Projet C Version Finale : Service d'annuaires partagés

But du Projet : Réaliser une application répartie sur le modèle client/serveur gérant des annuaires partagés.

BRONGNIART Clément - ISSALIS Dorian
NICOLAS Vincent – PONS Jean-louis

2020

Sommaire

Introduction : 3

Etape 1 : Spécification et conception protocolaire de l'application répartie 4

PDU requête : 4

Exemples requêtes : 4

PDU réponse : 4

Exemples réponses : 4

Fichiers : 5

Fichier Utilisateurs : 5

Fichier annuaire : 5

Les codes utilisés et leur description : 6

Menus de l'application : 8

Etape 2 : Conception des algorithmes du client et du serveur 11

1er raffinage : (main global) 11

Client : 11

Serveur : 13

2ème raffinage : 14

Client : 14

Serveur : 18

Attribution des droits : 27

Etape 3 : Codage du client et du serveur (1ère partie) 31

Exécution de l'application : 31

Fonctions non implémentées 36

Répartition des tâches 36

Planning 37

Introduction :

Notre mission est de mettre en place un service d'annuaires partagés sur le modèle client/serveur pouvant fonctionner en réseau. Notre serveur de l'application stockera dans un ou plusieurs fichiers texte ou binaire les données de ces annuaires partagés. Il aura par conséquent à gérer plusieurs annuaires.

Pour chaque contact dans l'annuaire, on stockera plusieurs « fichierInfos » :

- *Nom
- *Prénom
- Num. tél portable
- Adresse postale
- *Adresse mail
- Date de naissance
- Remarques concernant le contact (Description)

* champs obligatoires

En ce qui concerne les fonctionnalités à implémenter, l'administrateur doit gérer les comptes utilisateurs (création, modification et suppression). Un utilisateur ayant un compte peut créer et gérer un seul annuaire. Cet annuaire sera accessible en lecture seule à tout utilisateur autorisé. Un utilisateur peut consulter un annuaire si le propriétaire lui a donné les permissions nécessaires. Enfin, le client possèdera un menu texte permettant de choisir les actions à réaliser.

Etape 1 : Spécification et conception protocolaire de l'application répartie

PDU requête :

1. Code (type requête)
2. Champ username
3. Champ Password
4. Champ data

Nos requêtes sont constituées de 4 champs code, user, password et data. Le code correspond à la requête demandée par le client. Les champs username et password correspondent aux informations du client connecté. Le champ data peut-être de 4 types différents message, utilisateur, contact et annuaire.

Exemples requêtes :

- 0 test test123 AuthReq
- 40 test test123 Robert

PDU réponse :

1. Code (code réponse ou erreur)
2. Champ data

Nos réponses sont constituées de 2 champs code et data qui correspondent à une réponse à la requête demandée.

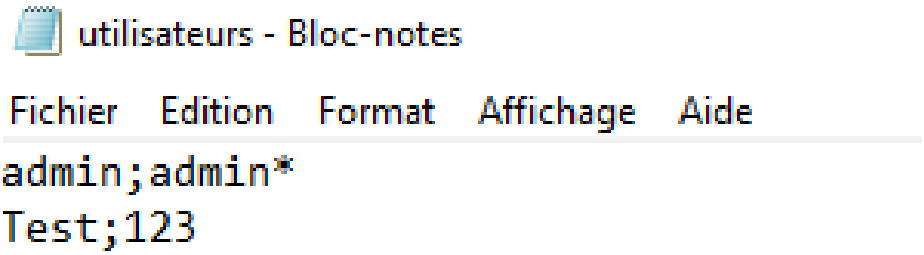
Exemples réponses :

- 50 AuthOK
- 60 ReqOK

Fichiers :

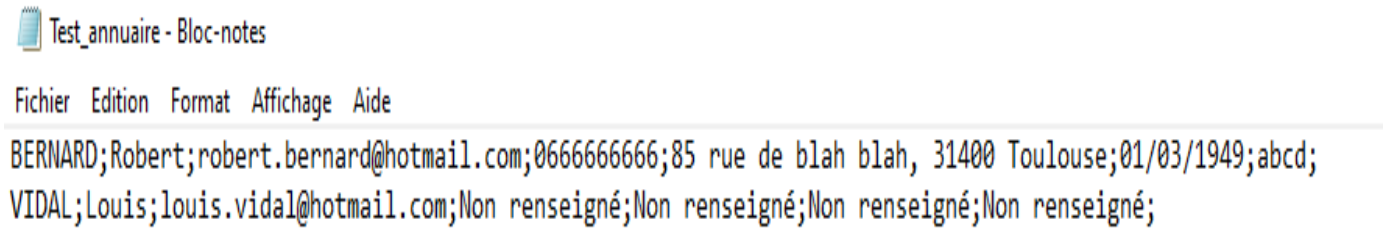
Nous avons décidé d'avoir 3 fichiers pour nos fonctions qui sont le fichier « utilisateurs », « annuaire » et « droits de lecture ». Nous n'avons pas eu le temps d'implémenter les fonctions d'attributions des droits par conséquent nous n'avons pas un fichier bien formaté pour ce menu.

Fichier Utilisateurs :



Pour le fichier utilisateurs, nous avons créé un seul fichier qui contient tous les login et password des clients de notre application. Chaque ligne correspond à un utilisateur et à son mot de passe. Ces 2 éléments sont séparés par un « ; » pour faciliter la lecture du fichier pour nos autres fonctions.

Fichier annuaire :



Pour le fichier annuaire, nous avons créé un fichier par utilisateur. Ce fichier contient l'ensemble des contacts de l'utilisateur. Chaque ligne correspond à un contact de l'annuaire de l'utilisateur. Dans cette ligne, on retrouve le nom, le prénom, l'adresse mail ainsi que les informations supplémentaires du contact. En ce qui concerne les informations supplémentaires, si elles ne sont pas remplies lors de la création du contact alors elles seront en « Non renseigné » dans l'annuaire. Pour faciliter la lecture de l'annuaire, nous avons séparé les différents champs par des « ; ».

