

Automatizar 10 minutos al día

objetivo

a partir de las imágenes en local (descartado el sistema a través de FEEDLY, que da muchos «image not found»), automatizar el proceso de subida, importación, salvaguarda en la plataforma, sincronización con copia local, búsqueda de duplicados y eliminación de los mismos

1. copiar ficheros locales (estructura específica) a carpeta import en diezminutosaldia.com (scp)
2. realizar petición de importación de ficheros subidos a proceso CodeIgniter (wget)
3. mover ficheros remotos a import.old (eliminando los que hubiesen) para dejar apunto para próxima importación (script remoto)
4. eliminar los ficheros locales (no la estructura de directorios) ya importados (find)
1. sincronización entre ficheros remotos y ficheros locales (rsync)
2. búsqueda de duplicados (findimagedupes)
3. generación fichero .yumml (script)
4. generar HTML imágenes duplicadas (mustache + yumml)
5. mostrar HTML con duplicados (html con ajax que se comunica con plataforma 10minutosaldia.com)
6. eliminar ficheros temporales

método

el proceso está dividido en dos partes:

1. subida imágenes, importación a la BDD, eliminado imágenes subidas (en caso de no mostrar errores)
2. proceso de sincronización, búsqueda duplicados y mantenimiento de los mismos
 1. [findimagedupes10minutos](#)

```
alias 10minutosProcesoCompleto='~/myscripts/10minutos/10minutos-upload.sh &&
~/myscripts/10minutos/10minutos-dupes.sh'
```

1

[10minutos-upload.sh](#)

```
#!/bin/bash

#
#       Sincroniza fotos candidatas con plataforma 10m, realiza importación,
#       deja directorios de importación apunto para próximo uso
#

#
#       variables
#

L1_PATH="/home/mate/Baixades/10minutos"
R1_PATH="/home/diezminutos/import"
```

```
LOG_FILE="$L1_PATH/10minutos.log"

#
#      MAIN
#

#      sincroniza ficheros plataforma 10 minutos
#      (/home/mate/myscripts/10minutos-sync.sh)
echo -e "Copiando imágenes en diezminutos...\n"
scp -r -i $HOME/.ssh/diezminutos@bungalow.dreamhost.com $L1_PATH/*
diezminutos@bungalow.dreamhost.com:$R1_PATH

#      importar fotos
echo -e "Lanzando petición para importación...\n"
/usr/bin/wget --delete-after
http://10minutosaldia.com/maintenance/import_fotos
STATUS=$?

#      limpiando directorios remotos para próximo uso
#      comprobar petición OK
if [ $STATUS -eq 0 ]; then
    echo -e "Lanzando petición de limpieza del directorio...\n"
    ssh -v -i /home/mate/.ssh/diezminutos@bungalow.dreamhost.com
diezminutos@bungalow.dreamhost.com "~/10m-upload-and-update.sh"
    STATUS=$?
else
    echo -e "ERROR wget!\n"
    exit 1
fi

#      limpiar directorios locales
#      comprobar petición OK
if [ $STATUS -eq 0 ]; then
    echo -e "Limpiando directorios de imágenes locales...\n"
    find $L1_PATH/ -type f -delete;
    STATUS=$?
else
    echo -e "ERROR SSH!\n"
    exit 1
fi

if [ $STATUS -eq 0 ]; then
    echo -e "proceso finalizado felizmente\n"
else
    echo -e "Algún error final detectado\n"
    exit 1
fi
```

2

tecnologías usadas

- findimagedupes
- bash
- mustache
 - <https://en.wikipedia.org/wiki/FreeMarker> ?
- bootstrap
- CodeIgniter

objetivo y método

se pretende buscar las imágenes iguales o muy iguales de todas las disponibles en la plataforma 10 minutos al día para eliminar esos duplicados.

al no poder instalar el programa findimagedupes en la plataforma de Dreamhost, optamos por:

- hacer una copia de todas las fotos en local con la herramienta rsync (para agilizar las actualizaciones)
- buscar duplicados (o similares muy similares) con la herramienta findimagedupes
- del fichero TXT generado por findimagedupes, generar un fichero YUML con un formato específico para poder usar con mustache
- usar mustache para generar a partir de los datos de ficheros duplicados un HTML donde poder visualizar las diferentes imágenes detectadas como duplicadas y eliminar cuando convenga estas imágenes
- para la eliminación, hemos desarrollado en el controlador y modelo de la plataforma 10 minutos al día unas funciones específicas que se encargan de borrar de la BDD y del FS el fichero correspondiente
- queda por desarrollar la parte AJAX del proyecto, para que la eliminación sea asíncrona y así poder ir eliminando diferentes imágenes sin perder la situación.

uso

- al hacer un UPLOAD de imágenes, habrá que hacer un RSYNC con el repositorio local
 - las imágenes borradas de otras cribas desaparecerán con el RSYNC
- ejecutar el procedimiento de búsqueda de duplicados
 - hay que hacer que las imágenes que han sido borradas desaparezcan de la BDD de findimagedupes
 - <http://www.jhnc.org/findimagedupes/manpage.html>
 - `-purge + -f=`
- hacer la comprobación y eliminar para que no se envíen los duplicados detectados

implementación

- petición AJAX con JQuery + Bootstrap:
 - problemas para recoger los datos de la plataforma 10 minutos:
 - tips:
 - petición desde HTML:

```
var posting = $.post( {  
    url: URL10M+sha,  
    data: '',  
    dataType: 'jsonp'
```

```
}) ...
```

- devolución del server:

```
$callback = ( isset( $_GET['callback'] ) ? $_GET['callback'] : '' );  
$retorno = json_encode($retorno);  
header('Content-type: application/json; charset=utf-8');  
echo $callback."(".$retorno.)";  
exit();
```

- <https://stackoverflow.com/questions/6809053/simple-jquery-php-and-jsonp-example>

mejoras/problemas

- ~~¿qué pasa con las fotos que no son duplicados pero que findimagedupes detecta como tal?~~
 - subida sensibilidad de FINDIMAGEDUPES al 95%
- ¿qué pasa con las fotos existentes pero que no constan en la BDD?
 - automatizar eliminado ¿cómo?
 - reinsertar en la BDD si necesario?
 - ~~¿qué pasa si borro el duplicado que si que existe en la BDD, pero dejo una que no está en la BDD?~~
 - petición AJAX comprobación en BDD, previo a dejar borrar
- ~~automatizar proceso de rsync + findimagedupes + mustache~~
- ~~petición AJAX tropieza con protección CodeIgniter → no protección CI, petición AJAX, resuelto con 'jsonp'~~
 - «No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access. The response had HTTP status code 500.»

