

2022-2023

Software architecture

Project - Velo



In vrijwel elke grote stad zijn deelsystemen beschikbaar om je flexibel te verplaatsen op een ecologisch verantwoorde manier. Je bouwt in dit vak software voor een dergelijk systeem.

Probleembeschrijving

Op verschillende plaatsen verspreid over de stad bevinden zich *stations* van verschillende grootte waarin ‘dock-vehicles’ zijn verankerd in een elektronische *lock*. Een gebruiker kan bij een station een vehicle unlocken¹. Het systeem bepaalt welke lock zich opent. Later kan hij dit vehicle weer inchecken bij een station door het in een beschikbare door de gebruiker gekozen lock te plaatsen. Naast de vehicles waarmee je van het ene naar het andere station kan rijden zijn er ook ‘dockless-vehicles’ (steps) beschikbaar die je gewoon ergens kan oppikken en achterlaten. Zij hebben een elektronisch slot dat zichzelf los of vastzet wanneer een gebruiker via een app² aangeeft dat hij zijn rit wil starten/beëindigen.

Om van het systeem gebruik te maken dient men een subscription te nemen. Afhankelijk van het subscription type (dag, week of jaar) en het vehicle type (dock-vehicles, dockless-vehicles) is de prijs berekening verschillend. De prijs wordt telkens bij het beëindigen van een rit berekend en aan een extern facturatie-systeem afgeleverd.

Dockless-vehicles hebben een GPS die op geregelde tijdstippen de vehicle locatie doorstuurt naar het systeem (behalve wanneer het vehicle in maintenance zit). Dit laat toe om vehicles op locatie te zoeken, maar wordt ook gebruikt voor ‘open ride’ detectie. Een open ride komt voor wanneer een gebruiker een vehicle vergeet te locken na gebruik.

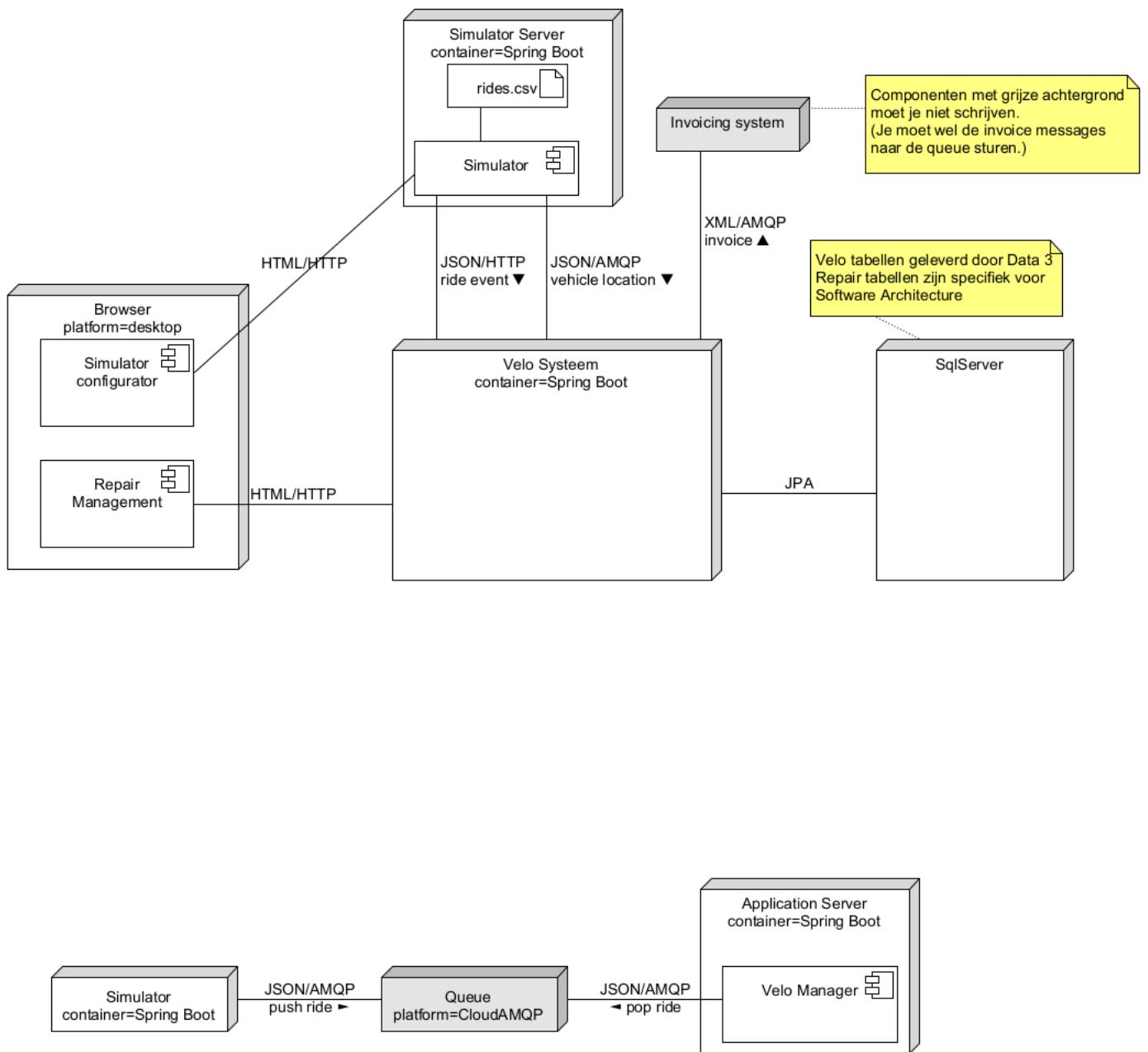
Een vehicle (dockless en dock) moet natuurlijk ook op regelmatige basis worden onderhouden. Een vehicle wordt door een maintenance systeem geflagged wanneer het op onderhoud moet komen. Wanneer een vehicle geflagged is zal het niet meer beschikbaar zijn voor gebruikers tot het terug is vrijgegeven na onderhoud. De beslissing of een vehicle op onderhoud moet, wordt berekend op basis van 3 verschillende parameters: het aantal gereden kilometers, de lengte van de periode tussen nu en het vorig onderhoud of wanneer een gebruiker het vehicle als defect opgeeft tijdens het beëindigen van zijn rit. Indien een gebruiker een vehicle als defect opgeeft, zal de rit niet worden aangerekend. Een onderhoud wordt uitgevoerd door een technieker, die via een web interface verschillende acties kan aangeven zoals banden vervangen, oppompen, ketting smeren, ...

