Reactive REST webservices

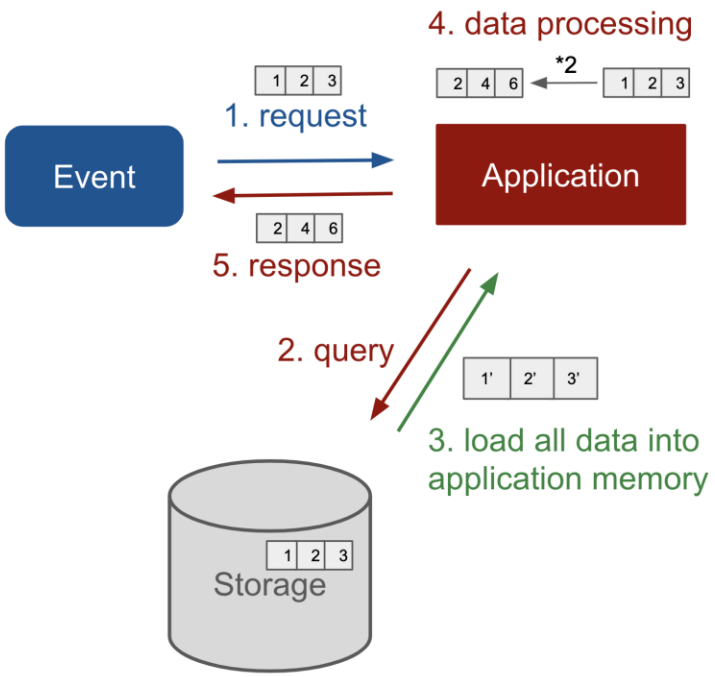
Veerle Ongenae

Overzicht

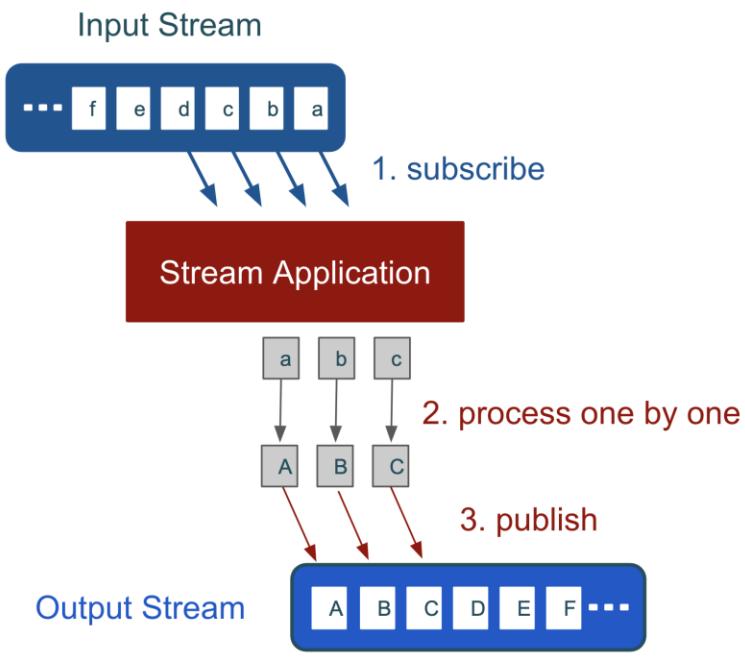
- Herhaling reactive programming

Waarom Reactive Programming ?

UNIVERSITEIT GENT



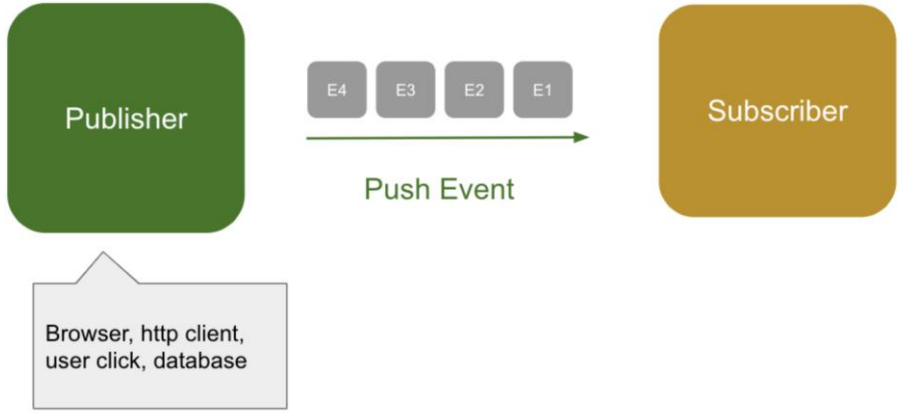
Traditional Data Processing

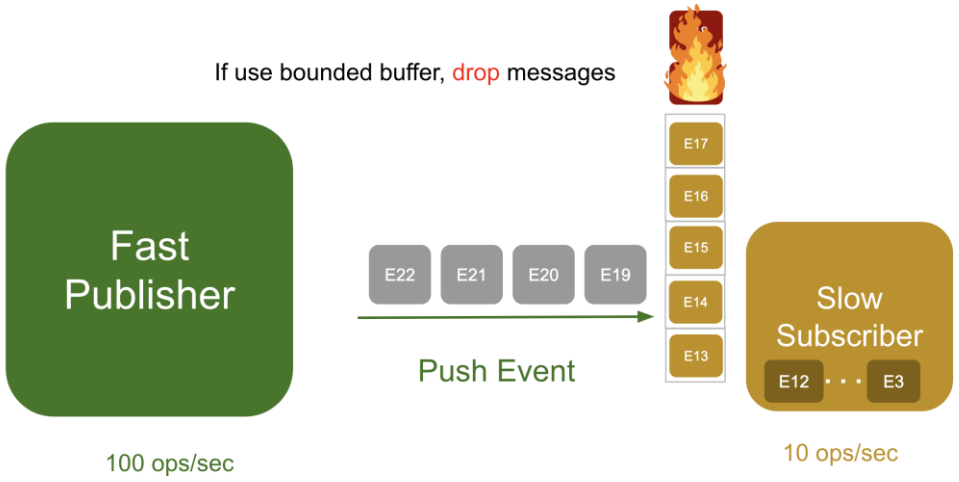


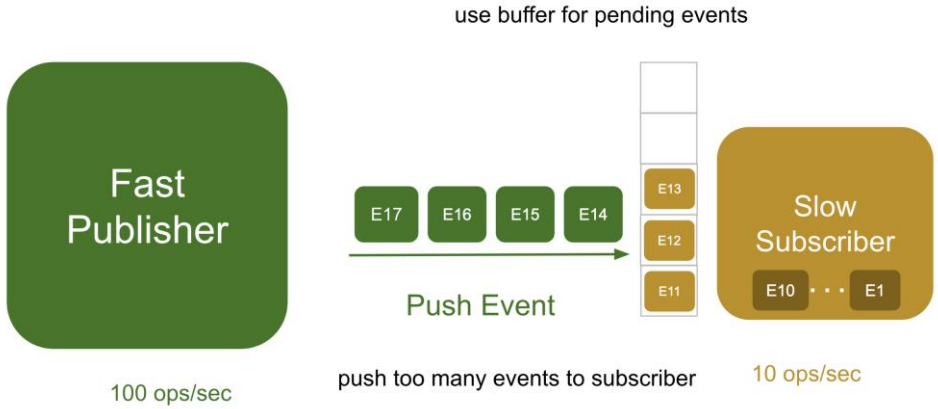
Stream Processing

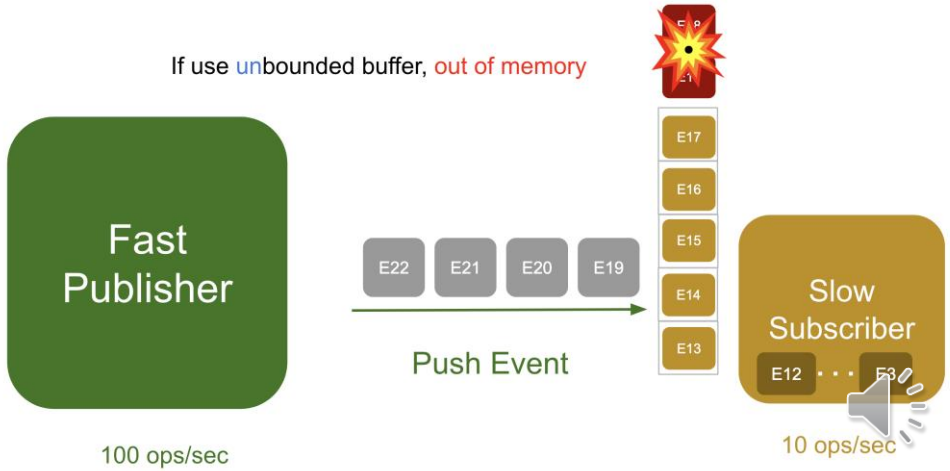
bron: https://engineering.linecorp.com/en/blog/reactive-streams-armeria-1/

