Ein Python Script als Appimage bauen

Beschreibung:

Es gibt situationen, da möchte man gerne ein python script mit all seinen abhängigkeiten gerne als eine einzige Executable haben. Mit App image und env ist dies machbar. Es können sogar Parameter an das pythonscript übergeben werden.

Erstellung:

Verzeichnis erstellen

Als erstes unser Project VErzeichnis erstellen. MyApp natürlich mit dem namen des Projektes oder dateinamen des scriptes ersetzen. Ich nenne das prject nextcloudimport

```
mkdir MyApp.AppDir
```

Python environment erstellen und Abbhängigkeiten für das Script installieren

Nun in das Verzeichnis gehen und ein ENV erstellen. Ein ENV ist eine Virtualumegebung für python.

```
cd MyApp.AppDir
virtualenv --no-download AppRun
source AppRun/bin/activate

Beispiel:

cd nextcloudimport.AppDir
virtualenv --no-download python_env
source python_env/bin/activate
```

Nun alle abhängigkeiten installieren die unser script braucht

```
pip install --upgrade pip
pip install <Ihre benötigten Pakete>

Beispiel:
pip install --upgrade pip
pip install requests
pip install BeautifulSoup4
pip install tabulate
pip install qrcode
pip install reportlab
```

Nun die apprun wieder deaktivieren

```
deactivate
```

Script kopieren

Nun das eigentliche Script reinkopieren.
In diesem Fall kommt der importer von https://github.com/t-markmann/nc-userimporter

Die datei nc-userimporter.py ins AppDir Verzeichnis kopieren.

Launcher Script erstellen

```
nano AppRun
```

Inhalt:

```
#!/bin/bash
DIR="$(dirname "$(readlink -f "$0")")"
$DIR/python_env/bin/python $DIR/your_script.py "$@"



Beispiel:

#!/bin/bash
DIR="$(dirname "$(readlink -f "$0")")"
$DIR/python_env/bin/python $DIR/nc-userimporter.py "$@"
```

AppRun ausführbar machen

```
chmod +x AppRun
```

Icon und .desktop-Datei hinzufügen

Eine .desktop erstellen, die muss im auch im Hauptverzeichnsi von myapp.AppDIr liegen.
Wnn ein icon gewüncht ist eine png mit 32x32 pixeln, wir nehmen hier das Nextcloud icon Papirus-Team-Papirus-Apps-Nextcloud.32.png

diese dann in nextcloudimport.png umbenennen.

Für dien Schlüssel können hier diese Nachgeschlagen werden, welche zur Verfügung stehen : https://specifications.freedesktop.org/menu-spec/menu-spec-1.0.html#category-registry

Dies sind die gängigsten : `Utility`, `Development`, `Graphics`, `AudioVideo`

DerName ist auch gleichzeitg der Dateiname ohne Leerzeichnen

```
[Desktop Entry]
Type=Application
Name=MyApp
Icon=MyApp
Exec=AppRun
Categories=<Ihre Category>

Beispiel:

[Desktop Entry]
Type=Application
Name=Nextcloud CSV import
Icon=nextcloudimport
Exec=AppRun
Categories=Utility
```

Nun eine Verzeichnissebene zurück gehen und das appimage bauen.
Die Architektur davor angeben.

```
ARCH=i386  (32bit)
ARCH=x86_64  (64bit)
```

```
cd ..
ARCH=x86_64 ./appimagetool.AppImage MyApp.AppDir

Beispiel:
cd ..
ARCH=x86_64 ./appimagetool.AppImage
appimagetool nextcloudimport.AppDir
```

So sieht dann unsere Datei aus, mit Icon

Da dies ein Terminal python script ist was wir in ein appimage gesteckt haben probieren wir dieses jetzt im Terminal aus

Pfad von Dateien im Scriptverzeichnis außerhalb des AppImages verweisen wie config files etc.

Sollte das script auf dateien die im AppImage liegen aber eigentlich außerhalb des appimage änderbar sein sollen wie zum beispiel config Dateien.

Beispiel: das python script wenn es so laufen würde sucht in seinem pfad nach einer Datei config.xml
Wenn es so aufgerufen wird, also ohne ein Appimage können wir die datei ja einfach editieren.
In einem Appimge würde die config.xml ja im AppImage liegen.
Also müssten wir jedes mal das Appimage neubauen, wenn wir die config ändern.
Nicht sehr flexibel.

