Nom du web service	Login
URL du web service	user/login/ avec POST
Description du service	Permet de récupérer une clef de connexion valide pendant un certain temps
Paramètres en entrée	login; password
Format de sortie	JSON
Exemple de sortie	● { "status": 200, "message": "Connexion réussie"} ● { "status": 400, "message": "Requête invalide : login et password nécessaires"} ● { "status": 401, "message": "Utilisateur inconnu"} ● { "status": 403, "message": "Login et/ou le mot de passe invalide(s)"} ● { "status": 500, "message": "Erreur interne"}
Erreurs possibles	● Champs manquants(400) ● Utilisateur inconnu (401) ● Login et/ou le mot de passe invalide(s) (403) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	api.js
Informations additionnelles	1. Si tout se passe bien renvoyer le code 200 et Connexion réussie, et création de la session avec req.session.userid 2. Tous les champs doivent être complétés, si non -> 400 3. Vérification de l'existence du login, si non -> 401 4. Vérification du login et mot de passe, si non -> 403

Nom du web service	Logout
URL du web service	user/logout/ avec POST
Description du service	Permet la déconnexion de l'utilisateur au service et la suppression de la clé de connexion
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { "status": 200, "message": "Déconnexion réussie"} ● { "status": 401, "message": "Non connecté"} ● { "status": 500, "message": "Erreur interne"}
Erreurs possibles	● Non connecté (401) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	api.js
Informations additionnelles	1. Vérification de la connexion, si non connecté -> 401 2. Si tout se passe bien, code 200 et "Déconnexion réussie"

Nom du web service	CreateUser
URL du web service	user/ avec PUT
Description du service	Permet la création d'un nouvel utilisateur du service
Paramètres en entrée	login; password; firstName, lastName
Format de sortie	JSON
Exemple de sortie	• { "status": 201, "message": "Utilisateur créé avec succès", "details": ""} • { "status": 400, "message": "Champs manquants"} • { "status": 401, "message": "Utilisateur déjà existant", "details": "Pikachu déjà enregistré"}
Erreurs possibles	• paramètres manquants, requête pas en JSON (400) • login déjà existant (401) • le serveur de bdd ne répond pas (504)
Avancement du Service	Fini
Classes/Fichiers JavaScript	Users/Users.js
Informations additionnelles	1. Vérification des paramètres (il ne faut pas que les paramètres soient nuls), sinon bad request -> 400 2. Vérifier si un utilisateur avec le même login existe déjà. Si oui erreur -> 401 3. Ajouter l'utilisateur à la base de données (avec Users.create) 4. Renvoyer 201 et message si tout s'est bien passé

Nom du web service	GetUser
URL du web service	user/:user_id/ avec GET
Description du service	Permet d’obtenir un utilisateur avec son id
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { “_id”: “3495hjk76hk6ugj098765434”, “login”: “pikachu”, “password”: 1234, “lastName”: “pika”, “firstName”: “chu”} ● { "status": 404, "message": "Utilisateur non trouvé"} ● {"status": 500, "message": "Erreur interne"}
Erreurs possibles	● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Uses/Users.js
Informations additionnelles	1. Lecture de l’utilisateur à la base de donnée avec Users.get 2. Si tout se passe bien, code 200, et le JSON de l'utilisateur correspondant 3. Sinon, code 404, l’utilisateur n'existe pas

Nom du web service	DeleteUser
URL du web service	user/:user_id/ avec DELETE
Description du service	Permet la suppression d'un utilisateur par son id
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	• { "status": 200 "status": "Suppression de l'utilisateur 3495hjk76hk6ugj098765434 réussie"} • { "status": 401, "message": "Non connecté"} • { "status": 500, "message": "Erreur interne"}
Erreurs possibles	• Non connecté (401) • Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Uses/Users.js
Informations additionnelles	1. Vérification de la connexion de l'utilisateur, non connecté->401 2. Suppression de l'utilisateur de la base de donnée avec Users.delete 3. Si tout se passe correctement code 200, "Suppression de l'utilisateur $_id réussie"

Nom du web service	EditUser
URL du web service	user/:login/edit avec POST
Description du service	Permet d'éditer son profil
Paramètres en entrée	login, password, confirmPassword, lastName, firstName, birthday, bio
Format de sortie	JSON
Exemple de sortie	● { "status": 200, "message": Modification enregistré} ● { "status": 400, "message": "Champs manquants"} ● { "status": 401, "message": "Non connecté"} ● { "status": 405, "message": "Utilisateur déjà existant" "détails": "pikachu existe déjà"} ● { "status": 500, "message": "Erreur interne"}
Erreurs possibles	● Champs manquants (400) ● Non connecté (401) ● Utilisateur déjà existant (405) ● Erreur interne (500)
Avancement du Service	à finir
Classes/Fichiers JavaScript	Uses/Users.js
Informations additionnelles	1. Vérification de la connexion, si non connecté -> 401 2. Tous les champs doivent être 3. Existence de l'utilisateur, si non -> 401 4. Edition de l'utilisateur avec Users.edit 5. Si tout se passe bien -> 200, "Modification enregistrée"

