

Documentation d'installation

SciVisu

Interface de visualisation de données
scientifique en WebGL

Page web nécessaire :

Pour que le JavaScript du template Geometry puisse correctement fonctionner, il faut un fichier HTML contenant au minimum :

- une balise « div » d'id « container »
- une balise script de source « <https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js> », utiliser pour la partie MQTT du template. Ou un chemin vers votre librairie paho-mqtt.
- 6 balises scripts avec des sources, ces fichiers JavaScript sont disponibles sur le GitHub du projet, à l'adresse : « https://github.com/NICOLASGON/mqtt_webgl_template »

Les sources utilisées par nos fichiers JavaScript se trouvent dans la branche « siteweb », et sont disponibles dans le répertoire « js » pour :

- three.min.js
- OrbitControls.js
- dat.gui.min.js

et dans le répertoire SciVisuJS pour :

- init.js
- obj.js
- mqtt.js

Une page HTML simpliste ne contenant que les éléments nécessaires est disponible dans le répertoire « Exemple_Installation », la page « [example_templateGeo.html](#) » toujours dans la branche siteweb du GitHub.

Celle-ci contient également une balise de style CSS pour permettre d'avoir une page complètement noire (la scène webGL possédant des bords, et des barres de scroll, ceux-ci seront caché par ce CSS).

```

<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
    <meta charset="utf-8">
    <style>
      body {
        background:black;
        overflow:hidden;
      }
    </style>
  </head>
  <body>
    <div id="container"></div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js"></script>

    <script src="../js/three.min.js"></script>
    <script src="../js/OrbitControls.js"></script>
    <script src="../js/dat.gui.min.js"></script>

    <script src="../SciVisuJS/init.js"></script>
    <script src="../SciVisuJS/obj.js"></script>
    <script src="../SciVisuJS/mqtt.js"></script>

  </body>
</html>

```

Fichier example_templateGeo.html

css	14/03/2018 20:52	Dossier de fichiers	
img	14/03/2018 20:52	Dossier de fichiers	
js	14/03/2018 20:52	Dossier de fichiers	
models	14/03/2018 20:52	Dossier de fichiers	
SciVisuJS	14/03/2018 20:52	Dossier de fichiers	
scss	14/03/2018 20:52	Dossier de fichiers	
sidenav	14/03/2018 20:52	Dossier de fichiers	
templateGeometry	14/03/2018 20:52	Dossier de fichiers	
vendor	14/03/2018 20:52	Dossier de fichiers	
templateObject	16/03/2018 18:58	Dossier de fichiers	
Exemple_Installation	16/03/2018 19:09	Dossier de fichiers	
	14/03/2018 20:52	Document texte	1 Ko
gulpfile	14/03/2018 20:52	Fichier JS	4 Ko
LICENSE	14/03/2018 20:52	Fichier	2 Ko
package	14/03/2018 20:52	Fichier JSON	2 Ko
package-lock	14/03/2018 20:52	Fichier JSON	207 Ko
index.nginx-debian	14/03/2018 20:53	Firefox HTML Doc...	10 Ko

Branche siteweb du GitHub, avec répertoire correspondant à la page minimaliste

Connexion MQTT

Une fois la page HTML créée, il faut configurer un broker MQTT pour faire la passerelle entre nos envois de données et le JavaScript inclus dans la page html.

Dans notre cas, nous utilisons le broker MQTT Mosquitto qui doit être configuré après avoir été installé sur votre serveur. Une simple modification du fichier de configuration est nécessaire :

```
[root@ptutwebgl /]$ cd /etc/mosquitto/conf.d ; ls
broker.conf  frugal.conf.save  README
[root@ptutwebgl conf.d]$ cat broker.conf
listener 1883

listener 1882
protocol websockets

allow_anonymous false
password_file /etc/mosquitto/passwd
```

Dans ce fichier, on écrit sur quel port le broker va écouter, on peut voir ici que le broker écoute sur le port 1883 (port par défaut), il faut également rajouter le port 1882 (ou port de votre choix) suivi de « protocol websockets » pour permettre au broker de communiquer avec les navigateurs. Les deux lignes qui suivent permettent de préciser où sont stockés les logins et mots de passe, si vous ne voulez pas utiliser l'authentification, il suffit de supprimer les deux dernières lignes. (Un reboot du broker est nécessaire)

Une fois l'infrastructure réseau prête, il manque une dernière étape, l'adaptation du JavaScript à votre infrastructure.

