaru i ID klienta, który ją dodaje. Baza danych zwraca wynik operacji, który jest przekazywany do serwera a następnie do modułu użytkownika.

10.7 Anulowanie rezerwacji



Na diagramie przedstawiony jest proces komunikacji modułów w celu anulowania rezerwacji przez użytkownika. W pierwszym requestcie moduł użytkownika żąda listy rezerwacji. Serwer pobiera od bazy danych listę rezerwacji użytkownika, a następnie (jeśli lista nie jest nullem) zwraca ją do modułu użytkownika. Po otrzymaniu listy rezerwacji użytkownik wybiera rezerwację, którą chce anulować, a następnie moduł użytkownika przekazuje ID tej rezerwacji do serwera. Następnie ID przekazywane jest do bazy danych, która próbuje anulować rezerwację. Wynik operacji przekazywany jest przez serwer do modułu użytkownika.

11 Przypadki szczególne

Podczas działania systemu może wystąpić sytuacja, w której jeden z modułów ulegnie awarii i nie będzie przez pewien czas dostępny.

11.1 Awarie modułu klienta lub menedżera

W projektowanym systemie, komunikacja modułów klienta i managera z serwerem odbywa się poprzez wysyłanie odpowiednich zapytań na API, a następnie czekaniu na odpowiedź, informującą o poprawnym wykonaniu żądanej akcji. Zatem jeśli moduł klienta lub managera ulegnie awarii przed wysłaniem zapytania, lub otrzymaniu odpowiedzi z serwera, nie wpłynie to na przebieg komunikacji w systemie. W mało prawdopodobnej sytuacji, w której moduł klienta lub menedżera ulegnie awarii po wysłaniu zapytania, a przed otrzymaniem odpowiedzi z serwera, użytkownik powinien sprawdzić, czy żądana akcja została wykonana, np. poprzez wejście w listę aktywnych rezerwacji, aby sprawdzić, czy nowa rezerwacja została pomyślnie dodana.

11.2 Awarie modułu serwera

W sytuacji, w której serwer ulega awarii i jednocześnie pewien moduł próbuje wysłać zapytanie na API, moduł ten powinien otrzymać odpowiedź o kodzie 501, oznaczającą błąd po stronie serwera i spróbować wysłać dane zapytanie ponownie. Po 3 próbach, należy zaprzestać kolejnych i poinformować użytkownika o zaistniałej sytuacji. Tak samo należy postąpić, jeśli awarii uległaby baza danych, z której korzysta serwer.

12 Schematy wiadomości

12.1 Informacje ogólne

Poniżej przedstawione zostają schematy wiadomości, za pomocą których odbywać się będzie komunikacja z API. Schematy te zostały zapisane w języku RAML.

```
#%RAML 1.0
title: REPLAY api
baseUri: http://Replay.com/api
version: 1.0

uses:
  assets: assets.lib.raml

annotationTypes:
  monitoringInterval:
    type: integer
```