código

10minutos-dupes.sh

```
#!/bin/bash  
  
#  
# Sincroniza 10m-Local, busca duplicados, genera el TXT y el subsecuente YML  
# para generar los HTML a través de mustache  
#  
  
#  
# variables  
#  
  
PATH1="/media/mate/WD2-3TB-EXT4/10minutos"  
MAX_IMAGENES=100 # número de líneas de cada fichero YML (y por lo tanto  
HTML). Entero (1500 duplicados) no carga!  
OTHER_BIN="/home/mate/myscripts/10minutos"  
  
  
#  
# variables rsync  
#  
  
R_PATH="$~/uploads/"
```

```
L_PATH="/media/mate/WD2-3TB-EXT4/10minutos"
L2_PATH="$L_PATH/uploads/"
LOG_FILE="$L_PATH/10minutos.log"
BROWSER_PATH="/usr/bin/firefox"

#
# MAIN
#

# sincroniza ficheros plataforma 10 minutos (/home/mate/myscripts/10minutos-
sync.sh)
echo -e "Sincronizando...\n"
rsync -vrt --delete --exclude-from "$L_PATH/exclude.txt" -e "ssh -i
$HOME/.ssh/diezminutos@bungalow.dreamhost.com"
diezminutos@bungalow.dreamhost.com:$R_PATH $L2_PATH >> $LOG_FILE

# busca duplicados
echo -e "Dupes Searching...\n"
findimagedupes -t=95% --prune -f=$L_PATH/findimagedupes-10minutos.db
$L_PATH/uploads > $L_PATH/dupes.txt 2> /dev/null

# generar fichero .yuml a partir de duplicados
echo -e "Generando YUML...\n"
LINEAS=$(cat $L_PATH/dupes.txt | wc -l)
FICHEROS=$(( $LINEAS/$MAX_IMAGENES ))

if (( $FICHEROS > 0 )); then
    echo -e "Existen varios ficheros para procesar...\n"
    split -d --lines=$MAX_IMAGENES $L_PATH/dupes.txt $L_PATH/dupes_ --
additional-suffix=.txt
    for i in `seq 0 $FICHEROS` ;
    do
        if (( $i <= 9 )); then
            SUFIJO="0$i"
        else
            SUFIJO="$i"
        fi
        $OTHER_BIN/findimagedupes-YML-generator.sh $L_PATH/dupes_$SUFIJO.txt
    done
else
    $OTHER_BIN/findimagedupes-YML-generator.sh $L_PATH/dupes.txt
fi

# generar HTML con .yuml y .mustache
echo -e "Generando HTML con mustache...\n"

if (( $FICHEROS > 0 )); then
    split -d --lines=100 $L_PATH/dupes.txt $L_PATH/dupes_ --additional-
suffix=.txt
    for i in `seq 0 $FICHEROS` ;
    do
        if (( $i <= 9 )); then
            SUFIJO="0$i"
        else
            SUFIJO="$i"
        fi
    done
fi
```

```
        fi
        mustache $_PATH/dupes_${SUFIXO}.yml
$_PATH/findimagedupes-10minutos.mustache > $_PATH/10minutos-
findimagedupes_${SUFIXO}.html
    done
else
    mustache $_PATH/dupes.yml $_PATH/findimagedupes-10minutos.mustache >
$_PATH/10minutos-findimagedupes.html
fi

# ejecutar Chrome con resultados
$BROWSER_PATH --private-window file:///media/mate/WD2-3TB-
EXT4/10minutos/10minutos-findimagedupes.html 2>/dev/null

# eliminar ficheros temporales
if (( $FICHEROS > 0 )); then
    rm $_PATH/dupes_*. *
fi
```

findimagedupes-YML-generator.sh

```
#!/bin/bash

#
# Genera fichero YML a partir del TXT generado por findimagedupes
# Necesita un argumento: nombre_fichero.txt
#

#
# FUNCIONES
#

cabecera() {
cat > $OUTPUT << EOF
---
images: [
EOF
}

footer() {
cat >> $OUTPUT << EOF
]
---
EOF
}

#
# MAIN
#
```

```
filename="$1"
base_filename=${filename%%.*}
OUTPUT="$base_filename.yml"
LINEAS=0
IMAGENES=0

# llamada función cabecera
cabecera

while read -r linea
do
    LINEAS=$((LINEAS+1))
    # deseamos: part: [ [ url: "1.jpg", name: "name1" ], [url: "2.jpg", name:
    "name2" ] ],
    # inicio cadena
    echo "part: [ " >> $OUTPUT

    #genera un item por cada imagen, tantos como sean necesarios, en la cadena
    for i in ${linea[@]}
    do
        CRC=${i##*/}          # name ha de ser solamente el CRC, sin path!
        CRC=${CRC:0:-4}        # elimina la extensión
        echo "[ url: '$i', name: '$CRC' ]," >> $OUTPUT
        IMAGENES=$((IMAGENES+1))
    done

    # final cadena
    echo " ]," >> $OUTPUT
done < "$filename"

# llamada función footer
footer

echo -e "LINEAS: $LINEAS"
echo -e "IMAGENES: $IMAGENES"
<code bash>
```

From:
<https://miguelangel.torresegea.es/wiki/> - miguel angel torres egea

Permanent link:
<https://miguelangel.torresegea.es/wiki/misproyectos:automatizacion10minutos?rev=1517745693>

Last update: 04/02/2018 04:01