Analisten en operationele medewerkers van de stad moeten het systeem kunnen testen en finetunen. Het ontwikkelen van een aparte toepassing die ritten kan simuleren is daarom een essentieel onderdeel van het project. Deze simulator heeft een web interface voor het beheren van simulatie-instellingen.

¹ Dit gebeurt door een pas voor een magneetlezer bij het station te houden

² Het schrijven van deze app zelf behoort niet tot de opdracht

Volgende figuur geeft een overzicht van de globale architectuur van het systeem.



Functionaliteit

Simulator

In onderstaande tabel zijn de ‘events’ weergegeven die de simulator moet kunnen produceren. Sommige van deze events resulteren in een **call** naar de (REST) API van het vehicle systeem (zie verder), andere in het versturen van een **message** via een queue.

Event	Type	Params/Data	Return
unlock dock-vehicle	call	userID stationID	http status code lockID
unlock dockless-vehicle	call	userID vehicleID	http status code
location of dockless-vehicle	message	timeStamp vehicleID latitude longitude	-
get available station locks	call	stationID	http status code list of lockID's
lock dock-vehicle	call	userID stationID lockID vehicleID defect	http status code
lock dockless-vehicle	call	userID vehicleID defect latitude longitude	http status code
find nearest dockless-vehicle	call	latitude longitude	http status code vehicleID latitude longitude

Deze tabel is niet finaal en mag worden aangepast en uitgebreid voor de compatibiliteit.

Het data formaat voor zowel de API calls als de queues is JSON. De precieze details van het formaat zijn vrij te kiezen.

De simulator kan 2 types van simulaties draaien: **file en random**.

In een **file simulatie** **USID9** worden de events (zie bovenstaande tabel) uitgelezen uit een CSV bestand. Hierin staat per event een lijn met de event data en een delay. De delay (in millis) bepaalt hoe lang er na het vorige event moet gewacht worden. **USID10** De simulator leest deze file en voert de nodige acties uit (API calls of messages op de queue zetten).

