Inżynieria Systemów Informatycznych 2021
Pierwszy punkt kontrolny specyfikacji systemu

Paweł Gelar
Jakub Grunas
Arkadiusz Kniaź
Hubert Kozubek
Nikola Miszalska
Andrzej Pióro

grudzień 2021

Spis treści

1. Wstęp - cel i ogólna koncepcja systemu

Głównym celem systemu jest ułatwienie korzystania z komunikacji miejskiej podróżnym, przez dostarczenie na bieżąco aktualizowanych przewidywań dotyczących czasu wjazdu autobusu na przystanek. Informacje te będą wyświetlane na tablicach umieszczonych na przystankach.
Dodatkowo system ułatwia zarządzanie pojazdami m.in. przez wyświetlenie ich listy i położenia każdego z nich, albo automatycznego zgłaszania awarii.
Działanie systemu opiera się na dokładnych czujnikach prędkości, oraz położenia umieszczonych w autobusach, nadajnikach będących w stanie wysyłać te informacje regularnie i z niewielkim opóźnieniem.

2. Opis funkcjonalności

2.1 Opis kluczowych obiektów biznesowych systemu

W systemie rozpoznaliśmy 4 kluczowych obiektów biznesowych. Są nimi byty, bez których nie można wyobrazić sobie poprawnego funkcjonowania systemu. Poniżej są one opisane i objaśnione.

- Tablica

  Tablica jest obiektem, który dostarcza informacji podróżującym o aktualnym stanie autobusów, które przyjadą na przystanek. Jej moduł dostarcza dwóch funkcjonalności: pobierania danych z systemu centralnego oraz ich wyświetlania.

- Autobus

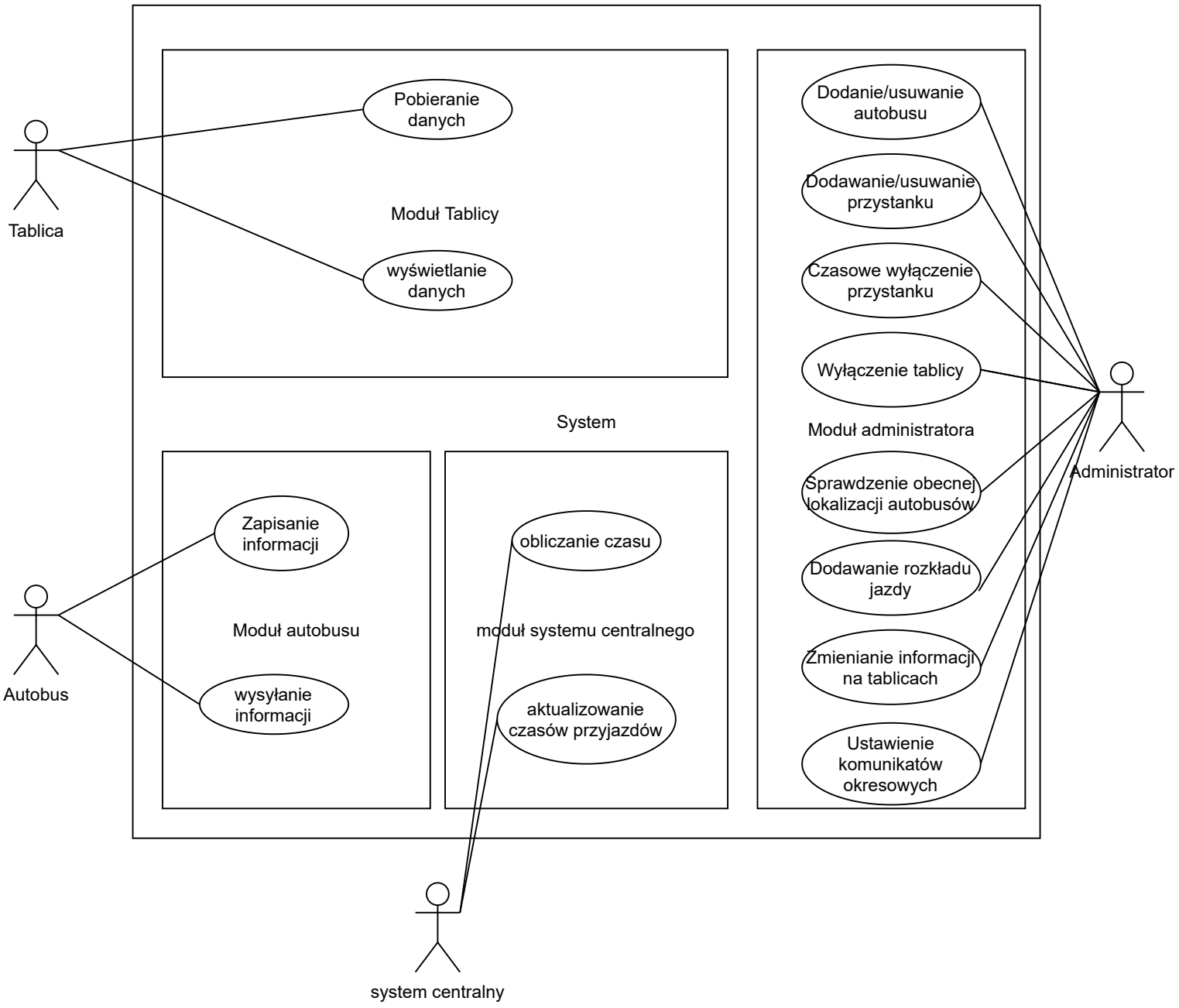
  Autobus również jest ważny z biznesowego punktu widzenia. Żeby system działał poprawnie, potrzebne są aktualne dane o lokalizacji i prędkości. Przy pomocy modułu autobusu te dane są wysyłane do systemu centralnego oraz zapisywane w celach statystycznych.

