

# Challenge

---

## RESTFul API

**Chess iT International B.V.**

Lichtfabriekplein 1

2031 TE Haarlem

P.O. Box 2031

2002 CA Haarlem

The Netherlands

T +31 (0)88 248 16 32

Bank 13.94.79.104

K.v.K. 53479335

BTW NL8508.95.698B01

## Document info

Date	
Title	Restful API
Version	0.1
Author	Mark Wigmans
Status	Draft
Chess iX reference	

## Document history

Version	Date	Author	Status	Remarks

## Reviews

Version	Date	Review by	Remarks

## 1 Introduction

In the Challenge a RESTful API is used. To make the exact usage of the API more clear a working demonstrator is being developed. In this document we describe every call and provide an example as well.

## 2 Interface

The whole interface is divided into the different phases on an event:

- Create stadium
- Sell tickets
- Report event selling results.

### 2.1 Create Stadium Phase

#### 2.1.1 Add Stadium

With the 'add' call a stadium is introduced into the system.

Field	Sub field	Value	Description
HTTP		POST	
URL		/stadiums	
Body			
	id	<String>	Unique ID of the event.
	name	<String>	Human readable of the event.
Response			Json of created stadium.

Example

- Create stadium with ID '123' and readable name 'ArenaA';
- Request: POST /stadiums
- Request body: `{"id": "123", "name": "ArenaA"}`
- Response :
  - HTTP OK (200)
  - `{"id": "123", "name": "ArenaA"}`

#### 2.1.2 Create Pricelist

With 'create pricelist' a overall usable pricelist will be created.

Field	Sub field	Value	Description
HTTP		POST	
URL		/pricelists	
Body			
	id	<String>	Unique ID of the pricelist.
	name	<String>	Human readable of the pricelist.
	pricecategories	Map	The map maps a category to a pricelist.

Field	Sub field	Value	Description
Response			Json of created pricelist.

#### Example

- Create a pricelist with ID 'p1' and prices for the categories ['seniors', 'adults', 'kids'];
- Request: POST /pricelists
- Request body :  

```
{ "id": "p1", "name": "p1", "pricecategories": { "seniors": 120, "adults": 130, "kids": 110 } }
```
- Response :
  - HTTP OK (200)
  - ```
{ "id": "p1", "name": "p1", "pricecategories": { "seniors": 120, "adults": 130, "kids": 110 } }
```

### 2.1.3 Create Block

Create a block within a stadium.

| Field    | Sub field    | Value                 | Description                            |
|----------|--------------|-----------------------|----------------------------------------|
| HTTP     |              | POST                  |                                        |
| URL      |              | /stadiums/<sid>/block |                                        |
|          | sid          | <String>              | Reference to stadium with its ID.      |
| Body     |              |                       |                                        |
|          | id           | <String>              | Unique ID of the block                 |
|          | name         | <String>              | Human readable of the block.           |
|          | rows         | <Int>                 | Number of rows in this block.          |
|          | seats        | <int>                 | Number of seats per row in this block. |
|          | defaultprice | <String>              | Reference to pricelist with its ID.    |
| Response |              |                       | Json of created block                  |

#### Example

- Create for stadium 's1' a block with id 'b1', with 10 rows and 20 seats per row. As default pricelist 'p3' will be used;
- Request: POST /stadiums/s1/block
- Request body :  

```
{ "id": "b2", "name": "b2", "rows": 10, "seats": 20, "defaultPrice": "p3" }
```
- Response :
  - HTTP OK (200)

- {"id": "b2", "name": "b2", "rows": 10, "seats": 20, "default Price": "p3", "availability": 200}

#### 2.1.4 Update Block

Once a block is created, it can be changed with this call.

| Field       | Sub field | Value                                             | Description                            |
|-------------|-----------|---------------------------------------------------|----------------------------------------|
| HTTP        |           | POST                                              |                                        |
| URL         |           | /stadiums/<sid>/block/<bid>/row/<row>/seat/<seat> |                                        |
|             | sid       | <String>                                          | Reference to stadium with its ID.      |
|             | bid       | <String>                                          | Reference to block with its ID.        |
|             | row       | <Int>                                             | Reference or row number                |
|             | seat      | <Int>                                             | Reference to seat number in given row. |
| <b>Body</b> |           |                                                   |                                        |
|             | Available | <Boolean>                                         | If given row/seat is available.        |
|             | Pricelist | <String>                                          | Reference to pricelist with its ID.    |
| Response    |           |                                                   | Json of updated seat.                  |

Example 1:

- Make 4<sup>th</sup> seat on row 3 unavailable of block 'b1' in stadium 's2';
- Request:
  - POST /stadiums/s2/block/b1/row/3/seat/4
  - {"available": false}
- Response
  - HTTP OK (200)
  - {"row": 3, "seat": 4, "available": false, "priceList": ""}

Example 2:

- Change the pricelist of 4<sup>th</sup> seat on row 3 of block 'b1' in stadium 's2' to pricelist 'p4';
- Request:
  - POST /stadiums/s2/block/b1/row/3/seat/4
  - {"pricelist": "p4"}
- Response
  - HTTP OK (200)
  - {"row": 3, "seat": 4, "available": true, "priceList": "4"}

## 2.2 Sell Tickets Phase

### 2.2.1 Check availability

Sell tickets from a given block within a stadium.