Een **random simulatie** is bedoeld voor bv. performance testen. **USID12** Deze worden niet uit een file gelezen maar random gegenereerd volgens default instellingen die uit een configuratiebestand worden gelezen:

- tijdsperiode van generatie (bv. 60min)
- aantal users die in dezelfde periode rides uitvoeren
- gemiddeld delay tussen de berichten
- grootte van random variatie op dit delay (bv. delay van 2000ms met variantie van 200ms geeft 1821, 2134,...)
- latitude range (random generatie binnen deze grenzen)
- longitude range (random generatie binnen deze grenzen)

De simulator heeft een web interface om deze te bedienen:

- **USID11** Simuleren van een rit via upload van een CSV file
- Starten van een random simulatie op basis van instellingen. **USID13** De default instellingen worden getoond in form fields en kunnen worden aangepast door de gebruiker (ze hoeven niet opgeslagen te worden) **USID39**

Simulaties worden asynchroon gestart zodat de webpagina niet ‘hangt’ terwijl de simulatie bezig is.

VeloSysteem

Beheer van ritten

Het systeem verwerkt de eigenlijke ritten en heeft volgende verantwoordelijkheden:

- **Aanbieden van een API voor**
 - Locken, unlocken, .. **USID1 USID2 USID3 USID4 USID6 USID50**
 - CRUD beheer van station, locks en vehicles **USID27**
- **Ontvangen van locations** van dockless-vehicles via een queue en deze verwerken **USID5**
- **Facturatie.** Bij het afsluiten van een rit wordt berekend wat de gebruiker hiervoor dient te betalen. **USID7** Van elke ritprijs wordt een **XML** bericht (userID, prijs, ride details) gemaakt dat op een uitgaande queue wordt geplaatst (deze queue wordt niet uitgelezen, effectieve verwerking van de facturatie is out of scope).. **USID8**
- **Open ride detectie.** Het kan gebeuren dat een gebruiker een *ride* niet goed afsluit aan het einde van een rit. Om oplopende kosten te vermijden is het belangrijk dat dit zo snel mogelijk wordt gedetecteerd zodat er een notificatie (je plaatst deze gewoon op de log) kan verstuurd worden. Er zijn 2 mechanismen:
 - **Tijd-gebaseerd** **USID28**
Een rit die langer duurt dan x minuten (setting) wordt als open-ended beschouwd. Tijd-gebaseerde detectie wordt zowel voor dockless- als dock-vehicles toegepast.
 - **Locatie-gebaseerd** **USID29**
Voor dockless-vehicles zal het achtergelaten vehicle positie informatie blijven uitzenden vanop dezelfde locatie. Door de beperkte nauwkeurigheid van GPS signalen zullen hier evenwel kleine fluctuaties op zitten. Als de verplaatsing gedurende y minuten (setting) onder een bepaalde fluctuatie z blijft (setting) wordt de rit als open-ended beschouwd. Locatie-gebaseerde detectie is enkel relevant voor dockless-vehicles.

Berekening van prijzen

Het systeem berekent de prijs van elke gereden rit op basis van een aantal parameters. **USID**

- **Subscription-type** De prijzen hieronder verschillen op basis van het subscription-type van de gebruiker.
 - Dag
 - Maand
 - Jaar
- **Unlock-fee** Deze fee wordt altijd aangerekend bij het starten van een rit met een dockless-vehicle.
- **Price / km** Deze prijs wordt berekend voor dockless-vehicles op basis van de verschillende location messages en de afstand tussen deze coördinaten. **USID40**
- **Price / min** Deze prijs wordt berekend voor dockless- en dock-vehicles per gereden minuut (tussen unlock & lock tijd). **USID41**

Het systeem hoeft geen rekening te houden met het aanrekenen van abonnementskosten.

1. Dockless-vehicles

Subscription-type Parameter	Dag	Maand	Jaar
Unlock-fee	€1,5	€1	€1
Price / km	€0,30	€0,22	€0,12
Price / min	€0,20	€0,12	€0,08

Voorbeeld: Een gebruiker met dag abonnement neemt een dockless-vehicle en rijdt hiermee 10km in 30min. De totaalprijs is dan $\text{€}1,5 \text{ unlock fee} + (10 \times \text{€}0,30 \text{ price/km}) + (30 \times \text{€}0,20 \text{ price/min}) = \text{€}10,5$

2. Dock-vehicles

Dock-vehicles hebben een gratis ride-period van 30min. De onderstaande prijs wordt dus pas aangerekend bij rides van langer dan 30min.