Les codes utilisés et leur description :

Codes requêtes :

- 1x = requêtes liées à la création
- 2x= requêtes liées à la mise à jour
- 3x= requêtes liées à la suppression
- 4x= requêtes liées à l'affichage

Code	Nom de la requête	Description
0	Bind	Connexion au serveur
1	Unbind	Déconnexion du serveur
10	CreateContact	Création d'un contact non existant dans l'annuaire de l'utilisateur
11	CreateContactsBook	Création d'un annuaire pour un utilisateur s'il n'en possède pas.
12	**CreateUser**	**Création d'un nouvel utilisateur et de son annuaire**
13	CreateRights	Création de droit de lecture pour un utilisateur
20	UpdateContact	Modification d'un contact existant dans l'annuaire de l'utilisateur
21	**UpdateUser**	**Modification des paramètres d'un utilisateur**
30	DeleteContact	Suppression d'un contact existant dans l'annuaire de l'utilisateur
31	DeleteContactsBook	Suppression de l'annuaire d'un utilisateur
32	**DeleteUser**	**Suppression d'un utilisateur et de son annuaire**
33	DeleteRights	Suppression des droits de lecture pour un utilisateur
40	GetContact	Affichage des infos d'un contact existant dans l'annuaire de l'utilisateur
41	GetContactsBook	Affichage des infos de tous les contacts existants dans l'annuaire de l'utilisateur
42	GetRights	Affichage des infos sur les droits de lecture

Gras = requêtes admin

Voici l'ensemble de nos requêtes pour les utilisateurs ainsi que pour l'admin.

Codes réponses :

- 5x = Réponses liées à la connexion
- 6x = Réponses liées au champ “Requête”
- 7x = Réponses liées à un annuaire
- 8x = Réponses liées à un utilisateur

Code	Nom de la requête	Description
50	AUTH_OK	Authentification OK
51	**AUTH_ERR**	**Erreur d’authentification**
52	DISCONNECT_CONFIRM	Confirmation de la déconnexion
60	REQ_OK	Traitement requête effectué
61	**REQ_ERR**	**Traitement requête échoué**
70	**NOT_ALLOWED**	**Accès non autorisé à l’annuaire**
71	**BOOK_NOT_FOUND**	**Annuaire non trouvé**
72	**UPDATE_FAILED**	**Modification impossible**
73	**BOOK_EXISTS**	**L’annuaire existe déjà**
80	**USER_EXISTS**	**L’utilisateur existe déjà**
81	**USER_NOT_FOUND**	**Utilisateur non trouvé**

Bleu = Codes d’erreur

Voici l’ensemble de nos réponses de notre serveur, on y retrouve notamment toutes les erreurs nécessaires au bon fonctionnement de l’application.

Menus de l'application :

Initialisation :

******* Connexion *******
Veuillez vous connecter…
Login : [username]
Mdp : [password]

Après l'établissement de la connexion, on se connecte à l'aide d'un login et d'un mot de passe.

Menu principal :

******* Menu principal *******
1. Menu Affichage
2. Menu Création
3. Menu Modification
4. Menu Suppression
5. Attribution de droits
6. Déconnexion

Choisissez un menu : [nb]

Après s'être connecté avec nos identifiants, on affiche le menu principal qui contient tous les menus de notre application. Pour aller dans un sous-menu, on entre le nombre correspondant par exemple 1 pour ouvrir le menu Affichage. Si l'utilisateur veut se déconnecter il entre la valeur 6 afin de lancer la fonction qui permet de se déconnecter.

Menu Affichage :

******* Menu Affichage *******
1. Contact
2. Annuaire
3. Retour en arrière

Choisissez un élément : [nb]

Dans ce menu, on peut choisir d'afficher un contact ou un annuaire. Pour ce faire, il suffit d'entrer la valeur correspondante au clavier. On peut aussi retourner au menu principal grâce au retour en arrière.

Menu Suppression :

******* Menu Suppression*******
1. Contact
2. Annuaire
3. Utilisateur
4. Retour en arrière
Choisissez un élément : [nb]

Dans ce menu, on peut choisir de supprimer un contact ou un annuaire. Pour ce faire, il suffit d'entrer la valeur correspondante au clavier. On peut aussi supprimer des utilisateurs si on est connecté en tant qu'Admin et on peut toujours retourner au menu principal grâce au retour en arrière.

Menu Modification :

******* Menu Modification*******
1. Contact
2. Utilisateur
3. Retour en arrière
Choisissez un élément :[nb]

Dans ce menu, l'utilisateur peut choisir de modifier un contact ou un utilisateur. Pour ce faire, il suffit d'entrer la valeur correspondante au clavier. On peut aussi retourner au menu principal grâce au retour en arrière.

Menu Attribution des droits :

******* Attribution des droits*******
1. Attribution des droits de lecture
2. Affichage des utilisateurs possédant des droits de lecture
3. Suppression des droits de lecture
4. Retour en arrière
Choisissez un élément :[nb]

Dans ce menu, l'utilisateur peut attribuer des droits de lecture à d'autres utilisateurs. Il peut aussi afficher tous les utilisateurs possédants des droits de lecture et supprimer les droits de lecture pour ces mêmes utilisateurs. On choisit l'action à effectuer au clavier et on peut aussi retourner au menu principal grâce au retour en arrière.

Menu création :

******* Menu Création *******
1. Contact
2. Annuaire
3. Utilisateur
4. Retour en arrière
Choisissez un élément :[nb]

Dans ce menu, l'utilisateur peut créer un nouveau contact dans son annuaire. Il peut aussi créer un nouvel annuaire. La fonction utilisateur est seulement utilisé pour l'admin car il est le seul à avoir le droit de créer de nouveaux utilisateurs.

