REST API – RESERVES

(MongoDB / Mongoose, Express, Node.js)

1. Model Espais

a) Action List

⇒ Request

- URL: https://localhost:8080/api/espais
- Opt. Filters: ?codi=A&descripcio=Espai+A
- Method: GET
- Headers: X_USERNAME: username
  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve los espacios que puede ver el usuario. Todos o filtrados, si es usuario admin.
- JSON:

[{"_id": "539f0f7b31d638b92408dd31", "codi": "A", "descripcio": "Espai A"},
{"_id": "539f0f7b31d638b92408dd32", "codi": "B", "descripcio": "Espai B"}]

b) Action Show

⇒ Request

- URL: https://localhost:8080/api/espais/:id
- Method: GET
- Headers: X_USERNAME: username
  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve el espacio correspondiente al id indicado en la URL. Si el usuario no puede ver el espacio, devuelve error 403. Si el espacio no existe, devuelve error 404.
- JSON:

{"_id": "539f0f7b31d638b92408dd31", "codi": "A", "descripcio": "Espai A"}

- Errors:

{"codError": 403, "descError": "SHOW: L'usuari no té permís per accedir a l'espai amb id=539f0f7b31d638b92408cc32"}

{"codError": 404, "descError": "SHOW: No existeix l'espai amb id=539f0f7b31d638b92408cc32"}

c) Action Create

⇒ Request

- URL: https://localhost:8080/api/espais/
- Method: POST
- Headers: Content-Type: application/x-www-form-urlencoded
  X_USERNAME: username
  X_PASSWORD: password_encriptat
- Data: codi=XX&descripcio=Descripció+XX

⇒ Response

- Return: Devuelve id del espacio creado. Si el usuario no es admin, devuelve error 403.
- JSON:

  {"_id": "539f0f7d31d638b92408dd35"}
- Errors:

  {"codError": 403, "descError": "CREATE: L'usuari no té permís per crear espais"}

d) Action Update

⇒ Request

- URL: https://localhost:8080/api/espais/:id
- Method: PUT
- Headers: Content-Type: application/json
  X_USERNAME: username
  X_PASSWORD: password_encriptat
- Data: {“descripcio": “Espai XX”},

⇒ Response

- Return: Devuelve id del espacio actualizado. Si el usuario no es admin, devuelve error 403. Si no existe el espacio, devuelve error 404.
- JSON:

  {"_id": "539f0f7b31d638b92408dd34"}
- Errors:

  {"codError": 403, "descError: "UPDATE: L'usuari no té permís per actualitzar espais"}

  {"codError": 404, "descError": "UPDATE: No existeix l'espai amb id=539f0f7b31d638b92408dd34"}

e) Action Delete

⇒ Request

- URL: https://localhost:8080/api/espais/:id
- Method: DELETE
- Headers: X_USERNAME: username

  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve id del espacio borrado. Si el usuario no es admin, devuelve error 403. Si no existe el espacio, devuelve error 404.
- JSON:

  {"_id": "539f0f7b31d638b92408dd33"}
- Errors:

  {"codError": 403, "descError: "DELETE: L'usuari no té permís per esborrar espais"}

  {"codError": 404, "descError": "UPDATE: No existeix l'espai amb id=539f0f7b31d638b92408dd33"}

2. Model Usuaris

a) Action List

⇒ Request

- URL: https://localhost:8080/api/usuaris

- Opt. Filters:

  ?nom_usuari=user&nom=nom&cognoms=cognoms&es_admin=true
- Method: GET
- Headers: X_USERNAME: username

  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve los usuarios que puede ver el usuario conectado. Todos o filtrados, si es usuario admin. Si no es admin, devuelve el propio usuario.
- JSON:

  [{"_id": "539ec5f8e1bb95101005695b", "nom_usuari": "user", "contrasenya": "password", "nom": "nom", "cognoms": "cognoms", "es_admin": true},
  {"_id": "539ee05ae310205c0830bb27", "nom_usuari": "user2", "contrasenya": "password2", "nom": "nom2", "cognoms": "cognoms2", "es_admin": false}]

b) Action Show

⇒ Request

- URL: https://localhost:8080/api/usuaris/:id
- Method: GET
- Headers: X_USERNAME: username

