Trazalog – Arbolado

Glosario de servicios

# login

## Uso

Validar las credenciales informadas por un usuario a fin de establecer si se permite o no el ingreso al sistema.

## Métodos soportados

POST

## Mock

Se ha confeccionado un mock que refleja la respuesta ante un caso exitoso (login success)

<https://demo3989873.mockable.io/user/login>

## Consideraciones y definiciones técnicas

En el request enviamos 2 parámetros: username y password

En caso que las credenciales sean válidas se retorna un token y datos adicionales de acuerdo a lo discutido.

En caso que las credenciales no sean válidas, se envía una estructura distinta.

Desde el front-end evaluaremos el status HTTP de la respuesta: 200, es login ok, >200 es login erróneo y se pinta un mensaje genérico en pantalla. La estructura de respuesta para login erróneo será distinta a la de login OK y esto está pendiente de definición por parte de Trazalog.

# miPerfil

## Uso

Obtención y modificación de datos de perfil para un usuario determinado

## Métodos soportados

GET y POST

## Mock de respuestas

GET: <https://demo3989873.mockable.io/user/profile/get>

POST: <http://demo3989873.mockable.io/user/profile/update>

## Consideraciones y definiciones técnicas

La imagen de perfil será enviada y recibida como base64. Los parámetros a enviar para el GET son:

* session\_id
* username

Para el POST, los parámetros a enviar son:

* session\_id
* username
* full\_name
* password (SI ES EMPTY OR NULL NO CAMBIAR PASSWORD EN LA DB, SINO SI CAMBIAR)
* email
* phone
* picture (base64 string)

# userTrees

## Uso

Obtener la lista de áreas asignadas / censadas por usuario

Obtener la lista de manzanas asignadas / censadas por usuario

Obtener la lista de calles de una manzana

Obtener la lista de árboles censados

Obtener / guardar la información de un árbol censado

Guardar la información de un nuevo árbol censado

## Métodos soportados

GET / POST

## Mock

<https://demo3989873.mockable.io/user/tree/get> (SOLO GET)

## Consideraciones y definiciones técnicas

Todos los usos están cubiertos por el mismo servicio

Los atributos de geo-localización de un árbol son parámetros propios de cada árbol.

Los datos de domicilio a censar / o censado son atributos propios de cada árbol.

***El mock actual no tiene incluido los atributos de domicilio a censar. Los mismos deben ser los atributos que se visualizan en la maqueta de pantallas (domicilioCalle, domicilioCalleOtra, domicilioNumero, domicilioBarrio y domicilioTaza).***

# TreeForm

## Uso

Obtener la lista de elementos a presentar en pantalla para el formulario de censo.

## Métodos soportados

GET

## Mock

No disponible

## Consideraciones y definiciones técnicas

Este servicio devolverá la lista de elementos que deben ser pintados en patalla como formulario de censo. Se toman como base los elementos que se muestran en la maqueta correspondientes a las páginas formulario, formulario 1, formulario 2, formulario 3, formulario 4, …. formulario 7.

El servicio deberá devolver un JSON con múltiples JSONs adentro (uno por cada campo), cada JSON correspondiente a campo, contendrá pares de atributo:valor, siendo el valor (en algunos casos) un arreglo.

Para cada tipo de campo se listan los atributos que debe devolver el servicio

* Atributos generales **para todos los tipos de elemento**:
  + **ID**: Id a asignar en el elemento del dom HTML.
  + **Type**: Text, Number, Select, radio, checkbox, textarea
  + **display**: (block / none) Establece si por defecto se muestra ese componente o permanece oculto hasta que se deba mostrar de acuerdo a las condiciones de la pantalla.
  + **displayCondition**: (ej: nombre=’pepe’) establece bajo qué condición se debe mostrar este campo. En el ejemplo este elemento se mostrará si el campo con id “nombre” toma valor “pepe”.
  + **disabled** (disabled / vacío): establece si el elemento se pinta habilitado o no en el DOM (inicialmente). Si el atributo viene con valor disabled, elelemento se muestra en pantalla pero desabilitado, si no tiene valor asignado (disabled:’’) el elemento se muestra habilitado.
  + **enabledCondition:** (ej: nombre=’pepe’) establece bajo qué condición se debe habilitar este campo. En el ejemplo este elemento se habilitará si el campo con id “nombre” toma valor “pepe”.
  + **Required**: establece si el elemento es obligatorio (debe ser completado por el usuario o no)
  + **Order**: Orden en el que el elemento se debe pintar en el HTML DOM
  + **Label**: Etiqueta que se mostrará asociada al campo de formulario.
* Atributos específicos **para Radiobuttons o Checkbox**
  + **Value**: valor asignable (para los radio y checkbox, este es el valor que se retorna cuando el usuario hace click.
* Atributos específicos para listas de selección
  + **options**: debe ser un arreglo de strings, cada elemento del arreglo corresponde con las opciones que se quieran mostrar en la lista de selección.
  + **multiple** (true/false): Establece si la lista de selección permite elegir más de una opción.
* Atributos específicos para elementos textArea
  + **rows**: cantidad de filas de texto habilitadas
  + **cols**: cantidad de columnas de texto habilitadas