

M2 PSB Javascript-PWA
Y. Stroppa
31/05/2023

Objectif :

Elaboration d'un formulaire à partir du plugin Everest, compréhension du stockage effectué dans la base de données, vérification du caractères non relationnel de ce type de stockage. Mise en place de l'échange en mode asynchrone entre le navigateur et le site web par des instructions de type fetch.

1/ Installation du plugin dans votre contexte Wordpress

Dans le cadre de votre projet, une fois votre instance de Wordpress démarrée vous devez incorporer le plugin Everest forms et Woody Snippets.

 <p>Everest Forms – Build Contact Forms, Surveys, Polls, Application Forms, and more with Ease!</p> <p>Extension légère et rapide de formulaire de contact sécurisé. Modèles prédéfinis responsifs. Créer un contact, une réservation, un paiement, un quiz, un sondage,...</p> <p><i>Par WPEverest</i></p> <p>★★★★★ (316) 100 000+ installations actives</p> <p>Dernière mise à jour : il y a 3 semaines ✓ Compatible avec votre version de WordPress</p> <p>Actif Plus de détails</p>	 <p>Woody code snippets – Insert Header Footer Code, AdSense Ads</p> <p>Insert Headers and Footers, executes PHP code, uses conditional logic to insert ads, text, media content and external service's code.</p> <p><i>Par Creative Motion, Will Bontrager Software, LLC</i></p> <p>★★★★★ (205) 80 000+ installations actives</p> <p>Dernière mise à jour : il y a 7 jours ✓ Compatible avec votre version de WordPress</p> <p>Mettre à jour Plus de détails</p>
---	---

A partir de ce plugin, veuillez créer un formulaire

2/ Utilisation du plugin dans votre CMS

A partir de ce module veuillez générer une "form" personnalisée avec les informations suivantes :
Demande d'intervention :

Name,
Email,
Paragraphe de texte
Subject
Message

Name	<input type="text"/>
Email	<input type="text"/>
Paragraphe de texte	<input type="text"/>
Subject	<input type="text"/>
Message	<input type="text"/>

3/ Insertion de ce formulaire dans le site Wordpress à partir d'un shortCode

Une fois le formulaire généré, veuillez créer une page qui permettra d'y accéder à partir de votre site.

4/ Utilisation du formulaire et observation des données :

Veuillez saisir à partir du formulaire des entrées pour constituer un jeu de données.

A partir de l'interface d'administration du site regardez les possibilités d'exploration existante dans la partie Everest

