# Réseaux pour ingénieurs
# GLO-2000

# TP4 : Serveur de courriels

**Professeur responsable :**
Ronald Beaubrun
Ronald.Beaubrun@ift.ulaval.ca

**Responsables des travaux pratiques :**
Pierre Wendling et Philippe Dessaints
GLO2000.laboratoires@gmail.com

Université Laval
Faculté des sciences et de génie
Automne 2021

Ce dernier TP a pour but de consolider vos acquis avec la programmation socket, avec la transmission de structures de données et la gestion simultanée de plusieurs clients.

# Modalités de remise

- À faire en équipe de 3 (2 en cas d'un abandon).
- Les fichiers à remettre sont :
  - Le fichier « TP4_server.py ».
  - Le fichier « TP4_client.py ».
  - Les deux fichiers sont à rendre dans une unique archive zip ou tar nommé : « TP4_equipe_*<votre numéro d'équipe>* ».
  - Notez qu'aucun des modules fourni n'est à inclure dans l'archive.

N. B. : Une pénalité de 5% sera appliquée pour chaque point non respecté.

- Remise uniquement via le portail des cours.
- Date limite de remise est la suivante : **1 décembre 2021 à 23h00 (fuseau horaire de Québec)**.
- Tout travail remis en retard se verra attribuer la note **0**.
- Vous devez impérativement suivre les modèles TP4_server.py et TP4_client.py qui vous sont fournis.

# Mise en situation

Vous avez été mandaté par une compagnie afin de leur concevoir un système de courriel interne (*@glo-2000.ca*). Le système doit être composé d'un client et d'un serveur. Le serveur sera un relais SMTP qui traitera les envois de courriels à l'externe et à l'interne. Les utilisateurs utiliseront le client afin d'envoyer et de consulter leurs courriels.

# Fonctionnement du programme

Cette section décrit deux scénarios d'utilisations du programme.

## Création d'un compte et envoi d'un courriel

- L'utilisateur démarre le client et arrive sur le menu de connexion :

```
1. Créer un compte
2. Se connecter
```

- Il entre ”1” pour se créer un compte.
- Il entre un nom d'utilisateur suivi d'un mot de passe :

```
Entrez votre nom d'utilisateur : Foo.Bar.1
Entrez votre mot de passe :
```

- Malheureusement, le serveur rejette le mot de passe, car il n'est pas assez robuste et l'indique au client. Le client retourne alors sur le menu de connexion après avoir affiché le message d'erreur.
- L'utilisateur entre à nouveau ”1” et crée un compte avec succès.
- L'utilisateur arrive alors sur le menu principal :

```
Menu principal
1. Consultation de courriels
2. Envoi de courriels
3. Statistiques
4. Quitter
```

- L'utilisateur entre ”2” afin d'envoyer un courriel.
- L'utilisateur est alors invité à entrer l'adresse courriel du destinataire, un sujet et un corps. Il termine la saisie en entrant un point seul sur une ligne.

```
Entrez l'adresse de destination : John.Doe@glo-2000.ca
Entrez le sujet : Exemple
Entrez le message, terminez la saisie avec '.' sur une ligne.
Ceci est le corps du message.
Il peut contenir plusieurs lignes.

Y compris des lignes vides.
.
Envoi du message...
```

- Le client transmet le message au serveur. Notez que la dernière ligne contenant le point seul n'est pas incluse dans le message transmis.
- Le serveur indique au client si l'envoi s'est déroulé avec succès. Le client affiche ce résultat à l'utilisateur :

```
L'envoi s'est déroulé avec succès.
```

- L'utilisateur est renvoyé sur le menu principal.
- Il entre ”4” afin de terminer le programme.

## Connexion, consultation de courriels et de statistiques

- L'utilisateur démarre le client et arrive sur le menu de connexion :

```
1. Créer un compte
2. Se connecter
```

- Il entre "2" pour se connecter à son compte.
- Il entre un nom d'utilisateur suivi d'un mot de passe :

```
Entrez votre nom d'utilisateur : Jon.Doe
Entrez votre mot de passe :
```

- Malheureusement, l'utilisateur a commis une erreur en entrant son nom d'utilisateur, le serveur rejette la connexion et l'indique au client. Le client retourne alors sur le menu de connexion après avoir affiché le message d'erreur.
- L'utilisateur entre à nouveau "2" et se connecte avec succès.
- L'utilisateur arrive alors sur le menu principal :

```
Menu principal
1. Consultation de courriels
2. Envoi de courriels
3. Statistiques
4. Quitter
```

