ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Rapport
Projet Web ENSIIE 1A 2018

twitIIE

Chamayou Julien
Mauillon Edwin
XU Jiahui
XU Kevin

22 mai 2018

Sommaire

Préambule

A COMPLETER
L'objectif de notre projet est de réaliser une application Web qui va tout copier de twitter et qui va leur voler tout le marché. Nous devons alors intégrer dans notre application les fonctionnalités suivantes :

— Inscription et Connexion d'utilisateur
— Publication de tweet
— Hashtag et noms d'utilisateur cliquables
— Commentaires de tweet et de commentaires
— Possibilité de like des tweet
— Envoi de messages privés

Le rapport du projet a un but complètement différent de la soutenance. La soutenance nous permet de voir le produit tel que vous avez imaginé son utilisation, le rapport nous permet de comprendre la partie interne du fonctionnement de votre groupe. Il est très important pour nous de bien comprendre quels ont été les enjeux de chaque groupe, quelles ont été les difficultés rencontrées et surtout quels ont été les solutions trouvées pour les contourner.

Le rapport Faire 10 pages maximum pour refléter le monde de l'entreprise où la concision est une qualité Expliquer l'approche mise en place, les problématiques rencontrées (techniques comme méthodes) et les solutions apportées Expliquer la répartition des rôles au sein de l'équipe La problématique à laquelle il répond.

1 Problématique

A COMPLETER

Nous avons essayé de respecter au maximum le modèle MVC durant le développement de l'application.

2 La connexion

L'application web TwitIIE est basée sur le fait même d'avoir un compte par personne, éventuellement un compte pour un groupe ou une association. Il est donc nécessaire de créer cette page de connexion, d'une part pour pouvoir accéder aux fonctionnalités de l'application mais également pour pouvoir authentifier les tweets envoyés. La page de connexion est donc la première page présentée à l'utilisateur. Pour pouvoir accéder à son compte il est nécessaire de remplir un formulaire contenant :

Login : Un login, propre à chacun et unique qui permet l'authentification de chaque personne.

Mot de passe : Un mot de passe qui permet à l'utilisateur de protéger son compte.

Après la soumission de ce formulaire, une recherche dans la base de données est faite afin de trouver s'il existe une personne correspondant au couple (login, mot de passe) donnée par l'utilisateur. Dans le cas contraire, un message d'erreur lui est transmis. Nous restons tout de même vague sur la nature de l'erreur afin de protéger le compte d'une personne mal intentionnée. Si la personne n'a pas de compte, elle est alors invitée à s'inscrire via le lien disponible sur cette même page.

3 Inscription

L'inscription va permettre à tout utilisateur de créer un compte TwitIIE. Il aura alors ensuite accès a toutes les fonctionnalités de l'application. Pour cela, quelques informations lui sont demandé via un formulaire de type POST. Les informations suivantes lui seront demandées :

Login : Un login, qui est unique et qui permettra à l'utilisateur de se connecter. Si le login choisi n'est pas disponible, un message d'erreur lui est envoyé afin q'il puisse le modifier.

Mot de passe : Afin de garantir de la sécurité de son compte, un mot de passe est requis.

Confirmation de mot de passe : Une confirmation du mot de passe précedement entré est demandé afin d'éviter toute faute de frappe.

Nom : Afin de renseigner son compte le nom de famille est demandé.

Prenom : De même, le prénom est demandé.

Date de naissance : Permet également de renseigner le profil de l'utilisateur. On pourrait également faire un test de majorité, mais l'application TwitIIE ne présente pour le moment aucun danger pour les personne mineures, l'accès leur est donc autorisé.

4 Les Abonnements

Pour gérer les abonnements, on a crée la table `amis` composé de deux id, clés étrangères de la table `user` :

personne1 : l'id de l'utilisateur qui s'abonne

personne2 : l'id de l'utilisateur à qui personne1 va s'abonner

C'est à l'aide de cette table que l'on récupère la liste des abonnements de l'utilisateur connecté, utile pour gérer ses messages ainsi que son fil d'actualité. De plus, il est possible d'effectuer une recherche (via le formulaire `recherche @` pour accéder au profil d'un utilisateur inscrit. Un formulaire est alors proposé :

-si l'utilisateur n'est pas abonné à celui recherché, on lui propose d'ajouter celui ci dans sa liste d'abonnements

-sinon, on lui propose de supprimer son abonnement.

