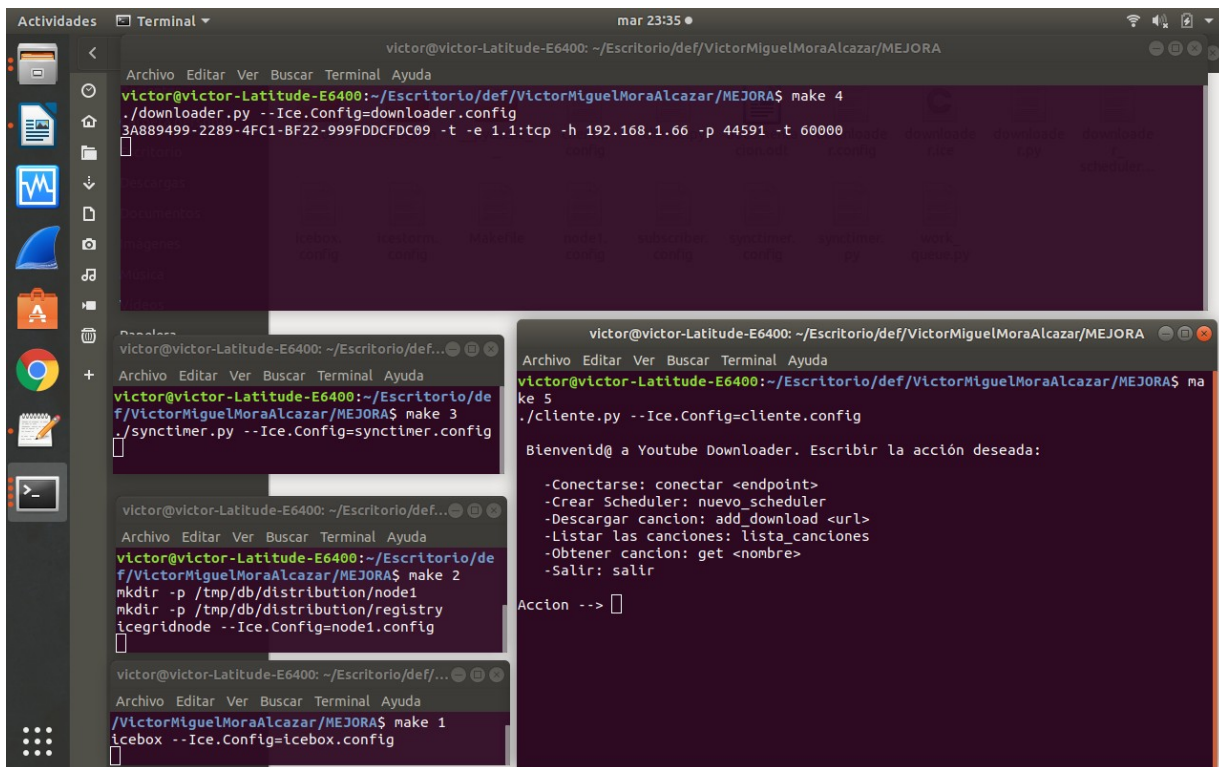


# DOCUMENTACIÓN BREVE: YOUTUBE DOWNLOADER

## 1. Inicio de la aplicación



The screenshot shows a Linux desktop environment with four terminal windows. The top window shows the command `make 4` being executed, which compiles the `downloader.py` file using the `Ice.Config=downloader.config` configuration. The second window shows `make 3` for `synctimer.py` with `Ice.Config=synctimer.config`. The third window shows `make 2` for `icegridnode` with `Ice.Config=node1.config`. The fourth window shows `make 1` for `icebox` with `Ice.Config=icebox.config`. The desktop background is dark, and the terminal windows have a light background.

*Inicio de la Aplicación (todos los actores juntos)*

En este documento, se explicarán brevemente las funciones que pueden realizarse. Al final del todo (punto 8) se hace un resumen de a que se da soporte y a que no.