- System centralny

  System centralny oblicza przewidywany czas dojazdu autobusu do przystanków oraz wysyła te dane do tablic. Podobnie jak w przypadku innych obiektów, używa w tym celu modułu systemu centralnego.

- Administrator

  Ostatni użytkownik, czyli administrator, również jest kluczowym obiektem biznesowym. Dzięki swojemu modułowi, ma on możliwość manualnego dodawania/usuwania autobusów i przystanków, czasowego wyłączania tablic, manualnego sprawdzenia obecnej lokalizacji autobusów, dodania rozkładu jazdy oraz ustawiania informacji, które wyświetlane są na tablicach. Te funkcjonalności oraz obiekt administratora są bardzo ważne, by móc zarządzać całym systemem i utrzymywać go.


Tablica
Pobieranie danych
Moduł Tablicy
wyświetlanie danych
Dodanie/usuwanie autobusu
Dodawanie/usuwanie przystanku
Czasowe wyłączenie przystanku
Wyłączenie tablicy
Moduł administratora
Sprawdzenie obecnej lokalizacji autobusów
Dodawanie rozkładu jazdy
Zmienianie informacji na tablicach
Ustawienie komunikatów okresowych
Administrator
System
Zapisanie informacji
Moduł autobusu
wysyłanie informacji
Autobus
obliczanie czasu
moduł systemu centralnego
aktualizowanie czasów przyjazdów
system centralny

2.2 User stories

Cały system opiera się na działaniu i współpracy ze sobą w czasie rzeczywistym modułu autobusu, modułu systemu centralnego i modułu tablicy. Dodatkowo istnieje jeszcze moduł administratora, który wszystkim zarządza, ma możliwości dodawania nowych instancji danego rodzaju i nadzoruje działanie całego projektu. W celu stworzenia dokładnego opisu sposobu działania systemu powstały User Stories, opisujące potrzeby poszczególnych modułów, na podstawie których można zaimplementować funkcjonalności. User Stories zostały podzielone ze względu na moduł którego dotyczą oraz jak potrzebne są dane funkcjonalności do prawidłowego działania systemu. W ten sposób powstała poniższa lista:

MUST

- Chce przesyłać informacje o sobie (Autobus) - celem jest przede wszystkim przesłanie swojej aktualnej pozycji, aby czas przybycia autobusu mógł zostać zaktualizowany
- Chce dodawać autobusy, przystanki, linie autobusowe (Administrator) - aby uaktualniać system, rozbudowywać infrastrukturę oraz zachować spójność pomiędzy rzeczywistymi elementami i ich reprezentacją w systemie
- Chce dodawać rozkład jazdy (Administrator) - funkcja ta pozwala na dodanie danych początkowych, aby inne moduły w systemie miały jakiś punkt odniesienia, co ułatwi ich działanie
- Chce przewidywać czas przyjazdu autobusów (System centralny) - głównym celem całego systemu jest dostarczyć pasażerom jak najlepszych przewidywanie przyjazdu autobusu , dlatego też system centralny za to odpowiedzialny musi się tym zajmować
- Chce aktualizować czas przyjazdu autobusów (System centralny) - by była możliwość wyświetlania bieżących czasów przyjazdów system potrzebuje je aktualizować co jakiś czas, dzięki czemu poszczególne dane nie będą przestarzałe
- Chce pobierać dane do wyświetlenia (Tablica) - funkcja pozwalająca tablicy być na bieżąco z czasami przyjazdów autobusów
- Chce wyświetlać dane (Tablica) - jedna z podstawowych funkcjonalności całego systemu, umożliwia osobą przychodzącym na przystanek i tym czekającym sprawdzić najbliższe autobusy oraz zobaczyć przybliżony czas oczekiwania

Wśród pozostałych user stories znajdują się również te które powinny znaleźć się w systemie, w jego końcowej wersji (SHOULD) oraz takie które mogłyby się tam znaleźć, ale ich umieszczenie nie jest niezbędne, ani w znacznym stopniu kluczowe (COULD).

