Université Mohammed V-Rabat
Ecole Nationale Supérieure
d'Informatique et d'Analyse des
Systèmes

RAPPORT

DE PROJET DE LANGAGE C

Par

NIDABRAHIM Youssef
NAJI Kawtar

Sujet :

Gestion des actes d'état civil numérisés

Soutenu le 3 Février 2016 devant le jury :

Pr. ELHASSOUNY Azeddine : Encadrant

Année universitaire 2015/2016


بسم الله الرحمن الرحيم

Résumé

Les registres contiennent généralement plusieurs actes mais les agents des actes ne doivent remettre au demandeur que les informations qui le concernant.

Actuellement, cette manipulation peut mobiliser un temps non négligeable (sortie du registre, copie des éléments, etc).

Un projet de numérisation revient à gagner du temps et du confort au niveau de la gestion du travail tant pour les agents territoriaux que pour l'attente du public. Notre application permet de faire une meilleure gestion des actes (naissances, mariages et décès).

Ses avantages majeurs sont la simplicité d'utilisation et le gain de temps c.-à-d. faire une copie intégrale ne prend plus que deux minutes devant l'ordinateur, sans toucher aux registres.

Remerciements

Il nous est agréable de nous acquitter d'une dette de reconnaissance auprès de toutes les personnes, dont l'intervention au cours de ce projet, a favorisé son aboutissement.

Nos sincères remerciements à **Monsieur Elhassouny Azeddine** pour le suivi interrompu de ce projet, pour ses conseils, et son appui tout au long de ce travail.

Il nous a fait l'honneur d'être notre encadrant. Nous le remercions pour son encouragement et aussi d'être toujours là pour nous écouter, nous aider et nous guidder à trouver le bon chemin par sa sagesse et ses précieux conseils, ce qui nous a donné la force d'accomplir ce projet

Nous rendons hommage aux **Monsieur Faker** qui témoigne par ses conseils de l'intérêt qu'ils a bien voulu porter au projet.

Table des matières

Introduction générale

Nous présentons dans ce rapport les différentes étapes suivies pour la réalisation de notre projet. Il est axé sur trois grandes parties.

La première partie présente le cahier des charges dans lequel nous définissons le processus de développement suivi.

La deuxième partie décrit la phase d'analyse et conception.

La dernière partie présente les différentes étapes de la mise en œuvre du projet en exposant les différentes interfaces de l'application réalisée.

CHAPITRE I Contexte général du Projet

I.1 Cahiers des charges

On nous a confié la conception et la réalisation d'une application par usage de langage C.

Cette application sert à informatiser le processus de la gestion des registres, d'état civil, des actes d'état civil, notamment ceux relatifs à la naissance, au mariage, et au décès.

On souhaite mettre en place cette application vue que les nouvelles technologies de l'information prennent de plus en plus d'importance dans le secteur de la gestion des actes, et s'avèrent aujourd'hui être indispensables pour permettre à l'individu d'obtenir son état civil complet, la famille avec tous les conjoints et la descendance facilement et rapidement.

CHAPITRE II Acte Civil Numérisés

II.1 Gestion des actes d'état civil numérisés

L'informatisation de la gestion de l'état civil s'impose progressivement aux communes, Les nouvelles technologies qui se généralisent permettent d'améliorer le service rendu aux citoyens et d'envisager des évolutions dans les différentes pratiques administratives.
La gestion des actes d'état civil numérises remplace les manipulations des registres de l'état civil conservés par le service de gestion des actes.
Ces registres sont stockés dans des armoires. Si une personne vient chercher son acte de naissance par exemple on doit localiser le registre dans lequel se trouvait l'acte, cette procédure de recherche lourde peut se répétait de 100 à 200 fois par jours, nombre moyens de demandes des actes auquel le service de l'état civil doit faire face quotidiennement.

II.2 Les avantages de l'application

Le système de gestion électronique des actes au service de l'état civil a plusieurs avantages :

- ✓ Il permet d'améliorer sensiblement la qualité du service rendu aux administrés.
- ✓ Rendre les documents d'état civil rapidement accessibles
- ✓ Une personne qui vient de solliciter une copie d'acte ne doit pas attendre plusieurs minutes, voire plus, en fait quelques minutes suffisent en moyenne à un employé du service pour lui délivrer un document.
- ✓ Le service de gestions des actes est libéré de taches fastidieuses.

CHAPITRE III Analyse et conception du projet

Introduction :

L'analyse conceptuelle a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins de l'utilisateur. Et aussi de lister les résultats attendus, en termes de fonctionnalités, de performance, de maintenance, d'extensibilité, etc.

III.1 Analyse de projet

a. Spécification des besoins fonctionnels

Cette application de la gestion d'état civil, vise à simplifier les démarches administratives des usagers et à limiter la fraude documentaire.
On imagine aisément que cette application deviendra un outil essentiel pour les services de l'état civil. Système souple et très sécurisé qui facilite grandement l'obtention rapide des informations d'une personne, les actes d'état civil, notamment ceux relatifs à la naissance, au mariage, et au décès.
Plusieurs avantages sont identifiés, des gains de gestion administrative de transmission d'actes, la réduction des délais.

L'application doit être sécurisé, fiable et les données doivent être facilement accessible et disponible.

L'application doit gérer d'une manière efficace les registres d'état civil en réduisant le temps de traitement et fournir des documents de meilleure qualité aux citoyens.

L'application doit être dynamique dans tous les sens du terme et avoir un accès simple en respectant les bonnes pratiques permettront à l'agent de trouver facilement les informations cherchées sur une personne ainsi que les éléments essentiels qui doivent être visibles et accessibles, l'ajout d'une personne, la suppression, la modification des enregistrements d'une personne, etc…

L'application doit permettre également de gérer les enfants d'une personne, à savoir l'affichage des informations sur les enfants d'une personne donnée ou la déclaration d'une naissance.

L'application doit mettre en place des formulaires pour bien ajouter des personnes où cas de déclaration d'une naissance et chaque personne doit avoir un unique identifiant généré automatiquement. Lors de l'ajout d'une personne, ses parents doivent être déjà inscrits.

L'application doit prendre en compte les relations entre les personnes.

L'application doit permettre à n'importe quel citoyen d'accéder à son propre espace afin de bien récupérer ses actes d'état civil. Cet espace citoyen doit regrouper toutes les demandes que l'utilisateur peut faire au service Etat Civil à savoir la demande d'acte de naissance, du mariage ou du décès.

L'accès à l'application doit être protégé par un mot de passe.

b. Services de l'application :

En se basant sur le cahier des charges et les précisions des besoins fonctionnels, on a établi une description du processus métier ainsi que la liste des principales règles de gestion.

Espace agent :

- Gestion des personnes :

  - Deux personnes ne peuvent avoir le même identifiant et il doit être générer automatiquement.
  - Vérifier la disponibilité des informations d'une personne.
  - Vérifier l'identité d'une personne.

- Fournir des formulaires pour la saisie et la modification des informations d'une personne.
- Ajouter une personne en prenant en considération les contraintes familiales par exemple le nom d'une personne doit être identique à celui de son père etc…
- Supprimer une personne.
- Modifier les enregistrements d'une personne.
- Consulter la liste des personnes inscrites.
- Chercher une personne par son nom ou prénom, identifiant.
- Retourner le nom de mère ou père d'une personne
- Retourner l'Age d'une personne.
- Trier la liste des personnes selon le nom et l'identifiant.
- Fournir la possibilité de télécharger ou d'imprimer l'acte de naissance.

- Gestion des enfants :

  - Afficher les informations sur les enfants d'une personne donnée.
  - Ajouter les enfants d'une personne où cas de naissance.

- Gestion des mariages :

  - Afficher le conjoint d'une personne.
  - Déclarer un mariage si les deux personnes ne sont pas encore mariées.
  - Déclarer un divorce.
  - Fournir la possibilité de télécharger ou d'imprimer l'acte de mariage.

- Gestion des décès :

  - Déclarer un décès.
  - Fournir la possibilité de télécharger ou d'imprimer l'acte de décès.