## 2. Que ve el cliente



The screenshot shows a terminal window with the command `make 5` and `./cliente.py --Ice.Config=cliente.config` being executed. The output shows the welcome message "Bienvenid@ a Youtube Downloader. Escribir la acción deseada:" followed by a list of actions: `-Conectarse: conectar <endpoint>`, `-Crear Scheduler: nuevo_scheduler`, `-Descargar cancion: add_download <url>`, `-Listar las canciones: lista_canciones`, `-Obtener cancion: get <nombre>`, and `-Salir: salir`. The prompt "Accion -->" is shown at the bottom.

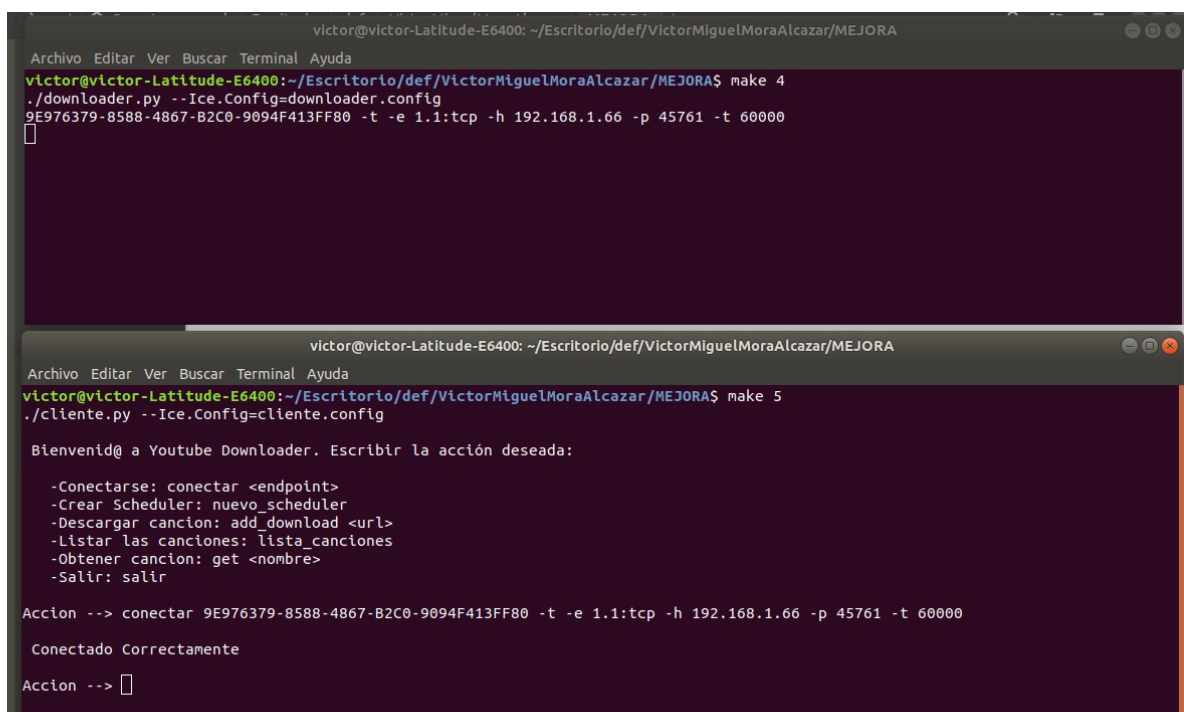
*Punto de partida del cliente*

Lo primero que se muestra es una intro de bienvenida, y a continuación se listan las funciones disponibles (y el “comando” para pedir su realización).