12.2 Dostępne typy

```
Gamebar:
  properties:
    id:
      type: integer
      description: ID of the gamebar
    name:
      type: string
      decription: Name of the gamebar
    description:
      type: string
      decription: Description of the gamebar.
```

```
Console:
  properties:
    id:
      type: integer
      decription: ID of the console
    name:
      type: string
      description: Name of the conolse.
    description:
      type: string
      decription: Description of the console.
```

```
Game:
  properties:
    id:
      type: integer
      decription: ID of the game.
    name:
      type: string
      decription: Name of the game.
    description:
      type: string
      decription: Description of the game.
```

```
SpecialOffer:
  properties:
    id:
      type: integer
      decription: ID of the special offer.
    name:
      type: string
      decription: Name of the special offer.
    description:
      type: string
      decription: Description of the special offer.
    discountType:
      type: string
      description: Type of discount e.g. FreeEntry.
```

```
DiscountToAdd:
  discountType:
    type: string
    description: Type of discount e.g. FreeEntry.
  discountData:
    type: string
    description: Data of the discount, different for each type.
```

```
Reservation:
  properties:
    id:
      type: integer
      description: ID of the reservation.
    gamebar:
      type: Gamebar
      description: Gamebar the reservation is for.
    Console:
      type: Console
      description: Console the reservation is for.
    game:
      type: Game
      description: Game the reservatinon is for.
    date:
      type: datetime
      description: Date of the reservtion (when it starts).
    time:
      type: datetime
      description: Duration of the reservation.
    specialOffer:
      type: SpecialOffer
      description: Special Offer (e.g. Discount Code) if added.
    isPaid:
      type: bool
      description: If user has already paid for the reservation.
    userId:
      type: integer
      description: ID of the user that makes the reservation.
```

```
GameOpinon:
  properties:
    id:
      type: integer
      description: ID of the game opinion.
    overallRating:
      type: integer
      description: Overall rating of the game.
    graphicsOpinion:
      type: integer
      description: Graphics rating.
    storyOpinion:
      type: integer
      description: Story rating.
    gameplayOpinion:
      type: integer
      description: Gameplay rating.
    text:
      type: string
      descritpion: Text the user writes when giving the opinion.
```

```
GamebarOpinon:
  properties:
    id:
      type: integer
      description: ID of the gamebar opinion.
    overallRating:
      type: integer
      description: Overall rating of the gamebar.
    serviceOpinion:
      type: integer
      description: Service rating of the gamebar.
    availableGamesOpinion:
      type: integer
      descritpion: Available Games rating.
    text:
      type: string
      descritpion: Text the user writes when giving the opinion.
```

```
GameToAdd:
  gameID:
    type: integer
    description: ID of the game being added.
  gamebarID:
    type: integer
    description: ID of the gamebar which is adding the game.
```

```yaml
User:
  properties:
    id:
      description: ID of the user.
      type: integer
    name:
      description: Name of the user.
      type: string
    surname:
      description: Surname of the user.
      type: string
    email:
      description: Email of the user.
      type: string
    nick:
      description: Nickname of the user.
      type: string
```

```yaml
ClientReservation:
  properties:
      userID:
        type: integer
        description: ID of client who is making the reservation.
      gamebarID:
        type: integer
        description: ID of gamebar the reservation is being made in.
      platform:
        type: string
        description: Name of console platform e.g. PS3
      date:
        type: datetime
      time:
        type: datetime
```

```
ClientCode:
  userID:
    type: integer
    description: ID of client who is applying the discount code.
  reservationID:
    type: integer
    description: ID of reservation for which the code
    is being applied.
  code:
    type: string
    description: Code to redeem
```

```
UserReport:
  userID:
    type: integer
    description: User to be reported.
  text:
    type: string
    description: Reason for reporting the user.
```

```
LoginCredentials:
  login:
    type: string
  password:
    type: string
```

12.3 Wiadomości użytkownika

```
/login:
  post:
    body:
      application/json:
        type: LoginCredentials
    responses:
      200:
        description: User credentials correct.
      401:
        description: User credentials incorrect.
      403:
        description: User blocked after third unsuccessful login trial.
```

```
/makeReservation:
  post:
    body:
      application/json:
        type: ClientReservation
    responses:
      200:
        description: Reservation made succesfully.
      404:
        description: Unable to make reservation.
```

```
/applyCode:
  post:
    body:
      application/json:
        type: ClientCode
    responses:
      200:
        description: Code applied successfully.
      404:
        description: Unable to apply code.
```

```
/makeGameOpinion:
  post:
    body:
      application/json:
        type: GameOpinion
    responses:
      200:
        description: Opinion added successfully.
      404:
        description: Unable to add opinion.
```

```
/makeGamebarOpinion:
  post:
    body:
      application/json:
        type: GameBarOpinion
    responses:
      200:
        description: Opinion added successfully.
      404:
        description: Unable to add opinion.
```