Deshalb ändern wir das Python script ab, wo es die config.xml finden soll.
In unserem Script Beispiel sind es zwei dateien eine config.xml und eine users.csv

Das Original script:

unser pythoc script komplett

```
#!/usr/bin/env python3
import os
import os.path
```

```
import sys
import time
import requests
import certifi
import csv
import string
import urllib.parse
import qrcode
import random
import codecs
import html
from reportlab.lib.enums import TA_JUSTIFY
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image, PageBreak
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from tabulate import tabulate
from bs4 import BeautifulSoup
from datetime import datetime

# This tool creates Nextcloud users from a CSV file, which you exported from some other
software.
# There is also an extra EduDocs mode, which takes into account the special default settings
# and security-related peculiarities when importing users in the school sector.

# Copyright (C) 2019-2020 Torsten Markmann
# Mail: info@uplinked.net
# WWW: edudocs.org uplinked.net

# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# any later version.

# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.

# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.

print("")
print("")
print("##################################################################
################################")
```

```
print("# NEXTCLOUD-USER-IMPORTER                                                    #")
print("#################################################################################")
print("")
print("Copyright (C) 2019-2020 Torsten Markmann (t-markmann), edudocs.org & uplinked.net")
print("Contributors: Johannes Schirge (Shen), Nicolas Stuhlfauth (nicostuhlfauth)")
print("This program comes with ABSOLUTELY NO WARRANTY")
print("This is free software, and you are welcome to redistribute it under certain conditions.")
print("For details look into LICENSE file (GNU GPLv3).")
print("")

# Useful resources for contributors:
# Nextcloud user API
https://docs.nextcloud.com/server/latest/admin_manual/configuration_user/instruction_set_for_users.html
# Nextcloud group API
https://docs.nextcloud.com/server/latest/admin_manual/configuration_user/instruction_set_for_groups.html
# CURL to Python request converter https://curl.trillworks.com/

# determine if running in a build package (frozen) or from seperate python script
frozen = 'not'
if getattr(sys, 'frozen', False):
  # we are running in a bundle
  appdir = os.path.dirname(os.path.abspath(sys.executable))
  ## print("Executable is in frozen state, appdir set to: " + appdir) # for debug
else:
  # we are running in a normal Python environment
  appdir = os.path.dirname(os.path.abspath(__file__))
  ## print("Executable is run in normal Python environment, appdir set to: " + appdir) # for debug

# read config from xml file
configfile = codecs.open(os.path.join(appdir,'config.xml'),mode='r', encoding='utf-8')
config = configfile.read()
configfile.close()

# load config values into variables
config_xmlsoup = BeautifulSoup(config, "html.parser") # parse
config_ncUrl = config_xmlsoup.find('cloudurl').string
config_adminname = config_xmlsoup.find('adminname').string
config_adminpass = urllib.parse.quote(config_xmlsoup.find('adminpass').string)
config_csvfile = config_xmlsoup.find('csvfile').string
config_csvDelimiter = config_xmlsoup.find('csvdelimiter').string
config_csvDelimiterGroups = config_xmlsoup.find('csvdelimitergroups').string
```

```
config_GeneratePassword = config_xmlsoup.find('generatepassword').string
config_sslVerify = eval(config_xmlsoup.find('sslverify').string)
config_language = config_xmlsoup.find('language').string
config_pdfOneDoc = config_xmlsoup.find('pdfonedoc').string
config_EduDocs = config_xmlsoup.find('edudocs').string
config_schoolgroup = config_xmlsoup.find('schoolgroup').string

# EduDocs info-text
if config_EduDocs == 'yes':
  print("")

print("#################################################################################")
  print("# EDUDOCS-MODUS (www.edudocs.org)                                             #")
  print("# Willkommen zum EduDocs-Nutzerimport.                                        #")
  print("# Dieser Modus ist für den Import schulischer Nutzeraccounts vorgesehen und   #")
  print("# berücksichtigt die besonderen datenschutzrechtlichen Vorgaben.              #")
  print("#                                                                             #")
  print("# Bitte stellen Sie sicher, dass die Import-Datei nur EINE Gruppe von Personen #")
  print("# enthält. Das heißt, entweder Lehrkräfte ODER Schüler ODER Schulpersonal.     #")
  print("# Ein zeitgleicher Import verschiedener Personengruppen ist nicht möglich.    #")
  print("#                                                                             #")
  print("# Prüfen Sie die Vorschau des Nutzerimports sehr genau, bevor Sie den         #")
