SORBONNE UNIVERSITE
Faculté des Sciences et Ingénerie

Licence Informatique - L3

TECHNOLOGIES WEB

Documentation du projet

DUCKY

Présentation du site

Client, Composants, Serveur, Services, Base de données

Auteurs

Ben Kabongo Buzangu
Théo Charlot

Janvier 2022 - Mai 2022

Table des matières

Première partie

Client

Nous allons présenter les composants React pour notre site web, **Ducky**, lequel reprend la philosophie de Twitter. Le canard de Ducky s'insipire de l'oiseau de Twitter.

Nous nous sommes voulus créatifs pour ce projet, et avons, à l'avance, pensé à la structure finale que nous voulons obtenir. Nous avons donc, pour le client de notre site, utilisé le patron de conception **Modèle - Vue - Contrôleur**, lequel nous permet de complètement séparer la logique derrière notre site et son rendu graphique (les composants).

Nous allons commencer par faire un tour d'horizon rapide du modèle de notre site web, car les éléments de ce modèle sont ensuite utilisés comme props ou états pour les composants. Par la suite, nous allons présenter les composants les plus pertinents : leurs props, leurs états, leurs fonctions, et indiquer où nous en sommes dans leur implémentation.

1 Modèle

Le modèle reprend la structure de notre site. Nous l'avons divisé en grandes entités, qui gèrent diverses briques de notre site, à savoir : les utilisateurs, les messages, les posts, les groupes, les stories, les sondages.

Nous nous baserons sur la structure de ce modèle notamment pour facilement gérer les props et les états internes des composants, également pour certains éléments du serveur comme les routes et pour les fondements de la base de données.

Nous avons implémenté les différents objets et classes du modèle dans le package **src/model/objects**.

1.1 Les utilisateurs

Cette partie du modèle gère les utilisateurs.

On y retrouve par exemple la classe **User** dans laquelle on renseigne les informations sur un utilisateur, telles que : le nom d'utilisateur, le nom, le prénom, le sexe, etc.

Un composant qui prend en props ou en état un objet utilisateur a donc accès à toutes ces informations, et les affiche à sa convenance. Cette logique s'applique pour tous les éléments de notre modèle.

Précisons que les informations contenues dans ces objets seront chargées depuis la base de données du site et ensuite utilisées par les composants.

1.2 Les messages

Cette partie du modèle gère les messages.

On y retrouve par exemple la classe **Message** dans laquelle on renseigne les informations sur un message, telles que : l'heure d'envoi, le texte du message, etc.

1.3 Les posts

Cette partie du modèle gère les posts.

On y retrouve par exemple la classe **Post** qui renseigne les informations sur un post, comme l'heure de publication, le texte du post, les likes, les commentaires, etc.

1.4 Les groupes

Cette partie du modèle gère les groupes de messagerie et de posts.

1.5 Les stories

Cette partie du modèle gère les stories : une forme de post visible pour une durée et une audience limitées.

1.6 Les sondages

Cette partie du modèle gère les sondages, que les utilisateurs peuvent créer et auxquels d'autres peuvent répondre.

2 Arbre des composants

3 Composants principaux

Deuxième partie

Base de données

Ci-dessous, nous présentons la structure de la base de données de notre site. Elle est divisée en différentes collections, s'inspirant du modèle. Ces dernières sont : users, messages, posts, stories, groups.

Certaines collections partagent quelques structures communes, telles que les **poll** pour les sondages, les **media** pour les média, les **message** pour les messages.

Nous allons présenter la structures des données pour les différentes collections.

1 Structures communes

1.1 Sondages : poll

- **id** *string, requis, unique* : identifiant unique du sondage.
- **title** *string, requis* : titre du sondage
- **creation_time** *date-time, requis* : instant de création du sondage.
- **closing_time** *date-time* : instant de fin du sondage.
- **closed** *boolean* : *true* si le sondage est fermé.
- **deleted** *boolean* : *true* si le sondage est supprimé.
- **items** *list(dict)* : liste des items du sondage.
  - **id** *int* : numéro de l'item.
  - **text** *string* : texte de l'item.
  - **votes** *list(dict)* : liste des votes de l'item.
    - **user_id** *string, requis, unique* : identifiant de l'utilisateur.
    - **time** *date-time* : instant du vote.

1.2 Media : media

- **id** *int* : numéro du média.
- **type** *string* : type de média : audio, image, video, document, etc.
- **src** *string* : source du média.