12.4 Wiadomości menedżera

```
/reportUser:
  post:
    body:
      application/json:
        type: UserReport
    responses:
      200:
        description: User reported successfully.
      404:
        description: Unable to report user.
```

```
/tryAddGame:
  post:
    body:
      application/json:
        type: GameToAdd
    responses:
      200:
        description: Game added successfully.
      404:
        description: Unable to add game.
```

```
/addDiscount:
  post:
    body:
      application/json:
        type: DiscountToAdd
    responses:
      200:
        description: Discount added successfully.
      404:
        description: Unable to add discount.
```

```
/getReport:
  displayName: getReport
  get:
    description: Gets company activity reoport.
    queryParameters:
      gamebarId:
        type: integer
        description: specifies the gamebar.
    responses:
      200:
        body:
          application/json:
            type: string
      404:
        decription:
          Not exists.
```

```
/getUsers:
  displayName: getUsers
  get:
    description: Gets clients of specified gamebar.
    queryParameters:
      gamebarId:
        type: integer
        description: specifies the gamebar.
    responses:
      200:
        body:
          application/json:
            type: User[]
      404:
        decription:
          Not exists.
```

12.5 Wiadomości wspólne

```
/getReservations:
  displayName: Reservations
  get:
    description: Gets reservations for specified user or gamebar
    queryParameters:
      userId:
        type: integer
        description: specifies the user (gamebar: -1 for all)
      gamebarId:
        type: integer
        description: specifies the user (user: -1 if for all)
    responses:
      200:
        body:
          application/json:
            type: Reservation
      404:
        decription:
          Not exists.
      405:
        description: UserId = -1 for request from user or
        description: gamebarId = -1 for request from gamebar.
```

```
/getGameOpinions:
  displayName: getGameOpinons
  get:
    description: Gets opinions for specified game
    queryParameters:
      gameId:
        type: integer
        description: specifies the game.
    responses:
      200:
        body:
          application/json:
            type: GameOpinion[]
      404:
        decription:
          Not exists.
```

```
/getGamebarOpinions:
  displayName: getGamebarOpinons
  get:
    description: Gets opinions for specified gamebar
    queryParameters:
      gamebarId:
        type: integer
        description: specifies the gamebar.
    responses:
      200:
        body:
          application/json:
            type: GamebarOpinion[]
      404:
        decription:
          Not exists.
```

```
/getSpecialOffers:
  displayName: getSpecialOffers
  get:
    description: Gets special offers for specified gamebar
    queryParameters:
      gamebarId:
        type: integer
        description: specifies the gamebar.
    responses:
      200:
        body:
          application/json:
            type: SpecialOffer[]
      404:
        decription:
          Not exists.
```

13 Przypadki testowe

13.1 Dodanie/usunięcie gamebaru

Scenariusz dotyczy dodawania oraz usuwania gamebaru przez administratora systemu. Moduły uczestniczące: moduł menedżera gamebaru (do sprawdzenia poprawności akcji dodawania gamebaru), moduł administratora (do dodania gamebaru). Ponieważ serwer wraz z REST Api jest częścią modułu administratora systemu, jedynymi wiadomościami przesyłanymi w tym scenariuszu są wiadomości niezbędne do pobrania danych w module menedżera.

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji administratora systemu.	Zalogowano.
2	Wybierz akcję dodawania nowego gamebaru. Wypełnij formularz tworzenia przykładowymi danymi.	Gamebar dodany pomyślnie.
3	Uruchom aplikację menedżera gamebaru, logując się do nowo utworzonej jednostki.	Logowanie zakończone sukcesem. Poprawne dane wpisane w formularzu tworzenie gamebaru po stronie administratora.
4	W aplikacji administratora usuń gamebar utworzony w punkcie 2	Gamebar usunięty.
5	W aplikacji menedżera wykonaj próbę logowania do gamebaru utworzonego w punkcie 2.	Próba zakończona porażką - gamebar został usunięty.

13.2 Dodanie kodu promocyjnego

Scenariusz dotyczy dodania nowego kodu promocyjnego przez menedżera konkretnego gamebaru. Modułu uczestniczące: moduł menedżera gamebaru (do dodania nowego kodu), moduł użytkownika-klienta (do sprawdzenia, czy nowy kod jest poprawnie wyświetlany). Pomiędzy modułami, REST Api i bazą danych przesyłane są wiadomości o dodaniu nowego kodu.