  print("# Importprozess starten.                                                      #")

print("#################################################################################")
  print("")
  print("")
  print("Wenn Sie sicher sind, dass Ihre Einstellungen in der config.xml korrekt sind,")
  print("drücken Sie eine beliebige Taste, um fortzufahren.")
  input("Andernfalls brechen Sie den Prozess mit [STRG + C] ab.")
  print("")
  print("Sie haben sich entschieden, fortzufahren. Eine Nutzerimport-Vorschau wird generiert.")
  print("")

else:
  print("")

print("#################################################################################")
  print("# Welcome to the Nextcloud user import.                                       #")
  print("# Please check the preview of the user import very carefully before you start #")
  print("# the import process.                                                         #")
```

```
print("##################################################################################")
  print("")
  print("")
  print("When you are sure that your settings in the config.xml are correct,")
  print("press [ANY KEY] to continue.")
  input("Otherwise, press [CONTROL + C] to abort the process.")
  print("")
  print("They have decided to continue. A user import preview is generated.")
  print("")

# check if user-import-csv-filme exists
if not os.path.isfile(config_csvfile):
    if config_EduDocs == 'yes':
      print("FEHLER!")
      print("Die csv-Datei (" + config_csvfile + "), die Sie in der config.xml eingetragen haben,
existiert nicht. Bitte speichern Sie die Datei '" + config_csvfile + "' im Hauptverzeichnis des
Scripts oder bearbeiten Sie die config.xml")
      input("Drücken Sie eine beliebige Taste, um zu bestätigen und den Prozess zu beenden.")
    else:
      print("ERROR!")
      print("The csv-file (" + config_csvfile + ") you specified in you config.xml does not exist.
Please save '" + config_csvfile + "' in main-directory of the script or edit your config.xml")
      input("Press [ANY KEY] to confirm and end the process.")
    sys.exit(1)

# cut http and https from ncUrl, because people often just copy & paste including protocol
config_ncUrl = config_ncUrl.replace("http://", "")
config_ncUrl = config_ncUrl.replace("https://", "")

# TODO optional: read config from input() if config.xml empty
# print('Username of creator (admin?):')
# config_adminname = input()
# print('Password of creator:')
# config_adminpass = input()

config_protocol = "https" # use a secure connection!
config_apiUrl = "/ocs/v1.php/cloud/users" # nextcloud API path (users), might change in the
future
config_apiUrlGroups = "/ocs/v1.php/cloud/groups" # nextcloud API path (groups), might change
in the future

# Headers for CURL request, Nextcloud specific
requestheaders = {
  'OCS-APIRequest': 'true',
```

```
}

# set/create output-directory
output_dir = 'output'
if not os.path.exists(output_dir):
  os.makedirs(output_dir)

# set/create temporary-directory
tmp_dir = 'tmp'
if not os.path.exists(tmp_dir):
  os.makedirs(tmp_dir)

# adds date and time as string to variable
today = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

# Mapping for umlauts and and special characters. The listed umlauts and special characters
are automatically converted to a compatible spelling for the user name.
mapping = {
          ord(u"Ä"): u"Ae",
          ord(u"ä"): u"ae",
          ord(u"Ë"): u"E",
          ord(u"ë"): u"e",
          ord(u"Ï"): u"I",
          ord(u"ï"): u"i",
          ord(u"Ö"): u"Oe",
          ord(u"ö"): u"oe",
          ord(u"Ü"): u"Ue",
          ord(u"ü"): u"ue",
          ord(u"Ÿ"): u"Y",
          ord(u"ÿ"): u"y",
          ord(u"ß"): u"ss",
          ord(u"À"): u"A",
          ord(u"Á"): u"A",
          ord(u"Â"): u"A",
          ord(u"Ã"): u"A",
          ord(u"Å"): u"A",
          ord(u"Æ"): u"Ae",
          ord(u"Ç"): u"C",
          ord(u"È"): u"E",
          ord(u"É"): u"E",
          ord(u"Ê"): u"E",
          ord(u"Ì"): u"I",
          ord(u"Í"): u"I",
          ord(u"Î"): u"I",
          ord(u"Đ"): u"D",
```

```
ord(u"Ñ"): u"N",
ord(u"Ò"): u"O",
ord(u"Ó"): u"O",
ord(u"Ô"): u"O",
ord(u"Õ"): u"O",
ord(u"Ø"): u"Oe",
ord(u"Œ"): u"Oe",
ord(u"Ù"): u"U",
ord(u"Ú"): u"U",
ord(u"Û"): u"U",
ord(u"Ý"): u"Y",
ord(u"Þ"): u"Th",
ord(u"à"): u"a",
ord(u"á"): u"a",
ord(u"â"): u"a",
ord(u"ã"): u"a",
ord(u"å"): u"a",
ord(u"æ"): u"ae",
ord(u"ç"): u"c",
ord(u"è"): u"e",
ord(u"é"): u"e",
ord(u"ê"): u"e",
ord(u"ì"): u"i",
ord(u"í"): u"i",
ord(u"î"): u"i",
ord(u"ð"): u"d",
ord(u"ñ"): u"n",
ord(u"ò"): u"o",
ord(u"ó"): u"o",
ord(u"ô"): u"o",
ord(u"õ"): u"o",
ord(u"ø"): u"oe",
ord(u"œ"): u"oe",
ord(u"ù"): u"u",
ord(u"ú"): u"u",
ord(u"û"): u"u",
ord(u"ý"): u"y",
ord(u"þ"): u"Th",
ord(u"Š"): u"S",
ord(u"š"): u"s",
ord(u"Č"): u"C",
ord(u"č"): u"c"
}
```

```
# QR-Code class
qr = qrcode.QRCode(
    version=1,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=10,
    border=4,
)

