



Parte del Plan de Trabajo de Servicio Comunitario

# Sistema ANAR

## Manual de Uso

Propuesta elaborada por:  
Arturo Yepez | 15-11551  
Estudiante Ing. de la Computación



# Introducción

## Propósito del Documento

Bienvenido al Manual de Usuario del API para el Sistema ANAR, una guía detallada que te proporcionará la información necesaria para aprovechar al máximo las funcionalidades de nuestra potente interfaz de programación de aplicaciones (API).

El API del Sistema ANAR es una herramienta diseñada para facilitar la integración de tus aplicaciones con nuestro sistema, permitiéndote acceder y utilizar diversas funcionalidades de manera eficiente y segura. Ya sea que estés desarrollando una aplicación web, móvil o cualquier otro tipo de software, nuestro API te ofrece un conjunto de servicios robustos que te ayudarán a mejorar la experiencia del usuario y optimizar el rendimiento de tu aplicación.

Este manual está dirigido a desarrolladores, ingenieros y cualquier persona interesada en integrar el API en sus proyectos.

En las siguientes secciones, exploraremos los conceptos fundamentales de la API, los requisitos de autenticación, los puntos finales disponibles, ejemplos de solicitudes y respuestas, así como algunas recomendaciones de buenas prácticas para garantizar un desarrollo eficiente y seguro.

## Autenticación mediante *Bearer Token*

### Descripción

Una parte esencial de la integración con el API es la autenticación mediante el uso de tokens Bearer. Este mecanismo proporciona un nivel adicional de seguridad al permitir que solo las solicitudes autenticadas y autorizadas accedan a los recursos protegidos. Para obtener un token válido, sigue los pasos detallados a continuación.

### Pasos para Autenticar

1. Crear una cuenta y obtener credenciales de API

Antes de comenzar, asegúrate de tener una cuenta en nuestro sistema y las credenciales de API necesarias. Para eso, se puede utilizar el endpoint de registro en el sistema.



Esta solicitud devuelve el token de autenticación, pero para casos de usos posteriores se debe seguir los puntos 2 y 3.

## 2. Realizar una solicitud para obtener el token

Utiliza el endpoint de autorización del API del Sistema ANAR para solicitar un token de acceso. Realiza una solicitud HTTP POST al endpoint de login. Esta solicitud debe incluir el nombre de usuario (*username*), contraseña (*password*) y de forma opcional la aplicación con la que se está haciendo inicio de sesión.

## 3. Manejar la respuesta y utilizar el token

Si la solicitud es exitosa, recibirás una respuesta JSON que incluye el token de acceso. El token estará bajo la clave *token*. A continuación, puedes utilizar este token en los encabezados de autorización de las solicitudes subsiguientes para acceder a los recursos protegidos.

El token de acceso tiene un tiempo de vida limitado, establecida para que sea 24 horas. Asegúrate de manejar adecuadamente la renovación del token cuando sea necesario.

Con esta información, ahora estás listo para autenticar tus solicitudes al API XYZ utilizando el token Bearer generado. ¡Integra este flujo de autenticación en tu aplicación y comienza a aprovechar las funcionalidades ofrecidas por nuestro API!

## Inicialización del Sistema ANAR

El API del Sistema ANAR está diseñado para ser lo más dinámico posible y poder customizarse con cualquier tipo de aplicación y cualquier tipo de nuevo módulo a agregarse (para más información, hacer referencia al Manual de Desarrollo). Sin embargo, por motivos de simplicidad se decidió agregar un script para automatizar la inicialización en su primera vez del Sistema ANAR.

El script consiste en la inserción de algunos valores primitivos a la base de datos que consisten en el producto mínimo viable de este Sistema, como lo son: las aplicaciones definidas al inicio del desarrollo (Sistema ANAR y Memoria Rupestre), y el algoritmo inicial de Hashing para contraseñas que usa el módulo de autenticación (para más información, hacer referencia a la Propuesta de Módulo de Autenticación y Manual de Desarrollo).

Al ser un script dentro del ambiente de Node.JS, su ejecución se realiza mediante un comando clásico que se puede encontrar en las especificaciones del *package.json*



## Módulo de Autenticación

### POST - Register: /auth/register

#### Descripción

Registra al usuario dentro del sistema ANAR, con ninguna aplicación asociada a este.