Las desarrollare un poco mas adelante, pero las citaré brevemente:

- El cliente podrá conectarse a la factoría (puede y debe, si es que quiere usar la aplicación), con el comando **conectar** <endpoint>
- Una vez conectado, podrá mandarle una descarga al servidor, mediante la url de un video de youtube y el comando **add\_download** <url>
- Si al pedir el cliente una descarga u otra acción sobre los ficheros (get, lista de canciones...) no hay ningún servidor disponible para la descarga, se creará automáticamente
- El cliente puede solicitar la creación de uno manualmente con **nuevo\_scheduler**
- Para ver las canciones descargadas por el servidor, se usará **listar\_canciones**
- Para obtener el audio (almacenado localmente por el servidor al descargarlo) en el destino del cliente, se usara el comando **get** <cancion>
- Para salir, simplemente hay que escribir **salir**

### 3. Conectarse



```
victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 4
./downloader.py --Ice.Config=downloader.config
9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000
□

victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 5
./cliente.py --Ice.Config=cliente.config

Bienvenid@ a Youtube Downloader. Escribir la acción deseada:

-Conectarse: conectar <endpoint>
-Crear Scheduler: nuevo_scheduler
-Descargar cancion: add download <url>
-Listar las canciones: lista_canciones
-Obtener cancion: get <nombre>
-Salir: salir

Accion --> conectar 9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Conectado Correctamente

Accion --> □
```

*Conectarse mediante el endpoint*

```
def do_conectar(self, line):
    '''Conectarse a la factoria (excepto si ya se esta conectado)'''
    if self.activo:
        print('\n Ya estas conectado \n')
        return None
    self.cliente.conectar_factoria(line)
```

*Orden Shell*

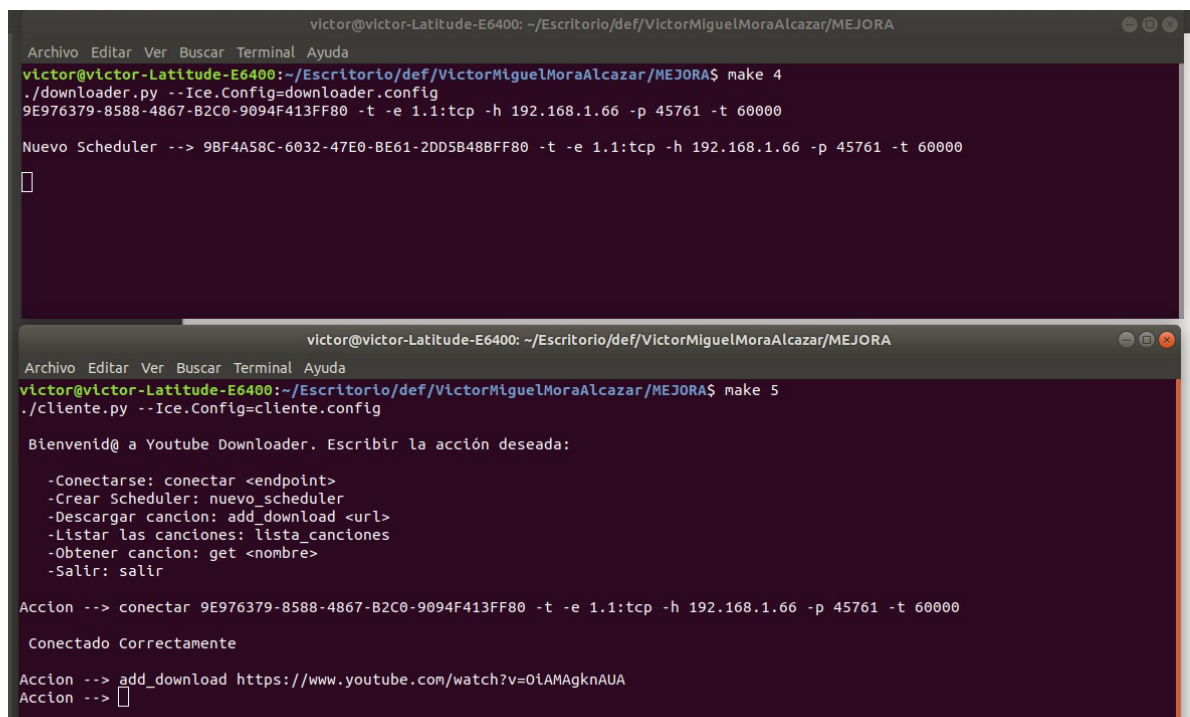
Si ya se esta conectado, se informa al usuario y no se realiza la conexión.  
Si no lo está, se crea la conexión a la factoría

```
def conectar_factoria(self, proxy):
    ''' Conectarse a la factoria. Avisar si la conexión se realiza'''
    proxy_factoria = self.communicator().stringToProxy(proxy)
    self.factoria = Downloader.SchedulerFactoryPrx.checkedCast(proxy_factoria)
    if self.factoria:
        print('\n Conectado Correctamente \n')
    else:
        print('\n Introducir el endpoint \n')
```

