Api Documentation

Table of Contents

Chapter 1. Overview

Api Documentation

1.1. Version information

Version : 1.0

1.2. License information

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0
Terms of service : urn:tos

1.3. URI scheme

Host : localhost
BasePath : /

1.4. Tags

- Address Master Data : Address Master Data Controller

- Authorization : Authentication Controller

- I18n : I 18 N Controller

- Internment Patient : Internment Patient Controller

- Password : User Password Controller

- Patient : Patient Controller

- PatientMasterData : Patient Master Data Controller

- Person : Person Controller

- Role : Role Controller

- User : User Controller

- backoffice-address-controller : Backoffice Address Controller

- backoffice-bed-controller : Backoffice Bed Controller

- backoffice-city-controller : Backoffice City Controller

- backoffice-clinical-specialty-controller : Backoffice Clinical Specialty Controller

- backoffice-clinical-specialty-sector-controller : Backoffice Clinical Specialty Sector Controller

- backoffice-department-controller : Backoffice Department Controller

- backoffice-institution-controller : Backoffice Institution Controller

- backoffice-province-controller : Backoffice Province Controller

- backoffice-room-controller : Backoffice Room Controller

- backoffice-sector-controller : Backoffice Sector Controller

- basic-error-controller : Basic Error Controller

- operation-handler : Operation Handler

- web-mvc-links-handler : Web Mvc Links Handler

Chapter 2. Paths

2.1. links

```
GET /actuator
```

2.1.1. Responses

HTTP Code	Description	Schema
200	OK	< string, < string, Link > map > map

2.1.2. Produces

- application/vnd.spring-boot.actuator.v3+json
- application/json
- application/vnd.spring-boot.actuator.v2+json

2.1.3. Tags

- web-mvc-links-handler

2.1.4. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.2. handle

```
GET /actuator/health
```

2.2.1. Parameters

Type	Name	Description	Schema
Body	**body** *optional*	body	< string, string > map

2.2.2. Responses

HTTP Code	Description	Schema
200	OK	object

2.2.3. Produces

- application/vnd.spring-boot.actuator.v3+json
- application/json
- application/vnd.spring-boot.actuator.v2+json

2.2.4. Tags

- operation-handler

2.2.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.3. handle

```
GET /actuator/health/**
```

2.3.1. Parameters

Type	Name	Description	Schema
Body	**body** *optional*	body	< string, string > map

2.3.2. Responses

HTTP Code	Description	Schema
200	OK	object

2.3.3. Produces

- application/vnd.spring-boot.actuator.v3+json

- application/json
- application/vnd.spring-boot.actuator.v2+json

2.3.4. Tags

- operation-handler

2.3.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.4. handle

```
GET /actuator/info
```

2.4.1. Parameters

Type	Name	Description	Schema
Body	**body** *optional*	body	< string, string > map

2.4.2. Responses

HTTP Code	Description	Schema
200	OK	object

2.4.3. Produces

- application/vnd.spring-boot.actuator.v3+json
- application/json
- application/vnd.spring-boot.actuator.v2+json

2.4.4. Tags

- operation-handler

2.4.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.5. getAllCountries

```
GET /address/masterdata/countries
```

2.5.1. Responses

HTTP Code	Description	Schema
200	OK	< AddressProjection > array

2.5.2. Produces

- */*

2.5.3. Tags

- Address Master Data

2.5.4. Security

Type	Name	Scopes
apiKey	JWT	global

2.6. getByCountry

```
GET /address/masterdata/country/{countryId}/provinces
```

2.6.1. Parameters

Type	Name	Description	Schema
Path	**countryId** *required*	countryId	integer (int32)

2.6.2. Responses

HTTP Code	Description	Schema
200	OK	< AddressProjection > array

2.6.3. Produces

- */*

2.6.4. Tags

- Address Master Data

2.6.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.7. getCitiesByDepartment

```
GET /address/masterdata/department/{department_id}/cities
```

2.7.1. Parameters

Type	Name	Description	Schema
Path	**department_id** *required*	department_id	integer (int32)

2.7.2. Responses

HTTP Code	Description	Schema
200	OK	< AddressProjection > array

2.7.3. Produces

- */*

2.7.4. Tags

- Address Master Data

2.7.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.8. getCitiesByProvince

```
GET /address/masterdata/province/{provinceId}/cities
```

2.8.1. Parameters

Type	Name	Description	Schema
Path	**provinceId** *required*	provinceId	integer (int32)

2.8.2. Responses

HTTP Code	Description	Schema
200	OK	< AddressProjection > array

2.8.3. Produces

- */*

2.8.4. Tags

- Address Master Data

2.8.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.9. getDepartmentsByProvince

```
GET /address/masterdata/province/{provinceId}/departments
```

2.9.1. Parameters

Type	Name	Description	Schema
Path	**provinceId** *required*	provinceId	integer (int32)

2.9.2. Responses

HTTP Code	Description	Schema
200	OK	< AddressProjection > array

2.9.3. Produces

- */*

2.9.4. Tags

- Address Master Data

2.9.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.10. login

```
POST /auth
```

2.10.1. Parameters

Type	Name	Description	Schema
Body	**loginDto** *required*	loginDto	LoginDto

2.10.2. Responses

HTTP Code	Description	Schema
200	OK	JWTokenDto

2.10.3. Consumes

- `application/json`

2.10.4. Produces

- `*/*`

2.10.5. Tags

- Authorization

2.10.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.11. create

```
POST /backoffice/addresses
```

2.11.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Address

2.11.2. Responses

HTTP Code	Description	Schema
200	OK	Address

2.11.3. Consumes

- `application/json`

2.11.4. Produces

- `*/*`

2.11.5. Tags

- backoffice-address-controller

2.11.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.12. getMany

```
GET /backoffice/addresses
```

