

OPENAPI

PROGRAMACIÓN 3
PROF. MANGO EDUARDO

CONTENIDO

- | | |
|----|--------------------------|
| 01 | Documentación de una API |
| 02 | ¿Que es OpenAPI? |
| 03 | Documentar con OpenAPI |

Documentación de una API

La documentación es un pilar fundamental en el desarrollo de APIs, actuando como el manual de instrucciones esencial para cualquier desarrollador o sistema que necesite interactuar con ella. Una documentación clara, precisa y actualizada reduce drásticamente la curva de aprendizaje, facilita la integración, minimiza los errores y acelera el proceso de desarrollo.

Sin una documentación adecuada, las APIs se vuelven opacas y difíciles de utilizar, limitando su potencial y generando frustración entre los desarrolladores.



Documentaciones

Toda API desarrollada debe estar acompañada de su documentación. Aquí algunos ejemplos

OpenAI - <https://platform.openai.com/docs/api-reference/introduction>

GitHub Rest API - <https://docs.github.com/en/rest?apiVersion=2022-11-28>

Stripe - <https://docs.stripe.com/api>

Muchos proveedores de servicios web como AWS, Google Cloud y Azure ya están optimizados para trabajar con OpenAPI de manera nativa, facilitando mucho el despliegue de nuestra API.

OpenAPI

¿QUE ES?

Antes de OpenAPI, la documentación de las APIs a menudo era inconsistente, incompleta o estaba dispersa, lo que dificultaba enormemente su uso por parte de otros desarrolladores o sistemas.

La Especificación OpenAPI (anteriormente conocida como Swagger Specification) es un formato estándar y agnóstico de lenguaje para describir APIs REST. Permite definir la estructura, los endpoints, los parámetros, las respuestas, la autenticación y otros aspectos de una API de una manera legible tanto por humanos como por máquinas.

Anotaciones

Anotacion	¿Qué hace?	Parametros comunes
@Tag	Agrupar operaciones bajo una categoría o grupo común.	name, description
@Operation	Documenta una operación específica (endpoint).	summary, description, tags, operationId, parameters, responses, security
@Parameter	Describe un parámetro de la operación.	name, in, description, required, schema
@RequestBody	Describe el cuerpo de una solicitud HTTP.	description, required, content
@ApiResponse	Documenta una posible respuesta de un endpoint.	responseCode, description, content
@Content	Define el tipo de contenido de una respuesta o request.	mediaType, schema, examples

@Tag

```
@RestController  @EduardoMango
@RequestMapping(⌘"/auth")
@Tag(name = "Authentication", description = "Operations related to user authentication and account management")
public class AuthController {

    private final JwtService jwtService;  2 usages

    private final AuthService authenticationService;  2 usages

    @Autowired  @EduardoMango
    public AuthController(JwtService jwtService, AuthService authenticationService) {
        this.jwtService = jwtService;
        this.authenticationService = authenticationService;
    }
}
```

Documentando un Endpoint

```
@Operation(  📄 EduardoMango *  
    summary = "Authenticate a user and return a JWT token",  
    description = "This endpoint authenticates a user with their credentials and " +  
        "returns a JWT token if the authentication is successful."  
)  
  
@ApiResponses(value = {  
    @ApiResponse(responseCode = "200", description = "Successful authentication",  
        content = @Content(schema = @Schema(implementation = AuthResponse.class))),  
    @ApiResponse(responseCode = "401", description = "Invalid credentials")  
})  
  
@PostMapping(🌐"/login")  
public AuthResponse authenticate(@RequestBody AuthRequest loginUserDto) {  
    CredentialsEntity authenticatedUser = authenticationService.authenticate(loginUserDto);  
    String jwtToken = jwtService.generateToken(authenticatedUser);  
    return new AuthResponse(jwtToken);  
}
```


Documentando un Endpoint

```
@Operation(  EduardoMango
    summary = "Update user details",
    description = "Updates the information of an existing user based on the " +
        "provided user ID and new details.",
    parameters = {
        @Parameter(name = "id", description = "ID of the user to be updated",
            required = true)
    },
    requestBody = @io.swagger.v3.oas.annotations.parameters.RequestBody(
        description = "User details to be updated",
        required = true,
        content = @Content(
            mediaType = "application/json",
            schema = @Schema(implementation = UpdateUserDTO.class)
        )
    )
)
```

OPEN API

PROGRAMACIÓN 3
PROF. MANGO EDUARDO



Universidad Tecnológica Nacional de Mar del Plata