| Field    | Sub field | Value            | Description                                                                   |
|----------|-----------|------------------|-------------------------------------------------------------------------------|
| HTTP     |           | GET              |                                                                               |
| URL      |           | /available/<sid> |                                                                               |
|          | sid       | <String>         | Reference to stadium with its ID.                                             |
| Response |           |                  | Json of found tickets or empty list if no / not enough tickets are available. |

Example

- Request the availability for stadium 's1'
- Request:
  - GET /available/:sid
- Response
  - HTTP OK (200)
  - [{"id":"b21","count":150},{ "id":"b22","count":100}]

### 2.2.2 Sell Tickets

Sell tickets from a given block within a stadium.

| Field    | Sub field      | Value                    | Description                                                                   |
|----------|----------------|--------------------------|-------------------------------------------------------------------------------|
| HTTP     |                | POST                     |                                                                               |
| URL      |                | /tickets/buy/<sid>/<bid> |                                                                               |
|          | sid            | <String>                 | Reference to stadium with its ID.                                             |
|          | bid            | <String>                 | Reference to block with its ID.                                               |
| Body     |                |                          |                                                                               |
|          | Request-id     | <String>                 | Unique ID of ticket request                                                   |
|          | Ticket-request | Array of category        |                                                                               |
| Response |                |                          | Json of found tickets or empty list if no / not enough tickets are available. |

#### Example

- Request 2 kid tickets, 1 adult ticket and 1 senior ticket. The tickets are available;

- Request:

- POST /tickets/buys2/b1

- ```
{ "request-id": "147c98ad-a02f-4089-aec9-6ab30cc5848f", "ticket-request": [ "kids", "kids", "adults", "seniors" ] }
```

- Response

- HTTP OK (200)

- ```
[ { "requestId": "147c98ad-a02f-4089-aec9-6ab30cc5848f", "seat": { "row": 1, "seat": 1, "available": true, "priceList": "3" }, "category": "kids", "price": 110 }, { "requestId": "147c98ad-a02f-4089-aec9-6ab30cc5848f", "seat": { "row": 1, "seat": 2, "available": true, "priceList": "3" }, "category": "kids", "price": 110 }, { "requestId": "147c98ad-a02f-4089-aec9-6ab30cc5848f", "seat": { "row": 1, "seat": 3, "available": true, "priceList": "3" }, "category": "adults", "price": 210 }, { "requestId": "147c98ad-a02f-4089-aec9-6ab30cc5848f", "seat": { "row": 1, "seat": 4, "available": true, "priceList": "3" }, "category": "seniors", "price": 130 } ]
```

## 2.3 Reporting Phase

### 2.3.1 Count All Sold Tickets

Return the number of all tickets sold.

| Field    | Sub field | Value          | Description            |
|----------|-----------|----------------|------------------------|
| HTTP     |           | GET            |                        |
| URL      |           | /tickets/count |                        |
| Response |           |                | Number of sold tickets |

#### Example

- Count all tickets sold;
- Request:
  - POST / tickets/count
- Response
  - HTTP OK (200)
  - ```
{ "count " : 7 }
```

### 2.3.2 Count Sold Tickets of a Stadium

Return the number of all tickets sold for a given stadium.



Field	Sub field	Value	Description
HTTP		GET	
URL		/tickets/count/<sid>	
	sid	<String>	Reference to stadium with its ID.
Response			Number of sold tickets

#### Example

- Count all tickets sold of stadium 's1';
- Request:
  - POST / tickets/count/s1
- Response
  - HTTP OK (200)
  - ```
{ "count " : 4 }
```

### 2.3.3 Show Sold Tickets of a Stadium

Return the list of all tickets sold for a given stadium.

| Field    | Sub field | Value                     | Description                                    |
|----------|-----------|---------------------------|------------------------------------------------|
| HTTP     |           | GET                       |                                                |
| URL      |           | /tickets/<sid>?index&page |                                                |
|          | index     | <Int>                     | Index of one of the chunks, starting by '0'.   |
|          | Page      | <Int>                     | Partitioned into chunks of the specified size. |
| Response |           |                           | Number of sold tickets                         |

#### Example

- Count all tickets sold of stadium 's1' and show the result of page size '2' and show the first page
- Request:
  - POST / tickets/s1?index=0&page=2
- Response
  - HTTP OK (200)
  - ```
[{"requestId":"bfd891c1-d51c-459a-a82f-1ddda3a3a06c","seat":{"row":1,"seat":3,"available":true,"priceList":"3"},"category":"adults","price":210}, {"requestId":"bfd891c1-d51c-459a-a82f-1ddda3a3a06c","seat":{"row":1,"seat":4,"available":true,"priceList":"3"},"category":"seniors","price":130}]
```

## 2.4 Miscellaneous

### 2.4.1 Start Selling Tickets

The stadium is created, so the signal is given from the client that the selling can be started.

Field	Sub field	Value	Description
HTTP		POST	
URL		/start/<sid>	
	sid	<String>	Reference to stadium with its ID.
Response		-	

### 2.4.2 Stop Selling Tickets

The Selling is ended, so report calls can be expected.

Field	Sub field	Value	Description
HTTP		POST	
URL		/stop/<sid>	
	sid	<String>	Reference to stadium with its ID.
Response		-	

### 2.4.3 Reset the Simulation

The whole simulation is reset to do another complete run.

Field	Sub field	Value	Description
HTTP		POST	
URL		/reset	
Response		-	