Wszystkie user stories można znaleźć w tabeli na następnej stronie. Większość z nich jest intuicyjna i nie wymaga tłumaczenia. Poniżej przedstawiono jeszcze te, które mogą wzbudzać pewne wątpliwości.

- Chce zmieniać informacje umieszczone na tablicach (Administrator, SHOULD) - w nieprzewidzianych sytuacjach, takich jak nagła awaria autobusy czy wypadek na trasie autobusu, administrator powinien móc zmienić

informacje o czasach przyjazdów autobusów

- Chce wyłączać czasowo przystanki (Administrator, COULD) - w wyjątkowej sytuacji, administrator powinien móc zmieniać informacje o trasach autobusów. Może to być potrzebne np. w przypadku remontu drogi, czy zamknięcia pewnych dróg ze względu na marsze.

Jako	Chcę	Aby	MoSCoW
autobus	przesyłać informacje o sobie	zaktualizować czas przybycia	must
autobus	zapisywać informacje o swojej lokalizacji i stanie	dostarczać dane statystyczne	should
administrator	sprawdzić obecną lokalizację autobusów	sprawniej działać w przypadku incydentów	should
administrator	zmieniać informacje umieszczone na tablicach	dostosowywać się do warunków	should
administrator	wyświetlać komunikaty okresowe	zapewnić aktualne informacje	should
administrator	dodawać autobusy	uaktualniać dane systemu	must
administrator	dodawać przystanki	uaktualniać dane systemu	must
administrator	wyłączać czasowo przystanki	dostosowywać się do sytuacji	could
administrator	usuwać przystanki	utrzymywać porządek w systemie	should
administrator	usuwać autobusy	utrzymywać porządek w systemie	should
administrator	wyłączać tablicę	stwarzać możliwość konserwacji	should
administrator	dodawać linię autobusową	uaktualniać dane systemu	must
administrator	dodawać rozkład jazdy linii	system posiadał dane początkowe	must
administrator	zmodyfikować linię autobusową	dostosować się do zmian	should
administrator	usuwać linię autobusową	utrzymywać porządek w systemie	should
system centralny	przewidywać czas przyjazdu autobusów	wyświetlić informacje na tablicy	must

system centralny	aktualizować czas przyjazdu autobusów	informować użytkowników o przybliżonym czasie oczekiwania	must
tablica	pobierać dane do wyświetlenia	wyświetlać dane	must
tablica	wyświetlać dane	przekazać informacje pasażerowi	must

2.3 Rewersy

Wymagania funkcjonalne dotyczą tego, co ma realizować system, jakie musi spełniać funkcje, jakich dostarczać usług oraz jak zachowywać się w określonych sytuacjach. Wymagania niefunkcjonalne określają pożądane cechy tworzonego systemu. Źródłem wymagań niefunkcjonalnych są klienci i odbiorcy. W wielu wypadkach wymagania niefunkcjonalne są równie istotne jak wymagania funkcjonalne systemu.

Dla wybranych user stories utworzyliśmy wymagania zarówno funkcjonalne jak i niefunkcjonalne, które powinny być spełnione aby system działał prawidłowo.

System centralny

- Wymagania funkcjonalne, które zostały postawione dla systemu centralnego to przewidywanie przyjazdu autobusów na podstawie dnia tygodnia, aktualnej godziny oraz obecnych świąt. System musi dostosowywać się do różnych warunków, takich jak wzmożony ruch na drodze dzień przed świętami, czy zatłoczone ulice w godzinach szczytu i wiarygodnie przewidywać czas przybycia autobusu.

- Do niefunkcjonalnych wymagań zaliczamy kompatybilność z systemem GPS, obliczanie nowego czasu w 30 sekund. Czy system będzie działał przez 99.9% czasu w ciągu kolejnych 5 lat? – system musi być niezawodny. Czy można aktualizować model przewidywania bez przerwy w działaniu systemu? – musi być pewność, że w trakcie obliczania czasu system będzie działał bez przerwy.

Administrator

- Wymagania funkcjonalne obejmują takie rzeczy jak zapewnienie administratorowi dodawanie kilku autobusów na raz, planowanie dodanie/usunięcie/modyfikacji autobusu oraz możliwość wykonania tych operacji przez Gui. Dodatkowo zapewniona jest możliwość wyświetlania tabeli z obecnymi autobusami oraz zapisywanie operacji w dziennikach. Powyższe funkcjonalności zostały stworzone dla wygodnej i łatwej obsługi systemu przez administratora. Dodawanie wielu autobusów na raz, czy planowanie różnych akcji może zdecydowanie zminimalizować i przyspieszyć czas pracy administratora, a dzięki zapisywaniu ich w dziennikach, w razie takiej potrzeby można do nich w każdym momencie wrócić.