1.3 Message : message

- **id** *string, requis, unique* : identifiant du message.
- **user_id** *string, requis* : identifiant de l'auteur du message.
- **reply_to_id** *string* : identifiant du message auquel le message courant est une réponse.
- **time** *date-time, requis* : instant d'envoi du message.

— **text** *string* : contenu textuel du message.
— **polls** *list(poll)* : liste des sondages.
— **media** *list(media)* : liste des médias : images, vidéos, audios, documents, etc.
— **hashtags** *list(string)* : liste des hashtags du message.
— **links** *list(str)* : liste des liens dans le message.
— **mentionned_users_ids** *list(string)* : identifiants (ou noms d'utilisateurs) des utilisateurs mentionnés dans le message.
— **status** *string* : status du message : *sent* envoyé, **notSent** non envoyé, *deleted* supprimé.
— **likes** *list(dict)* : listes des likes du message.
  — **user_id** *string, requis, unique* : identifiant de l'utilisateur ayant liké le message.
  — **time** *date* : instant du like.
— **views** *list(dict)* : listes des vues du message.
  — **user_Id** *string, requis, unique* : identifiant de l'utilisateur.
  — **time** *date* : instant de mise à jour du status.
  — **status** *string* : status de vue de l'utilisateur : *seen* vu, *received* reçu, *distributed* distribué.

2 Collection pour les utilisateurs : users

— **_id**
— **user_id** *string, requis, unique* : identifiant unique de chaque utilisateur.
— **password** *string, requis, unique* : mot de passe unique de chaque utilisateur.
— **username** *string, requis, unique* : nom d'utilisateur, modifiable par l'utilisateur.
— **firstname** *string, requis*
— **lastname** *string, requis*
— **sex** *string, requis*
— **profile_picture_src** *image* : photo de profil.
— **phone** *string* : numéro de portable.
— **mail** *string* : adresse mail.
— **city** *string* : ville actuelle.
— **country** *string* : pays actuel.
— **birthday** *date* : date de naissance.
— **creation_time** *date, requis* : date de création du compte.
— **private** *boolean* : *true* si le compte est privé.
— **deleted** *boolean* : *true* si le compte est supprimé.
— **login_time** *date-time* : instant du login. A chaque connexion de l'utilisateur à son compte, on y enregistre l'instant de début de connexion. Lors de la déconnexion, rajouter les instants de début et de fin dans l'historique des connexions.
— **online** *boolean* : *true* si l'utilisateur est en ligne.

— **online_history** *list(dict)* : historique des connexions utilisateurs.
  — **start** *date*
  — **end** *date*
— **ban** *dict* : informations sur le bannissement courant de l'utilisateur.
  — **start** *date-time* : début.
  — **end** *date-time* : fin.
  — **reason** *string* : raison.
— **ban_history** *list(dict)* : historique des banissements.
  — **start** *date-time*
  — **end** *date-time*
  — **reason** *string*
— **followers** *list(dict)* : liste des abonnés de l'utilisateur.
  — **user_id** *string, requis, unique* : identifiant de l'abonné.
  — **time** *date-time*
— **followings** *list(dict)* : liste des abonnementss de l'utilisateur.
  — **user_id** *string, requis, unique* : identifiant de l'abonnement.
  — **time** *date-time*
— **views** *list(dict)* : historique des profils visités par l'utilisateur.
  — **user_id** *string, requis* : identifiant du profil utilisateur vu.
  — **time** *date-time*
— **views_by** *list(dict)* : historique des vues du profil de l'utilisateur.
  — **user_id** *string, requis* : identifiant de l'utilisateur ayant visité le profil.
  — **time** *date-time*
— **reports** *list(dict)* : liste des signalements par l'utilisateur.
  — **user_id** *string, requis* : identifiant de l'utilisateur signalé.
  — **time** *date-time*
  — **reason** *string*
— **reports_by** *list(dict)* : liste des signalements de l'utilisateur.
  — **user_id** *string, requis* : identifiant de l'utilisateur qui a signalé le profil.
  — **time** *date-time*
  — **reason** *string*
— **blocks** *list(dict)* : liste des profils bloqués par l'utilisateur.
  — **user_id** *string, requis* : identifiant de l'utilisateur bloqué.
  — **time** *date-time*
  — **reason** *string*
— **blocks_by** *list(dict)* : liste des profils qui ont bloqué l'utilisateur.
  — **user_id** *string, requis* : identifiant de l'utilisateur qui a bloqué le profil.

