Tchatator

Documentation utilisateur

A2.1 The void

Mattéo KERVADEC

Kylian Houedec

Gabriel FROC

Ewen JAIN

Antoine GUILLERM

Benjamin GERARD

IUT Lannion - SAÉ 3 2024-2025

Le projet TripEnArvor a pour objectif de mettre en place un mécanisme d'échange de messages au format texte brut entre les professionnels et les visiteurs/clients de la plateforme. Ce mécanisme fonctionnera en mode asynchrone, permettant à chaque correspondant d'envoyer et recevoir des messages de manière décalée, sans nécessiter de connexion simultanée.

Ce document fournit une explication détaillée sur l'installation, l'utilisation et la configuration du service.

Dernière modification : 2 février 2025

Table des matières

Chapitre 1

Introduction

Tchatator est un service de discussion asynchrone en C utilisant les sockets et une API permettant aux clients, professionnels et administrateurs d'envoyer et de recevoir des messages. Ce document fournit une explication détaillée sur l'installation, l'utilisation et la configuration du service.

1.1 Installation

1.1.1 Prérequis

Avant d'installer le service, assurez-vous d'avoir les prérequis nécessaires installés sur votre machine. Cela inclut la bibliothèque json-c, qui est utilisée pour le traitement des données JSON, ainsi que d'autres outils nécessaires à la compilation et à l'exécution des scripts.

Prérequis communs :

- **Système d'exploitation :** Linux
- **Compilateur :** gcc pour compiler le code C
- **Bibliothèque json-c :** Bibliothèque pour le traitement des données JSON.

Pour installer la bibliothèque json-c : Sur une distribution Linux (comme Ubuntu/Debian), vous pouvez installer json-c avec apt :

```
sudo apt-get update
sudo apt-get install libjson-c-dev
```

1.1.2 Installation du Service côté Serveur

Pour installer et exécuter le serveur, suivez les étapes suivantes.

a) Télécharger les fichiers du projet : Clonez ou téléchargez le dépôt contenant le code source du projet.

```
git clone https://github.com/Matteo-K/PACT.git
cd ./PACT/tchatator
```

b) Compilation et exécution du serveur : Le serveur utilise le script tchatator.sh pour compiler le programme. Ce script gère la configuration, l'inclusion des bibliothèques nécessaires (comme json-c), et la compilation du code source.

1. Ouvrez un terminal et naviguez jusqu'au répertoire PACT/tchatator.
2. Exécutez le script tchatator.sh pour compiler le programme.

```
chmod +x tchatator.sh
./tchatator.sh
```

Le script s'assurera que toutes les dépendances nécessaires sont présentes, comme la bibliothèque json-c, et compilera le code C dans un exécutable. Ainsi ce dernier sera exécuter.

c) Options d'exécution

```
./tchatator.sh [option]
```

OPTIONS

-h, –help

— afficher l'aide

-v, –version

— afficher la version actuelle du tchatator

-b, –verbose

— afficher les logs

d) Service côté Client : Le script user.sh est utilisé pour lancer un terminal interactif pour l'utilisateur. Ce script gère l'installation de telnet, l'ouverture d'une session utilisateur et la connexion au serveur de chat. Il suffit de rendre le script exécutable et de l'exécuter.

1. Ouvrez un terminal et naviguez jusqu'au répertoire PACT/tchatator.
2. Rendez le script user.sh exécutable :

   ```
   chmod +x user.sh
   ```

3. Exécutez le script pour ouvrir un terminal et commencer à interagir avec le serveur.

   ```
   ./user.sh
   ```

1.1.3 Dépannage

Voici quelques problèmes courants et leurs solutions possibles :

— Problème de dépendances (json-c) :
  — Assurez-vous que la bibliothèque json-c est correctement installée.
  — Si vous avez des problèmes lors de la compilation, vérifiez que le répertoire de l'en-tête json-c est correctement inclus dans les chemins du compilateur.

— Problème de connexion au serveur :
  — Vérifiez si le serveur est en cours d'exécution et écoute bien sur le port attendu.