# Function: Generate random password
# This will generate a random password with 1 random uppercase letter, 3 random lowercase letters,
# 3 random digits, and 1 random special character--this can be adjusted as needed.
# Then it combines each random character and creates a random order.
def pwgenerator():
  PWUPP = random.SystemRandom().choice(string.ascii_uppercase)
  PWLOW1 = random.SystemRandom().choice(string.ascii_lowercase)
  PWLOW2 = random.SystemRandom().choice(string.ascii_lowercase)
  PWLOW3 = random.SystemRandom().choice(string.ascii_lowercase)
  PWDIG1 = random.SystemRandom().choice(string.digits)
  PWDIG2 = random.SystemRandom().choice(string.digits)
  PWDIG3 = random.SystemRandom().choice(string.digits)
  PWSPEC = random.SystemRandom().choice('!@*(§')
  PWD = None
  PWD = PWUPP + PWLOW1 + PWLOW2 + PWLOW3 + PWDIG1 + PWDIG2 + PWDIG3 + PWSPEC
  PWD = ''.join(random.sample(PWD,len(PWD)))
  return(PWD)

# display expected results before executing CURL
  # display expected results for EduDocs-users
if config_EduDocs == 'yes':
  usertable = [["Nutzername","Anzeigename","Passwort","E-Mail","Gruppen","Gruppen-Admin für","Speicherplatz"]]
  with codecs.open(os.path.join(appdir, config_csvfile),mode='r', encoding='utf-8') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=config_csvDelimiter)
    next(readCSV, None)  # skip the headers
    for row in readCSV:
      if (len(row) != 7): # check if number of columns is consistent
        print("FEHLER: Die Zeiles des Nutzers",html.escape(row[0]),"hat",len(row),"Spalten. Es müssen 7 sein. Bitte korrigieren Sie dies in der csv-Datei.")
        input("Drücken Sie eine beliebige Taste, um den Prozess zu beenden.")
        sys.exit(1)
      pass_anon = html.escape(row[2])
      if len(pass_anon) > 0:
        pass_anon = "*" * (len(pass_anon)) # replace password for display on CLI
```

```
      line = html.escape(row[0])
      row[0] = line.translate(mapping) # convert special characters and umlauts
      if row[4]:
        grouplist = html.escape(row[4]).split(config_csvDelimiterGroups) # Groups in the CSV-file
are split by semicolon --> load into list
        if grouplist: # if grouplist contains group SchuelerInnen or Lehrkraefte, remove it
          if "SchuelerInnen" in grouplist:
            grouplist.remove('SchuelerInnen')
          if "Lehrkraefte" in grouplist:
            grouplist.remove('Lehrkraefte')
          grouplist.append(config_schoolgroup) # and add group which is set in config-file
(config_schoolgroup)
      if not config_schoolgroup == 'SchuelerInnen': # if you import students in an EduDocs-
instance, it is not possible to set groupadmins
        if row[5]:
          groupadminlist = html.escape(row[5]).split(config_csvDelimiterGroups) # Groupadmin
Values in the CSV-file are split by semicolon --> load into list
        else:
          groupadminlist = []
      else:
        groupadminlist = []
      currentuser =
[html.escape(row[0]),html.escape(row[1]),pass_anon,html.escape(row[3]),grouplist,groupadmin
list,html.escape(row[6])]
      usertable.append(currentuser)
  print(tabulate(usertable,headers="firstrow"))

  # ask EduDocs-user to check values and continue
  print("\nÜberprüfen Sie genau, ob die oben aufgeführten Nutzerdaten und
Gruppenzuordnungen korrekt sind.")
  if not config_GeneratePassword == 'yes':
    print ("ACHTUNG: Sie haben festgelegt, dass Nutzer, bei denen kein Passwort eingetragen
wurde, eine E-Mail erhalten, um sich selbst ein Passwort zu setzen. Bitte überprüfen Sie
unbedingt, dass bei jedem Nutzer, bei dem kein Passwort vorgegeben ist, eine korrekte E-
Mailadresse eingetragen ist!")
  print("Wenn alles korrekt ist, drücken Sie eine beliebige Taste, um den Import-Prozess zu
starten.")
  input("Wenn nicht, drücken Sie [Strg + C], um abzubrechen.")
  print("")
  print("\nSie haben den Import-Prozess gestartet. Die Nutzer und Gruppen werden nun
angelegt. Dies kann viel Zeit in Anspruch nehmen...\n")

  # display expected results for nextcloud-users
else:
  usertable = [["Username","Display name","Password","Email","Groups","Group admin
```

```
for","Quota"]]
  with codecs.open(os.path.join(appdir, config_csvfile),mode='r', encoding='utf-8') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=config_csvDelimiter)
    next(readCSV, None)  # skip the headers
    for row in readCSV:
      if (len(row) != 7): # check if number of columns is consistent
        print("ERROR: row for user",html.escape(row[0]),"has",len(row),"columns. Should be 7.
Please correct your csv-file.")
        input("Press [ANY KEY] to confirm and end the process.")
        sys.exit(1)
      pass_anon = html.escape(row[2])
      if len(pass_anon) > 0:
        pass_anon = "*" * (len(pass_anon)) # replace password for display on CLI
      line = html.escape(row[0])
      row[0] = line.translate(mapping) # convert special characters and umlauts
      currentuser =
