**Analyse projet IFC**

**I. Tâches à réaliser et fonctions associées :**

1. Affichage du menu au démarrage

Cette fonction affiche le menu principal dans lequel l’utilisateur doit entrer le numéro de l’action qu’il souhaite faire. Les trois choix d’actions sont les suivants :  
 1. Se connecter  
 2. Créer un compte  
 3. Quitter

Prototype de la fonction : VIDE FONCTION menu\_démarrer (VIDE)

2. Création d’un compte

Cette fonction va permettre à l’utilisateur de créer un compte. Il devra saisir un nom d’utilisateur, qui déterminera son adresse Zimbra qui sera sous la forme nom\_d’utilisateur@utbm.fr, ainsi qu’un mot de passe. Nous contrôlerons à l’aide de deux fonctions intermédiaire que le nom d’utilisateur ne comporte que de lettres, puis qu’il est unique. Nous vérifierons également que le mot de passe proposé comporte au moins 8 caractères.

La fonction vérifiant la présence de lettres uniquement renvoie 0 s’il n’y a que des lettres et n’importe quel autre nombre sinon.  
La fonction vérifiant l’unicité du nom d’utilisateur renvoie 1 si le nom existe et n’importe quel autre nombre sinon.

Une fois les données correctement saisies, elles seront stockées dans un fichier contenant tous les comptes existant, leur mot de passe, ainsi que les informations sur leur messagerie (voir page 6 pour plus de détails sur ce fichier).

Prototype de la fonction : VIDE FONCTION créer\_compte (VIDE)

Prototype des fonctions intermédiaires :

ENTIER FONCTION nom\_valide (CHAÎNE DE 20 CARACTÈRES nom)  
ENTIER FONCTION existe\_nom (CHAÎNE DE 20 CARACTÈRES nom)

3. Connexion à un compte

Cette fonction va demander à l’utilisateur de saisir son nom d’utilisateur, puis son mot de passe et retourne une structure nommée « UTILISATEUR » contenant toutes les informations sur l’utilisateur (voir page 5 pour plus de détails sur cette structure). Nous contrôlerons que le nom d’utilisateur existe en utilisant le même fonction intermédiaire que dans la partie 1., puis que le mot de passe correspond à celui du compte sélectionné.

Prototype de la fonction : UTILISATEUR FONCTION connexion (VIDE)

Prototype des fonction intermédiaires :

ENTIER FONCTION existe\_nom (CHAÎNE DE 20 CARACTÈRES nom)  
ENTIER FONCTION corresp\_mdp (CHAÎNE DE 20 CARACTÈRES mdp, CHAÎNE DE 20 CARACTÈRES nom)

4. Affichage du menu des actions

Cette fonction affiche les actions que l’utilisateur peut réaliser sur sa messagerie après connexion à un compte et acquiert son choix, qui sera retourné. Trois choix seront possibles :  
 1. Consulter vos messages  
 2. Envoyer un message  
 3. Se déconnecter

Prototype de la fonction : ENTIER FONCTION menu\_messagerie (UTILISATEUR utilisateur\*)

5. Consultation des messages

Cette fonction affiche les différents dossiers d’un utilisateur donné et acquiert le choix de l’utilisateur sur le dossier à ouvrir. Il existera des dossiers par défaut, et des dossiers que l’utilisateur a lui-même créé (5 maximum). Nous obtiendrons les dossiers de l’utilisateur et leur nombre à l’aide de la structure utilisateur. Par ailleurs, nous n’afficherons pas l’option « Créer un dossier » si l’utilisateur possède 5 dossiers personnalisés. De même, nous n’afficherons pas l’option « Supprimer un dossier » s’il n’existe aucun dossier personnalisé. Les choix seront les suivants :  
 1. Boîte de réception  
 2. Messages envoyés  
 3. Messages effacés  
 4. Dossier personnalisé 1  
 5. Dossier personnalisé 2  
 …  
 n + 3. Dossier personnalisé n  
 (n + 4. Créer un dossier)  
 (n + 5. Supprimer un dossier)  
 n + 6. Retour