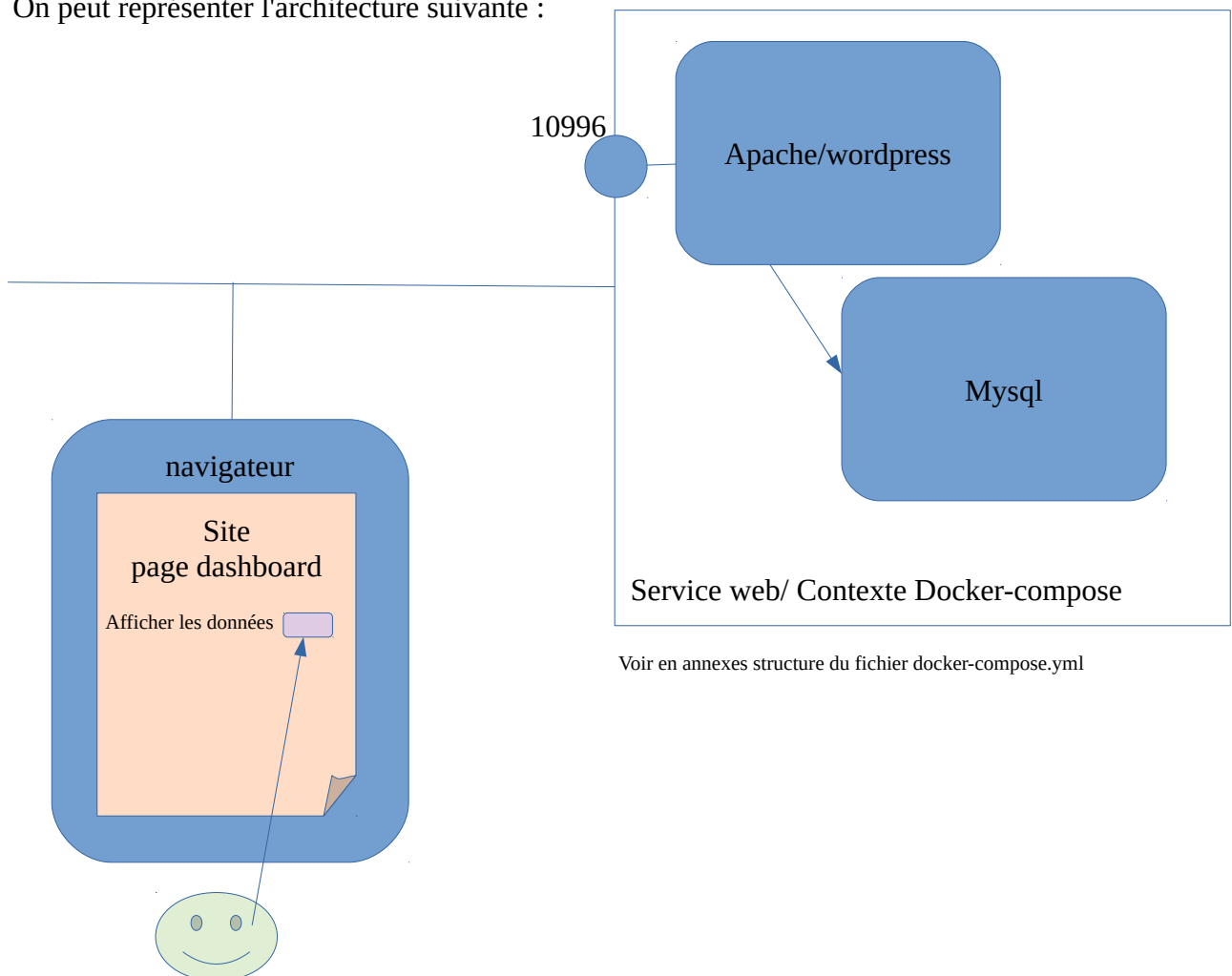
5/ Examinez le contenu de la base de données :

Veuillez vous connecter sur la base de données Mysql qui accompagne votre site et regardez où se trouvent les données que vous venez de saisir
dans quel table ,
et sous quelle forme ?

Analyse de la table et de sa structure ?.

6 / Comment mettre en place une page dans laquelle on souhaite afficher l'ensemble des éléments saisis sous forme d'un dashboard (graphe / tableau ou autres formes)

On peut représenter l'architecture suivante :



Voir en annexes structure du fichier docker-compose.yml

Modèles d'appel à partir de l'instruction fetch (anciennement ajax , xmlhttpRequest)

Exemple d'appel pour la récupération d'une image

```
function fetchImage() {
  fetch('/personnalisation/images/img1.png', {
    method: 'GET',
    mode: "same-origin",
    credentials: "same-origin",
    headers: {
      "Content-Type": "image/png",
      "charset": "utf-8"
    },
  })
  .then((response) => response.blob())
  .then((blob) => {
    const imageUrl = URL.createObjectURL(blob);
    const imageElement = document.createElement("img");
    imageElement.src = imageUrl;
    const container = document.getElementById("image-container");
    container.appendChild(imageElement);
  });
}
```

Création d'un objet de type img
Affectation du contenu
Accrochage au parent

Exemple d'appel pour l'invocation d'une ressource php

```
let formattedFormData={"donnee":1} ;
fetch("/personnalisation/get_expression_schema.php", {
  method: "POST",
  mode: "same-origin",
  credentials: "same-origin",
  headers: {
    "Content-Type": "application/json",
    "charset": "utf-8"
  },
  body: JSON.stringify(formattedFormData)
})
.then((res) => res.json())
.then((data) => {
  // faire le traitement de data
})
.catch((error) => console.log(error))
```

```
fetch("/personnalisation/evf_get_entries.php", {
  method: "GET",
  mode: "same-origin",
  credentials: "same-origin",
  headers: {
    "Content-Type": "application/json",
    "charset": "utf-8"
  },
})
.then((res) => res.json())
.then((data) => {
  // faire le traitement de data
})
.catch((error) => console.log(error))
```

Fonctionnement du fetch :

L'instruction fetch fonctionne en asynchrone et permet d'invoquer une ressource du côté du serveur (url) accessible dans le contexte wordpress. Cette ressource peut-être un fichier image, json ou l'appel et invocation d'un php. Lors de l'appel de cette ressource, on fixe le mode GET, POST et les paramètres qui l'accompagnent. Le fetch s'exécute comme une **promise** en javascript et retourne les résultats que l'on peut traiter de façon séquentiel à l'aide de l'instruction then une fois la ressource trouvée.

