

Universidad ORT Uruguay

Facultad de Ingeniería

Evidencia del diseño y especificación de la API.

Ignacio Loureiro - 191659

Malvina Jaume - 151281

Entregado como requisito de la materia

Diseño de Aplicaciones 2

14 de mayo de 2020

Índice general

1. Criterios seguidos	2
1.1. Descripción de criterios REST	2
1.2. Descripción del mecanismo de autenticación de requets	2
1.3. Descripción general de códigos de error	4
2. Descripción de los resources de la API.	6
2.1. URL base.	6

1. Criterios seguidos

El siguiente documento ha sido realizado para describir el diseño realizado para solución de la implementación del API REST descrita en la letra del obligatorio entregado.

1.1. Descripción de criterios REST

Para el desarrollo de esta aplicación se tuvieron en cuenta las buenas prácticas aprendidas, los principios REST.

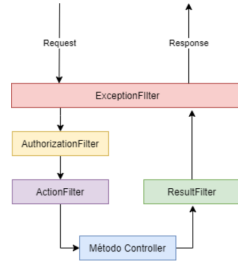
- Separamos los recursos lógicos En esta instancia identificamos 4:
 - Admins (Manejo de administradores)
 - Requests (Manejo de solicitudes)
 - Sessions (Manejo de sesiones)
 - Types (Manejo de diferentes tipos de solicitudes)
- Para cada recurso le definimos las acciones necesarias que deben de realizar usando métodos HTTP. NOTA: En la parte del informe swagger si puede ver la definición de los métodos usados.
- Endpoints en plural
- Generamos documentación con herramienta para que quede entendible por todos
- Utilizamos filtros, para la autenticación.

1.2. Descripción del mecanismo de autenticación de requets

Al tener la necesidad de realizar una autenticación o autorización de los request nos tenemos que asegurar de que el cliente que realiza esta solicitud tenga los permisos necesarios, para esto lo que se decidió aplicar en el solución fue la implementación de un filtro de seguridad en la WebApi.

ASP.NET Web Api proporciona una opción de extensibilidad para la realización del filtro. Los filtros nos permiten la ejecución de código antes o después de un procesamiento de una solicitud HTTP.

El orden de ejecución sería el siguiente:



Cada usuario al loguearse generará un token aleatorio. Este token será el usado para validar que el usuario relacionado a dicho token tiene los permisos necesarios para realizar el request.

A grandes rasgos los pasos para la implementación fue:

- Dentro de WebApi creamos una clase que representa al filtro **AuthenticationFilter** la cual extiende de **Attribute** e implementa la interfaz **IActionFilter**. Esta clase es la responsable de la validación del encabezado que contenga un identificador. Validará que el token es el correcto.
- Se crea una clase **SessionsController** también en el paquete WebApi. Esta clase es la responsable de la comunicación con la lógica.
- Dentro del paquete Logic implementamos la clase que hace posible todo lo necesario para la acción de login, la validación y registro del token en la base de datos.

Para que el filtro se aplique en la request, lo único que es necesario tener es tener el tag en el request que se quiere aplicar. En la siguiente imagen mostramos como se utiliza en la request GetAll colocamos [Filters.AuthenticationFilter]

- 404 : Falla cuando se intenta borrar o actualizar un recurso inexistente
- 500 : Falla en el servidor

2. Descripción de los resources de la API.

2.1. URL base.

La url base de nuestra aplicación está formada por el hostname, en nuestro caso, con un desarrollo local **http://localhost:5000** y cada recurso está prefijado por la palabra “api”. Por lo tanto la url base de nuestra WebApi, corriendo localmente es **http://localhost:5000/api**

La api fue documentada a medida que se desarrollaba utilizando swagger. para utilizar swagger tuvimos que:

- descargar el paquete de swashbuckle, una implementación de Swagger para .NET Core <https://www.nuget.org/packages/Swashbuckle.AspNetCore/>
- configurar el startup, siguiendo la guía de Microsoft <https://www.nuget.org/packages/Swashbuckle.AspNetCore/>
- para poder generar metadata de la Api es necesario decorar los métodos con sintaxis conocida por Swagger.

```
/// <summary>
/// Creates a new request in the system
/// </summary>
/// <param name="request">request body</param>
/// <response code="200">Request created</response>
/// <response code="400">There's something wrong with the request body</response>
/// <response code="404">A field name could not be found</response>
/// <response code="500">Something is wrong with the server</response>
[HttpPost]
public ActionResult CreateRequest([FromBody] CreateRequest request)
```

En la imagen anterior se puede ver como está marcado nuestro endpoint, lo que ocasiona que Swagger pueda presentar la metadata de nuestra operación en el navegador.

POST

/api/Requests

Creates a new request in the system

Parameters

Try it out

No parameters

Request body

application/json

request body

Example Value

Schema

```
{
  "details": "string",
  "email": "string",
  "name": "string",
  "phone": "string",
  "typeId": 0,
  "additionalFields": [
    {
      "name": "string",
      "value": "string"
    }
  ]
}
```

Responses

Code	Description	Links
200	Request created	No links
400	There's something wrong with the request body	No links
404	A field name could not be found	No links
500	Something is wrong with the server	No links

Una vez levantada la app se puede ingresar en localhost:5000/swagger/index.html

IMM Request API v1 OAS3

</swagger/v1/swagger.json>

Admins



GET

</api/Admins> Get all the admin users

Parameters

Try it out

No parameters

Responses

Code

Description

Links

200

Success

No links

Media type

text/plain



Controls Accept header.

