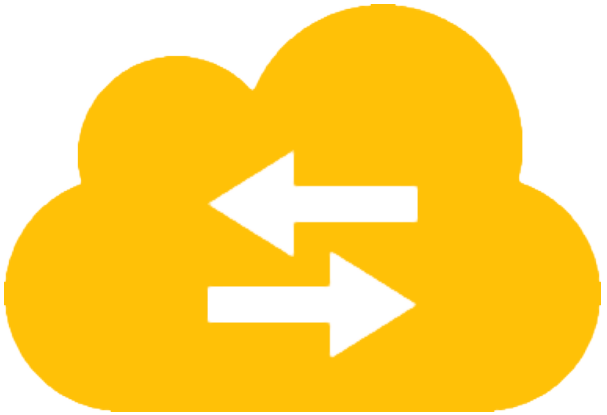
Développer une API REST



en Go

Benoît Masson
benoit@open-agora.com

Golang Rennes, jeudi 15 juin 2017

API REST

- **API** : *Application Programming Interface*
  - manipulation distante d'objets
  - selon une interface de transfert (support, chemins, paramètres, données) bien définie

- **REST** : *Representational State Transfer*
  - API web
  - un **style**, pas un protocole
  - une **ressource** = une **URL** (sans état)
  - une **action** = une **méthode** HTTP

Limitation de responsabilité

Disclaimer 1

Je ne suis pas un gourou des architectures REST… certains choix peuvent être discutés.

Disclaimer 2

Nous n'allons aborder que les aspects HTTP (pas d'ORM, …)

Disclaimer 3

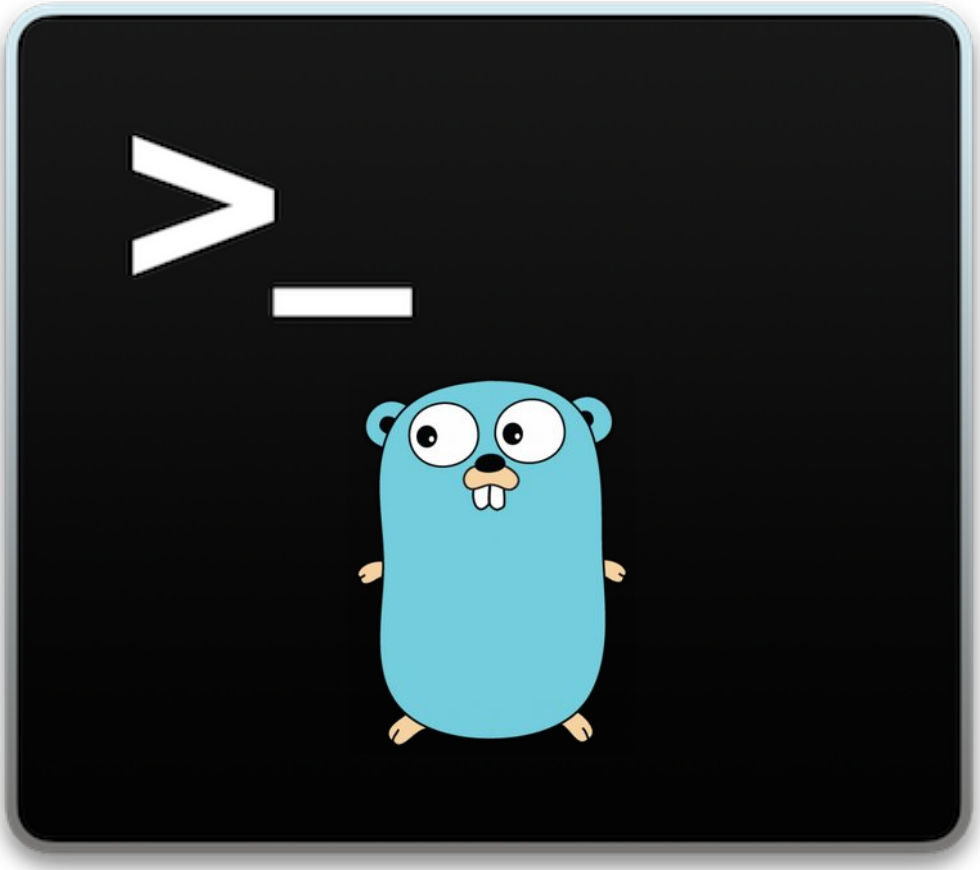
Je ne suis pas bricoleur…

open agora

1.

