**Proyecto API-CHAT**

**DIGID - CIDESO**

Mayo 2022

**ÍNDICE:**

1. [Características generales del proyecto API-CHAT](#Vision_general)
2. [Definición de APIs](#Definición_APIs)
   1. [Solicitud de conversaciones posibles para un usuario logueado](#Conversacion_usuario)
   2. [Solicitud de los mensajes de una conversación dada](#Mensaje_conversacion)
   3. [Solicitud de creación de nuevo mensaje de tipo TEXTO](#Creacion_mje_texto)
   4. [Solicitud de creación de nuevo mensaje de tipo FILE](#Creacion_mje_file)
   5. [Solicitud de creación de nuevo mensaje de tipo POSITION](#Creacion_mje_position)
3. ***Características generales del proyecto API-CHAT:***

* 1. Para el desarrollo se utilizó **PHP** + **MYSQL** + **Laravel** (framework PHP)
  2. Implementación de ***HTTPS*** *-* Hypertext Transfer Protocol Secure. Protocolo que permite establecer una conexión segura entre dispositivos de usuarios y los sitios web.
  3. *Seguridad de acceso a las APIs a través de rutas protegidas … (Analizar qué se utilizará)*
  4. ***Buenas prácticas*** para garantizar que el desarrollo de la aplicación minimice los riesgos de seguridad a través del desarrollo de código seguro.
  5. Implementación del estándar ***CORS*** (Cross-origin resource sharing)

1. **Definición de APIs**
2. *Solicitud de conversaciones posibles para un usuario logueado:*
3. El *endpoint* devolverá un JSON con la información de las conversaciones activas más la información de los usuarios con los que todavía no se han iniciado conversaciones.
4. Petición:
   * + **https://{{HOST}}/api/{user\_id}/conversations/**
     + Método: GET

(Debe enviarse el *token* dentro de los headers de la petición.

*Authorization*: Bearer ***token)***

1. ENTRADA: parámetros por URL
   * user\_id
2. SALIDA:
   * Para una petición correctamente ejecutada, se obtendrá el código de estado “**200 OK”** y la estructura de datos que devolverá el JSON es la siguiente:

{

"user\_origin": 1,

"conversations": [

{

"conversation\_id": 5,

"conversation\_name": "Grupo CIDESO",

"conversation\_members": [

{

"user\_id": 3,

"last\_read\_at": "2022-03-22 16:11:35"

},

{

"user\_id": 2,

"last\_read\_at": "2022-03-22 16:11:35"

},

{

"user\_id": 4,

"last\_read\_at": "2022-03-22 16:11:35"

},

{

"user\_id": 1,

"last\_read\_at": "2022-05-14 01:09:16"

}

],

"ammount\_no\_read": 0

},

{

"conversation\_id": 1,

"conversation\_name": **null**,

"conversation\_members": [

{

"user\_id": 2,

"name": "Javier"

}

],

"ammount\_no\_read": 0

}

]

}

* + - Donde:
* "*user\_origin*": es el ID del usuario logueado
* "*conversations*":
* "conversation\_id": es el ID de la conversación (si es distinto de ***0*** es porque la conversación ya existe, de lo contrario quiere decir que todavía no existe una conversación iniciada con el usuario/grupo especificado en el campo "contact\_dest").
* "contact\_dest": información del usuario / grupo destinatario de la conversación.
* "ammount\_no\_read": Cantidad de mensajes NO leídos por el usuario logueado, en dicha conversación.
  + - Las Conversaciones se devuelven ordenadas de la siguiente manera:
      * Primero los usuarios o grupos con los que ya existe una conversación y se muestran, desde los que tienen mensajes más recientes a los que tienen mensajes menos recientes.
      * El resto de los contactos individuales con quienes no se ha iniciado una conversación y contactos grupales que no contengan mensajes aún.

1. *Solicitud de los mensajes de una conversación dada:*
2. El *endpoint* devolverá un JSON con la información de los mensajes de una determinada conversación, con paginación de a 10.

El ***ID*** de la conversación que se debe pasar por URL, fue obtenido previamente con el endpoint de Solicitud de conversaciones para el usuario logueado.

1. Petición:

* **https://{{HOST}}/api/{user\_id}/conversations/{conversation\_id}**
* Método: GET

(Debe enviarse el *token* dentro de los headers de la petición.