Example Value | Schema

```
[
  {
    "password": "string",
    "token": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "id": 0,
    "name": "string",
    "email": "string",
    "phoneNumber": "string"
  }
]
```

500

Something is wrong with the server

No links

POST

</api/Admins> Create new admin user

Parameters

Try it out

No parameters

Request body

application/json



request body

Example Value | Schema

```
{
  "password": "string",
  "name": "string",
  "email": "string",
  "phoneNumber": "string"
}
```

Responses		
Code	Description	Links
200	Success	No links
500	Something is wrong with the server	No links

PUT

/api/Admins/{id} Update admin user

Parameters

Try it out

Name	Description
<div><div>id * required</div><div>integer</div><div>(path)</div></div>	<div>id</div>

Request body

application/json

request body

Example Value | Schema

```
{  "password": "string",  "name": "string",  "email": "string",  "phoneNumber": "string"}
```

Responses

Code	Description	Links
200	Success	No links
500	Something is wrong with the server	No links

DELETE

/api/Admins/{id} Delete admin user

Parameters

Try it out

Name	Description
<div><div>id * required</div><div>integer</div><div>(path)</div></div>	<div>id</div>

Responses

Code	Description	Links
200	Success	No links
500	Something is wrong with the server	No links

Requests



POST /api/Requests Creates a new request in the system

Parameters

Try it out

No parameters

Request body

application/json

request body

Example Value | Schema

```
{
  "details": "string",
  "email": "string",
  "name": "string",
  "phone": "string",
  "typeId": 0,
  "additionalFields": [
    {
      "name": "string",
      "value": "string"
    }
  ]
}
```

Responses

Code	Description	Links
200	Request created	No links
400	There's something wrong with the request body	No links
404	A field name could not be found	No links
500	Something is wrong with the server	No links

GET /api/Requests Gets a list of all the status in the system

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Success	No links
<div>Media type</div> <div>text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>[{ "requestId": 0, "requestedBy": "string", "status": "string", "details": "string" }]</pre></div>		

GET /api/Requests/{id} Gets a request details

Parameters

Try it out

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
200	Success	No links
<div>Media type</div> <div>text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>[{ "requestId": 0, "requestedBy": "string", "status": "string", "details": "string" }]</pre></div>		

PUT /api/Requests/{id} Updates the State of a request

Parameters

Try it out

<

<div>POST</div> <div>/api/Sessions</div> <div>Login in the system</div>	
Parameters	<div>Try it out</div>
No parameters	
Request body	<div>application/json</div>
<div>Example Value Schema</div> <div><pre>{ "email": "string", "password": "string"}</pre></div>	
Responses	

Code	Description	Links
200	Success	No links

Types



POST

/api/Types

Creates a new type in the system

Parameters

Try it out

No parameters

Request body

application/json

request body

Example Value

Schema

```
{
  "name": "string",
  "topicId": 0,
  "additionalFields": [
    {
      "name": "string",
      "fieldType": "string",
      "isRequired": true,
      "range": [
        {
          "name": "string",
          "value": "string"
        }
      ]
    }
  ]
}
```

Responses

Code	Description	Links
200	Type created	No links
400	There's something wrong with the request body	No links
500	Something is wrong with the server	No links

DELETE

/api/Types/{id}

Creates a new type in the system

Parameters

Try it out

Name	Description
<div><div>id * required</div><div>integer</div><div>(path)</div></div>	Type id

id - Type id

Responses

Code	Description	Links
200	Type created	No links
400	There's something wrong with the request body	No links
500	Something is wrong with the server	No links

Schemas

Admin ▾ {
 password string
 nullable: true
 token string(\$uuid)
 id integer(\$int32)
 name string
 nullable: true
 email string
 nullable: true
 phoneNumber string
 nullable: true
}

CreateAdminRequest ▾ {
 password string
 nullable: true
 name string
 nullable: true
 email string
 nullable: true
 phoneNumber string
 nullable: true
}

FieldRequestModel ▾ {
 name string
 nullable: true
 value string
 nullable: true
}

```
CreateRequest ∨ {
  details      string
               nullable: true
  email        string
               nullable: true
  name         string
               nullable: true
  phone        string
               nullable: true
  typeId       integer($int32)
  additionalFields ∨ [
                  nullable: true
    FieldRequestModel ∨ {
      name      string
                 nullable: true
      value     string
                 nullable: true
    }
  ]
}
```

```
GetAllRequestsStatusResponse ∨ {
  requestId    integer($int32)
  requestedBy  string
               nullable: true
  status       string
               nullable: true
  details      string
               nullable: true
}
```

```
UpdateStateRequest ∨ {
  newState     string
               nullable: true
}
```

```
ModelAdminLogin ∨ {
  email        string
               nullable: true
  password     string
               nullable: true
}
```

```
NewTypeAdditionalField ∨ {
  name         string
               nullable: true
  fieldType    string
               nullable: true
  isRequired   boolean
  range        > [...]
}
```



```
CreateTypeRequest {
  name          string
               nullable: true
  topicId       integer($int32)
  additionalFields {
    NewTypeAdditionalField {
      name          string
                   nullable: true
      fieldType     string
                   nullable: true
      isRequired    boolean
      range         {
        FieldRequestModel {
          name      string
                   nullable: true
          value     string
                   nullable: true
        }
      }
    }
  }
}
```