Rapport de projet

R4.01 - Architecture logicielle

Parcours : C
Groupe : E
Membres : Alexandre Mole, Alexandre Calmet

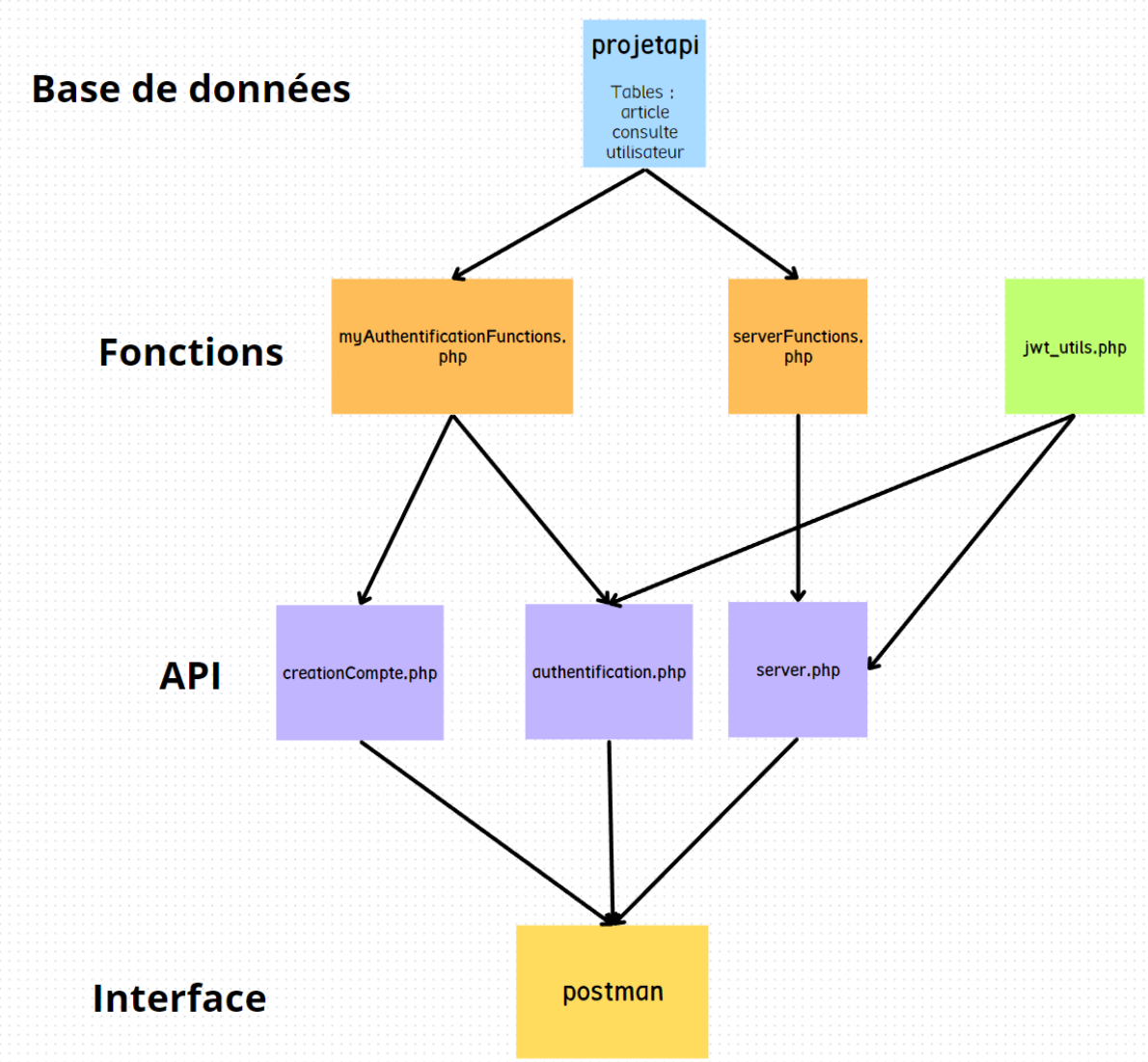
Sommaire

Lien Git :

Lien github : https://github.com/AlexandreCalmet/ProjetPHP.git

Postman : les URL pour manipuler les différentes requêtes sont dans la partie Requêtes postman de la collection

Architecture logicielle & dépendances :

