IMTRA

Image Manager and TRansfer Application

Unai Díaz, Marck Carrión

Índice

1. Introducción

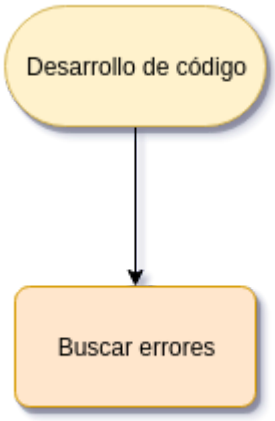
Image Manager and TRansfer Application (IMTRA) es un programa orientado a aquellos usuario de Linux que sean apasionados de la fotografía. Está desarrollado principalmente en **python 3** y cuenta con una interfaz atractiva al público desarrollada con **tecnologías web**.

IMTRA surge de la necesidad de programas para la organización y transferencia de fotos en el entorno linux, pues como siempre ocurre en la comunidad linux, existen programas funcionales pero carecen de una interfaz gráfica agradable. Casi todos tienen interfaces que no siguen los estandares de diseño de interfaces. IMTRA proporciona una interfaz minimaista y limpia para que el usario tenga lo que necesita siempre a mano.

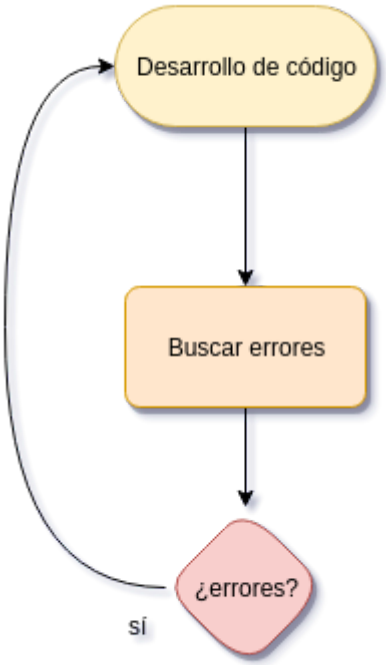
Así mismo *IMTRA* desde el punto de vista del desarrollo permite utilizar sus modulos de forma independiente, se trata de una aplicación totalmente modular. **No existe una dependencia directa entre los módulos** es el controlador de la aplicación quien se encarga de interconectarlas.

2. WorkFlow

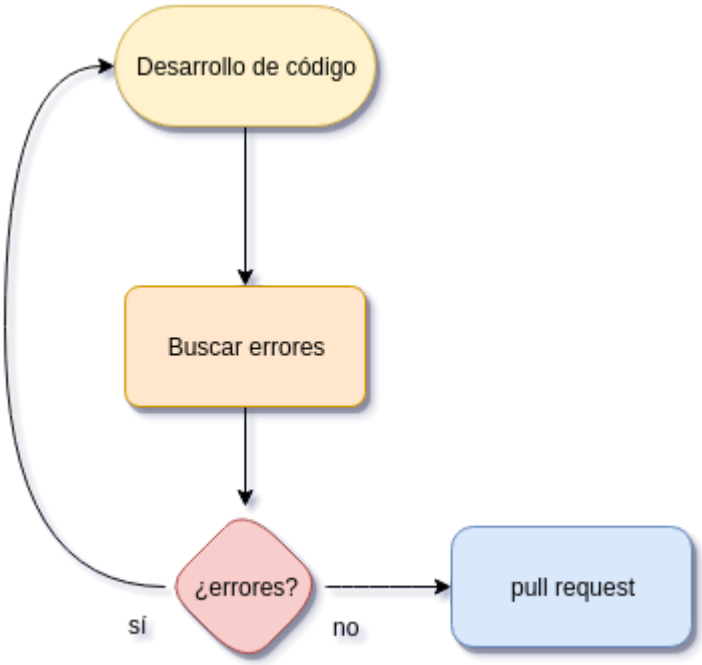
Para el desarrollo del proyecto se preestableció un *flujo de trabajo (WorkFlow)* para automatizar el trabajo en grupo y evitar posibles conflictos.
El workflow empieza con el desarrollo de código, bien sea una función, un método o una clase entera, al terminarla el desarrollador se encarga de probar el código que acaba de escribir, realizando las pruebas que vea necesarias, en busca de errores.