    - **time** *date-time*
    - **reason** *string*
- **messages** *list(dict)* : liste des messages de l'utilisateur.
    - **type** *string, requis* : type de messages : *simple* entre l'utilisateur et un autre utilisateur, *group* messages d'un groupe duquel l'utilisateur fait parti.
    - **messages_id** *string, requis* : identifiant des messages simples ou du groupe de message, en fonction du type.
    - **other_user_id** *string* : identifiant de l'autre utilisateur, quand il s'agit de messages simples.
    - **group_id** *string* : identifiant du groupe, quand il s'agit de messages de groupe.
- **notifications list(dict)** : liste des notifications de l'utilisateur.
    - **id** *string, requis, unique* : identifiant unique de la notification.
    - **title** *string* : titre de la notification.
    - **description** *string* : description de la notification.
    - **type** *string* : type de la notification.
    - **time** *date-time, requis* : instant de création de la notification.
    - **image** *image* : image associée à la notification.
    - **link** *string, requis* : lien associé à la notification.
    - **status** *status* : statut de la notification : inidique si elle a été lue, vue ou cliquée par l'utilisateur.
- **posts_ids** *list(string)* : liste des identifiants des posts de l'utilisateur.
- **liked_posts_ids** *list(string)* : liste des identifiants des posts likés par l'utilisateur.
- **search_history** *list(dict)* : historique de recherche de l'utilisateur.
    - **search** *string, requis* : intitulé de la recherche.
    - **time** *date-time, requis* : instant de la recherche.
    - **deleted** *boolean* : *true* si l'utilisateur a supprimé la recherche.
- **stories_groups** *list(dict)* : groupes de stories de l'utilisateur.
    - **id** *string, requis, unique* : identifiant du groupe de stories.
    - **name** *string, requis* : nom du groupe de stories.
    - **description** *string* : description du groupe de stories.
    - **creation_time** *date-time* : instant de création du groupe de stories.
    - **deleted** *boolean* : *true* si le groupe de stories a été supprimé.
    - **members** *list(dict)* : liste des membres du groupe de stories.
        - **user_id** *string, requis, unique* : identifiant du membre.
        - **time** *date-time* : instant d'ajout au groupe.
        - **status** *string* : status du membre : membre, supprimé.
- **stories** *list(dict)* : liste des stories publiées par l'utilisateur.

—

— **stories_by** *list(dict)* : liste des stories livrées à l'utilisateur.

3 Collection pour les messages : messages

— **_id**

— **messages_id** *string, requis, unique* : identifiant de l'échange.

— **users_id** *list(string)* : listes des identifiants des utilisateurs concernés par l'échange.

— **messages** *list(message)* : listes des messages échangés entre les utilisateurs.

4 Collection pour les posts : posts

— **_id**

— **post_id** *string, requis, unique* : identifiant unique du post.

— **user_id** *string, requis* : identifiant de l'utilisateur créateur du post.

— **reply_to_id** *string* : identifiant du post dont ce post est une réponse.

— **time** *date-time, requis* : instant de publication du post.

— **text** *string* : contenu textuel du post.

— **location** *string* : localisation indiquée dans le post : pays, ville, lieux, etc.

— **private** *boolean* : *true* si le post est privé.

— **deleted** *boolean* : *true* si le post a été supprimé.

— **banished** *boolean* : *true* si le post a été bannis.

— **banished_reason** *string* : raison du bannissement.

— **polls** *list(poll)* : liste des sondages.

— **media** *list(media)* : liste des médias : images, vidéos, audios, documents, etc.

— **hashtags** *list(string)* : liste des hashtags du post.

— **mentionned_users_ids** *list(string)* : identifiants (ou noms d'utilisateurs) des utilisateurs mentionnés dans le post.

— **likes** *list(dict)* : listes des likes.

  — **user_id** *string, requis, unique* : identifiant de l'utilisateur qui a liké le post.

  — **time** *date-time* : instant du like.

— **shares** *list(dict)* : listes des partages.

  — **user_id** *string, requis* : identifiant de l'utilisateur qui a partagé le post.

  — **time** *date-time* : instant du partage.

— **views** *list(dict)* : listes des vues.