Subscription-type Parameter	Dag	Maand	Jaar
Price / min	€0,15	€0,10	€0,08

Onderhoud van vehicles

Een vehicle moet in maintenance wanneer deze staat geflagged als defect door een gebruiker **USID14**, een aantal gereden kilometers heeft **USID15** of wanneer het al even geleden is tot zijn laatste onderhoud **USID16**.

Vehicle-type Parameter	Dockless-vehicle	Dock-vehicle
Distance	100km	250km
Period	3 weken	8 weken

Het systeem heeft een web interface die gebruikt wordt door techniekers om onderhoudsacties te koppelen aan vehicles (CRUD) **USID17** en de vehicles terug vrij te geven. **USID18**

Voor vehicles (dockless en dock) die in onderhoud zijn, wordt er telkens een rit aangemaakt met een blanco subscription ID. Bij het aanmaken van een nieuwe onderhoudsbeurt moet er dus aan gedacht worden dat zo'n rit aangemaakt wordt. **USID19**

Voorzien zijn op mogelijke wijziging en uitbreidings

De code moet (met behulp van interfaces, patterns,...) open zijn voor toekomstige uitbreidingen

- **USID20** andere methoden van prijsberekening voor een rit
- **USID21** andere methoden voor maintenance flagging
- **USID22** andere methoden om open rides te detecteren
- **USID23** verschillende soorten dockless-vehicles
- **USID24** uitbreiden van stations
- **USID25** andere data formaten en protocollen (JSON, XML, verschillende soorten queues, ...)
- **USID26** applicatie opdelen in verschillende micro-services

Change requests in de loop van het project zijn mogelijk.

Testen

Volgende geautomatiseerde testen worden minstens voorzien op het vehiclesysteem (niet op de simulator):

- **USID42** Unit testen voor de prijsberekening
- **USID43** Unit testen voor open ride detectie met mocking van alle afhankelijkheden
- **USID44** Integratie testen vanuit de controllers

We verwachten een code coverage van minstens 70%.

Configuratie en Logging

USID34, USID35

De code wordt voorzien van zinvolle logging statements zowel op vlak van diagnose (debug, info) als foutafhandeling (warn, error...). Zaken die kunnen mislopen en dus gelogd moeten worden zijn onder meer:

- *De API wordt aangeroepen met foutieve input data* (bv. niet gekend vehicleID). Een error code (volgens REST conventies) wordt teruggestuurd en een warning gelogd.
- *Databases kunnen niet bereikt worden of geven onverwachte fouten terug*. Een error wordt gelogd en een error code teruggestuurd (indien de database toegang het gevolg was van een API call).
- *Queues kunnen niet bereikt worden of geven fouten terug*. Een error wordt gelogd. Indien het de uitgaande facturatie queue betreft wordt de rit op een later tijdstip opnieuw verwerkt.
- *Location berichten kunnen niet geconverteerd worden van wire format naar memory*. Het bericht wordt niet verwerkt. Een error wordt gelogd
- *CSV files kunnen niet correct ingelezen of verwerkt worden*. De file wordt niet verwerkt. Een error wordt gelogd

USID30, USID37

Queue en database settings (host,...) worden uit een configuratie bestand gelezen³

³ application.properties in Spring terminologie

Security

Security valt buiten de scope van de applicatie. De API calls gebeuren dus zonder tokens en je hoeft niet aan te loggen op de web interfaces.

Application security komt aan bod in Integratieproject 2.

Technische specificaties

Je gebruikt **Spring Boot** (Java of Kotlin) en Gradle voor de ontwikkeling. **USID45, USID46**

Je gebruikt een SQL Server 2019 database **USID48**. De SQL Server database wordt aangeleverd als een databank export. Dit is ook de database die gebruikt wordt voor het project in de vakken Data persistency/engineering/analytics. Het is geadviseerd om deze database in een Docker container te draaien.

Voor de queue gebruik je CloudAMQP messaging. **USID49**

USID31, USID32

Het versiebeheer gebeurt in **GitLab**. Ook de project planning en opvolging gebeurt aan de hand van **GitLab**.

Zie de slides *Inleiding* en de infosessies voor verdere details over de opvolging, technologieën, architectuur etc..