2.12.1. Parameters

Type	Name	Description	Schema
Query	**ids** *required*	ids	string

2.12.2. Responses

HTTP Code	Description	Schema
200	OK	IterableOfAddress

2.12.3. Produces

- */*

2.12.4. Tags

- backoffice-address-controller

2.12.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.13. getOne

```
GET /backoffice/addresses/{id}
```

2.13.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.13.2. Responses

HTTP Code	Description	Schema
200	OK	Address

2.13.3. Produces

- */*

2.13.4. Tags

- backoffice-address-controller

2.13.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.14. update

```
PUT /backoffice/addresses/{id}
```

2.14.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Address

2.14.2. Responses

HTTP Code	Description	Schema
200	OK	Address

2.14.3. Consumes

- application/json

2.14.4. Produces

- */*

2.14.5. Tags

- backoffice-address-controller

2.14.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.15. delete

```
DELETE /backoffice/addresses/{id}
```

2.15.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.15.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfAddress

2.15.3. Produces

- */*

2.15.4. Tags

- backoffice-address-controller

2.15.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.16. create

```
POST /backoffice/beds
```

2.16.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Bed

2.16.2. Responses

HTTP Code	Description	Schema
200	OK	Bed

2.16.3. Consumes

- application/json

2.16.4. Produces

- */*

2.16.5. Tags

- backoffice-bed-controller

2.16.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.17. getMany

```
GET /backoffice/beds
```

2.17.1. Parameters

Type	Name	Description	Schema
Query	**ids** *required*	ids	string

2.17.2. Responses

HTTP Code	Description	Schema
200	OK	IterableOfBed

2.17.3. Produces

- */*

2.17.4. Tags

- backoffice-bed-controller

2.17.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.18. getOne

```
GET /backoffice/beds/{id}
```

2.18.1. Parameters

Type	Name	Description	Schema
Path	id *required*	id	integer (int32)

2.18.2. Responses

HTTP Code	Description	Schema
200	OK	Bed

2.18.3. Produces

- */*

2.18.4. Tags

- backoffice-bed-controller

2.18.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.19. update

```
PUT /backoffice/beds/{id}
```

2.19.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Bed

2.19.2. Responses

HTTP Code	Description	Schema
200	OK	Bed

2.19.3. Consumes

- application/json

2.19.4. Produces

- */*

2.19.5. Tags

- backoffice-bed-controller

2.19.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.20. delete

```
DELETE /backoffice/beds/{id}
```

2.20.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.20.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfBed

2.20.3. Produces

- */*

2.20.4. Tags

- backoffice-bed-controller

2.20.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.21. create

```
POST /backoffice/cities
```

2.21.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	City

2.21.2. Responses

HTTP Code	Description	Schema
200	OK	City

2.21.3. Consumes

- `application/json`

2.21.4. Produces

- `*/*`

2.21.5. Tags

- backoffice-city-controller

2.21.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.22. getMany

```
GET /backoffice/cities
```

2.22.1. Parameters

Type	Name	Description	Schema
Query	**ids** *required*	ids	string

2.22.2. Responses

HTTP Code	Description	Schema
200	OK	IterableOfCity

2.22.3. Produces

- `*/*`

2.22.4. Tags

- backoffice-city-controller

2.22.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.23. getOne

```
GET /backoffice/cities/{id}
```

2.23.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.23.2. Responses

HTTP Code	Description	Schema
200	OK	City

2.23.3. Produces

- */*

2.23.4. Tags

- backoffice-city-controller

2.23.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.24. update

```
PUT /backoffice/cities/{id}
```

2.24.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	City

2.24.2. Responses

HTTP Code	Description	Schema
200	OK	City

2.24.3. Consumes

- `application/json`

2.24.4. Produces

- `*/*`

2.24.5. Tags

- backoffice-city-controller

2.24.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.25. delete

```
DELETE /backoffice/cities/{id}
```

2.25.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.25.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfCity

2.25.3. Produces

- */*

2.25.4. Tags

- backoffice-city-controller

2.25.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.26. create

```
POST /backoffice/clinicalspecialties
```

2.26.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	ClinicalSpecialty

2.26.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialty

2.26.3. Consumes

- `application/json`

2.26.4. Produces

- `*/*`

2.26.5. Tags

- backoffice-clinical-specialty-controller

2.26.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.27. getList

```
GET /backoffice/clinicalspecialties
```

2.27.1. Parameters

Type	Name	Schema
Query	**description** *optional*	string
Query	**id** *optional*	integer (int32)
Query	**name** *optional*	string
Query	**page** *optional*	integer (int32)
Query	**sctidCode** *optional*	string

Type	Name	Schema
Query	**size** *optional*	integer (int32)
Query	**sort** *optional*	string

2.27.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfClinicalSpecialty

2.27.3. Produces

- */*

2.27.4. Tags

- backoffice-clinical-specialty-controller

2.27.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.28. getOne

```
GET /backoffice/clinicalspecialties/{id}
```

2.28.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.28.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialty

2.28.3. Produces

- */*

2.28.4. Tags

- backoffice-clinical-specialty-controller

2.28.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.29. update

```
PUT /backoffice/clinicalspecialties/{id}
```

2.29.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	ClinicalSpecialty

2.29.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialty

2.29.3. Consumes

- application/json

2.29.4. Produces

- */*

2.29.5. Tags

- backoffice-clinical-specialty-controller

2.29.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.30. delete

```
DELETE /backoffice/clinicalspecialties/{id}
```

2.30.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.30.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfClinicalSpecialty

