Protocoles Internet
TP 2 : REST

Juliusz Chroboczek

9 octobre 2023

Un serveur de chat tourne à l'adresse `https://jch.irif.fr:8082`

Exercice 1. Postez un message dans le *chat* depuis l'interface *web*.

Exercice 2. En quel langage est implémenté le client *web* ? Comment pouvez-vous examiner son code source ?

Le serveur implémente l'API REST suivante :

- une requête `GET` à l'URL `/chat/` retourne la liste des identificateurs des messages du *chat*, en ordre chronologique ;
- une requête `POST` à l'URL `/chat/` crée un nouveau message, et retourne son identificateur ;
- une requête `GET` à l'URL `/chat/`*id* retourne la valeur d'un message de *chat* ; la date du message est retournée dans l'entête `Last-Modified` ;
- une requête `DELETE` à la même URL supprime un message.

Exercice 3. Cette interface est sans état de session du côté serveur. Qu'est-ce que ça veut dire ?

Si vous vous connectez à une instance tournant sur un serveur autre que `jch.irif.fr`, votre client devrait se plaindre du certificat TLS invalide. Dans ce cas, il faudra ajouter l'option « `-k` » à la ligne de commande de `curl`. De même, il faudra ajouter ce qui suit à tous vos programmes Go :

```
transport := &*http.DefaultTransport.(*http.Transport)
transport.TLSClientConfig = &tls.Config{InsecureSkipVerify: true}
client := &http.Client{
    Transport: transport,
    Timeout:   50 * time.Second,
}
```

Ensuite, au lieu de `http.Get(...)`, il faudra utiliser `client.Get(...)`, etc.

Exercice 4.

1. À l'aide de la commande `curl -i` (ou `curl -ki`, voir ci-dessus), examinez la liste des messages du *chat*.
2. À l'aide de la commande `curl -d`, postez un message dans le *chat*.

Exercice 5. Écrivez un programme Go qui affiche les 50 derniers messages du *chat* (ou tous les messages s'il y en a moins de 50). Combien de requêtes HTTP votre programme fait-il ? Serait-il possible de les exécuter en parallèle ? Quel est le nombre de RTT minimal ? Quel problème pourrait-on avoir ?

Exercice 6. Modifiez votre programme pour qu'il affiche les messages du *chat* au fur et à mesure qu'ils sont publiés par le serveur. Comme HTTP ne prévoit pas la possibilité de notifications asynchrones, il faudra faire périodiquement faire une requête `GET` à l'URL `/chat/`, par exemple toutes les 10 secondes.

Exercice 7. Modifiez votre programme pour qu'il évite de recevoir une liste où aucun message n'a changé en utilisant l'entête `If-None-Match`. Pourquoi est-ce préférable à l'entête `If-Modified-Since` ?

Exercice 8. Ajoutez à votre programme la possibilité de publier un nouveau message dans le *chat* ainsi que d'en supprimer un.