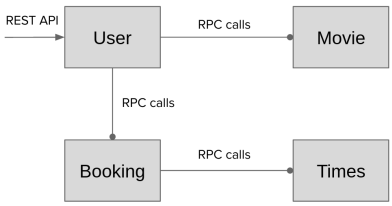
# TP Flask, REST et OpenAPI

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

## 1. TP vert




Le but de ce TP est de construire le même ensemble de services que dans le TP REST, à savoir : `Movie`, `Booking`, `Showtime` et `User`. Cette fois seul le service `User` recevra des requêtes de type REST depuis l'extérieur. Tous les appels internes entre les services seront eux effectués en utilisant `gRPC`.

| | |
|---|---|
| **IMPORTANT** | Toute modification de l'API nécessite une recompilation du fichier `proto` ! |
| **IMPORTANT** | `Booking` est à la fois un Service et un client en `gRPC`. `User` est aussi à la fois un service et un client mais son service offre une API REST. |
| **TIP** | Pour tester au fur et à mesure vous pouvez éventuellement créer un client test pour chaque service. |

Voici les étapes à suivre pour ce TP :

1. Terminer l'écriture du service Movie en utilisant `gRPC`. Vous devez retrouver les fonctionnalités que vous aviez dans le précédent TP.
2. Ecrivez les fichiers d'API `proto` pour les services `Booking` et `Showtime`
3. Ecrivez le code du service `Showtime` et testez ce code.
4. Ecrivez le code du service `Booking` et testez ce code (qui doit appeler le service `Showtime` en utilisant le stub correspondant).
5. Reprenez votre service REST `user`, remplacez-y les requêtes REST vers `Movie` et `Booking` par des appels de procédures distantes en utilisant les `stub` associés.

| | |
|---|---|
| **TIP** | Pour le service `Booking` vous aurez besoin de `Nested` types pour les messages Protocol Buffers, voir https://developers.google.com/protocol-buffers/docs/proto#nested |
| **TIP** | Attention à la manipulation des sorties de l'appel de procédure distant pour `User` qui doit retourner du `json` ! |

# 2. TP bleu

Allez plus loin dans l'utilisation de `gRPC` en utilisant les appels asynchones des procédures distantes et l'utilisation de `futures`. Vérifiez que les appels asynchrones fonctionnent bien.

| | |
|---|---|
| **TIP** | voir la documentation https://grpc.github.io/grpc/python/ |

# 3. TP rouge

Ici nous souhaitons tester la performance de l'utilisation de gRPC par rapport à REST. Pour cela vous pouvez écrire un script de benchmark qui aura les paramètres suivants :

- taille du message d'entrée
- taille du message de retour
- nombre de requêtes à effectuer

Il faudra faire en sorte de faire des appels asynchrones vers REST et les procédures distantes de façon à lancer beaucoup de requêtes simultannées.

| | |
|---|---|
| **TIP** | vous pouvez lire et vous inspirer des articles suivants : https://medium.com/@EmperorRXF/evaluating-performance-of-rest-vs-grpc-1b8bdf0b22da https://the-worst.dev/rest-vs-grpc-performance-benchmark-in-net-core-3-1/ |
| **TIP** | vous pouvez vous renseigner sur les métriques "Web Vitals" proposées par Google pour évaluer l'expérience utilisateur : https://web.dev/vitals/#core-web-vitals |