Projet 3I017 - Fiche service

15 avril 2019

Table des matières

1. **[Services utilisateur](#bookmark) [1](#bookmark1)**
2. **[Service ami](#bookmark2) [6](#bookmark3)**
3. **[Services message](#bookmark4) [8](#bookmark5)**
4. **[Services commentaire](#bookmark6) [10](#bookmark7)**
5. **Services utilisateur**

Table 1: Fiche *login*

|  |  |
| --- | --- |
| Nom du *Web service* | Login |
| URL du *Web service* | */Login* |
| Description du *Web service* | Connexion de l’utilisateur vers le site |
| Paramètre(s) du *Web service* | Le login de l’utilisateur et son mot de passe |
| Format de sortie | *{"status" : "ok", "key" : clé\_connexion}* ou *{"sta-*  *tus" : "ko", "message" : message\_erreur, "code" : data\_erreur}* |
| Exemple de sortie | *{"status" : "ok", "key" : "5Gb4PH6K..."}* |
| Erreurs possibles | Paramètre(s) manquant, login non présent dans la base de données, mauvais mot de passe, erreur *JSON*, erreur  base de données *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet Login.java* |
| Informations additionnelles |  |

Table 2: Fiche *logout*

|  |  |
| --- | --- |
| Nom du *Web service* | Logout |
| URL du *Web service* | *Logout* |
| Description du *Web service* | Déconnexion de l’utilisateur |
| Paramètre(s) du *Web service* | La clé de connexion de l’utilisateur |
| Format de sortie | *{"status" : "ok"}* ou *{"status" : "ko", "message" :*  *message\_erreur, "Errorcode" : data\_erreur}* |
| Exemple de sortie | {"User already connected.":1007,"Output":"Service Accepted","key":"W3hmt4ki4oSmy5xYSfgrJCurvAgBG1xG"} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, er-  reur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet Logout.java* |
| Informations additionnelles |  |

Table 3: Fiche *createUser*

|  |  |
| --- | --- |
| Nom du *Web service* | CreateUser |
| URL du *Web service* | *CreateUser* |
| Description du *Web service* | Ajoute un utilisateur à la base de données *MySQL* |
| Paramètre(s) du *Web service* | Nom, Prénom de l’utilisateur, un login et un mot de passe  Son adresse e-mail |
| Format de sortie | *{"status" : "ok"}* ou *{"status" : "ko", "message" :*  *message\_erreur, "Errorcode" : data\_erreur}* |
| Exemple de sortie | {"user has been added successfully":1,"Output":"Service Accepted","hTSV4H7ONrS6XOTIeAcBa0Lstzfk1HM5":1} |
| Erreurs possibles | Paramètre(s) manquant, login déjà présent dans la base de données, erreur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet CreatUser.java* |
| Informations additionnelles | *(‘e’ oublié dans creat x) )* |

Table 5: Fiche *PasswordRecovery*

|  |  |
| --- | --- |
| Nom du *Web service* | PasswordRecovery |
| URL du *Web service* | */PasswordRecovery* |
| Description du *Web service* | Envoie un mail à l’utilisateur avec un nouveau mot de passe. |
| Paramètre(s) du *Web service* | L’adresse mail |
| Format de sortie | *{"status" : "ok"}* ou *{"status" : "ko", "message" :*  *message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"Message":"MAIL does not exist.","ErrorCode":1002} |
| Erreurs possibles | Paramètre(s) manquant, les deux mails sont différents,  erreur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet PasswordRecovery.java* |
| Informations additionnelles | Le Service check seulement si le mail et dans la BD et non si réellement exist. |

1. **Service ami**

Table 10: Fiche *addFriend*

|  |  |
| --- | --- |
| Nom du *Web service* | AddFriend |
| URL du *Web service* | */AddFriend* |
| Description du *Web service* | L’utilisateur s’abonne à un autre utilisateur et va pouvoir par conséquence voir ses twists. |
| Paramètre(s) du *Web service* | L’id de l’utilisateur, l’id de l’ami et la clé de connexion |
| Format de sortie | *{"status" : "ok"}* ou *{"status" : "ko", "message" :*  *message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"user1: just followed user2":1,"Output":"Service Accepted"} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, l’utilisateur ou l’abonné n’existent pas dans la base de données *MySQL*, erreur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Pas fonctionnel |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet AddFriend.java* |
| Informations additionnelles | Le service marche coté serveur mais ne fonctionne pas coté client. |

