humand

Manual Web Api

Introducción

Humand ofrece un conjunto de integraciones que a partir de ahora denominaremos "WEB API". Estas integraciones están disponibles sin costo alguno para todos nuestros clientes y tienen como objetivo principal facilitar el acceso en tiempo real y de forma sincrónica a su información en nuestra plataforma.

La finalidad principal de esta API es proporcionar una interfaz amigable y segura que pueda ser utilizada por cualquier programador. Además, está diseñada para simplificar la integración de nuestra solución en la nube con sus sistemas legados existentes, haciendo uso de una arquitectura ampliamente probada y difundida en la actualidad, como es el caso de API REST.

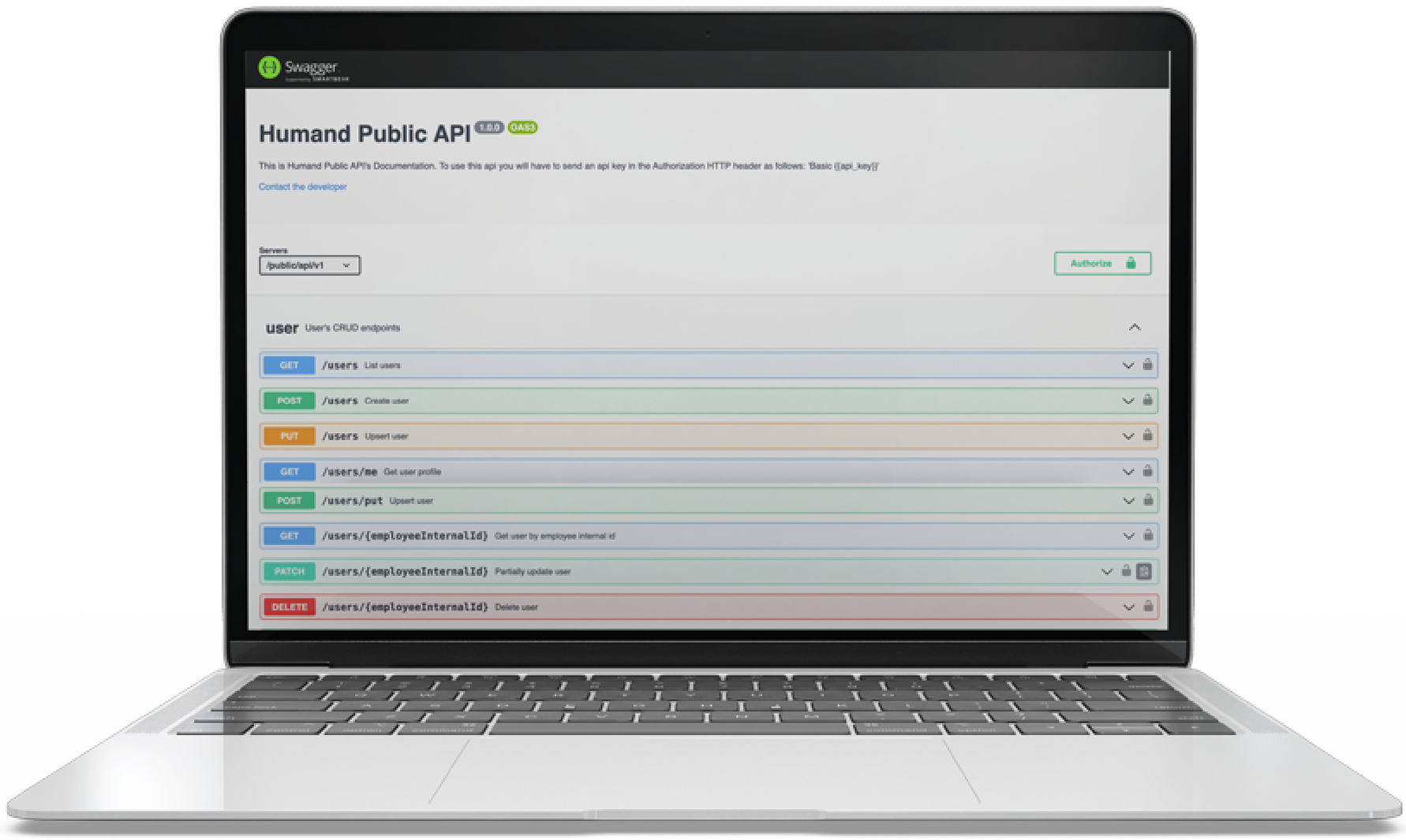
Para alcanzar esta meta, la WEB API proporciona una variedad de servicios estándar que incluyen la creación y obtención de empleados, la gestión de organigramas, la capacidad de realizar publicaciones, la carga de recibos de nómina, la segmentación de usuarios en grupos específicos, y mucho más.

Documentación detallada

Contamos con un sitio donde se puede consultar toda la información necesaria para crear una integración con nuestro sistema “WEB API” (ejemplos de invocación, respuesta y mucho más). Utilizamos una herramienta estándar en la industria para detallar toda esta información, como lo es Swagger.

La herramienta Swagger permite documentar y probar las APIs. Es así como no sólo se puede explorar la lista de servicios y su correspondiente información (cabecera de los mensajes, parámetros, estructura de las respuestas, etc.) sino que también se pueden probar los servicios a través del navegador mismo, sin necesidad de escribir código de programación.

Ver sitio



Generalidades del procedimiento de Integración

A. Actualmente, el acceso a la plataforma de pruebas se otorga mediante el uso de una API Key. Esta información específica de acceso a la plataforma de pruebas se proporcionará por parte de Humand una vez que se tome la decisión de llevar a cabo la integración.

Una vez que se hayan completado satisfactoriamente las pruebas en el entorno de pruebas y la integración esté lista para su implementación en el entorno de producción, se realizará la conexión a la comunidad de producción utilizando una API Key diferente que será suministrada por nosotros.

