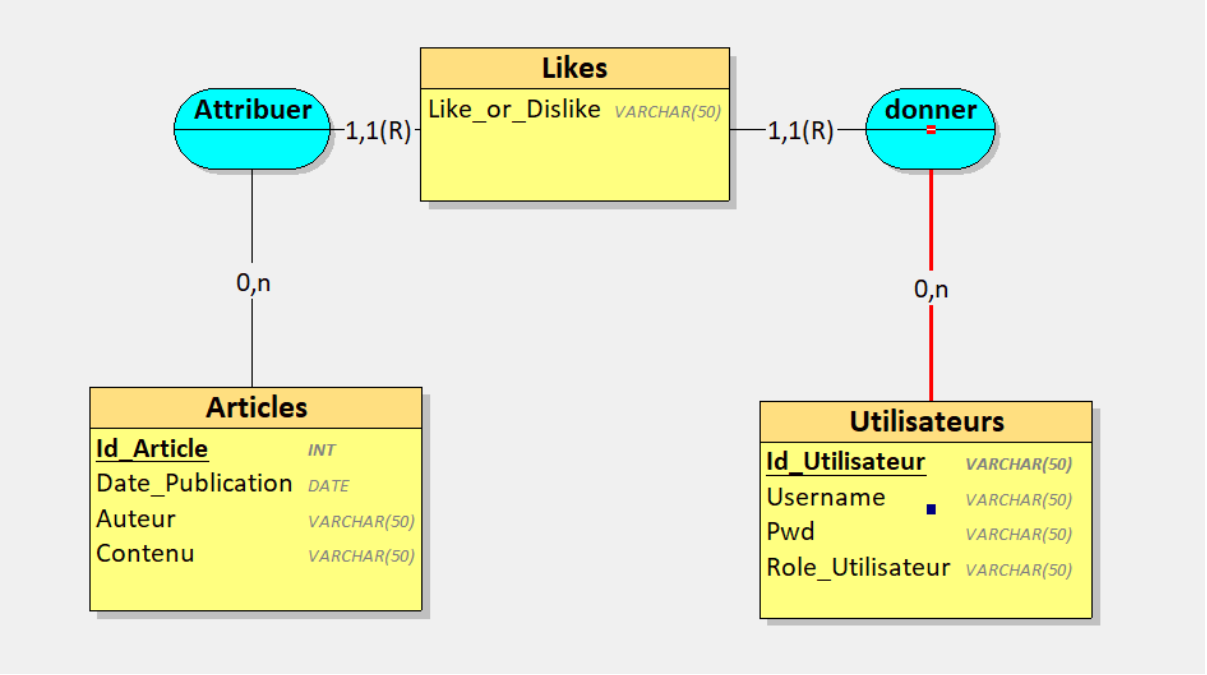
PROJET R401
COMPTE RENDU

1. URL d'accès au dépôt GIT

https://github.com/LamaTable/R401-Projet-BdtNins.git

2. MCD



3. Spécifications de la ou les méthodes API REST proposées :

a. Les méthodes proposées
b. Pour chaque méthode, les données éventuellement attendues en entrée ainsi que celles retournées
c. Pour chaque méthode, les traitements associés (en langage naturel ou pseudo-code par exemple)
d. Pour chaque méthode, les types d'erreur pris en compte

Nous avons proposé 18 méthodes dans le fichier ServeurFonction :

Serveur authentification :

1. **connectToDatabase() :**
   - Cette méthode permet de se connecter à la base de données.
   - Donnée(s) attendue(s) en entrée : aucune
   - Donnée(s) attendue(s) en sortie : un objet pdo (le nom d'utilisateur, le mot de passe et le lien pour se connecter à phpmyadmin, représentant la connexion à la base de données)
   - Traitement(s) associé(s) :
     - Essaye
       - créer la connexion avec la base
       - retourne la connexion $pdo
     - Attrape erreur
       - renvoie erreur
   - Type(s) d'erreur pris en compte : Si la connexion ne fonctionne pas

