2016-2017

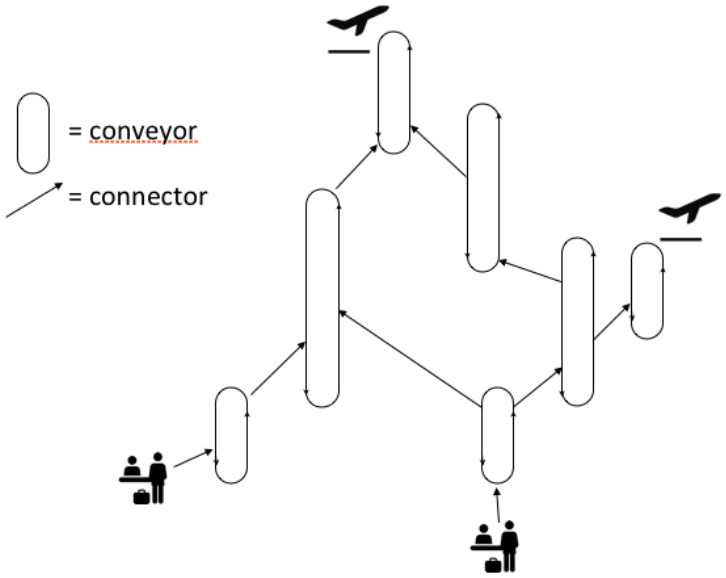
Bagage afhandeling

Examenproject 1e zittijd

Probleemstelling

De behandeling van de bagage van passagiers op luchthavens is een complexe aangelegenheid die wordt gecoördineerd door wat men een **Baggage Handling System** (BHS) noemt.

Bij het inchecken wordt de bagage voorzien van een label dat automatische identificatie toelaat[1]. De bagage komt dan terecht in een netwerk van lopende banden (*conveyors*) en verbindingen (*connectors*) dat ervoor moet zorgen dat het stuk tijdig bij de juiste gate aankomt om in het vliegtuig geladen te worden. Elke conveyor (verder in dit document 'band' genoemd) vormt een lus waarop de bagage in principe oneindig kan blijven ronddraaien. Een connector is een verbindingsstuk[2] tussen 2 lopende banden. Bij het betreden van een band (op het eindpunt van een inkomende connector) wordt de bagage geïdentificeerd door een sensor. Een sorteermachine[3] bevindt zich bij het beginpunt van elke uitgaande connector en moet de bagage automatisch op de juiste connector naar een andere lopende band duwen. Wanneer bepaalde banden op het netwerk vol dreigen te lopen dienen alternatieve routes door het systeem te worden bepaald. De volgende figuur kan hierbij ter illustratie dienen (dit is slechts een voorbeeld):



Bij aankomst van een vliegtuig vindt het omgekeerde proces plaats (van de gate naar een bagage ophaalpunt of naar een andere gate voor transit).

[1] Bijvoorbeeld via barcode scanning of NFC (near field communication)
[2] Dit is meestal ook een soort lopende band, maar volledig afgesloten, zonder splitsingen en geen lus.
[3] Deze bevatten tevens een sensor en kunnen ingesteld worden om een bepaald bagagestuk door te laten of opzij te duwen op de connector naar een andere band

Opdracht

Ontwerp en implementeer een BHS dat bestaat uit 2 delen: een *transport systeem simulator* en een *trafiek organisator*.
Opmerking. Met ***instelbaar*** wordt in wat volgt steeds bedoelt: instelbaar vanuit de test code, daar waar alle objecten worden geassembleerd ('wiring') en de toepassing wordt gestart. Met *in-memory collectie* wordt bedoelt dat gegevens hard-coded als een lijst in de test code worden aangemaakt.

A. Transport systeem simulator

De *simulator* is een standalone Java applicatie die in een testopstelling de volledige infrastructuur van lopende banden en splitsingen (met sensoren en sorteermachines) vervangt.
De simulator kan gestart en gestopt worden in 2 verschillende modi: *Generatie* of *Replay*.