- L'utilisateur entre "1" afin de consulter son courriel.
- Le client reçoit du serveur une liste de sujets et les affiche à l'utilisateur :

```
n°1 Bienvenue - admin.2@glo-2000.ca
n°2 Exemple - Foo.Bar.1@glo-2000.ca
```

- L'utilisateur entre "2" afin de consulter le courriel de Foo.Bar.1.
- Le client reçoit du serveur le courriel et l'affiche à l'utilisateur :

```
De : Foo.Bar.1@glo-2000.ca
À : John.Doe@glo-2000.ca
Sujet : Exemple
---------------------------------------
Ceci est le corps du message.
Il peut contenir plusieurs lignes.

Y compris des lignes vides.
```

- L'utilisateur est renvoyé au menu principal.
- Il entre "3" afin de consulter ses statistiques.

- Le client reçoit du serveur les statistiques de l'utilisateur et les affiche :

```
Nombre de messages : 2
Taille du dossier : 4592 octets
```

- L'utilisateur est renvoyé sur le menu principal.
- Il entre "4" afin de terminer le programme.

# Modules à utiliser

Vous aurez à utiliser plusieurs modules dans le cadre de ce travail pratique. Cette section vous présente ceux qui sont nouveaux.

## Modules standards

Ces modules sont inclus avec les versions de Python disponibles sur le site officiel. Assurez-vous d'utiliser une version récente de Python 3.

- Vous pouvez hacher une séquence d'octets avec les fonctions du module hashlib :

```python
import hashlib

plain_text: str = "Une chaine de caractère à hacher."
hashed_text: str = hashlib.sha256(plain_text.encode()).hexdigest()
```

- Le module os permet de manipuler les chemins de fichiers indépendamment du système d'exploitation :

```python
import os

documents_dir_path: str = os.getcwd() + os.sep + "Documents"
file_list: list[str] = os.listdir(documents_dir_path)
first_file_name: str = file_list[0]
first_file_path: str = os.path.join(document_dir_path, first_file_name)
first_file_size: int = os.path.getsize(first_file_path)
```

- Le module getpass permet de récupérer un mot de passe sans l'afficher dans le terminal :

```python
import getpass

password = getpass.getpass("Entrez votre mot de passe : ")
```

- Vous pouvez également utiliser les modules présentés lors des séances précédentes tels que : email, json, re, socket, select, …

### Modules du TP4

Deux modules vous sont fournis et sont à utiliser pour la correction automatique :

- Le module glosocket est le même que celui du TP3. Vous devez l'utiliser pour toutes vos communications entre le client et le serveur.
- Le module TP4_utils vous fournit des constantes, gabarits, une énumération et une classe d'annotations :

```python
import TP4_utils

# Utilisation d'un gabarit
TP4_utils.EMAIL_DISPLAY.format(
    source="foo@glo-2000.ca",
    destination="bar@glo-2000.ca",
    subject="Exemple",
    content="Corps du message"
)

# Création d'un dictionnaire conforme à la classe GLO_message
message = TP4_utils.GLO_message(
    header=TP4_utils.message_header.OK,
    data={"hello": "world", "number": 42}
)
```

- **Notez que les tests unitaires requièrent l'utilisation de ces constantes, gabarits et de l'énumération**. Il n'est pas nécessaire d'utiliser le constructeur de la classe GLO_message. Cependant, les messages échangés doivent suivre le format indiqué dans la classe.

## Fonctionnement interne du client

Cette section décrit le fonctionnement interne du client comme attendu lors de la correction. Cette description suit le gabarit qui vous est fourni. La récupération des arguments étant déjà faite dans la fonction **main**, vous n'avez pas à modifier cette dernière.

**___init___** Cette méthode est appelée quand un objet Client est créé, elle doit initialiser un socket TCP en IPv4 et le connecter au serveur dont l'adresse est passée en paramètre. Cette méthode initialise également les attributs "_logged_in" et "_username" qui seront utilisés plus tard.

**_recv_data** Cette méthode est à utiliser systématiquement pour recevoir des données du serveur. Elle fait appel au module glosocket pour recevoir les données et au module json pour décoder le message vers un dictionnaire Python. Ce dictionnaire doit contenir les deux clés "header" et "data" pour être considéré comme valide et être retourné. Si les données sont invalides (None, erreur de décodage, clés absentes), le client doit se terminer avec immédiatement avec le code d'erreur -1.