Se crea la conexión, si se conecta informa al usuario, si no se proporciona un endpoint se pide introducirlo para conectarse

#### 4. Función add\_download <url>

Solicitar la descarga de un audio de youtube mediante la url.



```
victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 4
./downloader.py --Ice.Config=downloader.config
9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Nuevo Scheduler --> 9BF4A58C-6032-47E0-BE61-2DD5B48BFF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000
[]

victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 5
./cliente.py --Ice.Config=cliente.config

Bienvenid@ a Youtube Downloader. Escribir la acción deseada:

-Conectarse: conectar <endpoint>
-Crear Scheduler: nuevo_scheduler
-Descargar cancion: add_download <url>
-Listar las canciones: lista_canciones
-Obtener cancion: get <nombre>
-Salir: salir

Accion --> conectar 9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Conectado Correctamente

Accion --> add_download https://www.youtube.com/watch?v=0iAMagknAUA
Accion --> []
```

*Solicitar descarga mediante url*

```
def do_add_download(self, line):
    '''Añadir nueva descarga mediante una url de youtube'''
    if not self.activo:
        print('Necesitas estar conectado a una factoria')
        return None
    self.cliente.add_download(line)
```

*Orden Shell*

```

@property
def scheduler(self):
    """
    Crear un scheduler si no hay ninguno. Fuentes consultadas:
    https://www.tutorialpython.com/listas-en-python/
    https://stackoverflow.com/questions/306400/how-to-randomly-select-an-item-from-a-list
    https://www.programiz.com/python-programming/property
    """
    if not self.dicc_schedulers:
        self.make_scheduler(str(uuid.uuid4()))
    return random.choice(list(self.dicc_schedulers.values()))

```

*Crear un scheduler si no hay ninguno disponible*

```

def add_download(self, url):
    ''' Añadir nueva descarga con el scheduler, se le pasa la url del audio a descargar '''
    if url is '':
        print ('Es necesaria una url de youtube')
    else:
        self.scheduler.addDownloadTask(url)

```

Lo primero, se comprueba si el cliente esta conectado a la factoria

Al solicitar la descarga si no hay ningún scheduler disponible se crea uno automáticamente usando uuid (se pueden crear de forma manual, lo explicare en otro apartado).

El scheduler encargado se selecciona de forma aleatoria en el diccionario de schedulers.

Si no se introduce ninguna url, se solicita al cliente una. En caso contrario, se descarga usando la función de *downloader\_scheduler.py* dispuesta para ello y atendiendo al slice proporcionado en el enunciado.

## 5. Función nuevo\_scheduler

```

victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 4
./downloader.py --Ice.Config=downloader.config
9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Nuevo Scheduler --> 9BF4A58C-6032-47E0-BE61-2D05B48BF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Nuevo Scheduler --> DA0C65F0-EE17-4C5C-A906-29EE7231FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000
[]

victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 5
./cliente.py --Ice.Config=cliente.config

Bienvenid@ a Youtube Downloader. Escribir la acción deseada:

-Conectarse: conectar <endpoint>
-Crear Scheduler: nuevo_scheduler
-Descargar cancion: add_download <url>
-Listar las canciones: lista_canciones
-Obtener cancion: get <nombre>
-Salir: salir

Accion --> conectar 9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Conectado Correctamente

Accion --> add_download https://www.youtube.com/watch?v=0IAMAgknAUA
Accion --> nuevo_scheduler
Accion --> []