Accès utilisateur et admin :

Utilisateur : Vert = accès autorisé – Rouge = accès refusé

Menu Affichage	• Contact • Annuaire
Menu Création	• Contact • Annuaire • Utilisateur
Menu Modification	• Contact • Utilisateur
Menu Suppression	• Contact • Annuaire • Utilisateur
Menu Attribution des droits	• Attribution des droits de lecture • Affichage des utilisateurs possédant des droits de lecture • Suppression des droits de lecture

Admin : Vert = accès autorisé – Rouge = accès refusé

Menu Affichage	• Contact • Annuaire
Menu Création	• Contact • Annuaire • Utilisateur
Menu Modification	• Contact • Utilisateur
Menu Suppression	• Contact • Annuaire • Utilisateur
Menu Attribution des droits	• Attribution des droits de lecture • Affichage des utilisateurs possédant des droits de lecture • Suppression des droits de lecture

Etape 2 : Conception des algorithmes du client et du serveur

Le but de cette étape est d'écrire les algorithmes généraux du client et du serveur.
On réalisera, au minimum, deux raffinages tels que : le premier niveau de raffinage présente un algorithme suffisamment succinct pour tenir sur une demi-page A4. Ceci permettant d'avoir les algorithmes de haut niveau du client et du serveur sur la même page.
Le second niveau de raffinage pourra présenter les algorithmes des différentes fonctions appelées dans les algorithmes de premier niveau de raffinage.

1er raffinage : (main global)

Client :

Algorithme mainClient
/* Permet d'accéder aux contacts d'un annuaire et de demander sa lecture ou son écriture.*/

Variables
login : chaîne de caractères
passwd : chaîne de caractères
reponseConnexion : enregistrement reponse
utilisateur : enregistrement user
choix1 : entier
fini :entier

Début

```
fini=0
Ecrire("******* Connexion *******")
Ecrire("Veuillez vous connecter")
Ecrire("Login : ")
Lire(utilisateur.login)

Ecrire("Mot de passe : ")
Lire(utilisateur.password)

Connexion(E: login, E: passwd, E: "13214")
Reception(&reponseConnexion)
```

```
Si (reponseConnexion.code = 50) Alors
   Affichage du menu principal
   Lire(choix1)

   TantQue !fini Faire
         Faire
            /* Chaque fonction affiche un sous-menu avec d'autres choix puis envoie une
            requête avec cible précise. */
                  Selon expression dans choix1 :
                        choix1 = 1:
                              EnvoyerRequeteAffichage(E: login, E: passwd)
                        choix1 = 2:
                              EnvoyerRequeteCreation(E: login, E: passwd)
                        choix1 = 3:
                              EnvoyerRequeteModification(E: login, E: passwd)
                        choix1 = 4:
                              EnvoyerRequeteSuppression(E: login, E: passwd)
                        Choix1 = 5:
                           EnvoyerRequeteDroits(E: login, E: passwd)
                        Choix1 = 6:
                           fini = 1
                           Deconnexion()
               Sinon
                  Ecrire("Ce choix n'existe pas.")
                     Fin Selon
            TantQue (choix1 != 1 et choix1 != 2 et choix1 != 3 et choix1 != 4 et choix1 != 5 et
         choix1 != 6)
   Fin TantQue

   SinonSi (reponseConnexion.code = 51) Alors
      Ecrire("Connexion échouée")
 Fin Si
Fin
```

Serveur :

Algorithme mainServeur:
/* Permet la gestion d'annuaires et de contacts.*/

Type
Requete = **enregistrement**
Type : entier
login : chaîne de caractères
pasword : chaîne de caractères
data : union data
finenreg

Variables
requete : Requete /* requête reçue par le serveur */

Début
Initialisation du service
TantQue 1 Faire :
Début
Mise en attente d'une réponse

requete= message reçu
Émettre(connexion au serveur)
Extraction des informations contenues dans le message
Si la requête contient le login et le mot de passe d'un utilisateur connu **alors**
Emettre(connexion ok)
Si la requête est de type création **alors**
Traiter la requête création (requete)
Sinon si la requête est de type mise à jour **alors**
Traiter la requête mise à jour(requete)
Sinon si la requête est de type suppression **alors**
Traiter la requête suppression(requete)
Sinon si la requête est de type lecture **alors**
Traiter la requête lecture(requete)
Sinon
Emettre(déconnexion du serveur)
Déconnexion du client
Finsi
Sinon
Émettre(déconnexion du serveur)
Déconnexion du client
Finsi
Fin
Fin TantQue
Fin

2ème raffinage :

Client :

Procédure EnvoyerRequeteAffichage(E: login, E: passwd) :
/*Permet d'afficher le sous-menu affichage et d'envoyer les requêtes correspondantes*/
Variable
choix2 : entier
req : structure requete

Début
req.login = login
req.passwd = passwd
Faire
Ecrire("******* Menu Affichage *******")
Ecrire("1. Contact")
Ecrire("2. Annuaire")
Ecrire("3. Retour en arrière")
Ecrire("Choisissez un élément : ")
Lire(choix2)

Selon expression Dans choix2 :
choix2 = 1:
req.type = "40"
Emission(req)
Retourne 0
choix2 = 2:
req.type = "41"
Emission(req)
Retourne 0
choix2 = 3:
Retourne 1
Sinon
Ecrire("Ce choix n'existe pas.")
Fin Selon
TantQue (choix2 != 1 et choix2 != 2 et choix2 != 3)

Emission(req)
Fin

Procédure EnvoyerRequeteCreation(E: login, E: passwd) :
/*Permet d'afficher le sous-menu création et d'envoyer les requêtes correspondantes*/
Variable
choix2 : entier
req: structure requete

Début
Ecrire("******* Menu Création*******")
Ecrire("1. Contact")
Ecrire("2. Annuaire")
Ecrire("3. Utilisateur")
Ecrire("4. Retour en arrière")
Ecrire("Choisissez un élément : ")
Lire(choix2)

Faire
Selon expression Dans choix2 :
choix2 = 1:
Si l'annuaire n'existe pas **Alors**
req= CreateContact()
Sinon
Ecrire("Un contact existe déjà pour cet utilisateur");
Fin Si
Retourne 0
choix2 = 2:
Si l'annuaire n'existe pas **Alors**
req= CreateAnnuaire()
Sinon
Ecrire("Un annuaire existe déjà pour cet utilisateur")
Fin Si
Retourne 0
choix2 = 3:
Si (login = "admin") **Alors**
req = CreateUser()
Sinon
Ecrire("Vous n'êtes pas autorisé à effectuer cette commande.")
Fin Si
Retourne 0
choix2 = 4:
Retourne 1
Sinon
Ecrire("Ce choix n'existe pas.")
Fin Selon
TantQue (choix2 != 1 et choix2 != 2 et choix2 != 3 et choix2 != 4)
Emission(req)
Fin

