Eliott BENAIM – Anass HMADOUCH

URL du GIT : <https://github.com/Govagnoli/API-REST-Blog.git>

Compte-rendu   
R4.01 - Architecture logicielle Projet :

Conception et développement d’API REST pour la gestion d’articles

[MCD 3](#_Toc131051192)

[Spécification des API REST du projet 3](#_Toc131051193)

[JW Token 3](#_Toc131051194)

[L’authentification 3](#_Toc131051195)

[Valeurs entrées 3](#_Toc131051196)

[Valeurs retour : Cas de succès 3](#_Toc131051197)

[Erreur Renvoyées 3](#_Toc131051198)

[Description Traitement 3](#_Toc131051199)

[GET 4](#_Toc131051200)

[Récupération d’un article 4](#_Toc131051201)

[Valeurs entrées 4](#_Toc131051202)

[Valeurs retour : Cas publisher (succès) 4](#_Toc131051203)

[Valeurs retour : Cas Modérateur(succès) 4](#_Toc131051204)

[Valeurs retour : Cas anonyme(succès) 4](#_Toc131051205)

[Erreur Renvoyées 4](#_Toc131051206)

[Description Traitement 4](#_Toc131051207)

[Récupération de ses articles 5](#_Toc131051208)

[Valeurs entrées 5](#_Toc131051209)

[Valeurs retour : Cas publisher (succès) 5](#_Toc131051210)

[Erreur Renvoyées 5](#_Toc131051211)

[Description Traitement 5](#_Toc131051212)

[POST 6](#_Toc131051213)

[Valeurs entrées 6](#_Toc131051214)

[Valeurs retour : Cas de succès 6](#_Toc131051215)

[Erreur Renvoyées 6](#_Toc131051216)

[Description Traitement 6](#_Toc131051217)

[DELETE 6](#_Toc131051218)

[Valeurs entrées 6](#_Toc131051219)

[Valeurs retour : Cas de succès 6](#_Toc131051220)

[Erreur Renvoyées 6](#_Toc131051221)

[Description Traitement 6](#_Toc131051222)

[PATCH 8](#_Toc131051223)

[Vote 8](#_Toc131051224)

[Valeurs entrées 8](#_Toc131051225)

[Valeurs retour : Cas de succès 8](#_Toc131051226)

[Erreur Renvoyées 8](#_Toc131051227)

[Description Traitement 8](#_Toc131051228)

[Modification Article 8](#_Toc131051229)

[Valeurs entrées 8](#_Toc131051230)

[Valeurs retour 8](#_Toc131051231)

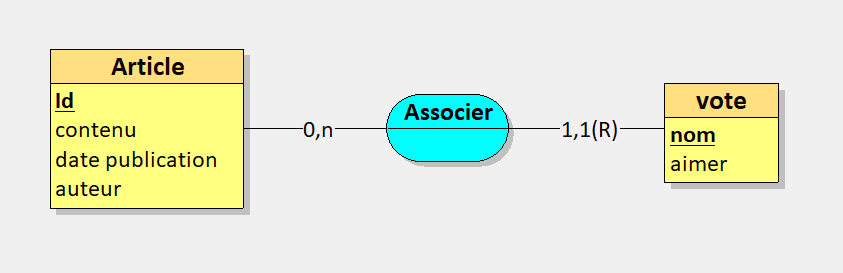
[Erreur Renvoyées 8](#_Toc131051232)

[Description Traitement 9](#_Toc131051233)

[User Story 9](#_Toc131051234)

MCD

Voici notre model conceptuel de données.



# Spécification des API REST du projet

JW Token

Notre application fournis trois rôles différents. Le rôle modérateur, publisher et anonyme.  
Les rôles modérateurs et publisher ont besoin de passer par notre serveur d’authentification avant de pouvoir accéder à notre service. Quant au rôle d’anonyme, dès lors que l’utilisateur ne s’authentifie pas, à chaque requête fait sans jeton, le rôle d’anonyme lui sera attribué.

## L’authentification

Voici l’url pour pouvoir s'authentifier : <http://localhost/R4.01/Projet/ServeurAuthentification.php>

### Valeurs entrées

L’authentification se fait avec une méthode HTTP POST. Dans le corp de la méthode il faut préciser son username et son password comme ceci :  
{"username":"XXX", "password":"XXX"}

### Valeurs retour : Cas de succès

200, "Vous êtes bien authentifié en tant que X Voici votre token : Y

X : le rôle de l’utilisateur  
Y : Le token renvoyé

### Erreur Renvoyées

* 401, "Mot de passe ou identifiant incorrect."
* 501, "Erreur d'authentification. Pour s'authentifier veuillez faire une méthode post. Vous pouvez lire la documention pour plus d'informations"

### Description Traitement

On regarde si l’username existe. S’il existe on regarde si son mot de passe concorde.  
Puis si le couple mot de passe nom utilisateur existe alors on renvoie le rôle et le token de l’utilisateur.