[html.escape(row[0]),html.escape(row[1]),pass_anon,html.escape(row[3]),html.escape(row[4]),
html.escape(row[5]),html.escape(row[6])]
      usertable.append(currentuser)
  print(tabulate(usertable,headers="firstrow"))

  # ask user to check values and continue
  print("\nPlease check if the users and groups above are as expected and should be created
like that.")
  if not config_GeneratePassword == 'yes':
    print ("ATTENTION: You have specified that users for whom no password has been entered
will receive an e-mail to set a password for themselves. Please make absolutely sure that a
correct e-mail address is entered for every user for whom no password has been set!")
  input("If everything is fine, press [ANY KEY] to continue. If not, press [CONTROL + C] to
cancel.")
  print("\nYou confirmed. I will now create the users and groups. This can take a long time...\n")

# prepare pdf-output (if pdfOneDoc == yes)
if config_pdfOneDoc == 'yes':
  if config_EduDocs == 'yes':
    output_filename = config_schoolgroup + "_" + today + ".pdf"
  else:
    output_filename = "userlist_" + today + ".pdf"
  output_filepath = os.path.join( output_dir, output_filename )
  doc = SimpleDocTemplate(output_filepath,pagesize=A4,
                          rightMargin=72,leftMargin=72,
                          topMargin=72,bottomMargin=18)
# prepare pdf-content
Story=[]
```

```
# read rows from CSV file
with codecs.open(os.path.join(appdir, config_csvfile),mode='r', encoding='utf-8') as csvfile:
  readCSV = csv.reader(csvfile, delimiter=config_csvDelimiter)
  next(readCSV, None)  # skip the headers
  for row in readCSV:
    line = html.escape(row[0])
    row[0] = line.translate(mapping) # convert special characters and umlauts
    if config_GeneratePassword == 'yes':
      if not row[2]:
        row[2] = pwgenerator()
    if config_EduDocs == 'yes':
      if row[4]:
        grouplist = html.escape(row[4]).split(config_csvDelimiterGroups) # Groups in the CSV-file
are split by semicolon --> load into list
        if grouplist: # if grouplist contains group SchuelerInnen or Lehrkraefte, remove it
          if "SchuelerInnen" in grouplist:
            grouplist.remove('SchuelerInnen')
          if "Lehrkraefte" in grouplist:
            grouplist.remove('Lehrkraefte')
          grouplist.append(config_schoolgroup) # and add group which is set in config-file
(config_schoolgroup)
      if not config_schoolgroup == 'SchuelerInnen': # if you import students in an EduDocs-
instance, it is not possible to set groupadmins
        if row[5]:
          groupadminlist = html.escape(row[5]).split(config_csvDelimiterGroups) # Groupadmin
Values in the CSV-file are split by semicolon --> load into list
        else:
          groupadminlist = []
      else:
        groupadminlist = []
      print("Nutzername:",html.escape(row[0]),"| Anzeigename:",html.escape(row[1]),"|
Passwort: ","*" * len(row[2]) +
      "| E-Mail:",html.escape(row[3]),"| Gruppen:",grouplist,"| Gruppen-Admin
für:",groupadminlist,"| Speicherplatz:",html.escape(row[6]),)
    else:
      print("Username:",html.escape(row[0]),"| Display name:",html.escape(row[1]),"| Password:
","*" * len(row[2]) +
    "| Email:",html.escape(row[3]),"| Groups:",html.escape(row[4]),"| Group admin
for:",html.escape(row[5]),"| Quota:",html.escape(row[6]),)
    # build the dataset for the request
    data = [
      ('userid', html.escape(row[0])),
      ('displayName', html.escape(row[1])),
      ('password', html.escape(row[2])),
```

```
      ('email', html.escape(row[3])),
      ('quota', html.escape(row[6])),
      ('language', config_language)
    ]

    # if value exists: append single groups to data array/list for CURL
    if not config_EduDocs == 'yes':
      if row[4]:
        grouplist = html.escape(row[4]).split(config_csvDelimiterGroups) # Groups in the CSV-file
are split by semicolon --> load into list
      else:
        grouplist = []
    # check if group exists
    for group in grouplist:
     try:
       groupresponse = requests.get(config_protocol + '://' + config_adminname + ':' +
config_adminpass + '@' +
         config_ncUrl + config_apiUrlGroups + '?search=' + group.strip(),
headers=requestheaders, verify=config_sslVerify)
     except requests.exceptions.RequestException as e:  # handling errors
       print(e)
       print("The CURL request could not be performed.")
       input("Press [ANY KEY] to confirm and end the process.")
       sys.exit(1)

     response_xmlsoup = BeautifulSoup(groupresponse.text, "html.parser")

    # if group does not exists, create group
     if not response_xmlsoup.find('element'):
       try:
         groupdata = {
           'groupid':group.strip()
         }
         groupresponse = requests.post(config_protocol + '://' + config_adminname + ':' +
config_adminpass + '@' +
           config_ncUrl + config_apiUrlGroups, headers=requestheaders, data=groupdata,
verify=config_sslVerify)
       except requests.exceptions.RequestException as e:  # handling errors