Generatie
In deze modus genereert de simulator realistische data.
Op basis van een frequentieschema (zie verder) wordt nieuwe bagage in het systeem gebracht. Elk stuk bagage heeft volgende eigenschappen:

- BaggageID – integer – een unieke identificatie code, deze wordt oplopend gegenereerd bij elke generatie run van de simulator.
- VluchtID – string – random gekozen uit een instelbare in-memory collectie van vluchtnummers (8356543, 4563487,…)
- ConveyorID – integer – identificatie van de lopende band waarop de bagage het systeem binnenkomt – random gekozen uit een instelbare in-memory collectie van band identificatie codes (11, 12,…)
- SensorID – integer – het punt waarop de bagage op de lopende band met bovenstaand ConveyorID terechtkomt (eindpunt inkomende connector) – voor de eenvoud van het gehele model gebruiken we hiervoor een ConveyorID – random gekozen uit een instelbare in-memory collectie van band identificatie codes (1, 2,…). Dit is enkel nodig voor de eerste band in de 'berekende' methode (zie verder)[4]
- Timestamp – het moment van generatie ('now')

Om de drukte (ochtend, begin van vakantie) in een luchthaven te simuleren wordt gewerkt met een frequentieschema. Dit bestaat uit instelbare tijdsblokken in een periode van 24u (bv. 7:00-9:00, 9:00-15:00,…) waarbij een generatie frequentie hoort. Wordt deze bv. voor het blok 17:00-19:00 ingesteld op 2000, dan betekent dit dat de simulator – na opstarten – en wanneer de huidige tijd tussen deze twee tijdstippen valt, 2 seconden wacht alvorens een nieuw bagagestuk in het systeem te brengen.

Elk volgens deze methode gegenereerd stuk bagage wordt als een XML message via de message broker (zie verder) verstuurd naar de transport organisator.

[4] Voor banden intern aan het systeem zijn er altijd een of meerdere 'voorgaande' banden die via een connector zijn verbonden met de band en op die manier een bijhorend sensorpunt op deze band definiëren. Voor het binnenkomen van bagage in het systeem (check-in) is er niet echt een 'voorgaande' band, enkel een connector van de incheckbalie naar de eerste lopende band.

Indien 'record' wordt ingesteld, schrijft de simulator elk bericht bijkomend ook weg in een bestand op de harde schijf (formaat naar keuze, op een instelbaar pad). De timestamp wordt hierbij vervangen door een relatieve tijd (delay in milliseconden), door deze op te tellen bij het tijdstip waarop de generator is gestart.

Replay
In deze modus speelt de simulator een opgenomen instelbaar bestand opnieuw af. Elk bagage bericht in het bestand wordt op het juiste moment via de broker naar de organisator gestuurd aan de hand van de delay in de message.

Route simulatie
De organisator zal – op basis van route berekening (zie verder deel C.) – bepalen wat de volgende lopende band is waar het stuk bagage naartoe moet geleid worden. De organiser meldt dit door een routebericht te versturen via de broker. Dit voor de simulator inkomende bericht bevat volgende info:

- BaggageID – integer – de bagage waarop het bericht betrekking heeft
- ConveyorID– integer – naar welke lopende band de bagage moet geleid worden door de bijhorende sorteermachine op de band.

Een routebericht wordt door de simulator (zowel in generatie als replay modus) gebruikt om – op het juiste moment – een uitgaand sensorbericht uit te sturen dat aangeeft dat de bagage effectief een nieuwe band heeft betreden:

- BaggageID – integer – de bagage waarop het bericht betrekking heeft
- ConveyorID– integer – op welke lopende band de bagage terecht is gekomen
- Timestamp – integer – het moment waarop dit gebeurt (en dit bericht wordt uitgestuurd)

De bepaling van het moment waarop dit sensorbericht wordt uitgestuurd kan op 2 (instelbare wijzen) gebeuren: *vast* of *berekend*.

