

COORDINACIÓN GENERAL DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

FICHA DE TÉCNICA

DATOS GENERALES

| | | | |
|--------------------------------|---|------------------------|--|
| Nombre de Documento: | Ficha técnica de APIs | | |
| Detalle de Documento: | Documento con la caracterización y descripción de la creación de APIs | | |
| Sistema: | MAG - Organización | | |
| Fecha de Elaboración: | 14 de julio de 2022 | | |
| Fecha de Actualización: | 14 de julio de 2022 | | |
| Fecha de Aprobación: | 14 de julio de 2022 | | |
| Elaborado por: | Mateo Alarcón Fausto Borja Oscar Narváez | Modificado por: | |
| Revisado por: | | Aprobado por: | |

DESCRIPCIÓN DEL SERVICIO

| | | | |
|---|--|---------------------------------------|--|
| Fecha de creación: | | Fecha de última actualización: | |
| Nombre del Servicio: | | | |
| Documentación | | | |
| Descripción del Servicio: | | | |
| <p>Creación de un APIs para el consumo del Microservicios:</p> <ul style="list-style-type: none"> • Actividades asociativas • Centro de acopio • Comercializacion socio • Comercializacion asociativa | | | |

**Métodos:**

- create
- delete
- findById
- findAll
- findByOrgId
-

Tipo:

- POST
- GET
- PUT
- DELETE

```
1 package ec.gob.mag.api.controller;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36 @RestController
37 @RequestMapping("/api")
38 @ApiResponses(value = { @ApiResponse(code = 200, message = "SUCESS"),
39     @ApiResponse(code = 404, message = "RESOURCE NOT FOUND"), @ApiResponse(code = 400,
40     message = "BAD REQUEST"),
41     @ApiResponse(code = 201, message = "CREATED"), @ApiResponse(code = 401, message =
42     "UNAUTHORIZED"),
43     @ApiResponse(code = 415, message = "UNSUPPORTED TYPE - Representation not supported
44     for the resource"),
45     @ApiResponse(code = 500, message = "SERVER ERROR") })
46
47 public class ActividadesAsociativasController implements ErrorController {
48
49     private static final String PATH = "/error";
50     public static final Logger LOGGER =
51     LoggerFactory.getLogger(ActividadesAsociativasController.class);
52
53     /*****
54     * SECCION - INYECCION DE DEPENDENCIAS
55     *****/
56     @Autowired
57     @Qualifier("consumer")
58     private Consumer consumer;
59
60     @Autowired
61     @Qualifier("convertEntityUtil")
62     private ConvertEntityUtil convertEntityUtil;
63
64     @Autowired
65     @Qualifier("util")
66     private Util util;
67
68     /*****
69     * SECCION - MICROSERVICIOS
70     *****/
71     @Value("${url.servidor_micro}")
72     private String urlServidor;
73
74     @Value("${url.servidor_mag}")
75     private String urlServidorMag;
76
77     @Value("${url.actividades_asociacion}")
78     private String urlMicroActividadesAsociativas;
79
80     /*****
81     * SECCION - END-POINTS
82     *****/
83
84     /*****
85     * Obtiene todos las entidades Actividades Asociativas dado un token
86     *
87     * @param token
88     * @return
89     * @throws NoSuchFieldException
90     * @throws SecurityException
91     * @throws IllegalArgumentException
```

```

90     * @throws IllegalAccessException
91     * @throws IOException
92     */
93
94     @GetMapping(value = "/actividadesAsociativas/findAll")
95     @ApiOperation(value = "Obtiene todas las actividades asociativas", response =
Object.class)
96     @ResponseStatus(HttpStatus.OK)
97     public Object obtenerActividadesAsociativas(@RequestHeader(name = "Authorization")
String token) throws NoSuchFieldException, SecurityException, IllegalArgumentException,
IllegalAccessException, IOException{
98
99         String pathMicro = urlServidorMag + urlMicroActividadesAsociativas +
"/actividades_asociacion/findAll" ;
100         //Object response = consumer.doGet(pathMicro, token);
101         Object response = consumer.doGet(pathMicro, token);
102         LOGGER.info("actividadesAsociativas/findAll" + " usuario: " + util.filterUsuId
(token));
103         return ResponseEntity.ok(response);
104
105     }
106     /*****
107     * Obtiene todas las entidades Actividades Asociativas dado un token y el id o llave
primaria
108     *
109     * @param acaId
110     * @param token
111     * @return
112     * @throws NoSuchFieldException
113     * @throws SecurityException
114     * @throws IllegalArgumentException
115     * @throws IllegalAccessException
116     * @throws IOException
117     */
118
119     @GetMapping(value = "/actividadesAsociativas/findById/{catId}")
120     @ApiOperation(value = "Obtiene las actividades asociativas por id", response =
Object.class)
121     @ResponseStatus(HttpStatus.OK)
122     public Object obtenerActividadesAsociativasById(@PathVariable Long acaId,
@RequestHeader(name = "Authorization") String token) throws NoSuchFieldException,
SecurityException, IllegalArgumentException, IllegalAccessException, IOException{
123
124         String pathMicro = urlServidorMag + urlMicroActividadesAsociativas +
"/actividades_asociacion/findById/" + acaId;
125         //Object response = consumer.doGet(pathMicro, token);
126         Object response = consumer.doGet(pathMicro, token);
127         LOGGER.info("actividades_asociacion/findById" + " usuario: " + util.filterUsuId
(token));
128         return ResponseEntity.ok(response);
129
130     }
131
132     /*****
133     * Crear una identidad dado el nombre de una Actividad Asociativa y un token de
seguridad
134     * @param actividadAsotivaNombre
135     * @param token
136     * @return
137     * @throws NoSuchFieldException
138     * @throws SecurityException
139     * @throws IllegalArgumentException