  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve el usuario correspondiente al id indicado en la URL. Si el usuario conectado no puede ver el usuario, devuelve error 403. Si el espacio no existe, devuelve error 404.
- JSON:

  {"id": "2", "nom_usuari": "user2", "contrasenya": "password2", "nom": "nom2", "cognoms": "cognoms2", "es_admin": false}
- Errors:

  {"codError": 403, "descError": "SHOW: L'usuari no té permís per accedir a l'usuarii amb id=539f31e777ade655d079bde5"}
  {"codError": 404, "descError": "SHOW: No existeix l'usuari amb id=539f31e777ade655d079bde6"}

c) Action Create

⇒ Request

- URL: https://localhost/:8080/api/usuaris/
- Method: POST
- Headers: Content-Type: application/x-www-form-urlencoded
  X_USERNAME: username
  X_PASSWORD: password_encriptat
- Data:

  nom_usuari=username&contrasenya=password_ecriptat&nom=nom&cog-noms=cognom1+cognom2&es_admin=true/false

⇒ Response

- Return: Devuelve id del usuario creado. Si el usuario conectado no es admin, devuelve error 403.
- JSON:

  {"_id": "11"}
- Errors:

  {"codError": 403, "descError": "CREATE: L'usuari no té permís per crear usuaris"}

d) Action Update

⇒ Request

- URL: https://localhost/api/usuaris/:id
- Method: PUT
- Headers: Content-Type: application/json
  X_USERNAME: username
  X_PASSWORD: password_encriptat

Data: {"nom": “nou_nom”, es_admin: true} , etc.
Para dar permiso usuario-espacios:
{"espais": [“539f0f7b31d638b92408dd31”,
"539f0f7b31d638b92408dd32",

"539f0f7b31d638b92408dd33"]}

⇒ Response

- Return: Devuelve id del usuario actualizado. Si el usuario conectado no es admin, devuelve error 403. Si no existe el usuario a actualizar, devuelve error 404.
- JSON:

  {"_id": "539ec606e1bb95101005695c"}
- Errors:

  {"codError": 403, "descError: "UPDATE: L'usuari no té permís per actualitzar usuaris"}

  {"codError": 404, "descError": "UPDATE: No existeix l'usuari amb id=539ec606e1bb95101005695d"}

e) Action Delete

⇒ Request

- URL: https://localhost:8080/api/usuaris/:id
- Method: DELETE
- Headers: X_USERNAME: username

  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve id del usuario borrado. Si el usuario conectado no es admin, devuelve error 403. Si no existe el usuario a borrar, devuelve error 404.
- JSON:

  {"_id": "539f31e777ade655d079bde5"}
- Errors:

  {"codError": 403, "descError: "DELETE: L'usuari no té permís per esborrar usuaris"}

  {"codError": 404, "descError": "DELETE: No existeix l'usuari amb id=539f31e777ade655d079bdf5"}

3. Model Reserves

f) Action List

⇒ Request

- URL: https://localhost:8080/api/reserves
- Opt. Filters:
  ?inici=aaaammddhh&fi=aaaammddhh&espai=539f0f7b31d638b92408dd31
- Method: GET
- Headers: X_USERNAME: username
  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve las reservas que puede ver el usuario conectado. Si el usuario conectado es admin, puede filtrar por "usuari" y ver los datos de los usuarios de las reservas. Si no es admin, se muestra null en “usuari” cuando la reserva no es propia.
  Advertencia: las fechas que se guardan en Base de Datos y que son devueltas en JSON tienen formato ISO UTC "**aaaa-mm-ddThh:mi:ss.mssZ**". Esto implica lo siguiente:
  a) "data_modif": Contiene el timestamp de la hora no local (UTC) en el servidor (igual que en el Log).
  b) "data_hora": Contine fecha y hora igual al valor "aaaammddhh[miss]" enviado por el cliente, pero interpretado como UTC.
- JSON:

[{
"data_modif": "2014-06-19T15:08:23.133Z",
"usuari": {
"_id": "539ec5f8e1bb95101005695b",
"nom_usuari": "user",
"contrasenya": "password",
"nom": "nom",
"cognoms": "cognoms",
"es_admin": true,
"espais": [
]
},
"data_hora": "2014-06-19T18:00:00.000Z",