Donc pour commencer il faut créer la ressource du côté du serveur dans un répertoire spécifique de wordpress que l'on va nommer personnalisation. Dans ce répertoire, il faudra créer les fichiers qui vont permettre de mettre à disposition la ressource en exécutant une requête sql dans la base de données.

Premier mode : appel en mode GET

Le script php qu'il va falloir saisir doit se connecter à la base de données, et exécuter une requête SQL pour récupérer l'ensemble du contenu de la table.

Dans la contexte PHP de wordpress, on peut charger un ensemble de fonctionnalités et de variables à l'aide d'un require wp_load.php qui se trouve sur la racine du site. Ce fichier permet d'accéder à des définitions de variables et de fonctions php.

Par exemple : la variable wpdb (objet) qui permet d'avoir un descripteur ouvert sur la base de données à l'aide du compte wordpress par défaut.

Une fois que l'on a ce descripteur on peut exécuter des appels SQL sur la base de données et récupérer une seule ligne ou plusieurs

Pour la récupération d'un seul résultat

```
$data=$wpdb->get_row($wpdb->prepare("SELECT user_id, user_droits from table")) ;
```

Pour la récupération de plusieurs résultats

```
$data=$wpdb->get_results($wpdb->prepare("SELECT * FROM $table_name"));
```

Maintenant si on a besoin de paramétrer les requêtes on peut utiliser des variables afin de filtrer l'extractions en fonction de critères que l'on pourra transmettre en mode POST.

```
function opal_get_objets($user_id, $user_login,$key,$corpus) {  
    global $wpdb;  
    $table_name='alimcorp_objets';  
    $data=$wpdb->get_row($wpdb->prepare("SELECT user_id, user_droits, user_categ,meta_key,meta_value  
                                         FROM $table_name  
                                         WHERE meta_key like %s and  JSON_EXTRACT(meta_value,'$.Modele')=%s",$key,$corpus));  
    return $data;  
}
```

Plusieurs instructions de SQL permettent d'explorer des structures de type JSON dans les attributs des tables, de type JSON_EXTRACT, JSON_CONTAINS,(voir en annexes structure de table)

Quelques extractions à faire :

On ne veut que les Subject dans lesquels se trouvent :

A FAIRE :

Action A : Définir votre fichier php et le déposer sur le répertoire personnalisation de votre site web

Action B : écrire le script javascript pour effectuer l'appel fetch à ce fichier php.

Action C : à l'exécution du fetch veuillez afficher le résultat de l'appel afin de voir comment est la structure data récupérée on utilisera pour ça à un affichage dans le console.

Action D : on s'occupe des l'affichage des résultats dans notre contexte.

Affichage des résultats :

Plusieurs cas possibles dans un tableau ou dans une zone div

Cas de la zone div :

Pour s'occuper de l'affichage en mode ligne des éléments hors tableau, il faut faire l'inventaire des attributs à afficher

Inventaire des données et définition de l'affichage que l'on souhaite

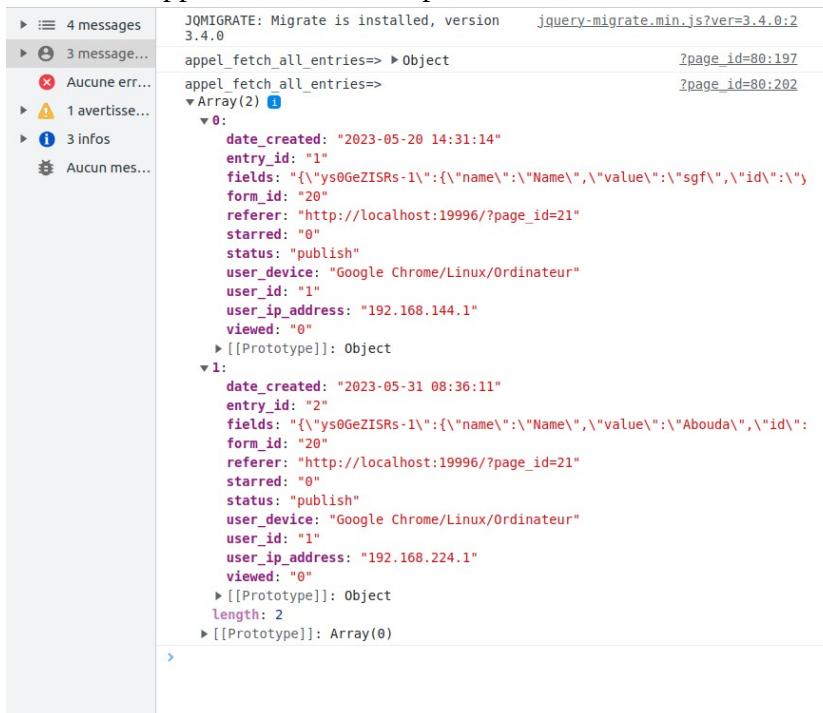
Contenu de la donnée à gérer à visualiser dans le contexte de développement de votre navigateur.