```

```

140     * @throws IllegalAccessException
141     * @throws IOException
142     */
143     @PostMapping(value = "/actividadesAsociativas/create")
144     @ApiOperation(value = "Crea una nueva actividad asociativa", response = Object.class)
145     @ResponseStatus(HttpStatus.CREATED)
146     public ResponseEntity<?> saveActividadesAsociativasPost(@Validated @RequestBody String
    actividadAsotivaNombre,
147         @RequestHeader(name = "Authorization") String token) throws
    NoSuchFieldException, SecurityException,
148         IllegalArgumentException, IllegalAccessException, IOException {
149         String pathMicro = urlServidorMag + urlMicroActividadesAsociativas +
    "actividades_asociacion/create/";
150         ActividadesAsociativasDTO actividadesAsociativasDTO =
    convertEntityUtil.ConvertSingleEntityPOST(pathMicro, actividadAsotivaNombre, token,
    ActividadesAsociativasDTO.class);
151         LOGGER.info("actividades_asociacion/create" + " usuario: " + util.filterUsuId
    (token));
152         return ResponseEntity.ok(actividadesAsociativasDTO);
153     }
154
155     /*****
156     * Actualia una entidad Actividad Asociativa dado su id, nombre y además un token de
    seguridad
157     * @param acaId
158     * @param actividadAsociativaNombre
159     * @param auth
160     * @return
161     * @throws NoSuchFieldException
162     * @throws SecurityException
163     * @throws IllegalArgumentException
164     * @throws IllegalAccessException
165     * @throws IOException
166     */
167
168     @PutMapping(value = "/actividadesAsociativas/update/{acaId}")
169     @ApiOperation(value = "Actualiza un registro de actividades asociativas", response =
    Object.class)
170     @ResponseStatus(HttpStatus.CREATED)
171
172     public ResponseEntity<?> updateActividadesAsociativas(@Validated @PathVariable Long
    acaId, @RequestBody String actividadAsociativaNombre,
173         @RequestHeader(name = "Authorization") String auth) throws
    NoSuchFieldException, SecurityException,
174         IllegalArgumentException, IllegalAccessException, IOException {
175         String pathMicro = urlServidor + urlMicroActividadesAsociativas +
    "actividades_asociacion/update/" + acaId;
176         ActividadesAsociativasDTO actividadesAsociativasDTO =
    convertEntityUtil.ConvertSingleEntityPOST(pathMicro, actividadAsociativaNombre, auth,
    ActividadesAsociativasDTO.class);
177         LOGGER.info("actividades_asociacion/update/: entidad:" + actividadAsociativaNombre
    + " usuario: " + acaId);
178         return ResponseEntity.ok(actividadesAsociativasDTO);
179     }
180
181     /*****
182     * Actualiza la entidad dado la llave foránea orgId, nombre y un token de seguridad
183     *
184     * @param orgId
185     * @param actividadAsociativaNombre
186     * @param auth
187     * @return

```

```

188     * @throws NoSuchFieldException
189     * @throws SecurityException
190     * @throws IllegalArgumentException
191     * @throws IllegalAccessException
192     * @throws IOException
193     */
194
195     @PutMapping(value = "/actividadesAsociativas/updateByOrgId/{acaId}")
196     @ApiOperation(value = "Actualiza un registro de actividades asociativas", response =
Object.class)
197     @ResponseStatus(HttpStatus.CREATED)
198
199     public ResponseEntity<?> updateActividadesAsociativasByOrgId(@Validated @PathVariable
Long orgId, @RequestBody String actividadAsociativaNombre,
200         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
201         IllegalArgumentException, IllegalAccessException, IOException {
202         String pathMicro = urlServidor + urlMicroActividadesAsociativas +
"actividades_asociacion/updateByOrgId/" + orgId;
203         ActividadesAsociativasDTO actividadesAsociativasDTO =
convertEntityUtil.ConvertSingleEntityPOST(pathMicro, actividadAsociativaNombre, auth,
ActividadesAsociativasDTO.class);
204         LOGGER.info("actividades_asociacion/update/: entidad:" + actividadAsociativaNombre
+ " usuario: " + orgId);
205         return ResponseEntity.ok(actividadesAsociativasDTO);
206     }
207
208
209     /*****
210     * Hace el eliminado lógico o por atributo de la entidad dado un ID
211     *
212     * @param acaId
213     * @param auth
214     * @return
215     * @throws NoSuchFieldException
216     * @throws SecurityException
217     * @throws IllegalArgumentException
218     * @throws IllegalAccessException
219     * @throws IOException
220     */
221     @DeleteMapping(value = "/actividadesAsociativas/delete/{acaId}")
222     @ApiOperation(value = "Elimina un registro en la tabla Actividades Asociativas",
response = Object.class)
223     @ResponseStatus(HttpStatus.OK)
224
225     public ResponseEntity<?> deleteActividadesAsociativas(@Validated @PathVariable Long
acaId,
226         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
227         IllegalArgumentException, IllegalAccessException, IOException {
228         String pathMicro = urlServidor + urlMicroActividadesAsociativas +
"actividades_asociacion/delete/" + acaId;
229         Object pruebaDTO = consumer.doGet(pathMicro, auth);
230         LOGGER.info("actividades_asociacion/delete: id:" + acaId);
231         return ResponseEntity.ok(pruebaDTO);
232     }
233
234     @Override
235     public String getErrorPath() {
236         return PATH;
237     }
238 }