         print(e)
         print("The CURL request could not be performed.")
         input("Press [ANY KEY] to confirm and end the process.")
         sys.exit(1)
       response_xmlsoup = BeautifulSoup(groupresponse.text, "html.parser")
```

```
      # catch wrong config (create group)
       if groupresponse.status_code != 200:
        print("HTTP Status: " + str(groupresponse.status_code))
        print("Your config.xml is wrong or your cloud is not reachable.")
        input("Press [ANY KEY] to confirm and end the process.")
        sys.exit(1)

      # show detailed info of response (create group)
       response_xmlsoup = BeautifulSoup(groupresponse.text, "html.parser")
       print('Create group "' + group.strip() + '": ' + response_xmlsoup.find('status').string + ' ' +
response_xmlsoup.find('statuscode').string +
        ' = ' + response_xmlsoup.find('message').string)

     data.append(('groups[]', group.strip())) # groups is parameter NC API

    # if value exists: append group admin values to data array/list for CURL
    if not config_EduDocs == 'yes': # if you import students in an EduDocs-Instance, it is not
possible to set groupadmins
      if row[5]:
        groupadminlist = html.escape(row[5]).split(config_csvDelimiterGroups) # Groupadmin
Values in the CSV-file are split by semicolon --> load into list
        try:
          for groupadmin in groupadminlist:
            data.append(('subadmin[]', groupadmin.strip())) # subadmin is parameter NC API
        except NameError:
          print("groupadminlist is not defined")

    # perform the request
    try:
      response = requests.post(config_protocol + '://' + config_adminname + ':' +
config_adminpass + '@' +
        config_ncUrl + config_apiUrl, headers=requestheaders, data=data,
verify=config_sslVerify)
    except requests.exceptions.RequestException as e:  # handling errors
      print(e)
      print("The CURL request could not be performed.")
      input("Press [ANY KEY] to confirm and end the process.")
      sys.exit(1)

    # catch wrong config
    if response.status_code != 200:
      print("HTTP Status: " + str(response.status_code))
      print("Your config.xml is wrong or your cloud is not reachable.")
      input("Press [ANY KEY] to confirm and end the process.")
      sys.exit(1)
```

```
    # show detailed info of response
    response_xmlsoup = BeautifulSoup(response.text, "html.parser")
    print(response_xmlsoup.find('status').string + ' ' +
response_xmlsoup.find('statuscode').string +
      ' = ' + response_xmlsoup.find('message').string)

    # append detailed response to logfile in output-folder
    logfile = codecs.open(os.path.join(output_dir,'output.log'),mode='a', encoding='utf-8')
    logfile.write("\nUSER: " + html.escape(row[0]) + "\nTIME: " + time.strftime("%d.%m.%Y
%H:%M:%S",time.localtime(time.time())) +
      "\nRESPONSE: " + response_xmlsoup.find('status').string + ' ' +
response_xmlsoup.find('statuscode').string +
      ' = ' + response_xmlsoup.find('message').string + "\n")
    logfile.close()

    # A QR code and a PDF file are only generated if the user has been successfully created.

    if response_xmlsoup.find('statuscode').string == "100":
      # generate qr-code
      qr.add_data("nc://login/user:" + html.escape(row[0]) + "&password:" + html.escape(row[2])
+ "&server:https://" + config_ncUrl)
      img = qr.make_image(fill_color="black", back_color="white")
      img.save(os.path.join( tmp_dir, html.escape(row[0]) + ".jpg" ))
      qr.clear()

      # prepare pdf-output (if pdfOneDoc == no)
      if config_pdfOneDoc == 'no':
        if config_EduDocs == 'yes':
          output_filename = config_schoolgroup + "_" + html.escape(row[0]) + "_" + today +
".pdf"
        else:
          output_filename = html.escape(row[0]) + "_" + today + ".pdf"

        output_filepath = os.path.join( output_dir, output_filename )

        doc = SimpleDocTemplate(output_filepath,pagesize=A4,
                                rightMargin=72,leftMargin=72,
                                topMargin=52,bottomMargin=18)

      if config_EduDocs == 'yes':
        nclogo = "assets/EduDocs_Logo.jpg" # EduDocs-logo (if in EduDocs-mode)
      else:
        nclogo = "assets/Nextcloud_Logo.jpg" # nextcloud-logo (if in normal mode)
      ncuserlogin = html.escape(row[0]) # loginname
      ncusername = html.escape(row[1]) # username
```

```
ncpassword = html.escape(row[2]) # password
nclink = config_protocol + "://" + config_ncUrl # adds nextcloud-url
  # adds nextcloud-logo to pdf-file
if config_EduDocs == 'yes':
  im = Image(nclogo, 150, 87)
else:
  im = Image(nclogo, 150, 106)
Story.append(im)
Story.append(Spacer(1, 12))

styles=getSampleStyleSheet()
styles.add(ParagraphStyle(name='Justify', alignment=TA_JUSTIFY))
# adds text to pdf-file
if config_EduDocs == 'yes':
  ptext = '<font size=14>Hallo %s,</font>' % ncusername
else:
  ptext = '<font size=14>Hello %s,</font>' % ncusername