- W wymaganiach niefunkcjonalnych zostały uwzględnione szybkość wykonywania operacji oraz otrzymanie jej potwierdzenia. Zapewnione zostało bezpieczeństwo poprzez wymóg każdorazowego uwierzytelniania oraz stabilność systemu.

Tablica

- Wymogi funkcjonalne zapewniają przede wszystkim czytelność i poprawność w wyświetlaniu danych. Umożliwienie przewijania tekstu dla długich nazw, listy autobusów i obecność paska na nagłe informacje jest konieczna, aby nie wprowadzić odbiorcy w błąd i przekazać ważne komunikaty.

- Wymogi niefunkcjonalne gwarantują dostępność, użytkowość oraz łatwość w odczytywaniu informacji przez odbiorcę.

Autobus

- Wymogi funkcjonalne dotyczące przesyłania informacji przez autobus gwarantują, że wszystkie niezbędne do działania systemu informacje (o lokalizacji, prędkości, stanie, paliwie oraz kierowcy) są wysyłane. Dodany jest również wymóg tworzenia kopii zapasowej w celu dodatkowego zabezpieczenia.

- Wymogi niefunkcjonalne dotyczą zapewnienia płynności w działaniu systemu oraz sposobów odbierania i przesyłania informacji. Uwzględniony jest również maksymalny dopuszczalny błąd w wysyłaniu informacji o czasie oraz szyfrowanie informacji. Niezawodność systemu zapewnia między innymi wymóg o możliwości wysyłania informacji z obszaru 97% tras.

Rewersy

- System centralny przewidywać cza przyjazdu autobusów

Funkcjonalne	Niefunkcjonalne
Czy przewiduje na podstawie dnia tygodnia?	Czy jest kompatybilny z systemem GPS w autobusie?
Czy przewiduje na podstawie aktualnej godziny?	Czy zdąży obliczyć nowy czas w 30 s?
Czy przewiduje na podstawie obecnych świąt?	Czy system będzie działał w 99.9% czasu w ciągu kolejnych 5 lat?
	Czy można aktualizować model przewidywania bez przerwy w działaniu systemu?

- Administrator dodawać/usuwać/modyfikować autobusy

Funkcjonalne	Niefunkcjonalne
Czy można dodawać wiele autobusów naraz?	Czy pojawia się potwierdzenie operacji?
Czy można zaplanować dodanie/usunięcie/modyfikację autobusu?	Czy operacja jest wykonywana natychmiast?
Czy ta operacja możliwa jest przez GUI?	Czy nowo dodany autobus jest inaczej oznaczony niż pozostałe?
Czy możliwe jest wyświetlenie tabeli z obecnymi autobusami?	Czy operacje wymagają każdorazowego uwierzytelniania?
Czy operacje są zapisywane w dziennikach?	Czy wykonanie operacji wymaga przerwania pracy systemu?

- Tablica wyświetlanie danych

Funkcjonalne	Niefunkcjonalne
Czy możliwe jest przewijanie nazw miejsc docelowych (dla długich nazw)?	Czy każda linia autobusowa wyświetla się innym kolorem?
Czy lista autobusów przewija się?	Czy nazwy linii są czytelne z odległości 10m przez osobę z przeciętnym wzrokiem?
Czy jest dodatkowy pasek na nagłe informacje?	Czy udogodnienia wyświetlane są innym kolorem?
	Czy udogodnienia występują w osobnej kolumnie?

- Autobus przesyłanie informacji

Funkcjonalne	Niefunkcjonalne
Czy wysyła informacje o aktualnej lokalizacji?	Czy podaje swoją lokalizację za pomocą współrzędnych geograficznych?
Czy wysyła informacje o aktualnym kierowcy?	Czy wysyła informacje co 10s?
Czy wysyła informacje o aktualnym stanie?	Czy wysyła informacje o pozycji z błędem <5m przez 80% czasu?

Czy wysyła informacje o aktualnej prędkości?	Czy informacje przesyłane są w formie pliku XML?
Czy wysyła informacje o poziomie paliwa/energii?	Czy możliwe jest wysłanie informacji z obszaru 97% tras?
Czy poza wysyłaniem autobus także przechowuje kopie tych danych?	Czy informacje są szyfrowane?