2.30.3. Produces

- */*

2.30.4. Tags

- backoffice-clinical-specialty-controller

2.30.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.31. create

```
POST /backoffice/clinicalspecialtysectors
```

2.31.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	ClinicalSpecialtySector

2.31.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialtySector

2.31.3. Consumes

- application/json

2.31.4. Produces

- */*

2.31.5. Tags

- backoffice-clinical-specialty-sector-controller

2.31.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.32. getMany

```
GET /backoffice/clinicalspecialtysectors
```

2.32.1. Parameters

Type	Name	Description	Schema
Query	**ids** *required*	ids	string

2.32.2. Responses

HTTP Code	Description	Schema
200	OK	IterableOfClinical SpecialtySector

2.32.3. Produces

- */*

2.32.4. Tags

- backoffice-clinical-specialty-sector-controller

2.32.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.33. getOne

```
GET /backoffice/clinicalspecialtysectors/{id}
```

2.33.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.33.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialtySector

2.33.3. Produces

- */*

2.33.4. Tags

- backoffice-clinical-specialty-sector-controller

2.33.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.34. update

```
PUT /backoffice/clinicalspecialtysectors/{id}
```

2.34.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	ClinicalSpecialtySector

2.34.2. Responses

HTTP Code	Description	Schema
200	OK	ClinicalSpecialtySector

2.34.3. Consumes

- `application/json`

2.34.4. Produces

- `*/*`

2.34.5. Tags

- backoffice-clinical-specialty-sector-controller

2.34.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.35. delete

```
DELETE /backoffice/clinicalspecialtysectors/{id}
```

2.35.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.35.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfClinicalSpecialtySector

2.35.3. Produces

- `*/*`

2.35.4. Tags

- backoffice-clinical-specialty-sector-controller

2.35.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.36. create

```
POST /backoffice/departments
```

2.36.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Department

2.36.2. Responses

HTTP Code	Description	Schema
200	OK	Department

2.36.3. Consumes

- application/json

2.36.4. Produces

- */*

2.36.5. Tags

- backoffice-department-controller

2.36.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.37. getList

```
GET /backoffice/departments
```

2.37.1. Parameters

Type	Name	Schema
Query	**description** *optional*	string
Query	**id** *optional*	integer (int32)
Query	**page** *optional*	integer (int32)
Query	**provinceId** *optional*	integer (int32)
Query	**size** *optional*	integer (int32)
Query	**sort** *optional*	string

2.37.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfDepartment

2.37.3. Produces

- */*

2.37.4. Tags

- backoffice-department-controller

2.37.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.38. getOne

```
GET /backoffice/departments/{id}
```

2.38.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.38.2. Responses

HTTP Code	Description	Schema
200	OK	Department

2.38.3. Produces

- */*

2.38.4. Tags

- backoffice-department-controller

2.38.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.39. update

```
PUT /backoffice/departments/{id}
```

2.39.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Department

2.39.2. Responses

HTTP Code	Description	Schema
200	OK	Department

2.39.3. Consumes

- `application/json`

2.39.4. Produces

- `*/*`

2.39.5. Tags

- backoffice-department-controller

2.39.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.40. delete

```
DELETE /backoffice/departments/{id}
```

2.40.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.40.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfDepartment

2.40.3. Produces

- */*

2.40.4. Tags

- backoffice-department-controller

2.40.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.41. create

```
POST /backoffice/institutions
```

2.41.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Institution

2.41.2. Responses

HTTP Code	Description	Schema
200	OK	Institution

2.41.3. Consumes

- application/json

2.41.4. Produces

- */*

2.41.5. Tags

- backoffice-institution-controller

2.41.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.42. getList

```
GET /backoffice/institutions
```

2.42.1. Parameters

Type	Name	Schema
Query	**addressId** *optional*	integer (int32)
Query	**cuit** *optional*	string
Query	**email** *optional*	string
Query	**id** *optional*	integer (int32)
Query	**name** *optional*	string
Query	**page** *optional*	integer (int32)
Query	**phone** *optional*	string

Type	Name	Schema
Query	**sisaCode** *optional*	string
Query	**size** *optional*	integer (int32)
Query	**sort** *optional*	string
Query	**website** *optional*	string

2.42.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfInstitution

2.42.3. Produces

- */*

2.42.4. Tags

- backoffice-institution-controller

2.42.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.43. getOne

```
GET /backoffice/institutions/{id}
```

2.43.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.43.2. Responses

HTTP Code	Description	Schema
200	OK	Institution

2.43.3. Produces

- **/**

2.43.4. Tags

- backoffice-institution-controller

2.43.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.44. update

```
PUT /backoffice/institutions/{id}
```

2.44.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Institution

2.44.2. Responses

HTTP Code	Description	Schema
200	OK	Institution

2.44.3. Consumes

- `application/json`

2.44.4. Produces

- `*/*`

2.44.5. Tags

- backoffice-institution-controller

2.44.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.45. delete

```
DELETE /backoffice/institutions/{id}
```

2.45.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.45.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfInstitution

2.45.3. Produces

- */*

2.45.4. Tags

- backoffice-institution-controller

2.45.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.46. create

```
POST /backoffice/provinces
```

2.46.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Province

2.46.2. Responses

HTTP Code	Description	Schema
200	OK	Province

2.46.3. Consumes

- application/json

2.46.4. Produces

- */*

2.46.5. Tags

- backoffice-province-controller

2.46.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.47. getList

```
GET /backoffice/provinces
```

2.47.1. Parameters

Type	Name	Schema
Query	countryId *optional*	integer (int32)
Query	description *optional*	string
Query	id *optional*	integer (int32)
Query	page *optional*	integer (int32)
Query	size *optional*	integer (int32)
Query	sort *optional*	string