```



```

1 package ec.gob.mag.api.controller;
2
3 import java.io.IOException;
36
37 @RestController
38 @RequestMapping("/api")
39 @ApiResponses(value = { @ApiResponse(code = 200, message = "SUCESS"),
40     @ApiResponse(code = 404, message = "RESOURCE NOT FOUND"), @ApiResponse(code = 400,
    message = "BAD REQUEST"),
41     @ApiResponse(code = 201, message = "CREATED"), @ApiResponse(code = 401, message =
    "UNAUTHORIZED"),
42     @ApiResponse(code = 415, message = "UNSUPPORTED TYPE - Representation not supported
    for the resource"),
43     @ApiResponse(code = 500, message = "SERVER ERROR") })
44
45 public class CentroAcopioController implements ErrorController {
46
47     private static final String PATH = "/error";
48     public static final Logger LOGGER =
    LoggerFactory.getLogger(CentroAcopioController.class);
49
50     /*****
51     * SECCION - INYECCION DE DEPENDENCIAS
52     *****/
53     @Autowired
54     @Qualifier("consumer")
55     private Consumer consumer;
56
57     @Autowired
58     @Qualifier("convertEntityUtil")
59     private ConvertEntityUtil convertEntityUtil;
60
61     @Autowired
62     @Qualifier("util")
63     private Util util;
64
65     /*****
66     * SECCION - MICROSERVICIOS
67     *****/
68     @Value("${url.servidor_micro}")
69     private String urlServidor;
70
71     @Value("${url.servidor_mag}")
72     private String urlServidorMag;
73
74     @Value("${url.centro_acopio}")
75     private String urlMicroCentroAcopio;
76
77
78     /*****
79     * SECCION - END-POINTS
80     *****/
81
82     /*****
83     * Obtiene todos las entidades de Centro Acopio dado un token
84     *
85     * @param token
86     * @return
87     * @throws NoSuchFieldException
88     * @throws SecurityException
89     * @throws IllegalArgumentException
90     * @throws IllegalAccessException

```



```

91     * @throws IOException
92     */
93
94     @GetMapping(value = "/CentroAcopio/findAll")
95     @ApiOperation(value = "Obtiene todos los datos de Centros de Acopios", response =
Object.class)
96     @ResponseStatus(HttpStatus.OK)
97     public Object obtenerCentroAcopios(@RequestHeader(name = "Authorization") String token)
throws NoSuchFieldException, SecurityException, IllegalArgumentException,
IllegalAccessException, IOException{
98
99         String pathMicro = urlServidorMag + urlMicroCentroAcopio + "/centro_acopio/findAll"
;
100         //Object response = consumer.doGet(pathMicro, token);
101         Object response = consumer.doGet(pathMicro, token);
102         LOGGER.info("centro_acopio/findAll" + " usuario: " + util.filterUsuId(token));
103         return ResponseEntity.ok(response);
104     }
105 }
106 /*****
107  * Obtiene todas las entidades dado su ID y un token de seguridad
108  *
109  * @param cacId
110  * @param token
111  * @return
112  * @throws NoSuchFieldException
113  * @throws SecurityException
114  * @throws IllegalArgumentException
115  * @throws IllegalAccessException
116  * @throws IOException
117  */
118
119     @GetMapping(value = "/CentroAcopio/findById/{id}")
120     @ApiOperation(value = "Obtiene los Centros de acopio por id", response = Object.class)
121     @ResponseStatus(HttpStatus.OK)
122     public Object obtenerCentroAcopioById(@PathVariable Long cacId, @RequestHeader(name =
"Authorization") String token) throws NoSuchFieldException, SecurityException,
IllegalArgumentException, IllegalAccessException, IOException{
123
124         String pathMicro = urlServidorMag + urlMicroCentroAcopio +
"/centro_acopio/findById/" + cacId;
125         //Object response = consumer.doGet(pathMicro, token);
126         Object response = consumer.doGet(pathMicro, token);
127         LOGGER.info("centro_acopio/findById" + " usuario: " + util.filterUsuId(token));
128         return ResponseEntity.ok(response);
129     }
130 }
131
132 /*****
133  * Crea unna entidad dado un nombre de tipo String y un token de seguridad
134  *
135  * @param centroAcopioaNombre
136  * @param token
137  * @return
138  * @throws NoSuchFieldException
139  * @throws SecurityException
140  * @throws IllegalArgumentException
141  * @throws IllegalAccessException
142  * @throws IOException
143  */
144     @PostMapping(value = "/CentroAcopio/create")
145     @ApiOperation(value = "Crea un nuevo Centro de Acopio", response = Object.class)

