

**Diseño de Aplicaciones 2**  
Primer Obligatorio



Cristian Palma 208443 Federico Alonso 182999

**Evidencia del diseño y especificación de la API**

**Grupo N6A**

Repositorio: https://github.com/ORT-DA2/182999\_208443

**Índice**

[1. Descripción de la API 2](#_Toc84443451)

[1.1 Criterios REST 2](#_Toc84443452)

[1.2 Descripción del mecanismo de autenticación de request 2](#_Toc84443453)

[1.3 Descripción general de códigos de error utilizados 3](#_Toc84443454)

[1.4 Descripción de los resources de la API 3](#_Toc84443455)

[1.4.1 Usuarios 3](#_Toc84443456)

[1.4.2 Reportes 4](#_Toc84443457)

[1.4.3 Proyectos 4](#_Toc84443458)

[1.4.4 Login 5](#_Toc84443459)

[1.4.5 Incidentes 5](#_Toc84443460)

[1.4.6 Importaciones 6](#_Toc84443461)

[1.4.7 Estados 6](#_Toc84443462)

[1.4.8 Asociaciones 6](#_Toc84443463)

# Descripción de la API

## Criterios REST

Se desarrollo una API utilizando el estilo arquitectónico REST teniendo en cuenta los siguientes puntos:

1. Siempre usar sustantivos y no verbos.
2. Plural antes singular para facilitar la interpretación de los recursos.
3. Intuitiva y simple manteniendo durante todo el desarrollo el mismo criterio como por ejemplo el idioma por parte del equipo.
4. El manejo de errores utilizando los status recomendados que se van a detallar en los siguientes puntos.

## Descripción del mecanismo de autenticación de request

Para el proceso de autenticación se creo un controlador especifico llamado: “LoginController” en la siguiente ruta: **~ /api/Login.**

Para hacer login debemos enviar mediante HTTP POST un request de login y como respuesta vamos a obtener un token y código 200. El token se almacenará en la BD y no expira a menos que ejecutemos el request logout **~ /api/Logout**. que se encarga de borrar el token antes generado de la BD.

## Descripción general de códigos utilizados

* 200: significa que salió todo bien.
* 201: significa que el objeto se creó correctamente.
* 204: significa que la eliminación fue exitosa
* 401: significa que no estamos autorizados a utilizar el recurso.
* 404: significa que no se encontró el objeto o recurso.
* 409: significa que existió un conflicto y no se pudo procesar la solicitud, como por ejemplo si queremos agregar un desarrollador inexistente a un proyecto.
* 422: significa que la entidad no se puede procesar, por ejemplo, por motivos que se envió información invalida (por ejemplo: un nombre con menos o más caracteres de lo aceptado)
* 500: errores que no se pudieron controlar de parte de nuestro servicio.

## Descripción de los resources de la API

### 1.4.1 Usuarios

* EndPoint: ~/api/Usuarios
* Verbos HTTP: POST
* Descripción : Como administrador dar de Alta un nuevo usuario.
* Parámetros {NombreUsuario, Nombre, Apellido, Contraseña, Email, Rol}
* Responses {201, 401, 422, 500 }
* Headers: { **Key**: autorización.

**Value**: el token que me fue asignado al hacer login con rol administrador }

* EndPoint: ~/api/Usuarios
* Verbos HTTP: GET
* Descripción : Como administrador obtener todos los usuarios registrados.
* Parámetros { }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Usuarios
* Verbos HTTP: GET
* Descripción : Como administrador obtener los datos de un usuario en particular.
* Parámetros { id }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización. / **Value**: token }
* EndPoint: ~/api/Usuarios
* Verbos HTTP: GET
* Descripción : Como administrador obtener lista de usuarios con un rol en particular.
* Parámetros { rol }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }

### 1.4.2 Reportes

* EndPoint: ~/api/Reportes
* Verbos HTTP: GET
* Descripción : Como administrador obtener la cantidad de incidentes por proyecto.
* Parámetros { }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Reportes
* Verbos HTTP: GET
* Descripción : Como administrador obtener la cantidad de incidentes resueltos por un desarrollador.
* Parámetros { idUsuario }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }

