Services User :

NOM	CreateUser
URL	/user/createUser
Description	Permet à un utilisateur de s'inscrire
Paramètres	nom, prenom, login, password, date, sexe
Format de sortie	{"status": « OK"} ou {"error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"error_message":"Le login existe déja.","errorID": 1003,"status":"KO"}
Erreurs possibles	paramètres manquants (-1), utilisateur existe déjà (1003), erreur de base de données(1005 ou 1000), mot de passe trop simple (1004)
Classe Java Correspondante	servlet.user.CreateUser

NOM	Login
URL	/user/login
Description	Permet à un utilisateur de se connecter, cela ajoute donc une entrée dans la table session et permet de générer une clef de connexion. Si root est coché, l'utilisateur reste connecté, sinon le temps sans faire d'actions est compté et la session expirera au bout de la variable durée.
Paramètres	login, password, root
Format de sortie	{« login »: <login>, « key »: <key> , »status" »: « OK",} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"login":"loulou","key":"9BFNKNFE347Q1OFWYTA 1A47CDZH0XBIH","status":"OK"}
Erreurs possibles	paramètres manquants (-1), login n'existe pas (1001), mauvais mot de passe ou login(1002), erreur de base de données(1005 ou 1000)
Classe Java Correspondante	servlet.user.Login

NOM	Logout
URL	/user/logout
Description	Permet à un utilisateur de se déconnecter. Cela enlève la clef de connexion dans la table session.
Paramètres	key
Format de sortie	{»status" »: « OK",} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"error_message":"Vous n'êtes pas connecté.","errorID":1005,"status":"KO"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), impossible de se déconnecter (1006), erreur de base de données(1000)
Classe Java Correspondante	servlet.user.Logout

NOM	GetUserInformations
URL	/user/getUserInformations
Description	Permet de récupérer quelques informations sur l'utilisateur par rapport à sa clef et à son login, notamment son id.
Paramètres	key, login
Format de sortie	{»status" »: « OK",} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"nombre de followers":7,"nombre de following »: 1,"id"2,"nom":"saleh","prenom":"sarah","status": "OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login n'existe pas, erreur de base de données(1000)
Classe Java Correspondante	servlet.user.GetUserInformations

NOM	IsFollowerInformation
URL	user/isFollowerInformation
Description	Permet de savoir si un utilisateur en suit un autre
Paramètres	key, userID
Format de sortie	{« status" »: « OK », « is_follower »: »<boolean>} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"is_follower":false,"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), erreur de base de données(1000)
Classe Java Correspondante	servlet.user.IsFollowerInformation

Services Friend :

NOM	AddFriend
URL	friend/addFriend
Description	Permet de suivre un utilisateur via son id.
Paramètres	key, id
Format de sortie	{« status" »: « OK », « following_added »: »<id>} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{{"following added ":5,"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login incorrect (1002), déjà ami(1011), impossible d'ajouter (1007), erreur de base de données(1000)
Classe Java Correspondante	servlet.friend.AddFriend

NOM	DeleteFriend
URL	friend/deleteFriend
Description	Permet de ne plus suivre un utilisateur via son id.
Paramètres	key, friendID
Format de sortie	{« status" »: « OK », « following_deleted »: »<id>} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"following deleted":2,"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login incorrect (1002), impossible car vous n'êtes pas ami(1008), impossible de supprimer (1009), erreur de base de données(1000)
Classe Java Correspondante	servlet.friend.DeleteFriend

NOM	ListFollowers
URL	friend/listFollowers
Description	Permet de connaitre les followers d'un utilisateur via son id.
Paramètres	key, id
Format de sortie	{« status" »: « OK », « list_followings »: » [{« id » :<id>, « login » : <login>} ,…]}ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{« list_followers":[ ],"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login incorrect (1002), erreur de base de données(1000)
Classe Java Correspondante	servlet.friend.ListFollowers