```

*Crear un nuevo scheduler de manera manual*



```
def do_nuevo_scheduler(self, line):
    '''Crear un nuevo scheduler (si se esta conectado), nombre con uuid.
    Fuentes consultadas: https://docs.python.org/3/library/uuid.html'''
    if not self.activo:
        print('Necesitas estar conectado a una factoria')
        return None
    self.cliente.make_scheduler(str(uuid.uuid4))
```

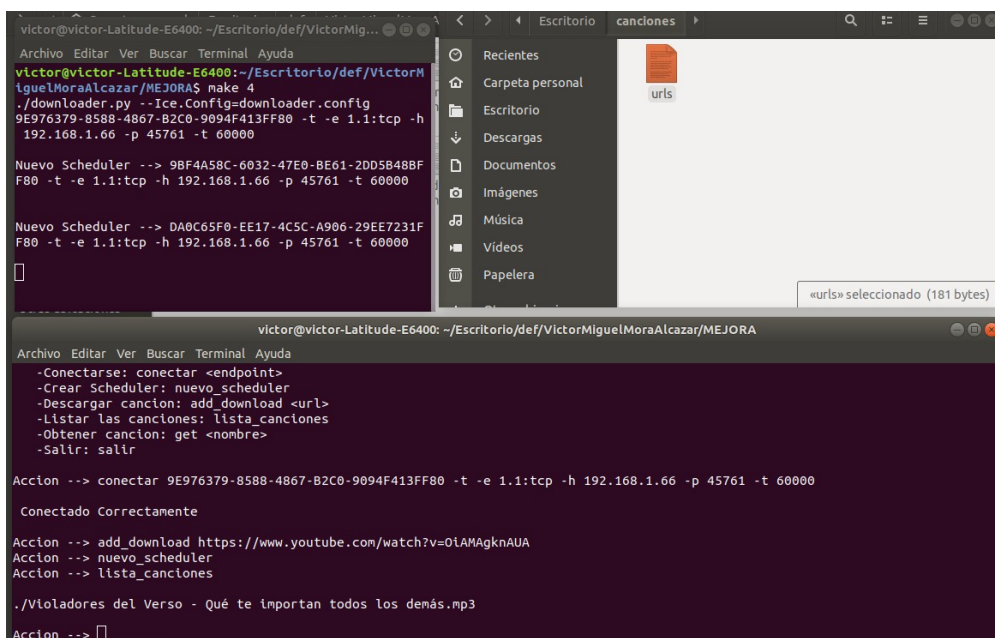
Orden Shell

```
def make_scheduler(self, nombre):
    '''
    Crear nuevo scheduler si se esta conectado a la factoria.
    Fuentes consultadas: https://www.youtube.com/watch?v=Ycu5Re7bEUE
    '''
    self.dicc_schedulers[nombre] = self.factoria.make(nombre)

def make(self, name=None, current=None):
    '''
    Crear un nuevo Scheduler y actualizar diccionario. Mostrar por pantalla.
    Fuentes externas consultadas:
    https://docs.python.org/3/library/uuid.html
    https://www.tutorialpython.com/listas-en-python/
    https://www.youtube.com/watch?v=Ycu5Re7bEUE
    '''
    servant = DownloadSchedulerI(self.tareas, self.sync)
    proxy = current.adapter.addWithUUID(servant)
    name = Ice.identityToString(proxy.ice_getIdentity())
    print('\nNuevo Scheduler --> %s\n' % (proxy))
    (self.dicc).update({name:proxy})
    return Downloader.DownloadSchedulerPrx.checkedCast(proxy)
```

El cliente puede crear un scheduler manualmente. Primero se comprueba (como en el resto de casos) si el cliente esta conectado, en caso de que lo este, se procede a la creación del scheduler, pasandole como nombre el resultante del identificador único (uuid), atendiendo también al slice proporcionado. Para el almacenamiento y gestión se usan diccionarios, asociando nombres y proxies.