2.47.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfProvince

2.47.3. Produces

- */*

2.47.4. Tags

- backoffice-province-controller

2.47.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.48. getOne

```
GET /backoffice/provinces/{id}
```

2.48.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.48.2. Responses

HTTP Code	Description	Schema
200	OK	Province

2.48.3. Produces

- */*

2.48.4. Tags

- backoffice-province-controller

2.48.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.49. update

```
PUT /backoffice/provinces/{id}
```

2.49.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Province

2.49.2. Responses

HTTP Code	Description	Schema
200	OK	Province

2.49.3. Consumes

- application/json

2.49.4. Produces

- */*

2.49.5. Tags

- backoffice-province-controller

2.49.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.50. delete

```
DELETE /backoffice/provinces/{id}
```

2.50.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.50.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfProvince

2.50.3. Produces

- */*

2.50.4. Tags

- backoffice-province-controller

2.50.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.51. create

```
POST /backoffice/rooms
```

2.51.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Room

2.51.2. Responses

HTTP Code	Description	Schema
200	OK	Room

2.51.3. Consumes

- `application/json`

2.51.4. Produces

- `*/*`

2.51.5. Tags

- backoffice-room-controller

2.51.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.52. getMany

```
GET /backoffice/rooms
```

2.52.1. Parameters

Type	Name	Description	Schema
Query	**ids** *required*	ids	string

2.52.2. Responses

HTTP Code	Description	Schema
200	OK	IterableOfRoom

2.52.3. Produces

- `*/*`

2.52.4. Tags

- backoffice-room-controller

2.52.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.53. getOne

```
GET /backoffice/rooms/{id}
```

2.53.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.53.2. Responses

HTTP Code	Description	Schema
200	OK	Room

2.53.3. Produces

- */*

2.53.4. Tags

- backoffice-room-controller

2.53.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.54. update

```
PUT /backoffice/rooms/{id}
```

2.54.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Room

2.54.2. Responses

HTTP Code	Description	Schema
200	OK	Room

2.54.3. Consumes

- application/json

2.54.4. Produces

- */*

2.54.5. Tags

- backoffice-room-controller

2.54.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.55. delete

```
DELETE /backoffice/rooms/{id}
```

2.55.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.55.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfRoom

2.55.3. Produces

- */*

2.55.4. Tags

- backoffice-room-controller

2.55.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.56. create

```
POST /backoffice/sectors
```

2.56.1. Parameters

Type	Name	Description	Schema
Body	**entity** *required*	entity	Sector

2.56.2. Responses

HTTP Code	Description	Schema
200	OK	Sector

2.56.3. Consumes

- `application/json`

2.56.4. Produces

- `*/*`

2.56.5. Tags

- backoffice-sector-controller

2.56.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.57. getList

```
GET /backoffice/sectors
```

2.57.1. Parameters

Type	Name	Schema
Query	**description** *optional*	string
Query	**id** *optional*	integer (int32)
Query	**institutionId** *optional*	integer (int32)
Query	**page** *optional*	integer (int32)
Query	**size** *optional*	integer (int32)

Type	Name	Schema
Query	**sort** *optional*	string

2.57.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfSector

2.57.3. Produces

- */*

2.57.4. Tags

- backoffice-sector-controller

2.57.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.58. getOne

```
GET /backoffice/sectors/{id}
```

2.58.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.58.2. Responses

HTTP Code	Description	Schema
200	OK	Sector

2.58.3. Produces

- */*

2.58.4. Tags

- backoffice-sector-controller

2.58.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.59. update

```
PUT /backoffice/sectors/{id}
```

2.59.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**body** *required*	body	Sector

2.59.2. Responses

HTTP Code	Description	Schema
200	OK	Sector

2.59.3. Consumes

- application/json

2.59.4. Produces

- */*

2.59.5. Tags

- backoffice-sector-controller

2.59.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.60. delete

```
DELETE /backoffice/sectors/{id}
```

2.60.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

2.60.2. Responses

HTTP Code	Description	Schema
200	OK	BackofficeDeleteResponseOfSector

2.60.3. Produces

- */*

2.60.4. Tags

- backoffice-sector-controller

2.60.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.61. errorHtml

```
POST /error
```

2.61.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.61.2. Consumes

- application/json

2.61.3. Produces

- text/html

2.61.4. Tags

- basic-error-controller

2.61.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.62. errorHtml

```
GET /error
```

2.62.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.62.2. Produces

- text/html

2.62.3. Tags

- basic-error-controller

2.62.4. Security

Type	Name	Scopes
apiKey	JWT	global

2.63. errorHtml

```
PUT /error
```

2.63.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.63.2. Consumes

- `application/json`

2.63.3. Produces

- `text/html`

2.63.4. Tags

- basic-error-controller

2.63.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.64. errorHtml

```
DELETE /error
```

2.64.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.64.2. Produces

- text/html

2.64.3. Tags

- basic-error-controller

2.64.4. Security

Type	Name	Scopes
apiKey	JWT	global

2.65. errorHtml

```
PATCH /error
```

2.65.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.65.2. Consumes

- application/json

2.65.3. Produces

- text/html

2.65.4. Tags

- basic-error-controller

2.65.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.66. errorHtml

```
HEAD /error
```

2.66.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.66.2. Consumes

- `application/json`

2.66.3. Produces

- `text/html`

2.66.4. Tags

- basic-error-controller

2.66.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.67. errorHtml

```
OPTIONS /error
```

2.67.1. Responses

HTTP Code	Description	Schema
200	OK	ModelAndView

2.67.2. Consumes

- `application/json`

2.67.3. Produces

- `text/html`

2.67.4. Tags

- basic-error-controller

2.67.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.68. getLanguages

```
GET /i18n/support-languages
```