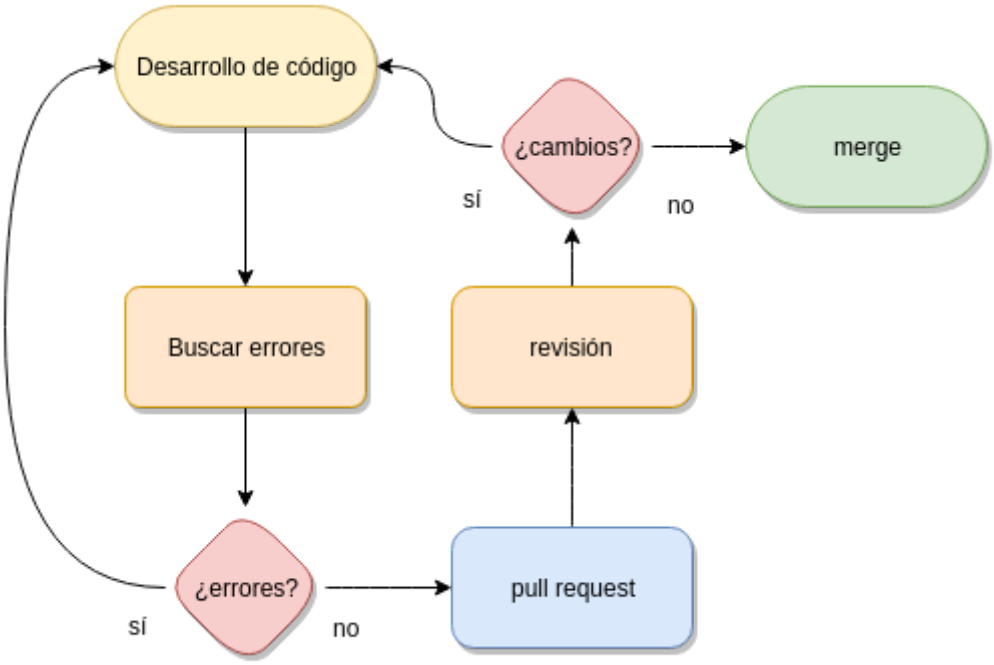
En caso de encontrar algún error en su código se volverá al principio, esta vez para arreglar los errores.



Si no ha encontrado ningún error se pasará a hacer una pull request, se creará una rama nueva en el repositorio remoto.



Estos combios se mantendran en la rama temporal hasta que otro miembro del grupo lo revise para comprobar que está comentado correctamente, que no haya errores de ejecución ni de compilación. Una vez comprobado que todo es correcto se procede a fusionar la rama temporal con la principal

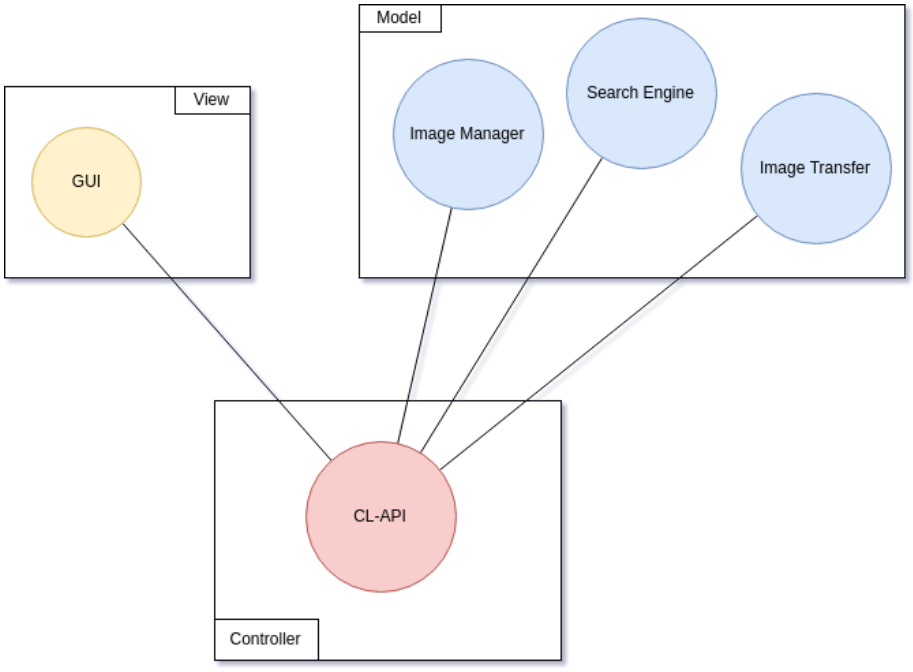


3. Overview

La aplicación cuenta con tres módulos principales más el controlador y la interfaz gráfica. Los tre módulos principales son:

- *Image Transfer*
- *Search Engine*
- *Image Manger*

La comunicación entre los módulos principales se hace mediante el módulo **CL-API (Command Line Application Programming Interface)**.



4. Librerías

En desarrollo de la aplicación se ha visto la necesidad de buscar librerías y APIs para facilitar y agilizar el desarrollo de la misma.

4.1 Interfaz

Para la interfaz se ha usado una librería que permite la creación de un servidor local temporal y mediante el motor de *Google Chrome* o de *Chromiun* permite visualizar una interfaz hecha completamente en **HTML 5** y **CSS3**. Se trata de la libreía **eel**.

