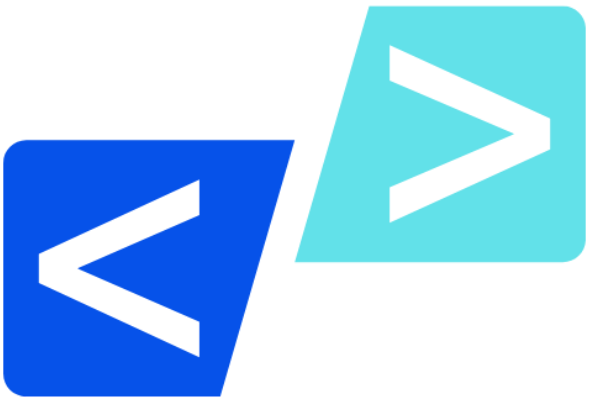

Documentation
Technique du Server

Sommaire

I. Lancement

Vous avez la possibilité de lancer le back-end indépendamment du reste des autres services avec cette commande:

```
docker-compose -f docker-compose.dev.yml up --build back
```

ou pour la base de données

```
docker-compose -f docker-compose.dev.yml up --build db
```

II. Ajouter de nouveau service

Pour ajouter un nouveau service tout d'abord il faut simplement créer un nouvel INSERT dans le fichier **init.sql** qui se trouve dans le dossier **db-dump** localisé à la racine du fichier.

```
27  --Table Services--
28  CREATE TABLE services (
29      id              SERIAL,
30      name            varchar(255) not null,
31      color           varchar(255) not null,
32      primary key(id)
33  );
34
35  INSERT INTO services(name, color) VALUES ('Twitter', '#1C9CEB');
36  INSERT INTO services(name, color) VALUES ('Spotify', '#1DB954');
37  INSERT INTO services(name, color) VALUES ('Discord', '#5562EA');
38  INSERT INTO services(name, color) VALUES ('Twitch', '#8C45F7');
39  INSERT INTO services(name, color) VALUES ('Youtube', '#F70000');
40  INSERT INTO services(name, color) VALUES ('Telegram', '#26A2E1');
```

Pour un nouveau service il faut renseigner le nom et sa couleur associés.
Voir avec la documentation du front pour l'implémentation de ces nouveaux services.

De plus, il faudra créer un nouveau dossier à la racine du dossier back où vous pourrez mettre tout votre code par rapport à ce service comme la gestion des actions et des réactions.

Pour la gestion des actions de vos services il faudra créer une fonction main qui sera appelé dans la fonction `function loopAR` que vous pourrez trouver dans le **app.js** a la racine du dossier back. Cette fonction a pour but de manager vos actions de votre action, en fonction des id qui sont générés en base de données.

III. Ajouter de nouveau Action Réaction

A. Action

Pour ajouter une nouvelle action tout d'abord il faut simplement créer un nouvel INSERT dans le fichier **init.sql** qui se trouve dans le dossier **db-dump** localisé à la racine du fichier.

```
74  --Actions--
75  CREATE TABLE actions (
76      id                  SERIAL,
77      description         varchar(255) NOT NULL,
78      id_service          INT,
79      params              text[],
80      primary key(id)
81  );
82
83  INSERT INTO actions(description, id_service, params) VALUES ('A new tweet from specific user is posted', 1, ARRAY['User @']);
84  INSERT INTO actions(description, id_service, params) VALUES ('A streamer starts a game specific stream', 4, ARRAY['Streamer name', 'Game name']);
85  INSERT INTO actions(description, id_service, params) VALUES ('You like a video', 5, null);
86  INSERT INTO actions(description, id_service, params) VALUES ('You start a stream', 4, ARRAY['Your username']);
87  INSERT INTO actions(description, id_service, params) VALUES ('When you are mentionned', 1, null);
88  INSERT INTO actions(description, id_service, params) VALUES ('A youtuber post a new video', 5, ARRAY['Youtuber name']);
89  INSERT INTO actions(description, id_service, params) VALUES ('You like a song', 2, null);
90  INSERT INTO actions(description, id_service, params) VALUES ('A streamer exceed an amount of viewer', 4, ARRAY['Streamer name', 'Amount']);
91  INSERT INTO actions(description, id_service, params) VALUES ('A selected streamer starts a new stream', 4, ARRAY['Streamer name']);
92  INSERT INTO actions(description, id_service, params) VALUES ('Spotify temp', 2, null);
93  INSERT INTO actions(description, id_service, params) VALUES ('Post a tweet', 1, null);
```

Pour une nouvelle action il faut renseigné sa description, ce qui équivaut à son nom, l'id du service en question, et un un ARRAY des paramètres que vous avez besoin de demander au user cela se formate par une simple chaine de caractère qui correspondra dans le front au label de l'input.