  — **user_id** *string, requis* : identifiant de l'utilisateur qui a vu le post.

  — **time** *date-time* : instant de la vue du post.

— **reports** *list(dict)* : listes des signalements.

- **user_id** *string, requis* : identifiant de l'utilisateur qui a signalé le post.
- **time** *date-time* : instant du signalement.
- **reason** *string* : raison du signalement.

5 Collection pour les stories : stories

6 Collection pour les groupes : groups

- **_id**
- **group_id** *string, requis, unique* : identifiant unique du groupe.
- **name** *string, requis* : nom du groupe.
- **description** *string* : description du groupe.
- **creation_time** *date-time, requis* : instant de création du groupe.
- **deleted** *boolean* : *true* si le groupe a été supprimé.
- **members** *list(dict)* : membres du groupe.
  - **user_id** *string, requis, unique* : identifiant de l'utilisateur.
  - **status** *string* : status du membre : *admin* adminisatrateur et membre, *member* membre, *deleted* supprimé.
  - **membership_time** *date-time* instant d'adhésion.
- **news** *list(dict)* : nouveautés (notifications) de groupe.
  - **users_id** *list(string)* : liste des identifiants des utilisateurs directement concernés.
  - **comments** *string* : commentaires.
- **messages** *list(message)* : liste des messages du groupe.

Troisième partie

Services

1 Services pour les utilisateurs

1.1 Compte utilisateur

1.1.1 login

Connecte un utilisateur sur sa page dans le site.

— *URL :* **POST /1/users/login**

— *Entrées :* **username** *string, requis ou* **mail** *string, requis ou* **phone** *string, requis*, **password** *string, requis* : mot de passe

— *Sorties :* OK 200

— *Erreurs :*

1.1.2 logout

Déconnexion d'un utilisateur, fermeture de la session.

— *URL :* **DELETE /1/users/ :user_id/logout**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* OK 200

— *Erreurs :*

1.1.3 createUser

Création d'un nouvel utilisateur.

— *URL :* **POST /1/users**

— *Entrées :* **lastname** *string, requis*, **firstname** *string, requis*, **username** *string, requis*, **password** *string, requis*, **sex** *string*, **phone** *string*, **mail** *string*, **country** *string*, **city** *string*, **birthday** *date-time*, **biography** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.1.4 getUsers

Renvoie des informations sur une liste d'utilisateurs dont on passe les identifiants ou les usernames.

— *URL :* **GET /1/users**

— *Entrées :* **user_ids** *list(string)* : liste d'identifiants des utilisateurs *ou* **usernames** *list(string)* : liste de noms d'utilisateurs

— *Sorties :* **OK list(dict)** : chaque item contient des informations sur un utilisateur.

— *Erreurs :*

1.1.5 getUser

Renvoie les informations sur l'utilisateur.

— *URL :* **GET /1/users/ :user_id**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK dict** : informations sur l' utilisateur.

— *Erreurs :*

1.1.6 getUserMe

Renvoie les informations sur l'utilisateur courant.

— *URL :* **GET /1/users/me**

— *Entrées :* -

— *Sorties :* **OK dict** : informations sur l' utilisateur.

— *Erreurs :*

1.1.7 getUserByUsername

Renvoie les informations sur l'utilisateur dont on passe le nom d'utilisateur

— *URL :* **GET /1/users/by/username/ :username**

— *Paramètres :* **username** *string* : nom d'utilisateur.

— *Entrées :* -

— *Sorties :* **OK dict** : informations sur l' utilisateur.

— *Erreurs :*

1.1.8 updateUser

Mise à jour des informations utilisateurs.

— *URL :* **PATCH /1/users/ :user_id**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* informations à mettre à jour.

— *Sorties :* OK 200

— *Erreurs :*

1.1.9 deleteUser

Marquage de l'utilisateur comme étant supprimé.

— *URL :* **DELETE /1/users/ :user_id**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* OK 200

— *Erreurs :*

1.2 Followers, Followings

Gestion des followers et des followings.

1.2.1 getFollowers

Récupère la liste des followers de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/followers**

— *Paramètres :*

— **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*

— *Erreurs :*

1.2.2 getFollowings

Récupère la liste des followings l'utilisateur.

— *URL :* **GET /1/users/ :user_id/followings**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*

— *Erreurs :*

1.2.3 addFollower

Ajoute un follower à l'utilisateur.