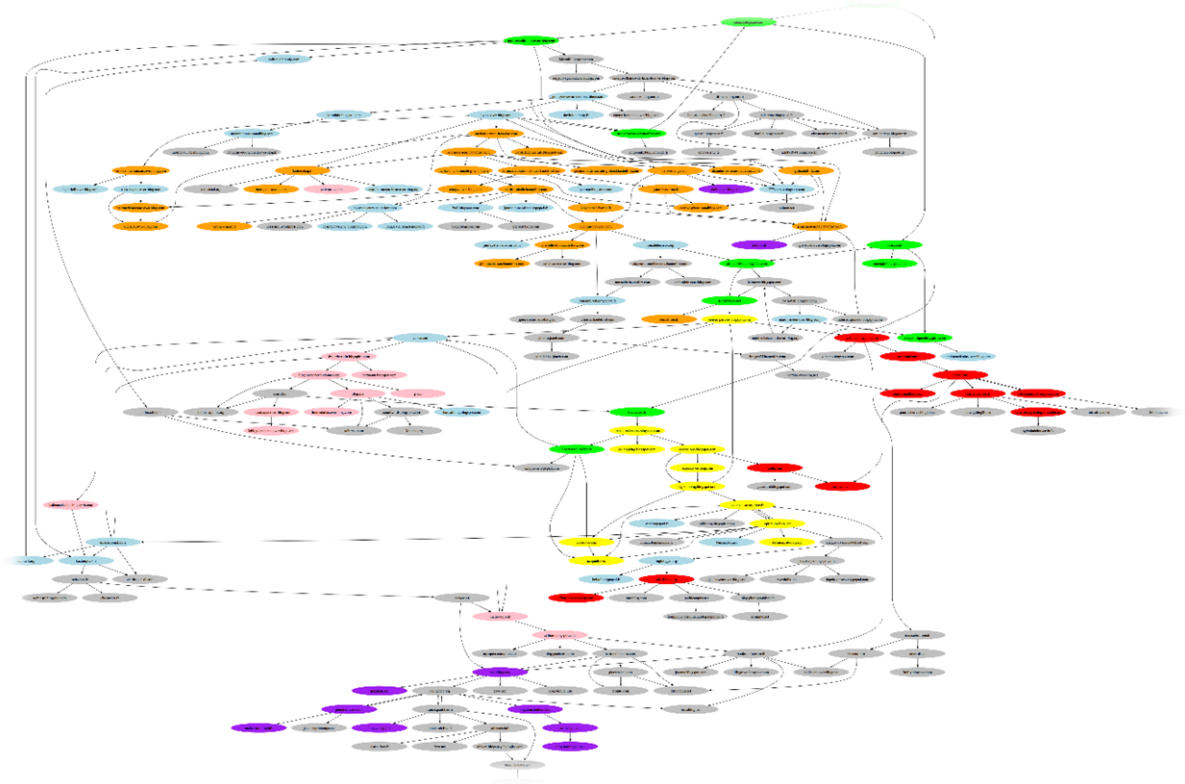
Espace citoyen :

- Vérifier l'identifiant d'une personne ainsi que son nom et son prénom afin d'accéder à son propre espace citoyen.
- Fournir la possibilité de télécharger ou d'imprimer les différents actes d'état civil.

III.2 Conception

a. Structuration des données :

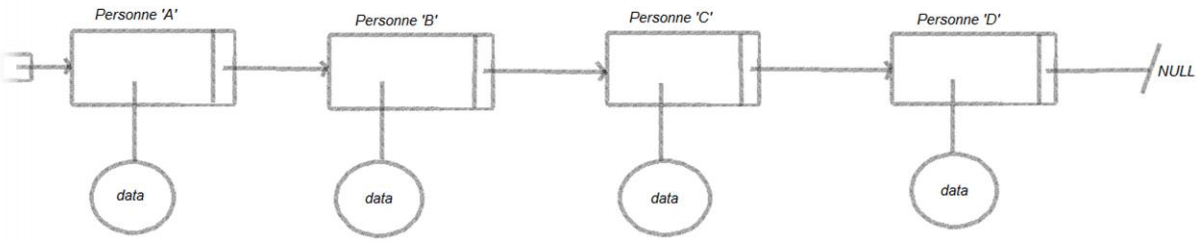
Nous souhaitons mettre en place une application qui gère un ensemble de personne, cela revient à gérer et manipuler des arbres généalogiques où les personnes seront tous reliées entre elles via leurs liens familiaux.



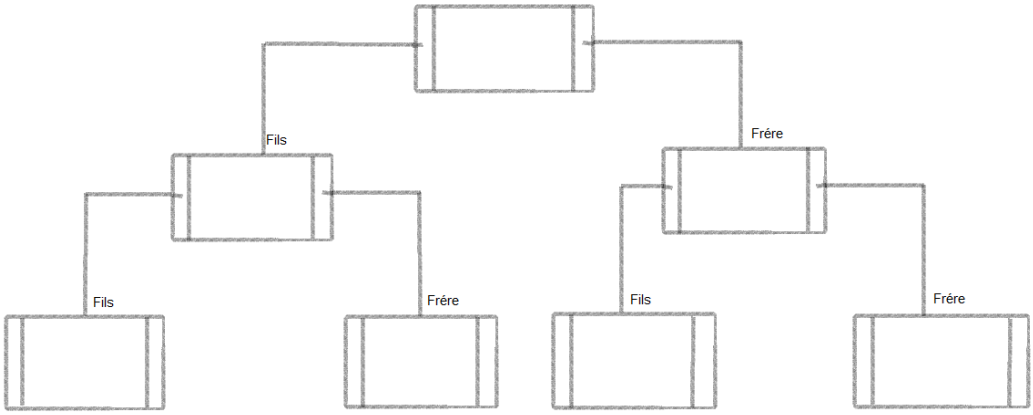
Nous parlerons plutôt de graphe généalogique puisque cette structure n'est pas un arbre au sens informatique du terme.

Donc comment représenter un tel graphe généalogique en programmation ? Nous disposons comme souvent de plusieurs moyens.

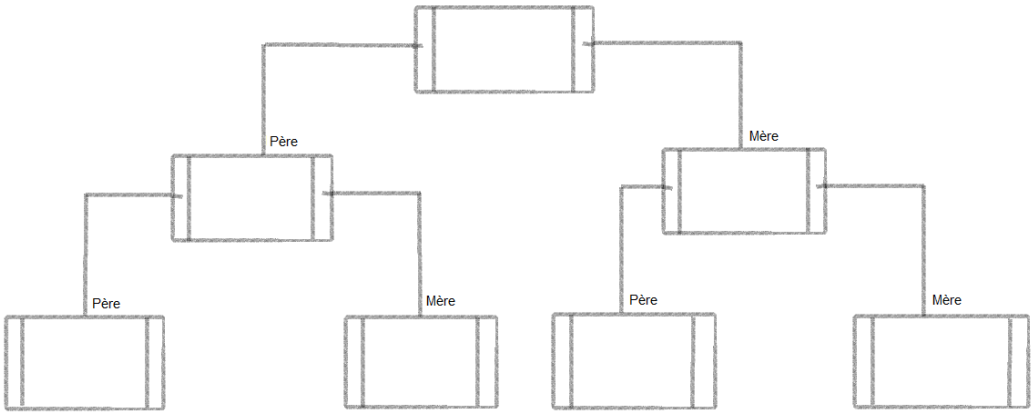
Connaissant les listes chainées, on peut bien s'en sortir en reliant chaque personne avec un autre en introduisant un pointeur :



Cependant cette solution ne permet pas de gérer les relations familiales entre les personnes, pour cela on va améliorer notre conception en ajoutant d'autres éléments.
Chaque personne dispose d'un fils et d'un frère, donc une personne pointe sur son fils ainé et sur son frère ainé.



Concernant les liens de parenté, on va ajouter à notre structure personne les identifiants des parents, l'identifiant du père et de la mère ainsi qu'un identifiant de son conjoint.