Este proceso garantiza una transición segura y controlada desde el ambiente de pruebas al ambiente de producción, manteniendo la seguridad y la integridad de los datos en todo momento.

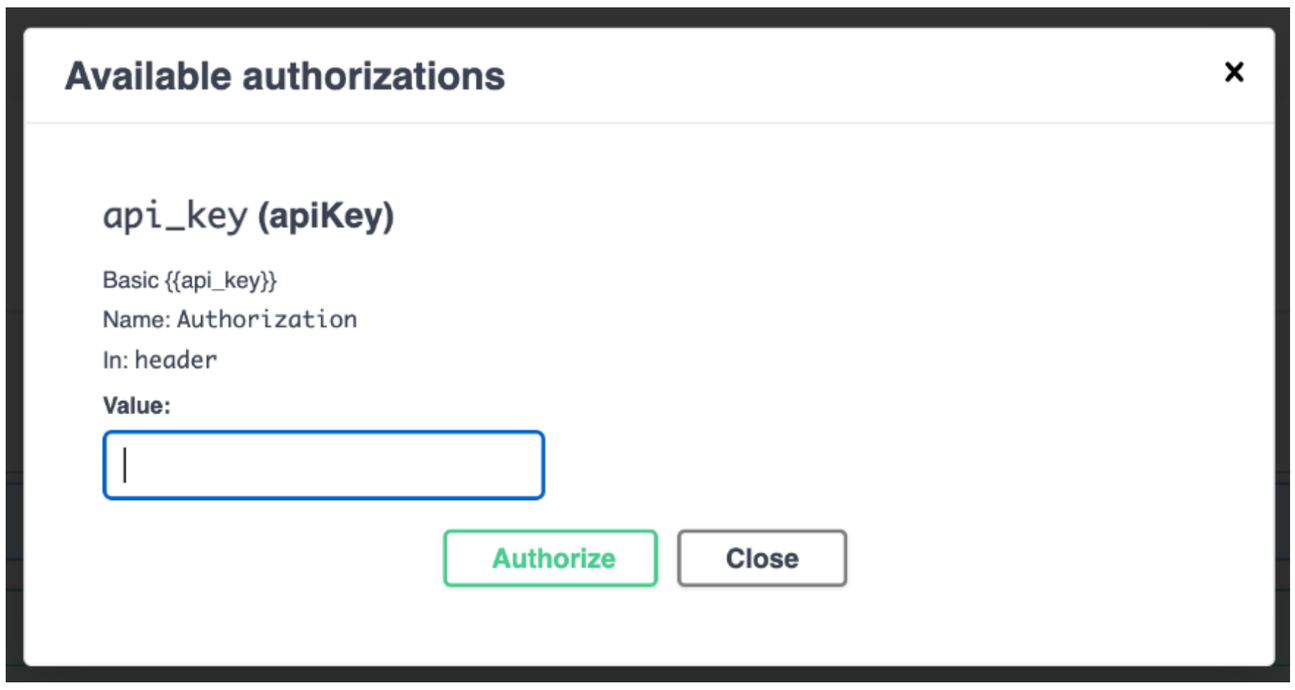
Actualmente, el acceso otorgado a través de una API Key es a la plataforma de testing (esta información se brindará desde Humand una vez que se decida por realizar la integración). Una vez finalizadas las pruebas, la conexión se realizará a la comunidad de producción con una API Key diferente que les proporcionaremos.

Importante: esta API Key debe estar asociada a un usuario que cuenta con permisos de administrador. Es importante que este Usuario NO sea eliminado para asegurar la integración de forma permanente.

B. El ingreso de la API Key para realizar los tests es siguiendo este formato :

Basic JyBpaEH7YMppAQjqyGMsV53PHk9-Pd_T

Siendo “JyBpaEH7YMppAQjqyGMsV53PHk9-Pd_T” la API Key asociada en este caso (el equipo de Humand proporcionará la correspondiente a cada comunidad). Notar que se debe introducir la palabra Basic seguido de un espacio y luego la API Key asociada.



Detalles del alcance de las integraciones que se pueden realizar a Humand:

Los usuarios en Humand cuentan con tres tipos de información pertinente que componen su perfil:

A. Campos básicos de información personal:
Esta es la información básica que un usuario tiene en su perfil en la aplicación. Los datos obligatorios son con los que debe contar sí o sí el usuario para ser creado exitosamente en la plataforma. Los campos optativos, pueden o no ser agregados dependiendo de la información con la que contemos en la base de datos.

Nombre de usuario

Obligatorio, string - "employeeInternalId": "string"
Aclaración: NO es el nombre de la persona, es el usuario por el cual entrará a Humand.

Nombre

Obligatorio, string - "firstName": "User first name”
No se distingue entre primer y segundo nombre, iría completo.

Apellido

Obligatorio, string - "lastName": "User last name"
No se distingue entre primer y segundo apellido, iría completo.

Contraseña inicial

Obligatorio, string - "password": "password"
Debe tener como mínimo 8 caracteres.

Jefe directo

Obligatorio, string
En el json debe completarse el “employeeInternalId” del usuario asignado como jefe.
El jefe directo se completa con la siguiente parte del json: **(Importante: Siempre hay que generar primero el usuario del jefe con su employeeInternalId correspondiente, y luego al momento de generar el usuario del colaborador subordinado colocar como employeeInternalId el useremployeeInternalId de su jefe. Ya que si se coloca como jefe un employeeinternalId que todavía no existe en la comunidad, surgirá un error).**

```
"relationships": [
   {
    "name": "BOSS",
    "employeeInternalId": "userEmployeeInternalId"
   }
 ]
```

Mail del usuario

Optativo, string
"email": "user@email.com",

Apodo del usuario

Optativo, string - "nickname": "userNickName",

Teléfono del usuario

(optativo, string) "phoneNumber": "+5491123456789",
- Es necesario poner el prefijo del país

Fecha de contratación del usuario

(optativo, string) "hiringDate": "2022-01-01",

Fecha de nacimiento del usuario

(optativo, string) "birthdate": "2022-01-01"

B. Segmentación asociada al usuario:
Estas propiedades nos permite agrupar a los usuarios de acuerdo a características en común que tengan, como por ejemplo pertenecer a una misma área o trabajar en una misma ubicación. Las segmentaciones están formadas por:

Grupo de Segmentación

"group": "Segmentation group name" - Cada grupo tiene un único título y un usuario puede pertenecer a varios grupos de segmentación.