Une fois l'appel du php paramétré.

```
date_created : "2023-05-20 14:31:14",
entry_id: "1",
fields: "{
  \"ys0GeZISRs-1\": {\"name\": \"Name\", \"value\": \"sgf\", \"id\": \"ys0GeZISRs-1\", \"type\": \"first-name\", \"meta_key\": \"first_name_4789\"},
  \"LbH5NxasXM-2\": {\"name\": \"Email\", \"value\": \"yvan.stroppa1@univ-orleans.fr\", \"id\": \"LbH5NxasXM-2\", \"type\": \"email\", \"meta_key\": \"email_1036\"},
  \"mlXm1HAU90-5\": {\"name\": \"Paragraphe de texte\", \"value\": \"qsdfqsdf\", \"id\": \"mlXm1HAU90-5\", \"type\": \"textarea\", \"meta_key\": \"paragraphe_de_texte_4320\"},
  \"66FR384cge-3\": {\"name\": \"Subject\", \"value\": \"qsdfqsdf\", \"id\": \"66FR384cge-3\", \"type\": \"text\", \"meta_key\": \"single_line_text_1212\"},
  \"yhGx3FOwr2-4\": {\"name\": \"Message\", \"value\": \"qsdfqsdf\", \"id\": \"yhGx3FOwr2-4\", \"type\": \"textarea\", \"meta_key\": \"paragraph_text_1823\"}},
form_id: \"20\",
referer: \"http://localhost:19996/?page_id=21\",
starred: \"0\",
status: \"publish\",
user_device: \"Google Chrome/Linux/Ordinateur\",
user_id: \"1\",
user_ip_address: \"192.168.144.1\",
viewed: \"0\"
```

```
date_created: \"2023-05-31 08:36:11\",
entry_id: \"2\",
fields: \"{
  \"ys0GeZISRs-1\": {\"name\": \"Name\", \"value\": \"Abouda\", \"id\": \"ys0GeZISRs-1\", \"type\": \"first-name\", \"meta_key\": \"first_name_4789\"},
  \"LbH5NxasXM-2\": {\"name\": \"Email\", \"value\": \"lotfi.abouda@univ-orleans.fr\", \"id\": \"LbH5NxasXM-2\", \"type\": \"email\", \"meta_key\": \"email_1036\"},
  \"mlXm1HAU90-5\": {\"name\": \"Paragraphe de texte\", \"value\": \"ceci est le paragraphe de texte\", \"id\": \"mlXm1HAU90-5\", \"type\": \"textarea\", \"meta_key\": \"paragraphe_de_texte_4320\"},
  \"66FR384cge-3\": {\"name\": \"Subject\", \"value\": \"Essai de saisie\", \"id\": \"66FR384cge-3\", \"type\": \"text\", \"meta_key\": \"single_line_text_1212\"},
  \"yhGx3FOwr2-4\": {\"name\": \"Message\", \"value\": \"Tout va bien\", \"id\": \"yhGx3FOwr2-4\", \"type\": \"textarea\", \"meta_key\": \"paragraph_text_1823\"}},
form_id: \"20\",
referer: \"http://localhost:19996/?page_id=21\",
starred: \"0\",
status: \"publish\",
user_device: \"Google Chrome/Linux/Ordinateur\",
user_id: \"1\",
user_ip_address: \"192.168.224.1\",
viewed: \"0\"
```