Observer - push



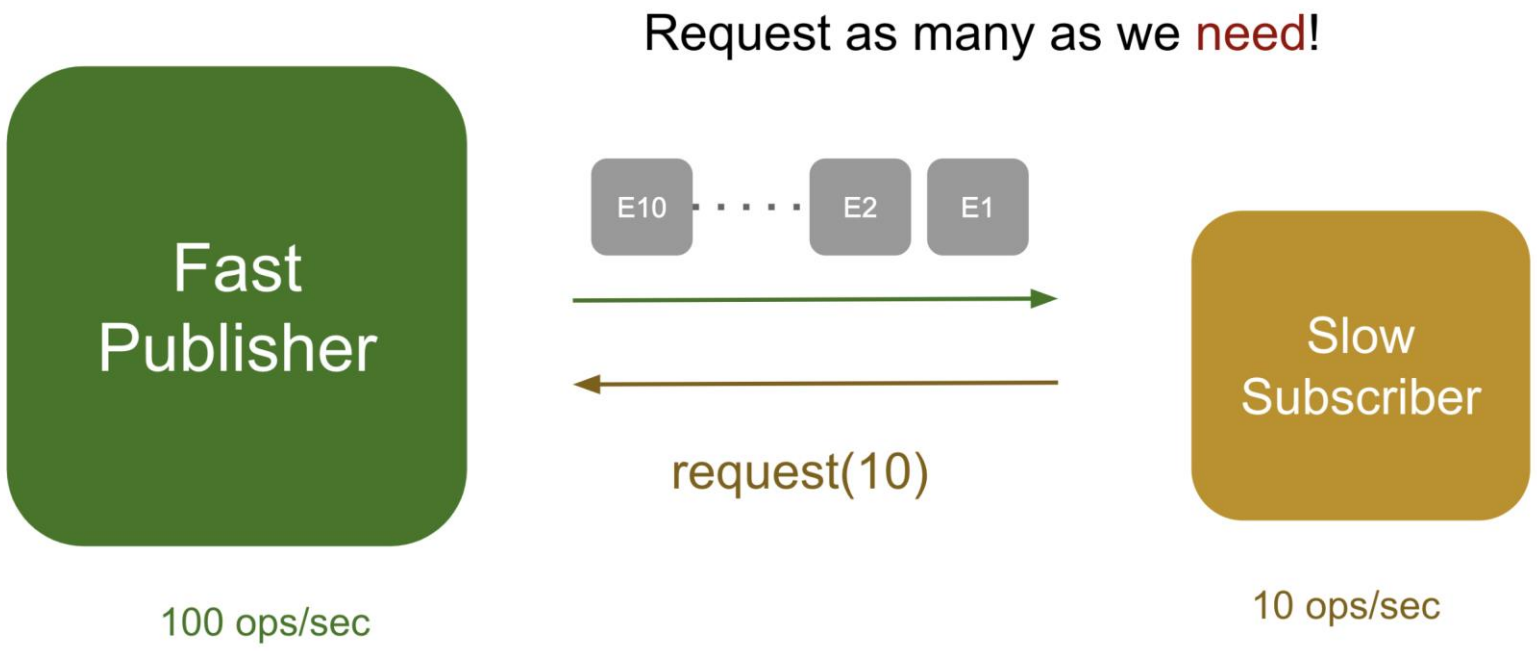






bron: https://engineering.linecorp.com/en/blog/reactive-streams-armeria-1/

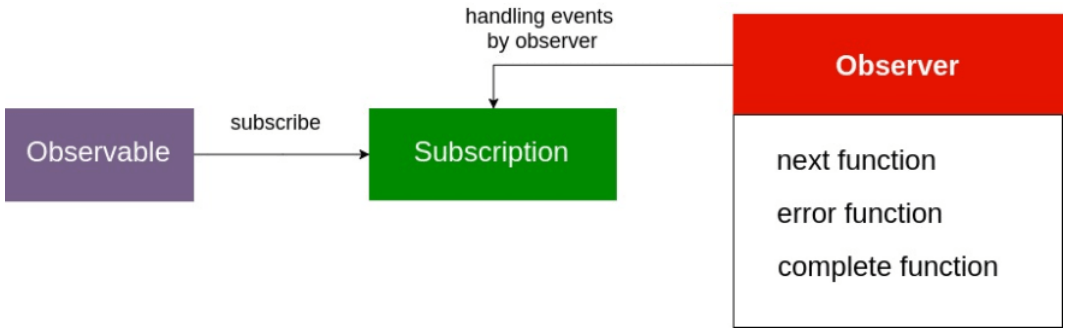
Observer - pull

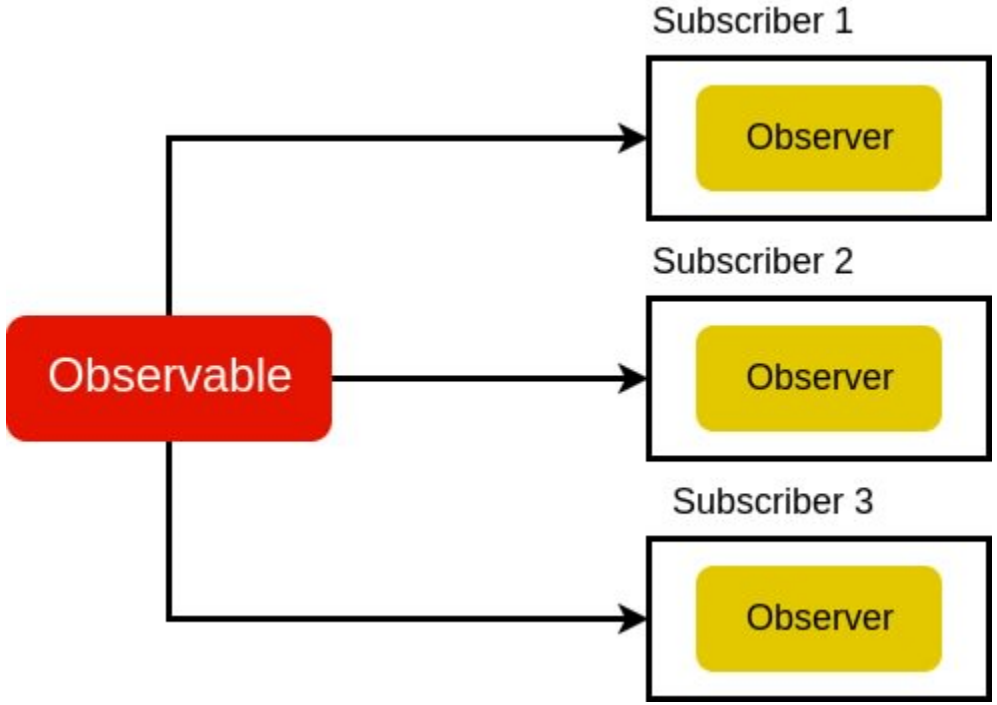


Backpressure (tegendruk) in softwaresystemen is het vermogen om de communicatie te overbelasten. Met andere woorden, zenders van informatie overstelpen consumenten met gegevens die zij niet kunnen verwerken. Men past deze term ook toe als het mechanisme om dit te controleren en af te handelen.

