1)

Je kan een klant aanmaken en opzoeken
Aanpassen met put en deleten zijn in theorie ook mogelijk maar heb ik nog niet getest.

Waarshuwing als je update zoek eerst de entiteit die je wilt updaten op in de repository. update property voor property en dan save je het. Als je gewoon blind saved ook al komt de id overeen dan heb je een dubbel.

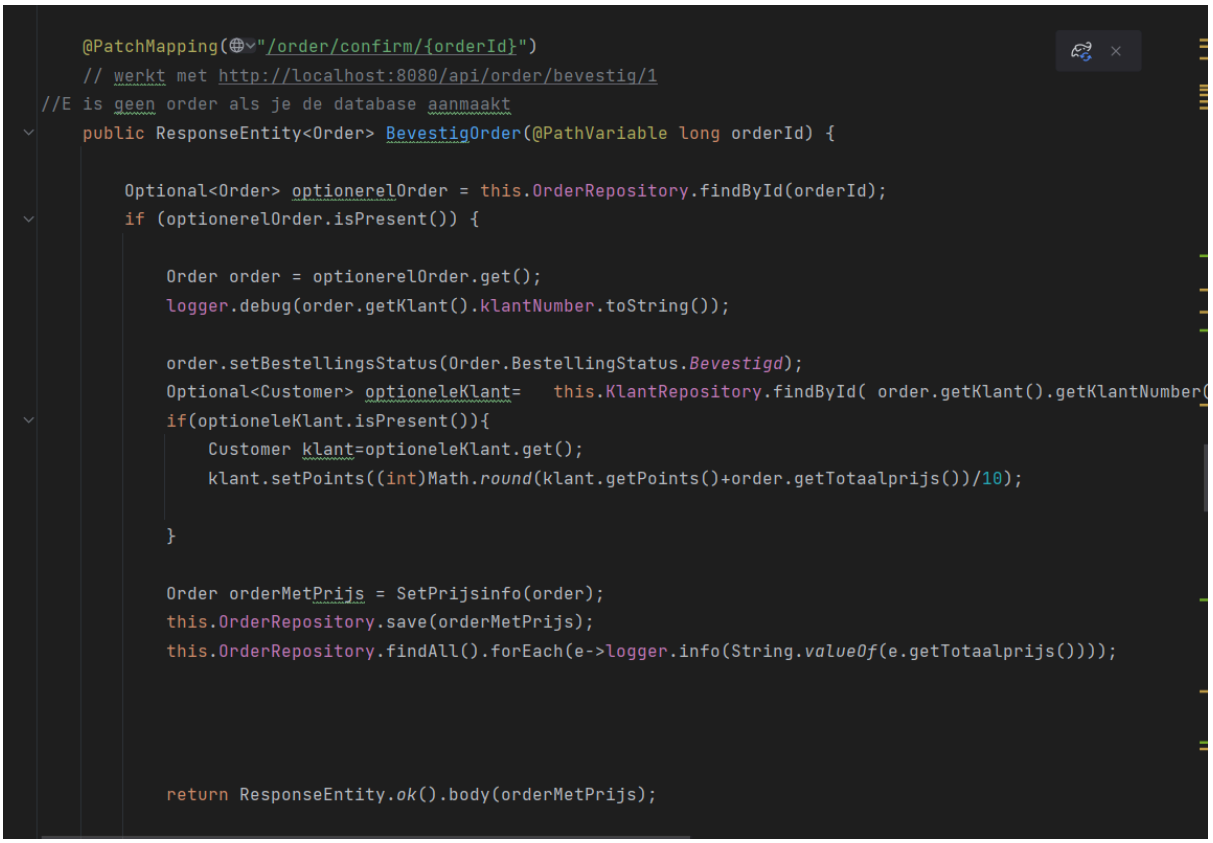
Herhaal een eerder gedane bestelling

```
@PostMapping("/order/history/{OrderId}")
public ResponseEntity<Order> HerhaalBestelling(@RequestBody Long klantId,@PathVariable Long OrderId) {
```

Doe een nieuwe bestelling
gebruikshandleiding: {"klantId":1,"order":{"1":3}}
klant met id 1 bestelt 3 keer het product met id 1.

```
//Json [] voor een lijst en {} voor een map of object
@PostMapping(value = "/order", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Order> PlaatsOrder(@RequestBody CustomerWithProducts customerWithProducts) {

Optional<Customer> optioneleKlant = KlantRepository.findById(customerWithProducts.getKlantId());
if (optioneleKlant.isPresent()) {
    Customer klant = optioneleKlant.get();
```