NOM	ListFollowings
URL	friend/listFollowers
Description	Permet de connaitre les personnes qui suit un utilisateur via son id.
Paramètres	key, id
Format de sortie	{« status" »: « OK », « list_followings »: » [{« id » :<id>, « login » : <login>} ,…]}ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"list_followings":[{"id":2,"login":"sarah"},{"id": 1,"login":"toto"}],"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login incorrect (1002), erreur de base de données(1000)
Classe Java Correspondante	servlet.friend.ListFollowings

Services messages :

NOM	AddPost
URL	message/addPost
Description	Permet à un utilisateur d'ajouter un message via sa clef de connexion.
Paramètres	key, message
Format de sortie	{« status" »: « OK », « date »:<date>, « message_added » : <content>, « comments »: [<comment>, « like »:{<like}, « author »: { « authorID » : <id>, « login »:<login>}, « _id » : <id>, « message » :<message> } ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"date":"Sun Apr 14 15:57:43 CEST 2019","message added":"\"hello\"","comments": [],"like":{},"author":{"authorID": 7,"login":"mimi"},"_id":"5cb33c57b99ef068e824c22 c","message":"\"hello\"","status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), erreur de base de données(1000)
Classe Java Correspondante	servlet.message.AddPost

NOM	DeletePost
URL	message/deletePost
Description	Permet de connaitre les personnes qui suit un utilisateur via son id.
Paramètres	key, messageID
Format de sortie	{« status" »: « OK »}, ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), le message n'existe pas (1009),le message n'appartient pas à l'utilsateur (1010) erreur de base de données(1000)
Classe Java Correspondante	servlet.message.DeletePost

NOM	AddComment
URL	message/addComment
Description	Permet d'ajouter un commentaire à un message via son id.
Paramètres	key, messageID, comment
Format de sortie	{« status" »: « OK » « comment_added »:<comment> , « id » : <id>, « date » :<date>, comment » : <comment>, « author » : { « authorID » : <id>, « login »:<login>}} ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"error_message":"Vous n'êtes pas connecté.","errorID":1005,"status":"KO"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), le post n'existe pas (1009), erreur de base de données(1000)
Classe Java Correspondante	servlet.message.AddComment

NOM	ListMessages
URL	friend/listFollowers
Description	Permet d'afficher selon les paramètres soit le fil d'actualité (avec id seulement), tous les messages d'un utilisateur (avec id et friends non nuls)
Paramètres	key, id, friends, nb
Format de sortie	{« status" »: « OK », « posts »: » [{« date» :date>, « comments » : [<comments>], « like » : {<like} , « messageID » : <messageID>, « authorID » : {« authorID » : <authorID>, « login » : <login>}, « message » : <message> } ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"posts":[],"status":"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), login incorrect (1002), erreur de base de données(1000)
Classe Java Correspondante	servlet.message.ListMessages

NOM	ListAllMessages
URL	message/listAllMessages
Description	Permet de lister tous les messages existants (utile quand un utilisateur vient de s'inscrire ou ne suit personne)
Paramètres	key
Format de sortie	{« status" »: « OK », «messages »: » [{« date» :date>, « comments » : [<comments>], « like » : {<like} , « messageID » : <messageID>, « authorID » : {« authorID » : <authorID>, « login » : <login>}, « message » : <message> } ou {« error_message":<message>,"errorID": <codeErreur> ,"status":"KO"}
Exemple de sortie	{"messages":[{"date":"Sun Apr 14 16:11:42 CEST 2019","comments":[{"date":"Sun Apr 14 16:12:14 CEST 2019 »,"comment":"\"hi\"","id":0,"author": {"commentatorID":7,"login":"mimi"}}],"like": {},"author":{"authorID": 7,"login":"mimi"},"_id":"5cb33f9eb99ef068e824c24 0","message":"\"hello\""}, …, status »:"OK"}
Erreurs possibles	paramètres manquants (-1), l'utilisateur n'est pas connecté (1005), erreur de base de données(1000)
Classe Java Correspondante	servlet.message.listAllMessages