bron: https://engineering.linecorp.com/en/blog/reactive-streams-armeria-1/

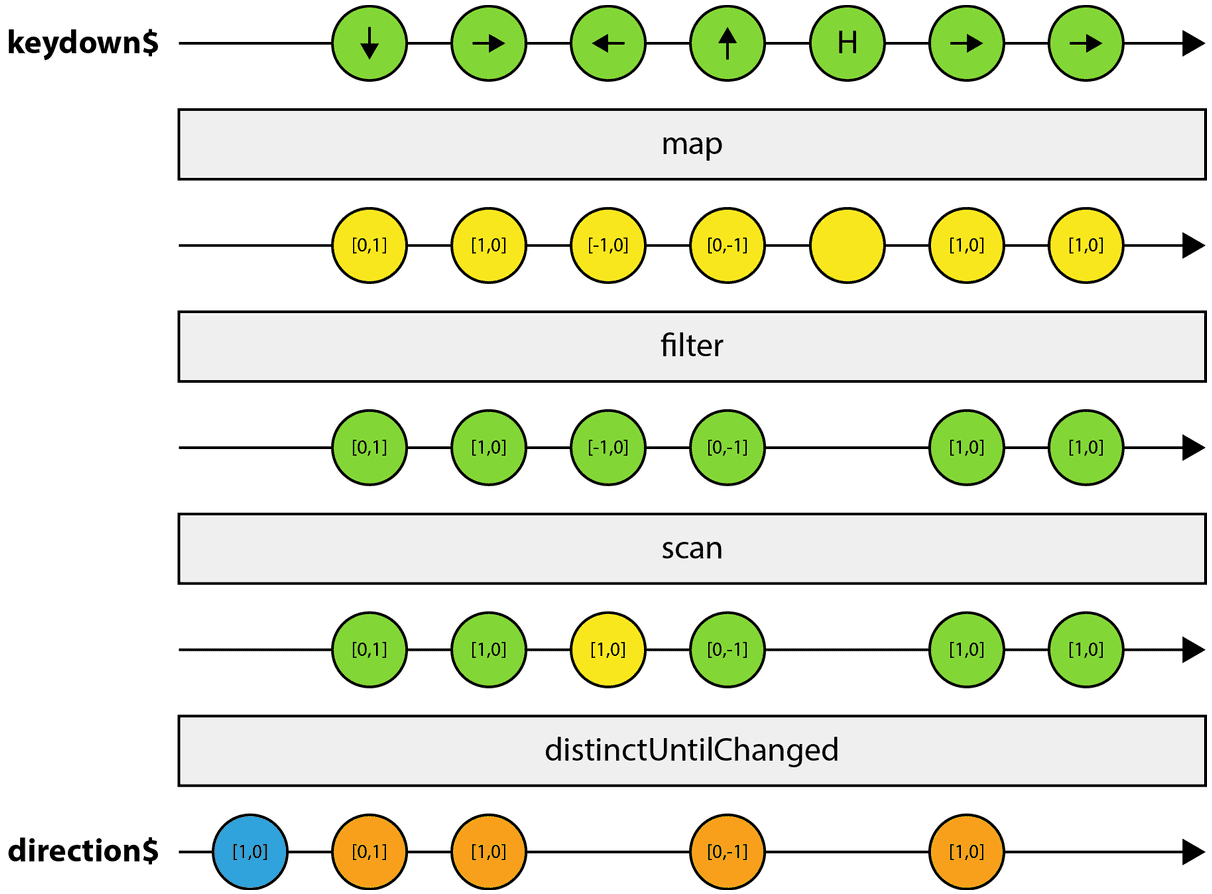
Reactive Programming - principe





bron: https://dev.to/sagar/reactive-programming-in-javascript-with-rxjs-4jom

Reactive Programming - pipeline

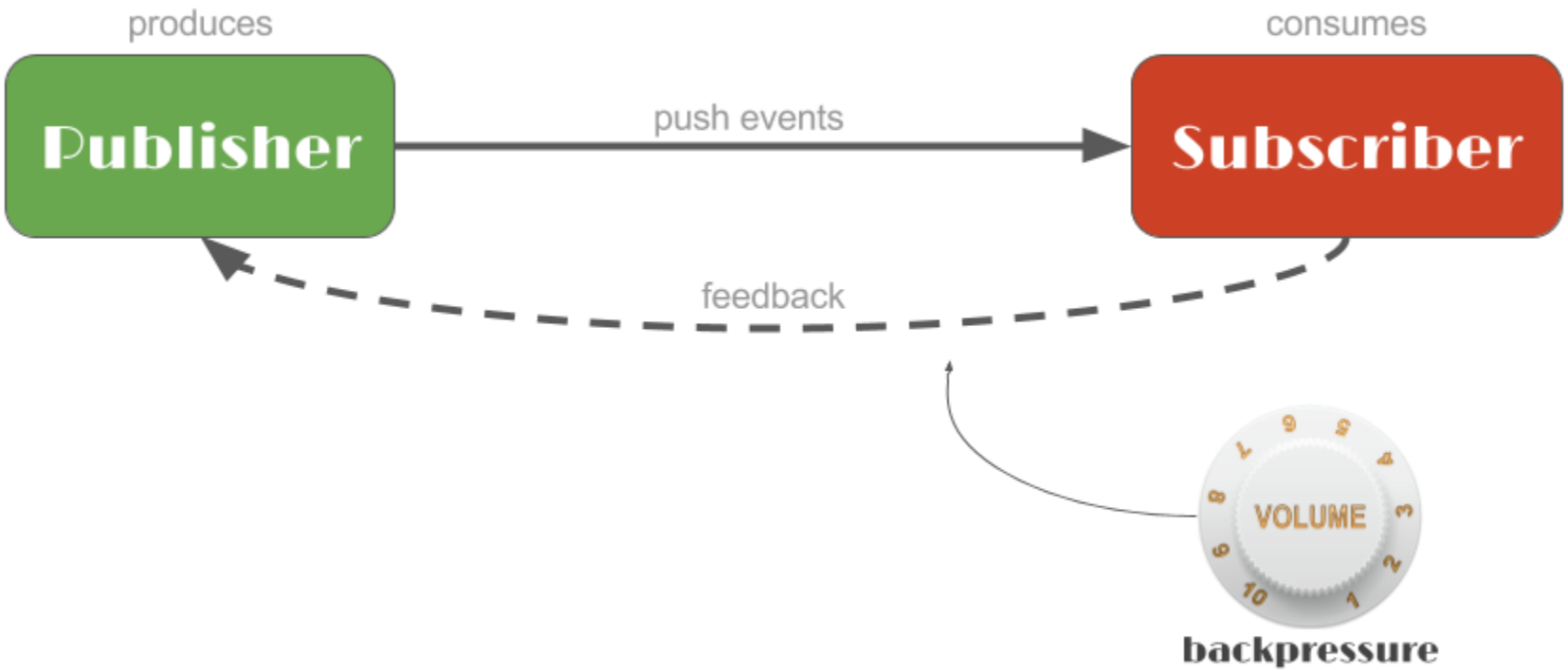


https://blog.thoughtram.io/rxjs/2017/08/24/taming-snakes-with-reactive-streams.html

Overzicht

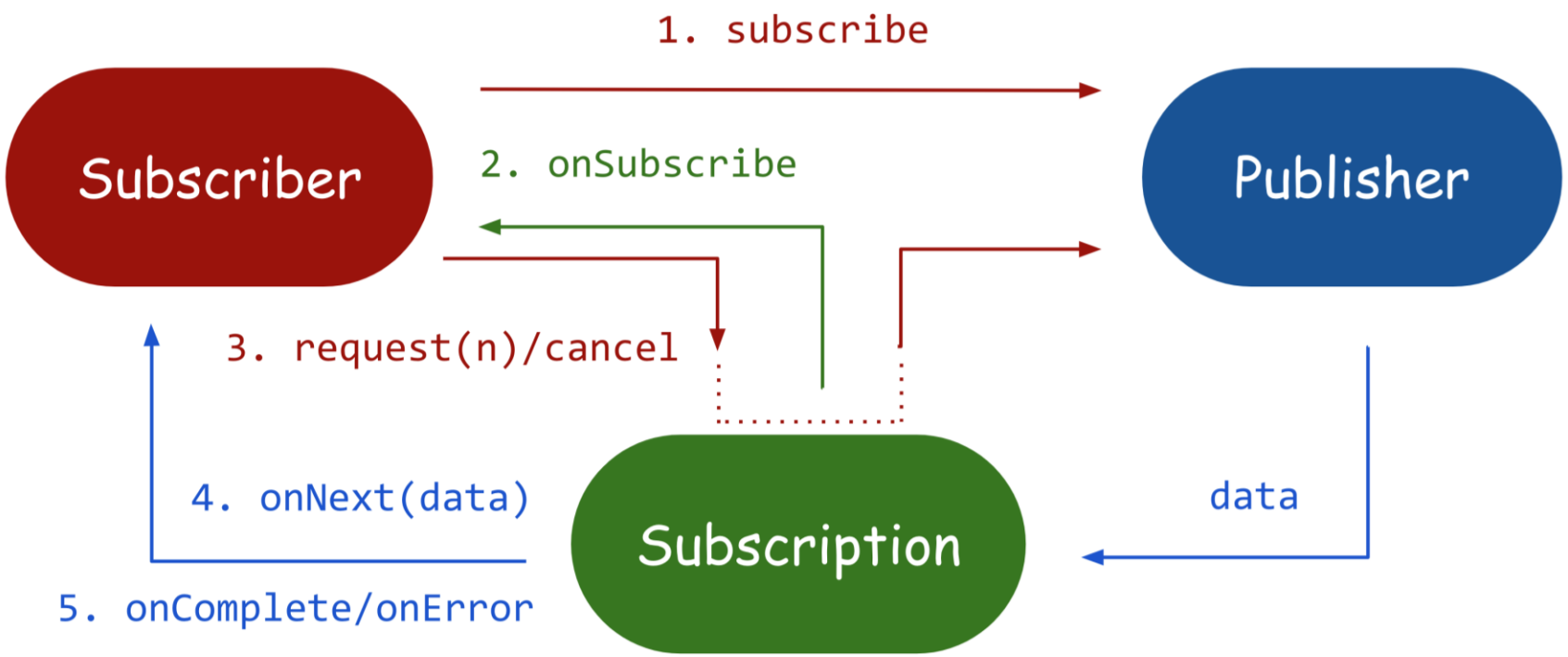
- Herhaling reactive programming
- Reactive Streams