**_authentication** Cette méthode est appelée en boucle par la méthode **run** tant que l'utilisateur ne s'est pas authentifié. Le client doit demander à l'utilisateur s'il souhaite se connecter ou créer un compte, suivi du nom d'utilisateur et enfin le mot de passe. Le client transmet la requête au serveur dans un GLO_message avec l'entête "AUTH_" appropriée, et ses identifiants contenus dans un dictionnaire {"username": <nom d'utilisateur>,"password": <mot de passe>} en données. Si le serveur répond avec un entête "OK", les attributs "_logged_in" et "_username" sont mis à jour, sinon l'erreur est affichée.

**_main_loop** Cette méthode est appelée en boucle par la méthode **run** une fois l'utilisateur authentifié. Le client demande à l'utilisateur de choisir une option dans le menu et appelle la fonction demandée ou termine immédiatement le programme.

**_reading** Cette méthode permet à un utilisateur de consulter ses courriels. Le client commence par l'envoi d'une requête contenant l'entête "INBOX_READING_REQUEST" et le nom de l'utilisateur en donnée. Si le serveur répond avec un entête "ERROR" ou qu'il n'y a pas de courriel, le client retourne au menu principal, sinon le client affiche la liste des sujets à l'utilisateur et récupère son choix. Le client, ensuite, envoie une requête au serveur contenant l'entête "INBOX_READING_CHOICE et en données un dictionnaire contenant son nom d'utilisateur et son choix dans les champs "unername" et "choice" respectivement. Si le serveur répond avec un entête "OK", le client charge le message depuis le champ "data" et l'affiche à l'utilisateur en suivant le gabarit EMAIL_DISPLAY, sinon l'erreur est affichée.

**_sending** Cette méthode permet à un utilisateur d'envoyer un courriel. Le client commence par demander à l'utilisateur l'adresse email du destinataire suivi du sujet du message et enfin le contenu du message. La saisie de ce dernier doit continuer jusqu'à ce que l'utilisateur entre un point (.) seul sur une ligne, cette ligne n'est pas incluse dans le message. Le client génère ensuite un objet **EmailMessage** et le transmet en tant que chaîne de caractère au serveur avec l'entête "EMAIL_SENDING". Si le serveur répond avec un entête "ERROR", l'erreur est affichée à l'utilisateur avant le retour de la fonction.

**_get_stats** Cette méthode permet à un utilisateur de voir les statistiques liées à son compte. Le client envoie une requête au serveur contenant l'entête "STATS_REQUEST" et le nom d'utilisateur en donnée. Si le serveur répond avec un entête "OK", le client affiche les statistiques à l'utilisateur en suivant le gabarit STATS_DISPLAY, sinon l'erreur est affichée.

## Fonctionnement interne du serveur

Cette section décrit le fonctionnement interne du serveur comme attendu lors de la correction. Cette description suit le gabarit qui vous est fourni. L'instanciation et le démarrage du serveur étant déjà faits dans la fonction **main**, vous n'avez pas à modifier cette dernière.

**___init___** Cette méthode est appelée quand un serveur est créé, elle doit initialiser un socket TCP en IPv4 et le mettre en mode écoute. Cette méthode doit également initialiser

deux listes pour les sockets clients, créer le dossier des données pour le serveur dans le dossier courant s'il n'existe pas, et compiler un motif regex qui sera utilisé pour vérifier les adresses courriel.

**__recv__data** Cette méthode est à utiliser systématiquement pour recevoir des données du client. Elle fait appel au module glosocket pour recevoir les données et au module **json** pour décoder le message vers un dictionnaire Python. Ce dictionnaire doit contenir les deux clés "header" et "data" pour être considéré comme valide et être retourné. Si les données sont invalides (None, erreur de décodage, clés absentes), le serveur doit fermer la connexion avec ce socket et le retirer de toutes les listes dont il fait partie et retourner None.

**__main__loop** Cette méthode est appelée en boucle par la méthode **run**. Elle attend sur select qu'un socket soit prêt à être lu. Ensuite, selon le socket, l'une des méthodes **__accept__client**, **__process__client** ou **__authenticate__client** est appelée.

**__accept__client** Cette méthode accepte la connexion d'un nouveau client et ajoute son socket aux listes appropriées.

**__auth...__client** Cette méthode traite les demandes de création de comptes et de connexion. Après la réception de données du client, le serveur vérifie si le nom d'utilisateur et le mot de passe sont valides.