Nom du web service	CreateMessage
URL du web service	user/:login/newMessage/ avec PUT
Description du service	Permet à l'utilisateur correspondant au login de poster un nouveau message en l'ajoutant dans sa liste de message
Paramètres en entrée	login, date, texte
Format de sortie	JSON, texte
Exemple de sortie	● { "status": 201, "message": "Nouveau message publié"} ● { "status": 400, "message": "Champs manquants"}
Erreurs possibles	● Champs manquants (400) ● Non connecté (401) ● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion, non connecté -> 401 2. Complété tous les champs, sinon -> 400 3. Existence de l'utilisateur ? -> 404 4. Ajout du message à la base de donnée avec Messages.create 5. Si tout se passe bien, -> 201 "Nouveau message publié"

Nom du web service	GetMessages
URL du web service	user/:login/messages/ avec GET
Description du service	Permet d'obtenir la liste des messages d'un utilisateur
Paramètres en entrée	xxxx
Format de sortie	JSON, Tableau
Exemple de sortie	• [{ "login": "pikachu", "date": "5/02", "texte": "Mon premier message"}, {…}] • { "status": 202, "message": Aucun message trouvé} • { "status": 401, "message": "Non connecté"}
Erreurs possibles	• Non connecté (401) • Utilisateur non trouvé (404) • Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion, non connecté -> 401 2. Existence de l'utilisateur, -> 404 3. Lecture dans la base de donnée des messages avec Messages.get 4. Si tout se passe bien, -> 200 renvoie la liste de message de l'utilisateur 5. Si tout se passe bien mais qu'il n'a aucun message -> 202 "Aucun message trouvé"

Nom du web service	DeleteMessage
URL du web service	user/:login/messages/ avec DELETE
Description du service	Permet à l'utilisateur de supprimer ses messages
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { "status": 200, "message": "Message supprimé"} ● { "status": 401, "message": "Non connecté",} ● { "status": 404, "message": "Message non trouvé"}
Erreurs possibles	● Non connecté (401) ● Message non trouvé (404) ● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Existence de l'utilisateur et du message -> 404 3. Supprime de la base de donnée le message de l'utilisateur correspondant (avec Messages.delete) 4. Si tout se passe bien -> 200 "Message supprimé"

Nom du web service	CreateFriend
URL du web service	user/:login/newFriend/ avec PUT
Description du service	Permet d'ajouter un utilisateur en ami
Paramètres en entrée	friend_login
Format de sortie	JSON
Exemple de sortie	● { "status": 201, "message": "Nouvel ami ajouté"} ● { "status": 400, "message": "Champs manquants"} ● { "status": 400, "message": "Déjà ami"}
Erreurs possibles	● Champs manquants (400) ● Déjà ami (400) ● Non connecté (401) ● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	5. Vérification de la connexion -> 401 6. Tester l'existence de l'utilisateur -> 404 7. On regarde si on ne l'a pas encore en ami, sinon -> 400 8. Ajout de l'ami dans la liste d'amis de l'utilisateur (Friend.create) 9. Si tout se passe bien -> 201 "Ami ajouté"

Nom du web service	GetFriends
URL du web service	user/:login/friends/ avec GET
Description du service	Permet d'obtenir la liste d'amis d'un utilisateur
Paramètres en entrée	xxxx
Format de sortie	JSON, Tableau
Exemple de sortie	• [{"friend_login": "pikachu"}, {"friend_login": "alex"}, …] • { "status": 202, "message": "Aucun ami trouvé"} • { "status": 401, "message": "Non connecté"}
Erreurs possibles	• Non connecté (401) • Utilisateur non trouvé (404) • Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. On teste l'existence de l'utilisateur -> 404 3. On lit dans la base de donnée la liste d'amis de l'utilisateur (Friends.get) 4. Si tout se passe bien -> 200 et la liste d'amis renvoyée 5. Si tout se passe bien mais aucun ami dans la liste -> 202

Nom du web service	DeleteFriend
URL du web service	user/:login/:friend_login/ avec DELETE
Description du service	Permet de supprimer un ami
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { “status”: 200, “message”: “Ami supprimé”} ● { “status”: 404, “message”: “Ami non trouvé”}
Erreurs possibles	● Ami non trouvé (404) ● Non connecté (401) ● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	fini
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Tester l’existence de l’utilisateur et de son ami, sinon -> 404 3. Suppression de l’ami dans la liste d’ami de l'utilisateur (Friend.delete) 4. Si tout se passe bien -> 200 “Ami supprimé”