De klant bevestigt een order. Pas bij het bevestigen van een order de bakkery informeren.



```
    @PatchMapping("/order/confirm/{orderId}")
    // werkt met http://localhost:8080/api/order/bevestig/1
//E is geen order als je de database aanmaakt
    public ResponseEntity<Order> BevestigOrder(@PathVariable long orderId) {

        Optional<Order> optionerelOrder = this.OrderRepository.findById(orderId);
        if (optionerelOrder.isPresent()) {

            Order order = optionerelOrder.get();
            logger.debug(order.getKlant().klantNumber.toString());

            order.setBestellingsStatus(Order.BestellingStatus.Bevestigd);
            Optional<Customer> optioneleKlant=   this.KlantRepository.findById( order.getKlant().getKlantNumber(
            if(optioneleKlant.isPresent()){
                Customer klant=optioneleKlant.get();
                klant.setPoints((int)Math.round(klant.getPoints()+order.getTotaalprijs())/10);

            }

            Order orderMetPrijs = SetPrijsinfo(order);
            this.OrderRepository.save(orderMetPrijs);
            this.OrderRepository.findAll().forEach(e->logger.info(String.valueOf(e.getTotaalprijs())));




            return ResponseEntity.ok().body(orderMetPrijs);
```

De verantwoordelijkheid of het voorbij 22:00 uur is.
Ligt momenteel op beide kanten, de bakkery moet de batch hebben maar de client moet de klant informeren.

Annuleren is ook mogelijk maar weeral de uurcheck.

De Batch

openstaand, aan het bakken of gebakken ⇒ verantwoordelijkheid bakker
order komt binnen ⇒ is er een batch die de status openstaand heeft. Nee maak een nieuwe aan. Is het na 22:00? setDatetime(Localdate.now().add one day,LocalTime(0,0))
anders plaats ze op huidige tijd.

Sluit de bath af (status gebakken om 22:00) @Sheduled

```
//Enkel bevestige orders toevoegen!
/*@Component
public class BatchManager implements IBatchManager {
Logger logger= LoggerFactory.getLogger(BatchManager.class);

   private BatchRepository batchRepository;
```

```
   public    BatchManager(BatchRepository batchRepository){

       this.batchRepository=batchRepository;

   }

   public void AddOrderToBatch(Order order){

       List<Batch> openBatches=
batchRepository.findBatchByProductsStatusEnum(ProductState);
       if(openBatches.size()>0){

           openBatches.get(0).getOrdersInProcess().add(order);

       } else if (LocalTime.now().isBefore(LocalTime.of(22,0,0)))
{
           Batch batch=new Batch(LocalDateTime.now());
           batch.getOrdersInProcess().add(order);
           batchRepository.save(batch);

       }
       else{
           LocalDateTime
tommorow=LocalDateTime.of(LocalDate.now().plusDays(1),LocalTime.of
(0,0));
           Batch batch=new Batch(tommorow);
           batchRepository.save(batch);
           throw new AfterTenOClockError();
       }

       }

   @Scheduled(cron = "22 0 * * * ?")  Geen parameters toegelaten)
       public void CloseBatch(){
       List<Batch> openBatches=
batchRepository.findBatchByProductsStatusEnum(ProductsStatusEnum.O
PEN);
List<Batch>
batchesToClose=openBatches.stream().filter(batch->batch.getCreateT
ime().toLocalDate().equals(LocalDate.now())&&batch.getProductState
().equals(ProductsStatusEnum.OPEN)).toList();}}
```

Alle loyaliteitsklasse

```
 @GetMapping("/loyality")
public List<LoyalityClasses> ShowLoyalityClasses(){

   return loyalityClassManager.findAll();


   }
```

Een loyaliteistsklasse heeft een naam, een mimimumaantapunten en een korting tss de 0 en 1.

```
  @PostMapping("/loyality/create")
   public List<LoyalityClasses> CreateLoyalityClass(@RequestBody LoyalityClasses
loyalityClasses){

       loyalityClassManager.save(loyalityClasses);

       return loyalityClassManager.findAll();


   }
```

//TODO

Het is niet mogelijk om de prijs te setten van producten zonder prijs die niet gedeactiveerd zijn. Er is wel een query geschreven die je op het goede been zet (niet getest)
TODO Klanten mogen geen producten zien die een 0.0 prijs hebben of gedeactiveerd zijn.

Het gebruik van rabbitMQ.

