

<b>2026</b>	<b>PROJET SIO JANVIER</b>
<b>BTS SIO 1 SIOA Groupe A</b>	Auteurs : BALAN—HAYASHI Aïto & Orieux Adrien
	Date de rédaction : 05/01/2026
	Groupe : 05

## Compte-rendu d'étape

### Objectif de la phase

L'objectif de cette phase est de renforcer la sécurité des comptes en mettant en place une génération automatique de mots de passe conformes à des règles strictes. L'idée est d'éviter les mots de passe faibles ou générés manuellement, tout en produisant un fichier CSV exploitable pour la suite du projet.

### Méthode de génération des mots de passe

- **Principe général**

Les mots de passe sont générés automatiquement en respectant une structure précise. Ils ne contiennent ni espace, ni lettre **O** majuscule, ni chiffre **0**, afin d'éviter toute confusion visuelle. Le résultat final est exporté dans un fichier CSV contenant une seule colonne nommée *password*.

- **Règles appliquées**

Chaque mot de passe respecte les contraintes suivantes :

- longueur minimale de 12 caractères ;
- au moins 2 majuscules, 2 minuscules, 2 chiffres et 2 caractères spéciaux ;
- caractères spéciaux autorisés limités à une liste définie ;
- aucun caractère spécial en début ou en fin ;
- aucun doublon dans le fichier généré.

Pour garantir la robustesse, une vérification simple est également appliquée afin d'éviter les mots trop proches de mots courants (français ou anglais).

### Logique de construction

Le script commence par sélectionner le nombre minimum de caractères requis pour chaque catégorie (majuscules, minuscules, chiffres et spéciaux). Les caractères restants sont ensuite ajoutés de manière aléatoire à partir des ensembles autorisés.

Une fois le mot de passe construit, les caractères sont mélangés afin d'éviter toute structure prévisible, puis les contraintes de position et de validité sont vérifiées. Si une règle n'est pas respectée, le mot de passe est rejeté et régénéré.

### Vérification des mots de passe

Un programme de contrôle permet de vérifier automatiquement chaque mot de passe présent dans le fichier CSV. Pour chaque ligne, le script teste la longueur, la présence de caractères interdits, le respect des quotas et la position des caractères spéciaux.

Les doublons sont également détectés afin de garantir l'unicité des mots de passe.

### Résultats et livrables

Deux fichiers sont générés :

- un fichier *validation\_report.csv* indiquant, pour chaque mot de passe, s'il est valide ou non et les éventuelles erreurs ;
- un fichier *validation\_summary.txt* présentant un résumé global (nombre total de mots de passe, valides, invalides et doublons).

Cette phase permet de s'assurer que tous les mots de passe générés sont sécurisés, conformes aux règles et prêts à être utilisés.

```
password
tY5qf59DtN
nR4ed97CsP
cS1ff190pF
uF5ss75FkD
bG2cs24LnR
zD6ug31CgF
jE2ee62VmD
iM6bu96RjT
mL1vs92EjZ
dY8ve12XmU
gI9cl83GbU
uX1jr85L1Z
eD9nc31Ntr
hZ8xg48NaZ
eL6ma81GqI
lE5nr11TuH
oE3tm17AiH
lF2vd86YbA
oB7cx16BpB
mN5hq67ZnU
iW5oc43EvW
oH2hg75Hpx
hF6xe33NnC
uC3sd66ExF
t58ac95CgK
aI5mw740gK
iM6aj11JaV
wH3dl195nx
mY1tu83JdZ
```

```
environment.py
# environment.py
# This script generates a random password of length 12
# including uppercase, lowercase, numbers, and symbols.
# It also creates an environs.env file from environs.csv

import string
import random
import csv
from random import choice, randint, sample

# Define lists of characters
UPPERCASE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
LOWERCASE = "abcdefghijklmnopqrstuvwxyz"
NUMBERS = "0123456789"
SPECIALS = "!@#$%^&*()_+-=[]{};:,.<>?`~`"

# Define CSV file
CSV_FILE = "environs.csv"

# Define output file
ENV_FILE = "environs.env"

# Function to generate a random password
def generate_password(length=12):
    if length < 12:
        length = 12
    attempts = 0
    candidate = []
    while attempts < 1000:
        attempts += 1
        candidate.append(choice(UPPERCASE))
        candidate.append(choice(LOWERCASE))
        candidate.append(choice(NUMBERS))
        candidate.append(choice(SPECIALS))
        if len(candidate) > length - 10:
            candidate.pop()
        else:
            candidate.append(choice(UPPERCASE))
            candidate.append(choice(LOWERCASE))
            candidate.append(choice(NUMBERS))
            candidate.append(choice(SPECIALS))
        if len(candidate) == length:
            break
    random.shuffle(candidate)
    password = ''.join(candidate)
    if all(char in string.printable and char != ' ' for char in password):
        return password
    else:
        return generate_password(length)

# Function to generate a password with length 12
def create_password_12(n, length=12):
    env = {}
    attempts = 0
    while attempts < n and attempts < 1000:
        attempts += 1
        env['password'] = generate_password(length)
        if env['password'] != None:
            return env['password']
        else:
            attempts += 1
    return env

# Main function
if __name__ == "__main__":
    if len(sys.argv) > 2:
        env = create_password_12(int(sys.argv[1]), int(sys.argv[2]))
        print(env)
    else:
        env = create_password_12(1, 12)
        print(env)

environs.py
# environs.py
# This script reads environs.csv and writes its contents to environs.env
# environs.csv is a CSV file with columns 'variable' and 'value'
# environs.env is an environment variable file where each row is 'variable=value'

# Read CSV file
with open(CSV_FILE, 'r') as f:
    reader = csv.reader(f)
    envs = [row for row in reader]
    envs.pop(0) # Remove header

# Write to environs.env
with open(ENV_FILE, 'w') as f:
    for env in envs:
        f.write(f'{env[0]}={env[1]}\n')

print(f"Generated {len(envs)} environment variables to {ENV_FILE}!")
```