#### Parámetros

#### Cuerpo

- *first\_name*
- *last\_name*
- *username*
- *email*
- *password*

#### Respuesta

- *token*: para uso como método de autenticación.

### POST - Login: /auth/login

#### Descripción

Iniciar sesión del usuario dentro del sistema, indicando la aplicación a la que se quiere asociar

#### Parámetros

#### Cuerpo

- *email*
- *password*
- *application*: (opcional) de las posibles listas de aplicaciones que reconoce el Sistema ANAR (ANAR, Memoria Web). En caso de existir, carga los roles específicos de la aplicación al contexto del usuario

#### Respuesta

- *token*: para uso como método de autenticación.



## POST - Confirm Account: /auth/confirm-email

### Descripción

Al crearse una cuenta de usuario dentro del Sistema ANAR, esta debe confirmarse dentro del sistema para que su respectivo estatuto sea actualizado y permite hacer acciones en el sistema que requiera a un usuario confirmado.

Este endpoint es de doble uso, se hace una primera solicitud al endpoint para que devuelva el token de validación del usuario, luego se hace otra solicitud al endpoint con el token como parámetro para confirmar la validación al usuario.

NOTA: Debe desarrollarse un sistema de envío de correos para establecer el flujo completo de este requerimiento.

### Parámetros

- *token* (opcional)

### Cuerpo

- *email* (opcional)

### Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## POST - Change Password: auth/change-password

### Descripción

Permite al usuario poder cambiar su contraseña.

### Parámetros

### Cuerpo

- *email*
- *oldPassword*
- *newPassword*
- *newPasswordRepeated*

### Respuesta



- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## Módulo de Aplicaciones

### GET - Get Applications: /applications

#### Descripción

Permite obtener las distintas aplicaciones que pueden existir dentro del Sistema ANAR

#### Parámetros

- *:name* (opcional)

#### Cuerpo

#### Respuesta

- Un arreglo de objetos/instancias de aplicaciones o un objeto/instancia de aplicación (en caso de que exista el parámetro opcional en la solicitud).

### POST - Create Applications: /applications

#### Descripción

Crea una nueva aplicación para el sistema.

#### Parámetros

#### Cuerpo

- *name*
- *description*: es la descripción de la funcionalidad de la aplicación.

#### Respuesta

- La instancia de la aplicación recién creada.

### PUT - Update Application

#### Descripción



Actualiza la información relacionada a una aplicación previamente creada.

#### Parámetros

- *:name* el identificador de la aplicación

#### Cuerpo

- *name*
- *description*: es la descripción de la funcionalidad de la aplicación.

#### Respuesta

- *application*: la instancia actualizada de la aplicación

### DELETE - Delete Application: /applications/:name

#### Descripción

Elimina un registro de permiso.

#### Parámetros

- *:name*

#### Cuerpo

#### Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## Módulo de Permisos

### GET - Get Permissions: /applications/permissions

#### Descripción

Permite obtener los distintos tipos de permisos que pueden existir entre distintas aplicaciones del sistema

#### Parámetros



- *:name* (opcional)

Cuerpo

Respuesta

- Un arreglo de objetos/instancias de permisos o un objeto/instancia de permiso(en caso de que exista el parámetro opcional en la solicitud).

## POST - Create Permissions: */applications/permissions*

Descripción

Crea un nuevo tipo de permiso que puede ser utilizado entre los distintos roles de las aplicaciones en el sistema

Parámetros

Cuerpo

- *name*
- *description*: es la descripción de la funcionalidad del permiso dentro del sistema, puede o no ser uno asociado a una funcionalidad específica de una aplicación.

Respuesta

- La instancia del permiso recién creado.

## PUT - Update Permissions: */applications/permissions/:name*

Descripción

Actualiza la información relacionada a un permiso previamente creado.

Parámetros

- *:name* el identificador del permiso

Cuerpo

- *name*



- *time*: el tiempo máximo que todos los niveles asociados a esa dificultad van a tener para resolver el nivel.

Respuesta

- *permission*: la instancia actualizada del permiso

## DELETE - Delete Permissions: /applications/:name

Descripción

Elimina un registro de permiso.