Chapitre 2

Démarrage

2.1 Exécution du service avec Telnet

Telnet est un protocole de communication réseau qui permet à un utilisateur de se connecter à un ordinateur distant via un terminal en ligne de commande. Il offre la possibilité d'exécuter des commandes sur le serveur distant.

```
telnet the-void.ventsdouest.dev 8081
```

2.1.1 Connexion au service

Le principe d'identification dans Tchatator permet de sécuriser les échanges en assurant que seuls les utilisateurs autorisés peuvent participer aux conversations. Il permet aussi de personnaliser l'expérience, gérer les rôles (clients, professionnels, administrateurs) et garantir la traçabilité des interactions.

```
LOGIN: <clé_api>
```

— clé api : clé api de l'utilisateur voulant se connecter

La connexion au service renvoie un token si tout se passe bien.
Exemple :

```
{
  "statut" : "200/OK",
  "tokken" : "Age8Ku"
}
```

Réponses possibles après la tentatives de connexion

— 200/OK : accès autorisé
— 400/TOO _ MANY _ ARGS : Trop de paramètres fournit
— 400/MISSING _ ARGS : Pas assez de paramètres fournit
— 403/CLIENT _ BANNED : accès refusé, client banni
— 403/CLIENT _ BLOCKED : accès refusé, client bloqué
— 429/QUOTA _ EXCEEDED : Quota dépassé pour la clé API

2.1.2 Déconnexion du service

La déconnexion du service permet de sécuriser les sessions utilisateur en mettant fin à l'accès à leurs données et fonctionnalités après une période d'inactivité ou lorsque l'utilisateur choisit de se déconnecter. Cela protège les informations personnelles et prévient les accès non autorisés.

```
BYE BYE: <tokken>
```

— tokken : identifiant de l'utilisateur, reçu lors de la connexion

Exemple :

```
{
  "statut" : "200/OK"
}
```

Réponses possibles après la tentatives de déconnexion

— 200/OK : accès autorisé

— 400/TOO _ MANY _ ARGS : Trop de paramètres fournit

— 400/MISSING _ ARGS : Pas assez de paramètres fournit

— 403/CLIENT _ BANNED : accès refusé, client banni

— 403/CLIENT _ BLOCKED : accès refusé, client bloqué

2.1.3 Avoir de l'aide

La commande AIDE permet d'afficher de l'aide et des informations sur le fonctionnement du service ou de l'application. Elle fournit des instructions sur l'utilisation des différentes fonctionnalités et peut guider l'utilisateur en cas de besoin, facilitant ainsi la compréhension et la gestion du service.

```
AIDE: <fonctionnalité>
```

— fonctionnalité : une des fonctionnalités du tchatator

En cours de développement

2.1.4 Envoyer un message

Envoyer un message entre un professionnel et un membre (ou inversement) permet de faciliter la communication bidirectionnelle. Cela permet aux professionnels de répondre aux questions, fournir des informations ou clarifications, tandis que les membres peuvent poser des questions ou exprimer des besoins spécifiques. Cette interaction renforce l'efficacité du service et améliore l'expérience utilisateur.

```
MSG: <tokken> | <clé api destiantaire> | <message>
```

— tokken : identifiant de l'utilisateur après une connexion

— clé api destiantaire : identifiant du destinataire qui n'est pas forcément connecté.

— message : message émit pour le destinataire

Réponses possibles après la tentatives d'envoi de message

— 200/OK : accès autorisé
— 400/TOO _ MANY _ ARGS : Trop de paramètres fournit
— 400/MISSING _ ARGS : Pas assez de paramètres fournit
— 401/UNAUTH/UNKNOWN _ RECIPIENT : Destinataire inconnue
— 403/CLIENT _ BANNED : accès refusé, client banni
— 403/CLIENT _ BLOCKED : accès refusé, client bloqué
— 403/UNAUTHORIZED _ USE : utilisateur non autorisé d'accéder à cette REQUÊTE.
— 416/MISFMT : Message mal formaté ou trop long
— 429/QUOTA _ EXCEEDED : Quota dépassé pour la clé API
— 500/INTERNAL_ SERVER _ ERROR : Erreur du côté du serveur

2.1.5 Consulter l'historique de message

Consulter l'historique des messages permet de garder une trace des échanges passés, ce qui facilite le suivi des conversations, la résolution de problèmes et la référence à des informations précédemment partagées. Cela améliore la continuité du service et aide à éviter les malentendus ou répétitions dans les interactions.

```
HISTORIQUE: <tokken>
```

— tokken : identifiant de l'utilisateur après une connexion

2.1.6 Arrêt du serveur

La commande STOP permet à l'administrateur d'arrêter le serveur de manière contrôlée. Cela garantit que toutes les connexions et les processus en cours sont terminés proprement, évitant ainsi la perte de données et assurant la sécurité du système avant son arrêt complet.

```
STOP:
```

En cours de développement (sécurisation)