Après exécution de l'appel de fetch et récupération des résultats



```
JQMIGRATE: Migrate is installed, version 3.4.0
appel_fetch_all_entries=> Object
appel_fetch_all_entries=>
  Array(2)
    0: {
      date_created: "2023-05-20 14:31:14"
      entry_id: "1"
      fields: {"ys0GeZISRs-1": {"name": "Name", "value": "sgf", "id": ""}}
      form_id: "20"
      referer: "http://localhost:19996/?page_id=21"
      starred: "0"
      status: "publish"
      user_device: "Google Chrome/Linux/Ordinateur"
      user_id: "1"
      user_ip_address: "192.168.144.1"
      viewed: "0"
    }
    1: {
      date_created: "2023-05-31 08:36:11"
      entry_id: "2"
      fields: {"ys0GeZISRs-1": {"name": "Name", "value": "Abouda", "id": ""}}
      form_id: "20"
      referer: "http://localhost:19996/?page_id=21"
      starred: "0"
      status: "publish"
      user_device: "Google Chrome/Linux/Ordinateur"
      user_id: "1"
      user_ip_address: "192.168.224.1"
      viewed: "0"
    }
  
```

Donc si on veut afficher tout ça sous une forme prédéfinie, il faut d'abord la définir et ensuite voir comment on peut la paramétrer.

Du style pour chaque entrée classée dans l'ordre d'arrivée

date	Subject	Email
Paragraphe de texte		
Message		

Comment générer de manière automatique cette affichage ?

Solution possible :

On va créer une structure de type string dans laquelle on va prototyper les éléments d'affichage que l'on viendra substituer par la suite pour les adapter à chaque élément à afficher et ensuite on les ajoute à la div résultat.

Mise en place :

```
const modele="<div class='row col-sm-12'><div class='col col-sm-3' style='background-color:lightblue'>Date</div><div class='col col-sm-3' style='background-color:lightblue'>Subject</div></div>";

const affichage=function() {
    let id=0;
    entries.forEach((e)=> {
        let champs=JSON.parse(e.fields);
        let mod=model.replace("Date",e.date_created).replace("Subject",champs["66FR384cge-3"]["value"]);
        addElement("resultat","p","element_"+id,mod);
        id++;
    });
}
```

A vous de compléter la suite pour ajouter les autres champs dans cet affichage.

Partie PHP

Ecriture du script php du côté du serveur pour effectuer un appel Fetch par la suite

Dans le répertoire personnalisation du site wordpress veuillez créer les fichiers suivants :
evf_get_entries.php qui sera utilisé en mode GET

```
<?php
require('func_lib_inc.php');

ecrire_log("evf_get_entries.php == recuperation de toutes les entrees" );
function evf_get_all_entries() {
    global $wpdb;
    $table_name='wp_evf_entries';
    $data=$wpdb->get_results($wpdb->prepare("SELECT * FROM $table_name"));
    return $data;
}

$人id=get_current_user_id();
$人login=wp_get_current_user();
ecrire_log("evf_get_entries.php ==> ".$人login->user_login);
$人Data=evf_get_all_entries();
$人comma_separated = json_encode($人Data);
ecrire_log("evf_get_entries.php ==> user:".$人id." recuperation des objets".$人comma_separated);
echo json_encode(array("success"=>"yes","details"=> $人Data));
//echo $人specialities;
?>
```

Accompagné par la fichier d'inclusion : func_lib_inc.php

```
<?php
// -----
// librairies des fonctions de base utilisees par les autres scripts
// realisation : CNRS Y. stroppe
// -----

require('../wp-load.php');

function ecrire_log($人errtxt) {
    $人fp = fopen('log.txt', 'a+');
    fseek($人fp, SEEK_END);
    $人nouverr = $人errtxt . "\r\n";
    fputs($人fp, $人nouverr);
    fclose($人fp);
}
?>
```

Plusieurs vérifications peuvent être faites à ce niveau :

Vérification de la requête SQL : vérifiez sous mysql sql que la requête est correcte

Vérification erreur de syntaxe php : exécuter du côté du serveur la commande sous le répertoire personnalisation suivante : php evf_get_entries.php

Du côté de la page et du contexte navigateur, il faut élaborer une page nommée Dashboard qui va nous permettre d'exploiter les données saisies par les utilisateurs :

