Projekt API dla “Systemu kolejek górskich”

Założenia Ogólne

System zarządza kolejkami górskimi oraz przypisanymi do nich wagonami.

System informuje, czy kolejki górskie posiadają wystarczającą ilość wagonów i personelu dla podanej ilości klientów. System działa w dwóch trybach: deweloperskim oraz produkcyjnym, obsługiwanym przez Docker Compose. Dane dotyczące kolejek i wagonów są zapisywane w plikach JSON, a system wykorzystuje Redis do zarządzania danymi w środowisku rozproszonym. Kluczowa jest wydajność i optymalizacja połączenia między węzłami oraz zapisu danych.

API dla Systemu Kolejek Górskich

1. **Rejestracja nowej kolejki górskiej**

    1. **Endpoint:** POST /api/coasters
    2. Dodaje nową kolejkę górską do systemu, uwzględniając dane takie jak liczba personelu, liczba klientów dziennie i długość trasy (w metrach), godziny operacyjne. Dane te są podawane ręcznie i same się nie zmieniają
    3. Przykład danych: `{ liczba_personelu: 16, liczba_klientow: 60000, dl_trasy: 1800, godziny_od: "8:00", godziny_do: "16:00" }`

2. **Rejestracja nowego wagonu**

    1. **Endpoint:** POST /api/coasters/:coasterId/wagons
    2. Dodaje nowy wagon do określonej kolejki górskiej, z danymi dotyczącymi liczby miejsc i prędkości (m/s)
    3. Przykład danych: `{ ilosc_miejsc: 32, predkosc_wagonu: 1.2 }`

3. **Usunięcie wagonu**

    1. **Endpoint:** DELETE /api/coasters/:coasterId/wagons/:wagonId
    2. Usuwa wybrany wagon z danej kolejki górskiej

4. **Zmiana kolejki górskiej**

    1. **Endpoint:** PUT /api/coasters/:coasterId
    2. Aktualizuje dane istniejącej kolejki górskiej, takie jak liczba personelu, liczba klientów dziennie i godziny operacyjne. Długość trasy się nie zmienia

Wersja deweloperska

1. Nasłuchuje na porcie 3050, niedostępna dla osób z zewnątrz
2. Wyświetla wszystkie logi (.log, .error, .warn) wyświetlają się w konsoli
3. Logi zapisywane są do plików zgodnie z typem:
    4. console.error -> error.log

    5. console.warn -> warn.log
    6. console.log -> info.log
7. **Ograniczenia:** Dane na wersji deweloperskiej nie mogą kolidować z danymi na wersji produkcyjnej

Wersja produkcyjna

1. Nasłuchuje na porcie 3051
2. W konsoli wyświetlane są tylko logi typu .warn i .error
3. Logi zapisywane są do plików zgodnie z typem:
    4. console.error -> error.log
    5. console.warn -> warn.log
6. **Ograniczenia:** Dane w wersji produkcyjnej muszą być odseparowane od danych w wersji deweloperskiej

Redis:

1. zakładamy, że Redis jest uruchomiony na zewnętrznej maszynie, do której dostęp mamy poprzez adres IP i port
2. w praktyce oznacza to, że w konfiguracji projektu **nie tworzymy kontenera Redis w Docker Compose**, ale zamiast tego aplikacja powinna połączyć się z Redisem uruchomionym na tej samej maszynie, używając zdefiniowanego adresu IP i portu

Zarządzanie kolejkami i wagonami:

1. system umożliwia rejestrację, edycję i kolejek górskich oraz rejestrację i usuwanie wagonów przez API
2. każda kolejka górska działa w określonych godzinach (od, do)
3. każdy wagon musi wrócić przed końcem czasu działania kolejki górskiej
4. wagony potrzebują 5 minut przerwy, zanim ponownie będą mogły działać po skończonej trasie

Zarządzanie personelem (p):

1. do obsługi każdej kolejki górskiej wymagany jest 1 **p** (np. sprzedawca biletów - nie jest to istotne)
2. do obsługi każdego wagonu dodatkowo wymagane są 2 **p** (np. maszynista i mechanik - nie jest to istotne)
3. jeśli w systemie brakuje odpowiedniej liczby **p** do obsługi kolejki lub wagonu, system informuje o tym oraz wylicza brakującą liczbę **p**
4. jeśli w systemie jest za dużo **p**, system informuje o tym oraz wylicza nadmiarową liczbę **p**

Zarządzanie klientami:

1. system monitoruje liczbę klientów, których kolejka górska powinna obsłużyć w ciągu dnia

2. jeśli kolejka nie będzie w stanie obsłużyć wszystkich klientów w ciągu dnia, system informuje o tym i wylicza, ile brakuje wagonów oraz personelu
3. jeśli kolejka górska ma więcej mocy przerobowych niż wymagane, tj. obsłuży ponad dwukrotnie więcej klientów niż zaplanowano, system informuje o nadmiarowej liczbie wagonów i personelu

Rozproszony system zarządzania:

1. każda kolejka górska powinna móc działać autonomicznie - nawet gdy nie jest podłączona do sieci, lub gdy jest jedyną kolejką podłączoną do systemu
2. gdy więcej niż jedna kolejka jest podłączona do sieci, jeden z systemów przejmuje rolę centralnego węzła, zarządzającego wszystkimi kolejkami
3. system zarządza synchronizacją danych między wszystkimi węzłami, które są podłączone do sieci

Synchronizacja danych:

1. synchronizacja danych między węzłami działa asynchronicznie, co oznacza, że operacje w systemie nie są blokowane w oczekiwaniu na pełną synchronizację danych między wszystkimi kolejkami
2. zmiany wprowadzone w jednym węźle (np. aktualizacja kolejki A1 z komputera obsługującego kolejkę A3) są natychmiast wprowadzane lokalnie, a synchronizacja z innymi węzłami następuje w tle

Statystyki i monitorowanie (konsola):

1. system w czasie rzeczywistym wyświetla w konsoli statystyki dla każdej kolejki górskiej
2. statystyki obejmują liczbę dostępnych kolejek, wagonów, personelu, klientów
3. statystyki i potencjalne problemy są aktualizowane na bieżąco w oparciu o zmieniające się dane dotyczące kolejek, wagonów, personelu i klientów

```
[Godzina 14:27]

[Kolejka A1]

1. Godziny działania: 08:00 - 18:00
2. Liczba wagonów: 5/5
3. Dostępny personel: 12/12
4. Klienci dziennie: 200
5. Status: OK

[Kolejka A2]

1. Godziny działania: 09:00 - 17:00
2. Liczba wagonów: 4/6
3. Dostępny personel: 8/10
4. Klienci dziennie: 150
```

5. Problem: Brakuje 2 pracowników, brak 2 wagonów