Procédure EnvoyerRequeteModification(E: login, E: passwd) :
/*Permet d'afficher le sous-menu modification et d'envoyer les requêtes correspondantes*/

Variable
choix2 : entier
requete : chaîne de caractères

Début
Ecrire("******* Menu Modification*******")
Ecrire("1. Contact")
Ecrire("2. Utilisateur")
Ecrire("3. Retour en arrière")
Ecrire("Choisissez un élément : ")
Lire(choix2)

Faire
Selon expression Dans choix2 :
choix2 = 1:
requete = UpdateContact()
choix2 = 2:
Si (login = "admin" et passwd = "admin123*") **Alors**
requete = UpdateUser()
Sinon
Ecrire("Vous n'êtes pas autorisé à effectuer cette commande.")
Fin Si
choix2 = 3:
break
Sinon
Ecrire("Ce choix n'existe pas.")
Fin Selon
TantQue (choix2 != 1 et choix2 != 2 et choix2 != 3)

Emission(requete)
Fin

Procédure EnvoyerRequeteSuppression(E: login, E: passwd) :
/*Permet d'afficher le sous-menu suppression et d'envoyer les requêtes correspondantes*/

Variable
choix2 : entier
req : structure requete

Début
Ecrire("******* Menu Suppression*******")
Ecrire("1. Contact")
Ecrire("2. Annuaire")
Ecrire("3. Utilisateur")
Ecrire("4. Retour en arrière")
Ecrire("Choisissez un élément : ")
Lire(choix2)

Faire
Selon expression Dans choix2 :
choix2 = 1:
DeleteContact(req)
Retourne 0
choix2 = 2:
DeleteAnnuaire(req)
Retourne 0
choix2 = 3:
Si (login = "admin") **Alors**
DeleteUser(req)
Sinon
Ecrire("Vous n'êtes pas autorisé à effectuer cette commande.")
Fin Si
Retourne 0
choix2 = 4:
Retourne 1
Sinon
Ecrire("Ce choix n'existe pas.")
Fin Selon
TantQue (choix2 != 1 et choix2 != 2 et choix2 != 3 et choix2 != 4)
Fin

Procédure AffichageMenuPrincipal() :
/*Permet d'afficher le menu principal*/
Début
Ecrire("******* Menu principal *******")
Ecrire("1. Menu Affichage")
Ecrire("2. Menu Création")
Ecrire("3. Menu Modification")
Ecrire("4. Menu Suppression")
Ecrire("5. Menu Attribution de Droits")
Ecrire("6. Déconnexion")
Ecrire("Choisissez un menu : ")
Fin

Serveur :

Type
user = **enregistrement**
login : chaîne de caractères
pasword : chaîne de caractères
finenreg

contact = **enregistrement**
nom : chaîne de caractères
prénom : chaîne de caractères
portable : chaîne de caractères
adresse : chaîne de caractères
mail : chaîne de caractères
naissance : chaîne de caractères
remarques : chaîne de caractères
finenreg

/* Procédure qui crée un utilisateur, un annuaire ou un contact*/
Procédure *TraiterRequeteCreation*(E: struc requete : requete)

Variables

tmp : chaine de 51 caractères

F pointeur vers FILE

Rep : variable de type réponse

Debut

Si req.login = 0 et req.type = CREATE_USER **Alors**

Ouvrir en lecture le fichier utilisateur.txt dans f

Si f=NULL **Alors**

Afficher « Erreur lors de l'ouverture du fichier »

Retourne -1

FinSi

tmp ← req.data.utilisateur.login

Ajouter « ; » à la chaine de caractères tmp

Ajouter req.data.utilisateur.password à la chaine de caractères tmp

Ajouter «\r\n» à la chaine de caractères tmp

. rep.code = REQ_OK

Rep.data.message← « RequestSuccesful »

Ecrire(f, « %s »,tmp)

Emission(rep) :