Je hebt queues

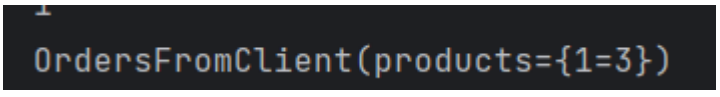
```
new *
@Configuration
public class RabbitConfiguration {

    @Value("${rabbitMQ.orderQueu}")
    private String orderQueue;

   /*  @Value("${exchange.newRecepyExchange}")
    private String exchangeName;*/


    //Kies de juiste import voor de queue
    new *
    @Bean
    public Queue queue(){

        return new Queue( name: "receiveOrder", durable: true);


    }

    new *
```

durable = als de broker herstart blijven de boodschappen bewaart.

1) Queus
2 Exchanges
TopicExchange (met routing key) ⇒ onderscheid tss boodschappen
Deactiveren van recepy (niet getest)
Doorgeven van orders naar bakery.

FanoutExchange (iedeeen krijgt de boodschap en kiest wat er mee te doen)
Een nieuw Recepy (werkende) Zowel warehouse als client moete weten. Warehouse heeft ingrediënten, Client voor de rest (prijs setting, naam)

De andere kant ontvangt een object dat hij niet per se kent. Het deserialiseren loopt waarschijnlijk mis. Meestal ontvang ik voor de eerste keer met een string.
Kijk eens aan, de bakery ontving zonet een order (= het werkt)

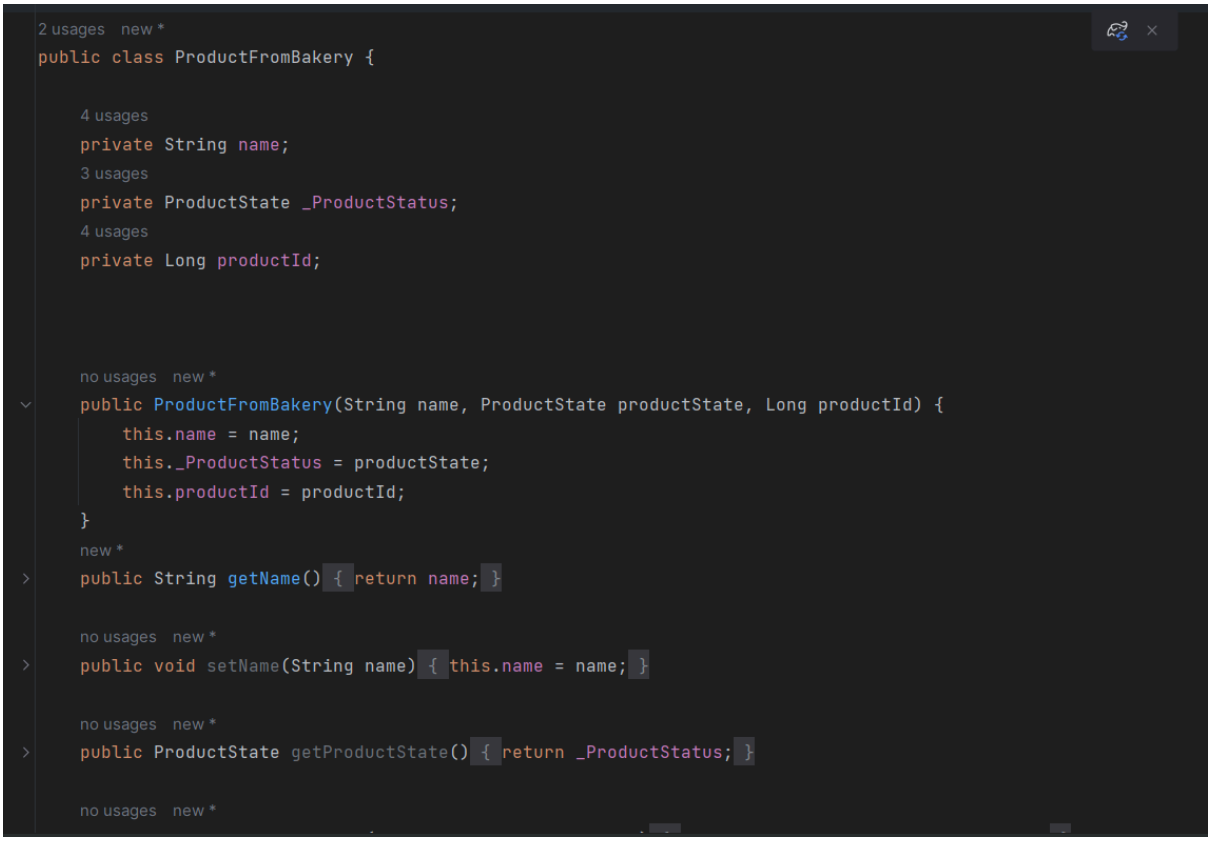


```
OrdersFromClient(products={1=3})
```

```
@Bean
    public FanoutExchange fanoutExchange(){

        return new FanoutExchange(exchangeName);


    }
    new *
    @Bean
    public TopicExchange topicExchange(){

        return new TopicExchange(deactivateRecepyQueue);


    }
```

Een queue heeft een naam, een exchange (elke soort) heeft een naam. topicExchange kent principe van keys om verder onderscheid te maken in berichten.

