MaaS Provider

Enterprise Integration Project Proposal

Rui Ventura and Pedro Cerejo
81045 81338

Alameda – Group 11

Instituto Superior Técnico, Universidade de Lisboa
pedro.cerejo+rui.ventura@tecnico.ulisboa.pt

1 Business Case

Urban mobility is rapidly evolving, not only due to the increased awareness with environmental issues, but because of the huge impact digitalization has on the traditional ways people are able to use transportation systems.

The idea behind Mobility as a Service (MaaS) is analogous to that of Software as a Service (SaaS), a term popularized by Cloud technology: people use the transport network they see better fits their needs while doing so in a flexible and easy fashion, also enabling the inclusion of innovative alternatives for personal transportation like rental bike, scooters, motorcycles, and the likes.

The main challenge arises when it comes to seamlessly integrating all of these services together without the hassles that come with the need of multiple ticketing mechanisms or devices, be it through the usage of physical cards or different mobile applications, proving card/device and fare payment management arduous tasks.

1.1 Business Description

The MaaS platform must be capable of incorporating several transportation services with ease, enabling further operation adherence in the future, as well as providing the customer a satisfying experience by removing the need to manage all of the little intricacies of using multiple different transportation networks.

To that end, the goal is to model an event-driven system agnostic from the transportation operator by implementing mechanisms for handling certain types of "ways of consuming transportation". Take, for example, the myriad of ways public transportation can be consumed: via pre-paid cards, for one-off trips, or using a temporal tickets, where the price can be correlated to the amount of time spent within the public transport's boundaries.

Besides public transportation, a rise in private transportation providers popularity requires attention. However, one can easily integrate with such providers with a similar approach by appropriately adapting to the ways such providers offer transportation services.

1.2 Business Value

By successfully achieving integration goals described previously, we must establish an interoperable service bus through which the various entities of the system can interact and communicate with, effectively assembling a network of transportation operators. The ease of integration brings more transportation operators into the system, which increases the appeal for a wider range of customers to enroll in the platform.

From the transportation operators' perspective, a platform that gathers the users of several other operators has the potential of bringing new customers and increase revenue with almost no effort.

2 Entities

The MaaS platform is comprised of a few different entities, among which are the platform's customers, or clients, capable of using the means of transportation offered by the transportation service providers, or operators, another set of entities in the system, that the platform integrates with.

Transportation service providers, as the designation entails, provide a series of heterogeneous transportation services for the customer to consume. As it stands, the platform provides three different types of services: (1) check-in/check-out, which resembles an ephemeral one-time on-demand pre-paid model of using transportation, (2) flat fare, for one-shot temporally independant trips or even similar to a subscription-like consumption model (e.g. monthly pass), and (3) operator-defined, where the operator might possess complex logic for computing the fare of a trip, hence calculating it *in-premises* and providing the value after the fact (e.g uber, taxify).

To incorporate the aforementioned entities into the system, we essentially require a set of both internal and external services in order to enroll transportation service providers, their respective products/services, and to enroll users/customers.

Essentially, a human management corp is required for assessing operator registration requests as part of a semi-automatic process for doing so, as well as for revising subsequent product/service registration requests, again, on the operator's behalf.

To enroll transportation service providers, we must integrate with a few external fiscal information and postal address services to perform proper validation and information gathering.

With respect to the customers, the system also encompasses a virtual payment service provider (outside the scope of the actual implementation) in order for the customers to perform payment orders emitted by the system as transportation services are provided.

3 Functional Specification

The platform is comprised of a series of business processes and services that enable correct functionality of the system. Customer and transportation operators enrolment is made available through business processes. When enrolling, transportation operators provide their information and that information is verified by the system with the help of a human (MaaS management corp). When the process completes successfully, the system creates a channel through which the operator can send events related to the user trips. Our system provides a simple contract for those events.

When the users register in the MaaS platform, a token is generated and associated to that user. The operator/validator sends events related to a user and the platform is responsible for handling the payment orders for that user and to adequately distribute the revenue across the operators. The payment order is ran by an account management service that attempts to make a funds request to a payment service provider. If the payment is rejected, the user is blacklisted and all the operators are notified. Otherwise, the system registers a trip for the user. Based on the trips made, the user gets rewarded with discounts that can be percentage-based or flat.

4 Business Processes

In whole, we've designed three business processes: one for registering a customer in the platform, illustrated in Figure 1, and two other processes for registering a transportation service provider and the products/services it provides, respectively shown in Figures 2 and 3.