Vast betekent dat een bij opstart instelbare delay wordt gebruikt alvorens het sensorbericht uit te sturen, om zo een zeker mate van tijdsvertraging (en dus realisme) in het systeem te brengen.

Om nog meer realisme te bekomen kan de tijdsvertraging echter ook *berekend* worden door volgende gegevens te combineren:

- het moment waarop het stuk op de vorige band is terechtgekomen (eindpunt uitgaande connector=sensor)
- de tijd die de bagage nodig heeft om van dit punt tot aan de te nemen splitsing naar de volgende band te komen (beginpunt inkomende connector=sorteermachine)
- de tijd die de bagage doorbrengt op de connector tussen de 2 banden
- de tijd voor een bagagestuk om een volledige lus te maken op de vorige band (enkel nodig indien de berekende tijdsvertraging is overschreden bij binnenkomst routebericht, zie verder)

Deze tijdsgegevens worden opgevraagd aan een HTTP gebaseerde *ConveyorService*. De te gebruiken proxy hiervoor vind je op BB. Indien deze service niet bereikbaar is of fouten teruggeeft wordt het uitgaande bericht verstuurd volgens de 'vast' methode.

Wanneer (door uitval of een te trage werking van het gehele systeem) het routebericht wordt ontvangen terwijl de berekende tijdsvertraging reeds voorbij is (m.a.w. de bagage is reeds voorbij de sorteermachine), wordt de tijdsvertraging opnieuw berekend rekening houdend met de tijd nodig voor een volledige lus. De bagage blijft dus op de lus 'draaien'[5] tot het bericht alsnog ontvangen wordt.

Resultaten van de *ConveyorService* service dienen om performantie redenen in memory gecached te worden. Dit betekent dat het resultaat wordt bijgehouden in memory, zodat bij volgende calls met dezelfde data, de service niet opnieuw hoeft aangeroepen te worden. Het moet instelbaar zijn na welke periode deze cache automatisch leeggemaakt wordt.

Security
De simulator kan ingesteld worden met een lijst van (BagageID, ConveyorID). Wanneer een routebericht binnenkomt met (BagageID, ConveyorID) dat voorkomt in deze lijst wordt geen sensorbericht uitgestuurd. Hiermee wordt gesimuleerd dat deze bagage door het veiligheidspersoneel permanent van de band is gehaald.

B. Message broker

Simulator en organisator communiceren uitsluitend met elkaar via een messaging broker (RabbitMQ,..). Het transport formaat dat wordt gebruikt is XML. Indien een bericht niet kan afgeleverd worden via de broker wordt dit gelogd.

C. Trafiek organisator

Deze standalone Java applicatie zorgt voor een optimale stroom van de bagage doorheen het transport systeem. Het transport systeem (en dus ook de simulator ervan) is in die zin 'dom' dat het enkel de hardware aanstuurt: de banden met sensoren en sorteermachines. Alle 'routing logica' wordt afgehandeld in de organisator.

Route bepaling
Zodra een nieuw bagage bericht (zie hoger) via de broker wordt ontvangen, volgt de organisator de bagage op tot deze is aangekomen bij de 'gate'. Een gate is de laatste lopende band op de route van de bagage, hiervan wordt het rechtstreeks in het vliegtuig geladen of door passagiers opgehaald.

Om te bepalen naar welke gate de bagage moet geleid worden, contacteert de organisator op basis van het VluchtID een *FlightService* van de luchthaven (proxy op BB). Indien de FlightService niet bereikbaar is of een fout teruggeeft wordt later (bv. bij binnenkomst van een volgend bericht) opnieuw geprobeerd deze te contacteren voor het betreffende bagagestuk.