Arrêter de lire

Retourne 0

Sinon

Retourne 1

FinSi

Fin

```
/* Procédure qui supprime un utilisateur, un annuaire ou un contact.*/
Procédure TraiterRequeteSuppression (E: struc requete : requete)

Variable
        fichierIn : pointeur vers fichier          /* fichier à ouvrir */
        fichierOut :  pointeur vers fichier        /* fichier sans le contact à supprimer */
        utilisateur : structure user               /* utilisateur à supprimer du fichier */
        utilisateurLu : chaine de caractères       /* utilisateur contenu dans le fichier */
        contact: structure contact                 /* contact à supprimer du fichier de l'utilisateur */
        contactLu : chaine de caractères           /* contact contenu dans le fichier */

Début
  Si  requete contient  le login et le mot de passe de l'administrateur  alors
            Début
                fichierIn : Ouvrir le fichier des utilisateurs
                utilisateur : utilisateur à supprimer contenu dans requete
                Si le fichier n'est pas vide et utilisateur conforme alors
                  Début
                    fichierOUT : Ouvrir un fichier temporaire
                    utilisateurLu. = premier utilisateur dans le fichier

                    Tant que ce n'est pas la fin du fichier faire
                      Début
                        Si utilisateurLu est différent de utilisateur alors
                                Ecrire utilisateurLu dans fichierOut
                        utilisateur = Lire(utilisateur suivant du fichier)
                      Fin
                    FinTantque

                  Fermer les fichiers
                  Supprimer fichierIn
                  Renommer fichierOut en fichier des utilisateurs
                  Emettre(Requête Effectuée)
                  Fin
                Sinon
                  Début
                  Fermer fichierIn
                  Emettre(Utilisateur non trouvé)
                  Fin
                Finsi
            Fin
        Sinon
            Début
              contact : contact à supprimer contenu dans requete
              fichierIn : Ouvrir le fichier de l'utilisateurs
              fichierOUT : Ouvrir un fichier temporaire
```

```
                contactLu = Lire( premier contact du fichier)
                Tantque ce n'est pas la fin du fichier faire
                        Si contactLu est différent de contact alors
                                Ecrire contactLu dans FichierOut
                        contactLu = Lire( contact suivant du fichier)
                Fin Tantque

                Fermer les fichiers
                Supprimer fichierIn
                Renomer fichierOut en fichier de l'utilisateur
                Emettre(Requête Effectuée)
            Fin
        Finsi
Fin
```

```
/* Procédure qui modifier le mot de passe d'un utilisateur ou les informations d'un contact*/
Procédure TraiterRequeteModification (E: struc requete : requete)

Variable
        fichierIn :  fichier          /* fichier à ouvrir */
        fichierOut :  fichier         /* fichier avec le contact modifié */
        utilisateurAncien : user      /* utilisateur à modifier dans le fichier */
         utilisateurNouveau : user    /* utilisateur modifié dans le fichier */
        utilisateurLu : user          /* utilisateur contenu dans le fichier */
         contactAncien: contact       /* contact à modifier dans le fichier de l'utilisateur */
        contactNouveau: contact       /* contact  modifié dans le fichier de l'utilisateur */
        contactLu : contact           /* Contact contenu dans le fichier */
Début
  Si  requete contient  le login et le mot de passe de l'administrateur  alors
                Début
                        fichierIn : Ouvrir le fichier des utilisateurs
                        utilisateurAncien : utilisateur à modifier contenu dans requete
                        utilisateurNouveau : nouvelles valeurs de utilisateur contenu dans requete

                        Si le fichier n'est pas vide et UtilisateurAncien conforme alors
                          Début
                                fichierOUT : Ouvrir un fichier temporaire
                                utilisateurLu = premier utilisateur dans le fichier
```

```
                    Tant que ce n'est pas la fin du fichier faire
                      Début
                            Si utilisateurLu est différente de utilisateurAncien alors
                              Ecrire utilisateurLu dans fichierOut
                            Sinon
                              Ecrire utilisateurNouveau dans fichierOut
                            Finsi
                            utilisateurLu = Lire( utilisateur suivant du fichier)
                      Fin
                    Fin Tantque
                  Fermer les fichiers
                  Supprimer fichierIn
                  Renomer fichierOut en fichier des utilisateurs
                  Emettre(Requête Effectuée)
                  Fin
                Sinon
                  Début
                  Fermer fichierIn
                  Emettre(Utilisateur non trouvé)
                  Fin
                Finsi
          Fin
      Sinon
    Début
          contactAncien: contact à modifier contenu dans requete
          contactNouveau: nouvelle valeur de contact contenu dans requete
          fichierIn : Ouvrir le fichier de l'utilisateurs
          fichierOUT : Ouvrir un fichier temporaire
          contactLu = Lire( premier contact du fichier)
          Tant que ce n'est pas la fin du fichier faire
                    Si contactLu est différent de contactAncien alors
                        Ecrire contactLu dans FichierOut
                    Sinon
                        Ecrire contactNouveau dans fichierOut
                    Finsi
                    contactLu = Lire(contact suivant du fichier)
                FinTantque

                Fermer les fichiers
               Supprimer fichierIn
               Renomer fichierOut en fichier de l'utilisateur
               Emettre(Requête Effectuée)
          Fin
      Finsi
Fin
```

```
/* Procédure qui affiche les informations d'un contact ou de tous les contacts de l'annuaire*/
Procédure TraiterRequeteAffichage (E: struc Requete : requete) :
Variable
        fichier :  pointeur vers fichier       /* fichier à ouvrir */
        utilisateur : chaine de caractères    /* utilisateur lu dans le fichier */
        contact: chaine de caractères    /* contact lu dans le fichier de l'utilisateur */

Début
        Si  requete contient  le login et le mot de passe de l'administrateur alors
                Début
                        fichier : Ouvrir le fichier des utilisateurs
                        Si le fichier n'est pas vide alors
                          Début
                                Faire
                                  Début
                                        utilisateur =  lire utilisateur dans le fichier
                                        Emettre utilisateur
                                  Fin
                                Tantque il y a des utilisateur à lire dans le fichier
                          Fin
                        Sinon
                                Emettre(Pas de données trouvées)
                        Finsi
                        Fermer le fichier
                        Emettre fin d'émission
                        Emettre(Requête Effectuée)
                Fin
        Sinon
                Début
                  fichier : Ouvrir le fichier de l'utilisateur
                  Si le fichier n'est pas vide alors
                        Début
                                Faire
                                  Début
                                        contact =  lire contact dans le fichier
                                        Emettre contact
                                  Fin
                                Tantque il y a des contacts à lire dans le fichier
                          Fin
                        Sinon
                                Emettre(Pas de données trouvées)
                        Finsi
                        Fermer le fichier
                        Emettre fin d'émission
                        Emettre(Requête Effectuée)
                Fin
        Finsi
```

Fin

Algorithme VerificationConnexion

/*Vérifie la combinaison login/password envoyé par le client*/

Variables

Contenufichier, concatLogpass : chaine de 51 caractères

F pointeur vers FILE

Rep : variable de type réponse

Debut

Req = Reception

Si req.type = BIND **Alors**

concatLogPass ← req.login

Ajouter « ; » à la chaine de caractères concatLogPass

Ajouter req.password à la chaine de caractères concatLogPass

Ajouter «\r\n» à la chaine de caractères concatLogPass

Ouvrir en lecture le fichier utilisateur.txt dans f

Si f=NULL **Alors**

Afficher « Erreur lors de l'ouverture du fichier »

Retourne -1

FinSi

Tantque on lit contenufichier

Si contenufichier = concatLogPass **Alors**

Rep.code = AUTH_OK

Rep.data.message ← « AuthOk »

Emission(rep)

Ecrire(« Client connecté)

FinSI

FinTantQue

Si contenufichier != concatLogPass **Alors**

Rep.code = AUTH_ERR

Rep.data.message ← « AuthError »

Emission(rep)

Ecrire(« Erreur d'authentification du client »)

FinSI

Arrêter de lire

Sinon

Ecrire (« Requête incorrecte »)

Rep.code = REQ_ERR

Rep.data.message ← « RequestError »