3. Opis struktury systemu

3.1 Diagramy klas

Klasa system centralny agreguje wiele obiektów trzech klas: linia, autobus, tablica, przechowuje dane o całym systemie.
Jej pola to:

- listaAutobusów: list<autobus> - lista w której są przechowywane informacje o autobusach
- listaTablic: list<tablica> - lista w której są przechowywane informacje o tablicach
- rozkładJazdy: list<linia> - lista w której są przechowywane informacje o liniach autobusowych

Jej metody to:

- void zmienStanTablicy(id,mode): - metoda do zmieniania stanu tablicy np włączenia, wyłączenia
- void obliczNoweCzasyOdjazdow(): - metoda obliczająca nowe czasy odjazdów autobusów na podstawie nowych danych
- void addTablica(tablica): - metoda do dodania nowej tablicy do systemu
- void delTablica(tablica): - metoda do usuwania tablicy z systemu
- void addAutobus(autobus): - metoda do dodania nowego autobusu do systemu
- void delAutobus(autobus): - metoda do usuwania autobusu z systemu
- void addLiniaAutobusowa(linia): - metoda do dodania nowej linii autobusowej do systemu
- void delLiniaAutobusowa(linia): - metoda do usuwania linii autobusowej z systemu
- void wyslijDaneDoWyswietlenia(): - metoda zbierająca dane do wyświetlenia na danej tablicy i wysyłająca je

Klasa linia połączona z klasą tablica, odpowiada liniom autobusowym.
Jej pola to:

- numerLinii: int - pole z numerem linii autobusowej utożsamiany z nazwą linii
- listaPrzystankow: list<tablica> - lista przystanków przechowuje tablice
- listaGodzinOdjazdow: hashmap<tablica,time> - hashmapa przechowuje dokładne czasy odjazdów autobusów z poszczególnych przystanków

Jej metody to:

- void addPrzystanek(tablica): - metoda do dodania nowej tablicy(przystanku) do danej linii autobusowej
- void delPrzystanek(tablica): - metoda do usuwania tablicy(przystanku) z danej linii autobusowej

Klasa abstrakcyjna autobus jest klasą matką dla dwóch klas autobusSpalinowy i autobusElektryczny odpowiadających odpowiednio autobusom spalinowym i elektrycznym, jest zagregowana przez system centralny, korzysta z klasy lokalizacja

Jej pola to:

- id: int - pole z id autobusu, identyfikuje dany autobus
- lokalizacja: lokalizacja - pole z lokalizacją danego autobusu korzysta z klasy lokalizacja
- predkosc: float - pole z przechowujące prędkość danego autobusu
- idKierowca: int - pole z id kierowcy danego autobusu
- error: errorTypeAutobus - pole enum z rodzajami błędów które mogą wystąpić w autobusie: {brakPaliwa, rozladowanieBaterii, przebitaOpona, awariaHamulcow, awariaSilnika}
- nrlinii: int - pole z numerem linii po której porusza się dany autobus

Jej metody to

- void aktualizujLokalizaje(): - metoda ta aktualizuje dane o lokalizacji danego autobusu
- void wyslijDane(): - metoda ta zbiera i wysyła dane o lokalizacji prędkości i ewentualnej awarii do systemu centralnego
- void aktualizujPredkosc(): - metoda ta aktualizuje prędkość danego autobusu

Klasa autobusSpalinowy dziedziczy po klasie autobus i reprezentuje autobusy spalinowe w systemie

Jej pole to

- poziomPaliwa: - float pole to zawiera informacje o poziomie paliwa danego autobusu

Klasa autobusElektryczny dziedziczy po klasie autobus i reprezentuje autobusy elektryczne w systemie

Jej pole to

- poziomNaladowania:float - pole to zawiera informacje o poziomie naładowania danego autobusu

Klasa lokalizacja wykorzystywana przez klasę autobus w polu lokalizacja

Jej pola to:

- szerekosc: float - pole odpowiada szerokości geograficznej danego autobusu
- dlugosc: float - pole odpowiada długości geograficznej danego autobusu