Prototype de la fonction : ENTIER FONCTION menu\_dossier (UTILISATEUR utilisateur\*)

6. Affichage des messages dans un dossier

Cette fonction affiche le titre, la date et l’expéditeur de 10 messages qui sont sous la forme d’une structure (voir page 6 pour plus de détails sur cette structure), pour une page, un dossier (structure), et un utilisateur donné, et demande à l’utilisateur le numéro du message à afficher intégralement. Les messages non lus seront marqués d’un « \*\*\* ». D’autres choix permettent également de passer à la page suivante ou précédente (si possible), et d’entrer le numéro d’une page à afficher. La fonction retourne le choix que l’utilisateur a fait. Nous contrôlerons la possibilité de passer à une certaine page pour un utilisateur donné à l’aide d’une fonction intermédiaire retournant 0 si c’est impossible et 1 sinon, et afficherons les choix « Page suivante » et « Page précédente » en fonction du résultat de la fonction. Nous convertirons également la date, exprimé en nombre de secondes depuis l’epoch UNIX, en une date sous la forme jour/mois/année heure:minute à l’aide d’une fonction intermédiaire. Les choix se présentent comme ceci :  
 1. \*\*\* Expéditeur 1 : Titre du message 1 – Date  
 …  
 10. Expéditeur 10 : Titre du message 10 – Date  
 (11. Page suivante)  
 (12. Page précédente)  
 13. Aller à la page  
 14. Retour

Prototype de la fonction : ENTIER FONCTION affiche\_messages (UTILISATEUR utilisateur\*, DOSSIER dossier\*, ENTIER page)

Prototype des fonctions intermédiaire :

ENTIER FONCTION existe\_page (DOSSIER dossier\*, ENTIER page)  
VIDE FONCTION convertir\_date (ENTIER date\_unix, CHAÎNE DE 10 CARACTERES date)

7. Affichage d’un message

Cette fonction affiche le message que l’utilisateur a choisi auparavant. La fonction affichera l’expéditeur, le titre, la date, et le corps du message. Le message sera automatiquement marqué comme lu lorsque cette fonction est appelée. L’utilisateur dispose ensuite de plusieurs choix sur l’action à réaliser sur ce message, qui sera retournée. Les différents choix sont :  
 1. Répondre  
 2. Supprimer  
 3. Marquer comme non-lu  
 4. Déplacer vers  
 5. Bloquer cet expéditeur  
 6. Retour

Prototype de la fonction : ENTIER FONCTION affiche\_message (UTILISATEUR utilisateur\*, MESSAGE message\*)

8. Répondre à un message

Cette fonction permet au destinataire d’un message de répondre au message de l’expéditeur en saisissant seulement le corps du message. L’objet du message sera automatiquement noté « Re : » suivit de l’objet du message de l’expéditeur. Nous contrôlerons la longueur du message afin qu’elle n’excède pas 2000 caractères. Une fois le message envoyé, il sera également placé dans le dossier « Messages envoyés ».

Prototype de la fonction : VIDE FONCTION repondre\_message (MESSAGE message\*)

9. Supprimer un message

Cette fonction déplace le message voulu dans le dossier « Message effacés », ou le supprime définitivement s’il est déjà dans ce dossier.

Prototype de la fonction : VIDE FONCTION supprimer\_message (UTILISATEUR utilisateur\*, MESSAGE message)

10. Marquer comme non-lu

Cette fonction marque un message comme non-lu, puis affiche la liste des dossiers.

Prototype de la fonction : VIDE FONCTION nonlu\_message (UTILISATEUR utilisateur\*, MESSAGE message)

11. Déplacer un message vers un dossier

Cette fonction affiche la liste des dossiers (par défauts et personnalisés, similairement à celle de la partie 5.) et demande à l’utilisateur d’entrer le numéro du dossier dans lequel il souhaite déplacer le dossier, et réalise l’action.