Nom du web service	Search
URL du web service	search/ avec GET
Description du service	Permet d'obtenir une information associée à la recherche
Paramètres en entrée	searchString
Format de sortie	JSON, Tableau
Exemple de sortie	● [{ "login": "pikachu", "date": "5/02", "texte": "Mon premier message"}, { "_id": "3495hjk76hk6ugj098765434", "login": "pikachu", "password": 1234, "lastName": "pika", "firstName": "chu"}, {…}] ● { "status": 404, "message": "Aucune information correspondant à la recherche"}
Erreurs possibles	● Aucune information correspondant à la recherche (404) ● Non connecté (401) ● Champs incomplets (400)
Avancement du Service	à faire
Classes/Fichiers JavaScript	xxx
Informations additionnelles	1. Vérification de la connexion -> 401 2. Doit contenir une recherche, sinon -> 400 3. Si tout se passe bien renvoie une liste de recherche, utilisateur et/ou message -> 200

Nom du web service	LikeMessage
URL du web service	user/:login/:messUser/:message_id/ avec POST
Description du service	Permet d'aimer une publication
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { "status": 200, "message": "Vous avez aimez un message"} ● { "status": 401, "message": "Non connecté"} ● { "status": 404, "message": "Message non trouvé"}
Erreurs possibles	● Non connecté (401) ● Utilisateur non trouvé (404) ● Message non trouvé (404) ● Erreur interne (500)
Avancement du Service	à faire
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Existence de l'utilisateur et du message -> 404 3. Incrémente le compteur de like du message, et ajoute le login de l'utilisateur dans la liste des utilisateurs ayant aimé le message (Message.like) 4. Si tout se passe bien -> 200 "Vous avez aimé un message"

Nom du web service	CreateConversationPrivé
URL du web service	user/:login/:messUser/mp/ avec PUT
Description du service	Permet de créer une conversation privée avec un autre utilisateur
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { “status”: 201, “message”: “Démarre une conversation privée”} ● { “status”: 401, “message”: “Non connecté”} ● { “status”: 202, “message”: “Redirection vers la conversation privée”}
Erreurs possibles	● Non connecté (401) ● Utilisateur non trouvé (404) ● Erreur interne (500)
Avancement du Service	à faire
Classes/Fichiers JavaScript	MP/messageprive.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Existence de l’utilisateur -> 404 3. Ajoute une conversation privée dans la liste des conversations privées de l’utilisateur (MP.create) 4. Si tout se passe bien -> 200 “Envoyez un petit un coucou :)” 5. Si la conversation existe déjà, redirection vers la conversation -> 202

Nom du web service	DeleteConversationPrivée
URL du web service	user/:login/:messUser/mp/ avec DELETE
Description du service	Permet de supprimer une conversation privée avec un autre utilisateur
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { “status”: 200, “message”: “Conversation supprimée avec succès”} ● { “status”: 401, “message”: “Non connecté”} ● { “status”: 404, “message”: “Conversation non trouvé”}
Erreurs possibles	● Non connecté (401) ● Utilisateur / message non trouvé (404) ● Erreur interne (500)
Avancement du Service	à faire
Classes/Fichiers JavaScript	MP/messageprive.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Existence de l’utilisateur et du message -> 404 3. Supprime une conversation privée dans la liste des conversations privées de l’utilisateur (MP.delete) 4. Si tout se passe bien -> 200 “Conversation supprimée avec succès”

Nom du web service	SendMessagePrivé
URL du web service	user/:login/:messUser/:mp/send avec PUT
Description du service	Permet d'envoyer un message privée à un utilisateur
Paramètres en entrée	content, date
Format de sortie	JSON
Exemple de sortie	● { "status": 204} ● { "status": 401, "message": "Non connecté"} ● { "status": 404, "message": "Utilisateur non trouvé"}
Erreurs possibles	● Non connecté (401) ● Utilisateur / conversation (404) ● Erreur interne (500)
Avancement du Service	à faire
Classes/Fichiers JavaScript	MP/messageprive.js
Informations additionnelles	1. Vérification de la connexion -> 401 2. Existence de l'utilisateur et de la conversation -> 404 3. Ajoute un message privé dans la liste des messages privées de la conversation (MP.send) 4. Si tout se passe bien -> 204

Nom du web service	DeleteMessagePrivée
URL du web service	user/:login/:messUser/:mp/:id avec DELETE
Description du service	Permet de supprimer un message dans la conversation avec un utilisateur
Paramètres en entrée	xxxx
Format de sortie	JSON
Exemple de sortie	● { “status”: 200, “message”: “Message supprimée avec succès”} ● { “status”: 401, “message”: “Non connecté”} ● { “status”: 404, “message”: “Message non trouvé”}
Erreurs possibles	● Non connecté (401) ● Utilisateur / conversation non trouvé (404) ● Erreur interne (500)
Avancement du Service	à faire
Classes/Fichiers JavaScript	Messages/messages.js
Informations additionnelles	5. Vérification de la connexion -> 401 6. Existence de l’utilisateur et du message -> 404 7. Supprime un message dans la conversations privée de l’utilisateur (MP.deleteMP) 8. Si tout se passe bien -> 200 “Message supprimée avec succès”