## 6. Función lista\_canciones



```
victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
victor@victor-Latitude-E6400:~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA$ make 4
./downloader.py --Ice.Config=downloader.config
9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Nuevo Scheduler --> 9BF4A58C-6032-47E0-BE61-2DD5B48BF
F80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

Nuevo Scheduler --> DA0C65F0-EE17-4C5C-A906-29EE7231F
F80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000

victor@victor-Latitude-E6400: ~/Escritorio/def/VictorMiguelMoraAlcazar/MEJORA
Archivo Editar Ver Buscar Terminal Ayuda
-Conectarse: conectar <endpoint>
-Crear Scheduler: nuevo_scheduler
-Descargar cancion: add_download <url>
-Listar las canciones: lista_canciones
-Obtener cancion: get <nombre>
-Salir: salir

Accion --> conectar 9E976379-8588-4867-B2C0-9094F413FF80 -t -e 1.1:tcp -h 192.168.1.66 -p 45761 -t 60000
Conectado Correctamente

Accion --> add_download https://www.youtube.com/watch?v=0IAMAgknAUA
Accion --> nuevo_scheduler
Accion --> lista_canciones

./Violadores del Verso - Qué te importan todos los demás.mp3

Accion -->
```

Esta función permite consultar las canciones descargadas localmente por el servidor (en la foto vemos como en el directorio del cliente, aun no hay nada, primero se necesita el *get*)  
Las canciones descargadas por el servidor comparten carpeta.

```
def do_lista_canciones(self, line):
    '''Mostrar las canciones descargadas por el servidor'''
    if not self.activo:
        print('Necesitas estar conectado a una factoria')
        return None
    lista_canciones = self.cliente.scheduler.getSongList()
    if not lista_canciones:
        print('Ninguna canción descargada')
    else:
        for song in lista_canciones:
            print('\n%s\n' % song)
```

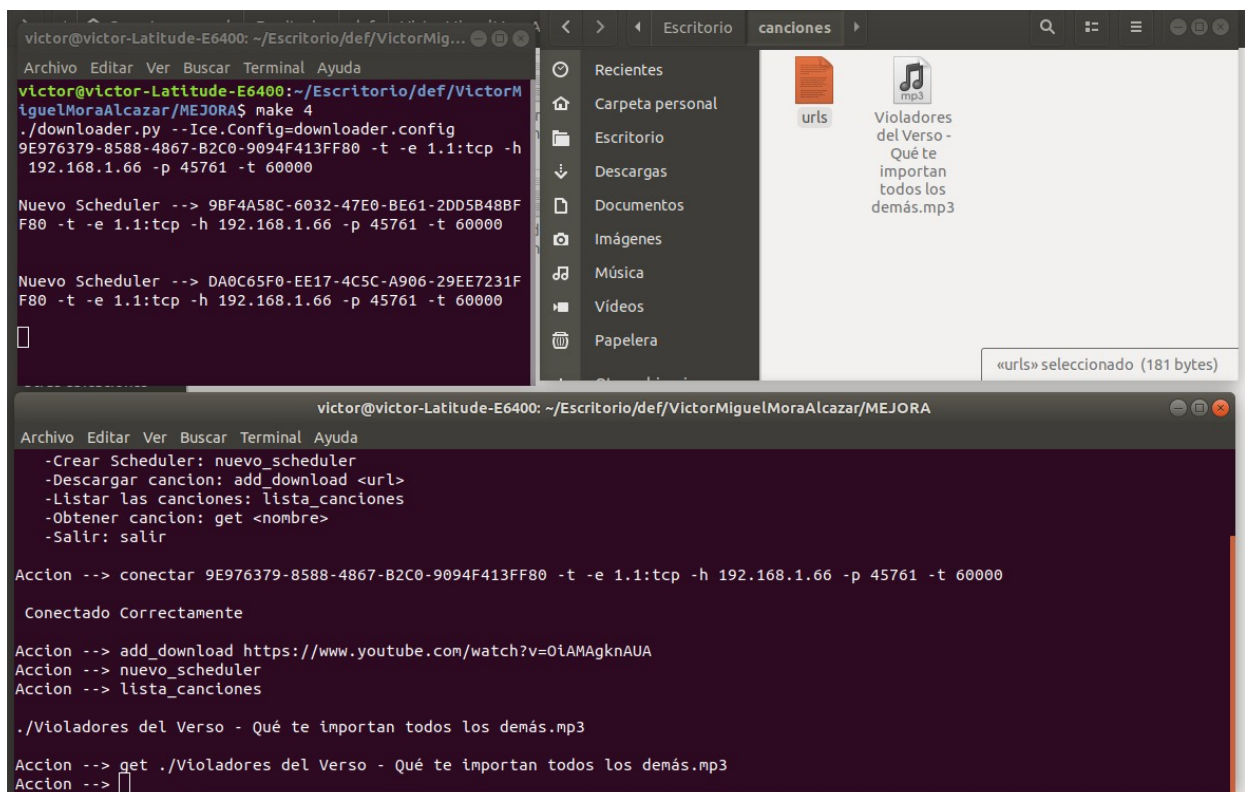
*Listar canciones disponibles*

```
def getSongList(self, current=None):
    ''' Obtener la lista de canciones '''
    return list(self.descargados)
```