2.68.1. Responses

HTTP Code	Description	Schema
200	OK	< string > array

2.68.2. Produces

- `*/*`

2.68.3. Tags

- I18n

2.68.4. Security

Type	Name	Scopes
apiKey	JWT	global

2.69. getAllInternmentPatient

```
GET /institutions/{institutionId}/internments/patients
```

2.69.1. Parameters

Type	Name	Description	Schema
Path	**institutionId** *required*	institutionId	integer (int32)

2.69.2. Responses

HTTP Code	Description	Schema
200	OK	< InternmentEpisodeDto > array

2.69.3. Produces

- */*

2.69.4. Tags

- Internment Patient

2.69.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.70. getIdentityVerificationStatus

```
POST /masterdata/patient/withoutIdentityReasons
```

2.70.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.70.2. Consumes

- `application/json`

2.70.3. Produces

- `*/*`

2.70.4. Tags

- PatientMasterData

2.70.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.71. getIdentityVerificationStatus

```
GET /masterdata/patient/withoutIdentityReasons
```

2.71.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.71.2. Produces

- `*/*`

2.71.3. Tags

- PatientMasterData

2.71.4. Security

Type	Name	Scopes
apiKey	JWT	global

2.72. getIdentityVerificationStatus

```
PUT /masterdata/patient/withoutIdentityReasons
```

2.72.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.72.2. Consumes

- `application/json`

2.72.3. Produces

- `*/*`

2.72.4. Tags

- PatientMasterData

2.72.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.73. getIdentityVerificationStatus

```
DELETE /masterdata/patient/withoutIdentityReasons
```

2.73.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.73.2. Produces

- */*

2.73.3. Tags

- PatientMasterData

2.73.4. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.74. getIdentityVerificationStatus

    PATCH /masterdata/patient/withoutIdentityReasons

2.74.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.74.2. Consumes

- application/json

2.74.3. Produces

- */*

2.74.4. Tags

- PatientMasterData

2.74.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.75. getIdentityVerificationStatus

```
HEAD /masterdata/patient/withoutIdentityReasons
```

2.75.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.75.2. Consumes

- `application/json`

2.75.3. Produces

- `*/*`

2.75.4. Tags

- PatientMasterData

2.75.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.76. getIdentityVerificationStatus

```
OPTIONS /masterdata/patient/withoutIdentityReasons
```

2.76.1. Responses

HTTP Code	Description	Schema
200	OK	< IdentityVerificationStatus > array

2.76.2. Consumes

- `application/json`

2.76.3. Produces

- `*/*`

2.76.4. Tags

- PatientMasterData

2.76.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.77. updatePassword

```
PATCH /passwords/{userId}
```

2.77.1. Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int32)
Body	**password** *required*	password	string

2.77.2. Responses

HTTP Code	Description	Schema
200	OK	boolean

2.77.3. Consumes

- `application/json`

2.77.4. Produces

- `*/*`

2.77.5. Tags

- Password

2.77.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.78. addPatient

```
POST /patient
```

2.78.1. Parameters

Type	Name	Description	Schema
Body	**patientDto** *required*	patientDto	APatientDto

2.78.2. Responses

HTTP Code	Description	Schema
200	OK	BMPatientDto

2.78.3. Consumes

- `application/json`

2.78.4. Produces

- */*

2.78.5. Tags

- Patient

2.78.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.79. getPatientMinimal

```
GET /patient/minimalsearch
```

2.79.1. Parameters

Type	Name	Description	Schema
Query	**genderId** *required*	genderId	integer (int32)
Query	**identification Number** *required*	identificationNumber	string
Query	**identification TypeId** *required*	identificationTypeId	integer (int32)

2.79.2. Responses

HTTP Code	Description	Schema
200	OK	< integer (int32) > array

2.79.3. Produces

- */*

2.79.4. Tags

- Patient

2.79.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.80. searchPatient

```
GET /patient/search
```

2.80.1. Parameters

Type	Name	Description	Schema
Query	**searchFilterStr** *required*	searchFilterStr	string

2.80.2. Responses

HTTP Code	Description	Schema
200	OK	< PatientSearchDto > array

2.80.3. Produces

- */*

2.80.4. Tags

- Patient

2.80.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.81. getBasicDataPatient

```
GET /patient/{patientId}/basicdata
```

2.81.1. Parameters

Type	Name	Description	Schema
Path	**patientId** *required*	patientId	integer (int32)

2.81.2. Responses

HTTP Code	Description	Schema
200	OK	BasicPatientDto

2.81.3. Produces

- */*

2.81.4. Tags

- Patient

2.81.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.82. addPerson

```
POST /person
```

2.82.1. Parameters

Type	Name	Description	Schema
Body	**personDto** *required*	personDto	APersonDto

2.82.2. Responses

HTTP Code	Description	Schema
200	OK	BMPersonDto

2.82.3. Consumes

- `application/json`

2.82.4. Produces

- `*/*`

2.82.5. Tags

- Person

2.82.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.83. getPersonMinimal

```
GET /person/minimalsearch
```

2.83.1. Parameters

Type	Name	Description	Schema
Query	**genderId** *required*	genderId	integer (int32)
Query	**identification Number** *required*	identificationNumber	string
Query	**identification TypeId** *required*	identificationTypeId	integer (int32)

2.83.2. Responses

HTTP Code	Description	Schema
200	OK	< integer (int32) > array

2.83.3. Produces

- */*

2.83.4. Tags

- Person

2.83.5. Security

Type	Name	Scopes
apiKey	**JWT**	global

2.84. getPersonalInformation

```
GET /person/{personId}/personalInformation
```

2.84.1. Parameters

Type	Name	Description	Schema
Path	**personId** *required*	personId	integer (int32)