Donc la variable de type **personne** sera un pointeur sur une structure pour stocker :

- Son identifiant ;
- Son nom ;
- Son prénom ;
- Sa date de naissance ;
- Son sexe ;
- Indicateur de décès;
- Identifiant du père ;
- Identifiant de la mère ;
- Identifiant du conjoint ;
- Un pointeur sur fils ;
- Un pointeur sur frère ;
- Un pointeur sur suivant ;

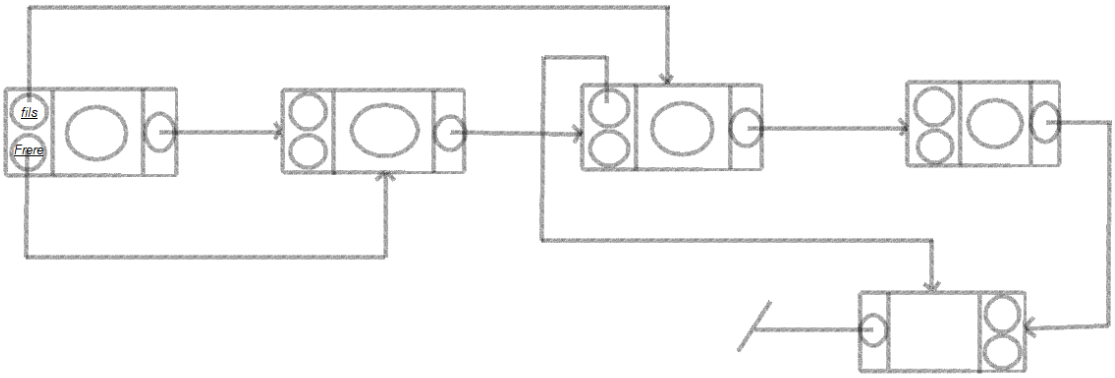
Pour bien organiser notre travail, on a défini une autre structure qui regroupe les informations concernant une personne :

```
/*************************/
/* Structure Data Personne */
/*************************/


typedef struct _data
{
   int id;
   int idPere;
   int idMere;
   int idConjoint;
   char nom[CMAX];
   char prenom[CMAX];
   date dateNaissance;
   char sexe;
   int deces;

}dataPersonne;



typedef dataPersonne* ptrDataPersonne;
```

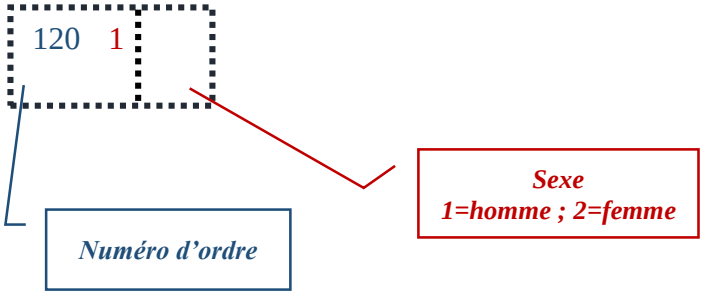
Notre structure devient comme ceci:

```
/**********************/
/* Structure Personne */
/**********************/


typedef struct _personne
{
   ptrDataPersonne data;

   struct _personne* frere;
   struct _personne* fils;

   struct _personne* suivant;

}personne;


typedef personne* listePersonne;
```

Donc on est arrivé à trouver une solution qui consiste à lier toutes les personnes sous forme une simple liste chainée mais en plus de çac chaque personne pointe sur d'autre personne qui ont une relation familiale avec lui à savoir les liens de parenté ou de fraternité.



Lors de l'ajout, on va introduire à chaque personne un numéro qui sert à identifier chaque personne d'une manière unique et il générer automatiquement.
Un identifiant doit obligatoirement terminer par : 1 pour homme ou 2 pour femme, il doit commencer par un numéro d'orde.



Grâce à cet identifiant, on peut facilement déterminer le sexe d'une personne sans accéder à ses données et ça va nous servir beaucoup dans des opérations qui nécessite la vérification du sexe comme le cas de la déclaration du mariage.

b. Description des principales fonctions :

Notre application est basée sur plusieurs fonctions, dans cette partie on va expliquer le fonctionnement de chacune.

- ### Gestion des identifiant :

 Prototype :

int *genererId*(char sexe,int* nbrePersonne)

- Rôle :

Cette fonction génère un unique identifiant qui se compose d'un chiffre qui indique le sexe (1 pour homme et 2 pour femme) et d'un numéro d'ordre.

Pour générer un identifiant, on aura besoin du sexe ainsi que le nombre de personne qui a été récupérer depuis un fichier binaire.

On incrémente le nombre de personne puis on ajoute le chiffre qui représente le sexe, pour le faire on va multiplier le nombre de personne incrémenté par 10 pour le décaler et après on ajoute 1 pour homme et 2 pour femme ça dépend du sexe.

```
/************************/
/* Identifiant personne */
/************************/


int genererId(char sexe,int* nbrePersonne)
{
    int idPersonne;

    (*nbrePersonne)++;
    /* On incremente le nombre de personnes, pour creer un autre id */
    /* puis on ajoute le chiffre de sexe */
    /*  1 : Homme ; 2 : Femme */

    if( sexe == 'M' ) idPersonne = (*nbrePersonne)*10 + 1;
    else
       idPersonne = (*nbrePersonne)*10 + 2;

    return idPersonne;
}
```

- Gestion des dates :

Notre application nécessite la manipulation des dates à savoir la récupération de la date actuelles, la vérification d'une date, la calcule de l'âge d'une personne etc…

Afin de faciliter la gestion de ses dates, on a défini une structure date :

```c
/*****************/
/* Structure Date */
/*****************/

typedef struct _date
{
    int jour;
    int mois;
    int annee;

}date;
```

- Prototypes :

  date *DateActuelles*()

- Rôle :

Cette fonction nous permet de savoir la date actuelle qui va nous servir pour calculer l'âge d'une personne et d'indiquer la date de délivrance des actes d'état civil etc…

```c
/*******************************/
/* Récupérer la date actuelles */
/*******************************/

date DateActuelles()
{
    date  t;
    time_t temps;

    time(&temps);
    /* t contient maintenant la date et l'heure courante */

    struct tm d;

    d = *localtime(&temps);
    /* décomposer la date 'temps' en année, mois, jour */

    t.jour = d.tm_mday;
    t.mois = d.tm_mon+1;
    t.annee = d.tm_year+1900;

    return t;

}
```

- Prototype :

  int *verifierDate*(date d);

- Rôle :

Cette fonction vérifie si une date de naissance est valide.

Une date de naissance ne doit pas dépasser la date actuelle.

Le mois doit être inférieur à 12 et supérieur ou égale à 1.

Le jour doit être compris entre 1 et 31 pour les mois 1 3 5 7 8 10 12 et entre 1 et 30 pour les mois 4 6 9 11.

Si le mois de naissance est Février, le jour doit être compris entre 1 et soit 28 ou 29 ça dépend si l'année est bissextile ou pas.

On dit qu'une année est bissextile s'il est divisible par 4 et non pas par 100.

-Voir le code source en Annexe-

- Prototype :

int *verifierAge*(listePersonne debut,date dateFils, int idPere);

- Rôle :

Cette fonction vérifie si la date de fils est valide en le comparant avec la date du père. Elle a été créée pour nous servir lors de l'ajout d'un enfant, son âge doit être bien évidemment inférieur à l'âge de ses parents.

```
/*****************/
/* Age personne */
/*****************/
int verifierAge(listePersonne debut,date dateFils, int idPere)
{
    int agePere,ageFils;

    listePersonne pere= chercherPersonneById( debut, idPere);

     //Calculer l'âge du père:
    agePere=agePersonne(pere);

     //Calculer l'âge du fils:
    ageFils=calculerAge(dateFils);

     //Si l'âge est invalid, on retourne 0 sinon 1
    return agePere>ageFils?1:0;


}
```

- Prototype :

int *calculerAge*(date date_naissance);

- Rôle:

Cette fonction retourne calcule l'âge d'une personne.

```
/***************/
/* Calcule Age */
/***************/

int calculerAge(date date_naissance)
{
    int age;

    //On récupére la date actuelle:
    date date_actuelle = DateActuelles();

    //L'âge est la différence entre l'année actuelle et l'année de naissance:
    age = date_actuelle.annee - date_naissance.annee;

    // On décremante l'âge si le mois et le jour de naissance
    //  sont supérieurs au mois et le jour actuelle:
    if( (date_actuelle.mois < date_naissance.mois)
        || ( (date_actuelle.mois == date_naissance.mois)
          && (date_actuelle.jour < date_naissance.jour) ) )

        age--;

    return age;

}
```

- Prototypes :

    int *agePersonne*(listePersonne p);

- Rôles :

Retourne l'âge d'une personne en appelant la fonction calculerAge.

```
/****************/
/* Age personne */
/****************/

int agePersonne(listePersonne p)
{
    return calculerAge(p->data->dateNaissance);

}
```

- La recherche d'une personne :

- Prototypes : void *chercherPersonne*(listePersonne debut, int choix);
- Roles :

L'utilisateur à le choix entre la recherche par identifiant, nom ou prénom. Pour la recherche par identifiant, on récupère la personne en utilisant une autre fonction qui prend en paramètre son identifiant, le prototype de cette fonction est la suivante :

listePersonne *chercherPersonneById*(listePersonne debut,int id)

Cette fonction parcours toute la liste en comparant chaque identifiant d'une personne avec id de son paramètre, après il retourne son adresse.