5 Architecture

The system's composed of an assortment of micro-services deployed in a Cloud environment (specifically, Amazon Web Services (AWS)), and services generated from Business Process Execution Language (BPEL) and Business Process Model and Notation (BPMN) processes using the Oracle 11g Service Oriented Architecture (SOA) Suite software. A visual depiction can be seen in Figure 4.

The cloud services are all written in different languages whilst the Oracle software ones are pure Java.

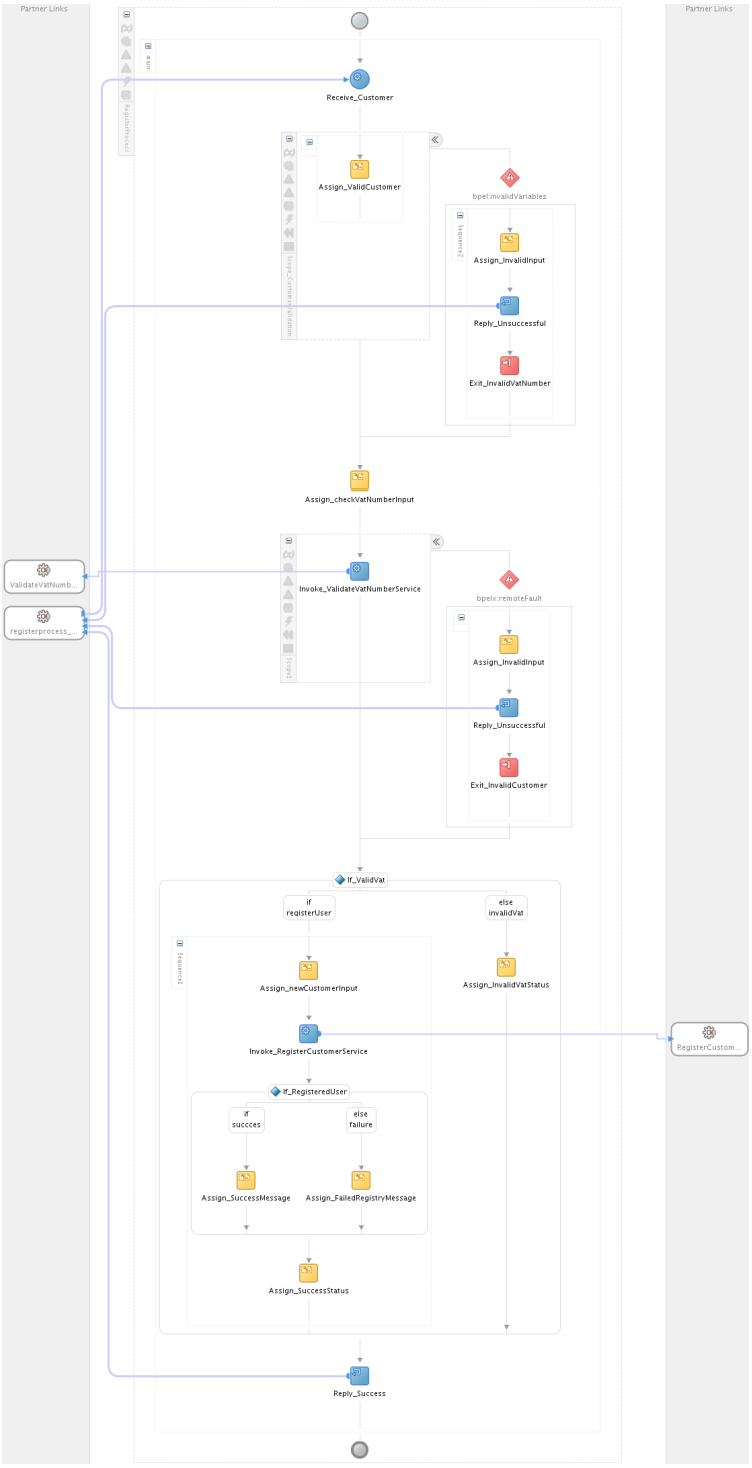

Fig. 1: Business process for enrolling a transportation service provider

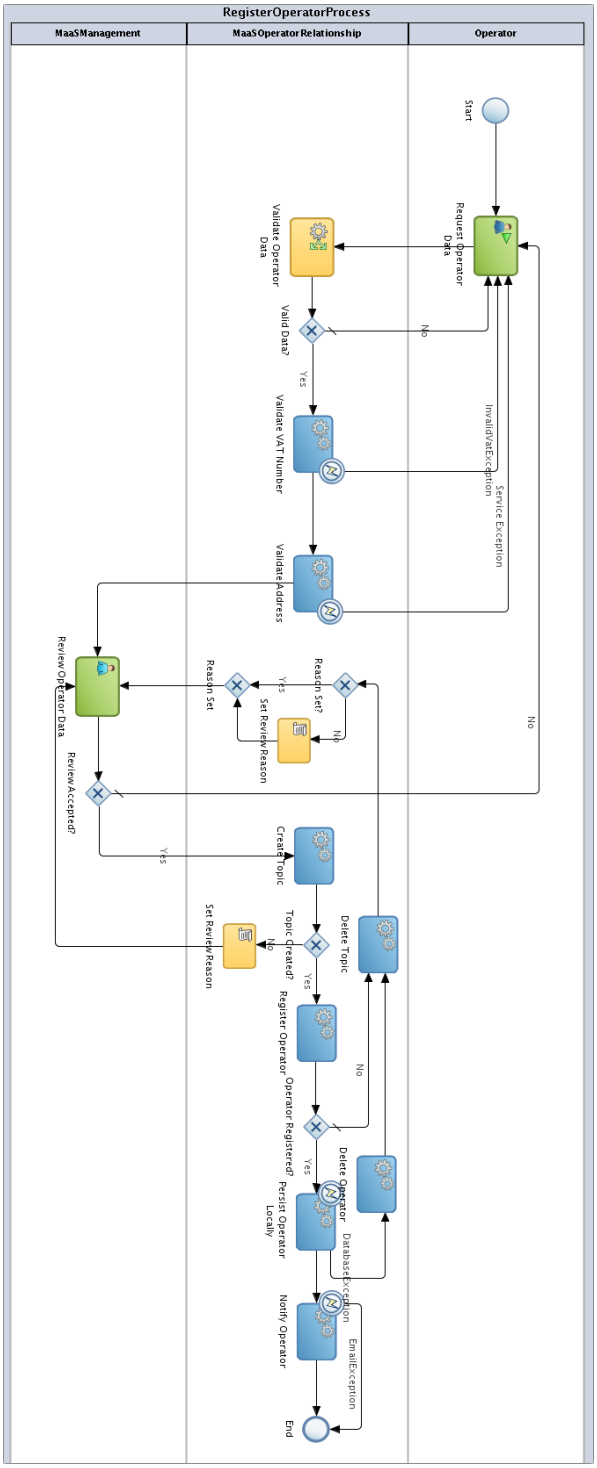

Fig. 2: Business process for enrolling a transportation service provider

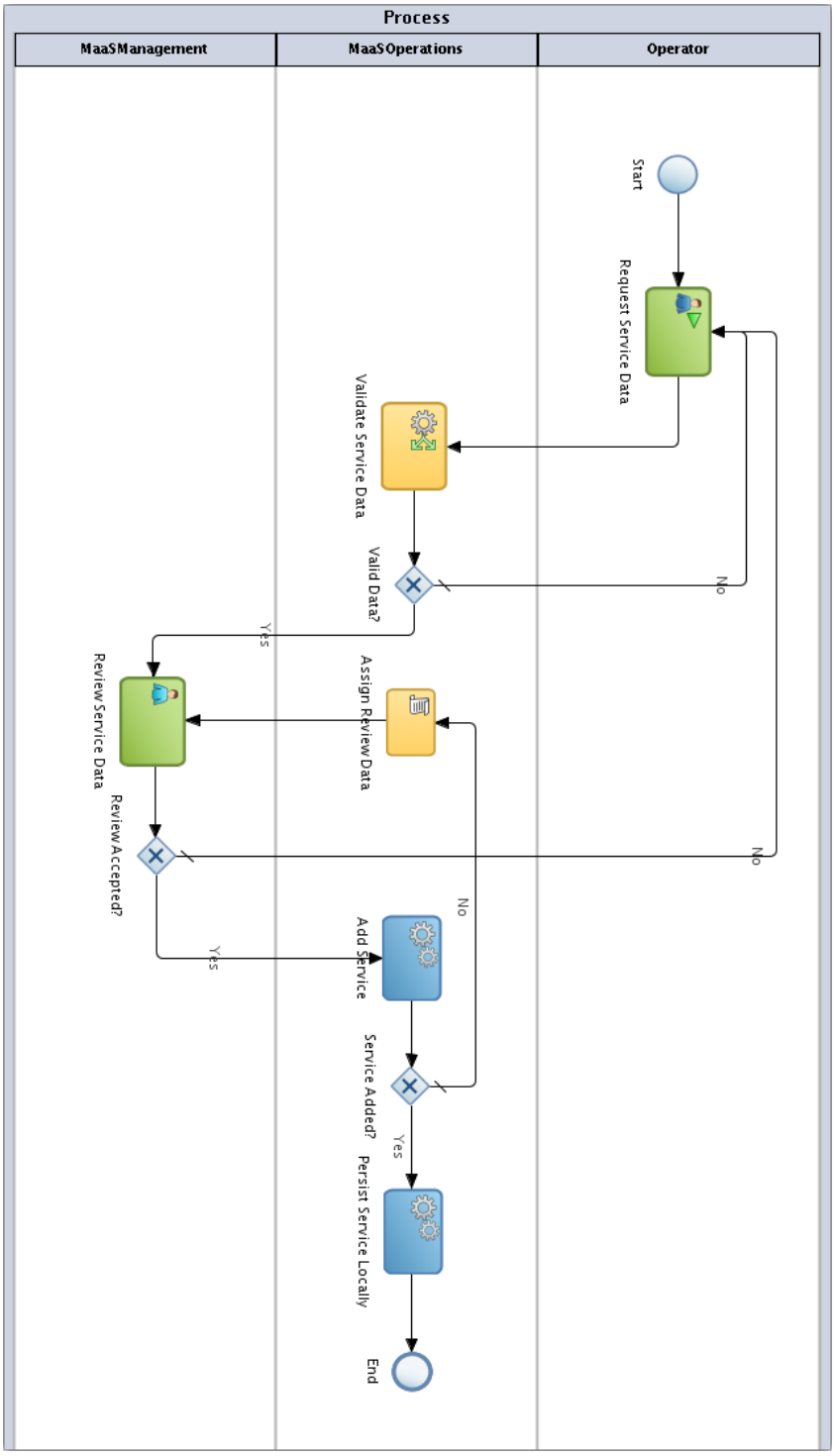