À la main

Package `http`



Package `http` : exemple

```
http.HandleFunc("/users", func(w http.ResponseWriter, req *http.Request) {
    // request filtering
    if req.Method != http.MethodGet {
        w.WriteHeader(http.StatusMethodNotAllowed)
        return
    }

    // get users
    allUsers := data.GetAll()

    // return users
    js, err := json.Marshal(allUsers)
    if err != nil {
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    w.Header().Set("Content-Type", "application/json")
    w.Write(js)
})
```

Package `http` : récap

- Pas de limitations techniques, mais un peu laborieux...

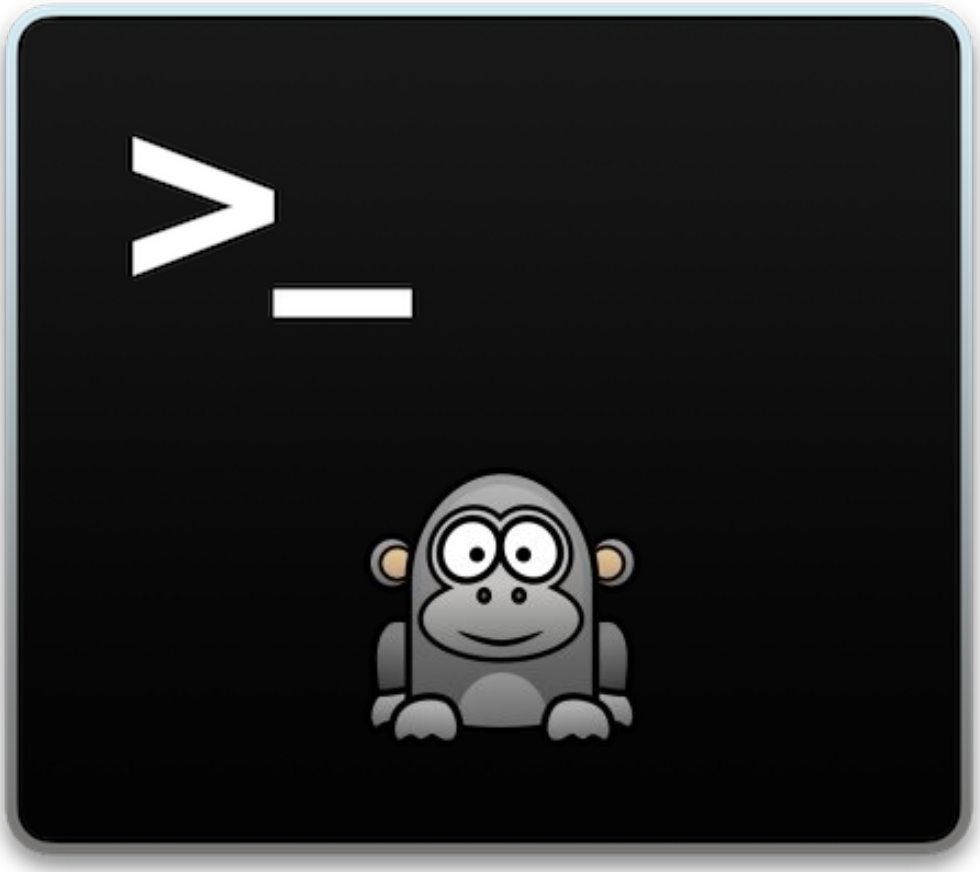
open agora

2.