### 1.4.3 Proyectos

* EndPoint: ~/api/Proyectos
* Verbos HTTP: GET
* Descripción : Como administrador obtener todos los proyectos registrados.
* Parámetros { }
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Proyectos
* Verbos HTTP: GET
* Descripción : Como administrador obtener los datos de un proyecto en particular.
* Parámetros { id}
* Responses {200, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Proyectos
* Verbos HTTP: POST
* Descripción : Como administrador dar de alta un nuevo proyecto.
* Parámetros { Nombre}
* Responses {201, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Proyectos
* Verbos HTTP: DELETE
* Descripción : Como administrador dar de baja un proyecto.
* Parámetros { id}
* Responses {204, 401, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Proyectos
* Verbos HTTP: PUT
* Descripción : Como administrador modificar el nombre de un proyecto.
* Parámetros { Nombre}
* Responses {204, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }

### 1.4.4 Login

* EndPoint: ~/api/Login
* Verbos HTTP: POST
* Descripción : loguearme en el sistema (Genera un token en la BD para el usuario)
* Parámetros { NombreUsuario, Contrasenia}
* Responses {200 + token, 401, 422, 500 }
* Headers: {}
* EndPoint: ~/api/Logout
* Verbos HTTP: POST
* Descripción : desloguearme del sistema (elimina el token de la BD)
* Parámetros { }
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: el token que me fue asignado al hacer login }

### 1.4.5 Incidentes

* EndPoint: ~/api/Incidentes
* Verbos HTTP: GET
* Descripción : Como administrador obtener todos los incidentes (bug) generados en el sistema de todos los proyectos.
* Parámetros { }
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Incidentes
* Verbos HTTP: GET
* Descripción : Como administrador obtener un incidente.
* Parámetros { id }
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Incidentes
* Verbos HTTP: POST
* Descripción : Como tester dar de alta un incidente a un proyecto que pertenezco.
* Parámetros { Nombre, ProyectoId, Descripción, Versión, EstadoIncidente, DesarrolladorId, UsuarioId}
* Responses {201, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Incidentes
* Verbos HTTP: DELETE
* Descripción : Como administrador eliminar cualquier incidente o como tester eliminar un incidente de un proyecto al cual pertenezco.
* Parámetros { Id}
* Responses {204, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Incidentes
* Verbos HTTP: PUT
* Descripción : Como administrador modificar cualquier incidente o como tester modificar un incidente de un proyecto al cual pertenezco.
* Parámetros { Nombre, ProyectoId, Descripcion, Version, EstadoIncidente, DesarrolladorId, UsuarioId }
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }

### 1.4.6 Importaciones

* EndPoint: ~/api/Importaciones
* Verbos HTTP: POST
* Descripción : Dar de alta una colección de incidentes desde un archivo TXT o XML a un proyecto.
* Parámetros { rutaFuente}
* Responses {201, 401, 422, 500 }
* Headers: {}

### 1.4.7 Estados

* EndPoint: ~/api/Estados
* Verbos HTTP: PUT
* Descripción : Como desarrollador modificar el estado de un incidente (activo / resuelto ) de un proyecto al cual pertenezco.
* Parámetros { idIncidente}
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }

### 1.4.8 Asociaciones

* EndPoint: ~/api/Asociaciones
* Verbos HTTP: GET
* Descripción : Como desarrollador o como tester obtener la lista de proyectos de los cuales pertenece.
* Parámetros { idUsuario}
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Asociaciones
* Verbos HTTP: GET
* Descripción : Como desarrollador o como tester obtener un proyecto al cual pertenezco
* Parámetros { idUsuario, idProyecto}
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Asociaciones
* Verbos HTTP: GET
* Descripción : Como desarrollador o como tester obtener la información de un incidente de un proyecto al cual pertenece.
* Parámetros { idUsuario, idIncidente}
* Responses {200, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }
* EndPoint: ~/api/Asociaciones
* Verbos HTTP: POST
* Descripción : Como administrador agregar desarrolladores y testers previamente registrados a un proyecto existente.
* Parámetros { [ lista id usuarios ], ProyectoId}
* Responses {201, 401, 422, 500 }
* Headers: { **Key**: autorización./ **Value**: token }