Documentación servicios en **Plugcore**

plugcore.com

Plug Soluciones TIC S.L.
info@plugcore.com

https://github.com/plugcore/plugcore

https://www.linkedin.com/company/plugcore

https://twitter.com/plugcoreit

Documentación

1. **Introducción**
2. **¿Cómo documentar una ruta?**
3. **Visualizador de documentación**
4. **Configuración**

Introducción

- Todo lo que vamos a comentar hoy está en el paquete @plugcore/web, con lo cual si no lo tenemos instalado lo vamos a necesitar.

- Para que sea sencillo conectarse con nuestra API siempre es recomendable crear algún tipo de documentación, y a poder ser que esté actualizada.

- Para ayudarnos además vamos a utilizar el sistema de validación explicado en @plugcore/core, de esta manera documentamos y validamos a la vez.

- La documentación la haremos generando automáticamente un documento https://github.com/OAI/OpenAPI-Specification OAS y lo podremos visualizar y usar usando Swagger (de los creadores de SoapUI)

¿Cómo documentar una ruta?

- Lo primero que tenemos que crear son los modelos, para ello utilizaremos el sistema de validación de objetos de plugcore, es decir crear clases y decorar las propiedades con “@IsString()”, “@IsNumber()”, etc.

- Una vez tengamos los modelos creados tendremos que ir a la ruta y definir la propiedad “routeSchemas”, la cual tendrá los campos:
  - `request:` Body de la petición, no sirve si es una petición GET
  - `response:` Estructura de la respuesta
  - `query:` Los parámetros que van en la url, ej: domain.com**?a=123&b=456**
  - `urlParameters:` En el caso de definir una ruta con identificador en la ruta, ejemplo: /api/post/**:id**, este id puede estar documentado y validado también.
  - `headers:` Cabeceras que pueda necesitar el servicio

Ejemplo de modelo

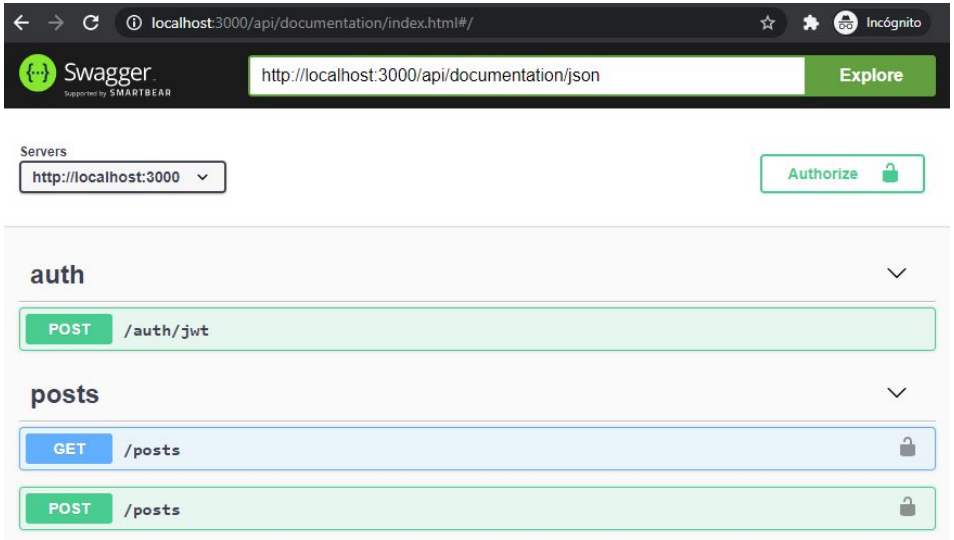
```
// Aquí por ejemplo extendemos un modelo creado previamente
// y tomamos todas su propiedades menos el id
@ExtendsSchema(PostModel, { ignoreProperties: ['id'] })
export class CreatePostReqSchema { }
// Ejemplo típico de model
export class CreatePostResSchema {
    @IsBoolean()
    success: boolean;
    @IsString({ maxLength: 128 })
    newId: string;
}
// Usamos los tipos en vez de las clases para tener más flexibilidad
// dado que podemos extender modelos usando las clases
export type CreatePostReq = Omit<PostModel, 'id'>;
export type CreatePostRes = CreatePostResSchema;
```

Ejemplo ruta

```
@Post({
    security: 'jwt',
    routeSchemas: {
        tags: ['posts'],
        request: CreatePostReqSchema,
        response: CreatePostResSchema
    }
})
public async createPost(request: Request<CreatePostReq>, response: Response) {
    this.log.info(request.body);
    return <CreatePostRes>{
        success: true,
        newId: '1'
    };
}
```

Visualizar documentación

- Una vez documentada nuestra ruta podemos ver el documento OAS generado en: http://localhost:3000/api/documentation/json y visualizarlo en http://localhost:3000/api/documentation (la ruta es configurable)



Configuración

```
{
    "web": {
        "oas": { // Todo lo relacionado con OAS va en esta propiedad, y todo es OPCIONAL
            "enableDocumentation": true, // Activar/desactivar la generación de documentación
            "documentationPath": "/api/documentation", // Url de la documentación, por
defecto /api/documentation
            // A partir de aquí todas las propieades son las que se definen en
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md
            // y hacen referencia a las propiedades generales del documento, no las que son
específicas de cada método/url expuesta
        },
        ...
    }
    ...
}
```

GRACIAS POR SU TIEMPO

Para más información
info@plugcore.com
germanml@plugcore.com
sergiolc@plugcore.com

plugcore.com