-Voir le code source en Annexe-

• Les parents d'une personne :

- Prototype : void *nomPereMere*(listePersonne debut,int id)
- Rôles :

Cette fonction prend en paramètre un identifiant d'une personne qu'on veut chercher ses parents, tout d'abord on récupère l'adresse de cette personne par la fonction :

listePersonne *chercherPersonneById*(listePersonne debut,int id)

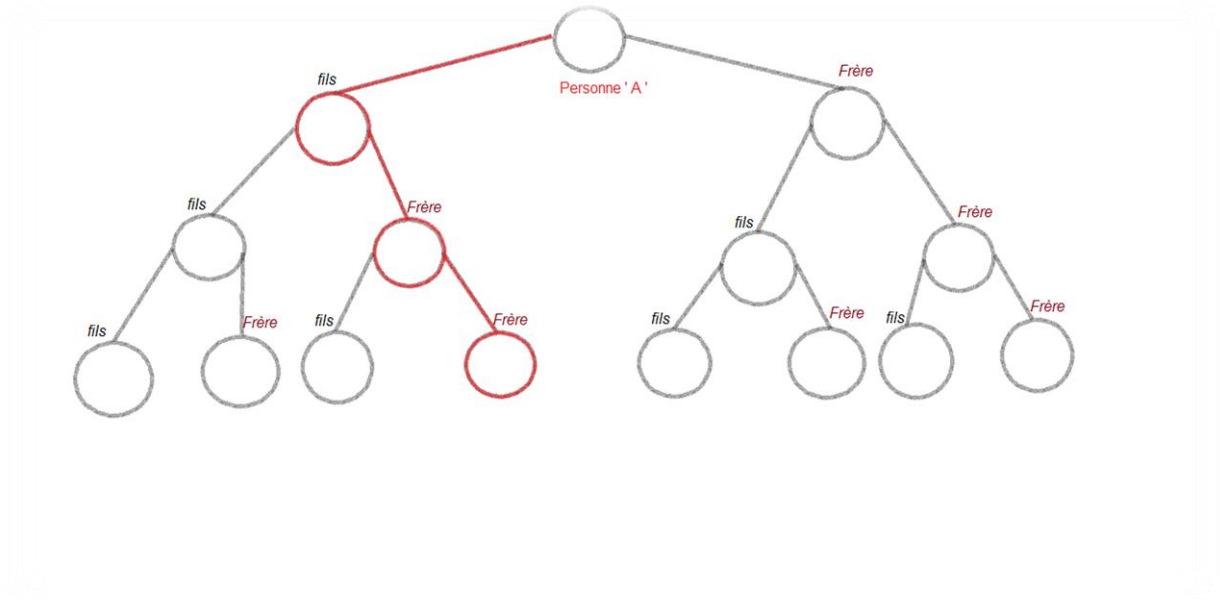
Et on récupère directement les identifiants de ses parents depuis les champs introduites dans la structure Personne. Et afin d'afficher leurs nom, on doit chercher leurs adresses en utilisant la même fonction précédente.

-Voir le code source en Annexe-

• Les enfants d'une personne :

- Prototype : void *afficherEnfants*(listePersonne debut,int id)
- Rôles :

Chaque personne possède un pointeur sur son fils ainé et un autre sur son frère ainé. Donc les frères de son fils ainé sont ses enfants. La recherche des enfants d'une personne donnée revient à parcourir l'arbre qui a comme racine la personne concernée en récupérant son fils ainé, fils gauche, puis on passe au fils droits de ce dernier qui est son frère ainé, par conséquent c'est parmi les enfants de la racine. Ainsi de suite, on parcourt cette arbre de cette manière jusqu'à ce qu'on récupère tous les enfants de la racine.



Code :

```
void afficherEnfants(listePersonne debut,int id)
{
     //Récupèrer l'adresse de la personne:
    listePersonne pers = chercherPersonneById(debut,id);

     //Vérifier si cette personne existe ou pas:
    if( pers == NULL )
      printf("\n - ID n'existe pas. \n\n");
    else
    {
        //Si cette personne n'a pas encore d'enfants:
       if( pers->fils == NULL )
         printf("\n - %s %s n'a pas encore d'enfants.\n",pers->data->nom,pers->data->prenom);
       else
       {
          //Si la personne a au moins un enfant:
         printf("\n\t - - -  LES ENFANTS DE %s %s - - - \n\n",pers->data->nom,pers->data->prenom);

          //pers->fils représente le fils ainé
          //les enfants de pers sont les fréres de pers->fils
         afficherFrere(pers->fils);
       }
    }

}
```

```
void afficherFrere(listePersonne r)
{
    if(r != NULL)
    {
         //Afficher les informations d'une personne donnée:
        afficherPersonne(r);

         //ainsi que celles de son frére:
        afficherFrere(r->frere);
    }

}
```

- L'ajout d'une personne :

L'utilisateur saisie les données d'une personne à ajouter, à savoir :

Son prénom, son sexe, l'identifiant du père et celui de la mère et sa date de naissance.

On n'aura pas besoin de saisir son nom, puisque c'est le même que celui de son père, on va l'affecter automatiquement. Chaque personne possède un identifiant unique générer automatiquement en utilisant la fonction *genererId()*.

Lors de l'ajout, les parents de la personne à ajouter doivent être déjà inscrits, on doit vérifier l'existence des parents avant toutes choses.

```
/************************/
/* Identifiant personne */
/************************/


int genererId(char sexe,int* nbrePersonne)
{
    int idPersonne;

    (*nbrePersonne)++;
    /* On incremente le nombre de personnes, pour creer un autre id */
    /* puis on ajoute le chiffre de sexe */
    /*  1 : Homme ; 2 : Femme */

    if( sexe == 'M' ) idPersonne = (*nbrePersonne)*10 + 1;
    else
       idPersonne = (*nbrePersonne)*10 + 2;

    return idPersonne;
}
```

On doit vérifier aussi que sa date de naissance saisie est bien valide ainsi que son âge qui doit être bien évidement inférieurs à celui de ses parents.

```
//On vérifier la date de naissance
do{

        printf("\n  - DATE DE NAISSANCE : ");
        scanf("%d/%d/%d",&dateNaissance.jour,&dateNaissance.mois,&dateNaissance.annee);

         /* Si la date est invalid */
        testDate = verifierDate(dateNaissance);

         /* Si l'age de fils est plus grand que celui du pere */
        testAge = verifierAge(listeP,dateNaissance,idPere);

  }while( testDate == 0 || testAge==0 );
```

- Prototype :

listePersonne *addNewPersonne*(listePersonne debut, int* nbrePersonne, char* prenom,date date_naissance, char sexe, int idPere, int idMere)

- Rôles :

On génère l'identifiant de la personne à ajouter par genererId( ), puis on crée un nœud qui va contenir toutes les informations saisies de cette personne. On lie ce nœud dans la dernière position de la liste des personnes. Maintenant on n'a pas pris en compte ses relations familiales, pour le faire on lie cette personne avec ses parents :

- On cherche son père et sa mère.
- Cette personne est un nouvel enfant, on va l'ajouter parmi les enfants de ses parents.
- On remplit le champ fils des parents par ce nouvel enfant s'il est le seul enfant qu'ils ont.
- Si non, on parcourt l'arbre fils/frère d'une manière à trouver le plus petit enfant puis on ajoute la personne on vient de créer comme son frère.