*Authorization*: Bearer ***token)***

1. ENTRADA: parámetros por URL
   * user\_id
   * ***ID*** de la conversación. El ID de la conversación es ‘0’, si todavía no se inició la conversación (No hay mensajes dentro de esa conversación)
2. SALIDA:

* Para una petición correctamente ejecutada, se obtendrá el código de estado “**200 OK”** y la estructura de datos que devolverá el JSON es la siguiente:

{

"user\_origin": 1,

"messages": {

"current\_page": 1,

"data": [

{

"conversation\_id": 5,

"sender\_id": 1,

"message\_type": "App\\Models\\PositionMessage",

"message\_id": 2,

"created\_at": "2022-05-14T00:26:02.000000Z",

"display\_type": "PositionMessage",

"message": {

"id": 2,

"lat": "-37.588967",

"lon": "-38.588967",

"alt": "15.198195"

}

},

{

"conversation\_id": 5,

"sender\_id": 1,

"message\_type": "App\\Models\\TextMessage",

"message\_id": 2,

"created\_at": "2022-04-22T16:11:35.000000Z",

"display\_type": "TextMessage",

"message": {

"id": 2,

"text": "Hola GRUPO!!"

}

}

],

"first\_page\_url": "http://127.0.0.1:8000/api/1/conversations/5?page=1",

"from": 1,

"last\_page": 1,

"last\_page\_url": "http://127.0.0.1:8000/api/1/conversations/5?page=1",

"next\_page\_url": **null**,

"path": "http://127.0.0.1:8000/api/1/conversations/5",

"per\_page": 10,

"prev\_page\_url": **null**,

"to": 2,

"total": 2

}

}

1. *Solicitud de creación de nuevo mensaje de tipo TEXTO:*
2. El *endpoint* creará un nuevo mensaje de TEXTO para una conversación dada o para iniciar una nueva conversación.
3. Devolverá un JSON con la información del mensaje creado y el estado de respuesta a la petición.
4. Petición:
   * [**https://{{HOST}}/api/v1/messages**](https://emersis.casya.com.ar/api/v1/messages)**/textMessage**
   * Método: POST

(Debe enviarse el *token* dentro de los headers de la petición.

*Authorization*: Bearer ***token)***

1. ENTRADA:

* **Form-data:** 
  + - 'user\_id' => ['required','integer', 'exists:users,id'],
    - 'conversation \_id' => ['required','integer'],
    - 'message' => ['required','string', 'max:255'],

1. SALIDA:
   * Si el ***conversation\_type*** = INDIVIDUAL y ***dest\_id*** es el mismo id que el del usuario logueado, se obtendrá el código de estado “**500 - No se puede enviar mensaje al usuario destino”**.
   * Si el ***conversation\_type*** = INDIVIDUAL y ***dest\_id*** NO pertenece a un usuario existente, se obtendrá el código de estado “**500 - The given data was invalid”.**
   * Si el ***conversation\_type*** = GROUP y el usuario logueado NO pertenece a ese grupo, se obtendrá el código de estado **“500 - No se puede enviar mensaje al grupo destino”.**
   * Si el ***conversation\_type*** = GROUP y ***dest\_id*** NO pertenece a un grupo existente, se obtendrá el código de estado “**500 - The given data was invalid”.**
   * Para una petición correctamente ejecutada, se obtendrá el código de estado “**200 OK”** y la estructura de datos que devolverá el JSON es la siguiente:

{

"status": 200,

"message": "Creación del mensaje de TEXTO realizada con éxito",

"conversation\_id": 8,

"sender\_id": 6,

"message\_created": "Hola Gabriel! Soy Brian"

}

1. *Solicitud de creación de nuevo mensaje de tipo FILE:* **(FALTA!!)**
2. El *endpoint* creará un nuevo mensaje de tipo FILE para una conversación dada o para iniciar una nueva conversación.
3. Devolverá un JSON con la información del mensaje creado y el estado de respuesta a la petición.
4. Petición:
   * **https://{{HOST}}/api/v1/messages/fileMessage**
   * Método: POST

(Debe enviarse el *token* dentro de los headers de la petición.