Avec des outils génériques

Package `mux`

- http://www.gorillatoolkit.org/pkg/mux



Package mux : exemple

```
r.HandleFunc("/users", listUsers).Methods(http.MethodGet)
r.HandleFunc("/users/{id:[0-9]+}", getUser).Methods(http.MethodGet)
r.HandleFunc("/users", createUser).Methods(http.MethodPost)
```

```
func getUser(w http.ResponseWriter, req *http.Request) {
    // get ID from path
    vars := mux.Vars(req)
    id, _ := strconv.Atoi(vars["id"])

    // get user
    user := data.Get(data.Key(id))
    if user == nil {
        w.WriteHeader(http.StatusNotFound)
        w.Write([]byte(fmt.Sprintf("No user found with id %d", id)))
        return
    }

    // return user
    sendJSONResponse(user, w)
}
```

Package `mux` : récap

- Approche bas niveau raisonnable, simple et pas trop verbeuse…

- Un tuto assez complet :

  `https://semaphoreci.com/community/tutorials/building-and-testing-a-rest-api-in-go-with-gorilla-mux-and-postgresql`

open agora

3.

Avec un framework

Package beego

- https://beego.me/

Package beego : exemple

```
beego.Router("/users", &userController{},
             "get:ListUsers;post:CreateUser")
beego.Router("/users/:id:int", &userController{},
             "get:GetUser")
```

```
func (u *userController) GetUser() {
    // get ID from path
    id, _ := strconv.Atoi(u.Ctx.Input.Param(":id"))

    // get user
    user := data.Get(data.Key(id))
    if user == nil {
        u.Abort(strconv.Itoa(http.StatusNotFound))
    }

    // return user
    u.Data["json"] = &user
    u.ServeJSON()
}
```

Package beego : récap

- Approche plus haut niveau
  - plus compliqué à prendre en main...
- Pas forcément de gain de productivité pour une simple API
- Performances réduites