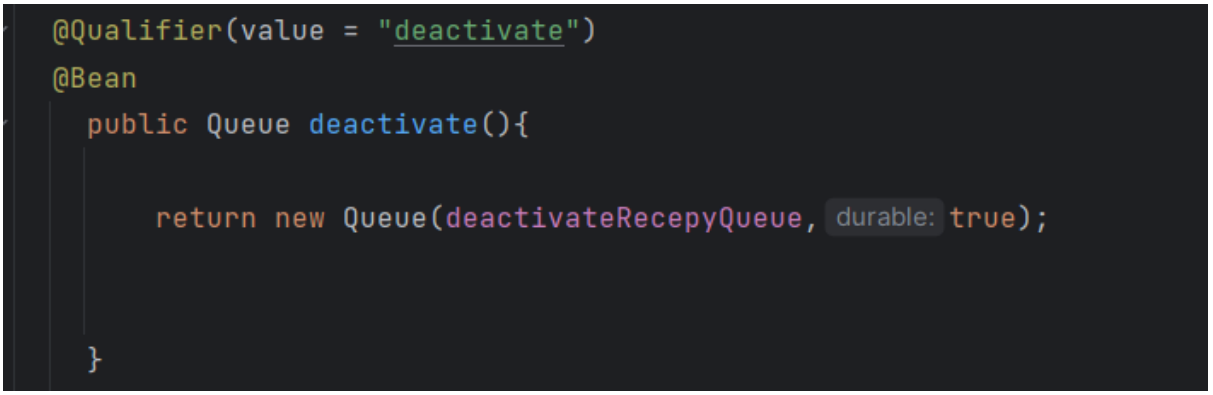
Dus het deserialiseer pobleem. Voorzie aan de andere zijde een dataTransfer object met de properties die je wilt overnemen van het object. **NEEM de id's van de bakker.** momenteel met een afzonerlijk voorziene property.
In de bakkey Product product=prouctManager.save(myproduct);
product bevat id.
rabitTemplate.send(product) ⇒ andere zijde vangt op.



```
2 usages  new *
public class ProductFromBakery {

    4 usages
    private String name;
    3 usages
    private ProductState _ProductStatus;
    4 usages
    private Long productId;



    no usages  new *
    public ProductFromBakery(String name, ProductState productState, Long productId) {
        this.name = name;
        this._ProductStatus = productState;
        this.productId = productId;
    }
    new *
    public String getName() { return name; }

    no usages  new *
    public void setName(String name) { this.name = name; }

    no usages  new *
    public ProductState getProductState() { return _ProductStatus; }

    no usages  new *
```

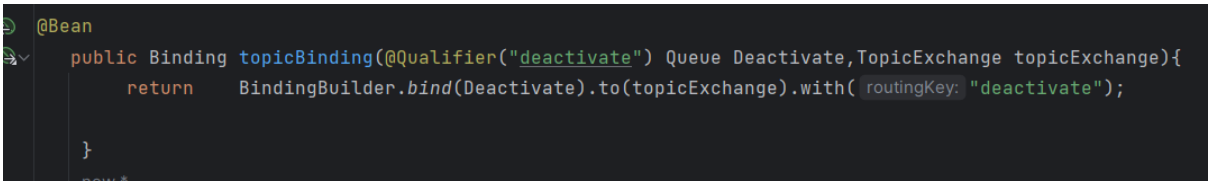
Eventueel annoteren met @Data

Meerdere queus…
Best annoteren met @ Qualifier.



```
@Qualifier(value = "deactivate")
@Bean
  public Queue deactivate(){

      return new Queue(deactivateRecepyQueue, durable: true);


  }
```

zodat spring niet in de war geraakt: zo te gebruiken



```
@Bean
  public Binding topicBinding(@Qualifier("deactivate") Queue Deactivate,TopicExchange topicExchange){
      return    BindingBuilder.bind(Deactivate).to(topicExchange).with( routingKey: "deactivate");

  }
```

TODO die XML file.