— *URL :* **POST /1/users/ :user_id/follower**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis* : identifiant du follower, **time** *date-time, requis* : instant du follow.

— *Sorties :* OK 200

— *Erreurs :*

1.2.4 addFollowing

Ajoute un following à l'utilisateur.

— *URL :* **POST /1/users/ :user_id/following**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis* : identifiant du following, **time** *date-time, requis* : instant du follow.

— *Sorties :* OK 200

— *Erreurs :*

1.2.5 delFollower

Supprime un follower de l'utilisateur.

— *URL :* **DELETE /1/users/ :user_id/follower**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis* : identifiant du follower.

— *Sorties :* OK 200

— *Erreurs :*

1.2.6 delFollowing

Supprime un following de l'utilisateur.

— *URL :* **DELETE /1/users/ :user_id/following**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur

— *Entrées :* **user_id** *string, requis* : identifiant du following.

— *Sorties :* OK 200

— *Erreurs :*

1.3 Blocks

Gestion des blocages utilisateurs.

1.3.1 getBlocks

Récupère la liste des utilisateurs bloqués par l'utilisateur

— *URL :* **GET /1/users/ :user_id/blocks**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*, **reason** *string*

— *Erreurs :*

1.3.2 getBlocksBy

Récupère la liste des utilisateurs qui ont bloqué l'utilisateur.

— *URL :* **GET /1/users/ :user_id/blocksby**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*, **reason** *string*

— *Erreurs :*

1.3.3 addBlock

Ajout d'un nouvel utilisateur bloqué.

— *URL :* **POST /1/users/ :user_id/block**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis*, **time** *date-time, requis*, **reason** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.3.4 delBlock

Déblocage d'un utilisateur par un autre.

— *URL :* **DELETE /1/users/ :user_id/block**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis* : identifiant de l'utilisateur bloqué.

— *Sorties :* OK 200

— *Erreurs :*

1.4 Reports

Gestion des signalements utilisateurs.

1.4.1 getReports

Récupère la liste des utilisateurs signalés par l'utilisateur

— *URL :* **GET /1/users/ :user_id/reports**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis* **time** *date-time* **reason** *string*

— *Erreurs :*

1.4.2 getReportsBy

Récupère la liste des utilisateurs qui ont signalé l'utilisateur.

— *URL :* **GET /1/users/ :user_id/reportsby**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*, **reason** *string*

— *Erreurs :*

1.4.3 addReport

Ajout d'un nouvel utilisateur signalé.

— *URL :* **POST /1/users/ :user_id/report**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis*, **time** *date-time, requis*, **reason** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.5 Views

Gestion des vues de profils utilisateurs.

1.5.1 getViews

Récupère la liste des profils utilisateurs visités par l'utilisateur

— *URL :* **GET /1/users/ :user_id/views**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*

— *Erreurs :*

1.5.2 getViewsBy

Récupère la liste des utilisateurs qui ont visité le profil de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/viewsby**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis*, **time** *date-time*

— *Erreurs :*

1.5.3 addView

Ajout d'une nouvelle vue de profil.

— *URL :* **POST /1/users/ :user_id/view**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **user_id** *string, requis* : identifiant du profil utilisateur vu, **time** *date-time, requis* : instant de la vue de profil.

— *Sorties :* OK 200

— *Erreurs :*

1.6 Messages

Les messages sont complètement gérés par les services de message.

Dans cette section, nous décrirons les services de messages du point de vue d'un utilisateur donné. Un utilisateur peut entretenir des échanges avec soit un autre utilisateur soit un groupe donné d'utilisateurs.

1.6.1 getUserMessages

Récupération de la liste des échanges de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/messages**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **type** *string, requis*, **messages_id** *string, requis*, **other_user_id** *string*, **group_id** *string*

— *Erreurs :*

1.6.2 createUserMessages

Création d'un échange.

— *URL :* **POST /1/users/ :user_id/messages**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **type** *string, requis*, **messages_id** *string, requis*, **other_user_id** *string*, **group_id** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.6.3 updateUserMessages

Mise à jour d'un échange.

— *URL :* **PATCH /1/users/ :user_id/messages**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **messages_id** *string*, **other_user_id** *string*, **group_id** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.6.4 deleteUserMessages

Suppression d'un échange.