Fig. 3: Business process for enrolling a transportation service provider's product/service

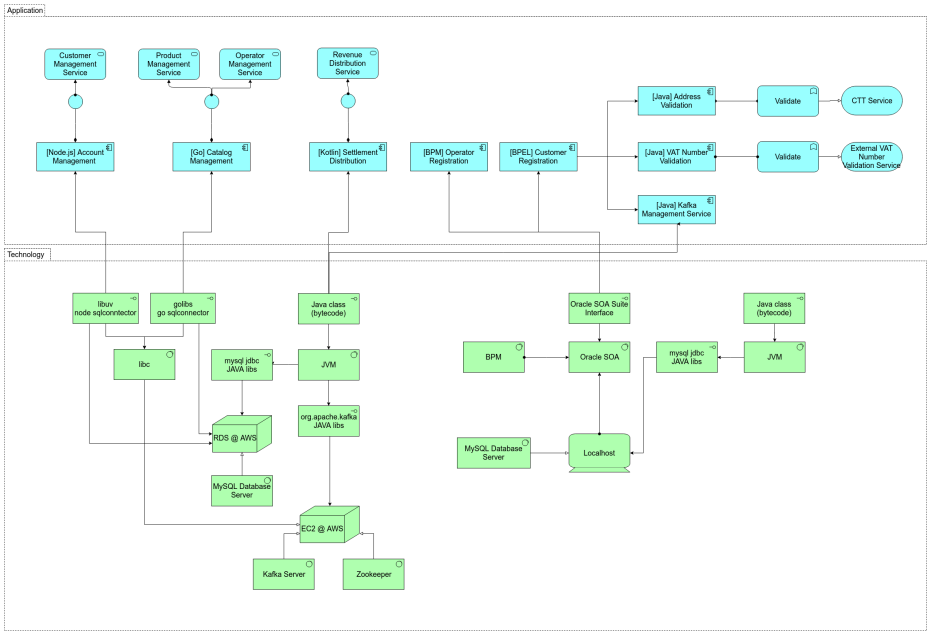

Fig. 4: Archimate model of the system architecture

6 Services

As previously mentioned, the platform is composed of a series of services crucial to the system's correct functionality. These come in the way of internal services hosted in the cloud and the ones used by the business processes, showcased in Figure 5.