Reactive Streams - concept



bron: https://tech.io/playgrounds/929/reactive-programming-with-reactor-3/Intro

Reactive Streams API (Java)

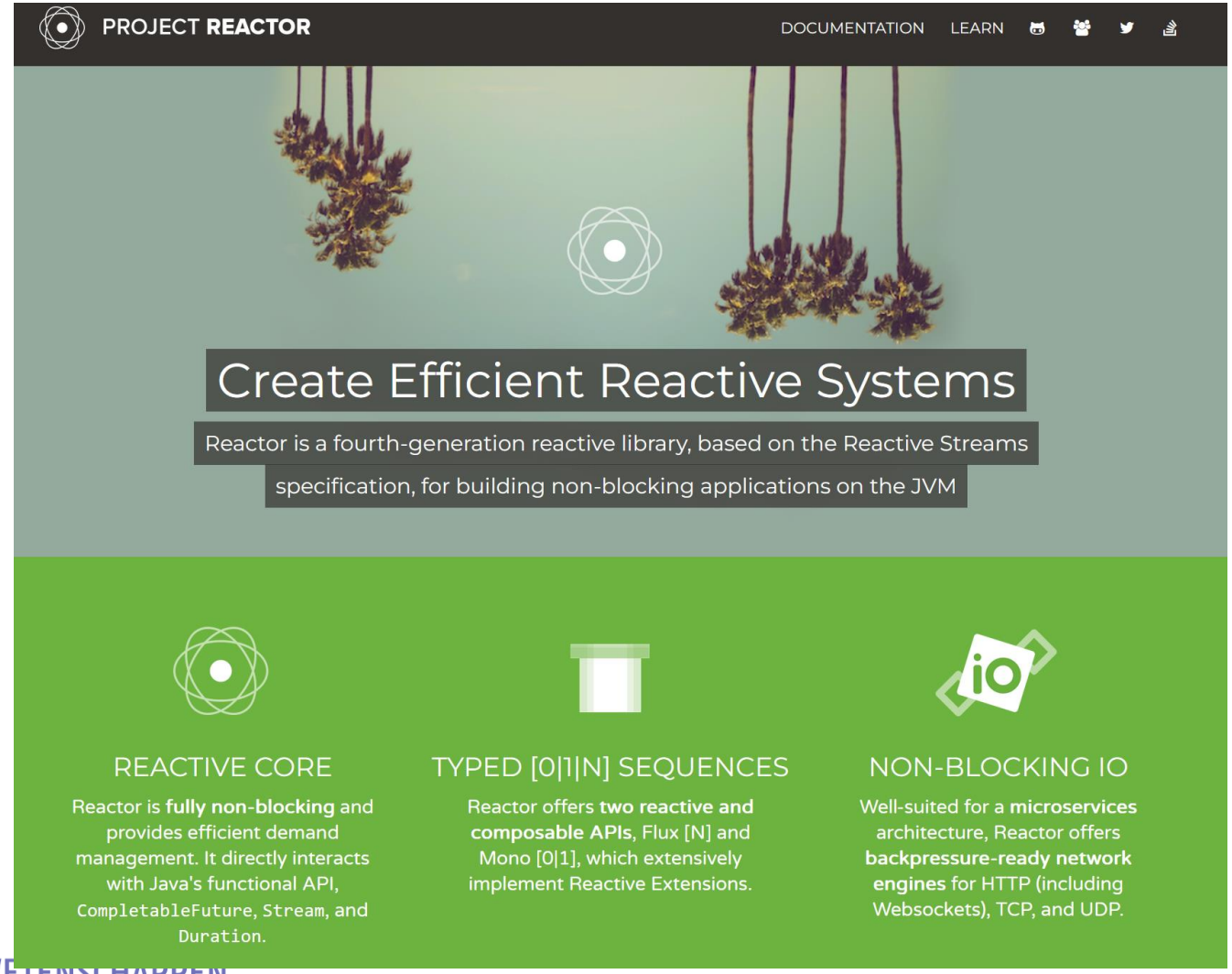


bron: https://engineering.linecorp.com/en/blog/reactive-streams-armeria-1/

Overzicht

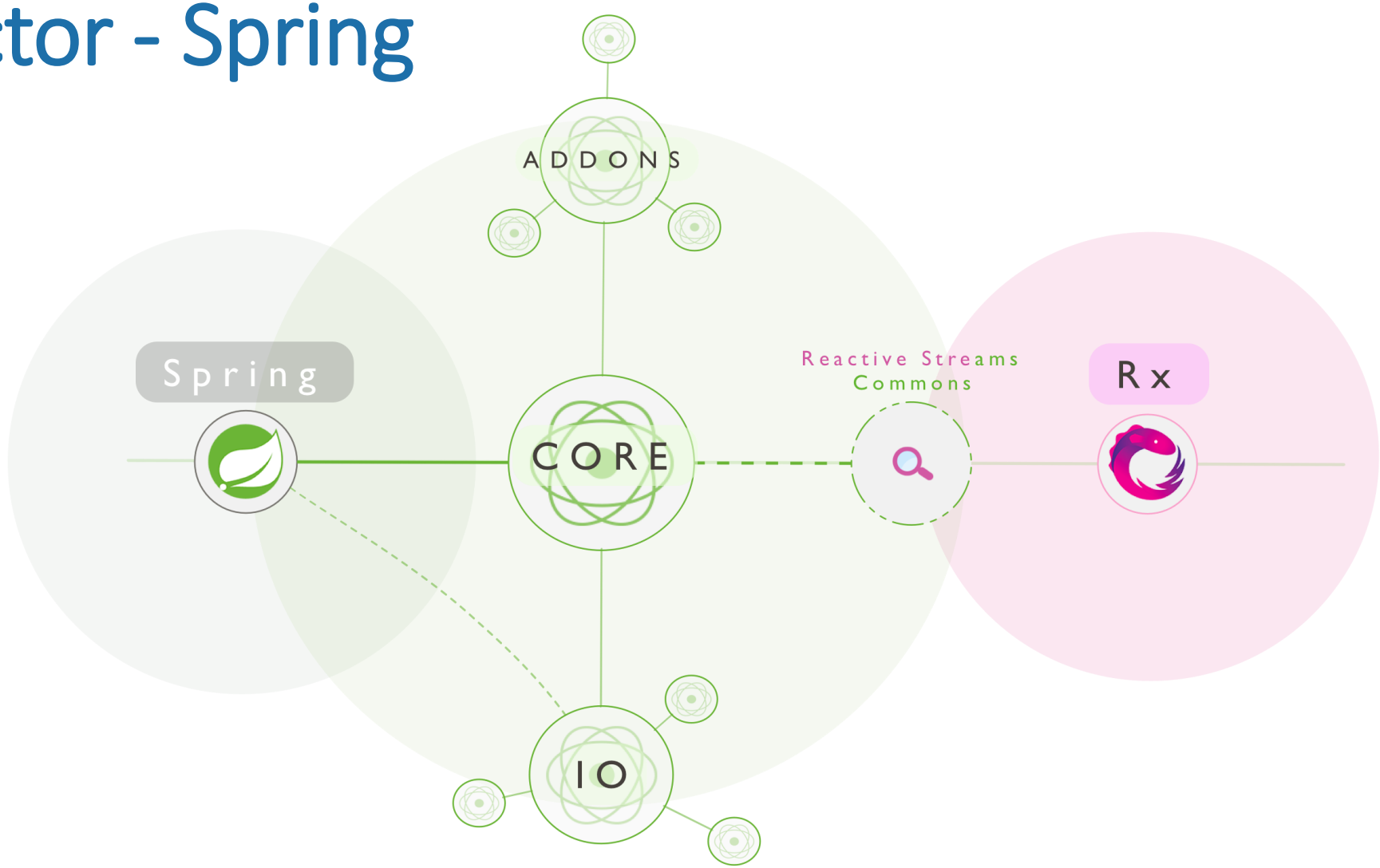
- Herhaling reactive programming
- Reactive Streams
- Reactor