Emission(rep)

DeconnexionClient()

FinSi

Retourne 0

Fin

Algorithme Requete_au_Client
/*Reçois la requête du client*/

Variables

Req variable de type requete

Début

Recevoir(socketService, &req, taille de (requete))

Retourne req

Fin

Algorithme Réponse_au_Client
/*Réponse du server suite à la requête du client*/

Debut

Si sennd(socketService, &rep, sizeof(rep) = -1 **Alors**

Ecrire(« Emission, probleme lors du send »)

Retourne 0

FinSi

Retourne 1

Fin

Algorithme Traiter_Requete_Delete
/* Traitement des différentes requêtes disponibles dans le menu Suppression. */

Variables

f pointeur vers FILE

replique pointeur vers FILE

Rep : variable de type réponse

Contenufichier, concatUser : chaine de 51 caractères

fname: chaine de 30 caractères = « utilisateurs.txt »

temp: chaine de 30 caractères = « temp.txt »

Debut

Si req.login = « admin » **ET** req.type = DELETE_USER **Alors**

Ouvrir en lecture le fichier fname dans f

Si f = NULL **Alors**

Ecrire(« Erreur lors de l'ouverture du fichier »)

Retourne -1

FinSi

Ouvrir en écriture le fichier temp dans replique

Si replique = NULL **Alors**

Ecrire(« Erreur lors de l'ouverture du fichier »)

Retourne -1

FinSi

concatUser < -- req.data.utilisateur.login

Ajouter « ; » à la chaine de caractères concatUser

Ajouter req.data.utilisateur.password à la chaine de caractères concatUser

Ajouter «\r\n» à la chaine de caractères concatUser

Tantque on lit contenufichier

Si contenufichier != concatUser **Alors**

Ecrire(replique, « %s », contenufichier)

FinSi

Rep.code = REQ_OK

Rep.data.message < -- « RequestSuccesful »

Emission(rep)

Fermer le fichier (f)

Fermer le fichier (replique)

Supprimer le ficher (fname)

Renommer temp en fname

Retourne 0

FinTantQue

Fin

Attribution des droits :

fonction **CREATE_RIGHTS**(E: struc requete : requete)

/*Permet d'attribuer des droits à un utilisateur*/

Variables

fichier : fichier /* fichier à ouvrir */

ligne : droit /* Nouvelle entrée dans le fichier de gestion des droits */

Début

Extraction des informations contenues dans le message:

nomUtilisateur

fichier : Ouvrir le fichier des droits

Si ouverture impossible **alors**

Emettre(Annuaire non trouvé)

Finsi

ligne = enregistrement(nomUtilisateur = Utilisateur.nom)

Ajouter ligne à fichier

Fermer fichier

Émettre(Requête effectuée)

Fin

fonction **GET_RIGHTS** (E: struc requete : requete, S ayantDroit)

/*Permet d'afficher les utilisateurs possédant des droits sur l'annuaire*/

Variable

fichier : fichier /* fichier à ouvrir */

ligne : chaîne de caractères /* Ligne dans le fichier de gestion des droits */

ayantDroit: droit /* Structure contenant les ayants droits extraits du fichier et retourné */

Début

Extraction des informations contenues dans le message:

nomUtilisateur

fichier : Ouvrir le fichier des droits

ayantDroit = droit(nomUtilisateur = Utilisateur)

Si ouverture impossible **alors**

Emettre(Annuaire non trouvé)

Finsi

Tant que ce n'est pas la fin du fichier faire

Lire une ligne

Si nomUtilisateur = Utilisateur **alors**

Ajouter les ayant droits à ayantDroit

Finsi

Fin tant que

Fermer fichier

Émettre(Requête effectuée)

Renvoyer ayantDroit

Fin

fonction **DELETE_RIGHTS**(E: struc requete : requete)

/*Permet de supprimer un utilisateur possédant des droits sur l'annuaire*/

Variable
fichierIn : fichier /* fichier à ouvrir */
fichierOut : fichier /* fichier sans le contact à supprimer */
ligne : chaîne de caractères /* Ligne dans le fichier de gestion des droits */

Début
fichierIn : Ouvrir le fichier des droits
Extraction des informations contenues dans le message:
nomUtilisateur
Si le fichier n'est pas vide et utilisateur conforme **alors**
Début
fichierOut = ouvrir un fichier temporaire
Tant que ce n'est pas la fin du fichier faire
Début
Lire une ligne
Si nomUtilisateur dans ligne est différent de Utilisateur.nom **alors** :
Ecrire ligne lue dans fichierOut
Finsi
Fin tant que
Fermer les fichiers
Supprimer fichierIn
Renommer fichierOut en fichier de droits
Émettre(Requête Effectuée)
Fin
Sinon
Début
Fermer fichierIn
Émettre(Utilisateur non trouvé)
Fin
Finsi
Fin

Etape 3 : Codage du client et du serveur (1ère partie)

Écrire le code en langage C de l'application Client/Serveur correspondant aux fonctions de l'administrateur du site et à la connexion d'un utilisateur au serveur. Il sera tenu compte des commentaires et de la décomposition fonctionnelle.

Le code source de notre application est disponible dans le dossier ci-joint.

Exécution de l'application :

Compilation :

Pour compiler nos fichiers, il faut exécuter les 2 commandes ci-dessous en étant en Bash :

- gcc mainServeur.c -o serv
- gcc mainClient.c -o cli

Connexion :

- Côté Client

```
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# gcc mainClient.c -o cli
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./cli
******* Connexion *******
Veuillez vous connecter…
Login : admin
Mot de passe : admin*
Connexion avec le serveur réussie.

******* Menu principal *******
1. Menu Affichage
2. Menu Création
3. Menu Modification
4. Menu Suppression
5. Attribution de droits
6. Déconnexion

Choisissez un menu :
```

- Côté Serveur

```
root@DESKTOP-66R9QG0: /mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C
Microsoft Windows [version 10.0.17763.914]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Users\dobli\OneDrive\Bureau\Prog Lang C>bash
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# gcc mainServeur.c -o serv
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./serv
Création du serveur réussie sur 13214.
Reçu une requête de connexion : 0
Client connecté.
```