5 Les Tweets

5.1 Tweet

Chaque utilisateur a possibilité d'écrire un tweet qui sera par la suite visible en allant sur son profil ou dans le fil d'actualité d'une personne abonnée. La table `tweet` est composée de :

id : l'id du tweet

auteur : l'id de l'auteur du tweet

date_envoie : la date du tweet

contenu : le contenu du tweet

Un formulaire sur la page `accueil` permet à l'utilisateur d'écrire un tweet. De plus, on récupère les tweets des abonnements sur le fil d'actualité à l'aide d'une jointure $n...n$ sur la table `amis` et `tweet`.

5.2 Likes

Il est aussi possible d'aimer un tweet. On crée une table `like` spécifique à cette fonctionnalité, prenant en compte les informations suivantes :

tweet _id : l'id du tweet aimé

user_id : l'id de l'utilisateur qui aime le tweet

On peut ainsi proposer à l'utilisateur d'aimer ou de ne plus aimer un tweet et récupérer le nombre de j'aime associé à un tweet.

5.3 Liens cliquables

Pour chaque tweet, les noms d'utilisateurs @ et les hashtags # sont convertis en liens cliquables. En effet, on peut ainsi directement accéder au profil d'un utilisateur ou encore à l'ensemble des tweets qui contiennent un hashtags.

Pour réaliser cela, lors de l'affichage d'un tweet, le contenu du tweet est analysé mot par mot afin de remplacer tous les noms et hashtags par des liens utilisant la méthode GET. Il suffit en effet juste de trouver les mots commençant par # ou @ puis de trouver l'id correspondant afin de l'inclure dans le lien.

6 Les Commentaires

Chaque utilisateur à la possibilité de commenter un tweet. Il peut aussi commenter un commentaire. Nous aurons alors plusieurs niveau de commentaires. La principale difficulté rencontrée ici était de gérer ces différents niveaux de commentaires.

Voici comment nous avons décidé de créer notre table `commentaire` :

id : l'id du commentaire ;

owner_id : l'id de l'auteur du commentaire

target_id : l'id de l'auteur du tweet ou commentaire commenté

date_envoie : la date du commentaire

contenu : le contenu du commentaire

parent_id : l'id du tweet ou commentaire commenté

parent_type : type du message commenté "tweet" ou "commentaire"

Nous avons aussi créer les classes `Commentaire` et `CommentaireManager` afin de manipuler plus facilement les commentaires.

Pour l'affichage des commentaires avec les différents niveaux, il a fallu une récursion. En effet, pour chaque commentaire, on va exécuter une nouvelle fois la requête pour récupérer tous les commentaires du commentaire courant.

7 Les Hashtags

L'utilisateur a la possibilité d'écrire des hashtags dans ses tweets, c'est-à-dire, des mots commençant par un #.

Voici comment est composée la table `hashtag` :

id : l'id du hashtag ;

mot : le contenu du hashtag

Pour relier chaque hashtag à des tweets, on a utilisée une table `hashtagEtTweet` :

id_hashtag : l'id du hashtag ;

id_tweet : l'id du tweet ;

L'ajout des hashtags dans la table est faite dès qu'un utilisateur envoie un tweet. Le contenu du tweet est analysé afin d'extraire tous les mots commençant par un #. Ensuite, on regarde s'il existe déjà un hashtag similaire dans la table. S'il en existe un, le tweet est ajouté dans la table `hashtagEtTweet`, sinon on ajoute le hashtag dans la table `hashtag` dans un premier temps.

On a aussi la possibilité de rechercher tous les tweets contant un certain hashtag dans la barre de recherche #. Il y a aussi des suggestions de hashtags lors de la recherche qu'on a réalisé grâce à AJAX.

8 Les Messages Privés

Nous avons décider d'ajouter une fonctionnalité d'envoi de messages privés entre chaque utilisateur. Cette fonctionnalité est directement accessible depuis la barre de navigation. L'utilisateur peut alors choisir un destinataire parmi ses amis afin de lui envoyer un message privé.

Pour réaliser cela, nous avons dû créer une nouvelle table dans la base de données, voici ses attributs :

`message(id, emetteur, recepteur, date_envoie, contenu)`

On a aussi décidé de créer une classe `Message` et une classe `MessageManager` afin de faciliter la manipulation des messages.

Pour cette partie, on a principalement utilisé de l'AJAX pour envoyer des messages privés et pour charger les messages reçus automatiquement sans avoir besoin de réactualiser la page. Ces deux éléments sont les principales difficultés rencontrées pour réaliser cette fonctionnalité.

9 Répartition des tâches

10 Conclusion

Malheureusement, nous n'avons pas réalisé la fonctionnalité de notifications par manque de temps.