```

```

146     @ResponseStatus(HttpStatus.CREATED)
147     public ResponseEntity<?> saveCentroAcopio(@Validated @RequestBody String
        centroAcopioaNombre,
148         @RequestHeader(name = "Authorization") String token) throws
        NoSuchFieldException, SecurityException,
149         IllegalArgumentException, IllegalAccessException, IOException {
150         String pathMicro = urlServidorMag + urlMicroCentroAcopio + "centro_acopio/create/";
151         CentroAcopioDTO centroAcopioDTO = convertEntityUtil.ConvertSingleEntityPOST
        (pathMicro, centroAcopioaNombre, token, CentroAcopioDTO.class);
152         LOGGER.info("centro_acopio/create" + " usuario: " + util.filterUsuId(token));
153         return ResponseEntity.ok(centroAcopioDTO);
154     }
155     /*****
156     * Actualiza una entidad dado su llave primaria, nombre y el token de seguridad
157     * @param acaId
158     * @param centroAcopioaNombre
159     * @param auth
160     * @return
161     * @throws NoSuchFieldException
162     * @throws SecurityException
163     * @throws IllegalArgumentException
164     * @throws IllegalAccessException
165     * @throws IOException
166     */
167     @PutMapping(value = "/CentroAcopio/update/{Id}")
168     @ApiOperation(value = "Actualiza un registro de Centro de acopios", response =
        Object.class)
169     @ResponseStatus(HttpStatus.CREATED)
170
171     public ResponseEntity<?> updateCentroAcopio(@Validated @PathVariable Long acaId,
        @RequestBody String centroAcopioaNombre,
172         @RequestHeader(name = "Authorization") String auth) throws
        NoSuchFieldException, SecurityException,
173         IllegalArgumentException, IllegalAccessException, IOException {
174         String pathMicro = urlServidor + urlMicroCentroAcopio + "centro_acopio/update/" +
        acaId;
175         CentroAcopioDTO centroAcopioDTO = convertEntityUtil.ConvertSingleEntityPOST
        (pathMicro, centroAcopioaNombre, auth, CentroAcopioDTO.class);
176         LOGGER.info("centro_acopio/update/: entidad:" + centroAcopioaNombre + " usuario: "
        + acaId);
177         return ResponseEntity.ok(centroAcopioDTO);
178     }
179
180     /*****
181     * Actualiza la entidad dado la llave foránea OrgId, nombre de la entidad y el token de
        autorización
182     * @param orgId
183     * @param centroAcopioaNombre
184     * @param auth
185     * @return
186     * @throws NoSuchFieldException
187     * @throws SecurityException
188     * @throws IllegalArgumentException
189     * @throws IllegalAccessException
190     * @throws IOException
191     */
192
193     @PutMapping(value = "/CentroAcopio/updateByOrgId/{Id}")
194     @ApiOperation(value = "Actualiza un registro de Centros de acopio", response =
        Object.class)
195     @ResponseStatus(HttpStatus.CREATED)
196

```

```

197     public ResponseEntity<?> updateCentroAcopioByOrgId(@Validated @PathVariable Long orgId,
198     @RequestBody String centroAcopioaNombre,
199     @RequestHeader(name = "Authorization") String auth) throws
    NoSuchFieldException, SecurityException,
199     IllegalArgumentException, IllegalAccessException, IOException {
200     String pathMicro = urlServidor + urlMicroCentroAcopio +
    "centro_acopio/updateByOrgId/" + orgId;
201     CentroAcopioDTO centroAcopioDTO = convertEntityUtil.ConvertSingleEntityPOST
    (pathMicro, centroAcopioaNombre, auth, CentroAcopioDTO.class);
202     LOGGER.info("centro_acopio/updateByOrgId: entidad:" + centroAcopioaNombre + "
    usuario: " + orgId);
203     return ResponseEntity.ok(centroAcopioDTO);
204 }
205 /*****
206  * Se hace un eliminado lógico de la entidad dado un ID, además del token de seguridad
207  * @param cacId
208  * @param auth
209  * @return
210  * @throws NoSuchFieldException
211  * @throws SecurityException
212  * @throws IllegalArgumentException
213  * @throws IllegalAccessException
214  * @throws IOException
215  */
216
217 @DeleteMapping(value = "/CentroAcopio/delete/{Id}")
218 @ApiOperation(value = "Elimina un registro en la tabla Centro de Acopios", response =
    Object.class)
219 @ResponseStatus(HttpStatus.OK)
220
221     public ResponseEntity<?> deleteCentroAcopio(@Validated @PathVariable Long cacId,
222     @RequestHeader(name = "Authorization") String auth) throws
    NoSuchFieldException, SecurityException,
223     IllegalArgumentException, IllegalAccessException, IOException {
224     String pathMicro = urlServidor + urlMicroCentroAcopio + "centro_acopio/delete/" +
    cacId;
225     Object centroAcopioDTO = consumer.doGet(pathMicro, auth);
226     LOGGER.info("actividades_asociacion/delete: id:" + cacId);
227     return ResponseEntity.ok(centroAcopioDTO);
228 }
229 /*****
230  *
231  */
232
233 @Override
234     public String getErrorPath() {
235         return PATH;
236     }
237 }
238