Création d'utilisateur (fonctionnalité admin) :

- Côté Client

```
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./cli
******* Connexion *******
Veuillez vous connecter…
Login : admin
Mot de passe : admin*
Connexion avec le serveur réussie.

******* Menu principal *******
1. Menu Affichage
2. Menu Création
3. Menu Modification
4. Menu Suppression
5. Attribution de droits
6. Déconnexion

Choisissez un menu : 2

******* Menu Création *******
1. Contact
2. Annuaire
3. Utilisateur
4. Retour en arrière

Choisissez un élément : 3

Nom du nouvel utilisateur : issalis
Mot de passe du nouvel utilisateur : issalis*
Utilisateur créé avec succès.
```

- Côté Serveur

```
root@DESKTOP-66R9QG0: /mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C
Microsoft Windows [version 10.0.17763.914]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Users\dobli\OneDrive\Bureau\Prog Lang C>bash
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# gcc mainServeur.c -o serv
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./serv
Création du serveur réussie sur 13214.
Reçu une requête de connexion : 0
Client connecté.

Reçu une requête de type CREATE : 12
```

On peut bien entendu se connecter avec l'utilisateur créé par l'admin comme vous pouvez le voir :

```
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./cli
******* Connexion *******
Veuillez vous connecter…
Login : issalis
Mot de passe : issalis*
Connexion avec le serveur réussie.
```

Suppression d'utilisateur (fonctionnalité admin) :

- Côté Client

```
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# ./cli
******* Connexion *******
Veuillez vous connecter…
Login : admin
Mot de passe : admin*
Connexion avec le serveur réussie.

******* Menu principal *******
1. Menu Affichage
2. Menu Création
3. Menu Modification
4. Menu Suppression
5. Attribution de droits
6. Déconnexion

Choisissez un menu : 4

******* Menu Suppression *******
1. Contact
2. Annuaire
3. Utilisateur
4. Retour en arrière

Choisissez un élément : 3

Nom de l'utilisateur à supprimer : issalis
Mot de passe de l'utilisateur à supprimer : issalis*
Utilisateur supprimé avec succès.
```

- Côté Serveur

```
Reçu une requête de connexion : 0
Client connecté.

Reçu une requête de type DELETE : 32
```

Déconnexion :

- Côté Client

```
******* Menu principal *******
1. Menu Affichage
2. Menu Création
3. Menu Modification
4. Menu Suppression
5. Attribution de droits
6. Déconnexion

Choisissez un menu : 6

Déconnexion...
root@DESKTOP-66R9QG0:/mnt/c/Users/dobli/OneDrive/Bureau/Prog Lang C# _
```

- Côté Serveur

```
Reçu une requête de déconnexion : 1
Envoie d'une requête de confirmation de déconnexion : 52
Client déconnecté.
```

Création de contact :

- Côté Client

```
******* Menu Création *******
1. Contact
2. Annuaire
3. Utilisateur
4. Retour en arrière

Choisissez un élément : 1

Veuillez remplir les champs obligatoire ci-dessous :
Nom du contact : Aoun
Prénom du contact : André
Mail du contact : andré.aoun@univ-tlse3.fr

Souhaitez-vous ajouter des données supplémentaires ? (o/O/n/N) : n
Contact créé avec succès.
```

- Côté Serveur

```
Reçu une requête de type CREATE : 10
Envoie d'une réponse avec le code : 60
```

Affichage annuaire :

- Côté Client

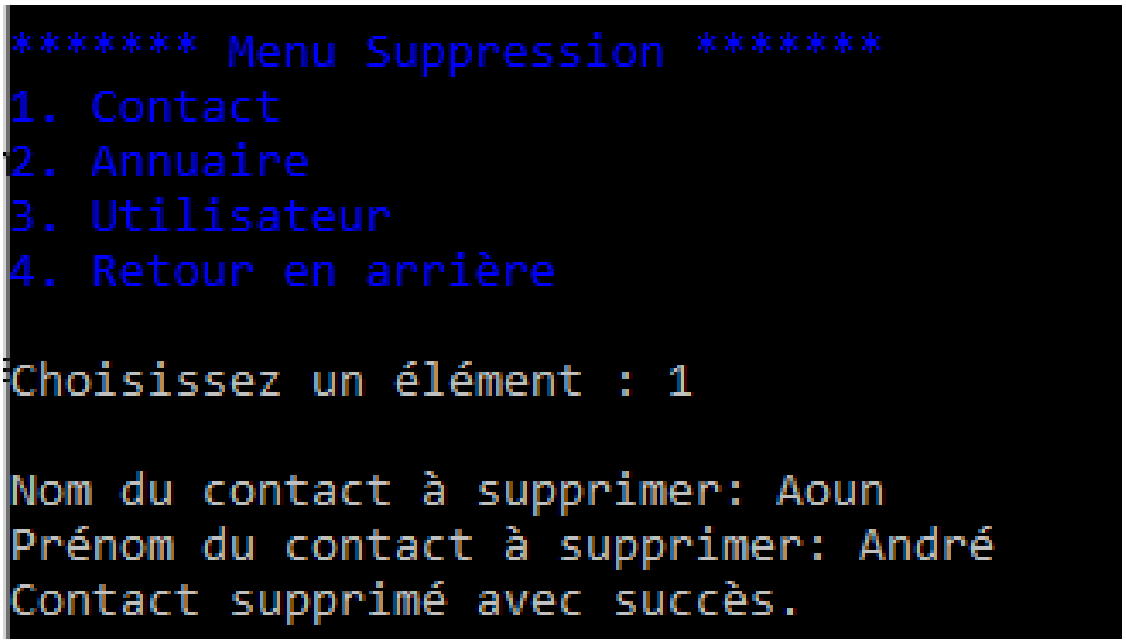
```
******* Menu Affichage *******
1. Contact
2. Annuaire
3. Retour en arrière

Choisissez un élément : 2

Souhaitez-vous afficher des données supplémentaires ? (o/O/n/N) : o

Nom : BERNARD
Prénom : Robert
Mail : robert.bernard@hotmail.com
Tél portable : 0666666666
Adresse postale : 85 rue de blah blah, 31400 Toulouse
Date de naissance : 01/03/1949
Remarques : abcd

Nom : VIDAL
Prénom : Louis
Mail : louis.vidal@hotmail.com
Tél portable : Non renseigné
Adresse postale : Non renseigné
Date de naissance : Non renseigné
Remarques : Non renseigné

Nom : Aoun
Prénom : André
Mail : andré.aoun@univ-tlse3.fr
Tél portable : Non renseigné
Adresse postale : Non renseigné
Date de naissance : Non renseigné
Remarques : Non renseigné
```

