Supprimer les vidéos, des enregistreurs, non revendiquées depuis un temps défini

Sur notre Pod les vidéos issues des enregistreurs sont revendiquées manuellement. On a crée ce script pour supprimer automatiquement les vidéos non revendiquées.

Le script récupère toutes vidéos de la table recorder_recordingfiletreatment. Si le délai de rétention est dépassé l'enregistrement est supprimé en base mais également le fichier de la vidéo.

Ajouter le délai de rétention dans le fichier de configuration custom/settings_local.py

RECORD_RETENTION = 30

Créer le fichier /pod/custom/management/commands/enregistrement.py

```
import os
import shutil
from django.core.management.base import BaseCommand
from django.conf import settings
from django.utils import translation, timezone
from pod.recorder.models import Recorder, Recording, RecordingFileTreatment
from datetime import timedelta, datetime
import logging

LANGUAGE_CODE = getattr(settings, "LANGUAGE_CODE", "fr")

RECORD_RETENTION = getattr(
    settings, "RECORD_RETENTION", "30")

DEFAULT_RECORDER_PATH = getattr(settings, "DEFAULT_RECORDER_PATH", "/data/ftp-pod/ftp/")

log = logging.getLogger(__name__)

def checkRecordingRetention():
    if RecordingFileTreatment.objects.all().count() > 0:
        enregistrements = RecordingFileTreatment.objects.all()
        for enregistrement in enregistrements:
            print(enregistrement.file)
            if retention_depasee(enregistrement):
                print("rétention dépassée")
                supprimer_enregistrement(enregistrement)

def supprimer_enregistrement(enregistrement):
    if os.path.isfile(enregistrement.file):
        log.info("SUPPRESSION DE L'ENREGISTREMENT %s" % enregistrement.file)
        os.remove(enregistrement.file)
        enregistrement.delete()

def retention_depasee(enregistrement):
    resultat = False
    dateretention = enregistrement.date_added + timedelta(days=int(RECORD_RETENTION))
    print(dateretention)
    if dateretention < timezone.now():
        resultat = True
    return resultat

class Command(BaseCommand):
    valid_args = ["checkRecordingRetention"]

    def add_arguments(self, parser):
        parser.add_argument("checkRecordingRetention", help="Supprime les enregistrements non revendiqués")

    def handle(self, *args, **options):
        translation.activate(LANGUAGE_CODE)
        if options["checkRecordingRetention"]:
            checkRecordingRetention()
```

Pour exécuter la commande manuellement

python manage.py enregistrement checkRecordingRetention

Pour exécuter la commande par cron

crontab -e

*/15 * * * * /usr/bin/bash -c 'export WORKON_HOME=/data/www/%userpod%/.virtualenvs; export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3.6; cd /data/www/%userpod%/django_projects/podv2; source /usr/bin/virtualenvwrapper.sh; workon django_pod; python manage.py enregistrement checkRecordingRetention'