2. **getIdUtilisateur():**
   - Cette méthode permet de récupérer l'identifiant de l'utilisateur en interrogeant la base de données pour trouver l'utilisateur correspondant au nom d'utilisateur et au mot de passe fournis. La fonction utilise la méthode 'prepare()' de l'objet pdo pour préparer la requête SQL paramétrée. Elle lie ensuite les paramètres de la requête aux valeurs fournies en utilisant la méthode 'bindParam()', puis exécute la requête avec la méthode 'execute(). Elle récupère les résultats de la requête avec la méthode 'fetchAll()' et les renvoie. En cas d'erreur, la fonction intercepte l'exception 'PDOException' et affiche un message d'erreur.
   - Donnée(s) attendue(s) en entrée : pdo, le nom d'utilisateur et le mot de passe
   - Donnée(s) attendue(s) en sortie : l'identifiant de l'utilisateur
   - Traitement(s) associé(s) :
     - Essaye
       - Si Username et password n'est pas vide alors
         - sélectionner l'id utilisateur ou Username est égale au username mis en paramètre et ou le password est égale au password mis en entrée
       - retourne la connexion $pdo
     - Attrape erreur
       - renvoie erreur
   - Type(s) d'erreur pris en compte : s'il y a une erreur quand on récupère les données, s'il y a une erreur de connexion à la base de données (par

exemple si les informations d'identification fournies sont incorrectes) et les erreurs liées à l'exécution de la requête (par exemple si la table "utilisateurs" n'existe pas.

3. **getUtilisateurRole() :**
   - Cette méthode permet de récupérer le rôle de l'utilisateur en interrogeant la base de données dans le but de trouver le rôle de l'utilisateur correspondant au nom d'utilisateur et au mot de passe fournis.
   - Donnée(s) attendue(s) en entrée : pdo, le nom d'utilisateur et le mot de passe
   - Donnée(s) attendue(s) en sortie : le rôle de l'utilisateur
   - Type(s) d'erreur pris en compte : les erreurs de connexion à la base de données, les erreurs liées à l'exécution de la requête SQL donc s'il y a une erreur quand on récupère les données.

4. **checkUser() :**
   - Cette méthode permet de vérifier s'il y a bien un nom d'utilisateur et s'il est correct dans la base de données.
   - Donnée(s) attendue(s) en entrée : pdo, l'utilisateur et le mot de passe
   - Donnée(s) attendue(s) en sortie : si le nom d'utilisateur et le mot de passe sont corrects, elle renvoie 'true'. Sinon elle lève une exception avec un message d'erreur correspondant.
   - Type(s) d'erreur pris en compte : les erreurs liées à la vérification des informations d'identification (par exemple, si le nom d'utilisateur ou le mot de passe fourni est incorrect).

All :

5. **getTokenUtilisateurRole():**
   - Cette méthode permet de vérifier que le jeton d'authentification est valide en utilisant la fonction 'is_jwt_valid()'. Si le jeton est valide, elle extrait le rôle de l'utilisateur et le retourne, sinon elle ne retourne rien.
   - Donnée(s) attendue(s) en entrée : null mais la fonction utilise la fonction 'get_bearer_token()' pour récupérer le jeton d'authentification de l'utilisateur.
   - Donnée(s) attendue(s) en sortie : le rôle de l'utilisateur qu'on prend dans le jeton d'authentification.
   - Type(s) d'erreur pris en compte : Si le token n'est pas valide, si la récupération du token échoue.

6. **getTokenIdUtilisateur():**

- La fonction vérifie si le jeton d'authentification est fourni et s'il est correct. Si le jeton est correct alors la fonction récupère l'id de l'utilisateur grâce au jeton et le renvoie.
- Donnée(s) attendue(s) en entrée : aucune mais la fonction appelle la fonction 'get_bearer_token()' pour récupérer le jeton d'authentification.
- Donnée(s) attendue(s) en sortie : l'identifiant de l'utilisateur associé au jeton d'authentification fourni.
- Type(s) d'erreur pris en compte : Si le token n'est pas valide, si la récupération du token a échoué (le décodage).

Get :

Pour tous les Get, dans la gestion d'erreurs, il envoie un message s'il y a une erreur avec la requête SQL.