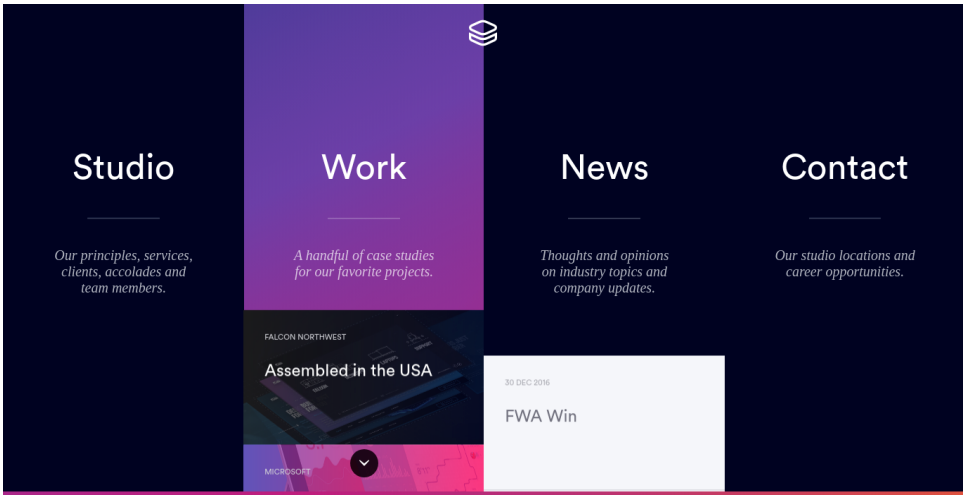
4.2 Inteligencia Artificial

Para el tema de la *inteligencia artificial* se ha utilizado una API REST **para agilizar el procesado de imágenes y evitar la dependencia del hardware**. En concreto se ha usado Clarifai.

5. GUI (Interfaz Gráfica)

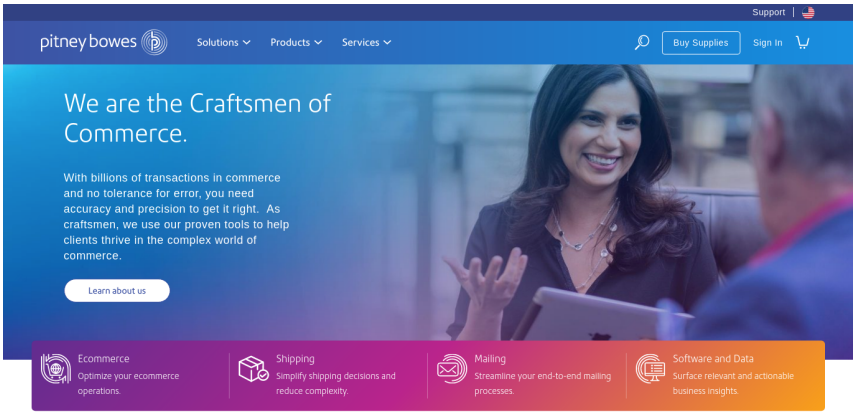
Se realizó un estudio para obtener lo más utilizado en diseño web y se observo una tendencia a colores vivos, alto contraste y degradados.