Reactor



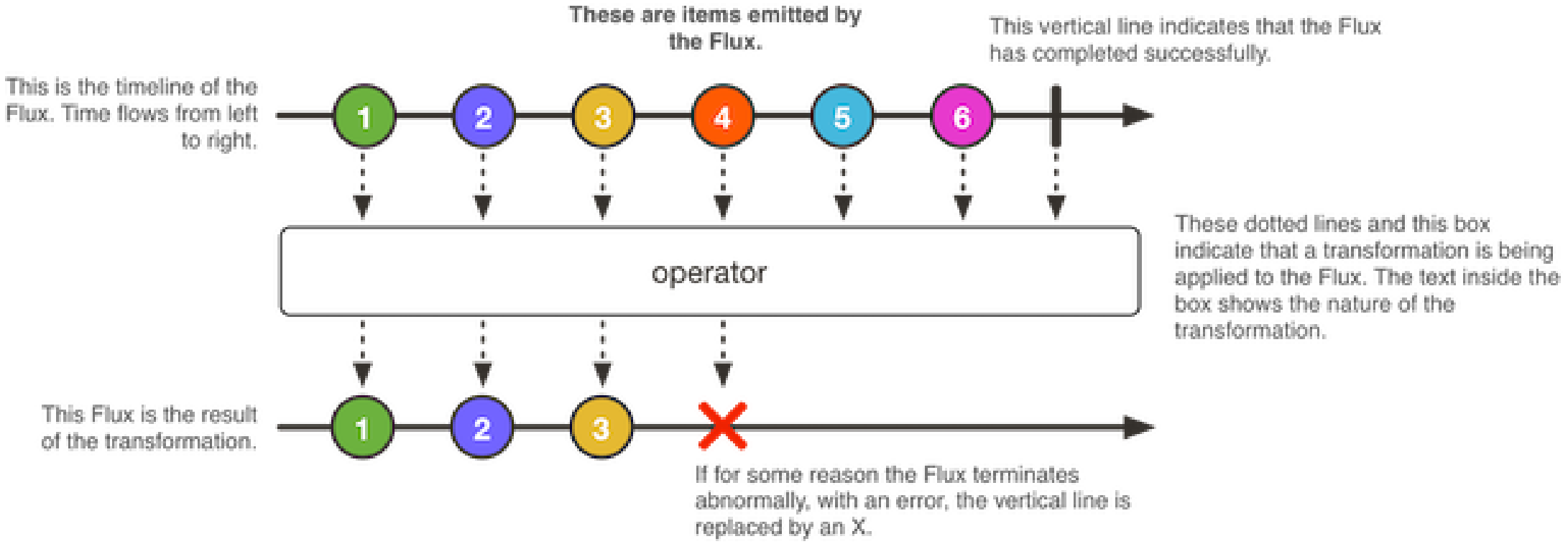
https://projectreactor.io/

Reactor - Spring



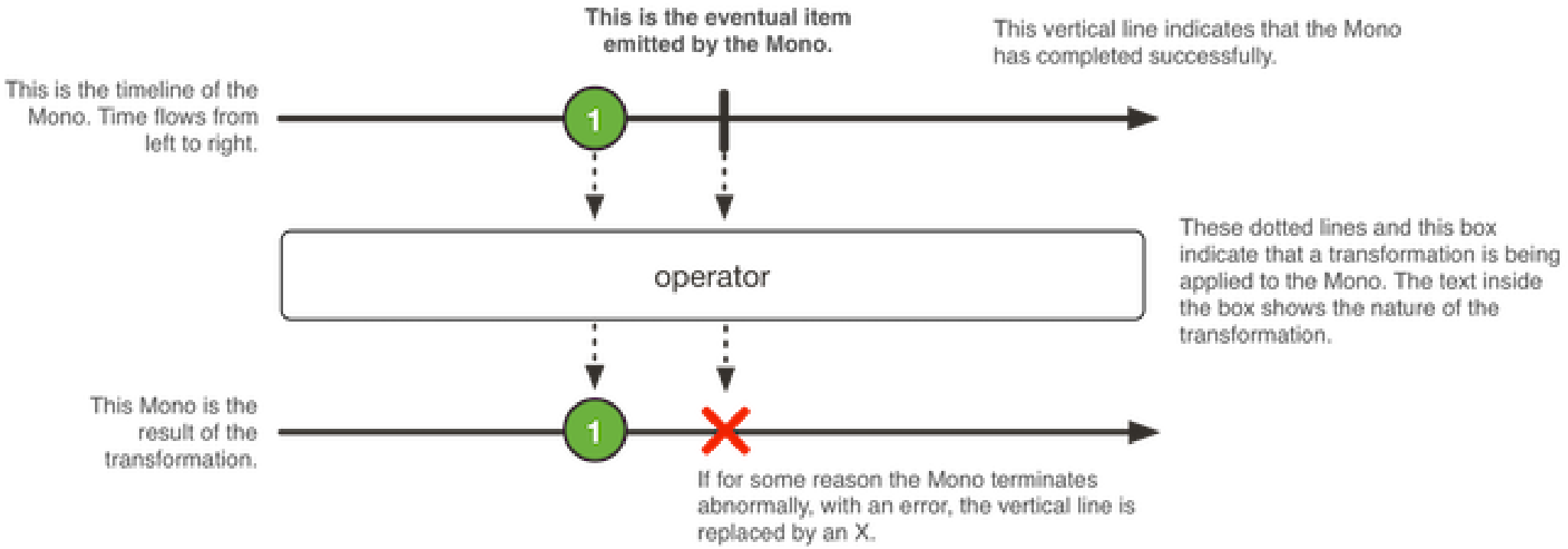
bron: https://spring.io/blog/2016/03/11/reactor-core-3-0-becomes-a-unified-reactive-foundation-on-java-8