7. **getDataModerateur() :**
   - Cette méthode permet en fonction des rôles de récupérer un certain nombre de données
   - Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article (optionnel)
   - Donnée(s) attendue(s) en sortie : toutes les informations de l'article dans un tableau, le nombre de like et de dislike de cet article et pour finir la liste des identifiants des utilisateurs ayant "liké" et "disliké" l'article
   - Traitement(s) associé(s) : on récupère les informations concernant l'article, le nombre like/dislike associé et la liste des personnes qui ont like/dislike
   - Type(s) d'erreur pris en compte : si l'id_article en paramètre est vide.

8. **getDataPublisher() :**
   - Cette méthode permet de récupérer les informations de l'article grâce à la fonction "getAllDataArticle", le nombre de likes à partir de la fonction "getNumberLike" et le nombre de dislikes à partir de la fonction "getNumberDislike".
   - Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
   - Donnée(s) attendue(s) en sortie : les informations de l'article rentré en paramètre, le nombre de likes et le nombre de dislikes.
   - Type(s) d'erreur pris en compte : si l'id_article en paramètre est vide.

9. **getDataAnonyme() :**
   - La fonction récupère les données grâce à une requête SQL, si l'identifiant n'est pas null.
   - Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article

- Donnée(s) attendue(s) en sortie : les informations de l'article telles que l'auteur, la date de publication et le contenu.
- Type(s) d'erreur pris en compte : si l'article n'existe pas.

10. getNumberLike() :

- Cette méthode permet d'obtenir le nombre de like d'un article (la fonction prépare une requête SQL pour compter le nombre de lignes dans la table "likes" où l'identifiant de l'article correspond à celui fourni en entrée, et où la colonne "Like_or_Dislike" vaut 1 (ce qui signifie que l'action est un "like"). La requête est ensuite exécutée, et le nombre de lignes retourné est stocké dans une variable "$nbrlike". C'est cette variable qui est retournée en sortie).
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
- Donnée(s) attendue(s) en sortie : le nombre de "likes" de l'article en question.

11. getNumberDislike() :

- Cette méthode permet d'obtenir le nombre de dislike d'un article (la fonction prépare une requête SQL pour compter le nombre de lignes dans la table "likes" où l'identifiant de l'article correspond à celui fourni en entrée, et où la colonne "Like_or_Dislike" vaut 0. Elle utilise une préparation de requête pour éviter les injections SQL).
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
- Donnée(s) attendue(s) en sortie : le nombre de "dislikes" de l'article en question.

12. getListUserLike() :

- La fonction, grâce à une requête SQL, récupère la liste des utilisateurs qui ont liké l'article et extrait seulement les identifiants de sutilisateurs.
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article.
- Donnée(s) attendue(s) en sortie : la liste des utilisateurs ayant "liké" l'article.

13. getListUserDislike() :

- La fonction, grâce à une requête SQL, récupère la liste des utilisateurs qui ont disliké l'article et extrait seulement les identifiants de sutilisateurs dans un tableau.
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
- Donnée(s) attendue(s) en sortie : la liste des utilisateurs ayant "disliké" l'article.

14. **getAllDataArticle() :**
    - La fonction récupère toutes les données concernant l'article et les stocke dans un tableau.
    - Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article.
    - Donnée(s) attendue(s) en sortie : toutes les données de l'article en question.

Post :

15. **addData() :**
    - Cette méthode permet d'ajouter une donnée dans la base de données (grâce à une requête d'insertion, ajouter les informations entrées en paramètres dans la base de données).
    - Donnée(s) attendue(s) en entrée : pdo, le nom de l'auteur et le contenu
    - Donnée(s) attendue(s) en sortie : un booléen qui indique si l'ajout a été effectué avec succès ou non.
    - Type(s) d'erreur pris en compte : s'il y a une erreur lors de l'ajout des données.

16. **updateData() :**
    - Cette méthode permet de modifier une donnée dans la base de données (grâce à une requête SQL on modifie le contenu de la abse de données pour mettre à jour l'article).
    - Donnée(s) attendue(s) en entrée : pdo, l'id article et le contenu
    - Donnée(s) attendue(s) en sortie : un booléen qui indique si la mise à jour a fonctionné ou non.
    - Type(s) d'erreur pris en compte : s'il y a une erreur lors de la mise à jour des données.