Story.append(Paragraph(ptext, styles["Justify"]))
Story.append(Spacer(1, 12))

if config_EduDocs == 'yes':
  if config_schoolgroup == 'SchuelerInnen':
    ptext = '<font size=14>Für dich wurde ein Edu-Docs-Account angelegt.</font>'
  else:
    ptext = '<font size=14>Für Sie wurde ein Edu-Docs-Account angelegt.</font>'
else:
  ptext = '<font size=14>a Nextcloud-account has been generated for you.</font>'
Story.append(Paragraph(ptext, styles["Justify"]))
Story.append(Spacer(1, 12))

if config_EduDocs == 'yes':
  if config_schoolgroup == 'SchuelerInnen':
    ptext = '<font size=14>Du kannst dich mit folgenden Nutzerdaten einloggen:</font>'
  else:
    ptext = '<font size=14>Sie können sich mit folgenden Nutzerdaten einloggen:</font>'
else:
  ptext = '<font size=14>You can login with the following user data:</font>'
Story.append(Paragraph(ptext, styles["Normal"]))
Story.append(Spacer(1, 36))

if config_EduDocs == 'yes':
  if config_schoolgroup == 'SchuelerInnen':
    ptext = '<font size=14>Link zu deiner EduDocs-Instanz:</font>'
  else:
    ptext = '<font size=14>Link zu Ihrer EduDocs-Instanz:</font>'
```

```
    else:
      ptext = '<font size=14>Link to your Nextcloud:</font>'
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 12))

    ptext = '<font size=14>%s</font>' % nclink
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 24))

    if config_EduDocs == 'yes':
      ptext = '<font size=14>Nutzername:</font>'
    else:
      ptext = '<font size=14>Username:</font>'
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 12))

    ptext = '<font size=14>%s</font>' % ncuserlogin
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 24))

    if config_EduDocs == 'yes':
      ptext = '<font size=14>Passwort:</font>'
    else:
      ptext = '<font size=14>Password:</font>'
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 12))

    ptext = '<font size=14>%s</font>' % ncpassword
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 24))

    if config_EduDocs == 'yes':
      if config_schoolgroup == 'SchuelerInnen':
        ptext = '<font size=14>Alternativ kannst du mithilfe der Nextcloud-App folgenden QR-
Code scannen:</font>'
      else:
        ptext = '<font size=14>Alternativ können Sie mithilfe der Nextcloud-App folgenden QR-
Code scannen:</font>'
    else:
      ptext = '<font size=14>Alternatively, you can scan the following QR-Code in the
Nextcloud app:</font>'
    Story.append(Paragraph(ptext, styles["Normal"]))
    Story.append(Spacer(1, 24))
    # adds qr-code to pdf-file
    im2 = Image(os.path.join( tmp_dir, html.escape(row[0]) + ".jpg" ), 200, 200)
```

```
      Story.append(im2)
      del im2
      if config_pdfOneDoc == 'no':
        # create pdf-file (single documents)
        doc.build(Story)
      else:
        Story.append(PageBreak())
        # create pdf-file (one document)
if config_pdfOneDoc == 'yes':
  doc.build(Story)

# Clean up tmp-folder
filelist = [ f for f in os.listdir(tmp_dir) ]
for f in filelist:
    os.remove(os.path.join(tmp_dir, f))

if config_EduDocs == 'yes':
  print("")

print("#################################################################################")
  print("# Kontrollieren Sie den Status-Code der Nutzergenerierung oben oder in          #")
  print("# der output.log-Datei im Verzeichnis des Import-Scripts.                       #")
  print("# Die erfolgreich importierten Nutzer sollten zudem nun in Ihrer EduDocs-Instanz  #")
  print("# zu sehen sein.                                                               #")
  print("#                                                                              #")
  print("# Es wurde für jeden Nutzer eine PDF-Seite mit Infos zur Anmeldung generiert.    #")
  print("#                                                                              #")
  print("# Löschen Sie zu Ihrer Sicherheit unbedingt die Zugangsdaten aus der config.xml.  #")

print("#################################################################################")
  print("")
  input("Drücken Sie eine beliebige Taste, um den Prozess zu beenden.")
else:
  print("")

print("#################################################################################")
  print("# Control the status codes of the user creation above or in the output.log.      #")
  print("# You should as well see the users in your Nextcloud now.                       #")
  print("#                                                                              #")
  print("# A PDF-File with login-info and qr-code has been generated for every user.      #")
  print("#                                                                              #")
  print("# For security reasons: please delete your credentials from config.xml          #")
```

```
print("##################################################################################")
  print("")
  input("Press [ANY KEY] to confirm and end the process.")
```