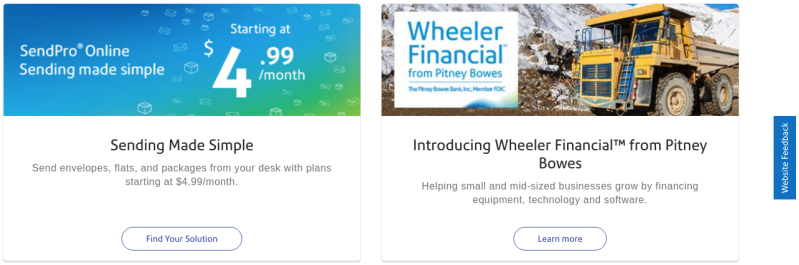
- http://www.impossible-bureau.com/

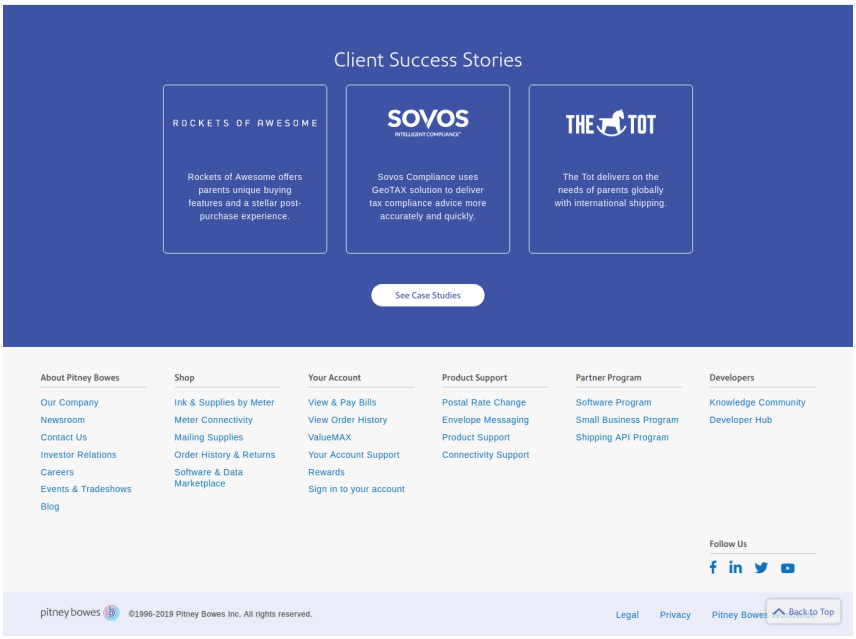


- http://www.melanie-f.com/en/

- https://www.pitneybowes.com/us

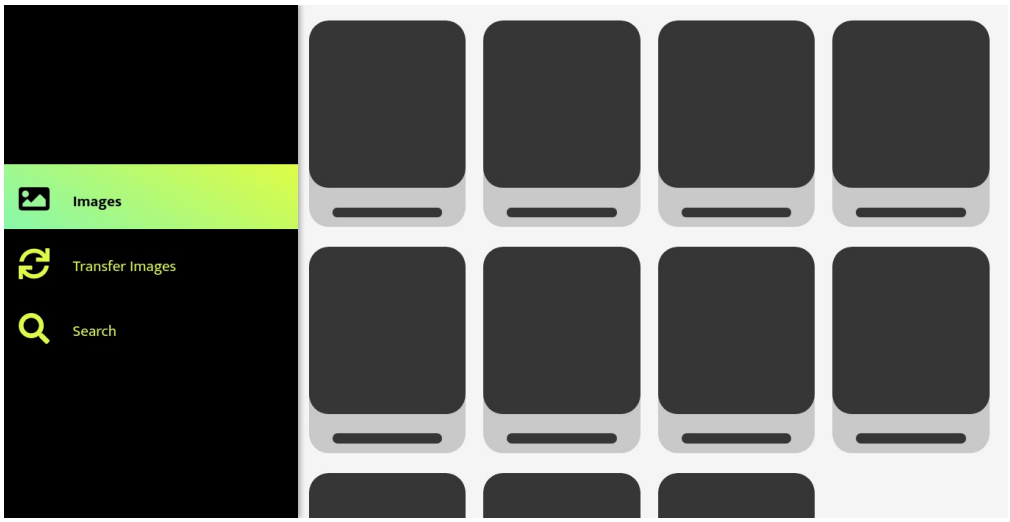




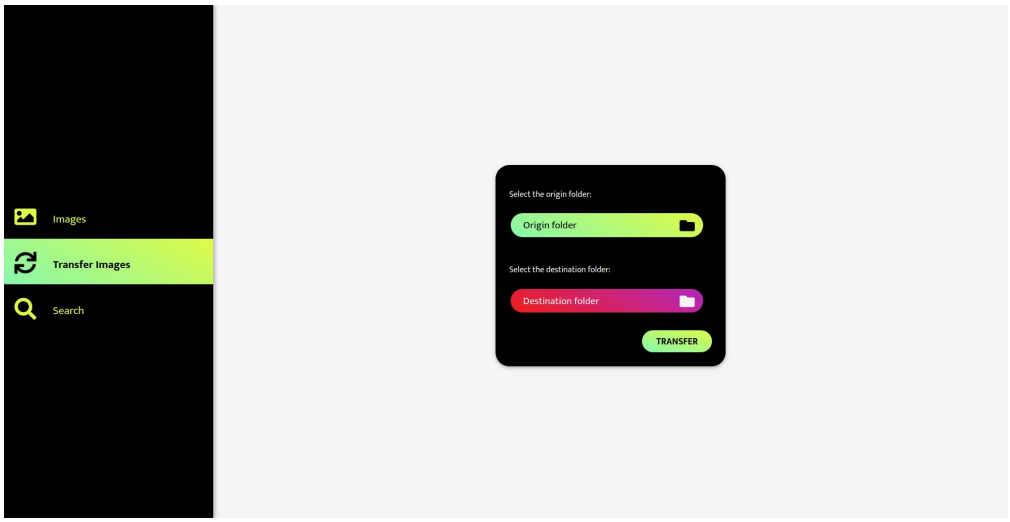


Teniendo en cuenta este estudio se estableció un diseño para la interfaz del programa:

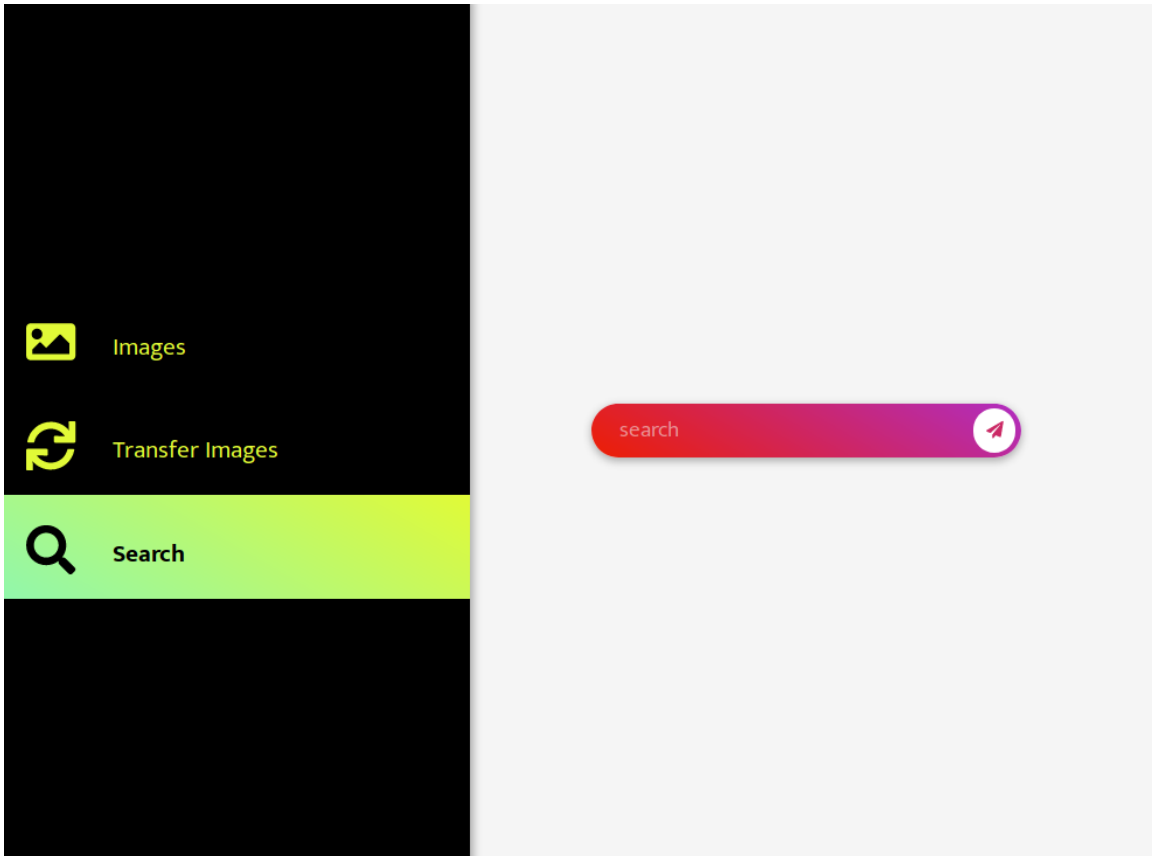
- **Pantalla de Image Manager**



- **Pantalla de Image Transfer**



- **Pantalla de Search**



6. Módulo Image Transfer

Este módulo se encarga de la transferencia de imágenes desde la carpeta de origen a la de destino, establecidas por el usuario. Durante la transferencia se generan carpetas siguiendo un patrón.

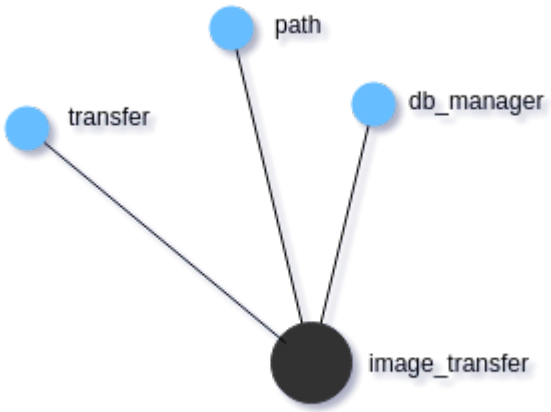
6.1 Introducción

El módulo consta de clases y funciones que están orientadas a la transferencias de imágenes, que facilitan la lectura de algunos metadatos de las imágenes.

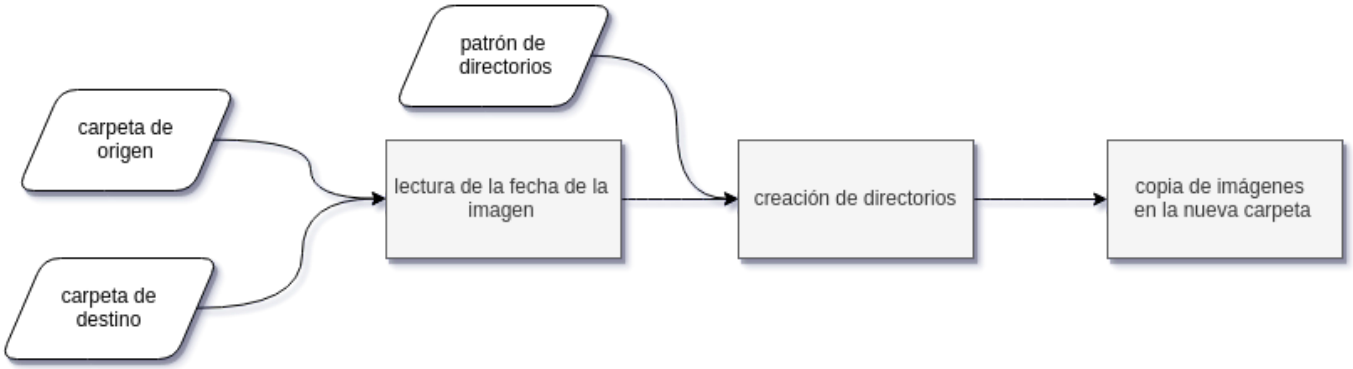
6.2 Esquema del módulo

El módulo tiene tres submodulos:

- *transfer*
- *path*
- *db_manager*



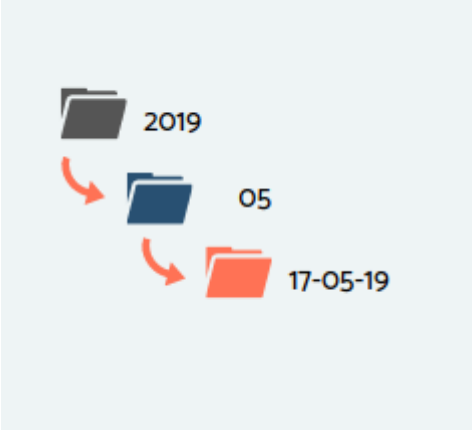
Los submodulos **transfer** y **path** trabajan conjuntamente para poder generar la estructura de carpetas leyendo la fecha de cada imagen y creando la carpeta. El funcionamiento es el siguiente:



Necesita los directorios de origen, donde se encuentran todas las imágenes, y un directorio de destino, donde se generará la estructura de carpetas. También necesita un patrón para poder generar la estructura de carpetas, el cual vendrá por defecto pudiendo cambiarla pasandola como parametro.

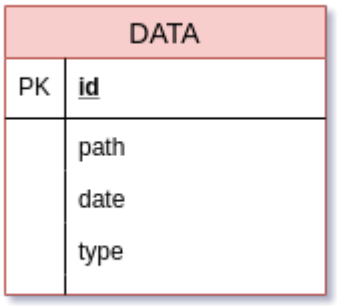
```
def dir(d, format = 'yyyy/MM/dd-MM-yy', sys_date=False):
```

Para generar la carpeta que corresponde a cada imagen, también se le pasa la misma imagen, la cual se lee y se extrae la fecha con la se juega para crear la estructura de carpetas.
Para el patrón `yyyy/MM/dd-MM-yy` se consigue una estructura:



6.3 Base de datos

Para este módulo se planteó utilizar una base de datos embebida para el almacenage masivo de información relacionada con las tranferencia de imágenes. Gracias a esta base de datos se dotará de cierta inteligencia a la interfaz, pues mediante una consulta se traerá las transferencias más realizadas y se podrá autocompletar las direcciones de las carpetas.



La base de datos almacena una dirección absoluta, junto con la fecha y el tipo de carpeta, carpeta de origen o de destino, luego mediante una consulta se consigue sacar las carpetas más utilizadas y mas cercanas a la feecha de la petición.

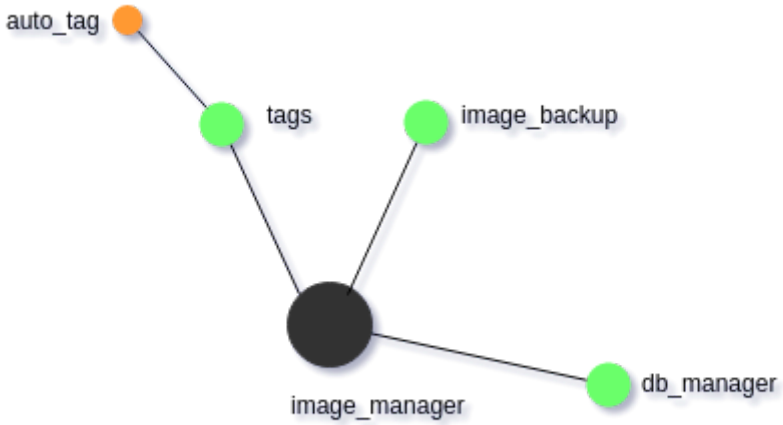
7. Módulo Image Manager

7.1 Introducción

La función principal del módulo es ofrecer tanto al usuario final como al desarrollador las herramientas necesarias para la organización de imágenes y la etiquetación de las mismas, contando una base de datos para almancenar la información imprescindible.

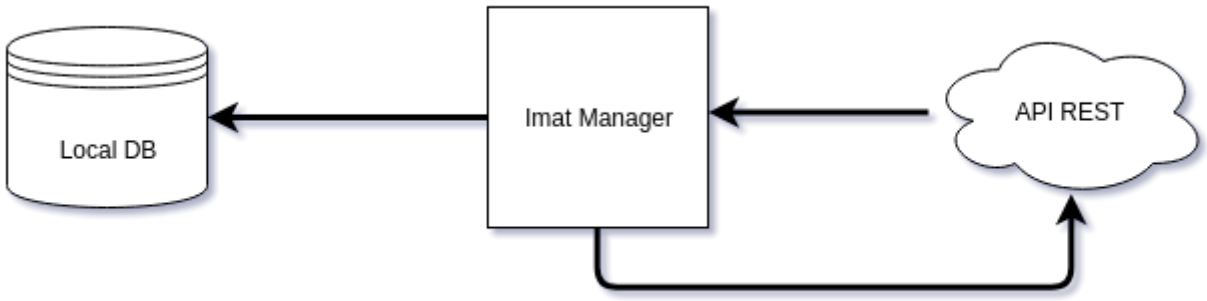
7.2 Esquema del módulo

Este consta de un etiquetador automático, una base de datos propia y un sistema de copias de seguridad.