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji menedżera.	Poprawne logowanie.
2	Przejdź do panelu tworzenia nowego kodu promocyjnego.	Wyświetlenie odpowiedniego okna.
3	Wybierz rodzaj kodu promocyjnego, który ma zostać dodany.	Wyświetlenie odpowiednich okien formularza dla wybranego rodzaju.
4a	Jeśli wybrano kod rabatowy - podaj wartość promocji (w %)	
4b	Jeśli wybrano kod przyznający dodatkowy czas - podaj wartość dodatkowego czasu (w godzinach)	
4c	Jeśli wybrano kod przyznający darmowe wejście dodatkowe informacje nie są potrzebne	
5	Uzupełnij obowiązkowe pola formularza niezależne od typu kodu (czas trwania promocji, adresaci)	
6	Zatwierdź dodanie nowej promocji.	Poprawne dodanie nowej promocji.
7	Zaloguj się do aplikacji klienta bez statusu stałego klienta	Poprawne logowanie.
8	Przejdź do okna odstępnych promocji w konkretnym gamebarze	Wyświetlenie listy promocji.
9	Sprawdź czy dodana promocja jest dostępna dla użytkownika, jeżeli jej adresatem są wszyscy klienci lub niedostępna, jeżeli jest skierowana tylko do stałych klientów.	Poprawne wyświetlenie promocji jako dostępnej lub niedostępnej.
10	Przeprowadź analogiczny proces z kroków 7 - 9 dla użytkownika ze statusem stałego klienta.	Przetestowanie dostępu do promocji wszystkich klientów.

13.3 Wykonanie rezerwacji z opcjonalnym wykorzystaniem kodu promocyjnego

Scenariusz dotyczy wykonania rezerwacji z opcjonalnym wykorzystaniem kodu promocyjnego przez użytkownika w konkretnym gamebarze. Moduły uczestniczące: moduł użytkownika-klienta (do wykonanie rezerwacji), moduł menedżera gamebaru (w celu sprawdzenia poprawnego dodania rezerwacji). Pomiędzy modułami, REST Api i bazą danych przesyłane są wiadomości o wykonaniu nowej rezerwacji.

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji użytkownika.	Poprawne logowanie.
2	Przejdź do odpowiedniego okna tworzenia nowej rezerwacji.	Wyświetlenie formularza do tworzenia rezerwacji.
3	Uzupełnij formularz przykładowymi danymi: wybór gamebaru, godziny rezerwacji.	
4	Opcjonalnie wykorzystaj dostępny kod promocyjny.	Uaktualnienie podsumowania rezerwacji (zmniejszenie kosztu, wydłużenie czasu).
5	Zatwierdź dokonanie rezerwacji.	Utworzenie rezerwacji w systemie.
6	Zaloguj się na konto menedżera gamebaru.	Poprawne logowanie.
7	Przejdź do panelu dokonanych rezerwacji.	Wyświetlenie na liście wszystkich rezerwacji nowo-utworzonej rezerwacji.

W podpunkcie 4 należy rozważyć 3 możliwe typy kodów promocyjnych w celu pełnego sprawdzenia ich poprawnego działania i operowania na obiekcie rezerwacji. Ponadto należy upewnić się, że kody, których czas aktywnego działania upłynął, nie są już aktywne i jest wyświetlany o tym odpowiedni komunikat. Podobny test powinien być przeprowadzony dla kodów dla stałych klientów.

13.4 Anulowanie rezerwacji

Scenariusz anulowania rezerwacji bada poprawności działania akcji anulowania dokonanej już rezerwacji. Możliwe są dwie drogi: anulowanie może zostać zaakceptowane (jeśli rezerwacja jeszcze się nie rozpoczęła) lub nie (jeśli rezerwacja już trwa nie można jej anulować). W tym scenariuszu uczestniczą moduły użytkownika-klienta gamebaru oraz menadżera gamebaru. Pomiędzy modułami, REST Api i bazą przesyłane są wiadomości o dokonaniu rezerwacji oraz jej anulowaniu.

