Sommaire

[ARCHITECTURE SERVEURS 2](#_Toc535424538)

[Présentation du MOTEUR API 3](#_Toc535424539)

[Terminologie : 3](#_Toc535424540)

[Design Pattern 3](#_Toc535424541)

[Dans les grandes lignes 3](#_Toc535424542)

[Schéma Example 3](#_Toc535424543)

# ARCHITECTURE SERVEURS

Requête http

Ex : http://54.38.183.200:**5000**/api/a2i/users

Réponse du SGBDR.

Connexion à la base. Requêtage SQL avec PHP.

Réponse

Ex : page HTML

Requête http

Ex : http://54.38.183.200:**3000**/a2i

Serveur OVH – 54.38.183.200

Serveur Express (JS)

Port : 3000

Moteur API

Port : 5000

Client web (Navigateur)

Server de données hébergeant un

SGBDR. (Système de Gestion de Base de Données Relationnelles).

(ex : OVH – 46.105.107.171)

Réponse

Ex : JSON

# Présentation du MOTEUR API

## Terminologie :

* Slug : c’est la partie de l’URL après avoir restante après y avoir retirer le protocole (HTTP ou HTTPS), le nom de domaine et l’extension . (ex : <http://mondomaine.com/user/ajouter> --> le slug: ‘/user/ajouter’) .
* Modèle : Il s’occupe de la partie logique de l’application. Il est chargé d’établir une connexion avec la base de données, et de restituer l’information au contrôleur.
* Contrôleur: il crée une instance du modele, execute une action dessus (méthode) et renvoie le résultat obtenu.

## Design Pattern

Le MOTEUR API est basé sur le design pattern MC (Modèle – Contrôleur). C’est le design pattern le plus connu dans la programmation orientée objet. Loin d’être une norme, cette pratique est reconnue car elle produit du code lisible et fortement maintenable.

## Dans les grandes lignes

Toutes les routes (URL) auxquelles l’API peut répondre sont définies dans un fichier de routes (routes.yml), auxquelles on y associe un contrôleur, et une action (méthode à appeler sur notre contrôleur) . Lorsqu’une requête vient frapper à la porte de notre serveur, si cette route correspond à une route définie, alors le contrôleur est instancié, et l’action (méthode) est appelée.

## Schéma Example

Réponse JSON

Voici le User 3

Appelle la méthode "getUser" sur l’objet "UserController". Renvoie des informations à l’index après réponse du modèle.

Création d’un objet User. (Modele)

Récupération des informations du User avec l’identifiant 3. Renvoie des infos au contrôleur.

index.php

J’ai trouvé une route qui correspond !

Recherche d’une route correspondante

routes.yml

"/api/a2i/users/{id}" :

"controller": User

"action" : getUser

Requête http

Ex : http://54.38.183.200:**5000**/api/a2i/users/3

Client web (Navigateur)

## Architecture des répertoires

* index.php *// fichier racine éxécuté*
* .htaccess *// fichier de surcharges des paramètres d’apache2*
* conf.yml *// definition des applications*
* core *// Dossier : contient les classes de fonctionnement du moteur.*
* <app\_name>-api *// Dossier : il contient les routes, les controllers et les modeles d’une application*

# LE SERVEUR WEB AVEC EXPRESS (JAVASCRIPT)

## Express, qu’est-ce que c’est ?

Express est un framework JavaScript qui permet la réalisation d’un serveur web en 4 lignes de codes, là ou il en faudrait peut être une centaine en JavaScript pur !

## Un serveur HTTP

Le serveur web monté sur le 54.38.183.200 sur le port 3000, n’écoute que pour le protocole http.

## Architecture des répertoires

* index.js *// fichier racine éxécuté*
* node\_modules *// Dossier : contient l’intégralité des dépendances nécessaires aux applications*
* package.json *// Fichier descriptif de l’application. (en l’occurrence le serveur)*
* package.lock.json *// Fichier autoalimenté et autogéré par npm (gestionnaire de paquets)*
* public *// Dossier : il contient les applications*
  + app1
    - build *// Dossier : contient le build applicatif.*
* routes *// Dossier : contient pour chaque application un fichier JS des routes.*
  + app1.js