---

stelle raussuchen wo die config.xml uns die users.csv deklariert sind

```
# read config from xml file
configfile = codecs.open(os.path.join(appdir,'config.xml'),mode='r', encoding='utf-8')
config = configfile.read()
configfile.close()
...

ändern in



...
#wichtig ganz am anfang des script wenn nicht schon vorhanden import os angeben
# Der Pfad zur AppImage-Datei
appimage_path = os.environ.get("APPIMAGE", "")

# Das Verzeichnis, aus dem die AppImage-Datei ausgeführt wird
appimage_dir = os.path.dirname(appimage_path)

# read config from xml file
configfile = codecs.open(os.path.join(appimage_dir, "config.xml"),mode='r', encoding='utf-8')
config = configfile.read()
configfile.close()
...

In diesem Python script wird die CSV Datei bzw der Pfad dazu aus der config.xml geladen
```

Aber generell gillt für jedes script:

```
#wichtig ganz am anfang des script wenn nicht schon vorhanden import os angeben
# Der Pfad zur AppImage-Datei
appimage_path = os.environ.get("APPIMAGE", "")
```

```
# Das Verzeichnis, aus dem die AppImage-Datei ausgeführt wird
appimage_dir = os.path.dirname(appimage_path)

# der neue os.path teil:
#wenn der Dateiname als string übergeben wird
os.path.join(appimage_dir, "config.xml")
wenn der Dateiname als variable übergeben wird
os.path.join(appimage_dir, configfile)
```

Version #4
Erstellt: 4 Oktober 2023 19:54:18 von Admin
Zuletzt aktualisiert: 5 Oktober 2023 09:12:51 von Admin