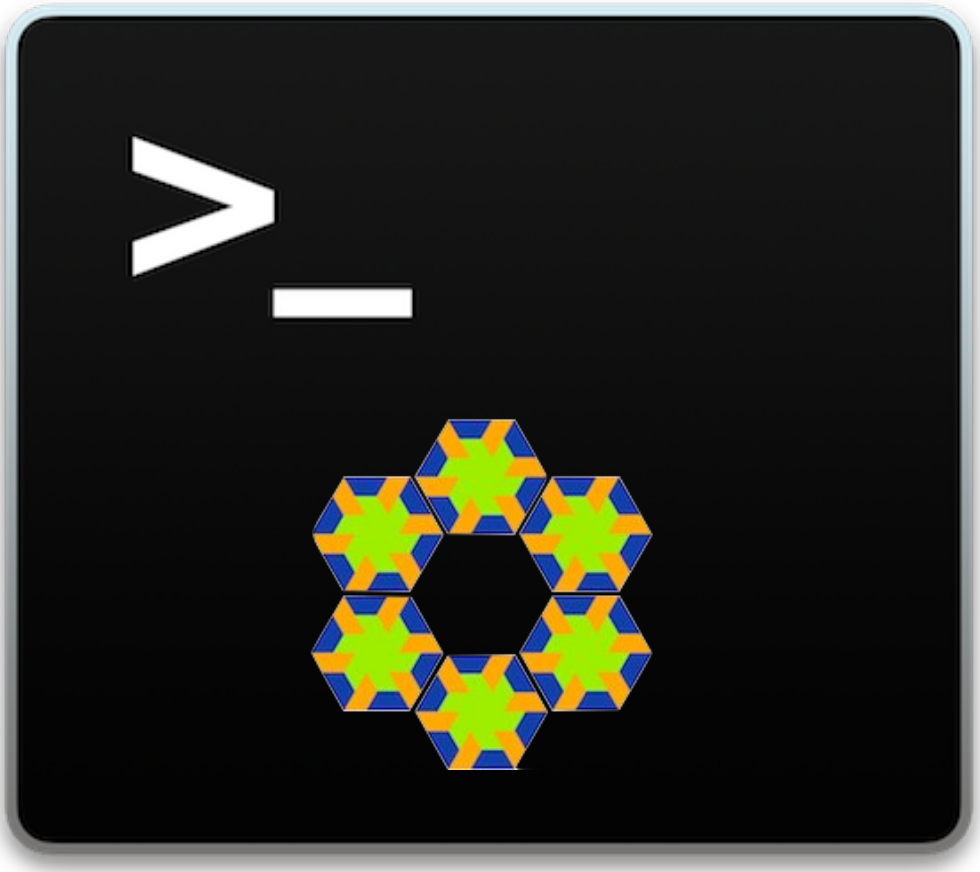
open agora

4.

Avec un outil dédié

Package `go-restful`

- https://github.com/emicklei/go-restful



Package `go-restful` : exemple

```
ws.Path("/users").
    Consumes(restful.MIME_JSON).
    Produces(restful.MIME_JSON)
ws.Route(ws.GET("/").To(listUsers))
ws.Route(ws.GET("/{id}").To(getUser))
ws.Route(ws.POST("/").To(createUser))
```

```
func getUser(req *restful.Request, resp *restful.Response) {
    // get ID from path
    id, _ := strconv.Atoi(req.PathParameter("id"))

    // get user
    user := data.Get(data.Key(id))
    if user == nil {
        resp.WriteHeader(http.StatusNotFound)
        resp.Write([]byte(fmt.Sprintf("No user found with id %d", id)))
        return
    }

    // return user
    resp.WriteEntity(user)
}
```

Package `go-restful` : exemple

```
ws.Path("/users").
    Consumes(restful.MIME_JSON).
    Produces(restful.MIME_JSON)
ws.Route(ws.GET("/").To(listUsers))
ws.Route(ws.GET("/{id}").To(getUser))
ws.Route(ws.POST("/").To(createUser))
```

```
func getUser(req *restful.Request, resp *restful.Response) {
    // get ID from path
    id, _ := strconv.Atoi(req.PathParameter("id"))

    // get user
    user := data.Get(data.Key(id))
    if user == nil {
        resp.WriteHeader(http.StatusNotFound)
        resp.Write([]byte(fmt.Sprintf("No user found with id %d", id)))
        return
    }

    // return user
    resp.WriteEntity(user)
}
```

Package `go-restful` : récap

- Approche intrinsèquement REST
  - code structuré selon les ressources
  - puis par les chemins et méthodes d'accès
  - (dé)sérialisation auto des données

Fonctionnalités avancées

- Gestion transparente du **`Content-type`** (requête et réponse)
  - éventuelle compression
- Gestion des **paramètres**
  - Path, Query, Header, Body, Form
- Gestion des messages d'**erreur**