17. **deleteData() :**
    - Cette méthode permet de supprimer une donnée dans la BD (grâce à une requête SQL on supprime l'article ayant l'identifiant qui correspond avec celui en paramètre).
    - Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
    - Donnée(s) attendue(s) en sortie : un booléen (true si l'article est bien supprimé).
    - Type(s) d'erreur pris en compte : s'il y a une erreur lors de la suppression des données des données

18. **getAuteurId() :**
    - Cette méthode permet de récupérer l'identifiant de l'utilisateur dans la base de données.

- Donnée(s) attendue(s) en entrée : pdo et le nom de l'auteur.
- Donnée(s) attendue(s) en sortie : l'identifiant de l'utilisateur.

19. getAuteurName() :

- Cette méthode permet de récupérer le nom de l'auteur à partir de l'identifiant.
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
- Donnée(s) attendue(s) en sortie : le nom de l'auteur.
- Type(s) d'erreur pris en compte : erreur SQL

20. addLike() :

- Cette méthode permet d'ajouter un like sur un article (l'ajout d'un like dans la table "likeordislike" de la bd).
- Donnée(s) attendue(s) en entrée : pdo, l'identifiant de l'article, l'identifiant de l'utilisateur et une valeur "LikeorDislike" qui signifie que si c'est 0 c'est un dislike ou si c'est 1 c'est considéré comme un like.
- Donnée(s) attendue(s) en sortie : un booléen qui indique si l'utilisateur a ajouté un like ou un dislike (true) ou s'il y a une erreur (false).
- Type(s) d'erreur pris en compte : s'il y a une erreur lors de l'ajout d'un like/dislike

Patch :

21. updateLike()

- Cette méthode permet de modifier un like sur un article (La fonction met à jour le champ Like_or_Dislike de la table "likes" avec la valeur $Like_or_Dislike pour l'article et l'utilisateur correspondants).
- Donnée(s) attendue(s) en entrée : pdo, l'identifiant de l'article et de l'utilisateur et LikeorDislike
- Donnée(s) attendue(s) en sortie : La fonction retourne un booléen qui vaut true si la mise à jour du like a été effectuée avec succès, et false sinon.
- Type(s) d'erreur pris en compte : s'il y a une Erreur lors de la mise à jour d'un like.

Delete :

22. deleteLikeUser() :

- Cette méthode permet de supprimer un like sur un article (suppression d'un enregistrement dans la table 'likes' correspondant à l'identifiant de l'article ($Id_Article) et de l'utilisateur ($Id_Utilisateur) donnés).
- Donnée(s) attendue(s) en entrée : pdo, l'identifiant de l'article et de l'utilisateur.

- Donnée(s) attendue(s) en sortie : booléen true en cas de succès de la suppression, sinon l'erreur est levée.
- Type(s) d'erreur pris en compte : s'il y a une Erreur lors de la suppression du like.

23. deleteAllLike() :

- Cette méthode permet de supprimer un like sur un article.
- Donnée(s) attendue(s) en entrée : pdo et l'identifiant de l'article
- Donnée(s) attendue(s) en sortie : un booléen qui vaut true si la suppression est faite, false sinon.
- Type(s) d'erreur pris en compte : s'il y a une Erreur lors de la suppression du like.

Nous avons proposé 4 méthodes et une fonction dans le fichier Serveur :

24. Méthode :

Cette fonction est une API qui gère les requêtes GET, POST, PUT et DELETE pour un article.

Entrées :

- $http_method: chaîne de caractères indiquant la méthode de requête HTTP utilisée (GET, POST, PUT, DELETE).
- $RoleUtilisateur: rôle de l'utilisateur (moderateur, publisher, ou vide pour anonyme).
- $_GET['Id_article']: l'identifiant de l'article (seulement pour la méthode GET).
- Données JSON dans le corps de la requête (seulement pour les méthodes POST, PUT et DELETE).

Sorties :

- Statut HTTP: entier représentant le statut de la réponse HTTP (par exemple, 200 pour OK ou 404 pour Non trouvé).
- Message: chaîne de caractères contenant un message décrivant la réponse.
- Données JSON dans le corps de la réponse (seulement pour les méthodes GET, POST et PUT).