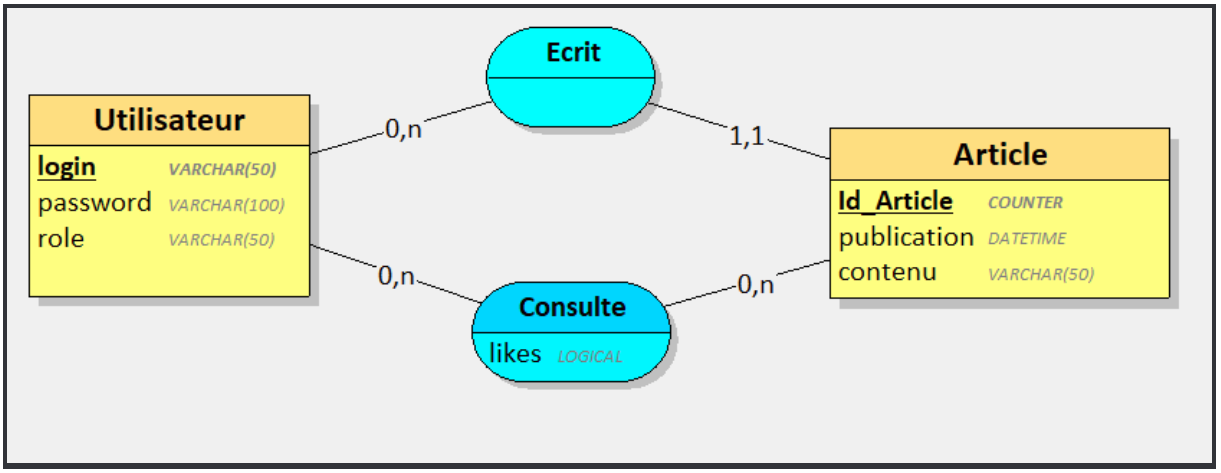


Installation :

1. Lancer le script sql “projetapi.sql” dans un query MySQL.
2. Importer les requêtes postman “PostManRequetes.json” dans postman.
3. Déplacer le dossier “/ProjetAPI” dans votre répertoire serveur local (wamp ou xampp).

1. Run “projetapi.sql” in a MySQL query.
2. Import “PostManRequetes.json” into postman.
3. Move folder “/ProjetAPI” into your local server folder. (wamp/xampp).

Base de données “projetapi” :



Fonctions de “myAuthentificationFunctions.php” :

$rouge : variables
vert : valeurs prédéfinies
bleu : tables de la bdd “projetapi”

connection() :	Retourne la connexion à la bdd “**projetapi**” en tant qu’utilisateur “**root**”.
getNameByName($utilisateur):	Retourne le login de l’utilisateur **$utilisateur**.
getPasswordByName($utilisateur):	Retourne le mot de passe de l’utilisateur **$utilisateur**.
isValidUser($utilisateur, $password):	Si le **$password** et **$utilisateur** existe bien dans la base de données et correspondent : retourne vrai. Sinon : retourne faux.
getRole($utilisateur):	Retourne le rôle de l’utilisateur **$utilisateur** qui lui est associé dans la bdd.
insert($utilisateur, $password):	Insère l’utilisateur **$utilisateur** avec son mot de passe **$password** déjà chiffré dans la table **utilisateur**.

Fonctions de “serverFunctions.php”:

$rouge : variables
vert : valeurs prédéfinies
bleu : tables de la bdd “projetapi”

connection() :	Retourne la connexion à la bdd “**projetapi**” en tant qu’utilisateur “**root**”.
getById($utilisateur=null):	Si **$utilisateur** est null ou invalide : retourne l’ensemble des publications sans leurs likes/dislikes. Si **$utilisateur** est valide : retourne l’ensemble des publications sans leurs likes/dislikes de l’utilisateur **$utilisateur**.
getWithLikesById($utilisateur=null):	Si **$utilisateur** est null ou invalide : retourne l’ensemble des publications avec leurs likes/dislikes. Si **$utilisateur** est valide : retourne l’ensemble des publications avec leurs likes/dislikes de l’utilisateur **$utilisateur.**
getLikesById($id_article):	Retourne le nombre de fois que l’article associé à **$id_article** a été aimé (Quand vote=**1**).
getDislikesById($id_article):	Retourne le nombre de fois que l’article associé à **$id_article** n’a pas été aimé (Quand vote=**0**).
insert($contenu, $utilisateur):	Insère l’article avec son auteur **$utilisateur**et son contenu **$contenu** avec un id_article auto incrémenté, la date d’aujourd’hui dans la table **article**.
deleteId($id_article):	Supprime l’article associé à **$id_article** dans la table **article**.
getAuthor($id_article):	Retourne le nom de l’auteur (son login) associé à l’article **$id_article.**
update($contenu, $id_article):	Met à jours le **$contenu** de l’article associé à **$id_article** présent dans la table **article**.