Ítems de Segmentación

"item": "Segmentation item name" - Cada grupo tiene uno o más ítems de segmentación.
Cada usuario puede tener ningún, uno o múltiples ítems para un grupo de segmentación en particular.

Las segmentaciones no son de carácter obligatorio, depende de la información con la cual contamos en la base de datos, algunos ejemplos pueden ser:

- Gerencia
- Sector
- Ubicación

La sección del json correspondiente a la segmentación de un usuario al momento de su creación sería:

```
"segmentation": [
  {
    "group": "Gerencia",
    "item": "Segmentation item name"
  },
  {
    "group": "Sector",
    "item": "Segmentation item name"
  },
  {
    "group": "Ubicación",
    "item": "Segmentation item name"
  },
]
```

C. Campos dinámicos o extras del perfil de los usuarios:

Esta es la información adicional que cada comunidad puede elegir crear para que los usuarios tengan. Los datos son optativos y pueden o no ser agregados dependiendo de la información con la que contemos en la base de datos. **Importante: Los campos de perfil que se escriban en la API deberán coincidir al 100% (respetando las mayúsculas y minúsculas) del campo ingresado en la parte de Ajustes → Perfiles dentro del Panel Administrador de Humand.**

Identificador único del Valor del campo dinámico (se podrá adquirir mediante GET).
"ID": "7b122126-f8ec-4410-b163-8389ed21e79f"

Nombre del campo de perfil
"name": "Dynamic field name"

Es importante tener en cuenta que se puede enviar ID o Name, no es necesario mandar ambos campos. En el caso de enviar ambos campos es necesario enviarlo con el valor del mismo (no vacio).

Información a completar del campo de perfil
"value": "Dynamic field value"

Dentro del json siguen la siguiente estructura general:

```
"fields": [
    {
      "ID": "7b122126-f8ec-4410-b163-8389ed21e79f",
      "name": "Dynamic field name",
      "value": "Dynamic field value"
  }
```

Campos dinámicos de ejemplo pueden ser:

- Puesto
- Número de Interno
- Gerencia
- Edificio
- Legajo

Considerando esto, la sección completa de esta parte del json de creación de un usuario asociada a los campos dinámicos sería:

```
"fields": [
    {
      "name": "Puesto",
      "value": "Dynamic field value"
    },
    {
      "name": "Número de Interno",
      "value": "Dynamic field value"
    },
    {
      "name": "Gerencia",
      "value": "Dynamic field value"
    },
    {
      "name": "Edificio",
      "value": "Dynamic field value"
    },
    {
      "name": "Fecha de Inicio Puesto Actual",
      "value": "Dynamic field value"
    }
 ],
```

Diferencia principal entre la Segmentación asociada al usuario y los Campos dinámicos o extras del perfil de los usuarios:

La gran diferencia entre estos dos tipos de información es que la Segmentación asociada al usuario actúa como un filtro en Humand. Un ejemplo sería que podríamos filtrar un posteo según la ubicación de los colaboradores de la empresa, siendo la Ubicación una de esas Segmentaciones. En cambio, los Campos dinámicos de perfil son detalles que aparecen en los perfiles de los colaboradores, un ejemplo de esto sería su cédula de identidad. Es decir que es un campo informativo que no se utiliza para filtrar.

A tener en cuenta:

Antes de enviar las segmentaciones o los campos dinámicos de perfil a través de la API, te recomendamos ingresar a **admin.humand.co** e ingresar las credenciales de la comunidad de Testing asignadas.

Allí, encontrarás la sección llamada **"Segmentación"**. En este apartado, podrás cargar previamente las distintas segmentaciones y los ítems asociados a cada una. Estos son personalizables y se pueden ajustar según la base de datos que estés utilizando. Puedes realizar esta carga de manera masiva.

A continuación, proporcionaremos un tutorial que explica cómo hacerlo:

Tutorial

Aclaración: verás que en los videos se habla sobre la sección de “Grupos” dentro del panel administrador. Actualmente, el nombre de esa sección fue modificada por **“Segmentación”**.

Otra sección que encontrarás será la de Ajustes. Una vez allí, te pedimos que te dirijas a la tercera pestaña llamada Perfiles. Al pie de la página, encontrarás una sección para cargar los campos dinámicos de perfil donde podrás agregar tantos como sea necesario. Los mismos son customizables y se pueden ajustar según la base de datos que estés utilizando.

A continuación, proporcionaremos un tutorial que explica cómo hacerlo a partir del minuto 2.02:

Tutorial

Es esencial que la nomenclatura de las segmentaciones y campos dinámicos dentro del panel administrador de Humand, coincida con la que se envía a través de la API.