Il va falloir modifier le fichier « mqtt.js » inclut précédemment.

Nous trouvons au début de ce fichier JavaScript :

```
client = new Paho.MQTT.Client("91.224.148.106", Number(2533), "templateGeo");
client.connect({
  username: "ptut",
  password: "ptut",
  onSuccess: onConnect
});
```

Il faut modifier l'adresse IP pour y mettre celle du serveur sur lequel votre broker MQTT est configuré, le paramètre de Number() va être le port **websocket** défini dans le fichier de configuration de votre serveur (ici le port 2533 qui est redirigé vers 1882).

Il faut également changer l'username et le password, si votre serveur est configuré comme utilisant une authentification.

Le template est maintenant prêt à l'usage !

Utilisation

Pour utiliser le template, il suffit d'envoyer des messages sur votre broker (correspondant à l'adresse IP renseignée précédemment), des applications Android gratuites existent pour envoyer des messages tel que MyMQTT (voir « Installation d'une application pour MQTT »).

Les messages doivent correspondre au format JSON défini dans la documentation (GitHub → branche master → DocumentsRendus → Etape_4_Documentation).

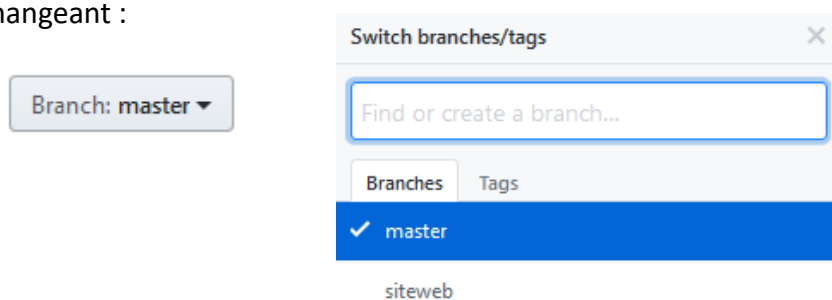
Vous pouvez maintenant utiliser le template.

Comment récupérer les sources depuis GitHub

Comme précisé ci-dessus, les sources et les documentations sont disponibles sur le GitHub du projet à l'adresse :

https://github.com/NICOLASGON/mqtt_webgl_template

Le changement de branche (entre *master* et *siteweb*) se fait en cliquant sur « Branch : » et en changeant :



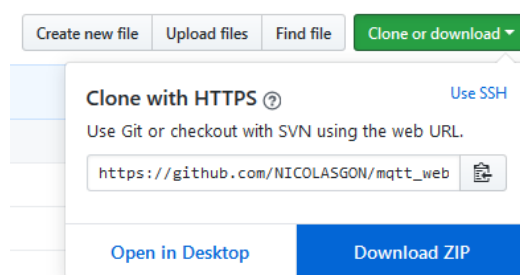
Pour pouvoir récupérer les sources sur son ordinateur, deux méthodes sont possibles : le clone du répertoire git, ou le téléchargement.