— *URL :* **DELETE /1/users/ :user_id/messages**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **type** *string, requis*, **messages_id** *string ou* **other_user_id** *string ou* **group_id** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.7 Posts

Les posts sont complètement gérés par les services de posts.

Dans cette section, nous décrirons les services de posts du point de vue d'un utilisateur donné.

1.7.1 getUserPosts

Récupération des identifiants des posts de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/posts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(string)** : liste des identifiants des posts de l'utilisateur.

— *Erreurs :*

1.7.2 addUserPost

Ajout l'identifiant d'un post de l'utilisateur.

— *URL :* **POST /1/users/ :user_id/posts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **post_id** *string* : identifiant du post.

— *Sorties :* OK 200

— *Erreurs :*

1.7.3 delUserPosts

Suppression d'une liste donnée d'identifiants de posts de l'utilisateur.

— *URL :* **DELETE /1/users/ :user_id/posts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **post_ids** *list(string)* : liste d'identifiants des posts à supprimer.

— *Sorties :* OK 200

— *Erreurs :*

1.7.4 getUserLikedPosts

Récupération des identifiants des posts likés par l'utilisateur.

— *URL :* **GET /1/users/ :user_id/likedposts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(string)** : liste des identifiants des posts likés par l'utilisateur.

— *Erreurs :*

1.7.5 addUserLikedPost

Ajout l'identifiant d'un post liké par l'utilisateur.

— *URL :* **POST /1/users/ :user_id/likedposts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **post_id** *string* : identifiant du post liké.

— *Sorties :* OK 200

— *Erreurs :*

1.7.6 delUserLikedPosts

Suppression d'une liste donnée d'identifiants de posts likés par l'utilisateur.

— *URL :* **DELETE /1/users/ :user_id/likedposts**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **post_ids** *list(string)* : liste d'identifiants des posts likés à supprimer.

— *Sorties :* OK 200

— *Erreurs :*

1.8 Notifications

Gestion des notifications.

1.8.1 getUserNotifications

Récupération des notifications.

— *URL :* **GET /1/users/ :user_id/notifications**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict) : id** *string, requis*, **title** *string*, **description** *string*, **type** *string*, **time** *date-time, requis*, **image** *image*, **link** *string, requis*, **status** *string*

— *Erreurs :*

1.8.2 createUserNotification

Création d'une notification.

— *URL :* **POST /1/users/ :user_id/notifications**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **title** *string*, **description** *string*, **type** *string*, **time** *date-time, requis*, **image** *image*, **link** *string, requis*, **status** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.8.3 updateUserNotification

Mise à jour d'une notification.

— *URL :* **PATCH /1/users/ :user_id/notifications**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **id** *string, requis*, **title** *string*, **description** *string*, **type** *string*, **time** *date-time, requis*, **image** *image*, **link** *string, requis*, **status** *string*

— *Sorties :* OK 200

— *Erreurs :*

1.8.4 deleteUserNotification

Suppression d'une notification.

— *URL :* **DELETE /1/users/ :user_id/notifications**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **id** *string, requis* : identifiant de la notification.

— *Sorties :* OK 200

— *Erreurs :*

1.9 Groupes de messages

1.10 Groupes de stories

Gestion des groupes de stories gérés par l'utilisateur.

1.10.1 getUserStoriesGroups

Récupération de la liste des groupes des stories de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/storiesgroups**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **id** *string, requis, unique*, **name** *string, requis*, **description** *string*, **creation_time** *date-time*, **members** *list(dict)* : (**user_id** *string, requis, unique*, **time** *date-time*, **status** *string*)

— *Erreurs :*

1.10.2 getUserStorieGroup

Récupération d'un groupe de stories de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/storiesgroups/ :storie_group_id**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur, **storie_group_id** *string* : identifiant du groupe de storie.

— *Entrées :* -

— *Sorties :* **OK / dict** : **id** *string, requis, unique*, **name** *string, requis*, **description** *string*, **creation_time** *date-time*, **members** *list(dict)* : (**user_id** *string, requis, unique*, **time** *date-time*, **status** *string*)

— *Erreurs :*

1.10.3 createUserStorieGroup

Création d'un groupe de storie.

— *URL :* **POST /1/users/ :user_id/storiesgroups**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **name** *string, requis*, **description** *string*, **creation_time** *date-time*, **members** *list(dict)* : (**user_id** *string, requis, unique*, **time** *date-time*, **status** *string*)

— *Sorties :* OK 200

— *Erreurs :*

1.10.4 updateUserStorieGroup

Mise à jour d'un groupe de storie.