Usos de la API

Usuarios

POST - Create Users: este endpoint lo utilizaremos para la creación de nuevos usuarios. A continuación compartimos el ejemplo del json para el caso completo de un usuario con los ejemplos brindados (todas las segmentaciones y todos los campos dinámicos incluídos)

```
{
  "password": "password",
  "segmentation": [
     {
      "group": "Gerencia",
      "item": "Segmentation item name"
    },
    {
      "group": "Sector",
      "item": "Segmentation item name"
    },
    {
      "group": "Ubicación",
      "item": "Segmentation item name"
    },
  ],

"relationships": [
    {
      "name": "BOSS",
      "employeeInternalId": "userEmployeeInternalId"
    }
 ],

 "fields": [
    {
      "name": "Puesto",
      "value": "Dynamic field value"
    },
    {
      "name": "Número de Interno",
      "value": "Dynamic field value"
    },
    {
      "name": "Gerencia",
      "value": "Dynamic field value"
    },
    {
      "name": "Edificio",
      "value": "Dynamic field value"
    },
    {
      "name": "Fecha de Inicio Puesto Actual",
      "value": "Dynamic field value"
    }
],
"employeeInternalId": "string",
"email": "user@email.com",
"firstName": "User first name",
"lastName": "User last name",
"nickname": "userNickName",
"phoneNumber": "5491123456789",
"hiringDate": "2022-01-01",
  "birthdate": "2022-01-01"
}
```


PUT - Upsert Users: este endpoint lo utilizaremos para la actualización de usuarios existentes. El json que debe enviarse es el completo de la creación del usuario, incluyendo los campos modificados. El body del json en este caso es igual que en el del POST. **Importante: Para asegurarse de que no se borre información, es fundamental que el body del PUT contenga toda la información del usuario (incluso si esta no fue modificada)**

PATCH - Partially update user: este endpoint lo utilizaremos para la actualización parcial de usuarios existentes, desde el mismo no se podrán modificar los campos dinámicos de perfil. El json debe enviarse con la información que se quiere modificar solamente.

```
{
  "segmentation": [
    {
      "group": "Gerencia",
      "item": "Segmentation item name"
    },
    {
      "group": "Sector",
      "item": "Segmentation item name"
    },
    {
      "group": "Ubicación",
      "item": "Segmentation item name"
    },
   ],
  ],
  "employeeInternalId": "string",
  "email": "user@email.com",
  "firstName": "User first name",
  "lastName": "User last name",
  "nickname": "userNickName",
  "phoneNumber": "5491123456789",
  "hiringDate": "2022-01-01",
  "birthdate": "2022-01-01"
}
```

PATCH - Update user profile fields: este endpoint lo utilizaremos para la actualización parcial de campos dinámicos de perfil de usuarios existentes. Es importante que el json solo debe enviarse con la información que se quiere modificar.

```
 "fields": [
    {
      "name": "Puesto",
      "value": "Dynamic field value"
    },
    {
      "name": "Número de Interno",
      "value": "Dynamic field value"
    },
    {
      "name": "Gerencia",
      "value": "Dynamic field value"
    },
    {
      "name": "Edificio",
      "value": "Dynamic field value"
    },
    {
      "name": "Fecha de Inicio Puesto Actual",
      "value": "Dynamic field value"
    }
  ],
```

PATCH & POST - Partially update user segmentation: Estos Endpoints sirven para actualizar parcialmente la segmentación de un usuario. En el PATCH anterior correspondiente a la actualización del usuario, el bloque de segmentación se debe mandar completo para actualizar la información del usuario, es decir, se deben enviar todas las segmentaciones e ítems a las que pertenece el mismo. En el caso de querer actualizar parcialmente la segmentación de un usuario, se cuenta con 2 nuevos métodos de un nuevo endpoint.

POST - /users/:employeeInternalId/segmentations: Este endpoint lo utilizaremos para asignar al usuario los Ítems del segmento que se envían, sin necesidad de mandar el detalle completo de los otros Ítems o segmentos.
Importante: esto aplica para asignaciones nuevas, es decir, si ese usuario formaba parte de otro ítem del mismo segmento no lo "reemplaza" (en ese caso ver método siguiente).

Ejemplo: Supongamos que tenemos dos segmentos: 'Ubicación' y 'Tareas'. El segmento 'Ubicación' contiene los ítems 'Localidad A', 'Localidad B', y 'Localidad C'. El segmento 'Tareas' incluye los ítems 'Administrativo', 'Comercial' y 'Operativo'.
Ahora, consideremos un usuario X que forma parte del ítem 'Localidad A' en el segmento 'Ubicación' y del ítem 'Administrativo' en el segmento 'Tareas'. Si enviamos una solicitud a este endpoint para agregar el ítem 'Comercial' del segmento 'Tareas' al usuario X, este se añadirá a dicho ítem. El usuario X permanecerá también en 'Administrativo' del segmento 'Tareas' y en 'Localidad A' del segmento 'Ubicación'.

```
{
"segmentations": [
{
"group": "Segmentation group name",
"item": "Segmentation item name"
}
]
}
```

PATCH - /users/:employeeInternalId/segmentations: Este endpoint lo utilizaremos para reemplazar los items que se envian al usuario de una misma segmentación, sin modificar las segmentaciones que no se envían.

Ejemplo: Supongamos que tenemos dos segmentos: 'Ubicación' y 'Tareas'. El segmento 'Ubicación' contiene los ítems 'Localidad A', 'Localidad B' y 'Localidad C'. El segmento 'Tareas' incluye los ítems 'Administrativo', 'Comercial' y 'Operativo'.
Consideremos un usuario X que forma parte del ítem 'Localidad A' en el segmento 'Ubicación' y del ítem 'Administrativo' en el segmento 'Tareas'. Si enviamos una solicitud a este endpoint para agregar el ítem 'Comercial' del segmento 'Tareas' al usuario X, este se añadirá a dicho ítem y ya no formará parte del ítem 'Administrativo' del mismo segmento, aunque seguirá perteneciendo a 'Localidad A' del segmento 'Ubicación'.

```
{
"segmentations": [
{
"group": "Segmentation group name",
"item": "Segmentation item name"
}
]
}
```

DELETE - Delete Users: en este caso, el endpoint será utilizado para eliminar usuarios. Para ello lo único que debemos compartir es el nombre del usuario (employeeInternalId).

Segmentaciones

POST - Segmentations: Este caso será utilizado en caso de que se desee agregar una segmentación nueva. Para ello el esquema a seguir es:

```
{
  "name": "Segmentation group name",
  "itemNames": [
    "Segmentation item name"
  ],
  "visibility": "ALL"
}
```

En "itemNames" incluiremos todos los ítems asociados a esa nueva segmentación.