Pour cloner le répertoire git, il faut faire la commande

```
git clone https://github.com/NICOLASGON/mqtt_webgl_template
```

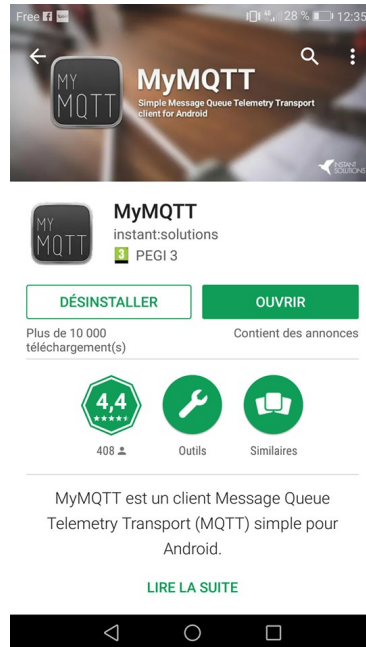
qui permet de créer une copie complète du GitHub dans un répertoire à l'emplacement où s'est effectué le clonage. Une installation de GIT est nécessaire.

Si GIT n'est pas installé sur votre machine, vous pouvez également utiliser le téléchargement par ZIP disponible sur GitHub pour obtenir une copie en .zip de la branche dans laquelle vous vous trouvez :

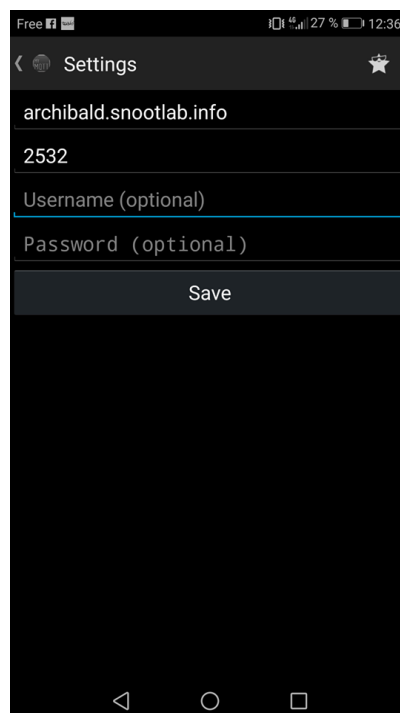


Installation d'une application pour MQTT

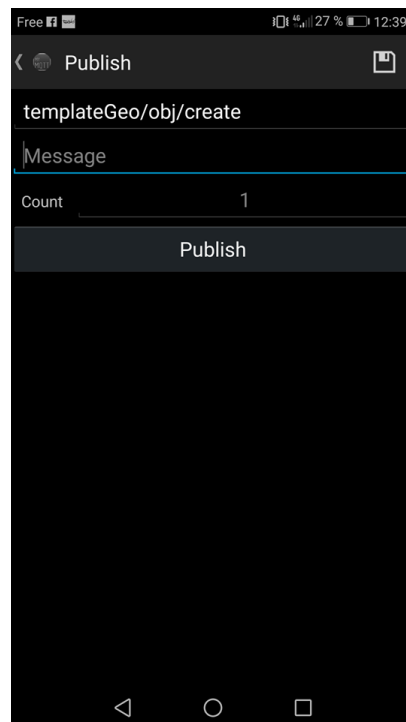
L'application MyMQTT est gratuite et disponible sur le PlayStore, elle est très simple d'utilisation et permet d'envoyer des messages par le biais du protocole MQTT sur un broker.



Une fois installé il faut aller dans l'onglet Settings pour paramétrer l'application, il faut rentrer l'adresse du serveur ou est hébergé votre broker ainsi que le port d'écoute. Vous pouvez également rentrer l'username et le mot de passe si vous l'avez défini.



Pour publier un message, il suffit d'aller dans l'onglet Publish qui permet de pousser un message vers le broker avec le topic donné (ici : templateGeo/obj/create). Il reste à ajouter le contenu du message qui est à adapter en fonction du résultat que vous souhaitez. On peut également choisir combien de fois le message va être envoyé (Count).



Il est également possible de s'abonner à plusieurs topics depuis l'onglet Subscribe. On peut donc voir les messages qui passent dans ces topics depuis l'onglet Dashboard.