Table 11: Fiche *removeFriend*

|  |  |
| --- | --- |
| Nom du *Web service* | removeFriend |
| URL du *Web service* | */RemoveFriend* |
| Description du *Web service* | *L'utilisateur se désabonne d’un utilisateur existant dans liste abonnés (dans la base MySql)* |
| Paramètre(s) du *Web service* | L’id de l’utilisateur, l’id de l’ami et la clé de connexion |
| Format de sortie | *{"status" : "ok"}* ou *{"status" : "ko", "message" :*  *message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"user1 has unfollowed user2":1,"Output":"Service Accepted"} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, l’utilisateur ou l'abonné n’existent pas dans la base de données *MySQL*, erreur *JSON*, erreur base de données  *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet RemoveFriend.java* |
| Informations additionnelles |  |

Table 12: Fiche *listFriend*

|  |  |
| --- | --- |
| Nom du *Web service* | ListFollowers |
| URL du *Web service* | */ListFollowers* |
| Description du *Web service* | Liste de tous les abonnés |
| Paramètre(s) du *Web service* | L’id de l’utilisateur, et la clé de connexion |
| Format de sortie | * Followers: Array(3)   0: {id: 8, user: "user0"}  1: {id: 10, user: "user1"}  2: {id: 11, user: "user2"} |
| Exemple de sortie | {"Followers":[{"id":8,"user":"chikow"},{"id":10,"user":"Delpax"},{"id":11,"user":"Delpaxll"}]} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, er-  reur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet ListFriend.java* |
| Informations additionnelles |  |

Table 13: Fiche *searchUser*

|  |  |
| --- | --- |
| Nom du *Web service* | searchFriend |
| URL du *Web service* | */searchFriend* |
| Description du *Web service* | Effectue une recherche d’abonné |
| Paramètre(s) du *Web service* | la clé de connexion, login d’abonné |
| Format de sortie | *{"status" : "ok", "list" :[ {"user\_id" : user\_login},*  *, ...]}* ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | *{{"Message":"User not connected","ErrorCode":10000}* |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, er-  reur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | Pas implémenté |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet SearchFriend.java* |
| Informations additionnelles | Peut être améliorer avec Map Reduce |

1. **Services Twist**

Table 14: Fiche *AddTwist*

|  |  |
| --- | --- |
| Nom du *Web service* | AddTwist |
| URL du *Web service* | */AddTwist* |
| Description du *Web service* | Ajoute un twistà la base de données *MongoDB* |
| Paramètre(s) du *Web service* | La clé de connexion et le twist à ajouter |
| Format de sortie | *{"status" : "ok", "added\_twist" : twist}*  ou *{"status" : "ko", "message" : twist\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"Message":"User not connected","ErrorCode":10000} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, er-reur *JSON*, erreur base de données *MongoDB* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet AddTwist.java* |
| Informations additionnelles | Besoin de d’actualiser la page (côté client) pour un re-render de compensent afin d’afficher le twist |

Table 15: Fiche *RemoveTwist*

|  |  |
| --- | --- |
| Nom du *Web service* | RemoveTwist |
| URL du *Web service* | */RemoveTwist* |
| Description du *Web service* | Supprime un twist de la base de donnée *MongoDB* |
| Paramètre(s) du *Web service* | La clé de connexion et le message à supprimer |
| Format de sortie | *{"status" : "ok", "removed\_message" : le\_message}*  ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"Output":"Service Accepted","Comment has been added":1} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, erreur *JSON*, erreur base de données *MongoDB* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet RemoveTwist.java* |
| Informations additionnelles | Ça beug de temps en temps coté client |