vote($utilisateur, $id_article, $vote):	Met à jours le **$vote** de l'article associé à **$id_article** de la table **consulte**.
isNewVote($utilisateur, $id_article):	Si la liaison entre l'article **$id_article** et **$utilisateur** existe : retourne vrai Sinon : retourne faux.
insertNewVote($utilisateur, $id_article, $vote):	Insère la liaison entre l'utilisateur **$utilisateur** et un article **$id_article** dans la table consulte et insère aussi la valeur de **$vote** dans **vote**.

Fonctions de “jwt_utils.php”:

$rouge : variables
/!\ : Fonctions implémentées et utilisées pour le client qui n’est pas fonctionnel

/! **split_jwt($jwt) :**	Utilisée pour l’interface client qui n’est pas fonctionnelle. Retourne le token en trois parties dans un tableau associatif : ‘header’, ‘payload’, ‘signature’.
/! **stringPayloadToArray($payload):**	Utilisée pour l’interface client qui n’est pas fonctionnelle. Retourne le payload en trois parties dans un tableau associatif : ‘username’, ‘exp’, ‘role’.

User stories :

Utilisateur non authentifié :

- En tant qu'utilisateur non authentifié, je souhaite utiliser la requête **getPublicationAsNonAuthentified** pour voir la liste des publications avec les informations dont les noms d'auteur, leurs dates de publication et leurs contenus.
- En tant qu'utilisateur non authentifié, je souhaite utiliser la requête **getSomeonesPublicationsAsNonAuthentified** pour voir la liste des publications d'un auteur, les dates de publication et leurs contenus.
- En tant qu'utilisateur non authentifié, je souhaite utiliser la requête **createLogin** ($login, $password) pour pouvoir créer un compte.
- En tant qu'utilisateur non authentifié, je souhaite utiliser la requête **authentification** pour m'authentifier en tant que Modérateur.
- En tant qu'utilisateur non authentifié, je souhaite utiliser la requête **authentification** pour m'authentifier en tant que Publisher.

Utilisateur authentifié avec le rôle publisher :

- En tant que Publisher, je souhaite utiliser la requête **updatePublicationContenu** pour modifier une publication que j'ai postée qui ne me convient plus.
- En tant que Publisher, je souhaite utiliser la requête **deletePublication** pour supprimer une publication que j'ai posté qui ne me convient plus.
- En tant que Publisher, je souhaite utiliser la requête **getPublications** pour lire toutes les publications avec le nom de l'auteur, la date, le contenu, les likes et les dislikes.
- En tant que Publisher, je souhaite utiliser la requête **getSomeonesPublications** pour lire les publications créées par un utilisateur.
- En tant que Publisher, je souhaite utiliser la requête **getPublicationsLikes** pour connaître le nombre de fois que mon article a été aimé.
- En tant que Publisher, je souhaite utiliser la requête **upVote** pour augmenter la quantité de likes par 1 de la publication publiée par un autre Publisher (ou moi-même) que j'ai aimé.
- En tant que Publisher, je souhaite utiliser la requête **downVote** pour augmenter la quantité de dislikes par 1 de la publication publiée par un autre Publisher (ou moi-même) que je n'ai pas aimé.