Prototype de la fonction : VIDE FONCTION déplacer\_message (UTILISATEUR utilisateur\*, MESSAGE message)

12. Création d’un dossier

Cette fonction demande à l’utilisateur le nom du nouveau dossier qu’il souhaite créer puis l’ajoute à la liste des dossiers. Nous contrôlerons la longueur du nom du dossier afin qu’elle n’excède pas 20 caractères, ainsi que l’unicité du nom du dossier. La fonction affiche ensuite la liste des dossiers (partie 5.)

Prototype de la fonction : VIDE FONCTION créer\_dossier (UTILISATEUR utilisateur\*)

13. Suppression d’un dossier

Cette fonction affiche la liste des dossiers personnalisés et demande à l’utilisateur le numéro du dossier qu’il souhaite supprimer.

Prototype de la fonction : VIDE FONCTION supprimer\_dossier (UTILISATEUR utilisateur\*)

14. Blocage d’un expéditeur

Cette fonction permet à un utilisateur de bloquer un expéditeur afin de ne plus recevoir de mails de sa part.

Prototype de la fonction : VIDE FONCTION bloquer\_exp (UTILISATEUR utilisateur\*, UTILISATEUR expéditeur\*)

15. Composition d’un message

Cette fonction permet à un utilisateur de composer un message. Elle demande à l’utilisateur de saisir l’adresse mail du destinataire, puis le titre du message, et enfin, le corps du message qu’il souhaite envoyer par le biais de fonctions intermédiaires. Nous contrôlerons dans ces fonctions l’existence de l’adresse mail en utilisant la fonction intermédiaire de la partie 1., puis que le destinataire n’a pas bloqué l’expéditeur, ensuite que le titre du message n’excède pas 30 caractères, et enfin, que le corps du message n’excède pas 500 caractères. La fonction retourne une structure message contenant les informations sur le message à envoyer.

Prototype de la fonction : MESSAGE FONCTION composer\_message (UTILISATEUR utilisateur\*)

Prototype des fonctions intermédiaires :

ENTIER FONCTION existe\_nom (CHAÎNE DE 20 CARACTÈRES nom)  
ENTIER FONCTION est\_bloqué (UTILISATEUR expéditeur, UTILISATEUR destinataire)  
VIDE FONCTION get\_adresse (CHAÎNE DE 28 CARACTÈRES adresse)  
VIDE FONCTION get\_titre (CHAÎNE DE 30 CARACTÈRES titre)  
VIDE FONCTION get\_corps (CHAÎNE DE 500 CARACTÈRES corps)

16. Envoie d’un message

Cette fonction reçoit une structure message à envoyer et effectue l’opération. Le message envoyé sera automatiquement placé dans le dossier « Messages envoyés » de l’utilisateur expédiant le message.

Prototype de la fonction : VIDE FONCTION envoyer\_message (MESSAGE message\*)

17. Obtention d’informations sur un utilisateur

Cette fonction retourne un utilisateur (structure), à partir d’un nom d’utilisateur donné. Les informations sur celui-ci sont collectées grâce au fichier énoncé dans la partie 1.

Prototype de la fonction : UTILISATEUR FONCTION get\_utilisateur (CHAÎNE DE 20 CARACTÈRES nom)

18. Mis à jour du fichier

Cette fonction met à jour les information contenues dans le fichier énoncé en partie 1. pour un utilisateur donné à partir de sa structure. Il sera appelé lorsque toute modification a été apportée aux données d’un utilisateur, comme la création d’un dossier, l’envoie d’un message, la réception d’un message etc…

Prototype de la fonction : VIDE FONCTION maj\_fichier (UTILISATEUR utilisateur\*)

**II. Structures, tableaux et fichiers :**

1. Structure UTILISATEUR

Cette structure est utile dans la très grande partie des fonctions, et est créée dans la fonction get\_utilisateur à partir du nom d’utilisateur. Elle contient le nom d’utilisateur, le nombre de dossiers que possède l’utilisateur et un tableau de ces dossiers, et le nombre d’utilisateurs bloqués et un tableau de ces utilisateurs.