# GET

## Récupération d’un article

### Valeurs entrées

Il faut renseigner dans l’url de la méthode GET l’identifiant. Comme ceci :   
- http://localhost/R4.01/Projet/Serveur.php?id=1

### Valeurs retour : Cas publisher (succès)

200, "Résultat de la recherche de l'identifiant X”, Y  
X : l’identifiant associé à l’article créé  
Y : L’auteur, la date publication, le contenu, nombre de likes, nombres de dislikes

### Valeurs retour : Cas Modérateur(succès)

200, "Résultat de la recherche de l'identifiant X”, Y  
X : l’identifiant associé à l’article créé  
Y : L’auteur, la date publication, le contenu, Liste des utilisateurs ayant liké, nombre de likes, Liste des utilisateurs ayant disliké, nombres de dislikes

### Valeurs retour : Cas anonyme(succès)

200, "Résultat de la recherche de l'identifiant X”, Y  
X : l’identifiant associé à l’article créé  
Y : L’auteur, la date publication, le contenu

### Erreur Renvoyées

* 400, "L'identifiant renseigné n'existe pas."
* 400, "La syntaxe de la requête est erronée"
* 500, "Une erreur est survenue pendant l'exécution de la requête."
* 500, "Erreur de Token."

### Description Traitement

On regarde le rôle de l’utilisateur via le token.

**Si l’utilisateur est un publisher :**   
On récupère l’auteur, la date publication, le contenu, le nombre de likes et le nombres de dislikes. On renvoie les données si aucune erreur s’est glissée pendant la récupération des informations.

**Si l’utilisateur est un modérateur :**  
On récupère l’auteur, la date publication, le contenu, la liste des utilisateurs ayant liké, le nombre de likes, la liste des utilisateurs ayant disliké et nombres de dislikes. On renvoie les données si aucune erreur s’est glissée pendant la récupération des informations

## Récupération de ses articles

### Valeurs entrées

Aucune valeur est attendu en entrer. Il faut juste faire une méthode GET.  
- http://localhost/R4.01/Projet/Serveur.php

### Valeurs retour : Cas publisher (succès)

200, "Voici tous les articles de l'utilisateur X", Y  
X : Le nom de l’utilisateur  
Y : L’identifiant, l’auteur, la date publication et le contenu

### Erreur Renvoyées

* 500, "Une erreur est survenue pendant l'exécution de la requête."
* 500, "Erreur de Token."
* 401, "Permission non accordée. Veuillez-vous connecter.", "Rôle : anonyme”
* 401, "Permission non accordée, il faut être publisher pour consulter ses articles", "Rôle : Modérateur”
* 400, "La syntaxe de la requête est erronée"
* 500, "Une erreur est survenue pendant l'exécution de la requête."

### Description Traitement

On regarde le rôle de l’utilisateur via le token.

**Si l’utilisateur est un publisher :**   
On lui renvoie directement ses articles.

**Si l’utilisateur est un modérateur :**  
On envoi l’erreur : Permission non accordée, il faut être publisher pour consulter ses articles

**Si l’utilisateur est un publisher:**  
On envoi l’erreur : Permission non accordée. Veuillez-vous connecter

# POST

## Valeurs entrées

Pour la création d’un article, l’utilisateur doit renseigner dans le body, le contenu de l’article qu’il souhaite créer.

{“contenu” : “Contenu de la l’article”}

## Valeurs retour : Cas de succès

200, "Article crée avec succès", "ID de l'article : X”

## Erreur Renvoyées

Liste des potentiels erreurs :

* 500, "Erreur de Token."
* 401, "Permission non accordée", "Rôle : " avec le rôle de l’utilisateur
* 400, "La syntaxe de la requête est erronée"
* 500, "Une erreur est survenue pendant l'exécution de la requête."
* 400, "Contenu de l'article ne respectant pas les règles (plus de 140 caractères ou syntaxe invalide)." (Contrainte de la base de données)

## Description Traitement

La méthode POST permet la création d’un article d’un utilisateur “publisher”. “moderator” et “anonyme” n’ont pas les droits nécessaires à la création d’un article. Le “publisher” doit renseigner dans le body le contenu de l’article avec une limite de 140 caractères.

# DELETE

## Valeurs entrées

Il faut rentrer l’identifiant de l’article à supprimer dans l’url de la méthode DELETE.   
Exemple : http://localhost/R4.01/Projet/Serveur.php?id=2

## Valeurs retour : Cas de succès

200, "Suppression validée", "ID : X”

X : identifiant de l’article supprimé

## Erreur Renvoyées

* 401, "Permission non accordée. Veuillez-vous connecter"
* 400, "La syntaxe de la requête est erronée"
* 400, "La syntaxe de la requête est erronée"
* 404, "Veuillez renseigner un ID existant"
* 500, "Une erreur est survenue pendant l'exécution de la requête."

