Booking process

Step 1) Send authorization token by email or sms

`URL: POST /api/{webentities}/login/request/email`
Request parameters:

```
Email: string
webentities: string
```

`URL: POST /api/{webentities}/login/request/sms`
Request parameters:

```
PhoneNumber : string
CountryCode : string
webentities: string
```

- An authorization token is sent to your email or mobile.

Step 2) Token validation which we received in the previous step

`URL: POST /api/{webentities}/login/request/email`
Request parameters:

```
    Email: string
    webentities: string
```

`URL: POST /api/{webentities}/login/request/sms`
Request parameters:

```
    PhoneNumber : string
    CountryCode : string
    webentities: string
```

- In response we receive user information if the user exists in the database and status Accepted if the user doesn`t exists in the database, but token correct.

Step 3) Create reservation

You can get webProducts from the endpoint `/api/{webentities}/webproducts/{from}/{to}` or `/api/{webentities}/webproducts/{from}/{to}/{webProductId}`

`URL: POST /api/{webentities}/reservations`
Request parameters:

```
    model:
        {
            "webProductId": 0,
            "fromDate": "2019-12-18T11:03:14.383Z",
            "toDate": "2019-12-18T11:03:14.383Z",
            "priceId": 0,
            "numberOfPeople": 0,
            "additionalServices": [
                {
                    "id": 0,
                    "encryptedCompanyId": "string",
                    "count": 0
                }
            ],
            "additionalMerchandises": [
                {
                    "id": 0,
                    "encryptedCompanyId": "string",
                    "count": 0
                }
            ]
        }
    webentities: string
```

- In response we receive `reservationId` and `encryptedCompanyId`

Important : We have job that removes old reservations every 60sec.

- After doing reservation you should update it status to alive at least every 59 seconds (but better 30-40sec).
- You can update it status by API: `POST /api/{webentities}/reservations/ping` and send array of `[reservationId, encryptedCompanyId]`

Step 4) On checkout step we need to show all available company payment types for user and he can choose one of them

`URL: GET /api/{webentities}/checkout/paymentTypes`

In response we receive array of available payment types

Step 5) Perform checkout

- Email or phone should be verified. If user is not registered, he will be registered using information from request.
- If user exists, his data will be updated using information from request. User should accept terms from `POST /api/{webEntities}/setup/terms`

`URL: POST /api/{webentities}/checkout`
Request parameters:

```
        model:
        {
            "reservations": [
                {
                    "reservationId": 0,
                    "encryptedCompanyId": "string"
                }
            ],
            "successUrl": "string",
            "errorUrl": "string",
            "paymentType": "string",
            "amount": 0,
            "acceptedTerms: "bool",
            "customer": {
                "company": "string",
                "city": "string",
                "country": 0,
                "firstName": "string",
                "lastName": "string",
                "address": "string",
                "email": "string",
                "phone": "string",
                "zipCode": "string",
                "mobile": "string"
            }
        }
    webentities: string
```

- Response depends on paymentType. If it is main payment process (netAxept) we receive URL for payment (terminalUrl). In other cases - we receive status of payment.

Step 6) Go to the URL for payment and pay the reservation(s).

- After payment user will be redirected on the `successUrl` which we specified in the step 4 or `errorUrl` if something went wrong