*Funcion en downloader\_scheduler*

Tras comprobar que el cliente este conectado a la factoria, si no existen canciones descargadas, se informa al usuario con un mensaje, en el caso de existir canciones, se muestra una lista con las mismas (lista *descargados*)

## 7. Función get



```

def do_get(self, line):
    '''Obtener la canción descargada'''
    if not self.activo:
        print('Necesitas estar conectado a una factoria')
        return None
    self.cliente.get(line)

```

### Orden Shell

```

def get(self, cancion, destino='/home/victor/Escritorio/canciones'):
    '''
    Transferencia de ficheros entre el cliente y el servidor.
    Implementación copiada de transfer_snippet.py (ofrecido en el enunciado)
    Destino de los audios especificado. Fuentes consultadas:
    https://docs.python.org/3/library/os.path.html
    https://www.bogotobogo.com/python/python_files.php
    '''
    if cancion is not '':
        transfer = self.scheduler.get(cancion)
        with open(os.path.join(destino, cancion), "wb") as file_contents:
            remote_eof = False
            while not remote_eof:
                data = transfer.recv(BLOCK_SIZE)
                # Remove additional byte added by str() at server
                if len(data) > 1:
                    data = data[1:]
                data = binascii.a2b_base64(data)
                remote_eof = len(data) < BLOCK_SIZE
                if data:
                    file_contents.write(data)
            transfer.end()
    else:
        print('Sin canciones')

```

### Parte cliente

```

def get(self, filename, current=None):
    ''' Obtener el audio '''
    proxy = current.adapter.addWithUUID(TransferI(filename))
    return Downloader.TransferPrx.checkedCast(proxy)

```

```

class TransferI(Downloader.Transfer):
    '''
    Transferencia de ficheros entre cliente y servidor.
    Implementación copiada de transfer_snippet.py (ofrecido en el enunciado)
    '''
    def __init__(self, local_filename):
        self.file_contents = open(local_filename, 'rb')

    def recv(self, size, current=None):
        '''Send data block to client'''
        return str(
            binascii.b2a_base64(self.file_contents.read(size), newline=False)
        )

    def end(self, current=None):
        '''Close transfer and free objects'''
        self.file_contents.close()
        current.adapter.remove(current.id)

```

### Parte servidor

Primero como siempre, se comprueba si se esta conectado a la factoria correctamente. Con esta función, el cliente obtiene el audio (descargado por el servidor) en el directorio especificado. Para la transferencia de ficheros, se usa el código proporcionado en el enunciado (*transfer\_snippet.py*), situando cada parte del código en sus respectivos sitios. Si no se especifica ninguna canción, se avisa al cliente de ello.

## 8. Resumiendo

En la aplicación, se da soporte a las siguientes funciones:

- Conectarse a la factoria
- Crear schedulers (automaticamente y de forma manual)
- Descargar canción
- Listar las canciones descargadas
- Obtener la canción descargada por el servidor

Que no se ha implementado

- Extensión opcional
- Canales de eventos, planteados pero sin funcionar

**ALUMNO:** Victor Miguel Mora Alcázar (grupo C2)