```

```

1 package ec.gob.mag.api.controller;
2
3 import java.io.IOException;
37
38 @RestController
39 @RequestMapping("/api")
40 @ApiResponses(value = { @ApiResponse(code = 200, message = "SUCESS"),
41     @ApiResponse(code = 404, message = "RESOURCE NOT FOUND"), @ApiResponse(code = 400,
42     message = "BAD REQUEST"),
43     @ApiResponse(code = 201, message = "CREATED"), @ApiResponse(code = 401, message =
44     "UNAUTHORIZED"),
45     @ApiResponse(code = 415, message = "UNSUPPORTED TYPE - Representation not supported
46     for the resource"),
47     @ApiResponse(code = 500, message = "SERVER ERROR") })
48
49 public class ComercializacionAsociativaController implements ErrorController {
50
51     private static final String PATH = "/error";
52     public static final Logger LOGGER =
53     LoggerFactory.getLogger(ComercializacionAsociativaController.class);
54
55     /*****
56     * SECCION - INYECCION DE DEPENDENCIAS
57     *****/
58     @Autowired
59     @Qualifier("consumer")
60     private Consumer consumer;
61
62     @Autowired
63     @Qualifier("convertEntityUtil")
64     private ConvertEntityUtil convertEntityUtil;
65
66     @Autowired
67     @Qualifier("util")
68     private Util util;
69
70     /*****
71     * SECCION - MICROSERVICIOS
72     *****/
73     @Value("${url.servidor_micro}")
74     private String urlServidor;
75
76     @Value("${url.servidor_mag}")
77     private String urlServidorMag;
78
79     @Value("${url.comercializacion_asociativa}")
80     private String urlComercializacionAsociativa;
81
82     /*****
83     * SECCION - END-POINTS
84     *****/
85
86     * Obtiene todas las entidades dado sólo el token de seguridad
87     *
88     * @param token
89     * @return
90     * @throws NoSuchFieldException
91     * @throws SecurityException
92     * @throws IllegalArgumentException
93     * @throws IllegalAccessException
94     * @throws IOException

```

```

92     */
93
94     @GetMapping(value = "/ComercializacionAsociativa/findAll")
95     @ApiOperation(value = "Obtiene todos los datos de Comercializacion Asociativa",
96         response = Object.class)
97     @ResponseStatus(HttpStatus.OK)
98     public Object obtenerComercializacionAsociativa(@RequestHeader(name = "Authorization")
99         String token) throws NoSuchFieldException, SecurityException, IllegalArgumentException,
100         IllegalAccessException, IOException{
101
102         String pathMicro = urlServidorMag + urlComercializacionAsociativa +
103         "/comercializacion_asociativa/findAll" ;
104         //Object response = consumer.doGet(pathMicro, token);
105         Object response = consumer.doGet(pathMicro, token);
106         LOGGER.info("/comercializacion_asociativa/findAll" + " usuario: " +
107         util.filterUsuId(token));
108         return ResponseEntity.ok(response);
109     }
110
111     /*****
112     * Obtiene una entidad dado un ID primario y un token de seguridad
113     *
114     * @param coaId
115     * @param token
116     * @return
117     * @throws NoSuchFieldException
118     * @throws SecurityException
119     * @throws IllegalArgumentException
120     * @throws IllegalAccessException
121     * @throws IOException
122     */
123     @GetMapping(value = "/ComercializacionAsociativa/findById/{id}")
124     @ApiOperation(value = "Obtiene las Comercializaciones Asociativas por id", response =
125         Object.class)
126     @ResponseStatus(HttpStatus.OK)
127     public Object obtenerComercializacionAsociativaById(@PathVariable Long coaId,
128         @RequestHeader(name = "Authorization") String token) throws NoSuchFieldException,
129         SecurityException, IllegalArgumentException, IllegalAccessException, IOException{
130
131         String pathMicro = urlServidorMag + urlComercializacionAsociativa +
132         "/comercializacion_asociativa/findById/" + coaId;
133         //Object response = consumer.doGet(pathMicro, token);
134         Object response = consumer.doGet(pathMicro, token);
135         LOGGER.info("/comercializacion_asociativa/findById" + " usuario: " +
136         util.filterUsuId(token));
137         return ResponseEntity.ok(response);
138     }
139
140     /*****
141     * Crea una entidad dado un nombre de tipo String y token de seguridad
142     *
143     * @param comercializacionAsociativaNombre
144     * @param token
145     * @return
146     * @throws NoSuchFieldException
147     * @throws SecurityException
148     * @throws IllegalArgumentException
149     * @throws IllegalAccessException
150     * @throws IOException
151     */
152     @PostMapping(value = "/ComercializacionAsociativa/create")
153     @ApiOperation(value = "Crea una nuevaComercializacion Asociativa", response =