La visibilidad ("visibility") puede ser ALL (para todos los usuarios), ADMINS_ONLY (para los administradores nada más), USER_AND_ADMINS (para el propio usuario y administradores)

PUT - Segmentations: Si quisiéramos modificar las segmentaciones o ítems actuales, deberíamos utilizar el endpoint de PUT Segmentations con el json completo correspondiente (similar al anterior).

Documentos personales

POST - Document: Desde este endpoint se podrán subir documentos personales (recibos de nómina, contratos, entre otros) a los usuarios.

Algunos de los campos que podrán completar serán los siguientes:

file

Tipo: string (binary)
Descripción: En este campo se debe enviar el documento en formato binario, lo que significa que el archivo se adjunta en su formato crudo, sin procesamiento.

folderId

Tipo: integer
Descripción: El 'folderId' es un identificador numérico que corresponde a la carpeta donde se almacenará la información. Para encontrar este valor, se debe acceder a la plataforma desde la cuenta de administrador y navegar a la sección 'Documentos personales. El 'folderId' se encuentra en la URL de la carpeta en la que se desea almacenar la información. Este identificador se mantiene constante a lo largo del tiempo, es decir que siempre que un archivo específico vaya a una carpeta en particular el ID de esa carpeta se mantiene.Los colaboradores no podrán ver los archivos de otros usuarios.

En el siguiente ejemplo el folder id sería: 14326

name

Tipo: string
Descripción: Este campo se utiliza para especificar el nombre del archivo que se va a enviar, el mismo debe tener una nomenclatura única. Se recomienda que el nombre del archivo siga una nomenclatura que incluya el 'internalId' del colaborador.

sendNotification

Tipo: boolean
Descripción: Este campo se utiliza para indicar si se debe enviar una notificación cuando se envía el archivo.
Si el valor es 'true', se enviará una notificación. Si es 'false', no se enviará notificación.

signatureStatus

Tipo: string
Descripción: Este campo se utiliza para indicar si se requiere una firma en el archivo. Se debe especificar el estado de la firma, por ejemplo, si es necesaria o no. Una posible opción de respuesta es PENDING,

signatureCoordinates

Tipo: objeto
Descripción: En este campo, se solicita que expongamos las coordenadas en donde deberían estar las firmas.
Si es necesaria la firma pueden enviarle un template del recibo a su referente de Humand y pronto le comentaremos las coordenadas.

allowDisagreement

Tipo: boolean
Descripción: Este campo se utiliza para determinar si se permite firmar con disconformidad.

Un ejemplo de un Json a enviar podría ser el siguiente:


```
{
'folderId': '1',
'name': 'File name',
'sendNotification': 'false',
'signatureStatus': 'PENDING',
'signatureCoordinates': '[{"page":0,"x":0.090785285150645 61,"y":0.7916669970581939,
"height":0.052591723295774 42,"width":0.21792615674318508}]',
'allowDisagreement': 'false'
}
```


En el proceso de envío de archivos, es fundamental entender que no se deben enviar los datos en formato JSON, sino en un formato conocido como 'form data'. Para lograr esto, debemos realizar los siguientes pasos además de lo anteriormente mencionado:

Asegurarse de que en los encabezados (headers) de la solicitud se incluya la clave 'Content-Type' con el valor 'multipart/form-data'. Esto indica que estamos enviando datos en formato 'form data'.

Además de los campos y valores que se describen en el JSON, es necesario agregar el archivo que deseamos enviar. Esto se hace utilizando la clave 'file' para identificar el archivo adjunto.

Siguiendo estos pasos, estaremos listos para enviar archivos de manera adecuada a través de solicitudes HTTP utilizando 'form data', garantizando así una comunicación efectiva con el servidor.

Un ejemplo en Pyhton todo completo:

```
import requests

url = "https://api-prod.humand.co/public/api/v1/users/Employee_internal_id/documents/files"

payload = {'folderId': '1',
'name': 'File name',
'sendNotification': 'false',
'signatureStatus': 'PENDING',
'signatureCoordinates': '[{"page":0,"x":0.0907852851506456 1,"y":0.7916669970581939,"height":0.052591723295774 42,
"width":0.2179261567431850 8}]',
'allowDisagreement': 'false'}
file=[
  ('file',('document.pdf',open('/Users/username/Downloads/document.pdf','rb'),'application/pdf'))
]
headers = {
  'Content-Type': 'multipart/form-data',
  'Accept': 'application/json',
  'Authorization': 'API_KEY'
}

response = requests.request("POST", url, headers=headers, data=payload, files=files)

print(response.text)
```

Un ejemplo de Java de todo completo:

```
var myHeaders = new Headers();
myHeaders.append("Content-Type", "multipart/form-data");
myHeaders.append("Accept", "application/json");
myHeaders.append("Authorization", API_KEY);

var formdata = new FormData();
formdata.append("file", "voluptate adipisicing Duis veniam enim");
formdata.append("folderId", "1");
formdata.append("name", "File name");
formdata.append("sendNotification", "false");
formdata.append("signatureStatus", "PENDING");
formdata.append("signatureCoordinates", "[object Object]
```

Vacaciones

GET - Get time off requests: Desde este endpoint se podrán obtener las solicitudes de Vacaciones y permisos que se han realizado en Humand.