Javascript

```
<script>
    let entries=[]; // tableau des entrees
    const appel_fetch_all_entries=function() {
        let mots_valides=[];
        fetch("/personnalisation/evf_get_entries.php", {
            method: "GET",
            mode: "same-origin",
            credentials: "same-origin",
            headers: {
                "Content-Type": "application/json",
                "charset":"utf-8"
            },
        })
        .then((res) => res.json())
        .then((data) => {
            console.log("appel_fetch_all_entries=>",data);
            for (let entree of data.details) {
                entries.push(entree);
            }
            // attention au champd fields des éléments à parser si besoin
            // let structure_entries=JSON.parse(data.details.fields);
            console.log("appel_fetch_all_entries=>",entries);
        })
        .catch((error) => console.log(error))
    }

</script>

<h2> Liste des éléments saisis par le formulaire </h1>
<hr />
<button id="btn" onclick="appel_fetch_all_entries();"> Importer /button>
<div id="resultats"></div>
```

Petit fonction utilitaire pour ajouter un contenu à partir d'un point d'ancrage

```
const addElement=function(parentId, elementType, elementId, contenu,display="display:block") {
    var p = document.getElementById(parentId);
    var newElement = document.createElement(elementType);
    newElement.setAttribute("id", elementId);
    newElement.innerHTML = contenu;
    newElement.setAttribute("style", display);
    p.appendChild(newElement);
}
```

Annexes :

A / fichier docker-compose.yml

```
version: "3.3"

services:
  db:
    image: mysql:5.7
    volumes:
      - /home/ystroppa/PSB/2023/M2/grp6/db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - /home/ystroppa/PSB/2023/M2/grp6/wordpress_data:/var/www/html
    ports:
      - "19996:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

  phpmyadmin:
    image: phpmyadmin
    restart: always
    ports:
      - 18084:80
    environment:
      - PMA_ARBITRARY=1
      - PMA_HOST=db
```

B / Structure de table mysql

```
mysql> desc wp_evf_entries;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| entry_id | bigint(20) unsigned | NO | PRI | NULL | auto_increment |
| form_id | bigint(20) unsigned | NO | MUL | NULL | |
| user_id | bigint(20) unsigned | NO | | NULL | |
| user_device | varchar(100) | NO | | NULL | |
| user_ip_address | varchar(100) | YES | | | |
| referer | text | NO | | NULL | |
| fields | longtext | YES | | NULL | |
| status | varchar(20) | NO | | NULL | |
| viewed | tinyint(1) | NO | | 0 | |
| starred | tinyint(1) | NO | | 0 | |
| date_created | datetime | NO | | 0000-00-00 00:00:00 | |
+-----+-----+-----+-----+-----+-----+

```

Analyse du contenu de la table

```
{
  "ys0GeZISRs-1":{
    "name":"Name",
    "value":"sgf",
    "id":"ys0GeZISRs-1",
    "type":"first-name",
    "meta_key":
      "first_name_4789"},
  "LbH5NxasXM-2":{
    "name":"Email",
    "value":"yvan.stroppa1@univ-orleans.fr",
    "id":"LbH5NxasXM-2",
    "type":"email",
    "meta_key":"email_1036"},
  "mlXm1HAU90-5":{
    "name":"Paragraphe de texte",
    "value":"qsdfqsdf",
    "id":"mlXm1HAU90-5",
    "type":"textarea",
    "meta_key":"paragraphe_de_texte_4320"},
  "66FR384cge-3":{
    "name":"Subject",
    "value":"qsdfqsdf",
    "id":"66FR384cge-3",
    "type":"text",
    "meta_key":"single_line_text_1212"},
  "yhGx3FOwr2-4":{
    "name":"Message",
    "value":"qsdfqsdf",
    "id":"yhGx3FOwr2-4",
    "type":"textarea",
    "meta_key":"paragraph_text_1823"}
}
```

```
select JSON_EXTRACT(fields,"$.\"ys0GeZISRs-1\"") from
wp_evf_entries;
+-----+
| JSON_EXTRACT(fields,"$.\"ys0GeZISRs-1\"")
|
+-----+
| {"id": "ys0GeZISRs-1", "name": "Name", "type": "first-name",
"value": "sgf", "meta_key": "first_name_4789"} |
| {"id": "ys0GeZISRs-1", "name": "Name", "type": "first-name",
"value": "Abouda", "meta_key": "first_name_4789"} |
+-----+
2 rows in set (0.00 sec)

mysql> select JSON_EXTRACT(fields,"$.\"ys0GeZISRs-
1\".value") from wp_evf_entries;
+-----+
| JSON_EXTRACT(fields,"$.\"ys0GeZISRs-1\".value") |
+-----+
| "sgf" |
| "Abouda" |
+-----+
```

Attention au caractère - il faut rajouter des " pour l'expression