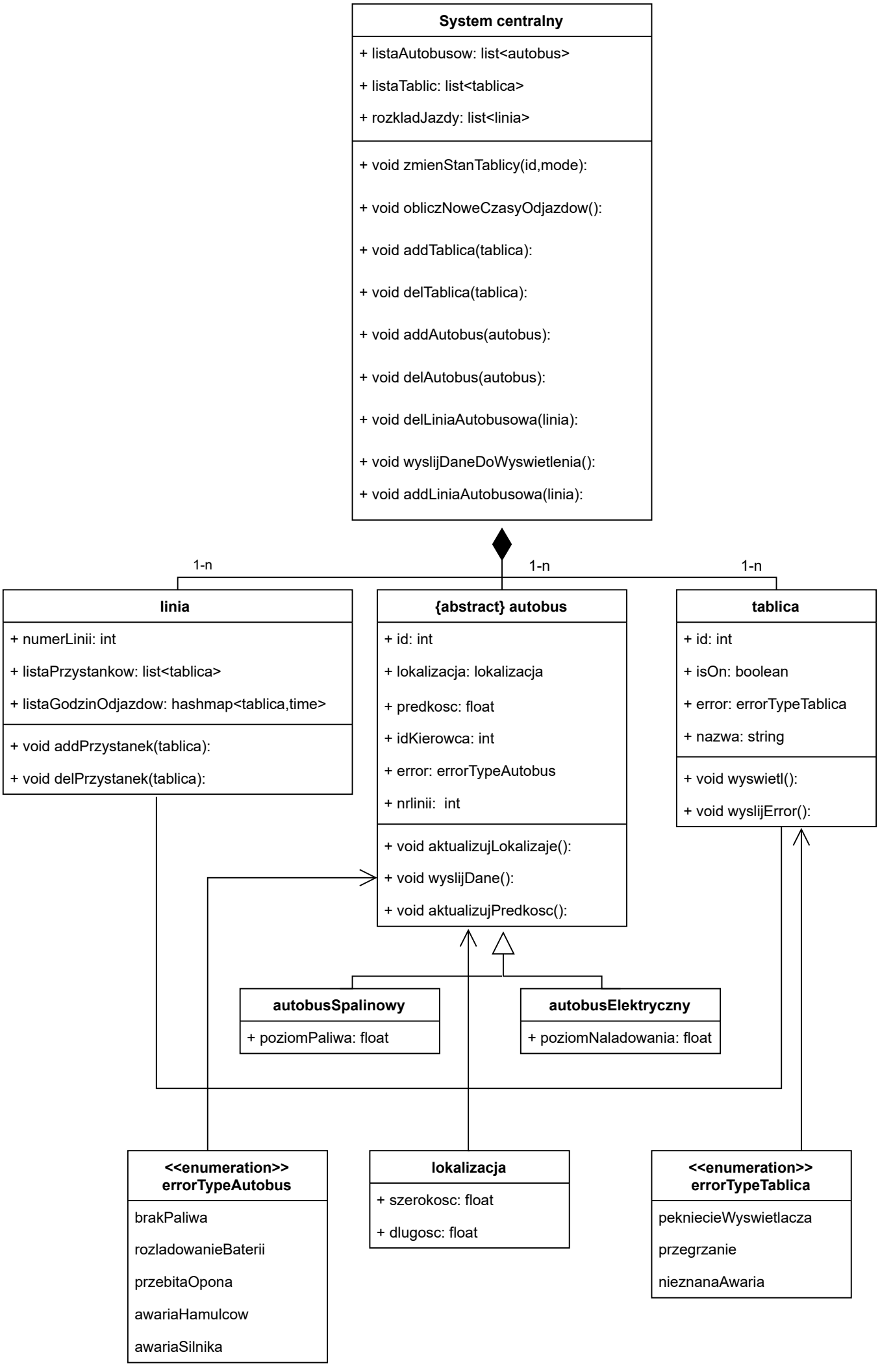
Klasa tablica zagregowana jest przez klasę system centralny odpowiada tablicom na przystankach, a więc i samym przystankom

Jej pola to:

- id: int - pole z id tablicy identyfikuje daną tablicę
- isOn: boolean - pole przechowuje informacje czy dana tablica jest włączona czy wyłączona
- error: errorTypeTablica - pole enum z rodzajami błędów które mogą wystąpić w tablicy: {pekniecieWyswietlacza, przegrzanie, nieznanaAwaria}
- nazwa: string - pole z nazwą przystanku na którym stoi dana tablica

Jej metody to:

- void wyswietl(): - metoda ta wyświetla dane z systemu centralnego o godzinach odjazdów
- void wyslijError(): - metoda ta wysyła komunikat o awarii w razie wystąpienia


System centralny
+ listaAutobusow: list<autobus>
+ listaTablic: list<tablica>
+ rozkladJazdy: list<linia>
+ void zmienStanTablicy(id,mode):
+ void obliczNoweCzasyOdjazdow():
+ void addTablica(tablica):
+ void delTablica(tablica):
+ void addAutobus(autobus):
+ void delAutobus(autobus):
+ void delLiniaAutobusowa(linia):
+ void wyslijDaneDoWyswietlenia():
+ void addLiniaAutobusowa(linia):
1-n
1-n
1-n
linia
+ numerLinii: int
+ listaPrzystankow: list<tablica>
+ listaGodzinOdjazdow: hashmap<tablica,time>
+ void addPrzystanek(tablica):
+ void delPrzystanek(tablica):
{abstract} autobus
+ id: int
+ lokalizacja: lokalizacja
+ predkosc: float
+ idKierowca: int
+ error: errorTypeAutobus
+ nrlinii: int
+ void aktualizujLokalizaje():
+ void wyslijDane():
+ void aktualizujPredkosc():
tablica
+ id: int
+ isOn: boolean
+ error: errorTypeTablica
+ nazwa: string
+ void wyswietl():
+ void wyslijError():
autobusSpalinowy
+ poziomPaliwa: float
autobusElektryczny
+ poziomNaladowania: float
<<enumeration>>
errorTypeAutobus
brakPaliwa
rozladowanieBaterii
przebitaOpona
awariaHamulcow
awariaSilnika
lokalizacja
+ szerokosc: float
+ dlugosc: float
<<enumeration>>
errorTypeTablica
pekniecieWyswietlacza
przegrzanie
nieznanaAwaria

