Documentation d'utilisation de l'API

Accès à l'API, visionnage de des différentes bases

Request

`GET http://localhost:8000/`

`curl -X GET http://localhost:8000/`

Result

```
[]
```

Ou en cas de sauvegarde effectuée affichera le contenu de la sauvegarde.

Pour créer une base

Request

`POST http://localhost:8000/`

Se mettre dans le body et écrire le nom de la base souhaitée (exemple : base01).

`curl -X POST -d 'base01' http://localhost:8000/`

Result

```
{
    Database created
}
```

Pour visualiser le contenu d'une base

Request

`GET http://localhost:8000/base01`

`curl -X GET http://localhost:8000/base01`

Va donc lister les différentes tables dans la base choisie.

Result

```
[
    "table01"
]
```

Pour créer une table dans notre base

Request

`POST http://localhost:8000/base01`

Se mettre dans le body et écrire le nom de la table souhaitée (exemple : table01).

`curl -X POST -d 'table01' http://localhost:8000/base01`

Result

```
{
    Table created
}
```

Pour visualiser le contenu d'une table

Request

`GET http://localhost:8000/base01/table01`

`curl -X GET http://localhost:8000/base01/table01`

Va donc lister les différents chemin possible dans la table.

Rules : affiche le modèle de donnée de la table

Data : affiche les données de la table

Result

```
[
    "config",
    "rules",
    "data"
]
```

Pour créer un modèle de donnée

Request

`POST http://localhost:8000/base01/table01/rules`

Se mettre dans le body et renvoyer un Json avec les différentes règles.

`curl -X POST -d '{ "name" : "text", "age" : "integer" }' http://localhost:8000/base01/table01/rules`

Exemple

```
{
    "name" : "text",
    "age" : "integer"
}
```

Result

```
{
    rules created
}
```

Pour visualiser le modèle de donnée

Request

`GET http://localhost:8000/base01/table01/rules`

`curl -X GET http://localhost:8000/base01/table01/rules`

Exemple

```
name qui contient un texte et age qui contient un integer.
```

Result

```
{
    "name": "text",
    "age": "integer"
}
```

Pour créer de la data

/!\ Attention /!\

Pour créer de la donnée dans une table il faut bien respecter et avoir défini le modèle de donnée de la table au préalable.

Request

`POST http://localhost:8000/base01/table01/data`

Se mettre dans le body et inscrire des donnnées qui respecte les règles.

`curl -X POST -d '{ "name" : "Thomas", "age" : "22" }' http://localhost:8000/base01/table01/data`

Une fois la création de donnée effectuée un ID est donné automatiquement au groupe de donnée en question.

Result

```
{
    data added
}
```

Pour récupérer les données

Request

`GET http://localhost:8000/base01/table01/data`

`curl -X GET http://localhost:8000/base01/table01/data`

Va donc lister toutes les données présente dans la table.

Result

```
{
    "1": {
            "name": "Thomas",
            "age": "22"
         }
}
```

Pour récupérer, modifier ou supprimer une donnée (et non toute la table)

Request

`GET http://localhost:8000/base01/table01/data/1`

`curl -X GET http://localhost:8000/base01/table01/data/1`

Par exemple pour visualiser que le contenu avec l'ID 1 mettre l'ID après data.

Result

```
{
    "name": "Thomas",
    "age": "22"
}
```

Modification d'une donnée dans la tables spécifier l'ID souhaité.

Request

`PUT http://localhost:8000/base01/table01/data/1`

Pour la modifier se mettre dans le body écrire les nouvelles données souhaitées et respecter les règles mise en place.

```
curl -X PUT -d '{ "name" : "Pierre", "age" : "23" }' http://localhost:8000/base01/table01/data/1
```

Result

```
{
    data updated
}
```

Supression d'une donnée souhaitée.

Request

```
DELETE http://localhost:8000/base01/table01/data/1
```

```
curl -X DELETE http://localhost:8000/base01/table01/data/1
```

Result

```
{
    data deleted
}
```

Sauvegarde

Étant une base de donnée in-memory un système de sauvegarde asynchrone à été mis en place. Donc tout des X minutes (peut être modifié dans le code) créer un fichier Json avec le contenu de la base au complet.