# GearTranslations Client

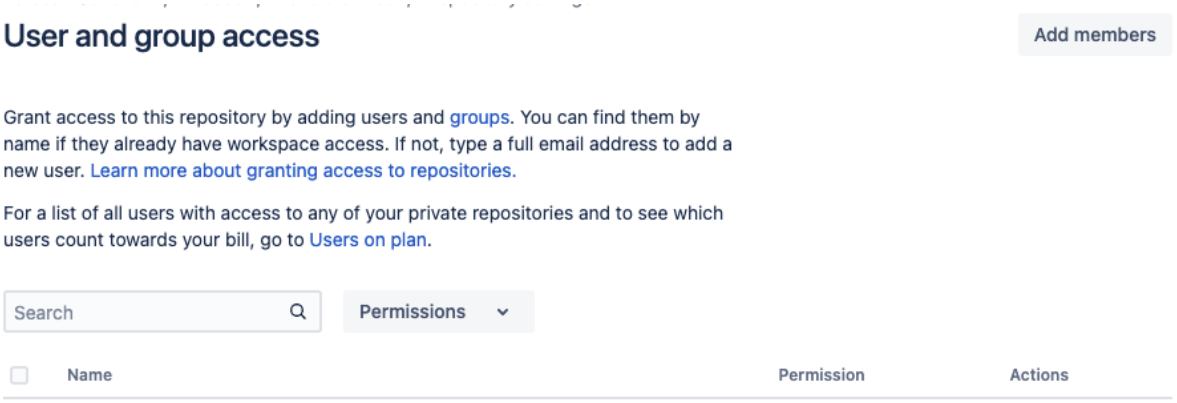
El cliente(gt-api-client) se encarga todos los días a las 21 hs UTC observar si hubo cambios en los locales de los repositorios que se dieron de alta. Por cada repositorio se levanta el archivo *geartranslations.yml* de esa forma el cliente sabe donde se encuentran los locales en dicho repositorio. Luego se procede a verificar por cada locale, si hubo un cambio desde la última ejecución, en caso de que así sea, se genera un proyecto de traducción en gears translations. Cuando el proyecto finaliza en gears translations se crea un pull request en el repositorio de dicho locale.
Un detalle a tener en cuenta, si existen pull request abiertos por el cliente para un locale, hasta que los mismos nos se encuentren mergeados o declinados el cliente no observara al locale involucrado.

## Usuario bitbucket para el cliente

Para poder observar un repositorio de bitbucket es necesario tener un usuario que pueda leer y escribir en el mismo, para esto asociarlo como un miembro del repositorio y por último generar las credenciales para que el cliente lo pueda utilizar.
El cliente permite establecer que usuario se utilizará para observar a cada repositorio, por lo tanto se puede tener tantos usuarios como repositorios o un único usuario para todos los repositorios, eso es a gusto, pero tener en cuenta que cada usuario se requerirá crearle unas credenciales(App Password).

Estos serían los pasos a realizar en un repositorio de bitbucket para agregar el usuario del cliente al repositorio.

1. Crear un usuario en bitbucket o tomar uno existente
2. Al usuario del paso previo, asociar el repositorio que se desea observar. Para esto necesitamos ingresar a bitbucket con un usuario que ya sea miembro del proyecto y tenga permisos para agregar nuevos usuarios.
    a. Ir al root del proyecto, en el menú lateral izquierdo nos vamos a Repository Settings -> User and group access y le damos a Add members, buscamos y agregamos el usuario-
    b. Es necesario darle permisos de escritura, para poder crear los pull request.

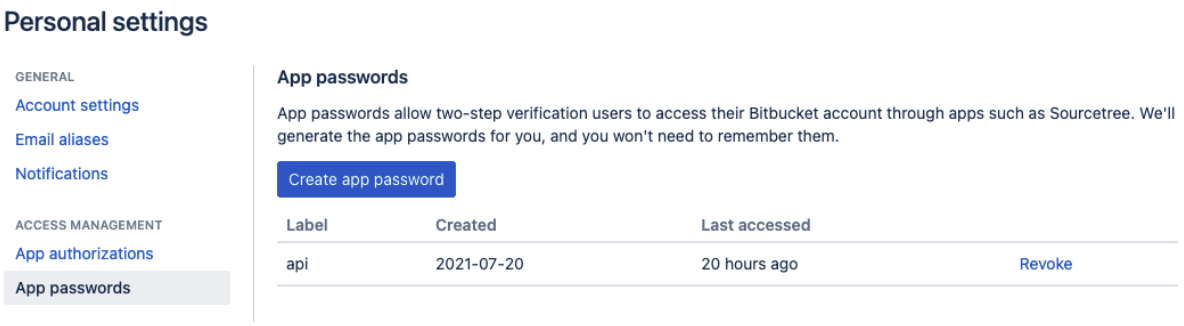



3. Esto es necesario realizarlo en cada repositorio.

## Generar credenciales del usuario

Cada usuario que se va a utilizar para observar un repositorio es necesario crearle un App Password.