6.1 Catalog Management

This service is tasked with keeping the set of products, their respective prices, and usage rules for each of the transportation operators' products, providing this information as a collection to the other services, or even the customer, when required.

6.2 Settlement Distribution

This service is responsible for registering trip events published by the user and appropriately distributing the revenue across the operators, taking into account several aspects that may influence the computation which can be service dependant, such as the trip's duration.

6.3 Account Management

This service handles customer related operations. It handles customer registration alongside the BPEL process showcased in Figure 1, persisting pertinent user information; it manages the user's finances inside the system, issuing debit orders and blacklisting the users whose balance prevents thJavaScript Object Notation (JSON)em from using the integrated transportation services. Furthermore, it is responsible for determining discount values for customers using the platform.

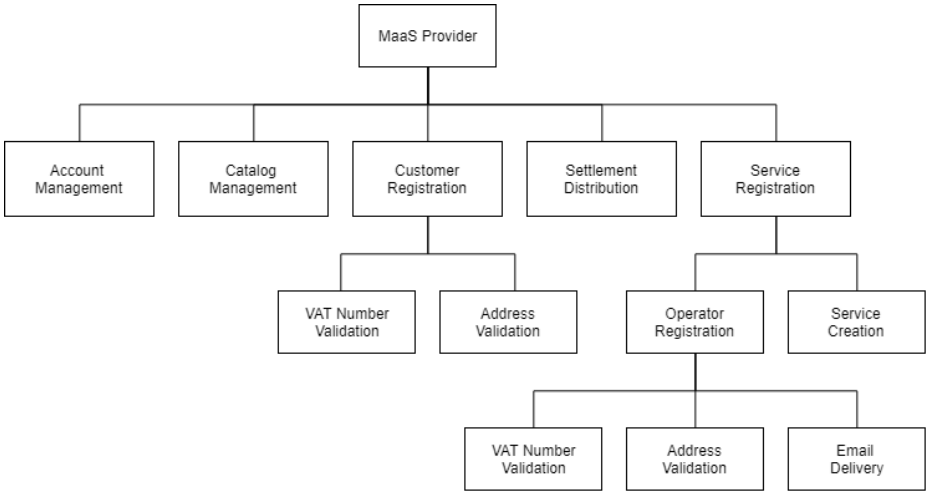

Fig. 5: MaaS platform service hierarchy

7 Data Schemas

The following figures illustrate the data being persisted on the services described in Section 6. As a sidenote, the business process persist operator and service data much akin to how the Catalog Management service does, as seen in Figure 7.

Additionally, the Settlement Distribution service persists events that come into the message broker (see Section 8) *quasi*-literally, using an Object-relational Mapping (ORM) framework, as it performs the required computation when they come in, using them later to distribute the revenue among operators. As such, instead of a relational database diagram, a Unified Modelling Language (UML) object diagram is presented (see Figure 8) to highlight similarities and relationships between objects hidden by the ORM.