Création de compte Le serveur doit vérifier si le nom d'utilisateur n'est pas déjà pris, il vérifie également que le mot de passe contient au moins une minuscule, une majuscule, un chiffre et que sa taille est supérieure ou égale à 9. Si les critères sont validés, le serveur crée un dossier au nom de l'utilisateur dans son dossier de données et y ajoute un fichier nommé "passwd" contenant sur la première ligne le mot de passe haché à l'aide de la fonction **sha384** avant de renvoyer au client un message avec l'entête "OK". Si l'un des critères n'est pas validé, le serveur répond avec un entête "ERROR" et ajoute dans le champ "data" un message précisant la source de l'erreur (exemples : "Mot de passe invalide.", "Nom d'utilisateur déjà pris.").

Connexion Le serveur doit vérifier si le nom d'utilisateur existe puis vérifie que le mot de passe fourni correspond à celui enregistré pour cet utilisateur. Si les critères sont validés, le serveur répond avec un entête "OK". Si l'un des critères n'est pas validé, le serveur répond avec un entête "ERROR" et ajoute dans le champ "data" un message précisant la source de l'erreur (exemples : "Mot de passe incorrect.", "Nom d'utilisateur inexistant.").

Avant de retourner, le serveur envoie au client la réponse, et si cette dernière est positive le serveur ajoute le socket du client aux listes appropriées.

**__process__client** Cette méthode traite les requêtes d'utilisateurs connectés. Cette fonction récupère la requête du client, et selon son entête la transmet les données à l'une des méthodes **__get__subject__list**, **__get__email**, **__send__email** ou **__get__stats**. Le message retourné par la méthode appelée est transmis au client avant le retour de cette méthode.

**__get__subject...** Cette méthode récupère la liste des courriels d'un utilisateur. Si le nom d'utilisateur est invalide, le serveur retourne un message avec l'entête "ERROR" et en précisant la source de l'erreur. Sinon, le serveur crée une liste des sujets à l'aide du gabarit SUBJECT_DISPLAY et retourne un message contenant l'entête "OK" et la liste.

**__get__email** Cette méthode récupère le contenu du courriel choisi par l'utilisateur. Si le choix ou le nom d'utilisateur est invalide, le serveur retourne un message avec l'entête "ERROR" et en précisant la source de l'erreur. Sinon, le serveur récupère le contenu du courriel et retourne un message contenant l'entête "OK" et la représentation du courriel en chaîne de caractère.

**__send__mail** Cette méthode envoie un courriel que lui a transmis un client. Le serveur doit s'assurer que les adresses sources et destinations sont bien formées, de plus il doit vérifier que l'expéditeur existe. Si l'un de ces critères n'est pas valide, le courriel n'est pas envoyé et le serveur retourne un message avec l'entête "ERROR" et en précisant la source de l'erreur. Sinon, le serveur examine le domaine de destination.

Interne Le serveur vérifie que l'utilisateur de destination existe, si ce n'est pas le cas le message est écrit dans le SERVER_LOST_DIR et l'entête retourné est "ERROR". Sinon, le message est écrit directement dans le dossier du destinataire. Vous être libre de choisir la nomenclature des fichiers de message tant que lors de la consultation les messages apparaissent du plus ancien (n°1) au plus récent.

Externe Le serveur établit une connexion avec le serveur SMTP et tente d'envoyer le courriel. L'entête du message retourné contient "OK" ou "ERROR" selon le résultat de l'opération.

**__get__stats** Cette méthode récupère les statistiques d'un utilisateur. Si le nom d'utilisateur est invalide, le serveur retourne un message avec l'entête "ERROR" et en précisant la source de l'erreur. Sinon le serveur récupère la taille de tous les fichiers contenus dans le dossier utilisateur (en octets) ainsi que le nombre de courriels. La méthode retourne alors un message avec l'entête "OK" et un dictionnaire contenant les champs "count" et "folder_size" en données.

# Environement de test

Les travaux seront testés dans une machine virtuelle basée sur l'image docker python:latest. Assurez-vous que votre travail fonctionne dans cet environnement, en particulier la manipulation de fichier.
Les tests utilisent la version de glosocket et TP4_utils qui vous ont été fournis. Vous ne pouvez donc pas modifier ces modules. Vous devez également utiliser impérativement les modèles TP4_server.py et TP4_client.py qui vous sont fournis.