```

```

    Object.class)
144     @ResponseStatus(HttpStatus.CREATED)
145     public ResponseEntity<?> saveComercializacionAsociativa(@Validated @RequestBody String
        comercializacionAsociativaNombre,
146         @RequestHeader(name = "Authorization") String token) throws
        NoSuchFieldException, SecurityException,
147         IllegalArgumentException, IllegalAccessException, IOException {
148         String pathMicro = urlServidorMag + urlComercializacionAsociativa +
            "comercializacion_asociativa/create/";
149         ComercializacionAsociativaDTO comercializacionAsociativaDTO =
            convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionAsociativaNombre,
            token, ComercializacionAsociativaDTO.class);
150         LOGGER.info("centro_acopio/create" + " usuario: " + util.filterUsuId(token));
151         return ResponseEntity.ok(comercializacionAsociativaDTO);
152     }
153     /*****
154     * Actualiza una entidad Comercialización Asociativa dado un nombre e ID, además de su
        token
155     *
156     * @param coaId
157     * @param comercializacionAsociativaNombre
158     * @param auth
159     * @return
160     * @throws NoSuchFieldException
161     * @throws SecurityException
162     * @throws IllegalArgumentException
163     * @throws IllegalAccessException
164     * @throws IOException
165     */
166     @PutMapping(value = "/ComercializacionAsociativa/update/{Id}")
167     @ApiOperation(value = "Actualiza un registro de Comercializacion Asociativas", response
        = Object.class)
168     @ResponseStatus(HttpStatus.CREATED)
169
170     public ResponseEntity<?> updateComercializacionAsociativa(@Validated @PathVariable Long
        coaId, @RequestBody String comercializacionAsociativaNombre,
171         @RequestHeader(name = "Authorization") String auth) throws
        NoSuchFieldException, SecurityException,
172         IllegalArgumentException, IllegalAccessException, IOException {
173         String pathMicro = urlServidor + urlComercializacionAsociativa +
            "comercializacion_asociativa/update/" + coaId;
174         ComercializacionAsociativaDTO comercializacionAsociativaDTO =
            convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionAsociativaNombre,
            auth, ComercializacionAsociativaDTO.class);
175         LOGGER.info("comercializacion_asociativa/update/: entidad:" +
            comercializacionAsociativaNombre + " usuario: " + coaId);
176         return ResponseEntity.ok(comercializacionAsociativaDTO);
177     }
178
179
180     /*****
181     * Actualiza una entidad dado la llave foránea orgId y nombre. Además de un token de
        seguridad
182     * @param orgId
183     * @param comercializacionAsociativaNombre
184     * @param auth
185     * @return
186     * @throws NoSuchFieldException
187     * @throws SecurityException
188     * @throws IllegalArgumentException
189     * @throws IllegalAccessException
190     * @throws IOException

```



```

191     */
192
193     @PutMapping(value = "/ComercializacionAsociativa/updateByOrgId/{Id}")
194     @ApiOperation(value = "Actualiza un registro de Comercializacion Asociativas", response
= Object.class)
195     @ResponseStatus(HttpStatus.CREATED)
196
197     public ResponseEntity<?> updateComercializacionAsociativaByOrgId(@Validated
@PathVariable Long orgId, @RequestBody String comercializacionAsociativaNombre,
198         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
199         IllegalArgumentException, IllegalAccessException, IOException {
200         String pathMicro = urlServidor + urlComercializacionAsociativa +
"comercializacion_asociativa/updateByOrgId/" + orgId;
201         ComercializacionAsociativaDTO comercializacionAsociativaDTO =
convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionAsociativaNombre,
auth, ComercializacionAsociativaDTO.class);
202         LOGGER.info("centro_acopio/updateByOrgId/: entidad:" +
comercializacionAsociativaNombre + " usuario: " + orgId);
203         return ResponseEntity.ok(comercializacionAsociativaDTO);
204     }
205     /*****
206     * Elimiando una entidad dado un ID de atributo primario, y un token de seguridad
207     *
208     * @param coaId
209     * @param auth
210     * @return
211     * @throws NoSuchFieldException
212     * @throws SecurityException
213     * @throws IllegalArgumentException
214     * @throws IllegalAccessException
215     * @throws IOException
216     */
217     @DeleteMapping(value = "/ComercializacionAsociativa/delete/{Id}")
218     @ApiOperation(value = "Elimina un registro en la tabla Comercializacion Asociativa",
response = Object.class)
219     @ResponseStatus(HttpStatus.OK)
220
221     public ResponseEntity<?> deleteComercializacionAsociativa(@Validated @PathVariable Long
coaId,
222         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
223         IllegalArgumentException, IllegalAccessException, IOException {
224         String pathMicro = urlServidor + urlComercializacionAsociativa +
"comercializacion_asociativa/delete/" + coaId;
225         Object comercializacionAsociativaDTO = consumer.doGet(pathMicro, auth);
226         LOGGER.info("comercializacion_asociativa/delete: id:" + coaId);
227         return ResponseEntity.ok(comercializacionAsociativaDTO);
228     }
229
230     @Override
231     public String getErrorPath() {
232         return PATH;
233     }
234 }
235