7.1.1 Etiquetador Automatico

Mediante una *API REST* de inteligencia artificial se procesa la imagen y detecta los elementos que se encuentran en la imagen.
El funcionamiento es simple, la función recibe una imagen que la lee de forma binaria y envía los bytes al servidor, este los procesa mediante inteligencia artificial, tomando como referencia un modelo preestablecido, y devuelve un archivo en **JSON**. La respuesta se procesa y se almacena en una base de datos embebida.

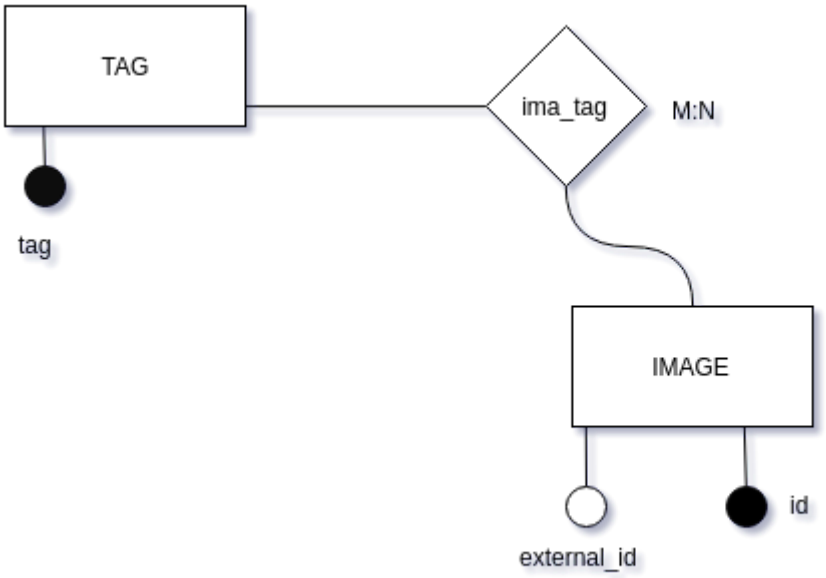


7.1.2 Backup Tools

Permite crear copias de seguridad de las carpetas y las imágenes que se le pasas como argumento. Utiliza la compresión tar para comprimir todos las carpetas de la forma más efieciente.

7.3 Base de datos

El administrador de imágenes utiliza una tabla para relacionar la imágen con un id, esto se hace para no almacenarla directamente, se relaciona la dirección absoluta de la imágen con un id numérico. Esto nos resultará utili para trabajar de forma más abstracta con las imágenes y poder crear interrelaciones con los otros módulos.



La base de datos interrelaciona las etiquetas que se leen desde el servidor con la imagen correspondiente.

8. Módulo Search Engine

Este módulo proporciona una herramienta de búsqueda completa y compleja. Permite al usuario hacer búsquedas atendiendo a los metadatos de las imágenes.

8.1 Introducción

El motor de búsqueda se encarga de leer los metadatos de lás imágenes y genera una base de datos para poder optimizar la búsqueda, ya que si se tuviese que leer los metadatos en cada búsqueda se tardaria alrededor de 500 milésimas de segundo en procesar cada imagen por lo que teniendo 100 imágenes el tiempo de espera sería de unos 50 segundos.
Este módulo proporciona el motor de búsqueda y a la vez el lector de metadatos, etiquetas EXIFs.

8.2 Esquema del módulo

El módulo cuenta con tres ficheros claves, el encargado de la lectura de las etiquetas EXIF, los metadatos de las imágenes, el que se encarga de gestionar las peticiones a la base de datos y el encargado de la búsqeda de las imágenes.

El *lector EXIF* tiene como objetivo facilitar la lectura de los metadatos de una imagen, pudiendo filtrar los metadatos según una lista de filtrado, dónde estarán las etiquetas que se desee.

```
class Reader:
    def __init__(self, img):
        pass
    def get_data(self) -> dict:
        pass
    def get_filter_tag(self, filter_list: list) -> dict:
        pass
```

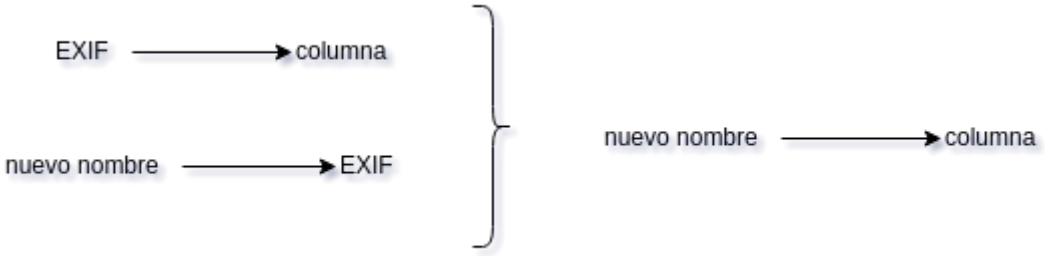
Obtener todos los datos de una image:

```
reader = Reader("/home/user/images/img.cr2") # asignamos la ruta de la imagen
all_data = reader.get_data() # leemos todos los metadatos
```