"espai": {
"_id": "539f0f7b31d638b92408dd31",
"codi": "A",
"descripcio": "Espai A"
},
"_id": "53a2fce79825061401f07f8b",
"__v": 0
},
{
"data_modif": "2014-06-19T15:28:50.363Z",
"usuari": {
"_id": "539ec5f8e1bb95101005695b",
"nom_usuari": "user",
"contrasenya": "password",
"nom": "nom",
"cognoms": "cognoms",
"es_admin": true,
"espais": [
]
},
"data_hora": "2014-06-19T19:00:00.000Z",
"espai": {
"_id": "539f0f7b31d638b92408dd31",
"codi": "A",
"descripcio": "Espai A"
},
"_id": "53a301b22782836005e08e53",
"__v": 0
}]

g) Action Show

⇒ Request

- URL: https://localhost:8080/api/reserves/:id
- Method: GET
- Headers: X_USERNAME: username

X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve la reserva correspondiente al id indicado en la URL. Si la reserva no existe, devuelve error 404. Si el usuario conectado es admin, puede ver lo datos de los usuarios de la reservas. Si no es admin, se muestra null en “usuari” cuando la reserva no es propia.
- JSON:

{
"data_modif": "2014-06-19T14:55:07.293Z",
"usuari": {
"_id": "539f31e777ade655d079bde5",
"nom_usuari": "user3",
"contrasenya": "password3",
"nom": "nom3",
"cognoms": "cognoms3",
"es_admin": false,
"espais": [
]
},
"data_hora": "2014-06-19T17:00:00.000Z",
"espai": {
"_id": "539f0f7b31d638b92408dd31",
"codi": "A",
"descripcio": "Espai A"
},
"_id": "53a2f9cb300abcfc06157e8b",
"__v": 0
}

- Errors:

{"codError": 404, "descError": "SHOW" No existeix la reserva amb id=53a2f9cb300abcfc06157e8c"}

h) Action Create

⇒ Request

- URL: https://localhost:8080/api/reserves/

- Method: POST
- Headers: Content-Type: application/x-www-form-urlencoded
  X_USERNAME: username
  X_PASSWORD: password_encriptat
- Data:
  data_hora=aaaammddhh&espai=539f0f7b31d638b92408dd31&usuari=539f31e777ade655d079bde5

⇒ Response

- Return: Devuelve el id de la reserva creada. Si el usuario conectado no es admin o no se envía "usuari", se guarda el propio usuario como el que ha realizado la reserva.
- JSON:
  {"_id": "53a2f9cb300abcfc06157e8b"}

i) Action Update

⇒ Request

- URL: https://localhost:8080/api/reserves/:id
- Method: PUT
- Headers: Content-Type: application/json
  X_USERNAME: username
  X_PASSWORD: password_encriptat

Data: {"usuari": ”539ec5f8e1bb95101005695b”}

⇒ Response

- Return: Devuelve id de la reserva actualizada. Si el usuario conectado no es admin, devuelve error 403. Si no existe la reserva a actualizar, devuelve error 404.
- JSON:
  {"_id": "53a2fce79825061401f07f8b"}
- Errors:
  {"codError": 403, "descError: "UPDATE: L'usuari no té permís per actualitzar reserves"}

{"codError": 404, "descError": "UPDATE: No existeix la reserva amb id=53a301b22782836005e08e53"}

j) Action Delete

⇒ Request

- URL: https://localhost:8080/api/reserves/:id
- Method: DELETE
- Headers: X_USERNAME: username
  X_PASSWORD: password_encriptat

⇒ Response

- Return: Devuelve id de la reserva borrada. Si el usuario conectado no es admin, devuelve error 403. Si no existe la reserva a borrar, devuelve error 404.
- JSON:
  {"_id": "53a301b22782836005e08e53"}
- Errors:
  {"codError": 403, "descError: "DELETE: L'usuari no té permís per esborrar reserves"}
  {"codError": 404, "descError": "DELETE: No existeix la reserva amb id=53a301b22782836005e08f53"}