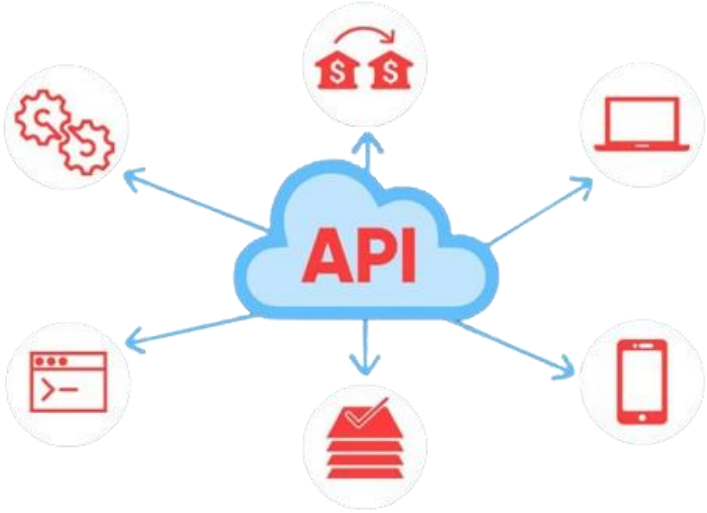
Groupie-Tracker API

Qu'est-ce qu'une API ?

Une API (Application Programming Interface) est un ensemble de règles et de protocoles qui permettent à deux logiciels de communiquer entre eux. Elle définit les méthodes et les données que les développeurs peuvent utiliser pour intégrer les fonctionnalités d'une application dans une autre.



Les API sont généralement utilisées dans divers contextes pour permettre la communication et l'échange de données entre différentes applications, services ou systèmes.

Type d’API ?

- **API Web : Les API Web, également appelées API HTTP, sont accessibles via le protocole HTTP.**
  Elles sont couramment utilisées pour permettre la communication entre des services en ligne.

- **API RESTful : Un style d'architecture de conception d'API web basé sur les principes de REST (Representational State Transfer).**
  Il utilise les méthodes HTTP standard (GET, POST, PUT, DELETE) pour effectuer des opérations CRUD (Create, Read, Update, Delete).

- **API SOAP : Un protocole de communication basé sur XML.**
  Les API SOAP utilisent généralement des appels de procédures distantes (RPC) pour effectuer des opérations.

Requêtes HTTP en Go

- `http.Get(url)`: Cette fonction envoie une requête GET à l'URL spécifiée et renvoie la réponse et une éventuelle erreur.
- `defer response.Body.Close()`: Le mot-clé `defer` est utilisé pour retarder l'exécution de la fermeture du corps de la réponse jusqu'à ce que la fonction courante soit terminée.
- `ioutil.ReadAll(response.Body)`: Cette fonction lit tout le corps de la réponse HTTP et renvoie les données sous forme de slice de bytes (`[]byte`)

```
package main
import (
	"fmt"
	"io/ioutil"
	"net/http"
)
func main() {
	url := "https://api.example.com/data"
	response, err := http.Get(url)
	if err != nil {
		fmt.Println("Erreur lors de la requête HTTP :", err)
		return
	}
	defer response.Body.Close()
	body, err := ioutil.ReadAll(response.Body)
	if err != nil {
		fmt.Println("Erreur lors de la lecture de la réponse :", err)
		return
	}
	fmt.Println(string(body))
}
```

Gérer les erreurs

La gestion des erreurs est cruciale pour garantir un fonctionnement fiable de votre application.

Dans l'exemple précédent, chaque opération qui pourrait générer une erreur est suivie d'une vérification d'erreur, comme dans les lignes `if err != nil { ... }`

```
package main
import (
        "fmt"
        "io/ioutil"
        "net/http"
)
func main() {
        url := "https://api.example.com/data"
        response, err := http.Get(url)
        if err != nil {
                fmt.Println("Erreur lors de la requête HTTP :", err)
                return
        }
        defer response.Body.Close()
        body, err := ioutil.ReadAll(response.Body)
        if err != nil {
                fmt.Println("Erreur lors de la lecture de la réponse :", err)
                return
        }
        fmt.Println(string(body))
}
```

MERCI.