3.2 Diagramy stanów

Autobus

Autobus po dodaniu do systemu (utworzeniu autobusu) znajduje się w zajezdni. Ze stanu zajezdni możliwe jest ponowne usunięcie autobusu z systemu lub przejście w dwa inne stany:

- Zepsucia, gdy następuje awaria autobusu
- W drodze, gdy autobus wyjeżdża w trasę z zajezdni

Gdy autobus trafia do stanu zepsucia, wysyła informacje do systemu centralnego. Stan zepsucia pozwala na powrót autobusu do zajezdni w przypadku naprawy oraz na usunięcie autobusu z systemu.

Autobus będący w drodze cyklicznie aktualizuje informacje o swojej prędkości oraz lokalizacji, aby następnie wysłać te dane do systemu centralnego. Z tego stanu możliwe jest przejście w stan w zajezdni, gdy autobus wraca do zajezdni oraz w stan zepsucia w przypadku nagłej awarii pojazdu.

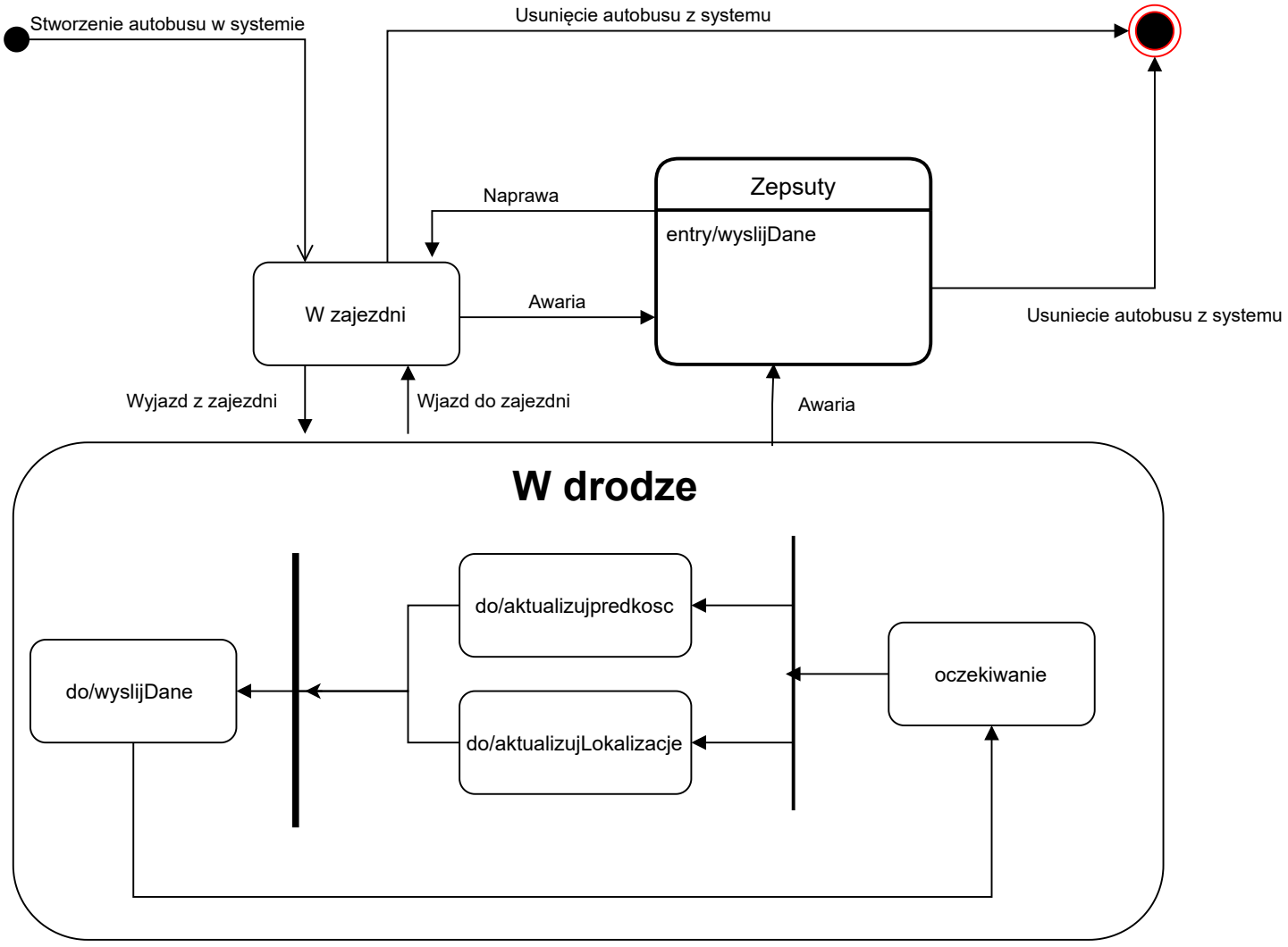
Tablica

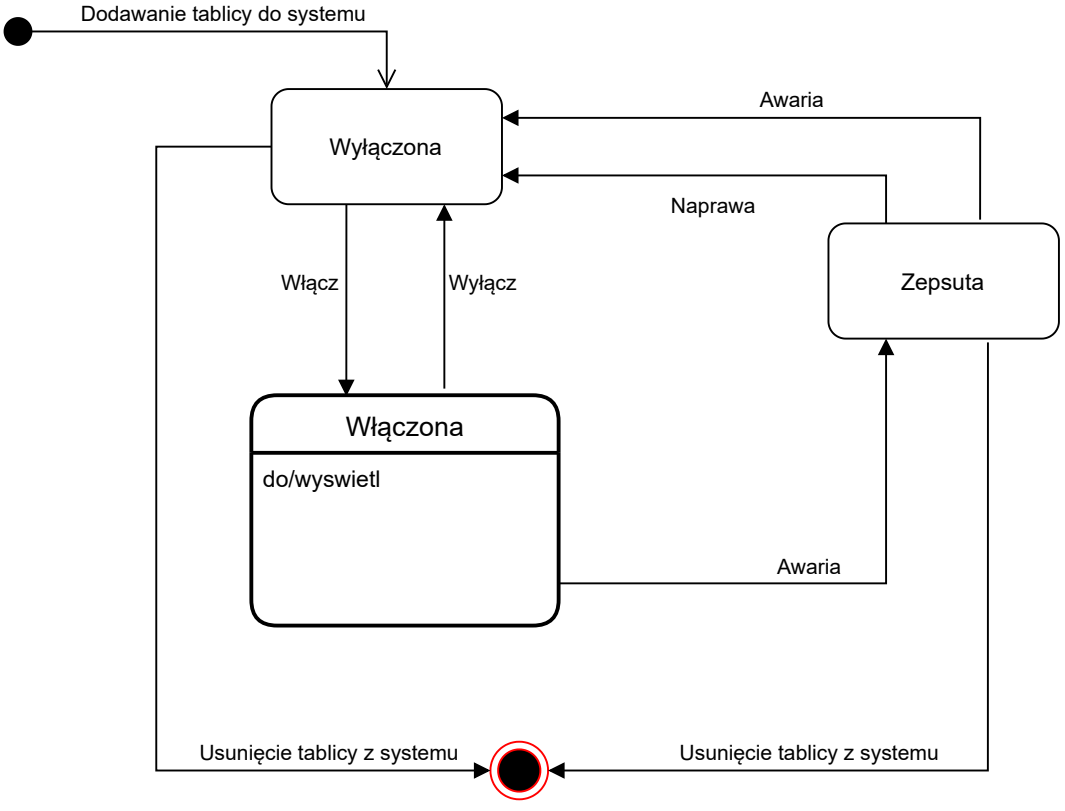