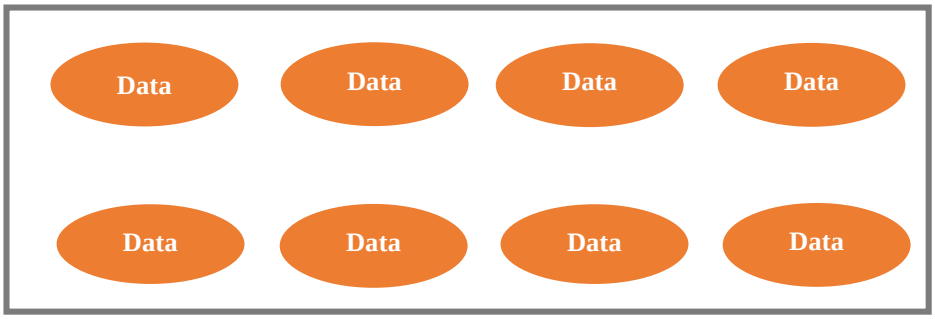
-Voir code source en Annexe -

- Sauvegarder les données :

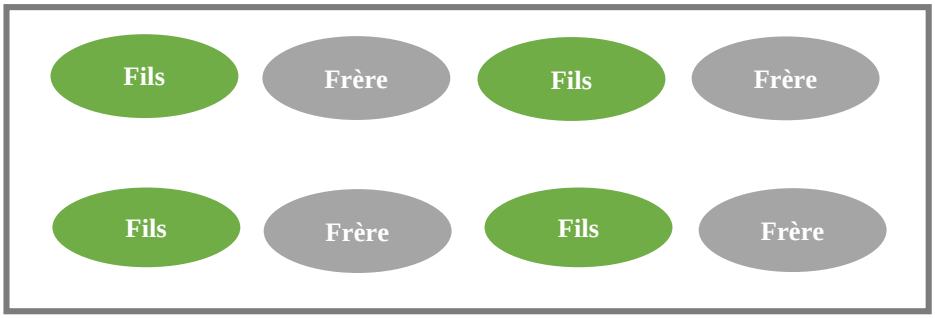
- Prototypes : void *sauvegarder* (listePersonne debut,int nbrPersonne);
- Rôles :

On sauvegarde les données de la liste dans le fichier binaire *listePersonne.dta*.
Apres on sauvegarde les identifiants du fils et du frère de chaque personne dans un autre fichier binaire *relation.dta*, pour bien garantir les liens familiales entre les personnes.

- Fichier listePersonne.dta :



- Fichier relation.dta :



-Voir le code source en Annexe -

- ## Charger les données :

- Prototypes : void charger (listePersonne debut,int nbrPersonne);
- Roles :

On charge premièrement la liste d'après les données du fichier *listePersonne.dta* puis on parcourt cette liste et on remplit les champs mère / père / fils / frère / depuis le fichier *relation.dta.*

-Voir le code source en Annexe -

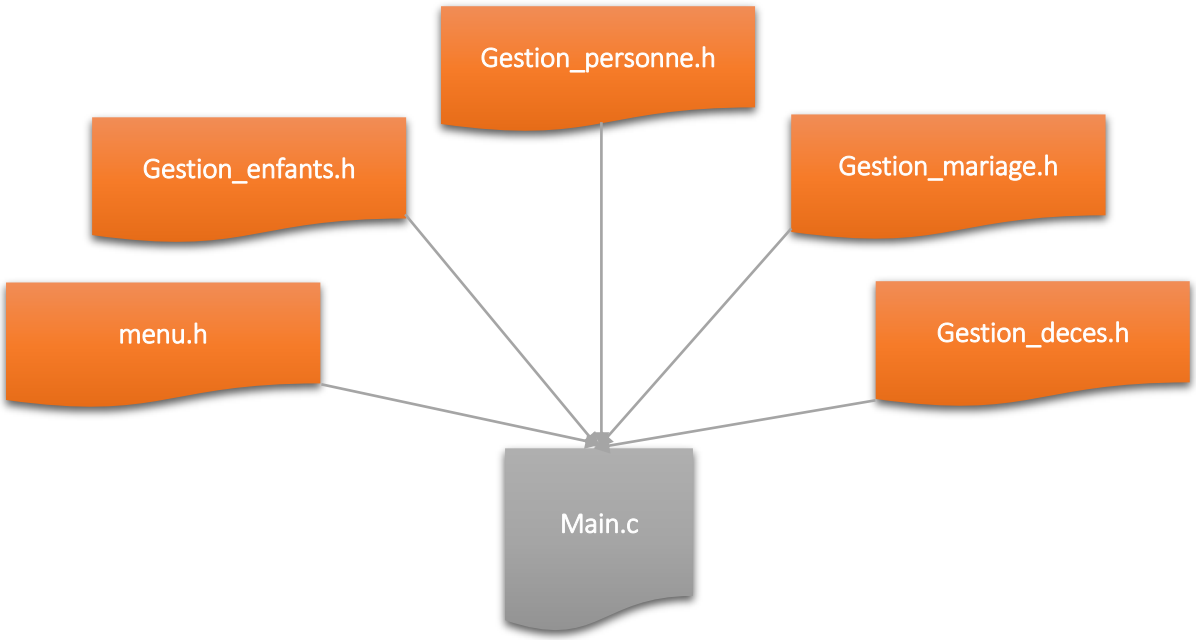
CHAPITRE IV Réalisation et mise en œuvre

IV.1 Les modules de l'application

Une application informatique, dès qu'elle dépasse une certaine de lignes de code, doit impérativement être structurée sous peine de se révéler difficile à mettre au point, difficile à modifier, donc inutilisable à terme. Pour cela on a découpé notre application en plusieurs modules, chacun d'eux a un rôle précis qui regroupe des fonctions correspondant à des fonctionnalités proches.

On a défini les modules suivants :

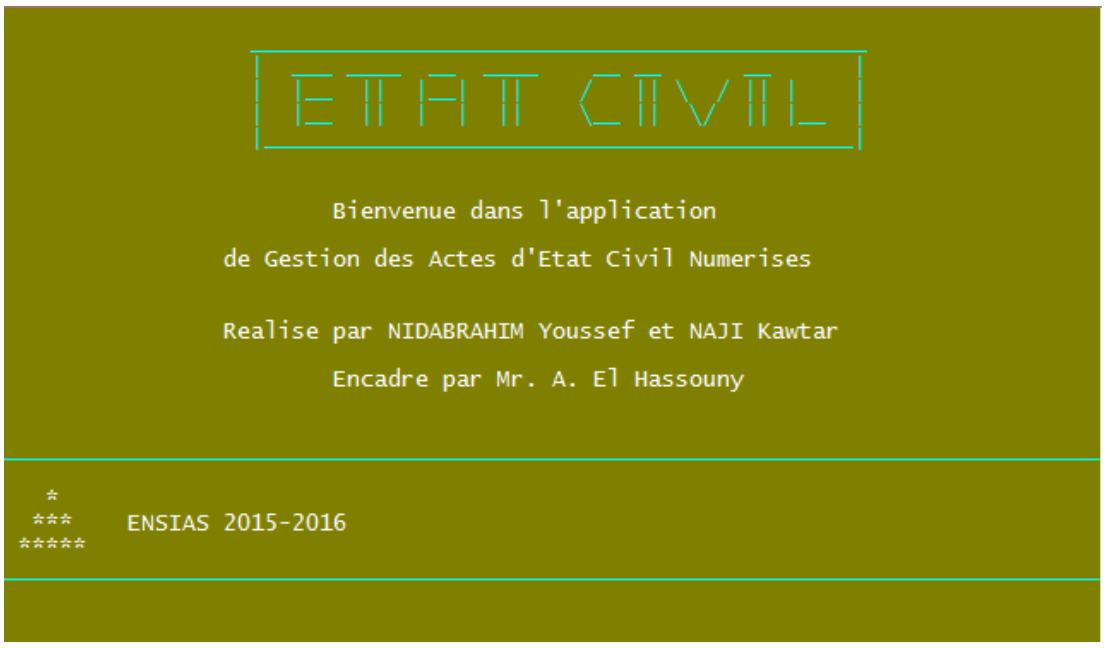
- Un module spécialisé dans la gestion des personnes, nommons-le *gestion_personne.c* celui-ci contient des fonctions de manipulation d'une personne telles que la recherche, la modification des informations d'une personne, l'ajout d'une personne ainsi que le tri etc…
- Un module spécialisé dans les fonctions qui traitent les enfants d'une personne *gestion_enfants.c.*
- Un module contenant les opérations de manipulation des relations de mariage entre deux personnes *gestion_mariage.c.*
- Un module contenant des fonctions qui gèrent les décès *gestion_deces.c.*
- Un module pour les menus *menu.c.*
- Un module contenant le programme principal *main.c.*