2.84.2. Responses

HTTP Code	Description	Schema
200	OK	PersonalInformationDto

2.84.3. Produces

- */*

2.84.4. Tags

- Person

2.84.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.85. setRoleUser

```
POST /roles
```

2.85.1. Parameters

Type	Name	Description	Schema
Body	**userRoleDto** *required*	userRoleDto	RequestUserRoleDto

2.85.2. Responses

HTTP Code	Description	Schema
200	OK	boolean

2.85.3. Consumes

- `application/json`

2.85.4. Produces

- `*/*`

2.85.5. Tags

- Role

2.85.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.86. addUser

```
POST /users
```

2.86.1. Parameters

Type	Name	Description	Schema
Body	**userDto** *required*	userDto	AddUserDto

2.86.2. Responses

HTTP Code	Description	Schema
200	OK	UserDto

2.86.3. Consumes

- application/json

2.86.4. Produces

- */*

2.86.5. Tags

- User

2.86.6. Security

Type	Name	Scopes
apiKey	JWT	global

2.87. getUsers

```
GET /users/page
```

2.87.1. Parameters

Type	Name	Schema
Query	**page** *optional*	integer (int32)
Query	**size** *optional*	integer (int32)
Query	**sort** *optional*	string

2.87.2. Responses

HTTP Code	Description	Schema
200	OK	PageOfPageableUsers

2.87.3. Produces

- */*

2.87.4. Tags

- User

2.87.5. Security

Type	Name	Scopes
apiKey	JWT	global

2.88. changeStatusAccount

```
PATCH /users/{id}
```

2.88.1. Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Type	Name	Description	Schema
Query	**status** *required*	status	boolean

2.88.2. Responses

HTTP Code	Description	Schema
200	OK	boolean

2.88.3. Consumes

- application/json

2.88.4. Produces

- */*

2.88.5. Tags

- User

2.88.6. Security

Type	Name	Scopes
apiKey	**JWT**	global

Chapter 3. Security

3.1. JWT

Type : apiKey
Name : X-Auth-Token
In : HEADER

Chapter 4. Definitions

4.1. APatientDto

Name	Description	Schema
apartment *optional*		string
birthDate *required*	**Example** : `"yyyy-MM-dd"`	string
cityId *optional*		integer (int32)
cuil *optional*		string
email *optional*		string
ethnic *optional*		string
firstName *required*		string
floor *optional*		string
genderId *required*		integer (int32)
genderSelfDet erminationId *optional*		integer (int32)
identification Number *optional*		string
identification TypeId *required*		integer (int32)

Name	Description	Schema
lastName *required*		string
middleNames *optional*		string
mothersLastName *optional*		string
nameSelfDetermination *optional*		string
number *optional*		string
otherLastNames *optional*		string
phoneNumber *optional*		string
postcode *optional*		string
quarter *optional*		string
religion *optional*		string
street *optional*		string

4.2. APersonDto

Name	Description	Schema
apartment *optional*		string

Name	Description	Schema
birthDate *required*	**Example** : `"yyyy-MM-dd"`	string
cityId *optional*		integer (int32)
cuil *optional*		string
email *optional*		string
ethnic *optional*		string
firstName *required*		string
floor *optional*		string
genderId *required*		integer (int32)
genderSelfDeterminationId *optional*		integer (int32)
identificationNumber *optional*		string
identificationTypeId *required*		integer (int32)
lastName *required*		string
middleNames *optional*		string

Name	Description	Schema
mothersLastName *optional*		string
nameSelfDetermination *optional*		string
number *optional*		string
otherLastNames *optional*		string
phoneNumber *optional*		string
postcode *optional*		string
quarter *optional*		string
religion *optional*		string
street *optional*		string

4.3. AddUserDto

Name	Schema
email *required*	string
password *required*	string

4.4. Address

Name	Schema
apartment *optional*	string
cityId *optional*	integer (int32)
floor *optional*	string
id *optional*	integer (int32)
number *optional*	string
postcode *optional*	string
quarter *optional*	string
street *optional*	string

4.5. AddressDto

Name	Schema
apartment *optional*	string
city *optional*	CityDto
floor *optional*	string
id *optional*	integer (int32)

Name	Schema
number *optional*	string
postcode *optional*	string
province *optional*	ProvinceDto
quarter *optional*	string
street *optional*	string

4.6. AddressProjection

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.7. BMPatientDto

Name	Description	Schema
apartment *optional*		string
birthDate *required*	**Example** : `"yyyy-MM-dd"`	string
cityId *optional*		integer (int32)
cuil *optional*		string

Name	Description	Schema
email *optional*		string
ethnic *optional*		string
firstName *required*		string
floor *optional*		string
genderId *required*		integer (int32)
genderSelfDeterminationId *optional*		integer (int32)
id *optional*		integer (int32)
identificationNumber *optional*		string
identificationTypeId *required*		integer (int32)
lastName *required*		string
middleNames *optional*		string
mothersLastName *optional*		string
nameSelfDetermination *optional*		string

Name	Description	Schema
number *optional*		string
otherLastNames *optional*		string
phoneNumber *optional*		string
postcode *optional*		string
quarter *optional*		string
religion *optional*		string
street *optional*		string

4.8. BMPersonDto

Name	Description	Schema
apartment *optional*		string
birthDate *required*	**Example** : `"yyyy-MM-dd"`	string
cityId *optional*		integer (int32)
cuil *optional*		string
email *optional*		string

