Chariot

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej
Informatyka, 3. rok, PWiR grupa 5a

Kacper Tonia Sławomir Kalandyk

22.01.2020

1 Cel programu

Zadaniem programu jest rozsyłanie wozów strażackich do incydentów zgłaszanych za pośrednictwem strony internetowej. Pula wozów jest ograniczona, zaś same pojazdy po powrocie z akcji przez pewien czas pozostają niedostępne (czas na uzupełnienie zapasów, naprawy itd.).

2 Moduły

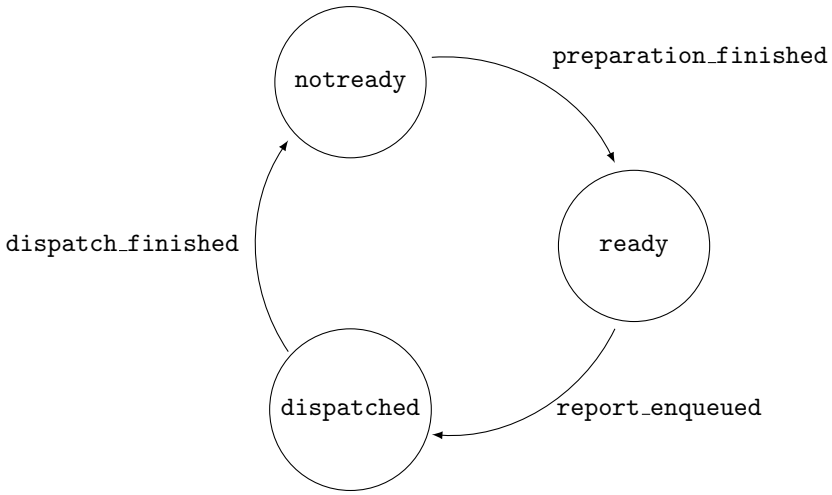
- **main** – uruchamia serwer i pełni rolę klienta.
- **central** – reprezentuje centralę powiadamiania ratunkowego, której można zgłosić żądanie pomocy straży pożarnej. Moduł ten implementuje zachowanie `gen_server`.
- **domain** – definiuje rekord `state` używany przez serwer centrali, a także tzw. *invoke functions* – funkcje realizujące podstawową logikę aplikacji (zapisywanie zgłoszeń, wysyłanie wozów strażackich itd.).
- **firetrucks** – definiuje rekord `firetruck` reprezentujący wóz strażacki, a także podstawowe operacje z tym rekordem związane.
- **server** – serwer HTTP zaimplementowany za pomocą `httpd` zawierającego się w pakiecie `inets`; są w nim definiowane funkcje odpowiadające na zapytania GET/POST

2.1 Serwer centrali

Operacje realizowane przez serwer centrali możemy podzielić na 3 kategorie:

- operacje opisane przez `handle_call` są synchroniczne, zwracają pewną wartość nie modyfikując stanu serwera.
  - `get_vehicles` pobiera aktualną listę pojazdów wraz z ich stanem
- operacje opisane przez `handle_cast` są asynchroniczne, nie blokują klienta. Nie zwracają istotnych wartości, ale mogą modyfikować stan serwera.
  - `report_incident` zgłasza incydent wymagający przyjazdu straży pożarnej
- operacje opisane przez `handle_info` sygnalizują serwerowi zajście jakiegoś zdarzenia. Nie powinny być wywoływane z zewnątrz.
  - `report_enqueued` sygnalizuje, że w kolejce zgłoszeń znajdują się oczekujące zgłoszenia

– `dispatch_finished` sygnalizuje, że pojazd wrócił z akcji strażackiej
– `preparation_finished` sygnalizuje, że pojazd jest gotowy do następnej akcji strażackiej



Rysunek 1: Cykl pracy wozu strażackiego

3 Główne komponenty programu i ich zadania

Main:

1. Tworzy pulę pojazdów
2. Uruchamia serwer centrali
3. Uruchomia serwer HTTP

Serwer centrali:

1. Zarządza pulą pojazdów
2. Informuje serwer HTTP o stanie pojazdów

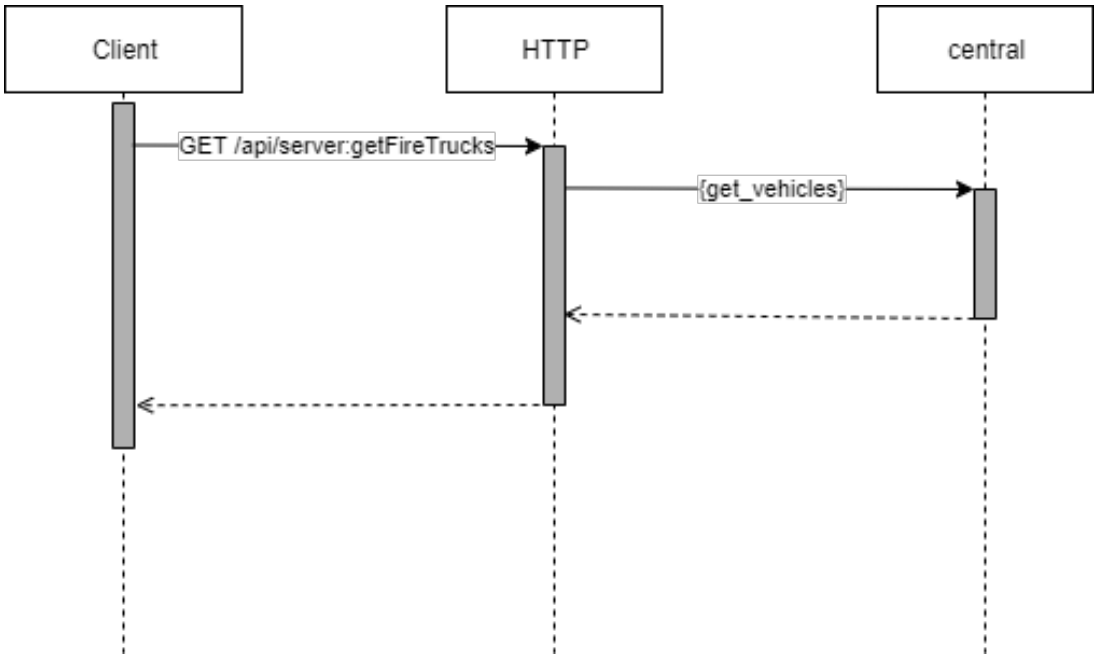
Serwer HTTP:

1. Po uruchomieniu, oczekuje na zapytania GET/POST
2. Przy zapytaniu GET: zwraca zserializowane pojazdy wraz z ich obecnym stanem
3. Przy zapytaniu POST: zleca zmianę stanu jednego z pojazdów na `dispatched`

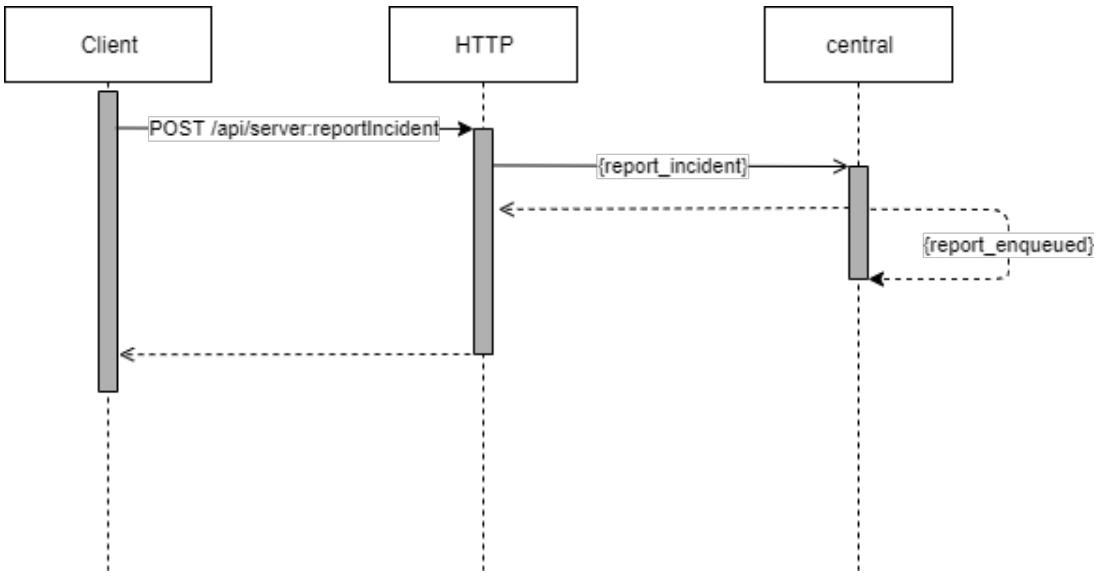
Strona WWW:

1. Udostępnia pole tekstowe wraz z przyciskiem "Report incident", który po naciśnięciu wykonuje zapytanie POST
2. Pokazuje obecne stany pojazdów w puli:
   - `ready` – barwa niebieska
   - `dispatched` – barwa pomarańczowa
   - `notready` – barwa żółta
3. Wyświetla logi - zserializowane rekordy pojazdów składające się z pól `id`, `status` oraz `waiting_since`
4. Co dany okres czasu (domyślnie 1s) odpytuje serwer o aktualny stan pojazdów

4 Diagramy sekwencji



Rysunek 2: Przebieg zapytania o stan pojazdów



Rysunek 3: Przebieg zgłaszania wypadku

5 Pakiety zewnętrzne

- Jsone - pakiet służący nam do serializowania rekordów do formatu JSON, posiada licencję MIT

6 Specyficzne rozwiązania

6.1 Problem "głodzenia" pojazdów

Domyślnie pojazdem wysyłanym do wypadku byłby pojazd w stanie `ready` o najmniejszym identyfikatorze spośród takich pojazdów. Potencjalnie mogło to doprowadzić do sytuacji, gdy część pojazdów nigdy nie zostałaby wykorzystana. Przykładowo:

- Posiadamy 6 pojazdów
- Pojazd 1 zmienia stan na `dispatched`, zaraz po nim przychodzi kolejna informacja o wypadku i pojazd 2 zmienia stan na `dispatched`
- Pojazd 1 wraca, zmienia stan na `notready` i zaraz później na `ready`
- Przychodzi informacja o wypadku: pojazd 1 zmienia stan na `dispatched`

Ostatecznie, gdyby taka sytuacja powtarzała się cały czas, pojazdy o ID 3, 4, 5, 6 potencjalnie mogłyby nie być w ogóle wykorzystywane.

Problem został rozwiązany poprzez dodanie do rekordu pojazdu parametru `waiting_since`, który zapisuje czas, w którym pojazd ostatni raz zmienił stan na `ready`. Do wypadku wysyłany jest pojazd, który najdłużej nie był wykorzystywany - różnica pomiędzy czasem obecnym, a zapisanym jest największa spośród pojazdów.

7 Instrukcja obsługi

1. W katalogu projektu wywołaj "make compile"
2. Wykonaj "cd beam", a potem "escript main.beam" (przejście do katalogu beam i uruchomienie programu)
3. Wejdź w przeglądarce internetowej na adres localhost:8080

8 Ograniczenia programu

- Liczba wozów strażackich jest określona przy starcie programu, nie można jej zmienić po starcie programu
- Pojazdy straży nie są w jakikolwiek sposób rozróżnialne od siebie
- Akcje, na które wysyłane są wozy strażackie są imitowane funkcją `timer:sleep()`. W stanie `dispatched` pojazd przebywa od 5 do 15 sekund, a w stanie `notready` od 2 do 12 sekund

9 Możliwe rozszerzenia programu

Jednym z możliwych rozszerzeń jest dodanie różnych typów wypadków i możliwość wyboru wypadku. Nadałoby to sens rozróżnieniu pojazdów ze względu na wyposażenie - to, czy pojazd mógłby wyruszyć do akcji zależałoby od tego, czy jest odpowiednio wyposażony, aby móc efektywnie nieść pomoc na miejscu zdarzenia.