Algunos de los campos a completar son los siguientes:

Page

Tipo: integer
Descripción: El campo 'page' especifica el número de la página actual que se está solicitando. Es obligatorio completar esta información. Por ejemplo “page=2” indica que se está solicitando la segunda página de resultados.
A continuación dejamos una imágen a modo de ilustración:

registro 1	page 1
registro 2	limit 10
registro 3	
registro 4	
registro 5	
registro 6	
registro 7	
registro 8	
registro 9	
registro 10	
registro 11	page 2
registro 12	limit 10
registro 13	
registro 14	
registro 15	
registro 16	
registro 17	
registro 18	
registro 19	
registro 20	
registro 21	page 3
registro 22	limit 10
registro 23	
registro 24	
registro 25	
registro 26	
registro 27	
registro 28	
registro 29	
registro 30	

Limit

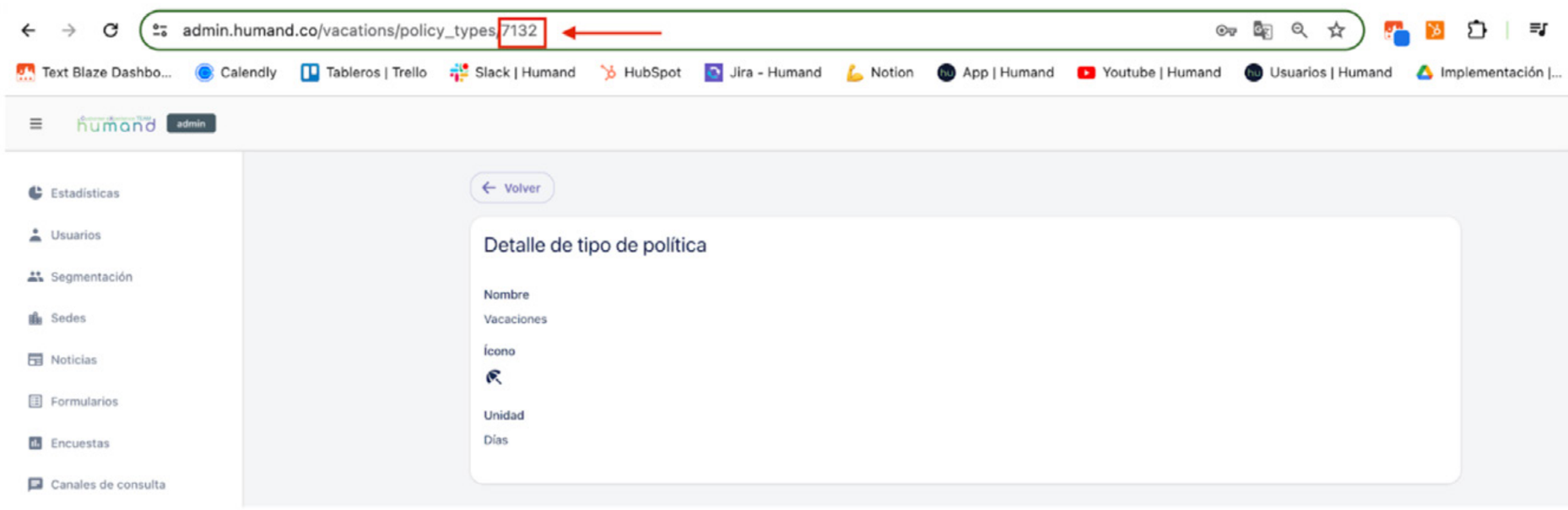
Tipo: integer
Descripción: Este campo define la cantidad máxima de elementos o registros que se desean obtener por página, el valor máximo permitido es 500, controlando así el tamaño de cada segmento de datos entregado. Es obligatorio completar esta información. Por ejemplo “limit=10” indica que se desean obtener hasta 10 registros por página.

States

Tipo: string
Descripción: Este campo da la posibilidad de especificar el estado de la solicitud a la hora de obtener la información. Las posibilidades son las que fueron aprobadas, las que están en progreso, las rechazadas y las canceladas. **Es opcional. APPROVED, REJECTED, IN_PROGRESS, CANCELLED.**

PolicyTypeIds

Tipo: string
Descripción: En este campo se podrá elegir sobre qué tipo política es la información sobre las solicitudes realizadas, Vacaciones, Día de estudio, Licencia por maternidad.
Para poder encontrar el ID de la política lo podrás hacer viendo los últimos números que figuran en la URL cuando se ingresa en el detalle de la misma:



from.date

Tipo: string
Descripción: En este campo podrás obtener las solicitudes cuyos días están contenidas a partir de la fecha seleccionada.

to.date

Tipo: string
Descripción: En este campo podrás obtener las solicitudes cuyos días están contenidas hasta la fecha seleccionada.

resolutionDateFrom

Tipo: string
Descripción: Fecha "desde" para la fecha que se aprobó, rechazó o canceló esa solicitud.

resolutionDateTo

Tipo: string
Descripción: Fecha "hasta" para la fecha que se aprobó, rechazó o canceló esa solicitud.

Un ejemplo de un Json podría ser el siguiente:


```
{
    "page": 1,
    "limit": 100,
    "states": "APPROVED",
    "PolicyTypeIds": "1,2,3",
    "fromDate": "2023-01-01",
    "toDate": "2023-12-31"
}
```


POST - Manual correction of time off balances: Desde este endpoint podrás ajustar los saldos de las vacaciones de los colaboradores sumando o restando días disponibles.

userEmployeeInternalId

Tipo: string
Descripción: Es el nombre de usuario al que se le editarán días en las políticas de Vacaciones. Es obligatorio completar esta información.

operation

Tipo: string
Descripción: Este campo sirve para decidir si se suman o se restan días. Las posibilidades de elección son “ADDITION” o “SUBTRACTION”. Es obligatorio completar esta información.

amount

Tipo: double
Descripción: Este campo es el resultado de días a agregar o quitar en una política de Vacaciones. Es obligatorio completar esta información.

observations

Tipo: string
Descripción: Es un campo visibile solo en la base de datos. Sirve para subir la descripción de la adición o substracción de saldos. Es opcional completar esta información.