[5] In het echte systeem betekent dit dat de sorteermachines op de band geen command ontvangen om de bagage (na detectie) op een connector te schuiven. In de simulator betekent dit dat gewacht wordt met het uitsturen van het band bericht

Om te bepalen welke route de bagage moet volgen, gebruikt de organisator de *ConveyorService* (zie hoger). Deze geeft voor 2 ConveyorID's de mogelijke routes hiertussen terug als een lijst van ConveyorID's. De organisator bepaalt de meest optimale route door het totaal aantal bagagestukken dat op dit moment op elke band in de route zit samen te tellen. De route met de laagste score wordt gekozen. Indien de ConveyorService niet bereikbaar is of een fout teruggeeft wordt later opnieuw geprobeerd.
Resultaten van de ConveyorService service dienen om performantie redenen in memory gecached te worden zoals hoger beschreven.

Op basis van de huidige ConveyorID en de gekozen route wordt de ConveyorID van de volgende band uitgestuurd via de message broker zodat de simulator hierop kan reageren (zie hoger):

- BaggageID – integer – de bagage waarop het bericht betrekking heeft
- ConveyorID– integer – naar welke lopende band de bagage moet geleid worden door de bijhorende sorteermachine op de band.

Deze zal hier op zijn beurt op reageren met een sensorbericht (zie hoger). Gate en routebepaling gebeurt telkens men een sensor bericht ontvangt over een stuk bagage. Gate info kan immers wijzigen[6] en ook de drukte op de banden varieert.

Zodra een bericht binnenkomt waaruit blijkt dat de bagage op de laatste band in de route (= de gate) is aangekomen, dient het stuk bagage niet langer opgevolgd te worden. Een status message 'arrived' wordt via een message queue verstuurd:

- BaggageID – integer – de bagage waarop het bericht betrekking heeft
- Status – string ('arrived', 'undeliverable', 'lost')
- ConveyorID– integer – de lopende band waarop de bagage voor het laatste gesignaleerd werd

(status berichten hoeven niet verder behandeld te worden door een ontvanger)

Verloren bagage
Zaken kunnen mislopen (bv. bagage die aan elkaar vasthaakt, foutieve sensor detectie,…). In dat geval klopt de binnenkomende ConveyorID niet met de verwachte. Indien alsnog een geldige route kan bepaald worden, volgt een gewone behandeling. Indien dit niet het geval is wordt een status bericht 'undeliverable' uitgestuurd.

Security
Bagage passeert tijdens het transport langs een security checkpoint. Het is mogelijk dat bagage op dat moment uit het systeem wordt gehaald door het veiligheidspersoneel. Dit betekent dat er nooit meer een band wijziging bericht voor dit stuk zal binnenkomen. Wanneer een algemeen instelbare tijd is verstreken wordt voor een dergelijk stuk een status bericht 'lost' uitgestuurd.

[6] Dit gebeurt wanneer door de vluchtleiding wordt beslist om een vliegtuig op een andere gate te boarden.

D. Verwachte uitbreidingen

Volgende wijzigingen/uitbreidingen kunnen verwacht worden in de toekomst

- Wijzigingen aan de API's van alle services (zowel protocol als formaat)
- Gebruik van een ander type broker en wijzigingen aan het formaat van berichten op de broker
- Andere methoden dan random om bagage in het systeem te brengen in generatie modus
- Andere methoden om de optimale route te bepalen
- Andere methoden om de doorloop tijd op banden te bepalen
- Aanbieden van bagage tracking functionaliteit aan reizigers via een mobile app
- Aansturen van de snelheid van banden om trafiek verder te optimaliseren