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji użytkownika-klienta.	Zalogowano.
2	Wykonaj rezerwację zgodnie z przypadkiem testowym. Zadbaj o to by czas do rozpoczęcia rezerwacji był wystarczający na jej anulowanie (np. 15 minut).	Rezerwacja dokonana pomyślnie.
3	Uruchom aplikację menedżera gamebaru. Znajdź rezerwację złożoną w punkcie 2.	Rezerwacja istnieje po stronie managera.
4	Wejdź w "Moje rezerwacje" w aplikacji użytkownika-klienta, znajdź i wybierz utworzoną w punkcie 2 rezerwację.	Rezerwacja istnieje.
5	Wybierz akcję "Anuluj rezerwację".	Rezerwacja przestaje być oczekująca, zostaje oznaczona jako anulowana, przenoszona jest do historii rezerwacji.
6	Otwórz aplikację menedżera gamebaru. Zobacz oczekujące rezerwacje.	Rezerwacja z punktu 2 nie jest oczekująca, oznaczona jest jako anulowana.
7	Wykonaj rezerwację zgodnie z przypadkiem testowym. Zadbaj, by czas do rozpoczęcia rezerwacji był krótki (np. 1 minuta).	Rezerwacja utworzona pomyślnie.
8	Odczekaj do momentu, gdy rezerwacja się rozpocznie (zacznie trwać) i wykonaj punkty 3-4.	Według punktów 3 - 4.
9	Wybierz akcję "Anuluj rezerwację".	Rezerwacja nie może zostać usunięta ze względu na fakt, że zaczęła już trwać - pojawia się odpowiednia informacja.

13.5 Dodanie opinii

Scenariusz dotyczy dodawania opinii na temat gry oraz gamebaru. Moduły uczestniczące: moduł użytkownika-klienta, moduł menedżera gamebaru (do dodania gry oraz sprawdzenia, czy widać dodaną przez użytkownika opinię), moduł administratora (do dodania gamebaru). Przesyłane są wiadomości o dodanej opinii (między bazą, REST Api oraz modułami).

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji menedżera gamebaru.	Zalogowano.
2	Dodaj nową grę według przypadku testowego.	Gra dokonana pomyślnie.
3	Uruchom aplikację użytkownika-klienta. Znajdź grę dodaną w punkcie 2.	Gra istnieje po stronie użytkownika.
4	Wejdź w dodawanie opinii na temat gry i wpisz opinię.	Opinia dodana.
5	Upewnij się, że opinia jest widoczna z modułów mendedżera i użytkownika.	Opinia widoczna.
6	Powtórz kroki 1-6 tym razem dodając opinię na temat gamebaru.	Według punktów 1-6.

13.6 Zgłaszanie użytkownika lub opinii

Moduły uczestniczące: moduł użytkownika-klienta do dodania oraz zgłoszenia opinii, moduł menedżera gamebaru do zgłoszenia użytkownika, moduł administratora do zatwierdzenia lub odrzucenia zgłoszenia. Przesyłane są wiadomości o zgłoszonych i usuniętych opiniach oraz o zgłoszonych i zbanowanych użytkownikach (między bazą, REST Api oraz modułami).

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji użytkownika-klienta.	Zalogowano.
2	Zgłoś dwie opinię według przypadku testowego.	Opinie zgłoszone pomyślnie.
3	W aplikacji administratora zatwierdź pierwszą zgłoszoną opinię i odrzuć drugą	Zgłoszenia rozpatrzone pomyślnie i pomyślne wysłanie powiadomienia do użytkownika o statusie jego zgłoszeń
4	Po stronie użytkownika sprawdź czy przyszły powiadomienia	Powiadomienia otrzymane pomyślnie
5	Zaloguj się do aplikacji menadżera	Zalogowano.
6	Zgłoś pewnych dwóch użytkowników	Użytkownicy zgłoszeni pomyślnie.
7	W aplikacji administratora zatwierdź pierwszego zgłoszonego użytkownika i go zbanuj i odrzuć drugie zgłoszenie	Użytkownik 1 zbanowany pomyślnie i powiadomienia do menadżera i do zbanowanego użytkownika wysłane pomyślnie
8	W module menadżera sprawdź powiadomienia o zgłoszeniach	Powiadomienia otrzymane pomyślnie.
9	Zaloguj się na konto zbanowanego użytkownika	Przy logowaniu pomyślnie otrzymujemy powiadomienie o zbanowaniu