Mettre tout vaut fichier code (.js) de votre nouvel action dans le dossier du service en question.

B. Réaction

Pour ajouter une nouvelle réaction tout d'abord il faut simplement créer un nouvel INSERT dans le fichier **init.sql** qui se trouve dans le dossier **db-dump** localisé à la racine du fichier.

```
96  --Reactions--
97  CREATE TABLE reactions (
98      id                  SERIAL,
99      description         varchar(255) NOT NULL,
100     id_service          INT,
101     params              text[],
102     primary key(id)
103 );
104
105 INSERT INTO reactions(description, id_service, params) VALUES ('Message a specific user on discord', 3, ARRAY['Username', 'Message']);
106 INSERT INTO reactions(description, id_service, params) VALUES ('Send message to group chat', 6, ARRAY['Group name']);
107 INSERT INTO reactions(description, id_service, params) VALUES ('Message on discord server', 3, ARRAY['Server name', 'Channel name']);
108 INSERT INTO reactions(description, id_service, params) VALUES ('Post a tweet', 1, ARRAY['Tweet text']);
109 INSERT INTO reactions(description, id_service, params) VALUES ('Send message at a specific user', 6, ARRAY['User']);
```

Pour une nouvelle réaction il faut renseigné sa description, ce qui équivaut à son nom, l'id du service en question, et un un ARRAY des paramètre que vous avez besoin de demander au user cela se formate par une simple chaine de caractère qui correspondra dans le front au label de l'input.

Mettre tout vaut fichier code (.js) de votre nouvelle réaction dans le dossier du service en question.

Et pour chaque réaction, une action peut être lien donc dans les fonctions de gestion des actions des services ajouter votre réaction avec son id correspondant, cela revient à créer une nouvelle condition dans les gestions d'action.

IV. Ajouter une nouvel route GET|POST|PUT|DELETE

Pour ajouter une nouvelle route il vous faut:

- Créer un nouveau dossier dans le dossier **routes** qui se trouve à la racine sur dossier back.
- Créer un fichier du même nom que le dossier, et mettre à l'intérieur le code de vos routes comme la template suivante:

```
const express = require('express');
const router = express.Router();

/* GET home page. */
router.get( path: '/', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ) {
  res.status( code: 200).send( body: "Welcome to Area API!");
});

module.exports = router;
```

- Par la suite, il vous faudra relier ce fichier dans le **app.js** global du back, se situant à la racine du dossier back. Pour le relier il vous faudra créer une nouvelle variable comme ceci:

```
const indexRouter = require('./routes/index');
const aboutRouter = require('./routes/about/about');
const authRouter = require('./routes/auth/auth');
const usersRoute = require('./routes/users/users');
const servicesRouter = require('./routes/services/services');
const apiWeatherRouter = require('./routes/api/weather/weather');
const apiLocationRouter = require('./routes/api/location/location');
const weatherRouter = require('./routes/weather/weather');
const dashboardRouter = require('./routes/dashboard/dashboard');
const downloadRouter = require('./routes/download/download');
const ARRouter = require('./routes/actionReaction/actionReaction');
const apiTwitchRouter = require('./routes/api/twitch/twitch');
const apiTwitterRouter = require('./routes/api/twitter/twitter');
const apiSpotifyRouter = require('./routes/api/spotify/spotify');
```

Par la suite avec cette variable utilisait la méthode **app.use(VOTRE PREFIX, FCT)** comme ceci:

```
42    app.use('/', indexRouter);
43    app.use('/', aboutRouter);
44    app.use('/auth', authRouter);
45    app.use('/users', usersRoute);
46    app.use('/services', servicesRouter);
47    app.use('/api/weather', apiWeatherRouter);
48    app.use('/api/location', apiLocationRouter);
49    app.use('/weather', weatherRouter);
50    app.use('/dashboard', dashboardRouter);
51    app.use('/download', downloadRouter);
52    app.use('/AR', ARRouter);
53    app.use('/twitter', apiTwitterRouter);
54    app.use('/spotify', apiSpotifyRouter);
```

Attention à ne pas créer deux fois un même préfixe.

V. Modification BDD

Pour tout nouvel élément, comme une nouvelle TABLE ou INSERT, ne pas oublier de le rajouter dans le fichier **init.sql** qui se trouve dans le dossier **db-dump** a la racine du projet. Ne pas oublier de mettre un commentaire qui correspond à une catégorie.

Cela permettra pour chaque lancement de docker d'être à jour et pas tous les recrée à la main.