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 na serwerze Redis.

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. Niedostępna dla osób z zewnątrz
2. W trybie deweloperskim konfiguracja logów powinna rejestrować wszystkie typy logów.
3. **Ograniczenia:** Dane na wersji deweloperskiej nie mogą kolidować z danymi na wersji produkcyjnej

Wersja produkcyjna

1. W trybie produkcyjnym tylko logi typu `warning` oraz `error` powinny być rejestrowane.
2. **Ograniczenia:** Dane w wersji produkcyjnej muszą być odseparowane od danych w wersji deweloperskiej

Redis

1. Aplikacja powinna wykorzystywać Redis jako magazyn danych.

Zarządzanie kolejkami i wagonami:

1. system umożliwia rejestrację i edycję 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

Statystyki i monitorowanie (konsola):

1. Stwórz asynchroniczny serwis monitorujący w PHP CLI bez użycia blokujących pętli, który wyświetla w czasie rzeczywistym statystyki dotyczące systemu kolejek górskich.
2. Możesz użyć rozszerzenia react, swoole lub innej dostępnej technologii do zapewnienia asynchroniczności

3. Niewymagane, ale preferowane jest wykorzystanie asynchronicznego klienta Redis, np. https://github.com/clue/reactphp-redis dla ReactPHP
4. statystyki obejmują liczbę dostępnych kolejek, wagonów, personelu, klientów
5. statystyki i potencjalne problemy są aktualizowane na bieżąco w oparciu o zmieniające się dane dotyczące kolejek, wagonów, personelu i klientów
6. Po wykryciu problemu, zapisz informację w pliku log jako symulację wysłania powiadomienia. Powinno to obejmować szczegóły problemu i czas jego wykrycia, np.
7. `[2024-11-29 00:12:30] Kolejka A1 - Problem: Brakuje 2 pracowników, brak 2 wagonów`

`[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`

Wymagania techniczne :

1. Użyj PHP w wersji 8.0 lub nowszej.
2. Użycie frameworka Codeigniter 4
3. Przygotuj aplikację do uruchomienia bazując na PHP-FPM i wsparciu serwera nginx.