Notificatie API's

Joost Farla / 15 oktober 2018

REST

- Met een REST API bevraag je de huidige state van een (set) resource(s)
  - Voorbeeld: wie is de eigenaar van perceel X?
  - `GET /percelen/X`
- Het resultaat is een “snapshot” van de state
- Het resultaat kan morgen anders zijn
- Wat als de API client een proces wil starten als de eigenaar van perceel X wijzigt?

Wat zijn notificaties?

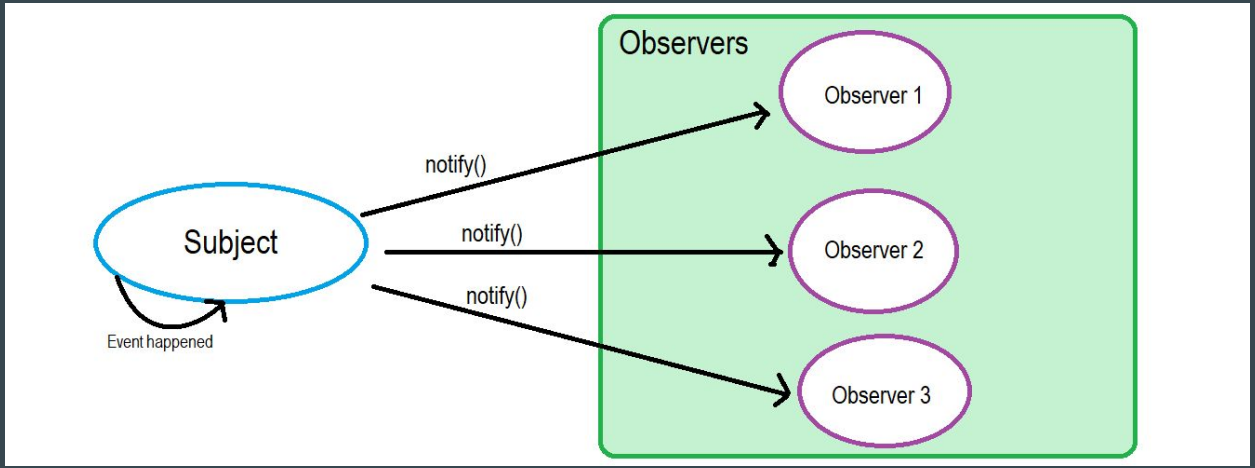
- Een event (gebeurtenis) is een verandering van state A naar state B
  - Voorbeeld: eigenaar van een perceel is gewijzigd
- Een event is altijd verleden tijd
- Een notificatie stelt andere systemen ervan op de hoogte dat een event heeft plaatsgevonden

Design patterns

...

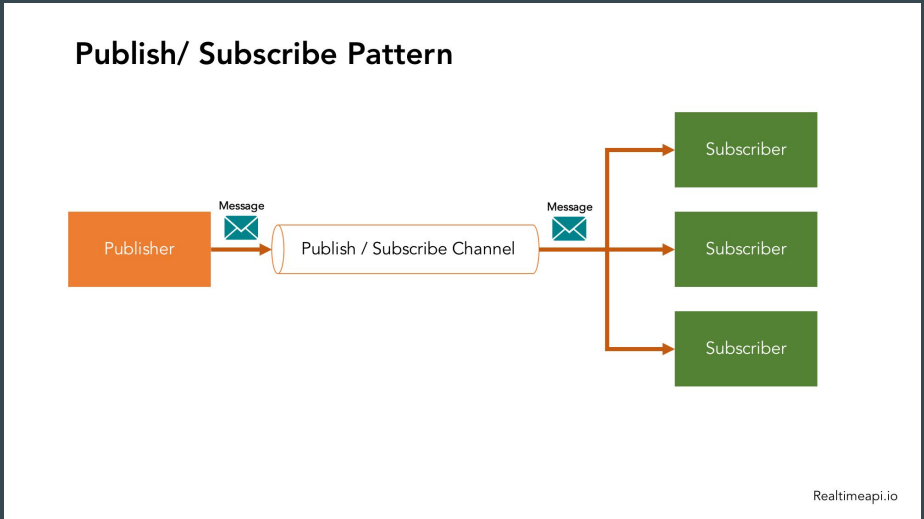
Observer Design Pattern

- Systeem X (subject) verzendt notificaties naar andere systemen (observers)
- Systeem X houdt zelf een lijst van observers bij
- Systemen zijn hierdoor “coupled”



Pub/Sub Design Pattern

- Publisher verzendt notificaties naar een broker
- Broker distribueert notificaties naar subscribers
- Systemen zijn “decoupled” en daardoor schaalbaarder



Pub/Sub Design Pattern

- Publisher heeft geen kennis van subscribers
- Subscriber kan filteren op de events waar hij in is geïnteresseerd
- Topic-based filtering
  - Events worden logisch gecategoriseerd in topics (channels)
  - Publisher verstuurt ieder event naar 1 of meerdere topics
  - Subscriber kan luisteren naar 1 of meerdere topics
- Content-based filtering
  - Publisher verstuurt alle events zonder categorisering
  - Client bepaalt zelf de constraints van de events waarop hij wil filteren

Voorbeelden

...

Voorbeeld 1: REST endpoint

- Een REST endpoint die alle notificaties terug geeft op chronologische volgorde:
  - `GET /notifications?since=1539179994`
- Pull-based, dus geen Pub/Sub
- Uitdagingen
  - Server moet een logboek van alle notificaties bijhouden
  - Client moet periodiek “pollen” of er nieuwe notificaties zijn
    - Inefficiënt bij lage update-frequentie
  - Client moet bijhouden welke notificaties er al zijn verwerkt en welke niet
  - Informatie is altijd “oud”
    - Alternatief: long polling

Voorbeeld 2: Webhooks

- Met een webhook kan een client subscriben op bepaalde events d.m.v. het registreren van een callback URL bij de service
  - ```
    POST /subscriptions
    Content-Type: application/json
    { "types" : ["perceel"], "url" : "https://example-client.org/notify" }
    ```
- De callback URL wordt door de service aangeroepen voor ieder event
- Event (meta)data wordt als payload meegestuurd
- Voordelen
  - Push based, dus realtime
  - Standaard HTTP/REST
- Uitdagingen
  - Beschikbaarheid webhook URL
  - Flood control / buffering
    ```

Voorbeeld 3: Notification Service

- Lichtgewichte “always-on” verbinding met notification service
- In alle soorten en maten: Amazon SNS, Google Cloud Pub/Sub, Azure Event Grid, RabbitMQ, ActiveMQ, Kafka, Pulsar
- Voordelen
    - Push based, dus realtime
    - Lichtgewicht
    - Veel standaard oplossingen beschikbaar
- Uitdagingen
    - Beschikbaarheid subscribers
    - Flood control / buffering
    - Message retention
    - Message ordering