Traitement :

- La fonction appelle différentes fonctions relativement à la méthode de requête HTTP et du rôle de l'utilisateur. Ces fonctions effectuent des requêtes SQL sur une base de données pour récupérer, ajouter, modifier ou supprimer des données.
- La fonction renvoie une réponse HTTP avec un code de statut approprié, un message de réponse et éventuellement des données JSON.

Erreurs :

- Erreur 400: le corps de la requête JSON est vide ou mal formé (seulement pour les méthodes POST, PUT et DELETE).
- Erreur 401: l'utilisateur n'est pas authentifié ou n'a pas les autorisations appropriées pour effectuer la requête.
- Erreur 404: l'article demandé n'existe pas dans la base de données.
- Erreur 500: erreur de serveur interne, par exemple une erreur de base de données ou une erreur de syntaxe SQL.

25. **deliverResponse() :** et pour finir, nous avons cette fonction déjà faite, qui permet de mettre en forme les messages.

4. Au moins une user story pour chacun des types d'utilisateur exploitant votre solution (utilisateur non authentifié ; utilisateur authentifié avec le rôle moderator ; utilisateur authentifié avec le rôle publisher)

- **Utilisateur non authentifié :**

L'auteur non authentifié peut consulter tous les articles : il a accès au contenu, auteur la date publication de l'article.

Exemple : l'utilisateur non authentifié fait un get de ID 17 et obtiendra seulement :


```
{
    "status": 200,
    "status_message": "Voici les données de la publication : ",
    "data": [
        {
            "Auteur": "Napoléon1er",
            "Date_Publication": "2023-03-30",
            "Contenu": "La mort n'est rien, mais vivre vaincu et sans gloire c'est mourrir tous
                les jours"
        }
    ]
}
```


Cependant, s'il essaie de modifier un article, le résultat sera le suivant :


```
{
    "status": 401,
    "status_message": "Erreur : Utilisateur non authentifié.",
    "data": null
}
```


- **Utilisateur authentifié avec le rôle moderator :**

Un utilisateur authentifié avec le rôle moderator peut :

○ Consulter n'importe quel article. Un utilisateur moderator doit accéder à l'ensemble des informations décrivant un article : auteur, date de publication, contenu, liste des

utilisateurs ayant liké l'article, nombre total de like, liste des utilisateurs ayant disliké l'article, nombre total de dislike.
○ Supprimer n'importe quel article.

Exemple : le moderator peut accéder à toutes les informations concernant l'article 16, le résultat est le suivant :


```
"status": 200,
"status_message": "Voici les données de la publication : ",
"data": {
    "dataArticle": [
        {
            "Id_Article": "16",
            "Date_Publication": "2023-03-29",
            "Auteur": "Homer",
            "Contenu": "J'aime les réçiss grec"
        }
    ],
    "nbrLike": "1",
    "nbrDislike": "1",
    "Liste_des_Utilisateur_Like": [
        [
            {
                "Username": "Napoleon1er"
            }
        ]
    ],
    "Liste_Des_Utilisateur_Dislike": [
        [
            {
                "Username": "Moliere"
            }
```


Cependant, s'il essaie de liker l'article 16, le résultat sera la suivant :


```
"status": 404,
"status_message": "Erreur",
"data": "Seul les personnes ayant le rôle publisher peuvent publier, les modérateur et les anonymes ne peuvent pas publier d'articles"
```


- **Utilisateur authentifié avec le rôle publisher :**

Un utilisateur authentifié avec le rôle publisher peut :
○ Poster un nouvel article.
○ Consulter ses propres articles.
○ Consulter les articles publiés par les autres utilisateurs. Un utilisateur publisher doit accéder aux informations suivantes relatives à un article : auteur, date de publication, contenu, nombre total de like, nombre total de dislike.
○ Modifier les articles dont il est l'auteur.
○ Supprimer les articles dont il est l'auteur.
○ Liker/disliker les articles publiés par les autres utilisateurs.

Exemple : si l'utilisateur veut modifier un article dont il est l'auteur, le résultat sera le suivant :


```
{
    "status": 201,
    "status_message": "Votre publication à bien été modifié",
    "data": true
}
```


Cependant, s'il veut modifier un article dont il n'est pas l'auteur, le résultat sera le suivant :


```
{
    "status": 404,
    "status_message": "Erreur",
    "data": "Seul l'auteur de l'article peut modifier l'article"
}
```