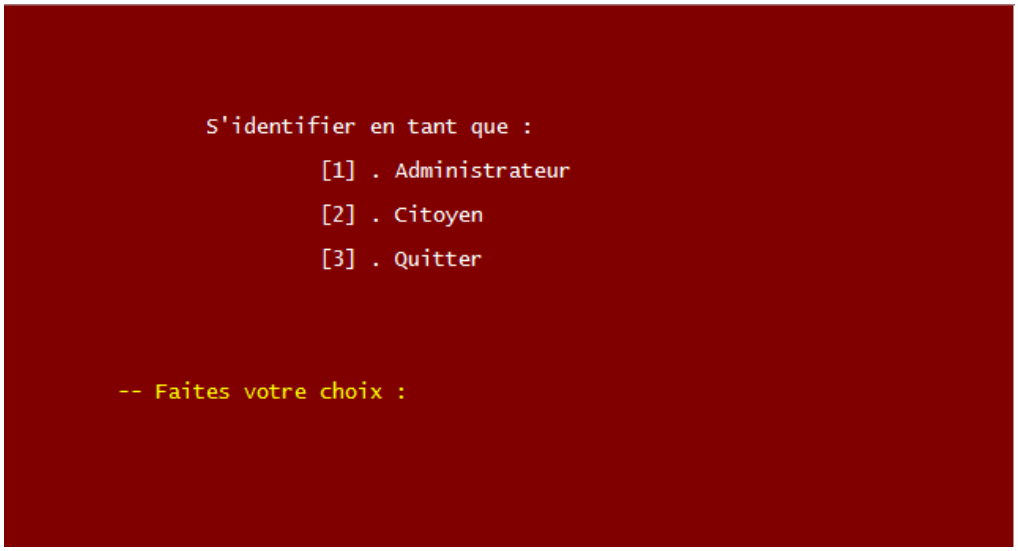
IV.2 Manuel de l'utilisation

L'objectif de cette partie est de présenter un scénario d'utilisation de notre application afin de donner un aperçu de l'interface homme / machine.

Un simple clic sur l'icône de l'application, le premier écran apparait.



En appuyant sur < Entrer >, une autre interface s'apparait qui demande à l'utilisateur de choisir un mode d'authentification, soit comme étant un agent ou bien un citoyen afin d'accéder à son propre espace :



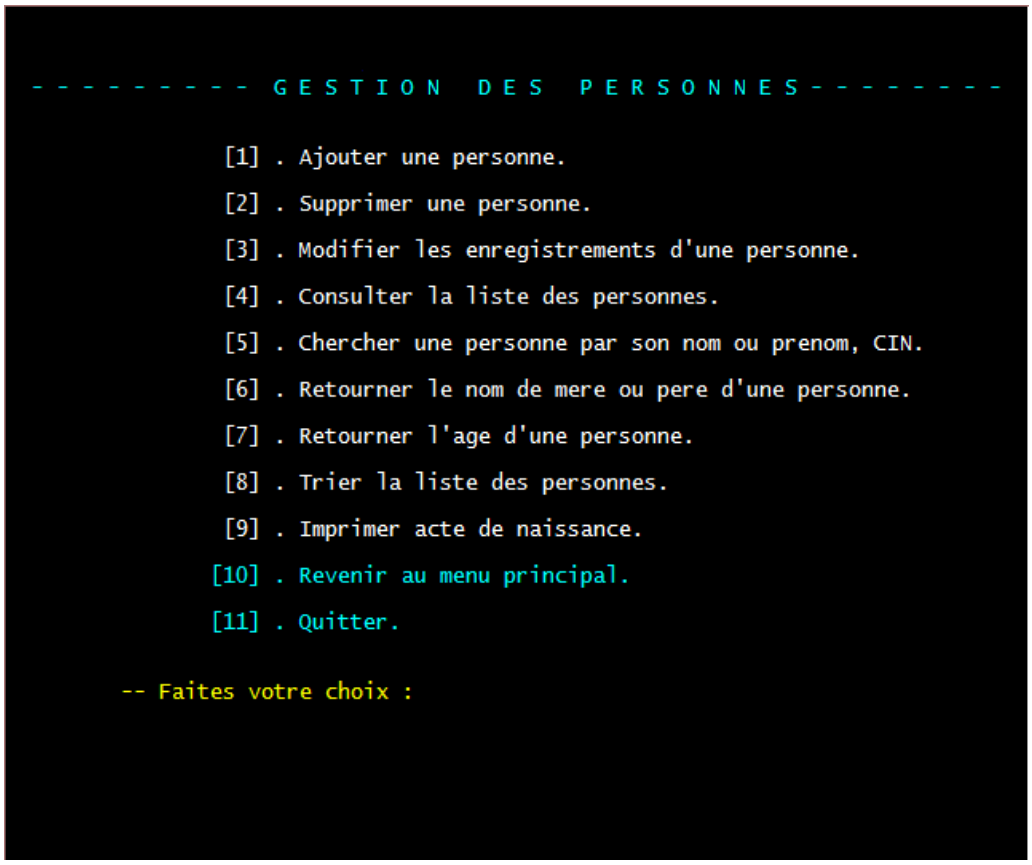
S'il a choisi de s'identifier comme étant un administrateur, il doit saisir login et le mot de passe afin d'accéder à son espace administrateur.

Vous remarquez que le mot de passe est bien caché pour bien garantir la sécurité.

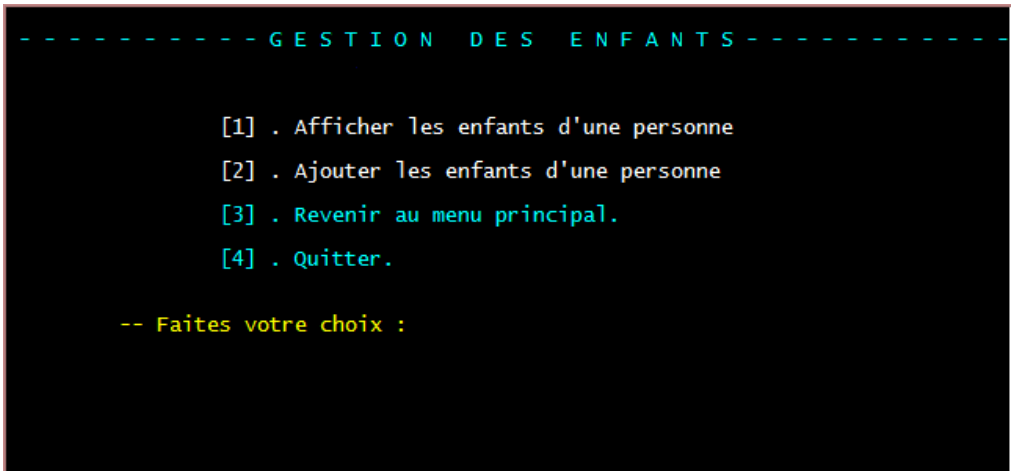
Dès que l'agent entre un login et un mot de passe valide, un menu principal s'ouvre en lui proposant des différents modules qui peut choisir et qui convient à ce qu'il veut faire :

Chacun de ses choix nous emmène à des sous menus qui donnent à l'agent la possibilité d'effectuer plusieurs opérations.

En choisissant la gestion des personnes, une liste des opérations possible que l'agent peut effectuer s'apparait dans la console :



L'agent a la possibilité d'afficher les enfants d'une personne donnée, d'ajouter plusieurs enfants :



L'agent a la possibilité d'ajouter une personne en choisissant le premier choix :
Il doit indiquer le nombre de personne qui veut saisir. Vous remarquez que 1221 n'est pas été prise car il n'existe aucune personne avec cet identifiant.

```
- - - - - - - - -  - - - - - AJOUTER DES PERSONNES - - - - - - - - - - -  - - - -

- Combien de personne voulez vous saisir : 1

- - - - - - - - - - - - - - - LA SAISIE DES DONNEES - - - - - - - - - - - - - - -

 - PRENOM : nisrine
 - SEXE : F
 - ID DU PERE : 1221
 - ID DU PERE : 391
 - ID DE LA MERE : 332
 - DATE DE NAISSANCE : 10/12/2014

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

L'agent peut également chercher une personne soit par son identifiant, par son nom ou bien par son prénom :

```
- - - - - - - - - - - -  RECHERCHE PAR NOM / PRENOM / ID  - - - - - - - - - - - -

                [1].Recherche par l'identifiant
                [2].Recherche par le Nom
                [3].Recherche par le prenom
        - Faites votre choix : 1
```

```
- - - - - - - - - - - -  RECHERCHE PAR NOM / PRENOM / ID  - - - - - - - - - - - -

                [1].Recherche par l'identifiant
                [2].Recherche par le Nom
                [3].Recherche par le prenom
        - Faites votre choix : 1
-- Saisissez un identifiant : 982

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - ID :  982
- - NOM :  ZAHIR
- - PRENOM :  nisrine
- - DATE DE NAISSANCE :  10 / 12 / 2014
- - SEXE :  Feminin
- - AGE :  1 ans
- - DECES :  Vivant
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