1. Ingresar con el usuario a bitbucket.
2. Crear un App Password.




   a. Es necesario seleccionar los permisos de repository:write y pullrequest:write
   b. Guardar este app password en algún sitio seguro.
3. Con el username y el app password obtenido vamos a poder utilizar la api de bitbucket. Estos datos se van a requerir a la hora de dar de alta un repositorio en el cliente.
4. Para más detalles, podemos acceder a la documentación oficial de bitbucket.

## Archivo de configuración

Es necesario agregar en cada repositorio a observar un archivo en el cual se indica la ubicación y los locales a observar, el formato del mismo, el idioma origen, el idioma destino y un tag.
Además, es necesario indicar el access token con el cual queremos crear los proyectos de traducción en geartranslations. Este access token se puede obtener o solicitar en la plataforma, el mismo se encuentra asociado a un usuario de la misma.
El archivo debe de ser ubicado en el root del proyecto bajo el nombre de *geartranslations.yml*, a continuación se deja un ejemplo del mismo.

```
geartranslations:
  tag: "URGENT"
  access_token: 'accesstoken'
  sources:
     - file:
        name: locales/fr.json
        locale: 'fr'
        aligned_from: 'es'
        format: 'json'
     - file:
        name: locales/en.json
        locale: 'en-uk'
        aligned_from: 'es'
        format: 'json'
```

Podemos encontrar un archivo de ejemplo similar en el root del cliente bajo el nombre de *example-geartranslations.yml.*

## Ejecutar el proyecto

1. Clonar el repositorio
2. Establecer en el docker-compose las siguientes variables
    a. ENCRYPTION_KEY tiene que ser una key de 64 bits, es utilizada para encriptar
    b. ATTRIBUTE_ENCODING_KEY tiene que ser una key de 32 bits de longitud
    c. SECRET_KEY_BASE tiene que ser un string aleatorio en lo posible arriba de los 100 bits
3. docker-compose build
4. docker-compose up

a. En caso de que el docker-compose up no cree la base de dato será necesario correr lo siguientes:
b. docker-compose run gt-api-client rake db:create
c. docker-compose run gt-api-client rake db:migrate

De esta forma tenemos al cliente corriendo en el puerto 3000, además en /sidekiq podemos ver los distintos jobs encargado de las distintas tareas dentro del cliente.
En este momento lo que nos falta es indicarle al cliente qué repositorios queremos observar.

## Alta de repositorios a observar

Para esto es necesario armar un archivo llamado repositories.yml y pasarlo al contenedor del cliente para luego correr una task que lo levante. Dentro del repositorio existe el archivo *example-repositories.yml* que sirve de ejemplo de como debe de ser el file..

El archivo repositories.yml tiene el siguiente formato:

```
repositories:
  - repository:
      workspace: ‘gears’
      repository_name: 'mobile-api'
      branch: 'master'
      branch_pull_request_destination: 'master'
      user_name: 'user_name'
      app_password: 'app_password'
      server_url: 'https://server.bitbucket-url.com'
      platform: 'bitbucket'
  - repository:
      workspace: ‘gears’
      repository_name: 'front'
      branch: 'main'
      branch_pull_request_destination: 'release'
      user_name: 'user_name'
      app_password: 'app_password'
      server_url: 'https://server.bitbucket-url.com'
      platform: 'bitbucket'
```

Por cada repositorio a observar es necesario cargar el workspace, nombre del repositorio, el branch donde se encuentra el *geartranslations.yml*, brach_pull_request_destination para indicar la rama a la que se desea aplicar los cambios a la hora de crear el pull request, el usuario y su app password, la url del servidor de bitbucket y platform, en este caso bitbucket.

Una vez generado el file repositories.yml, se lo pasamos al contenedor(gt-api-client) y ejecutamos la tarea, para esto seguir los siguientes pasos:

1. docker ps # Nos permite obtener el container id de la imagen gt-api-client
2. docker cp repositories.yml <container id>:/app
3. docker exec <container id> rake repositories:load

La tarea agrega todos los repositorios o ninguno, por lo tanto evitar duplicados o existentes porque la tarea va a fallar si sucede algo de lo mencionado.

## Actualizar el cliente

Es recomendado con cierta periodicidad actualizar el cliente para obtener las nuevas funcionalidades que el mismo puede tener, como así también la resolución de bugs. Cuando se desee realizar esto, es de suma importancia no modificar los valores de las variables de entorno ENCRYPTION_KEY, ATTRIBUTE_ENCODING_KEY y SECRET_KEY_BASE, porque si se hace, no vamos a poder recuperar las credenciales encriptadas. Si esto llega a suceder, va a ser necesario cargar nuevamente el access token, y las credenciales de bitbucket para cada repositorio.

Para esto es necesario traer los últimos cambios del repositorio y volver armar la imagen de docker, para realizar esto debemos seguir los siguientes pasos:

1. En el root del proyecto realizar un git pull origin main --rebase
2. docker-compose build
3. docker-compose up