Reactor - Flux



bron: https://tech.io/playgrounds/929/reactive-programming-with-reactor-3/Flux

Reactor - Mono

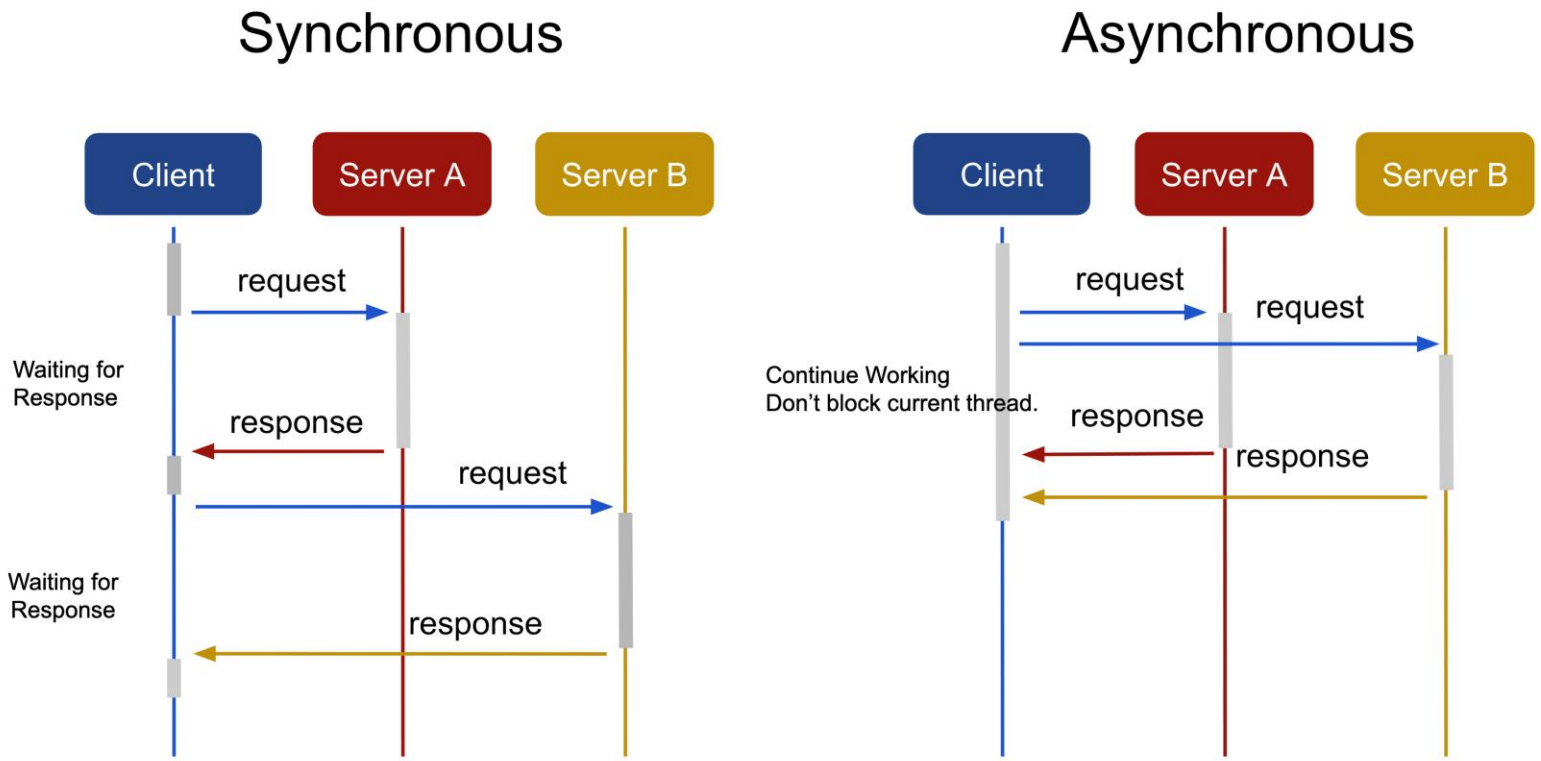


bron: https://tech.io/playgrounds/929/reactive-programming-with-reactor-3/Mono

Overzicht

- Herhaling reactive programming
- Reactive Streams
- Reactor
- Reactive REST-webservice