*Authorization*: Bearer ***token)***

1. ENTRADA:

* **Form-data:** 
  + - 'user\_id' => ['required','integer', 'exists:users,id'],
    - 'contact\_type' => ['required', Rule::in(['INDIVIDUAL', 'GROUP'])],
    - 'contact\_id' => ['required','integer'],
    - 'file' => ['required', 'array'],
    - 'file.\*' => ['file','required', 'mimes:doc,pdf,docx,txt,zip,jpeg,png,bmp,xls,xlsx,mov,qt,mp4,mp3,m4a' ,'max:10240'],
    - 'description' => ['sometimes', 'array'],
    - 'description.\*' => ['nullable', 'string'],

**NOTA**: Los nombres de los archivos que se guardan en el servidor tienen el siguiente formato: nombreOriginal\_***timestamp****.*extensión

1. SALIDA:
   * Si el ***type*** = INDIVIDUAL y ***dest\_id*** NO es un contacto válido para el usuario logueado, se obtendrá el código de estado “**500 - El usuario a quien se quiere enviar el mensaje, NO es un contacto válido”.**
   * Si el ***type*** = GROUP y el usuario logueado NO pertenece a ese grupo, se obtendrá el código de estado “**500 - El usuario NO pertenece grupo al que se quiere enviar el mensaje.”**.
   * Los archivos se envían en un array, si algún archivo del array pasa las validaciones, se crea el mensaje con los archivos válidos, si alguno no lo fuera, se lo ignora.
   * Si todos los archicos del array NO pasan las validaciones, se obtendrá el código de estado “**422 – Unprocessable Entity”** y lo que devolverá el JSON es lo siguiente, según corresponda:

{

"errors": {

"file.0": [

"Los archivos sólo pueden ser doc,pdf,docx,txt,zip,jpeg,bmp,xls,xlsx,mov,qt,mp4,mp3,m4a",

"Cada archivo no puede ser mayor a 10MB"

]

}

}

* + Para una petición correctamente ejecutada, se obtendrá el código de estado “**200 OK”** y la estructura de datos que devolverá el JSON es la siguiente:

{

"status": 200,

"message": "Creación del mensaje de FILE realizada con éxito",

"conversation\_id": 1,

"message\_id": 6,

"sender\_id": 1,

"type": 1,

    "dest\_id": 4,

"message\_created": {

"id": 24,

**"files": [**

**{**

**"id": 3,**

**"file": "files/ildFxLBGjBzXrwWzWvYP53N0CTNGnAppd6HZPZy2.png",**

**"original\_file": "iconos\_gral.png",**

**"description": "iconos\_gral.png",**

**"attachable\_type": "App\\Models\\FileMessage",**

**"attachable\_id": 24,**

**"novedad\_id": null,**

**"created\_at": "2020-08-24T20:19:12.000000Z",**

**},**

**{**

**"id": 4,**

**"file": "files/7mNlTh2qzxlzWaYFpGo2eVAPPYLPVJzuFZz6sQ2M.jpeg",**

**"original\_file": "maquina\_3.jpeg",**

**"description": "maquina\_3.jpeg",**

**"attachable\_type": "App\\Models\\FileMessage",**

**"attachable\_id": 24,**

**"novedad\_id": null,**

**"created\_at": "2020-08-24T20:19:12.000000Z",**

**}**

**]**

}

}

1. *Solicitud de creación de nuevo mensaje de tipo POSITION:*
2. El *endpoint* creará un nuevo mensaje de *POSITION* para una conversación dada o para iniciar una nueva conversación.
3. Devolverá un JSON con la información del mensaje creado y el estado de respuesta a la petición.
4. Petición:
   * **https://{{HOST}}/api/v1/messages/positionMessage**
   * Método: POST

(Debe enviarse el *token* dentro de los headers de la petición.

*Authorization*: Bearer ***token)***

1. ENTRADA:

* **Form-data:** 
  + - 'user\_id' => ['required','integer', 'exists:users,id'],
    - 'contact\_type' => ['required', Rule::in(['INDIVIDUAL', 'GROUP'])],
    - 'contact\_id' => ['required','integer'],
    - 'lat' => ['required','numeric'],
    - 'lon' => ['required','numeric'],
    - 'alt' => ['required','numeric'],

1. SALIDA:
   * Si el ***type*** = INDIVIDUAL y ***dest\_id*** NO es un contacto válido para el usuario logueado, se obtendrá el código de estado “**500 - El usuario a quien se quiere enviar el mensaje, NO es un contacto válido”.**
   * Si el ***type*** = GROUP y el usuario logueado NO pertenece a ese grupo, se obtendrá el código de estado “**500 - El usuario NO pertenece grupo al que se quiere enviar el mensaje.”**.
   * Para una petición correctamente ejecutada, se obtendrá el código de estado “**200 OK”** y la estructura de datos que devolverá el JSON es la siguiente:

{

"status": 200,

"message": "Creación del mensaje de POSICIÓN realizada con éxito",

"conversation\_id": 1,

"sender\_id": 1,

"message\_created": {

"id": 1,

"lat": "-37.588967",

"lon": "-38.588967",

"alt": "15.198195"

}

}