## Description Traitement

On récupère le rôle via le token. Puis si l’identifiant n’est pas renseigné dans l’url. Et on vérifie si l’utilisateur n’est pas anonyme.

**Si l’utilisateur est un publisher :**  
On vérifie que l’article à supprimer est bien supprimé par le même auteur. Puis on supprime l’article et on renvoie un message de succès.

**Si l’utilisateur est un modérateur :**   
Il peut supprimer n’importe quels articles si les informations envoyées sont correctes. (Identifiant valide etc.)

# PATCH

## Vote

### Valeurs entrées

Vous devez spécifier dans l’url de la méthode PATCH l’identifiant de l’article à modifier.   
Vous devez spécifier dans le body de la méthode PATCH un Json au format suivant :   
{“aimer”:”true”} ou { “aimer”:”false”}

### Valeurs retour : Cas de succès

200 "Votre like est bien pris en compte."   
Ou   
200 "Votre dislike est bien pris en compte."

### Erreur Renvoyées

Liste des erreurs pouvant être renvoyés :

* 400, "La syntaxe de la requête est erronée"
* 400, "Il manque des données dans le body. Veuillez préciser si vous avez liké ou disliké un article."
* 401, "Permission non accordée, un utilisateur ne peut pas liker ou disliker ses posts."
* 500, "Une erreur est survenue pendant l'exécution de la requête."
* 404, "Veuillez renseigner un ID existant", "ID : X introuvable"

### Description Traitement

Si l’utilisateur a renseigné l’identifiant dans l’url et la valeur aimer dans le body. Alors il rentre dans le traitement du vote.

On vérifie si l’identifiant renseigné existe dans notre base de données. On vérifie s’il aime ou pas son propre post. On vérifie que la valeur renseignée dans le body n’est pas null ou si la valeur n’est pas un booléen. Si toutes les vérifications passent. Alors on regarde s’il like ou dislike le post en question et on ajoute ou retire le like sur le post. Et on renvoi un message de retour.

Cependant, s’il like un post déjà liké, son like est retiré. S’il like un post alors qu’il avait déjà disliké, alors, son dislike est retiré et son like ajouté. Ces propriétés sont réciproques pour le dislike.

## Modification Article

### Valeurs entrées

Quant à la modification d’un article, il faut préciser dans l’url l’identifiant de l’article à modifier, puis dans le body, doit être écrit le nouveau contenu de l’article sous format JSON :

{“contenu” : “Nouveau contenu de la l’article”}

### Valeurs retour

200 “L'article a bien été modifié, ID : “ avec l’ID retourné

### Erreur Renvoyées

Liste des potentiels erreurs :

* 401, "Permission non accordée, un utilisateur ne peut modifier un article dont il n'est pas l'auteur."
* 401, "Permission non accordée" (quand il s’agit d’un moderator ou d’un anonyme)
* 404, "Veuillez renseigner un ID existant"
* 400, "La syntaxe de la requête est erronée"
* 500, "Une erreur est survenue pendant l'exécution de la requête."

### Description Traitement

Dès lors que l’utilisateur envoie une requête en précisant un ID dans l’url et en précisant dans le body de la requête le nouveau contenu de l’article, il entre dans le traitement de la modification d’un article.

On vérifie dans un premiers temps son rôle, pour ne permettre la modification d’un article que s’il s’agit d’un publisher. Par la suite, est vérifié si l’auteur de l’article correspond à l’username du publisher.

On vérifie l’existence de l’ID dans la base de données, s’il est existant, l’article est modifié.

# User Story

## Story utilisateur :

|  |
| --- |
| Story Authentification |
| **En tant** **que** Publisher ou modérateur **Je désire** m’authentifier avec le serveur d’authentification  **Quand** je le souhaite  **Afin d'**avoir un JW Token | |

|  |
| --- |
| Story Créer un article |
| **En tant** **que** publisher **Je désire** poster un article  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story voir un article |
| **En tant** **que** publisher, modérateur ou anonyme **Je désire** avoir accès aux articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story supprimer un article (Publisher) |
| **En tant** **que** Publisher **Je désire** supprimer l’un de mes articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story supprimer un article (modérateur) |
| **En tant** **que** modérateur **Je désire** supprimer n'importe quels articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story modification d’un article (publisher) |
| **En tant** **que** Publisher **Je désire** modifier n'importe quels de mes articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story vote pour article (publisher - like) |
| **En tant** **que** Publisher **Je désire** like n'importe quels articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |

|  |
| --- |
| Story vote pour article (publisher - dislike) |
| **En tant** **que** Publisher **Je désire** dislike n'importe quels articles  **Quand** je le souhaite  **Afin de** profiter du service proposé | |