Asynchrone versus synchrone client

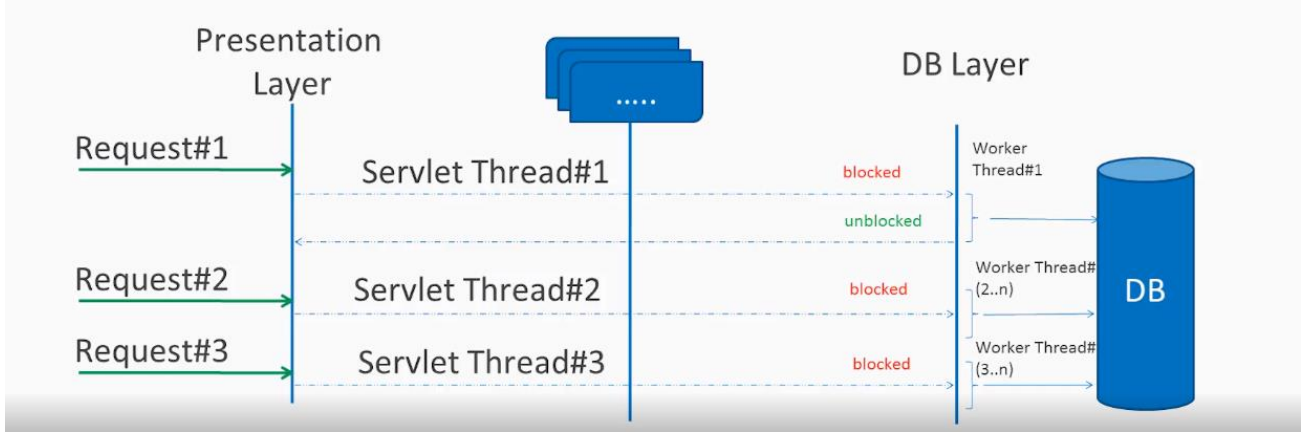


sneller

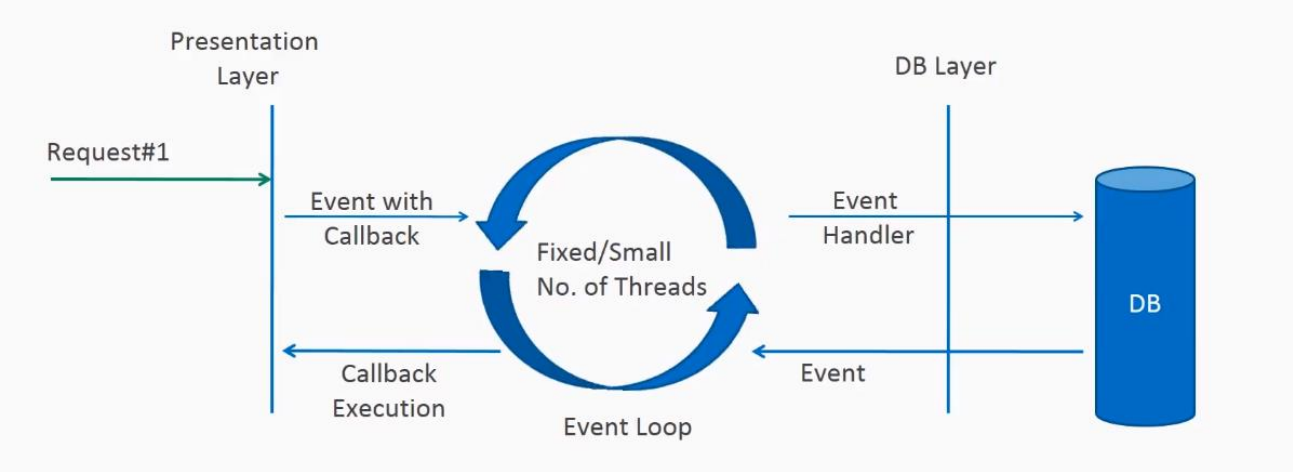
minder resources

bron: https://engineering.linecorp.com/en/blog/reactive-streams-armeria-1/

Asynchrone versus synchrone server



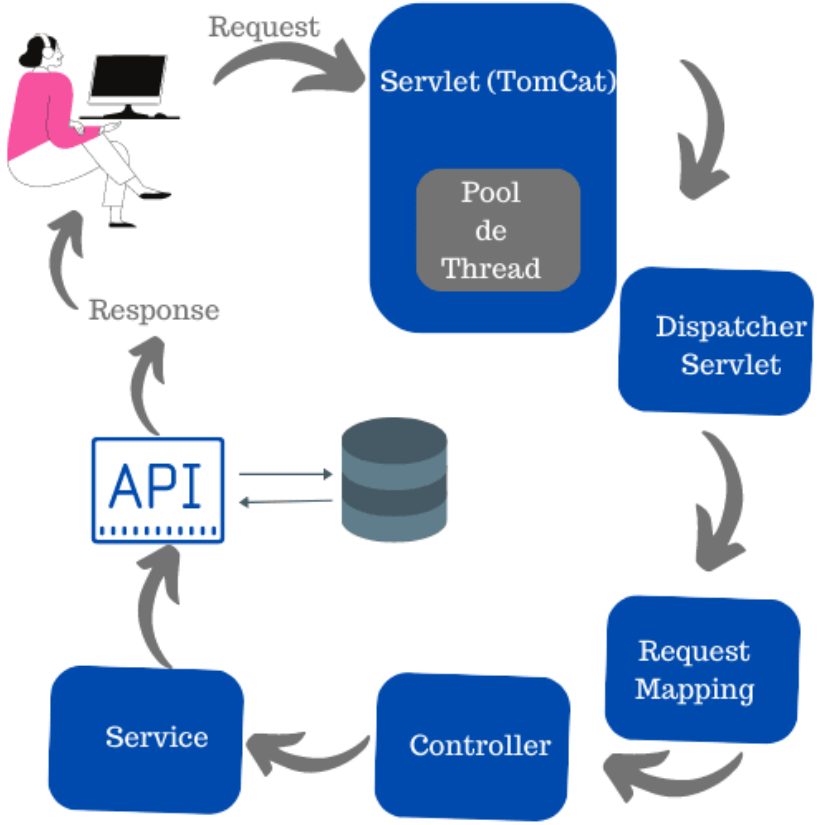
Blocking request processing

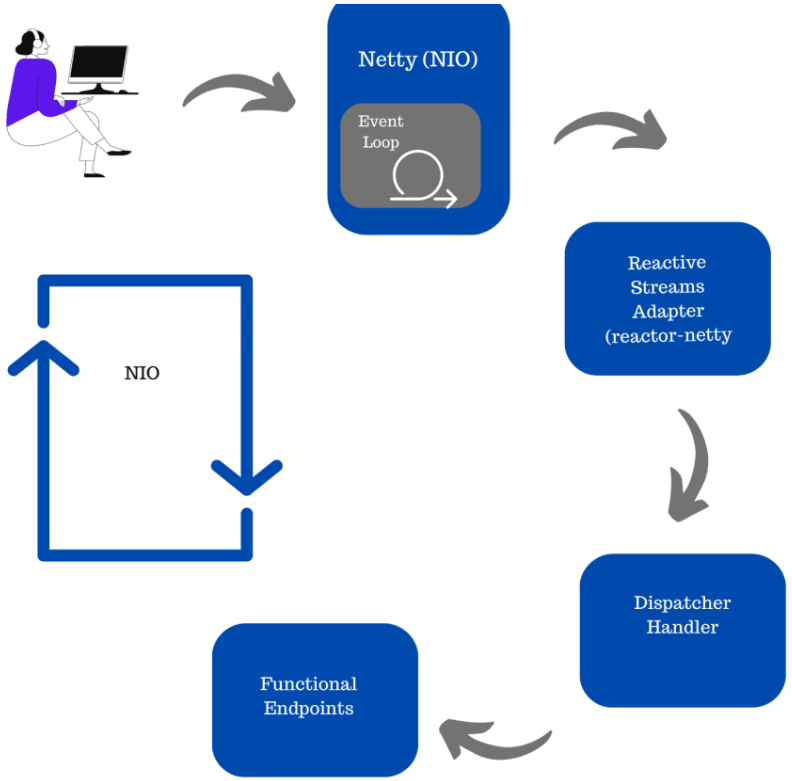


Non-blocking request processing

bron: https://howtodoinjava.com/spring-webflux/spring-webflux-tutorial/

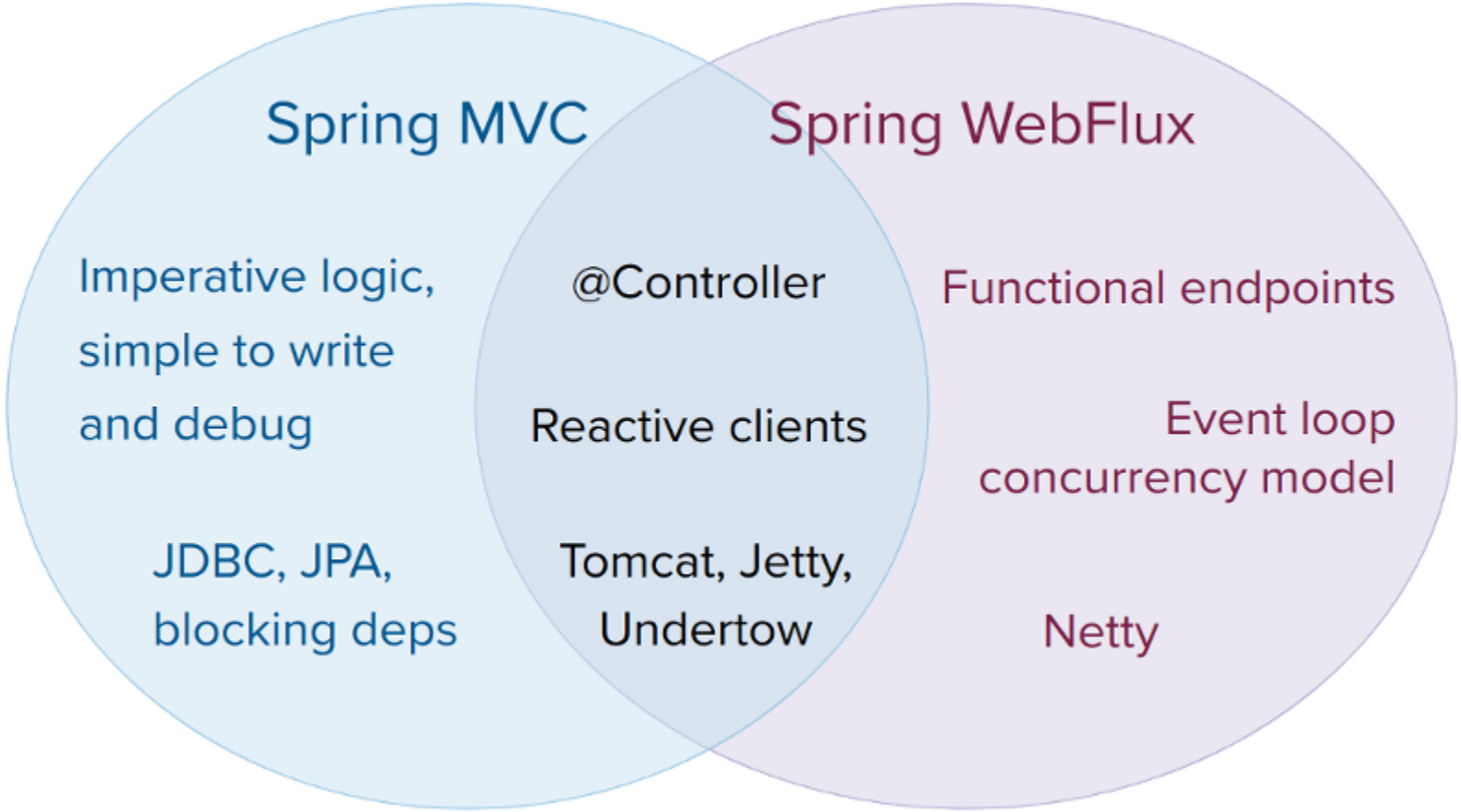
Spring MVC versus Spring WebFlux - Architectuur



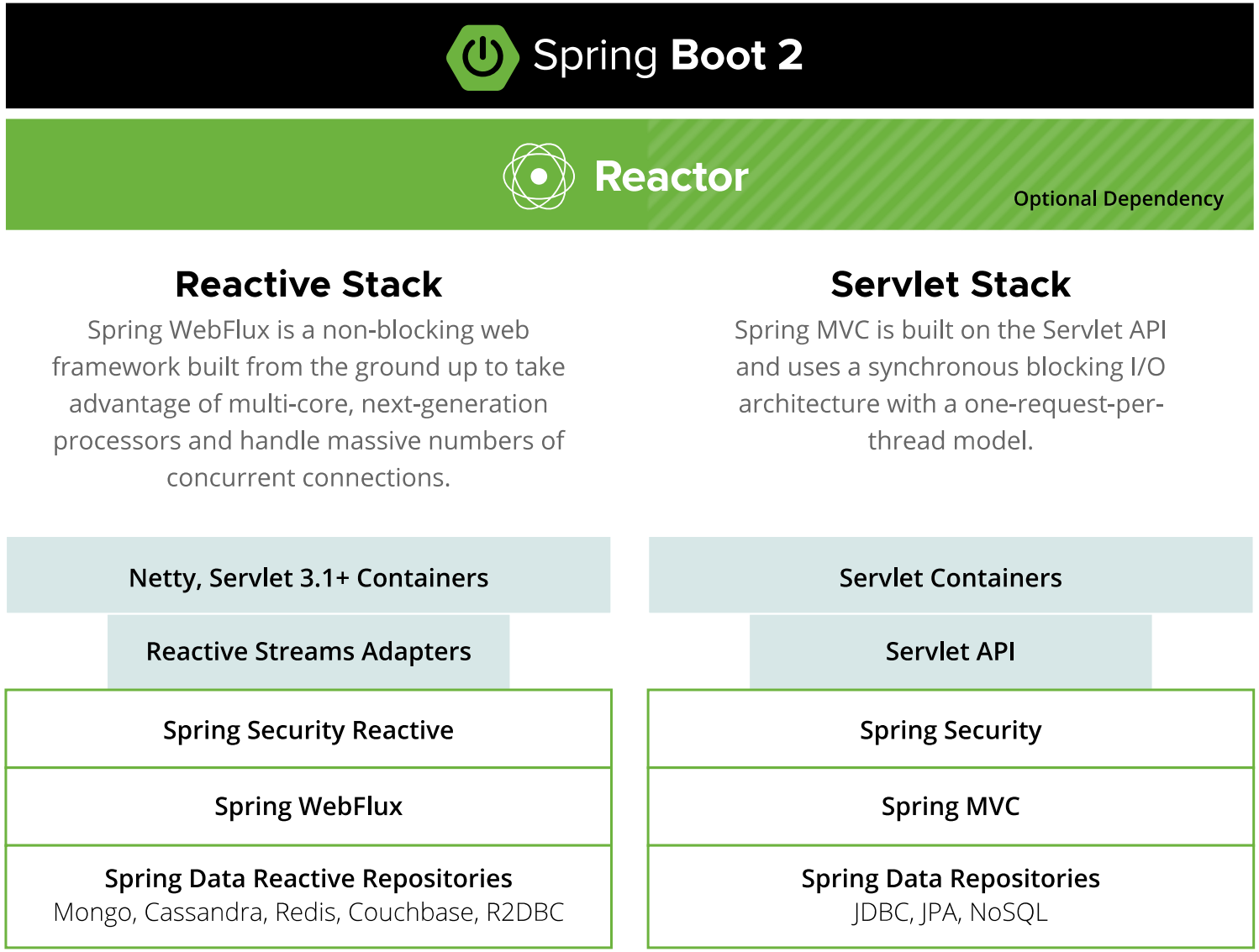


bron: https://dev.to/kamilahsantos/spring-webflux-reactive-java-applications-part-2-5b0l

