Publisher

GET *http://server:port/publisher*

Descrição: Retorna uma lista com todos os publishers disponíveis no middleware.

Parâmetros: void.

Retorno:

```
{
  status: {
    error: false,
    message: null
  },
  data: {
    publishers: [
      { description: String, status: String, active: Boolean, _id: String },
      {...}
    ]
  }
}
```

GET *http://server:port/publisher/id*

Descrição: Retorna informações do publisher que corresponde ao id informado.

Parâmetros: String. ID do publisher.

Retorno:

```
{
  status: {
    error: Boolean,
    message: String
  },
  data: {
    publisher: { description: String, status: String, active: Boolean, _id: String }
  }
}
```

POST *http://server:xxxx/publisher*

Descrição: Adiciona um publisher ao ambiente.

Parâmetros: Object. Detalhes:

```
{
  description: String,
  status: String,
  active: Boolean
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    key: String
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

PUT *http://server:xxxx/publisher/id*

Descrição: Atualiza um determinado publisher.

Parâmetros: Um String para o id do publisher (GET). E também um Object com os campos a serem atualizados (POST):

```
{
  description: String, # Optional
  status: String, # Optional
  active: Boolean # Optional
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    publisher: { description: String, status: String, active: Boolean, _id: String }
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

DELETE *http://server:port/publisher/id*

Descrição: Remove um determinado publisher do ambiente.

Parâmetros: String. ID do publisher.

Retorno:

```
{
  status: {
    error: Boolean,
    message: String
  }
}
```

Subscriber

GET *http://server:port/subscriber*

Descrição: Retorna uma lista com todos os subscribers disponíveis no middleware.

Parâmetros: void.

Retorno:

```
{
  status: {
    error: false,
    message: null
  },
  data: {
    subscribers: [
      { description: String, domain: String, port: Number, path: String, method: String, active: Boolean,
_id: String },
      {...}
    ]
  }
}
```

GET *http://server:port/subscriber/id*

Descrição: Retorna informações do subscriber que corresponde ao id informado.

Parâmetros: String. ID do subscriber.

Retorno:

```
{
  status: {
    error: Boolean,
    message: String
  },
  data: {
    subscriber: { description: String, domain: String, port: Number, path: String, method: String, active:
Boolean, _id: String }
  }
}
```

POST *http://server:xxxx/subscriber*

Descrição: Adiciona um subscriber ao ambiente.

Parâmetros: Object. Detalhes:

```
{
  description: String,
  domain: String,
  port: Number, # default 80
  path: String,
  method: String, # default POST
  active: Boolean # default true
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    key: String
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

PUT *http://server:xxxx/subscriber/id*

Descrição: Atualiza dados de uma determinada aplicação.

Parâmetros: Um String para o id do subscriber em atualização (GET). E também um Object com os campos a serem atualizados (POST):

```
{
  description: String, # Optional
  domain: String, # Optional
  port: Number, # Optional
  path: String, # Optional
  method: String, # Optional
  active: Boolean # Optional
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    subscriber: { description: String, domain: String, port: Number, path: String, method: String, active:
Boolean, _id: String }
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

DELETE *http://server:port/subscriber/id*

Descrição: Remove um subscriber do ambiente.

Parâmetros: String. ID do subscriber.

Retorno:

```
{
  status: {
    error: Boolean,
    message: String
  }
}
```

Contract

GET *http://server:port/contract*

Descrição: Retorna uma lista com todos os contratos firmados entre publishers e subscribers.

Parâmetros: void.

Retorno:

```
{
  status: {
    error: false,
    message: null
  },
  data: {
    contracts: [
      { publisher_id: String, subscriber_id: String, _id: String },
      {...}
    ]
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

GET *http://server:port/contract/id*

Descrição: Retorna uma lista com todos os contratos do publisher identificado pelo ID informado.

Parâmetros: String. ID do publisher.

Retorno:

```
{
  status: {
    error: false,
    message: null
  },
  data: {
    contracts: [
      { publisher_id: String, subscriber_id: String, _id: String },
      {...}
    ]
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

POST *http://server:xxxx/contract*

Descrição: Assina um contrato entre um publisher e um subscriber.

Parâmetros: Object. Detalhes:

```
{
  publisher_id: String,
  subscriber_id: String
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    key: String
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).

DELETE *http://server:port/subscriber/id*

Descrição: Encerra um determinado contrato entre um publisher e subscriber.

Parâmetros: String. ID do subscriber.

Object. Detalhes:

```
{
  publisher_id: String
}
```

Retorno:

```
{
  status: {
    error: Boolean,
    message: String
  }
}
```

Communicator

POST *http://server:xxxx/communicator*

Descrição: Publica uma mensagem/evento de um publisher, para que o middleware distribua para todos os subscribers assinantes desse publisher.

Parâmetros: Object. Detalhes:

```
{
  publisher_id: String,
  data: Object # json, string, etc…
}
```

Retorno:

```
{
  status: {
    error: Bolean,
    message: String
  },
  data: {
    subscribers: [
      String,
      …
      String
    ]
  }
}
```

Obs.: Será retornado o data: { ... } apenas se não ocorrer erro (status.error == false).