Name	Description	Schema
ethnic *optional*		string
firstName *required*		string
floor *optional*		string
genderId *required*		integer (int32)
genderSelfDet erminationId *optional*		integer (int32)
id *optional*		integer (int32)
identification Number *optional*		string
identification TypeId *required*		integer (int32)
lastName *required*		string
middleNames *optional*		string
mothersLastN ame *optional*		string
nameSelfDete rmination *optional*		string
number *optional*		string

Name	Description	Schema
otherLastNam es *optional*		string
phoneNumber *optional*		string
postcode *optional*		string
quarter *optional*		string
religion *optional*		string
street *optional*		string

4.9. BackofficeDeleteResponseOfAddress

Name	Schema
id *optional*	integer (int32)

4.10. BackofficeDeleteResponseOfBed

Name	Schema
id *optional*	integer (int32)

4.11. BackofficeDeleteResponseOfCity

Name	Schema
id *optional*	integer (int32)

4.12. BackofficeDeleteResponseOfClinicalSpecialty

Name	Schema
id *optional*	integer (int32)

4.13. BackofficeDeleteResponseOfClinicalSpecialtySector

Name	Schema
id *optional*	integer (int32)

4.14. BackofficeDeleteResponseOfDepartment

Name	Schema
id *optional*	integer (int32)

4.15. BackofficeDeleteResponseOfInstitution

Name	Schema
id *optional*	integer (int32)

4.16. BackofficeDeleteResponseOfProvince

Name	Schema
id *optional*	integer (int32)

4.17. BackofficeDeleteResponseOfRoom

Name	Schema
id *optional*	integer (int32)

4.18. BackofficeDeleteResponseOfSector

Name	Schema
id *optional*	integer (int32)

4.19. BasicDataPersonDto

Name	Schema
age *optional*	integer (int32)
firstName *optional*	string
gender *optional*	GenderDto
id *optional*	integer (int32)
lastName *optional*	string

4.20. BasicPatientDto

Name	Schema
id *optional*	integer (int32)
person *optional*	BasicDataPersonDto

4.21. Bed

Name	Schema
bedCategoryId *optional*	integer (int32)
id *optional*	integer (int32)
roomId *optional*	integer (int32)

4.22. BedDto

Name	Schema
bedNumber *optional*	integer (int32)
id *optional*	integer (int32)
room *optional*	RoomDto

4.23. Character

Type : object

4.24. City

Name	Schema
active *optional*	boolean
city_type *optional*	Character
departmentId *optional*	integer (int32)

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.25. CityDto

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.26. ClinicalSpecialty

Name	Schema
description *optional*	string
id *optional*	integer (int32)
name *optional*	string
sctidCode *optional*	string

4.27. ClinicalSpecialtyDto

Name	Schema
id *optional*	integer (int32)
name *optional*	string

4.28. ClinicalSpecialtySector

Name	Schema
clinicalSpecialtyId *optional*	integer (int32)
description *optional*	string
id *optional*	integer (int32)
sectorId *optional*	integer (int32)

4.29. Department

Name	Schema
description *optional*	string
id *optional*	integer (int32)
provinceId *optional*	integer (int32)

4.30. GenderDto

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.31. HealthInsuranceDto

Name	Schema
acronym *optional*	string
id *optional*	integer (int32)
rnos *optional*	string

4.32. IdentificationTypeDto

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.33. IdentityVerificationStatus

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.34. Institution

Name	Schema
addressId *optional*	integer (int32)
cuit *optional*	string
email *optional*	string

Name	Schema
id *optional*	integer (int32)
name *optional*	string
phone *optional*	string
sisaCode *optional*	string
website *optional*	string

4.35. InternmentEpisodeDto

Name	Schema
bed *optional*	BedDto
doctor *optional*	ResponsibleDoctorDto
id *optional*	integer (int32)
patient *optional*	PatientDto

4.36. IterableOfAddress

Type : object

4.37. IterableOfBed

Type : object

4.38. IterableOfCity

Type : object

4.39. IterableOfClinicalSpecialty

Type : object

4.40. IterableOfClinicalSpecialtySector

Type : object

4.41. IterableOfDepartment

Type : object

4.42. IterableOfInstitution

Type : object

4.43. IterableOfProvince

Type : object

4.44. IterableOfRoom

Type : object

4.45. IterableOfSector

Type : object

4.46. JWTokenDto

Name	Schema
refreshToken *optional*	string
token *optional*	string

4.47. Link

Name	Schema
href *optional*	string
templated *optional*	boolean

4.48. LoginDto

Name	Schema
password *required*	string
username *required*	string

4.49. MapOfstringAndLink

Type : < string, Link > map

4.50. ModelAndView

Name	Schema
empty *optional*	boolean
model *optional*	object
modelMap *optional*	< string, object > map
reference *optional*	boolean