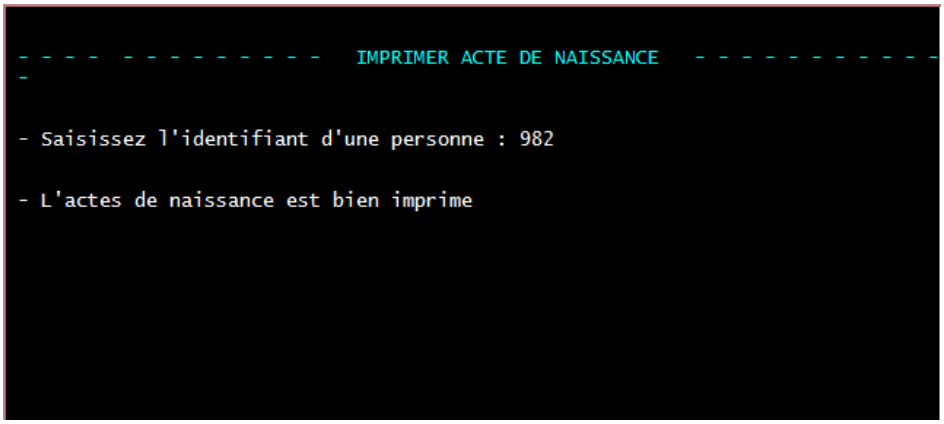
Il peut également savoir le nom des parents d'une personne donnée en saisissant son identifiant :

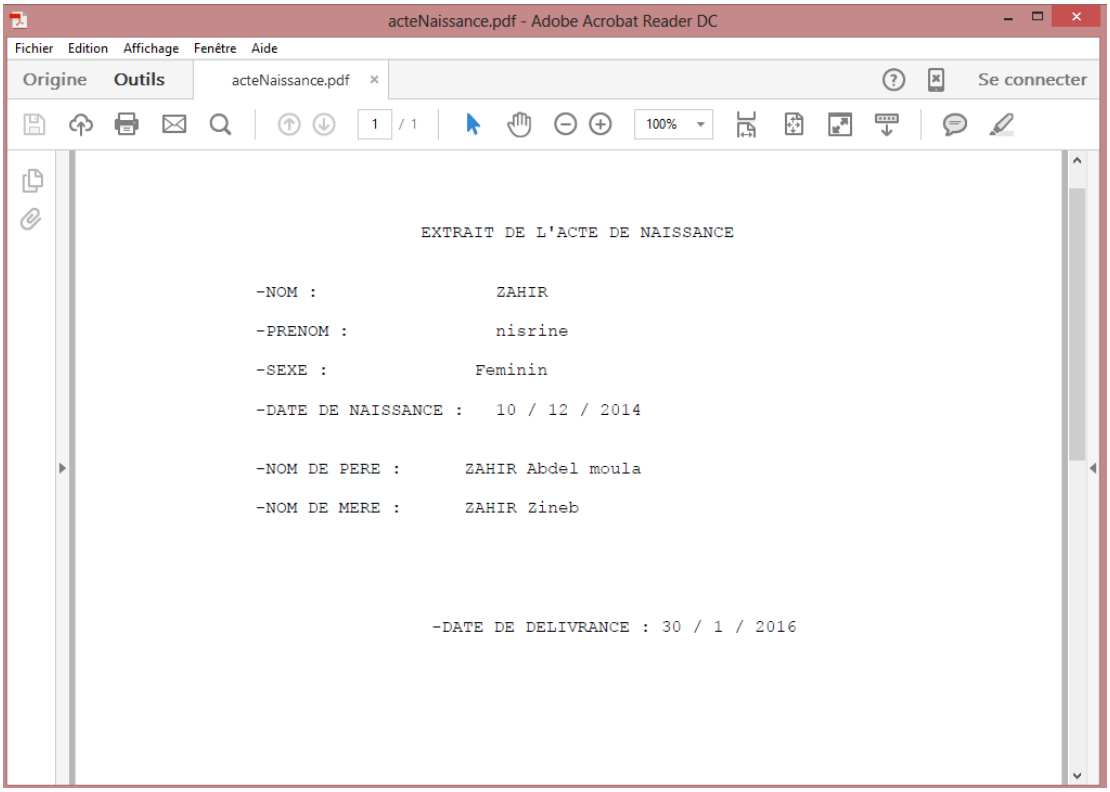
```
- - - - - - -   RETOURNER LE NOM DE MERE OU PERE D'UNE PERSONNE   - - - - - - -

- Saisissez l'identifiant d'une personne : 982
- Le Pere de ZAHIR nisrine est ZAHIR Abdel moula

- La Mere de ZAHIR nisrine est ZAHIR Zineb
```

Ainsi que son âge :

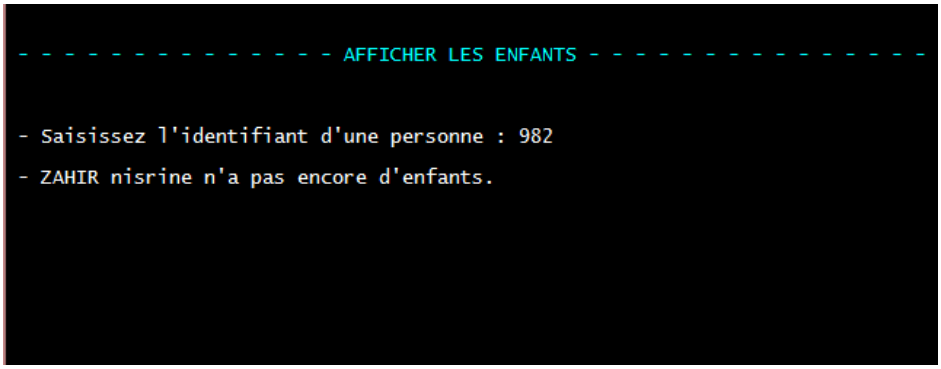
```
- - - - - - - - - - - -   RETOURNER L'AGE D'UNE PERSONNE  - - - - - - - - - - -

- Saisissez l'identifiant d'une personne : 241

- FARIS abdellah , qui a comme identifiant 241, a 53 ans
```

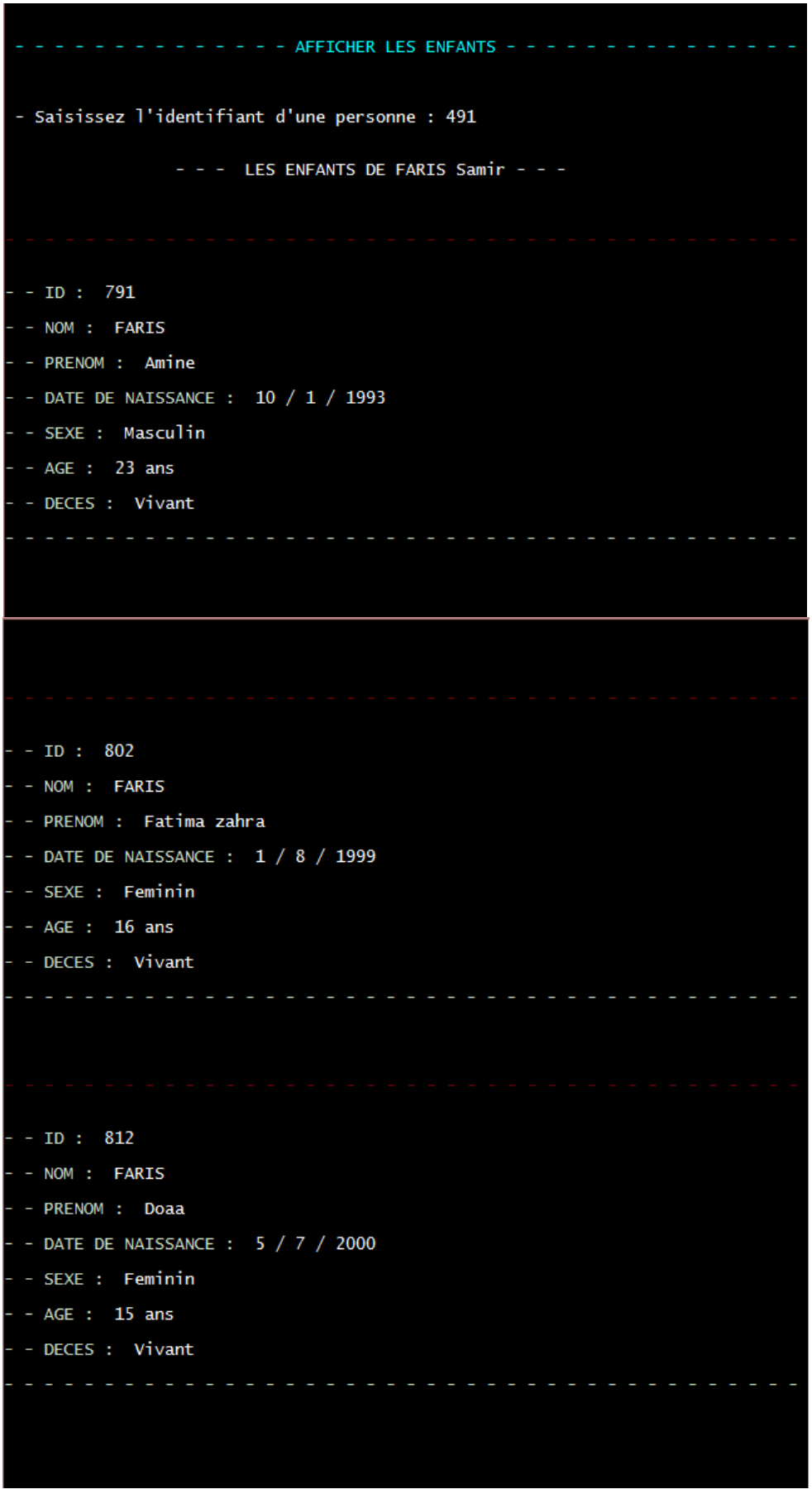
Il a aussi la possibilité d'imprimer ou de télécharger l'acte de naissance d'une personne en format PDF :