Parámetros

- *:name*

Cuerpo

Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## Módulo de Roles

### GET - Get Roles: /applications/roles

Descripción

Permite obtener los distintos tipos de roles y sus permisos asociados que pueden existir entre distintas aplicaciones del sistema

Parámetros

- *application* (opcional) identificador de la aplicación sobre la que se quieren obtener los roles

Cuerpo

Respuesta

- Un arreglo de objetos/instancias de roles o un objeto/instancia de rol (en caso de que exista el parámetro opcional en la solicitud), con todos los permisos asociados a ese rol.



## POST - Create Role: /applications/roles

### Descripción

Crea un nuevo tipo de rol para una aplicación en específico, y le asocia una serie de permisos existentes dentro del sistema.

### Parámetros

- *application* identificador de la aplicación sobre la que se quieren obtener los roles

### Cuerpo

- *name*
- *permissions*: son los identificadores de los permisos creados en el sistema que se quieren asociar al rol por crear

### Respuesta

- La instancia del rol recién creado.

## PUT - Update Roles: /applications/roles/:name

### Descripción

Actualiza la información relacionada a un rol previamente creado para una aplicación.

### Parámetros

- *application* el identificador de la aplicación a la que se le está actualizando el rol

### Cuerpo

- *name*
- *permissions*: son los identificadores de los permisos creados en el sistema que se quieren asociar al rol por actualizar

### Respuesta

- *role*: la instancia actualizada del role

## DELETE - Delete Role

### Descripción



Elimina un registro de un rol y su conexión con distintos permisos del sistema.

#### Parámetros

- *application* el identificador de la aplicación a la que se le está actualizando el rol.
- *:role* identificador del rol que se quiere eliminar

Cuerpo

Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## Módulo de Juego de Memoria Rupestre

### DIFFICULTY

GET - Get Difficulties: /difficulty/:name

#### Descripción

Permite obtener las dificultades que existen en todo el módulo de juego.

#### Parámetros

- *:name* (opcional)

Cuerpo

Respuesta

- Un arreglo de objetos/instancias de dificultades o un objeto/instancia de dificultad (en caso de que exista el parámetro opcional en la solicitud).

POST - Create Difficulty: /difficulty

#### Descripción



Crea un nuevo tipo de dificultad que va a ser utilizado en los niveles creados.

#### Parámetros

#### Cuerpo

- *name*
- *time*: el tiempo máximo que todos los niveles asociados a esa dificultad van a tener para resolver el nivel.

#### Respuesta

- La instancia de la dificultad recién creada.

## PUT - Update Difficulty: /difficulty/:name

#### Descripción

Actualiza la información relacionada a una dificultad previamente creada.

#### Parámetros

- *:name* el identificador de la dificultad

#### Cuerpo

- *name*
- *time*: el tiempo máximo que todos los niveles asociados a esa dificultad van a tener para resolver el nivel.

#### Respuesta

- *dificultad*: la instancia actualizada de la dificultad

## DELETE - Delete Difficulty: /difficulty/:name

#### Descripción

Elimina un registro de dificultad.

#### Parámetros

- *:name* el identificador de la dificultad



Cuerpo

Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## LEVEL

GET - Get User Info: /level/:levelId/user/:userId

Descripción

Se obtiene la información del usuario con respecto a un nivel en específico.

Parámetros

- *:levelId*
- *:userId*

Cuerpo

Respuesta

- La instancia relacional con la información del usuario con respecto a un nivel, incluyendo el estado de completitud de un nivel y la puntuación de este, en caso de existir.

POST - Complete Level: /level/:levelId/user/:userId

Descripción

Registra en el módulo de juego que existe, que el nivel indicado fue completado

Parámetros

- *:levelId*
- *:userId*

Cuerpo

Respuesta



- Crea o actualiza una entrada con el puntaje ganado y el estatus del nivel correspondiente.

## GET - Obtain scores of a level: */level/:levelId/scores*

### Descripción

Obtiene todos los puntajes habidos en un nivel.

### Parámetros

- *:levelId*

### Cuerpo

### Respuesta

- Un arreglo con los puntajes y sus respectivos usuarios con respecto a un nivel.

## GET - Obtain all scores: */user-scores*

### Descripción

Obtiene todos los puntajes habidos en todos los niveles.