Prototype de la structure : STRUCTURE UTILISATEUR :  
 CHAÎNE DE 20 CARACTÈRES : nom  
 ENTIER : num\_dossiers  
 TABLEAU DE 13 DOSSIERS : dossiers  
 ENTIER : num\_bloqués  
 TABLEAU DE 100 UTILISATEURS : util\_bloqués

2. Structure DOSSIER

Cette structure est utilisée afin d’afficher la liste des messages contenus dans un dossier. Elle contient le nom du dossier, et le nombre de messages dans ce dossier et un tableau de ces messages.

Prototype de la structure : STRUCTURE DOSSIER :  
 CHAÎNE DE 20 CARACTÈRES : nom  
 ENTIER : num\_messages  
 TABLEAU DE 1000 MESSAGES : messages

3. Structure MESSAGE

Cette structure est utilisée pour afficher toutes les informations sur un message. Elle contient notamment deux entier, dont « dates » signifiant le nombre de secondes depuis l’epoch UNIX, et « lu » valant 0 si le message est non-lu, et 1 sinon. La structure dossier signifie le dossier dans lequel le message se trouve.

Prototype de la structure : STRUCTURE MESSAGE :  
 UTILISATEUR : expéditeur, destinataire  
 CHAÎNE DE 50 CARACTÈRES : titre  
 CHAÎNE DE 2000 CARACTÈRES : corps  
 ENTIER : date, lu  
 DOSSIER : dossier

4. Tableau des dossiers

Ce tableau est utilisé lors de l’affichage de la liste des dossiers. Le tableau contient 13 dossiers (structures) (3 pour les dossiers par défaut, 10 pour les dossiers personnalisés), qui permettent par la suite d’obtenir les messages se trouvant dans ce dossier.

Exemple de tableau : dossiers[13] = {  
 dossier\_boite\_reception,  
 dossier\_messages\_envoyés,  
 dossier\_messages\_supprimés,  
 dossier\_personnalisé\_1,  
 dossier\_personnalisé\_2 }

5. Tableau des messages dans un dossier

Ce tableau est utilisé lors de l’affichage de l’aperçu d’un message dans un dossier, et de son contenu. Le tableau contient au plus 1000 messages (structures).

Exemple de tableau : messages[1000] = {  
 message1,  
 message2 }

6. Tableau des utilisateurs bloqués

Ce tableau est utilisé pour identifier les noms bloqués par un utilisateur. Le tableau contient au plus 100 utilisateurs (structures).

Exemple de tableau : util\_bloqués [100] = {  
 utilisateur1,  
 utilisateur2 }

7. Fichier contenant toutes les informations

Cet unique fichier sert de base données pour stocker toutes les informations sur touts les utilisateurs. Nous utiliserons la fonction énoncée en partie I.3. et I.17. pour passer des informations du fichiers à une structure et inversement, et utiliserons la fonction de la partie I.18 pour sauvegarder les informations de la structure dans le fichier. Nous prévoyons d’écrire dans ce fichier selon la structure présentée à la page suivante.