Un ejemplo de un Json podría ser el siguiente:

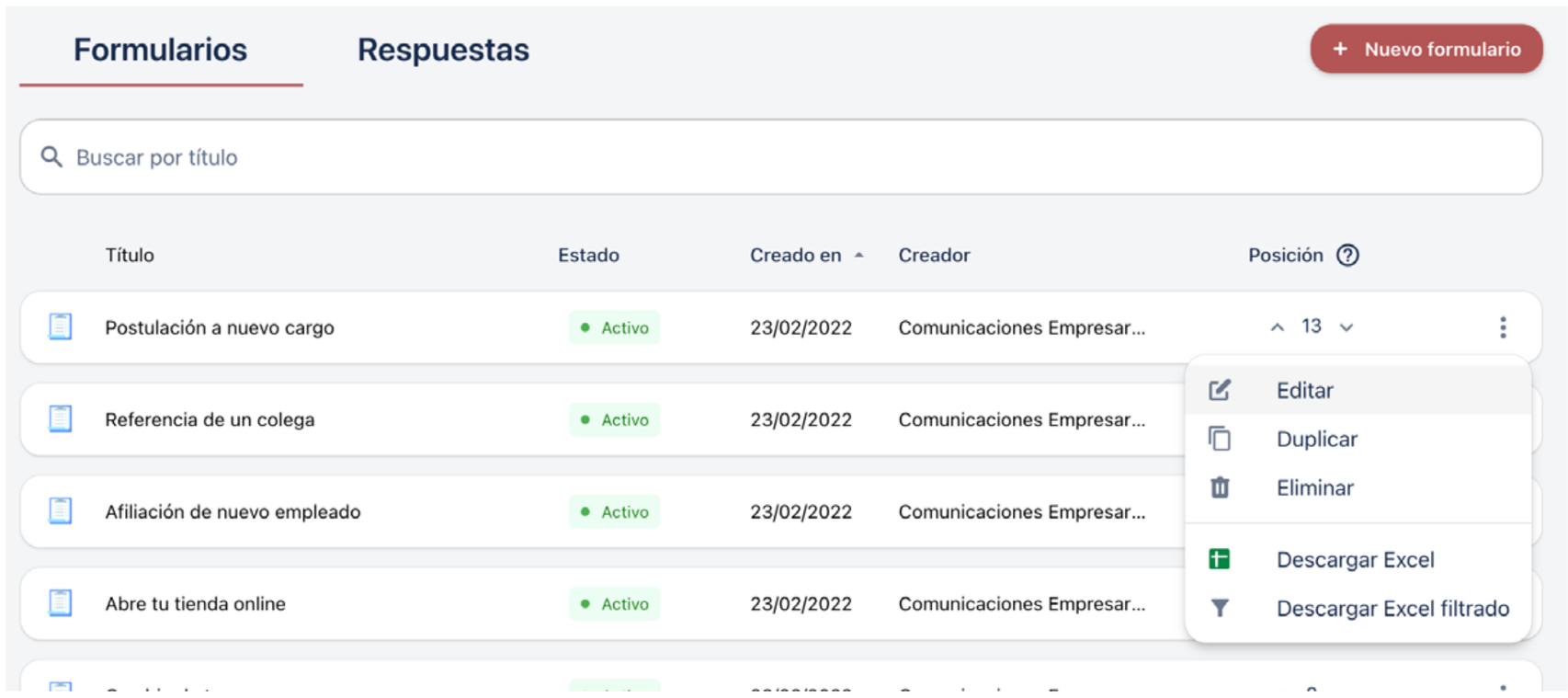
```
{
  "employeeInternalId": "userEmployeeInternalId",
  "operation": "ADDITION",
  "amount": 1,
  "observations": "Observations"
}
```

Formularios

GET - Forms: Desde este endpoint se podrán obtener los resultados de un formularios en particular.

ID

Tipo: Integer
Descripción: El 'Id' es un identificador numérico que corresponde a cada uno de los formularios. Para encontrar este valor, se debe acceder a la plataforma desde la cuenta de administrador y navegar a la sección Formularios. El 'ID' se encuentra en la URL del Formulario, al mismo se accede haciendo click en Editar:



Es importante tener en cuenta que este identificador se mantiene constante a lo largo del tiempo.

En el siguiente ejemplo el forms ID sería: 1465536

Page

Tipo: Integer
Descripción: El campo 'page' especifica el número de la página actual que se está solicitando. Es obligatorio completar esta información. Por ejemplo “page=4” indica que se está solicitando la cuarta página de resultados.

Limit

Tipo: Integer
Descripción: Este campo define la cantidad máxima de elementos o registros que se desean obtener por página, el valor máximo permitido es 500, controlando así el tamaño de cada segmento de datos entregado. Es obligatorio completar esta información. Por ejemplo “limit=10” indica que se desean obtener hasta 10 registros por página.

Order

Tipo: String
Descripción: Este campo sirve para ordenar las respuestas de los formularios y así ver cuales fueron Rechazados y cuales fueron aprobados.
Las posibles respuestas son las siguientes: ASC, DESC, ASC_NULLS_FIRST, DESC_NULLS_LAST

OrderBy

Tipo: String
Descripción: Este campo sirve para ordenar las respuestas según quien respondio el formulario o bien según su fecha de completado. Las posibles respuestas son las siguientes: CREATOR, CREATED_AT

creationDate

Tipo: String
Descripción: Este campo sirve para buscar respuestas en una fecha determinada. La manera de escribir la fecha es YYYY-MM-DD

Un posible Json sería el siguiente:

```
{
  "id": 1465536,
  "page": 1,
  "limit": 100,
  "order": "ASC",
  "orderBy": "CREATED_AT",
  "creationDate": "2024-05-02",
}
```

Time Tracking (WIP)

Humand Time Tracking cuenta con dos APIs dedicadas a recibir registros horarios para ser integrados al sistema de Control Horario y poder ser consumidos dentro de la plataforma y de sus reportes.

TIME TRACKING - TIME ENTRIES INTEGRATION

Mediante los siguientes métodos es posible sincronizar time-entries dentro de Humand generados a partir de otros sistemas o terminales.

