

# Modélisation - TD1

## 1 TD 1 : Cahier des charges et modélisation

Tout au long des TD/TP, nous allons développer le site associatif Organiz-asso. Dans ce TD, on s'intéresse à l'architecture du site web développé (côté serveur), qui inclut :

- les différents services associés ;
- la structure de la base de données ;
- les choix technologiques que nous ferons.

### 1.1 Cahier des charges

Notre site Organiz'asso permet à des membres d'une association d'échanger des messages sur des forums. L'association est pilotée par un conseil d'administration, qui sont des membres élus appelés administrateurs. Il propose deux forums :

- le forum ouvert, que chaque membre inscrit peut consulter et sur lequel il peut poster des messages ;
- le forum fermé, réservé aux membres du conseil d'administration.

Hors connexion, un utilisateur n'a que la possibilité de créer un compte. Son inscription doit être validée par un administrateur pour lui attribuer le statut de membre.

Lorsqu'un membre se connecte, cela permet d'ouvrir une page principale qui contient le forum ouvert.

Une personne arrivant sur le site peut :

- se créer un compte et ainsi demander le statut de membre
- se connecter

Une fois connecté, et sa demande d'inscription validée par un administrateur, un membre peut :

- créer des messages :
  - soit en réponse à un message précédemment posté
  - soit pour démarrer une nouvelle discussion
- visualiser son profil contenant au moins la liste des messages qu'il a publiés. À partir de son profil, il peut supprimer ses propres messages.
- visualiser le profil d'autres membres.
- rechercher des messages en précisant des mots-clés, un intervalle de temps de publication ou leur auteur.

Les administrateurs sont des utilisateurs bénéficiant de droits supplémentaires. Un administrateur :

- a accès au forum fermé
- peut donner ou retirer le statut administrateur à un autre utilisateur, sauf à lui-même
- revoit les inscriptions sur le site, et valide ou non le statut de membre d'un utilisateur inscrit.

À la fin de son activité, l'utilisateur a la possibilité de se déconnecter.

### 1.2 Modélisation

1. Définir le schéma d'interactions entre un utilisateur et les différents composants du site. Les composants peuvent aussi interagir entre eux. On considérera comme composants :

- l'utilisateur
- Le client du réseau social (navigateur)
- Le serveur du site
- La base de données

image interactionUML.pdf  
(leur montrer le début, les laisser réfléchir, puis corriger la suite).

2. Quels sont les services à implémenter ?

- User
  - CreateUser
  - SetAdmin
  - SetMember
  - GetProfile
  - Login
  - Logout
- SetAdmin et SetMember peuvent être remplacés par un unique SetStatus en en faisant varier les paramètres
- Message
  - CreateMessage
  - DeleteMessage
  - Search
  - GetMessage

3. Définir la structure de la base de données.

- document User : userID, identifier, password, admin, member
- document Message : messageID, userID, content, title, date, answeredMessageID, forumID

4. L'implémentation d'un service implique la création du service lui-même et parfois des interactions avec la base de données. Définir les algorithmes et fonctions d'accès à la base de données nécessaires pour créer le service « DeleteMessage ».

**service** DeleteMessage(idMess,idUser)

- Vérifier si le message appartient bien à l'utilisateur
  - Vérifier si l'utilisateur existe dans la BD : sinon, erreur
  - Vérifier si le message existe dans la BD, sinon erreur
  - Vérifier si le message appartient à l'utilisateur, sinon erreur
- Supprimer le message de la BD
  - Supprimer le message dans la BD, erreur possible
  - Renvoyer "ok" ou erreur

Bien souligner les différences de nature entre les erreurs liées à la base de données (BdD inaccessible) ou au système (l'utilisateur ou le message n'existent pas en base de données, le message n'appartient pas à l'utilisateur, la requête est mal formulée...)

5. Identifier à quel moment le service « GetMessage » peut-être utilisé. Quels sont les arguments nécessaires au service associé et quel est le contenu algorithmique de ce service ?

ListMessage est utilisé :

- dans la page principale. ListMessage()
- dans le profil d'un utilisateur. ListMessage(userID)
- pour la recherche. ListMessage(query)

On peut coder un seul service qui sera réutilisé sur tout le site : ListMessage(userID, query, contact) et on ajuste le résultat en fonction de la nullité ou du contenu des champs.

**service** ListMessage(userID, query,contact)

- Si userID et query vides : ListMessageBD()
- Si userID non null et (query) null : ListMessageBD(userID)
- Si userID null et (query) non null : ListMessageBD(query). query peut contenir des mots-clefs, des dates, *etc.*
- Pour chaque ListMessageBD, peuvent être associés des erreurs suivant si userExist() ou errorBD(). Retour : Liste JSON de messages. Remarque : si une requête (query) ne renvoie aucun résultat, la liste de messages est vide mais ce n'est pas une erreur.

6. Quelle est l'architecture matérielle/logicielle du projet à réaliser

Slide 7 du cours - Si possible, redessinez au tableau + positionner les différentes technologies :

- HTML/CSS/JavaScript et React (coté client)
- JSON/API REST pour la discussion client/serveur
- NodeJS sur le serveur
- MongoDB pour les données

7. Dessiner les différentes interfaces que vous imaginez pour le site. Faire correspondre les différentes interactions possibles avec les services web définis dans la question 2.

Ils font les dessins qu'ils veulent. Mais l'important est qu'ils comprennent qu'une page est une compilation de modules (des composants...), qui appellent des services, qui eux-mêmes récupèrent des messages en JSON renvoyés par le serveur (et non du code HTML). Le client ensuite processe le JSON et fait les différents traitements jusqu'à l'affichage en HTML/CSS (prépare au TME où ils vont faire du postman pour interroger les services).

Par exemple pour la page de connexion, on appelle le service createUser.

Dans la page principale, la barre de recherche c'est ListMessages, l'affichage de leurs messages, c'est aussi ListMessages mais avec des paramètres différents.