Spring MVC versus Spring WebFlux



bron: https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html



bron: https://spring.io/reactive

Spring WebFlux

UNIVERSITEIT GENT

- Twee programmeermodellen
  - Reactive components met annotaties
    - ➢ Zelfde annotaties als Spring MVC @RestController, …
    - ➢ Resultaat methodes: Mono of Flux
    - ➢ GEEN blocking methodes gebruiken
  - Functional Routing and Handling
    - ➢ Klasse met methodes die bepaalde aanvragen afhandelen (Handler)
      - Parameter: request-object
      - Resultaat: Mono met response-object
    - ➢ Configuratie: routing
      - Pad ~ methode handler

Reactive DAO

```
@Service
public class BoorputDAO {
    final Random random = new Random();
    String[] ids = {"BP1", "BP2", "BP3"};

    public Flux<Boorput> geefMetingen() {
        return Flux.interval(Duration.ofSeconds(1)).take(10)
                .map(pulse -> geefMeting());
    }

    private Boorput geefMeting() {
        Boorput boorput = new Boorput();
        boorput.setId(ids[random.nextInt(ids.length)]);
        boorput.setTijdstipDebiet(LocalDateTime.now());
        boorput.setTijdstipPeil(LocalDateTime.now());
        boorput.setPeil(28 + 5 * random.nextDouble());
        boorput.setDebiet(5 + 2 * random.nextDouble());
        return boorput;
    }
}
```

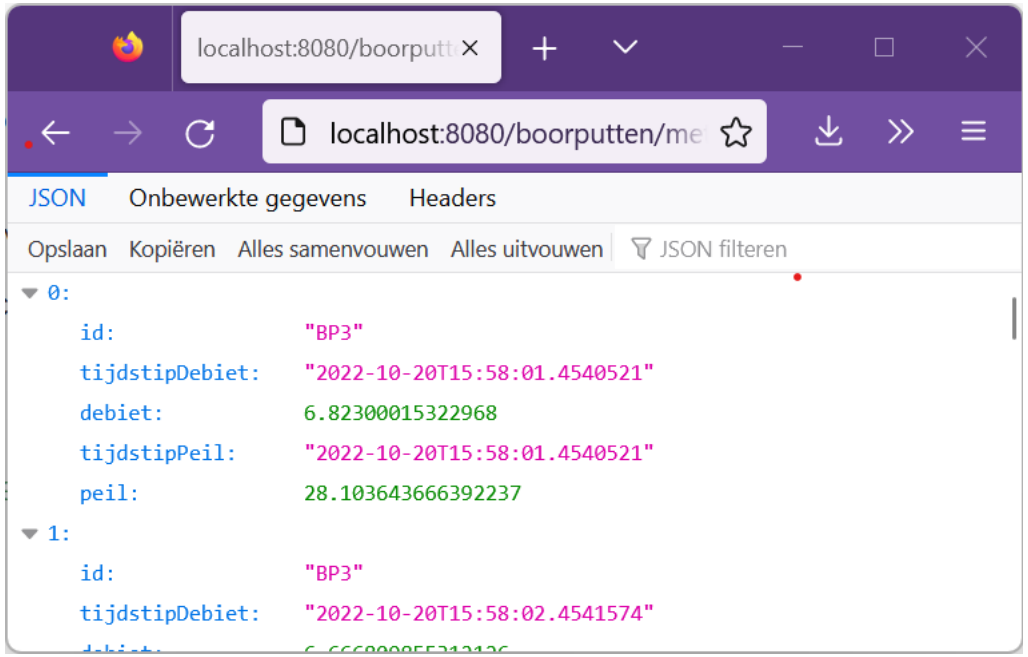
Reactive REST-webservice

UNIVERSITEIT GENT

```
@RestController
@RequestMapping("boorputten")
public class BoorputController {

    private BoorputDAO dao;

    public BoorputController(BoorputDAO dao) {
        this.dao = dao;
    }

    @GetMapping("metingen")
    public Flux<Boorput> haalMetingen() {
        return dao.geefMetingen();
    }
}
```

Browser: blocking/blokkerende client

even wachten



Webclient: niet blokkerend (non blocking) client

UNIVERSITEIT GENT

```
System.out.println("start test");
ClientTestWebflux client = new ClientTestWebflux();
client.testServer();
System.out.println("na test");
```

```
public class ClientTestWebflux {

    public void testServer() {
        WebClient client = WebClient.create("http://localhost:8080");

        Flux<Boorput> boorputFlux = client.get()
                .uri("/boorputten/metingen")
                .retrieve()
                .bodyToFlux(Boorput.class);

        boorputFlux.subscribe(System.out::println);
        System.out.println("in test webflux");
    }
}
```

```
start test
in test webflux
na test
iii.vbWebflux.web.Boorput@7d721e6c
iii.vbWebflux.web.Boorput@6514c4aa
iii.vbWebflux.web.Boorput@70fe8219
iii.vbWebflux.web.Boorput@2063102e
iii.vbWebflux.web.Boorput@4d8feb30
iii.vbWebflux.web.Boorput@e7d47bf
iii.vbWebflux.web.Boorput@a97f223
iii.vbWebflux.web.Boorput@324d613f
iii.vbWebflux.web.Boorput@a9ca2d
iii.vbWebflux.web.Boorput@648d2538
```

Spring WebFlux

UNIVERSITEIT GENT

- Twee programmeermodellen
  - Reactive components met annotaties
    - ➢ Zelfde annotaties als Spring MVC @RestController, …
    - ➢ Resultaat methodes: Mono of Flux
    - ➢ GEEN blocking methodes gebruiken
  - Functional Routing and Handling
    - ➢ Klasse met methodes die bepaalde aanvragen afhandelen (Handler)
      - Parameter: request-object
      - Resultaat: Mono met response-object
    - ➢ Configuratie: routing
      - Pad ~ methode handler

Reactive REST-webservice - handler

UNIVERSITEIT GENT

```
@Component
public class BoorputHandler {
    private BoorputDAO dao;

    public BoorputHandler(BoorputDAO dao) {
        this.dao = dao;
    }
    public Mono<ServerResponse> metingenVoorBoorputten(ServerRequest request) {
        return ServerResponse.ok().contentType(MediaType.APPLICATION_JSON)
                .body(dao.geefMetingen(),Boorput.class);
    }
}
```

```
@FunctionalInterface
public interface HandlerFunction<T extends ServerResponse> {
    Mono<T> handle(ServerRequest request);
}
```

Reactive REST-webservice - routing

UNIVERSITEIT GENT

```
@Configuration
public class BoorputRouter {
    @Bean
    public RouterFunction<ServerResponse> routeBoorput(BoorputHandler boorputHandler) {

        return RouterFunctions.route(RequestPredicates.GET("/boorputten/metingenAnders")
                .and(RequestPredicates.accept(MediaType.APPLICATION_JSON)),
                        boorputHandler::metingenVoorBoorputten);
    }
}
```

```
@FunctionalInterface
public interface RouterFunction<T extends ServerResponse> {
    Mono<HandlerFunction<T>> route(ServerRequest request);
    // ...
}
```

```
public static <T extends ServerResponse> RouterFunction<T> route(
  RequestPredicate predicate,
  HandlerFunction<T> handlerFunction)
```

hulpfunctie

Spring Data Reactive

UNIVERSITEIT GENT

- MongoDB
  - Document-georiënteerde databank (JSON)
- Cassandra
  - Mix tussen key-value store en databank gebruikmakend van tabellen (wide column store)
- Redis
  - In-memory key-value database
- NoSql databases

Overzicht

- Herhaling reactive programming
- Reactive Streams
- Reactor
- Reactive REST-webservice