Exemple de fichier :

0 : {  
 Nom : aadam01,  
 Mot de passe : exemple\_de\_mot\_de\_passe,  
 Dossiers : {  
 Boîte de réception : {  
 0 : {  
 Expéditeur : testeur,  
 Date : 1556438305,  
 Titre : test,  
 Corps : Ce message est un test,  
 Lu : 1,  
 },  
 1 : {  
 Expéditeur : tblanchet01,  
 Date : 1556438574,  
 Titre : Analyse projet IFC,  
 Corps : Salut, j’ai mis à jour l’analyse projet IFC, a +,  
 Lu : 0,  
 },  
 },  
 Messages envoyés : {  
 0 : {  
 Destinataire : tblanchet01,  
 Date : 155643743,  
 Titre : Projet IFC,  
 Corps : Salut, je viens de mettre en place le projet sur GitHub, à plus !  
 Lu : 1,  
 },  
 },  
 Messages supprimés: {},  
 Important : {  
 0 : {  
 Destinataire : fnika01,  
 Date : 155642871,  
 Titre : Date du final d’IFC,  
 Corps : Bonjour, je vous informe que le final d’IFC aura lieu le 28/06/2019 à 14h en salle P307. Bonnes révisions.  
 Lu : 1,  
 },  
 },  
 },  
 Utilisateurs bloqués : {  
 utilisateur1,  
 utilisateur2,  
 },  
}

**III. Répartition des tâches :**

Le tableau ci-dessous présente les fonctions que chaque membre est chargé d’écrire :

|  |  |  |
| --- | --- | --- |
| Membre | Ahmet | Théo |
| Fonctions | menu\_démarrer | menu\_messagerie |
| créer\_compte | menu\_dossier |
| nom\_valide | affiche\_messages |
| existe\_nom | existe\_page |
| connexion | convertir\_date |
| corresp\_mdp | affiche\_message |
| supprimer\_message | répondre\_message |
| déplacer\_message | nonlu\_message(j’ai fait sans) |
| supprimer\_dossier | créer\_dossier |
| get\_adresse | bloquer\_exp |
| get\_corps | composer\_message |
| get\_utilisateur | est\_bloqué |
| maj\_fichier | get\_titre |
|  | envoyer\_message |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**IV. Algorithme d’une fonction importante du projet :**

Algorithme de la fonction « composer\_message » :

STRUCTURE UTILISATEUR :  
 CHAÎNE DE 20 CARACTÈRES : nom  
 ENTIER : num\_dossiers  
 TABLEAU DE 13 DOSSIERS : dossiers  
 ENTIER : num\_bloqués  
 TABLEAU DE 100 UTILISATEURS : util\_bloqués

STRUCTURE MESSAGE :  
 UTILISATEUR : expediteur, destinataire  
 CHAÎNE DE 50 CARACTÈRES : titre  
 CHAÎNE DE 2000 CARACTÈRES : corps  
 ENTIER : date, lu  
 DOSSIER : dossier

FONCTIONS : get\_adresse, est\_bloqué, get\_titre, get\_corps, strlen, strcpy, get\_utilisateur, time

MESSAGE FONCTION composer\_message (UTILISATEUR utilisateur)  
VARIABLES  
 CHAÎNE DE 28 CARACTÈRES : mail\_dest  
 CHAÎNE DE 20 CARACTÈRES : nom\_dest  
 CHAÎNE DE 50 CARACTÈRES : titre  
 CHAÎNE DE 2000 CARACTÈRES : corps   
 MESSAGE : message  
 UTILISATEUR : destinataire  
DÉBUT  
 get\_adresse (mail\_dest)  
 strncpy(nom\_dest, mail\_dest, strlen(mail\_dest) – 8)  
 destinataire 🡨 get\_utilisateur (nom\_dest)  
   
 SI (est\_bloque (utilisateur, destinataire) = 1)  
 ALORS  
 ECRIRE (« Erreur: vous ne pouvez pas envoyer de messages à cette adresse »)  
 SINON  
 get\_titre (titre)  
 get\_corps (corps)  
   
 message.expediteur 🡨 utilisateur  
 message.destinataire 🡨 destinataire  
 strcpy (message.titre, titre)  
 strcpy (message.corps, corps)  
 message.date 🡨 time (0)  
 FINSI  
  
 RENVOYER message  
FIN