Utilisateur authentifié avec le rôle modérateur :

- En tant que Modérateur, je souhaite utiliser la requête **deletePublication** pour supprimer une publication d'un Publisher que je ne veux pas présente dans la base de données.
- En tant que Modérateur, je souhaite utiliser la requête **getPublications** pour lire les publications créées avec toutes les informations, l'auteur, la date de publication, le contenu, le nombre de likes/dislikes et le personne qui ont liké ou disliké l'article

Requêtes postman de la collection “ProjetAPI” :

$rouge : variables
vert : valeurs prédéfinies

GET (x5):

- **getPublicationsAsNonAuthentified**
  Retourne toutes les publications.
  Codes : 201 si réussie, 405 si méthode non reconnue.
  URL : http://localhost/ProjetAPI/server.php

- **getSomeonesPublicationsAsNonAuthentified**(param=(login=**$login**))
  Retourne toutes les publications postées par l’auteur associé à son **$login**.
  Codes : 201 si réussie, 405 si méthode non reconnue.
  URL : http://localhost/ProjetAPI/server.php?login=**$login**

Token valide nécessaire :

- **getPublications**
  Retourne toutes les publications avec leurs likes, dislikes et total (likes - dislikes)
  Codes : 201 si réussie, 405 si méthode non reconnue.
  URL : http://localhost/ProjetAPI/server.php

- **getSomeonesPublications**(param=(login=**$login**))
  Retourne toutes les publications avec leurs likes, dislikes et total (likes-dislikes) postées par l’auteur associé à son **$login**.
  Codes : 201 si réussie, 405 si méthode non reconnue.
  URL : http://localhost/ProjetAPI/server.php?login=**$login**

- **getPublicationsLikes**(param=(id_article=**$id_article**))
  Retourne le nombre de likes d’une publication associée à l’identifiant **$id_article**.
  Codes : 201 si réussie, 405 si méthode non reconnue.
  URL : http://localhost/ProjetAPI/server.php?id_article=**$id_article**

POST (x3):

1. **createLogin**(body={“username”:**$login**, “password:”**$password**})
   Créer un utilisateur dont le **$login** et **$password** sont ceux que l’utilisateur a transmis et lui attribue le rôle “Publisher”.
   Codes : 204 si réussie,403 si le login utilisateur existe déjà, 405 si méthode non reconnue.
   URL : http://localhost/ProjetAPI/creationCompte.php

2. **authentification**(body={“username”:**$login**, “password:”**$password**})
   Génère et retourne un token associé au **$login** comprenant son rôle, son **$login,** d’une validité de 1 heure.
   Codes : 201 si réussie, 404 si le login utilisateur n’existe pas, 405 si méthode non reconnue.

   URL : http://localhost/ProjetAPI/authentification.php
   IMPORTANT : Pour obtenir le rôle “**Moderator**” :
   authentification(body={“username”:**AlexM**, “password:”**6559**})

Token valide nécessaire :

- **createPublication**(body={“contenu”:**$contenu**})
  Créer une publication dont l’identifiant **$id_article** est un nombre auto-incrémenté, **$publication** est la date/heure au moment de sa publication, **$contenu** est celui enregistré par l’utilisateur, **$login** est l’auteur de cette nouvelle publication.
  Codes : 201 si réussie, 405 si méthode non reconnue, 498 si le jeton est invalide.
  URL : http://localhost/ProjetAPI/server.php

DELETE (x1):

Token valide nécessaire :

- **deletePublication**(param=(id_article=**$id_article**))
  Supprime la publication associée à l’identifiant **$id_article**.
  Codes : 204 si réussie, 401 si non autorisé, 405 si méthode non reconnue, 498 si jeton invalide.
  URL : http://localhost/ProjetAPI/server.php?id_article=**$id_article**

PUT (x3):

Token valide nécessaire :

- **updatePublicationContenu**(body={“contenu”:**$contenu**, “id_article”:**$id_article**})
  Change le contenu de la publication associée à l’identifiant **$id_article** par le nouveau contenu **$contenu**.
  Codes : 204 si réussie, 401 si non autorisé, 405 si méthode non reconnue, 498 si jeton invalide.
  URL : http://localhost/ProjetAPI/server.php

- **upVote**(body={“id_article”:**$id_article**, “vote”:**1**})
  Ajoute un like à la publication associée à l’identifiant **$id_article**.
  Codes : 204 si réussie, 401 si non autorisé, 405 si méthode non reconnue, 498 si jeton invalide.
  URL : http://localhost/ProjetAPI/server.php

- **downVote**(body={“id_article”:**$id_article**, “vote”:**0**})
  Ajoute un dislike à la publication associée à l’identifiant **$id_article**.
  Codes : 204 si réussie, 401 si non autorisé, 405 si méthode non reconnue, 498 si

jeton invalide.
URL : http://localhost/ProjetAPI/server.php