Name	Schema
status *optional*	enum (100 CONTINUE, 101 SWITCHING_PROTOCOLS, 102 PROCESSING, 103 CHECKPOINT, 200 OK, 201 CREATED, 202 ACCEPTED, 203 NON_AUTHORITATIVE_INFORMATION, 204 NO_CONTENT, 205 RESET_CONTENT, 206 PARTIAL_CONTENT, 207 MULTI_STATUS, 208 ALREADY_REPORTED, 226 IM_USED, 300 MULTIPLE_CHOICES, 301 MOVED_PERMANENTLY, 302 FOUND, 302 MOVED_TEMPORARILY, 303 SEE_OTHER, 304 NOT_MODIFIED, 305 USE_PROXY, 307 TEMPORARY_REDIRECT, 308 PERMANENT_REDIRECT, 400 BAD_REQUEST, 401 UNAUTHORIZED, 402 PAYMENT_REQUIRED, 403 FORBIDDEN, 404 NOT_FOUND, 405 METHOD_NOT_ALLOWED, 406 NOT_ACCEPTABLE, 407 PROXY_AUTHENTICATION_REQUIRED, 408 REQUEST_TIMEOUT, 409 CONFLICT, 410 GONE, 411 LENGTH_REQUIRED, 412 PRECONDITION_FAILED, 413 PAYLOAD_TOO_LARGE, 413 REQUEST_ENTITY_TOO_LARGE, 414 URI_TOO_LONG, 414 REQUEST_URI_TOO_LONG, 415 UNSUPPORTED_MEDIA_TYPE, 416 REQUESTED_RANGE_NOT_SATISFIABLE, 417 EXPECTATION_FAILED, 418 I_AM_A_TEAPOT, 419 INSUFFICIENT_SPACE_ON_RESOURCE, 420 METHOD_FAILURE, 421 DESTINATION_LOCKED, 422 UNPROCESSABLE_ENTITY, 423 LOCKED, 424 FAILED_DEPENDENCY, 425 TOO_EARLY, 426 UPGRADE_REQUIRED, 428 PRECONDITION_REQUIRED, 429 TOO_MANY_REQUESTS, 431 REQUEST_HEADER_FIELDS_TOO_LARGE, 451 UNAVAILABLE_FOR_LEGAL_REASONS, 500 INTERNAL_SERVER_ERROR, 501 NOT_IMPLEMENTED, 502 BAD_GATEWAY, 503 SERVICE_UNAVAILABLE, 504 GATEWAY_TIMEOUT, 505 HTTP_VERSION_NOT_SUPPORTED, 506 VARIANT_ALSO_NEGOTIATES, 507 INSUFFICIENT_STORAGE, 508 LOOP_DETECTED, 509 BANDWIDTH_LIMIT_EXCEEDED, 510 NOT_EXTENDED, 511 NETWORK_AUTHENTICATION_REQUIRED)
view *optional*	View

Name	Schema
viewName *optional*	string

4.51. PageOfAddress

Name	Schema
content *optional*	< Address > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.52. PageOfBed

Name	Schema
content *optional*	< Bed > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.53. PageOfCity

Name	Schema
content *optional*	< City > array

Name	Schema
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.54. PageOfClinicalSpecialty

Name	Schema
content *optional*	< ClinicalSpecialty > array
empty *optional*	boolean
first *optional*	boolean

Name	Schema
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.55. PageOfClinicalSpecialtySector

Name	Schema
content *optional*	< ClinicalSpecialtySector > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)

Name	Schema
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.56. PageOfDepartment

Name	Schema
content *optional*	< Department > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable

Name	Schema
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.57. PageOfInstitution

Name	Schema
content *optional*	< Institution > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort

Name	Schema
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.58. PageOfPageableUsers

Name	Schema
content *optional*	< PageableUsers > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.59. PageOfProvince

Name	Schema
content *optional*	< Province > array
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.60. PageOfRoom

Name	Schema
content *optional*	< Room > array

Name	Schema
empty *optional*	boolean
first *optional*	boolean
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.61. PageOfSector

Name	Schema
content *optional*	< Sector > array
empty *optional*	boolean
first *optional*	boolean

Name	Schema
last *optional*	boolean
number *optional*	integer (int32)
numberOfElements *optional*	integer (int32)
pageable *optional*	Pageable
size *optional*	integer (int32)
sort *optional*	Sort
totalElements *optional*	integer (int64)
totalPages *optional*	integer (int32)

4.62. Pageable

Name	Schema
page *optional*	integer (int32)
size *optional*	integer (int32)
sort *optional*	string

4.63. PageableUsers

Name	Schema
enable *optional*	boolean
id *optional*	integer (int32)
lastLogin *optional*	string (date-time)
role *optional*	string
username *optional*	string

4.64. PatientDto

Name	Schema
firstName *optional*	string
id *optional*	integer (int32)
lastName *optional*	string

4.65. PatientSearchDto

Name	Schema
person *optional*	BMPersonDto
ranking *optional*	number (float)

4.66. PersonalInformationDto

Name	Schema
address *optional*	AddressDto
birthDate *optional*	string (date)
cuil *optional*	string
email *optional*	string
healthInsurance *optional*	HealthInsuranceDto
id *optional*	integer (int32)
identificationNumber *optional*	string
identificationType *optional*	IdentificationTypeDto
phoneNumber *optional*	string

4.67. Province

Name	Schema
countryId *optional*	integer (int32)
description *optional*	string
id *optional*	integer (int32)

4.68. ProvinceDto

Name	Schema
description *optional*	string
id *optional*	integer (int32)

4.69. RequestUserRoleDto

Name	Schema
roleId *optional*	integer (int32)
userId *required*	integer (int32)

4.70. ResponsibleDoctorDto

Name	Schema
firstName *optional*	string
id *optional*	integer (int32)
lastName *optional*	string

4.71. Room

Name	Schema
description *optional*	string
dischargeDate *optional*	string (date)

Name	Schema
id *optional*	integer (int32)
sectorId *optional*	integer (int32)
type *optional*	string

4.72. RoomDto

Name	Schema
id *optional*	integer (int32)
roomNumber *optional*	integer (int32)
sector *optional*	SectorDto
specialty *optional*	ClinicalSpecialtyDto

4.73. Sector

Name	Schema
description *optional*	string
id *optional*	integer (int32)
institutionId *optional*	integer (int32)

4.74. SectorDto

Name	Schema
id *optional*	integer (int32)
name *optional*	string

4.75. Sort

Name	Schema
empty *optional*	boolean
sorted *optional*	boolean
unsorted *optional*	boolean

4.76. UserDto

Name	Schema
email *required*	string
id *optional*	integer (int32)

4.77. View

Name	Schema
contentType *optional*	string