Restaurant Manager API documentation

Content

1. Endpoints

1.1. Admin

1.1.1. Profile

GET `/rest/admin/users` *Get all*

GET `/rest/admin/users/{id}` *Get one*

GET `/rest/admin/users/by` *Get by email*

POST `/rest/admin/users` *Create*

PUT `/rest/admin/users` *Update*

PATCH `/rest/admin/menus/{id}` *Set enable*

DELETE `/rest/admin/users/{id}` *Delete*

1.1.2. Restaurant

GET `/rest/admin/menus/{restId}` *Get all for one restaurant*

GET `/rest/admin/menus/filter/{restId}` *Get filtered for one restaurant*

GET `/rest/admin/menus/{restId}/{id}` *Get one*

POST `/rest/admin/menus/{restId}` *Create*

PUT `/rest/admin/menus/{restId}` *Update*

PATCH `/rest/admin/menus/{restId}/{id}` *Set enable*

DELETE `/rest/admin/menus/{restId}/{id}` *Delete*

1.1.3. Menu

GET `/rest/admin/menus/{restId}` *Get all for one restaurant*

GET `/rest/admin/menus/filter/{restId}` *Get filtered for one restaurant*

GET `/rest/admin/menus/{restId}/{id}` *Get one*

Path

Table 1. /rest/admin/menus/{restId}/{id}

Parameter	Description
`restId`	Restaurant id
`id`	Menu id

Responce body

Table 2. /rest/admin/menus/{restId}/{id}

Parameter	Description
`restId`	Restaurant id
`id`	Menu id

Examples

Curl

```
$ curl 'http://localhost:8080/rest/admin/menus/0/0' -i -u
'admin@gmail.com:admin' -X GET
```

HTTP Request

```
GET /rest/admin/menus/0/0 HTTP/1.1
Authorization: Basic YWRtaW5AZ21haWwuY29tOmFkbWlu
Host: localhost:8080
```

Success HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 191

{"id":0,"name":"menu1","menuDate":"2020-01-
27","enabled":false,"menuItems":[{"id":0,"name":"menItm1","price":100},{"id":
1,"name":"menItm2","price":101},{"id":2,"name":"menItm3","price":102}]}
```

Error HTTP Responses

Get not found

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 210

{"url":"http://localhost:8080/rest/admin/menus/0/15","type":"DATA_NOT_FOUND",
"detail":"edu.volkov.restmanager.util.exception.NotFoundException: Not found
entity with menu by id: 15 for restId: 0 dos not exist"}
```

Get by unauthorized user

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Realm"
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
```

Get by forbidden user

```
HTTP/1.1 403 Forbidden
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
```

POST `/rest/admin/menus/{restId}` *Create*

PUT `/rest/admin/menus/{restId}` *Update*

PATCH `/rest/admin/menus/{restId}/{id}` *Set enable*

DELETE `/rest/admin/menus/{restId}/{id}` *Delete*

1.2. Authenticated User

1.2.1. Profile

GET `/rest/profile` *Get authenticated*

GET `/rest/profile/test` *Test UTF*

POST `/rest/profile/register` *Register*

PUT `/rest/profile` *Update authenticated*

DELETE `/rest/profile` *Delete authenticated*

1.2.2. Vote

GET `/rest/profile/votes/{restId}` *Vote for restaurant*

1.3. User

1.3.1. Restaurant

GET `/rest/admin/menus/{restId}` *Get all for one restaurant*

GET `/rest/admin/menus/filter/{restId}` *Get filtered for one restaurant*

GET `/rest/admin/menus/{restId}/{id}` *Get one*

POST `/rest/admin/menus/{restId}` *Create*

PUT `/rest/admin/menus/{restId}` *Update*

PATCH `/rest/admin/menus/{restId}/{id}` *Set enable*

DELETE `/rest/admin/menus/{restId}/{id}` *Delete*

2. Convention

2.1. HTTP verbs

Restaurant Manager tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs.

Verb	Usage
`GET`	Used to retrieve a resource
`POST`	Used to create a new resource
`PATCH`	Used to update an existing resource, including partial updates
`PUT`	Used to update an existing resource, full updates only
`DELETE`	Used to delete an existing resource

2.2. HTTP status codes

Restaurant Manager tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes.

Status code	**Usage**
`200 OK`	Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action.
`201 Created`	The request has been fulfilled and resulted in a new resource being created.
`204 No Content`	The server successfully processed the request, but is not returning any content.
`400 Bad Request`	The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).
`404 Not Found`	The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.
`409 Conflict`	The request could not be completed due to a conflict with the current state of the target resource.
`422 Unprocessable Entity`	Validation error has happened due to processing the posted entity.