13.7 Dodawanie i usuwanie gier i konsoli do gamebaru

Moduł menedżera gamebaru do dodawania i usuwania gier i konsoli, moduł użytkownika-klienta do sprawdzania czy widać dodane gry/konsole. Przesyłane są wiadomości o dodanych/usuniętych grach i konsolach (między bazą, REST Api oraz modułami).

Krok	**Akcja**	**Oczekiwany rezultat**
1	Zaloguj się do aplikacji menadżera.	Zalogowano.
2	Dodaj konsolę do gamebaru	Gra dodana pomyślnie
3	Do dodanej konsoli dodaj kilka nowych gier	Gry dodane pomyślnie
4	Zaloguj się do aplikacji użytkownika	Zalogowano
5	Pobierz listę konsol i gier w danym gamebarze i sprawdź czy są na niej właśnie dodana konsola i gry na niej	Konsola i gry są na liście.
3	Od dodanej konsoli usuń jedną grę	Gra usunięta pomyślnie
4	Po stronie użytkownika sprawdź czy gra została usunięta	Gra usunięta z listy
2	Usuń konsolę z gamebaru	Konsola usunięta i lista jej gier usunięta pomyślnie
4	Po stronie użytkownika sprawdź czy konsola została usunięta	Konsola usunięta z listy

14 Opis technologii

Projekt opisany w powyższej dokumentacji zostanie zaprogramowany w języku C# jako aplikacja webowa.
Projekt został podzielony na 3 odrębne aplikacje (solucje):

1. Serwer, z interfejsem graficznym dla administratora systemu.
2. Aplikacja kliencka.
3. Aplikacja managera gamebaru.

Wszystkie aplikacje porozumiewają się z bazą danych pośrednio przez serwer za pomocą REST API.

Aplikacja Serwera będzie podzielona na dwa "podmoduły": serwer oraz aplikację administratora. Serwer zajmuje się wystawianiem API i komunikacją aplikacji managera, klienta oraz administratora z bazą danych. Serwer nie posiada frontendu.
Aplikacja administratora to głównie frontend umożliwiający administratorowi systemu wykonywanie opisanych w powyższym dokumencie akcji.
Serwer oraz aplikacja administratora mają w pełni odizolowaną logikę.

Aplikacja kliencka to strona internetowa umożliwiająca klientowi opisane powyżej akcje. Aplikacja porozumiewa się z bazą danych za pośrednictwem serwera i wystawionego przez niego API. W logice aplikacji znajduje się tylko przetwarzanie danych zwróconych przez serwer oraz wykonywanie requestów do serwera.

Struktura aplikacji managera wygląda dokładnie tak samo jak struktura aplikacji klienta, jednak korzysta ona z innych zapytań API serwera. Również frontend aplikacji różni się od frontendu aplikacji klienta, tak aby umożliwiał managerowi wszystkie działania.

Zakładamy, że frontend wszystkich aplikacji zostanie wykonany za pomocą Bootstrapa (lub w wyniku późniejszych zmian - za pomocą Angulara).

Aplikacja może zostać uruchomiona z dowolnego systemu operacyjnego, na dowolnym urządzeniu posiadającym przeglądarkę internetową oraz dostęp do internetu.

Deploy aplikacji, hostowanie strony internetowej oraz serwera będą wykonywane na portalu AZURE.

Baza danych zostanie wykonana w technologii AZURE SQL na portalu AZURE.

Logowanie do aplikacji może zostać wykonane za pomocą AZURE ADB2C, jednak zgodnie z założeniami przedmiotu może zostać pominięte i wykonane jako "sztuczna" autoryzacja użytkownika, polegająca na wpisaniu loginu (który będzie rozróżniał użytkowników w bazie danych) oraz wybraniu przycisku "Log in".