Projet conception logicielle

> Équipe :
>
> - Alexandre FROEHLICH
> - Guillaume LEINEN
> - Jean-Noël CLINK
> - Ayrwan GUILLERMO
> - Erwan AUBRY

Introduction au projet

Nous représentons *EnStar.com*, une entreprise de conception d'outils de communication par texte.

Nous allons devoir concevoir une app desktop : *EnstarDesktop* qui permet à chaque utilisateur de communiquer avec les autres sur le réseau. Chaque utilisateur est identifié.

Composition des équipes

Team Backend

- Alexandre FROEHLICH
- Erwan AUBRY

Team Frontend

- Guillaume LEINEN
- Jean-Noël CLINK
- Ayrwan GUILLERMO

Points à implémenter

FrontEnd Utilisateur

- [ ] Identification (login, password)
- [ ] DM (messages privés entre utilisateurs)
- [ ] Groupes de discussion (plusieurs utilisateurs)
    - [ ] Création de groupes
    - [ ] Messagerie de groupe
- [ ] Rechercher un utilisateur
- [ ] Vérifier si un utilisateur est en ligne

FrontEnd Administrateur

- [ ] Identification identique à l'utilisateur
- [ ] Gestion des utilisateurs
    - [ ] Création des utilisateurs
    - [ ] Importation depuis un fichier (bdd, csv, ...)
- [ ] Avoir les utilisateurs en cours d'échange (?)

Backend

- [ ] Serveur TCP concurrent
- [ ] Motif sujet-observateur
- [ ] Motif délégation
- [ ] Motif stratégie
- [ ] Motif composant-composite

Bonus

- [ ] Discussions par sujet
- [ ] Historique chiffré (AES, XOR, ...)
- [ ] Historique des conversations (avec un cache local)
- [ ] Recherche dans les messages (full text search)

Méthode

L'interface sera développée avec JavaFX. Les tests unitaires seront réalisés avec JUnit. La documentation sera générée avec JavaDoc. Le diagramme de classe sera réalisé avec draw.io.

Pour respecter les contraintes du projet, une release sera effectuée sur Github toutes les semaines, l'archive ainsi crée sera envoyée sur moodle.

L'ID utilisateur sera un UUIDv4. Les mdp seront hashés avec BCrypt ou SHA256.

A rendre chaque semaine

- [ ] User-stories : sprint en cours avec revue courte
- [ ] Photo des cartes CRC
- [ ] Résultat des tests unitaires : fichier XML exporté par JUnit
- [ ] Un exécutable de l'application
- [ ] Diagramme de classe
- [ ] Documentation

> Penser à faire un index avec la localisation de tous les documents demandés

Documentation : commandes

Les commandes et les réponses seront formatées en JSON.

Client -> Serveur

Une fois loggé, l'id de l'utilisateur ne transite plus, il sera associé au socket crée entre le client et le serveur.

Commande	Arguments	Description
connect	*null*	Message envoyé au serveur pour se connecter
disconnect	*null*	Message envoyé au serveur pour fermer une connexion
login	username, mdp_hashed	Envoyé au serveur pour authentifier un utilisateur
createMP	dest_user_id	Créer un fil de discussion entre deux personnes
createGroup	group_name, array of user_id	Créer un fil de discussion entre plusieurs personnes (plus que 2)
sendMP	dest_user_id, message	Envoie un message dans un fil de discussion
sendGroup	group_id, message	Envoie un message dans un groupe
getMPMsg	dest_user_id, index=0, limit=5	Récupère les messages d'un fil de discussion
getGroupMsg	group_id, index=0, limit=5	Récupère les messages d'un groupe
getUserByID	user_id	Récupère les informations d'un utilisateur à partir de son identifiant unique
getUserByName	user_name	Récupère les informations d'un utilisateur à partir de son nom d'utilisateur (retourne le premier résultat)
createUser	username, mdp_hashed	Crée un utilisateur à partir d'un nom et d'un mot de passe hashé
removeUserFromGroup	group_id, user_id	Retire un utilisateur d'un groupe
blockUser	user_id	Ajoute un utilisateur à une blocklist

```
{
    "command": "getUserByID",
    "args": ["uuid"]
}
```

Réponse : Serveur -> Client

Les réponses seront au format :

```
commandeExecutee status message
```

Codes status :

- OK : succès
- DENIED : réfusé (permissions, user deja existant, blacklisté...)
- NOT_FOUND : user/group id non trouvé
- SERVER_ERROR : (try catch) + msg

```
{
    "command": "getUserByID",
    "status": "OK",
    "data": {
        "name": "alexandre",
        "password": "sqkfjlsdm",
        "role": "ROLE_USER"
    }
}
```

Structures

User

- Name : string
- MDP : string
- ROLE : string (ROLE_USER, ROLE_ADMIN)
- Blacklist : string[] –> "userid1, userid2

Message

- Date : date
- Sender_id : uuid
- Thread_id : fil de discussion (MP ou group)
- Content : string
- (isRead) : "none", "sent", "read"