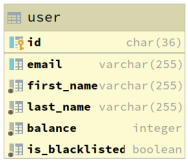

Fig. 6: Account Management service database schema

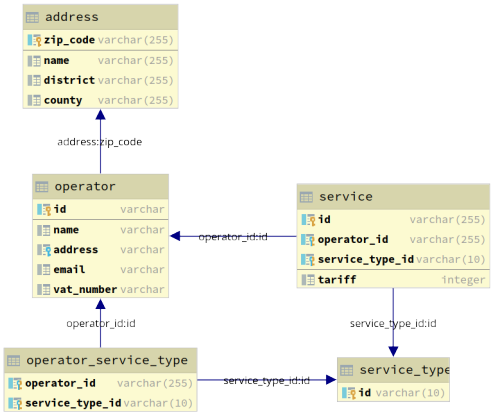

Fig. 7: Catalog Management service database schema

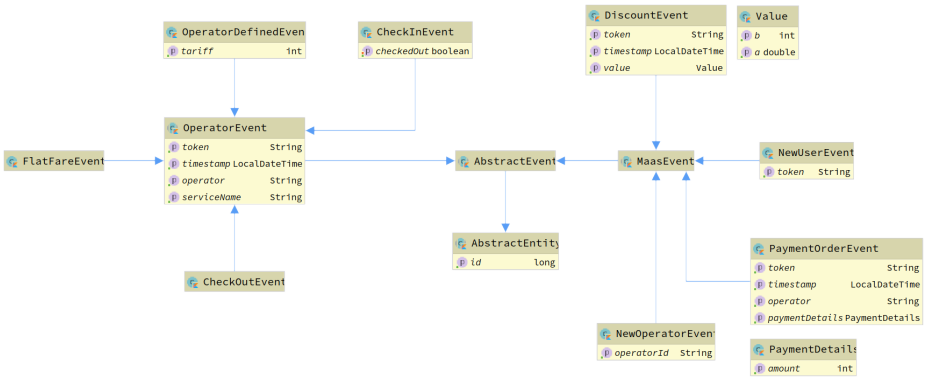

Fig. 8: Settlement Distribution service UML object diagram

8 Kafka

In order for transportation service providers to integrate with the platform, the system relies on a message broker in order to consume events emitted by operators and function properly.

8.1 Topics

The message broker will have as many topics as there are operators that are registered in the system, plus an additional topic for internal management. The name of the topic for each operator is chosen by the MaaS management revisor during registration, which is returned to the operator when registration goes through (see Figure 2).

For simplicity's sake, every topic only has a single partition and a replication factor of 1 since there is only one broker.

8.2 Events

Events are serialized as JavaScript objects, appropriately, in JSON. The following sections describe the events that arrive at the message broker, broken into two categories: (1) Operator events, which are the events that arrive at the operator specific topics, and (2) MaaS events, these being the internal events.

8.2.1 Operator Events

The general schema for operator events is as follows:

`type`	String - one of: `flat-fare`, `check-in check-out`, `operator-defined`
`token`	UUIDv4 - user token
`service_name`	String - Name of the service
`timestamp`	ISO 8601 UTC timestamp
`tariff`	Integer (optional, only in `operator-defined`)

An example can be seen in Listing 1.

8.2.2 MaaS Events Internal events are slightly more disparate from each other in terms of structure.

The schema for a new user event is as follows:

`type`	String - `new-user`
`user`	User object

The user object is described as follows:

`id`	UUIDv4 - user token
`email`	String
`first_name`	String
`last_name`	String

An example can be seen in Listing 2.
The schema for a new operator event is as follows:

`type`	String - `new-operator`
`operator`	String - operator identifier, i.e., topic name

An example can be seen in Listing 3.
The schema for a payment order event is as follows:

`type`	String - `payment-order`
`token`	UUIDv4 - user token
`payment-details`	Payment Details object

The payment details object is described as follows:

`amount`	Integer - amount user has to pay

An example can be seen in Listing 4.
The schema for a blacklist/whitelist events is as follows:

`type`	String - `blacklist/whitelist`
`user`	UUIDv4 - user token

N.B.: Unlike the other events described in this section, this event isn't sent to the internal MaaS system topic. It is sent to the operator topics so information about the blacklisted/whitelisted user can be propagated throughout the respective operators' systems. An example can be seen in Listing 5.
The schema for a discount event is as follows:

`type`	String - `blacklist/whitelist`
`user`	UUIDv4 - user token
`value`	Value object

The value object is described as follows:

`a`	Decimal - a ratio applied to the base value
`b`	Integer - flat discount

An example can be seen in Listing 6.

```
{
  "type": "flat-fare",
  "token": "8673b11b-0d3d-4870-8748-43a1acfd73e1",
  "service_name": "Passe mensal",
  "timestamp": "2019-05-24T17:42:55.204",
  "tariff": 3000
}
```

Listing 1: General example of all operator events

```
{
  "type": "new-user",
  "user": {
    "id": "69c594b8-9e87-45cd-a188-cfdeeaedd220",
    "email": "my.email@mail.com",
    "first_name": "Foo",
    "last_name": "Bar"
  }
}
```

Listing 2: New user event example

```
{
  "type": "new-operator",
  "operator": "carris"
}
```

Listing 3: New operator event example

```
{
  "type": "payment-order",
  "token": "1c0a3c6b-9405-4219-b550-fb28413a9b83",
  "payment_details": {
    "amount": 3000
  }
}
```

Listing 4: Payment order event example

```
{
  "type": "blacklist",
  "user": "49b5d29f-43fb-47ad-9ef9-69ba9680c16a"
}
```

Listing 5: Blacklist/Whitelist event example

```
{
  "type": "discount",
  "token": "7aef23eb-136a-4132-b7b4-bd0e6cc2f8b1",
  "timestamp": "2019-05-16T22:15:41.501",
  "value": {
    "a": 0.9,
    "b": 500
  }
}
```

Listing 6: Discount event example