Table 16: Fiche *ListTwist*

|  |  |
| --- | --- |
| Nom du *Web service* | ListTwist |
| URL du *Web service* | */ListTwist* |
| Description du *Web service* | Liste tous les twists d’un utilisateur |
| Paramètre(s) du *Web service* | La clé de connexion et le login de l’utilisateur, nombre de twists(static) , trie (boolean) |
| Format de sortie | *{"status" : "ok", "list\_message" :[ {"twist1" :*  *“hello”},...]}* ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | {"Output":"Service Accepted","My\_twists":[{"date":"Tue Apr 16 11:41:55 CEST 2019","comments":[],"id":"5cb5a3636d7bb05485887b67","login":"neel","content":"hellooooo"}]} |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, erreur JSON, erreur base de données MongoDB |
| Avancement du *Web service* | Fini (composent appelé sur la page profile) |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet ListTwist.java* |
| Informations additionnelles | Une fonctionnalité de choix du nombre de twists à afficher était ajoutée côté serveur mais pas implémentée côté client |

Table 17: Fiche *ListTwist utilisateurs suivis*

|  |  |
| --- | --- |
| Nom du *Web service* | searchMessage |
| URL du *Web service* | */searchMessage* |
| Description du *Web service* | Liste tous les twists des utilisateurs pour qui l’utilisateur est abonné |
| Paramètre(s) du *Web service* | La clé de connexion et le login de l’utilisateur, nombre de twists(static) , trie (boolean) |
| Format de sortie | *{"status" : "ok", "list\_message" :[ {"twist1" :*  *“hello”},...]}* ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | *{"status" : "ok", "list" :[ {"message1" : "Comment ca*  *va ?"}, {"message2" : Comme la "dernière fois !"]}* |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, er-  reur *JSON*, erreur base de données *MySQL* |
| Avancement du *Web service* | *Réalisée sur la page d’accueil* |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet ListTwistFollowers.java* |
| Informations additionnelles | Besoin de rafraichire la page côté client pour avoir la liste des twists à jour |

1. **Services commentaire**

Table 18: Fiche *addComment*

|  |  |
| --- | --- |
| Nom du *Web service* | addComment |
| URL du *Web service* | */addComment* |
| Description du *Web service* | Ajoute un commentaire de twist à la base de donnée *MongoDB* |
| Paramètre(s) du *Web service* | La clé de connexion, l’id du message auquel le commentaire fait référence, et le contenu du commentaire |
| Format de sortie | *{"status" : "ok", "added\_comment" :*  *le\_commentaire}* ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | *{"status" : "ok", "comment has been added" : "magnifique photo!"}* |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, erreur *JSON*, erreur base de données *MongoDB* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet AddComment.java* |
| Informations additionnelles |  |

Table 19: Fiche *removeComment*

|  |  |
| --- | --- |
| Nom du *Web service* | removeComment |
| URL du *Web service* | */removeComment* |
| Description du *Web service* | Supprime un commentaire de la base de données *MongoDB* |
| Paramètre(s) du *Web service* | La clé de connexion et l’id du commentaire à supprimer |
| Format de sortie | *{"status" : "ok", "removed\_message" : le\_message}*  ou *{"status" : "ko", "message" : message\_erreur, "code" : code\_erreur}* |
| Exemple de sortie | *{"Your comment has been removed" : "1"}* |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté erreur *JSON*, erreur base de données *MongoDB* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet RemoveComment.java* |
| Informations additionnelles |  |

Table 20: Fiche *ListComment*

|  |  |
| --- | --- |
| Nom du *Web service* | ListComment |
| URL du *Web service* | */ListComment* |
| Description du *Web service* | Liste tous les commentaires d’un twist |
| Paramètre(s) du *Web service* | La clé de connexion et l’id du message pour lequel nous voulons afficher les commentaires |
| Format de sortie | * comments: Array(2)   + 0:     - author: "user"     - content: "hey"     - date: "Mon Apr 15 15:28:21 CEST 2019"     - \_\_proto\_\_: Object   + 1:     - author: "user1"     - content: "hey there"     - date: "Mon Apr 15 15:28:29 CEST 2019"     - \_\_proto\_\_: Object |
| Exemple de sortie | *{"status" : "ok", "comments" :[ {"commentaire1" :*  *"t’as raison !"}, {"commentaire2" : "je ne suis pas tout a fait de votre avis..."}]}* |
| Erreurs possibles | Paramètre(s) manquant, utilisateur non connecté, erreur *JSON*, erreur base de données *MongoDB* |
| Avancement du *Web service* | Fini |
| Classe *JAVA* en rapport avec  le *Web service* | *Servlet ListComment.java* |
| Informations additionnelles | Pas besoin de rafraichir coté client affichage instantané |