```

```

1 package ec.gob.mag.api.controller;
2
3 import java.io.IOException;
38
39 @RestController
40 @RequestMapping("/api")
41 @ApiResponses(value = { @ApiResponse(code = 200, message = "SUCESS"),
42     @ApiResponse(code = 404, message = "RESOURCE NOT FOUND"), @ApiResponse(code = 400,
43     message = "BAD REQUEST"),
44     @ApiResponse(code = 201, message = "CREATED"), @ApiResponse(code = 401, message =
45     "UNAUTHORIZED"),
46     @ApiResponse(code = 415, message = "UNSUPPORTED TYPE - Representation not supported
47     for the resource"),
48     @ApiResponse(code = 500, message = "SERVER ERROR") })
49 public class ComercializacionSocioController implements ErrorController {
50     private static final String PATH = "/error";
51     public static final Logger LOGGER =
52     LoggerFactory.getLogger(ComercializacionSocioController.class);
53
54     /*****
55     * SECCION - INYECCION DE DEPENDENCIAS
56     *****/
57     @Autowired
58     @Qualifier("consumer")
59     private Consumer consumer;
60
61     @Autowired
62     @Qualifier("convertEntityUtil")
63     private ConvertEntityUtil convertEntityUtil;
64
65     @Autowired
66     @Qualifier("util")
67     private Util util;
68
69     /*****
70     * SECCION - MICROSERVICIOS
71     *****/
72     @Value("${url.servidor_micro}")
73     private String urlServidor;
74
75     @Value("${url.servidor_mag}")
76     private String urlServidorMag;
77
78     @Value("${url.comercializacion_socio}")
79     private String urlComercializacionSocio;
80
81     /*****
82     * SECCION - END-POINTS
83     *****/
84
85     /*****
86     * Obtiene todas las entidades con un token como parámetro
87     * @param token
88     * @return
89     * @throws NoSuchFieldException
90     * @throws SecurityException
91     * @throws IllegalArgumentException
92     * @throws IllegalAccessException
93     * @throws IOException

```



```

93     */
94     @GetMapping(value = "/ComercializacionSocio/findAll")
95     @ApiOperation(value = "Obtiene todos los datos de Comercializacion Socio ", response =
Object.class)
96     @ResponseStatus(HttpStatus.OK)
97     public Object obtenerComercializacionSocio(@RequestHeader(name = "Authorization")
String token) throws NoSuchFieldException, SecurityException, IllegalArgumentException,
IllegalAccessException, IOException{
98
99         String pathMicro = urlServidorMag + urlComerzializacionSocio +
"/comercializacion_socio/findAll" ;
100         //Object response = consumer.doGet(pathMicro, token);
101         Object response = consumer.doGet(pathMicro, token);
102         LOGGER.info("comercializacion_socio/findAll" + " usuario: " + util.filterUsuId
(token));
103         return ResponseEntity.ok(response);
104     }
105 }
106 /*****
107  * Obtiene todas las entidades dado un ID y un token de seguridad
108  * @param csoId
109  * @param token
110  * @return
111  * @throws NoSuchFieldException
112  * @throws SecurityException
113  * @throws IllegalArgumentException
114  * @throws IllegalAccessException
115  * @throws IOException
116  */
117 @GetMapping(value = "/ComercializacionSocio/findById/{id}")
118 @ApiOperation(value = "Obtiene las Comercializaciones de Socios por id", response =
Object.class)
119 @ResponseStatus(HttpStatus.OK)
120 public Object obtenerComercializacionSocioById(@PathVariable Long csoId,
@RequestHeader(name = "Authorization") String token) throws NoSuchFieldException,
SecurityException, IllegalArgumentException, IllegalAccessException, IOException{
121
122     String pathMicro = urlServidorMag + urlComerzializacionSocio +
"/comercializacion_socio/findById/" + csoId;
123     //Object response = consumer.doGet(pathMicro, token);
124     Object response = consumer.doGet(pathMicro, token);
125     LOGGER.info("comercializacion_socio/findById/" + " usuario: " + util.filterUsuId
(token));
126     return ResponseEntity.ok(response);
127 }
128 }
129 /*****
130  * Crear una entidad tipo Comercialización Socio dado un nombre y un token
131  *
132  * @param comercializacionSocioNombre
133  * @param token
134  * @return
135  * @throws NoSuchFieldException
136  * @throws SecurityException
137  * @throws IllegalArgumentException
138  * @throws IllegalAccessException
139  * @throws IOException
140  */
141 @PostMapping(value = "/ComercializacionSocio/create")
142 @ApiOperation(value = "Crea una nueva Comercializacion Socio", response = Object.class)
143 @ResponseStatus(HttpStatus.CREATED)
144 public ResponseEntity<?> saveComercializacionSocio(@Validated @RequestBody String