Funcionalidades generales:

- Recibe registros horarios con timezone UTC-0: Todos los registros recibidos deben encontrarse bajo ese timezone ya que luego el sistema reflejará esa información en función del timezone configurado en la comunidad
- El body de los siguientes endpoints permite recibir un registro horario por request
- En caso de recibir un registro seguido del anterior con un mismo "Type" para un mismo día, será rechazado el último registro recibido
- Puede sincronizarse tanto registros pasados como futuros respetando sincronizar primero el registro de tipo START y luego su registro de END

POST - /time-tracking/entries/clockIn - Clock-in entries:
Recibe registros horarios de tipo START de cualquier colaborador de la comunidad.

employeeId

Tipo: Integer
Descripción: Es el campo "Usuario" configurado dentro de la comunidad.

Now

Tipo: Timestamp
Descripción: Es el registro horario a sincronizar UTC-0.

Comment

Tipo: String
Descripción: Es un campo libre utilizado para adicionar algún tipo de identificador de la terminal utilizada que luego será incluida como un comentario del registro horario dentro del sistema.

Un posible Json sería el siguiente:

```
{
  "employeeId": 1,
  "now": "2021-07-10T22:50:00.010Z",
  "comment": "string"
}
```

POST -/time-tracking/entries/clockOut - Clock-out entries:
Recibe registros horarios de tipo END de cualquier colaborador de la comunidad.

employeeId

Tipo: Integer
Descripción: Es el campo "Usuario" configurado dentro de la comunidad.

Now

Tipo: Timestamp
Descripción: Es el registro horario a sincronizar UTC-0.

Comment

Tipo: String
Descripción: Es un campo libre utilizado para adicionar algún tipo de identificador de la terminal utilizada que luego será incluida como un comentario del registro horario dentro del sistema.

Un posible Json sería el siguiente:


```
{
  "employeeId": 1,
  "now": "2021-07-10T22:50:00.010Z",
  "comment": "string"
}
```


Soporte y Contacto

Si tiene alguna pregunta o necesita asistencia adicional, no dude en ponerse en contacto con nuestro equipo de soporte vía mail o por el grupo de Whatsapp.

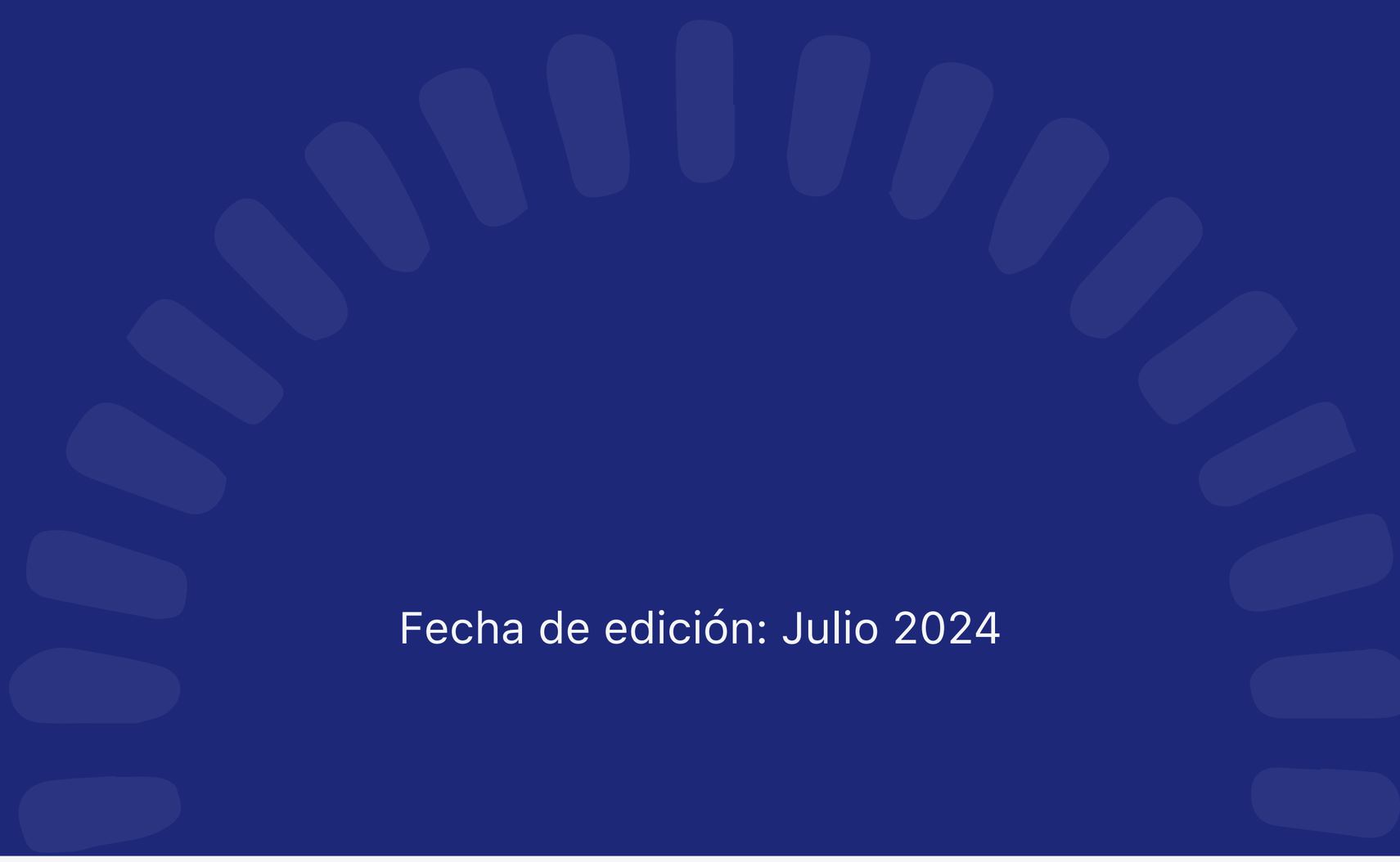
help@humand.co

Conclusión

Este manual proporciona una introducción completa a la API y su uso. Esperamos que esta guía te ayude a integrar y utilizar nuestra API de manera efectiva.

humand

¡Gracias por elegir Humand!



Fecha de edición: Julio 2024