— *URL :* **PATCH /1/users/ :user_id/storiesgroups**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **id** *string, requis, unique*, **name** *string, requis*, **description** *string*, **creation_time** *date-time*, **members** *list(dict)* : (**user_id** *string, requis, unique*, **time** *date-time*, **status** *string*)

— *Sorties :* OK 200

— *Erreurs :*

1.10.5 deleteUserStorieGroup

Suppression d'un groupe de stories.

— *URL :* **DELETE /1/users/ :user_id/storiesgroups/ :storie_group_id**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur, **storie_group_id** *string* : identifiant du groupe de storie.

— *Entrées :* -

— *Sorties :* OK 200

— *Erreurs :*

1.10.6 getUserStorieGroupMembers

Récupère la liste des membres du groupe de stories.

— *URL :* **GET /1/users/ :user_id/storiesgroups/ :storie_group_id/members**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur, **storie_group_id** *string* : identifiant du groupe de storie.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **user_id** *string, requis, unique*, **time** *date-time*, **status** *string*

— *Erreurs :*

1.10.7 addUserStorieGroupMembers

Ajout de nouveaux membres au groupe de stories.

— *URL :* **POST /1/users/ :user_id/storiesgroups/ :storie_group_id/members**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur, **storie_group_id** *string* : identifiant du groupe de storie.

— *Entrées :* **members** *list(string)* : liste des identifiants des membres à rajouter.

— *Sorties :* OK 200

— *Erreurs :*

1.10.8 delUserStorieGroupMembers

Suppression de membres d'un groupe donné de stories.

— *URL :* **DELETE /1/users/ :user_id/storiesgroups/ :storie_group_id/members**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur, **storie_group_id** *string* : identifiant du groupe de storie.

— *Entrées :* **members** *list(string)* : liste des identifiants des membres à rajouter.

— *Sorties :* OK 200

— *Erreurs :*

1.11 Stories

1.12 Recherches

Gestion des recherches de l'utilisateur.

1.12.1 getUserSearchs

Récupération de l'historique de recherche de l'utilisateur.

— *URL :* **GET /1/users/ :user_id/search**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK / list(dict)** : **search** *string, requis*, **time** *date-time, requis*, **deleted** *boolean*

— *Erreurs :*

1.12.2 addUserSearch

Ajout d'une recherche de l'utilisateur dans l'historique.

— *URL :* **POST /1/users/ :user_id/search**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* **search** *string, requis*, **time** *date-time, requis*

— *Sorties :* OK 200

— *Erreurs :*

1.12.3 delUserSearchs

Suppression de l'historique de recherche par l'utilisateur.

— *URL :* **DELETE /1/users/ :user_id/search**

— *Paramètres :* **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* OK 200

— *Erreurs :*

2 Services pour les messages

2.1 Messages

3 Services pour les posts

3.1 Posts

3.1.1 createPost

Création d'un post.

— *URL :* **POST /1/posts**

— *Entrées :* **user_id** *string, requis*, **reply_to_id** *string*, **time** *date-time, requis*, **text** *string*, **location** *string*, **private** *boolean*, **polls** *list(poll)*, **media** *list(media)*, **hashtags** *list(string)*, **mentionned_users_ids**

— *Sorties :* OK 200

— *Erreurs :*

3.1.2 updatePost

Mise à jour d'un post.

— *URL :* **PATCH /1/posts/ :post_id**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* informations à modifier dans le post.

— *Sorties :* OK 200

— *Erreurs :*

3.1.3 deletePost

Suppression d'un post.

— *URL :* **DELETE /1/posts/ :post_id**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* OK 200

— *Erreurs :*

3.1.4 getPost

Récupération d'un post.

— *URL :* **GET /1/posts/ :post_id**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK** / **dict** : les informations sur le post.

— *Erreurs :*

3.1.5 getPosts

Récupération d'une liste de posts.

— *URL :* **GET /1/posts**

— *Entrées :* **posts_ids** *list(string)* : liste des identifiants des posts.

— *Sorties :* **OK** / **list(dict)** : les informations sur les posts.

— *Erreurs :*

3.1.6 getPostsByUser

Récupération d'une liste de posts d'un utilisateur donné.