Tablica po dodaniu do systemu (utworzeniu tablicy) jest w stanie wyłączenia. Ze stanu wyłączenia możliwe jest ponowne usunięcie tablicy z systemu lub przejście w dwa inne stany:

- Zepsucia, gdy następuje awaria tablicy
- Włączenia, gdy zostaje uruchomiona

Stan zepsucia pozwala na powrót tablicy do stanu wyłączenia w przypadku naprawy oraz na usunięcie tablicy z systemu.

Tablica pozostaje w stanie włączenia i wyświetla pożądane informacje, gdy nie dzieje się nic niespodziewanego. Z tego stanu możliwe jest wyłączenie tablicy i powrotne przejście w stan wyłączenia oraz zmiana na stan zepsucia, w przypadku awarii.


Stworzenie autobusu w systemie
Usunięcie autobusu z systemu
Naprawa
Zepsuty
entry/wyslijDane
W zajezdni
Awaria
Usuniecie autobusu z systemu
Wyjazd z zajezdni
Wjazd do zajezdni
Awaria
W drodze
do/aktualizujpredkosc
do/wyslijDane
oczekiwanie
do/aktualizujLokalizacje


Dodawanie tablicy do systemu
Wyłączona
Awaria
Naprawa
Włącz
Wyłącz
Zepsuta
Włączona
do/wyswietl
Awaria
Usunięcie tablicy z systemu
Usunięcie tablicy z systemu