Documentation automatique

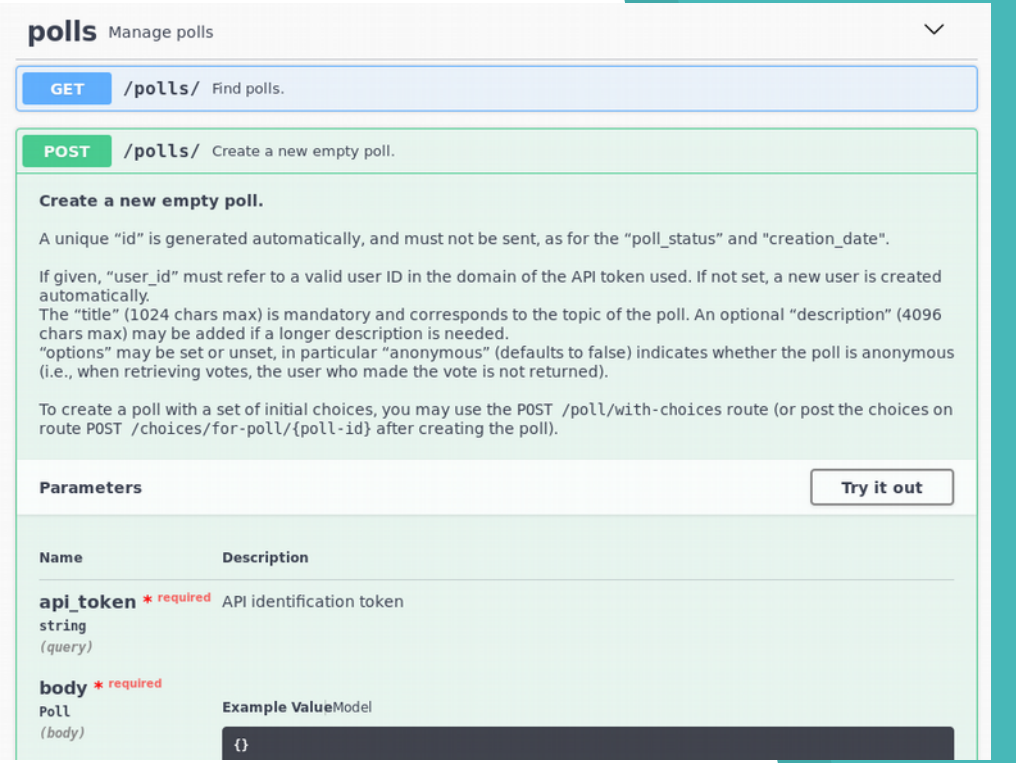
- https://www.openapis.org/
- http://swagger.io/swagger-ui/

Documentation automatique

```
ws.Route(ws.GET("/").To(findUsers).
    Metadata(openapi.KeyOpenAPITags, tags).
    Doc("gets all users list").
    Param(ws.QueryParameter("firstname", "filter users with this firstname").
        DataType("string").Required(false)).
    Writes([]data.User{}).
    Returns(http.StatusOK, "success, users list is returned", []data.User{}))
```

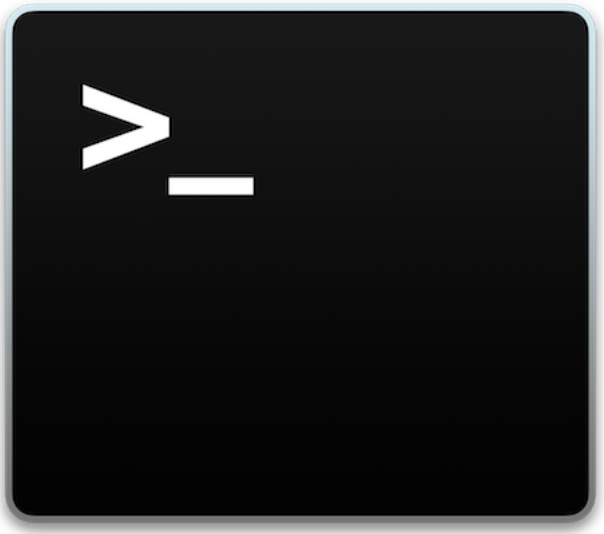
- Format **OpenAPI**

  (ou Swagger 1)
  - routes et modèles
  - pour visualisation avec **Swagger-UI**, par exemple



Filtres

- Filtre = *middleware* : fonction exécutée avant ou après le traitement de la requête
  - authentification
  - log
  - `OPTIONS`
  - recovery
  - ...



- Peuvent être activés à tous les niveaux (conteneur, ressource ou route)
- Peuvent définir des **attributs** de la requête

À améliorer

- Performances légèrement en retrait
- Aucun élément pour la pagination des résultats
- Pas d'outils pour faciliter les tests

open agora

VS

Conclusion

2 options recommandables

VS

- Performances
- Simplicité et liberté

- Résultat complet plus rapide
- Documentation

➡ Pourquoi ne pas tester les 2 ?

Merci !

Benoît Masson
benoit@open-agora.com