- Côté Serveur

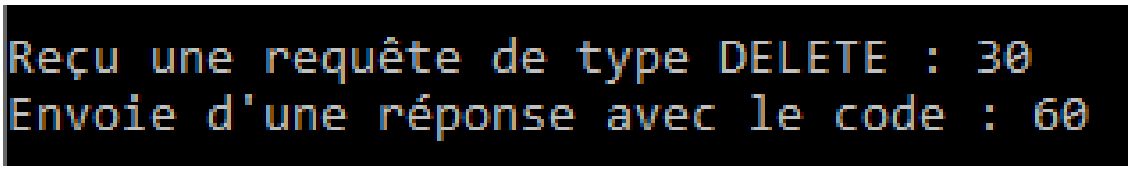
```
Reçu une requête de type GET : 41
Envoie d'une réponse avec le code : 60
```

Suppression contact :

- Côté Client



- Côté Serveur



Fonctions non implémentées

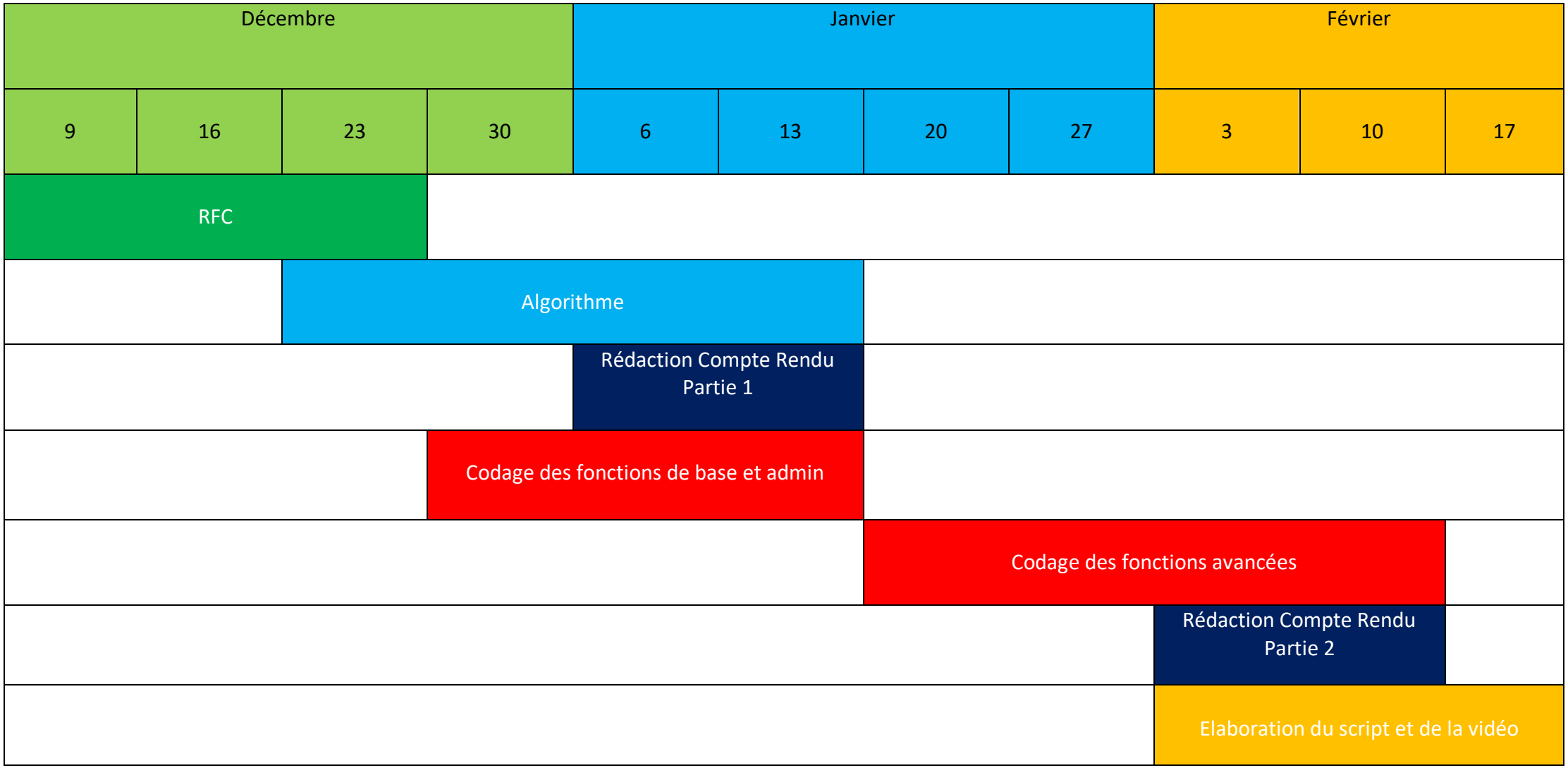
Par manque de temps, nous n'avons pas implémenter les fonctions suivantes :

- Fonctions du Menu Attribution des droits
- Fonction Update Contact

Répartition des tâches

- Brainstorming : Création des PDU
- BRONGNIART Clément : Codage des programmes en C, Création RFC, Script Vidéo
- PONS Jean-Louis : Création des algorithmes, Début de codage des fonctions attribution des droits
- NICOLAS Vincent : Création des algorithmes
- ISSALIS Dorian : Création des 2 compte-rendu et RFC, Codage des fonctions création contact et annuaire, Vidéo

Planning



BRONGNIART Clément - ISSALIS Dorian
NICOLAS Vincent – PONS Jean-louis

2020