— *URL :* **GET /1/posts/by/user/ :user_id**

— *Paramètres :* **post_id** *string* : identifiant du post. **user_id** *string* : identifiant de l'utilisateur.

— *Entrées :* -

— *Sorties :* **OK** / **list(dict)** : les informations sur les posts.

— *Erreurs :*

3.1.7 getPostsBy

Récupération d'une liste des posts qui contiennent une liste données d'informations. On peut par exemple récupérer des posts qui contiennent une liste donnée d'hashtags ou d'utilisateurs mentionnés, ou une liste de posts publié durant une période donnée.

— *URL :* **GET /1/posts/by**

— *Entrées :* les champs à faire matcher.

— *Sorties :* **OK** / **list(dict)** : les informations sur les posts.

— *Erreurs :*

3.2 Likes

Gestion des likes de posts.

3.2.1 getPostLikes

Récupération des likes d'un post.

— *URL :* **GET /1/posts/ :post_id/likes**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK** / **list(dict) user_id** *string, requis* **time** *date-time*

— *Erreurs :*

3.2.2 addPostLike

Ajout d'un like à un post.

— *URL :* **POST /1/posts/ :post_id/likes**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* **user_id** *string, requis*, **time** *date-time*

— *Sorties :* OK 200

— *Erreurs :*

3.2.3 delPostLike

Suppression d'un like d'un post.

— *URL :* **DELETE /1/posts/ :post_id/likes**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* **user_id** *string, requis* : identifiant de l'utilisateur duquel supprimer le like.

— *Sorties :* OK 200

— *Erreurs :*

3.3 Reports

Gestion des signalements de posts.

3.3.1 getPostReports

Récupération des signalements d'un post.

— *URL :* **GET /1/posts/ :post_id/reports**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK / list(dict) user_id** *string, requis*, **time** *date-time*, **reason** *string*

— *Erreurs :*

3.3.2 addPostReports

Ajout d'une vue à un post.

— *URL :* **POST /1/posts/ :post_id/reports**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* **user_id** *string, requis*, **time** *date-time*, **reason** *string*

— *Sorties :* OK 200

— *Erreurs :*

3.4 Shares

Gestion des partages de posts.

3.4.1 getPostShares

Récupération des partages d'un post.

— *URL :* **GET /1/posts/ :post_id/shares**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK** / **list(dict) user_id** *string, requis* **time** *date-time*

— *Erreurs :*

3.4.2 addPostShare

Ajout d'un partage à un post.

— *URL :* **POST /1/posts/ :post_id/shares**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* **user_id** *string, requis* **time** *date-time*

— *Sorties :* OK 200

— *Erreurs :*

3.5 Views

Gestion des vues de posts.

3.5.1 getPostViews

Récupération des vues d'un post.

— *URL :* **GET /1/posts/ :post_id/views**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK** / **list(dict) user_id** *string, requis*, **time** *date-time*

— *Erreurs :*

3.5.2 addPostView

Ajout d'une vue à un post.

— *URL :* **POST /1/posts/ :post_id/views**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* **user_id** *string, requis*, **time** *date-time*

— *Sorties :* OK 200

— *Erreurs :*

3.6 Poll

Gestion des sondages dans les posts.

3.6.1 getPostPolls

Récupération des sondages d'un post.

— *URL :* **GET /1/posts/ :post_id/polls**

— *Paramètres :* **post_id** *string* : identifiant du post.

— *Entrées :* -

— *Sorties :* **OK / list(polls)** : liste des sondages.

— *Erreurs :*

3.6.2 getPostPoll

Récupération d'un sondage d'un post.

— *URL :* **GET /1/posts/ :post_id/polls/ :poll_id**

— *Paramètres :* **post_id** *string* : identifiant du post, **poll_id** *string* : identifiant du sondage.

— *Entrées :* -

— *Sorties :* **OK / poll** : sondage.

— *Erreurs :*

3.6.3 addPostPollItemVote

Ajout d'un vote utilisateur à un item d'un sondage d'un post.

— *URL :* **GET /1/posts/ :post_id/polls/ :poll_id**

— *Paramètres :* **post_id** *string* : identifiant du post, **poll_id** *string* : identifiant du sondage.

— *Entrées :* **poll_item_id** *string* : identifiant de l'item, **user_id** *string* : identifiant de l'utilisateur, **time** *date-time*

— *Sorties :* OK 200

— *Erreurs :*