Tchatator

Documentation technique

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 a pour but de présenter le fonctionnement interne de l'API tchatator.

Dernière modification : 2 février 2025

Table des matières

Chapitre 1

Introduction

L'API sera composée de trois modes d'accès distincts : un accès professionnel pour les échanges entre professionnels et clients, un accès client pour la communication avec les professionnels, et un accès administrateur pour la gestion des actions de blocage et de bannissement. Cette solution standardisera et sécurisera les interactions entre les utilisateurs de la plateforme, tout en respectant les exigences du projet.

1.1 Glossaire

— **SERVICE ou SERVEUR :** le logiciel qui attend et traite les REQUÊTES.

— **CLIENT :** le logiciel qui se connecte et interroge le SERVICE. Ne pas confondre avec le client de la plateforme.

— **PROTOCOLE :** les règles et la GRAMMAIRE permettant à un CLIENT de soumettre des REQUÊTES au SERVICE.

— **REQUÊTE :** une action faite sur le SERVICE par le biais d'un CLIENT en utilisant le PROTOCOLE.

— **GRAMMAIRE :** la syntaxe décrivant comment doivent être formatées les REQUÊTES du PROTOCOLE. *

— **CLÉ D'API :** un code secret que le CLIENT envoie au SERVICE pour autoriser son utilisation. La fourniture d'une CLÉ D'API peut donner des droits spécifiques.

— **PARAMÉTRAGE :** un fichier texte contenant des valeurs modifiables par un administrateur, sans besoin de création d'une interface. Ces valeurs sont lues au démarrage du SERVICE.

1.2 Architecture du Code

1.2.1 Structure du projet

Le projet est structuré de manière modulaire et comprend les fichiers suivants :

— **tchatator.c :** Point d'entrée du programme, contenant la fonction main().

— **const.h :** Contient les constantes globales utilisées dans tout le programme.

— **const.c :** Contient les valeurs initiales des constantes et les paramètres de configuration.

— **fonction serveur.c :** Fichier avec les fonctions principales du serveur qui gèrent la logique de traitement des demandes des clients.

— **fonction serveur.h :** Fichier avec les entêtes des fonctions principales du serveur.

— **outils.c :** Contient des fonctions utilitaires pour la manipulation des données.

— **outils.h :** Contient les entêtes des fonctions utilitaires.

— **bdd.c :** Gère les interactions avec la base de données.

— **bdd.h :** Contient les entêtes des fonctions d'interactions de la base de données.

1.3 Installation

1.3.1 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
```

1.3.2 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
```

1.3.3 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.