```

```


    comercializacionSocioNombre,
145     @RequestHeader(name = "Authorization") String token) throws
    NoSuchFieldException, SecurityException,
146     IllegalArgumentException, IllegalAccessException, IOException {
147     String pathMicro = urlServidorMag + urlComercializacionSocio +
    "comercializacion_socio/create/";
148     ComercializacionSocioDTO comercializacionSocioDTO =
    convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionSocioNombre, token,
    ComercializacionSocioDTO.class);
149     LOGGER.info("comercializacion_socio/create" + " usuario: " + util.filterUsuId
    (token));
150     return ResponseEntity.ok(comercializacionSocioDTO);
151 }
152 /*****
153  *
154  * Actualiza la entidad dado su ID, nombre y un token de autorización
155  *
156  * @param csoId
157  * @param comercializacionSocioNombre
158  * @param auth
159  * @return
160  * @throws NoSuchFieldException
161  * @throws SecurityException
162  * @throws IllegalArgumentException
163  * @throws IllegalAccessException
164  * @throws IOException
165  */
166 @PutMapping(value = "/ComercializacionSocio/update/{Id}")
167 @ApiOperation(value = "Actualiza un registro de Comercializacion Socio", response =
    Object.class)
168 @ResponseStatus(HttpStatus.CREATED)
169
170 public ResponseEntity<?> updateComercializacionSocio(@Validated @PathVariable Long
    csoId, @RequestBody String comercializacionSocioNombre,
171     @RequestHeader(name = "Authorization") String auth) throws
    NoSuchFieldException, SecurityException,
172     IllegalArgumentException, IllegalAccessException, IOException {
173     String pathMicro = urlServidor + urlComercializacionSocio +
    "comercializacion_socio/update/" + csoId;
174     ComercializacionSocioDTO comercializacionSocioDTO =
    convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionSocioNombre, auth,
    ComercializacionSocioDTO.class);
175     LOGGER.info("comercializacion_socio/update/: entidad:" +
    comercializacionSocioNombre + " usuario: " + csoId);
176     return ResponseEntity.ok(comercializacionSocioDTO);
177 }
178
179
180 /*****
181  * Actualiza por la llave foránea OrgId, nombre y token de seguridad
182  *
183  * @param orgId
184  * @param comercializacionSocioNombre
185  * @param auth
186  * @return
187  * @throws NoSuchFieldException
188  * @throws SecurityException
189  * @throws IllegalArgumentException
190  * @throws IllegalAccessException
191  * @throws IOException
192  */
193 @PutMapping(value = "/ComercializacionSocio/updateByOrgId/{Id}")

```

```

194     @ApiOperation(value = "Actualiza un registro de Comercializacion de Socios", response =
Object.class)
195     @ResponseStatus(HttpStatus.CREATED)
196
197     public ResponseEntity<?> updateComercializacionSocioByOrgId(@Validated @PathVariable
Long orgId, @RequestBody String comercializacionSocioNombre,
198         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
199         IllegalArgumentException, IllegalAccessException, IOException {
200         String pathMicro = urlServidor + urlComercializacionSocio +
"comercializacion_socio/updateByOrgId/" + orgId;
201         ComercializacionSocioDTO comercializacionSocioDTO =
convertEntityUtil.ConvertSingleEntityPOST(pathMicro, comercializacionSocioNombre, auth,
ComercializacionSocioDTO.class);
202         LOGGER.info("comercializacion_socio/updateByOrgId/: entidad:" +
comercializacionSocioNombre + " usuario: " + orgId);
203         return ResponseEntity.ok(comercializacionSocioDTO);
204     }
205     /*****
206     * Realiza un eliminado lógico dado un ID de la entidad
207     *
208     * @param csoId
209     * @param auth
210     * @return
211     * @throws NoSuchFieldException
212     * @throws SecurityException
213     * @throws IllegalArgumentException
214     * @throws IllegalAccessException
215     * @throws IOException
216     */
217     @DeleteMapping(value = "/ComercializacionSocio/delete/{Id}")
218     @ApiOperation(value = "Elimina un registro en la tabla Comercializacion Socio",
response = Object.class)
219     @ResponseStatus(HttpStatus.OK)
220
221     public ResponseEntity<?> deleteComercializacionSocio(@Validated @PathVariable Long
csoId,
222         @RequestHeader(name = "Authorization") String auth) throws
NoSuchFieldException, SecurityException,
223         IllegalArgumentException, IllegalAccessException, IOException {
224         String pathMicro = urlServidor + urlComercializacionSocio +
"comercializacion_socio/delete/" + csoId;
225         Object comercializacionSocioDTO = consumer.doGet(pathMicro, auth);
226         LOGGER.info("comercializacion_socio/delete: id:" + csoId);
227         return ResponseEntity.ok(comercializacionSocioDTO);
228     }
229
230     @Override
231     public String getErrorPath() {
232         return PATH;
233     }
234 }
235

```

| Tipificación de errores | | |
|--|-----------------------|--|
| Código | Descripción | Mensaje Genérico |
| 200 | OK | OK |
| 400 | Bad Request | { "timestamp": "2020-04-21T09:27:15.848-0500", "status": 401, "error": "Unauthorized", "message": "Unauthorized", "path": " / micro_afc_centroAcopio/ centrog_acopio " } |
| 500 | Internal Server Error | Internal Server Error |
| Aprobación de ficha técnica | | |
| ELABORADO POR: | | REVISADO POR: |
|  | | |
| Mateo Alarcón Fausto Borja Oscar Narváez | | Carlos Burgos |



| APROBADO POR: |
|---------------|
| |
| |