E. Niet-functionele vereisten

- Er wordt **kwaliteitsvolle** en werkende code opgeleverd voor zowel de simulator als de organisator. Een UML class diagram mag gegenereerd worden uit de code en kan ter illustratie gebruikt worden.
- De 2 applicaties zijn onafhankelijk van elkaar, ze delen geen code en communiceren enkel via de message broker. Als broker wordt een open source messaging systeem zoals RabbitMQ of ActiveMQ gebruikt. Het berichtenformaat op de broker is XML. Alle verwerking van gegevens gebeurd verder volledig in memory, er wordt geen externe databank gebruikt.
- De code wordt geschreven in Java - in Engels[7] - en is onafhankelijk van een specifiek platform of framework. Je gebruikt dus enkel de standaard JDK (versie 8). Je gebruikt geen externe code/library's/… behalve voor het benaderen van de message broker, formaat conversie tussen JSON/XML/Java en voor logging (zie verder). Voor tijdsberekeningen kan gewerkt worden met classes in het JDK8 java.time package (LocalDateTime, LocalTime, Duration,..). Voor JSON en XML conversie met een library naar keuze (org.json, javax.json, castor,…).
  Error en info logging mag gebeuren
  - ofwel op System.out en dit in een static method van een zelf te schrijven Logger class waaraan info (message, level, exception) kan doorgegeven worden vanuit code die de log wil schrijven.
  - ofwel via log4J of een andere Java logging library
- Het gebruik van maven, gradle,… is toegelaten maar niet verplicht.
- Het uitwerken van unit testen is optioneel.

[7] Je wordt in dit vak niet beoordeeld op de kwaliteit van het Engels op zich, wel op de kwaliteit van de code/commentaar

Afspraken en verloop van het examen

- Het project wordt per 2 studenten uitgewerkt en afgegeven door dit voor de aanvang van het examen door te sturen naar het e-mail adres be.kdg.inf.se3@gmail.com. Je stuurt een **link** (drive, dropbox,…) naar een **zip** (geen rar!) met de naam van beide teamleden als volgt: <achternaamvoornaam>_<achternaamvoornaam>.zip. In de zip zitten alle sources.
- Zorg voor een **evenwichtige taakverdeling:** één student maakt de simulator en de andere de organisator. Het is **binnen 1 team toegelaten** om samen te ontwerp en elkaars code te reviewen.
  Studenten die (door omstandigheden) alleen werken, kiezen voor de uitwerking van een van beide delen en simuleren het andere deel aan de hand van bv. de RabbitMQ management console of een klein testprogramma dat hardcoded messages gebruikt.
- Het project (jouw deel) wordt **individueel** gedemonstreerd, verdedigd en beoordeeld aan de hand van vragen (zie beoordelingscriteria). Je komt per team het examenlokaal binnen zodra het vorige team het lokaal verlaat. Voor binnenkomst zet je op 1 van jullie laptop 's de 2 projecten open (in IntelliJ of andere IDE). Je start de broker en de twee applicaties. Bij binnenkomst koppel je deze laptop aan de beamer, zodat er tijdens het examen niet van laptop moet gewisseld worden. Kdg beamers hebben geen HDMI ingang. Een mini-display port adapter (voor Mac) is voorzien. Indien beide laptops enkel over een HDMI uitgang beschikken geef je dit direct aan bij binnenkomst zodat laptop docent kan gebruikt worden.
- Voorafgaand of aansluitend aan de verdediging los je **2 algemene vragen** op (op papier, gesloten boek) over **concepten** uit de cursus. Zie BB voor een overzicht van de vragen.
- Het is mogelijk dat je op het examen gevraagd wordt om wijzigingen aan de code aan te brengen.

Beoordelingscriteria

- Een goed begrip van de tijdens de les aangebrachte concepten (zie slides)
- Een correcte toepassing hiervan in het project
- Cleane, onderhoudbare en werkende code (met inbegrip van logging en error handling) die voldoet aan de standaard Java code conventies en zinvol gedocumenteerd is.
- De beheersing van het project: het ontwerp, de ontwerp variaties, de code, de uitvoering, de te verwachten uitbreidingen en hoe de code hierop voorzien is.

Veel succes!