Documentation Web Service

REGISTRE

`GET /users`

* Cette route n'attend aucun paramètre.

* Cette route retourne un tableau contenant tous les utilisateurs actuellement connectés au serveur de registre.

***

`GET /{user}`

* Cette route attend un paramètre sous forme de `string` dans l'URL correspondant au nom de l'utilisateur dont on souhaite récupérer les informations.

* Cette route retourne un tableau contenant les informations de l'utilisateur ciblé.

* Cette route peut renvoyer une erreur 404 si l'utilisateur ciblé n'est pas connecté au serveur de registre.

***

`POST /register`

* Cette route attend en paramètre du body un `object` contenant les valeurs `name`, `host` et `port`.

* Cette route retourne un `object` contenant le message de succès ou d'échec, un `string` contenant le pseudo de l'utilisateur s'étant connecté ainsi que la liste des utilisateurs actuellement connectés au serveur de registre.

* Cette route peut renvoyer une erreur 500 si le `body` de la requête est invalide ou si l'utilisateur qui tente de se connecter est déjà connecté au serveur de registre.

***

`DELETE /{user}`

* Cette route attend un paramètre sous forme de `string` dans l'URL correspondant au nom de l'utilisateur à déconnecter du serveur de registre.

* Cette route retourne un `object` contenant le message de succès ou d'échec, ainsi que la liste des utilisateurs actuellement connectés au serveur de registre.

* Cette route peut renvoyer une erreur 404 si l'utilisateur ciblé n'est pas connecté au serveur de registre.

SERVEURS CLIENTS

`GET /ping`

* Cette route n'attend aucun paramètre.

* Cette route retourne rien.

***

`GET /users`

* Cette route n'attend aucun paramètre.

* Cette route retourne un tableau contenant tous les utilisateurs actuellement connectés au serveur de registre.

***

`GET /all`

* Cette route n'attend aucun paramètre.

* Cette route retourne un tableau contenant tous les messages actuellement enregistrés en mémoire sur le client, chaque clé de tableau correspond à l'utilisateur émetteur du message.

***

`GET /{user}`

* Cette route attend un paramètre sous forme de `string` dans l'URL correspondant au nom de l'utilisateur dont on souhaite récupérer les messages.

* Cette route retourne un tableau contenant les messages émis par l'utilisateur ciblé.

***

`POST /login`

* Cette route attend en paramètre du body un `object` contenant les valeurs `string` `name`, `string` `host` et `string` `port`.

* Cette route retourne un `object` contenant le message de succès ou d'échec, un `string` contenant le pseudo de l'utilisateur s'étant connecté ainsi que la liste des utilisateurs actuellement connectés au serveur de registre.

* Cette route peut renvoyer une erreur 500 si le `body` de la requête est invalide ou si l'utilisateur qui tente de se connecter est déjà connecté au serveur de registre.

***

`POST /chat/{user}`

* Cette route attend un paramètre sous forme de `string` dans l'URL correspondant au nom de l'utilisateur à qui le message doit être envoyé ainsi que le message en `plaintext` dans le `body` de la requête.

* Cette route retourne un `object` contenant le message de succès ou d'échec.

* Cette route peut renvoyer une erreur 404 si l'utilisateur ciblé par l'envoi n'est pas connecté au serveur de registre ou si l'utilisateur qui tente l'envoi n'est pas connecté au serveur de registre.

***

`DELETE /logout`

* Cette route attend aucun paramètre.

* Cette route retourne un `object` contenant le message de succès ou d'échec, ainsi que la liste des utilisateurs actuellement connectés au serveur de registre.

* Cette route peut renvoyer une erreur 404 si l'utilisateur ciblé n'est pas connecté au serveur de registre.