Para obtemer los datos filtrados hace falta pasarle una lista con las etiquetas que se desee, esto supone que el desarrollador deberá conocer los nombres exactos de las etiquetas que quiere obtener:

```
reader = Reader("/home/user/images/img.cr2") # asignamos la ruta de la imagen
filter_list = ('EXIF OriginalDate', 'IMAGE CameraVendor', 'IMAGE maker')
all_data = reader.get_filter_tag(filter_list) # leemos los metadatos que estén en la li
```

Por otro lado el buscador permite al desarrollador establecer diccionarios de traducción mediante la propiedad transitiva, existen dos dicionarios en los que uno establece el nombre para cada etiqueta EXIF y otro que relaciona el nombre de las columnas con las etiquetas EXIF:

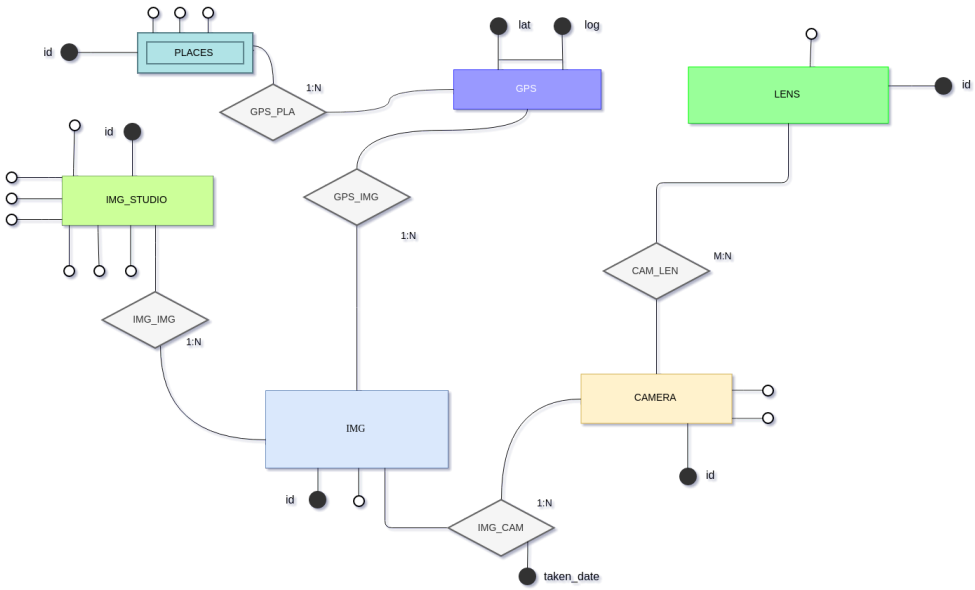


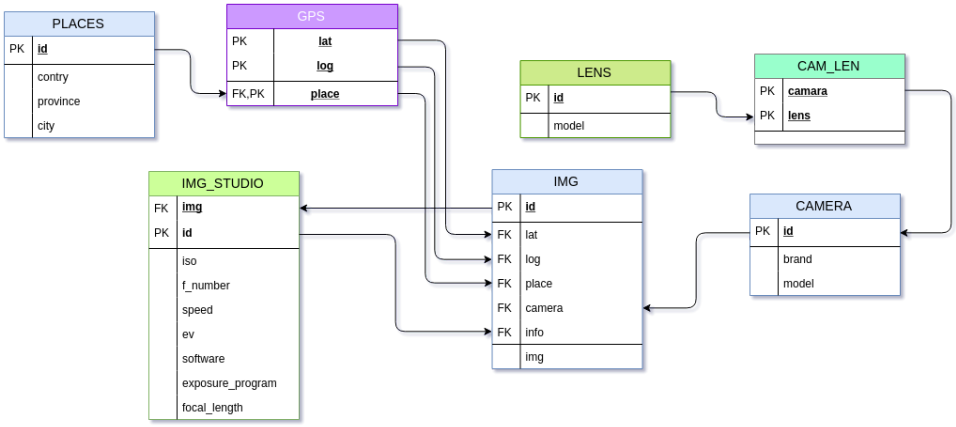
por lo que al combinar ambos tenemos acceso a las columnas de la base de datos por un alias que se establezca. Esto proporciona un abstracción a la hora de desarrollar porque no es necesario conocer la base de datos. A su vez esto permite la búsqueda utilizando solo diccionarios como si se tratase de una petición con JSON:

```
class Search:
    def search(self, data: dict):
        pass
```

8.3 Base de datos

Este módulo posee la base de datos más compleja de la aplicación, en ella se almacena todos los metadatos relevantes de las imágenes.





Gracias a esta base de datos se consigue optimizar la busqueda y la hace pausible.