### Parámetros

### Cuerpo

### Respuesta

- Un arreglo con todas las puntuaciones y sus respectivos usuarios con respecto a todos los niveles.

## GET - Get Levels: */level/:levelId*

### Descripción

Obtiene la información de todos los niveles o de un nivel en específico.

### Parámetros

- *:levelId* (opcional)

### Cuerpo



## Respuesta

- Un arreglo con objetos/instancias con la información de todos los niveles o un objeto/instancia con la información de ese nivel.

## POST - Create Level: /level

### Descripción

Crea en el juego de un nivel.

### Parámetros

#### Cuerpo

- *name*
- *description*
- *imageName*
- *goes\_after* (opcional): define cual es el nivel anterior.
- *goes\_before* (opcional): define cual es el nivel posterior.
- *difficulty\_id*: la dificultad asociada al nivel.

### Respuesta

- La instancia del nivel recién creado.

## PUT - Update Level: /level/:levelId

### Descripción

Actualiza la información relacionada a un nivel.

### Parámetros

- *:levelId* el identificador del nivel

#### Cuerpo

- *name*: el nombre del nivel que se quiere actualizar
- *difficulty\_id*: el identificador de la dificultad que se quiere asociar al nivel, este identificador debe ser un tipo de dificultad previamente creada

### Respuesta

- *level*: la instancia actualizada del nivel



## DELETE - Delete Level: /level/:levelId

### Descripción

Elimina un nivel del sistema.

### Parámetros

- *:levelId* el identificador del nivel

### Cuerpo

### Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## CARD

## GET - Get Cards: /card

### Descripción

Obtiene la información de todos las cartas o de una carta en específico.

### Parámetros

- *:cardId* (opcional)

### Cuerpo

### Respuesta

- Un arreglo con objetos/instancias con la información de todas las cartas o un objeto/instancia con la información de esa carta.

## POST - Create a Card: /card

### Descripción

Crea una carta que va a ser usada en un nivel.

### Parámetros

### Cuerpo



- *name*
- *description*
- *image*
- *levelId*: identificador asociado a un nivel previamente creado.

#### Respuesta

- La instancia de la carta recién creada.

### PUT - Update Card: /card/:cardId

#### Descripción

Actualiza la información relacionada a una carta previamente creada.

#### Parámetros

- *:cardId* el identificador del nivel

#### Cuerpo

- *name*
- *description*
- *image*
- *levelId*: identificador asociado a un nivel previamente creado.

#### Respuesta

- *card*: la instancia actualizada de la carta

### DELETE - Delete Card: /card/:cardId

#### Descripción

Elimina un registro de carta del sistema.

#### Parámetros

- *:cardId* el identificador del nivel

#### Cuerpo

#### Respuesta



- *message*: información respecto al estado de la solicitud, si fue exitosa o no.

## KNOWLEDGE

### GET - Get Knowledge: /knowledge

#### Descripción

Obtiene la información de todos las trivias o de una trivia en específico.

#### Parámetros

- *:id* (opcional)

#### Cuerpo

#### Respuesta

- Un arreglo con objetos/instancias con la información de todas las trivias o un objeto/instancia con la información de esa trivia.

### POST - Create Knowledge: /knowledge

#### Descripción

Crea una carta que va a ser usada en un nivel.

#### Parámetros

#### Cuerpo

- *question*: the main header or question that wants to be asked to the user
- *answer*: the knowledge that want to be shared
- *levelId*: identificador asociado a un nivel previamente creado.

#### Respuesta

- La instancia de la triviarrecién creada.



## PUT - Update Knowledge: /knowledge/:id

### Descripción

Actualiza la información relacionada a una trivia previamente creada.

### Parámetros

- *:id* el identificador de la trivia

### Cuerpo

- *question*: the main header or question that wants to be asked to the user
- *answer*: the knowledge that want to be shared
- *levelId*: el identificador del nivel donde se encuentra la trivia

### Respuesta

- *knowledge*: la instancia actualizada de la trivia

## DELETE - Delete Knowledge: /knowledge/:knowledgeId

### Descripción

Elimina un registro de conocimiento.

### Parámetros

- *:knowledgeId* el identificador de la trivia

### Cuerpo

### Respuesta

- *message*: información respecto al estado de la solicitud, si fue exitosa o no.