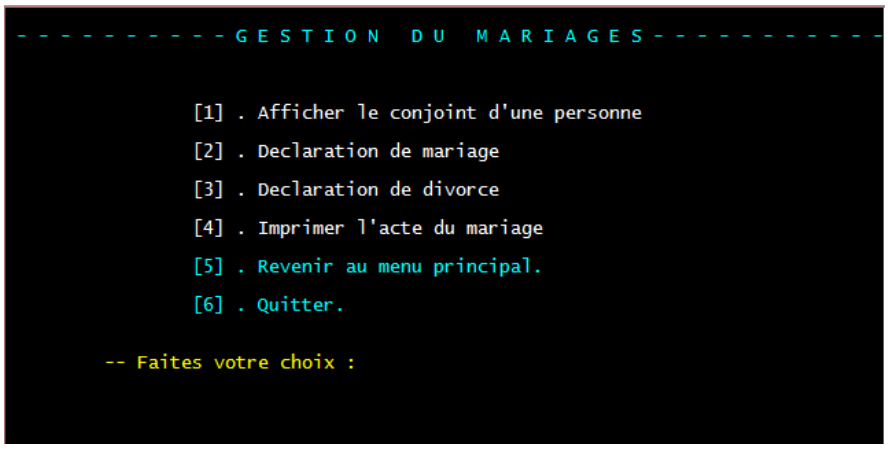


Il peut afficher les enfants d'une personne en indiquant son identifiant, si la personne n'a pas encore d'enfants un message s'affiche lui informe ça :



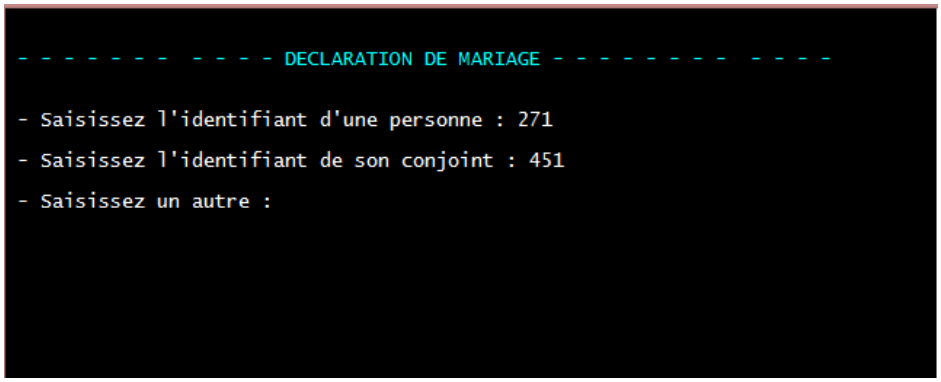

- - - - - - - - - - - - - - - AFFICHER LES ENFANTS - - - - - - - - - - - - - - -
- Saisissez l'identifiant d'une personne : 491
- - - LES ENFANTS DE FARIS Samir - - -
- - ID : 791
- - NOM : FARIS
- - PRENOM : Amine
- - DATE DE NAISSANCE : 10 / 1 / 1993
- - SEXE : Masculin
- - AGE : 23 ans
- - DECES : Vivant
- - ID : 802
- - NOM : FARIS
- - PRENOM : Fatima zahra
- - DATE DE NAISSANCE : 1 / 8 / 1999
- - SEXE : Feminin
- - AGE : 16 ans
- - DECES : Vivant
- - ID : 812
- - NOM : FARIS
- - PRENOM : Doaa
- - DATE DE NAISSANCE : 5 / 7 / 2000
- - SEXE : Feminin
- - AGE : 15 ans
- - DECES : Vivant

En choisissant la gestion des mariages dans le menu principal :



L'agent a la possibilité d'afficher le conjoint d'une personne :

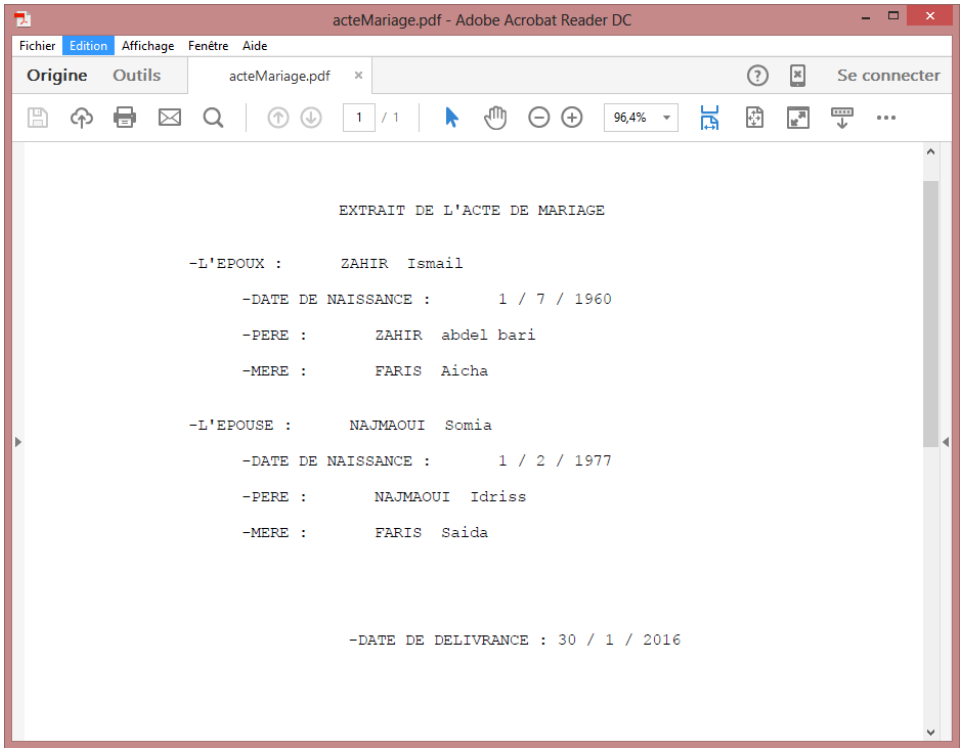
Comment il peut déclarer un mariage en indiquant les identifiants d'un couple de deux personnes, vous remarquez ici que 451 n'est pas accepté car il n'est pas valide puisque on a ici un couple de deux hommes :



```
- - - - - - - - - - - - DECLARATION DE MARIAGE - - - - - - - - - - - - - -

- Saisissez l'identifiant d'une personne : 271
- Saisissez l'identifiant de son conjoint : 451
- Saisissez un autre : 452
```

```
- - - - - - - - - - - - - - -   IMPRIMER ACTE DE MARIAGE   - - - - - - - - - - - - - -

- Saisissez l'identifiant d'une personne : 271

- L'actes de mariage est bien imprime, vous pouvez le recuperer.
```

L'agent peut récupérer facilement l'acte du mariage :



Il peut aussi déclarer un divorce :

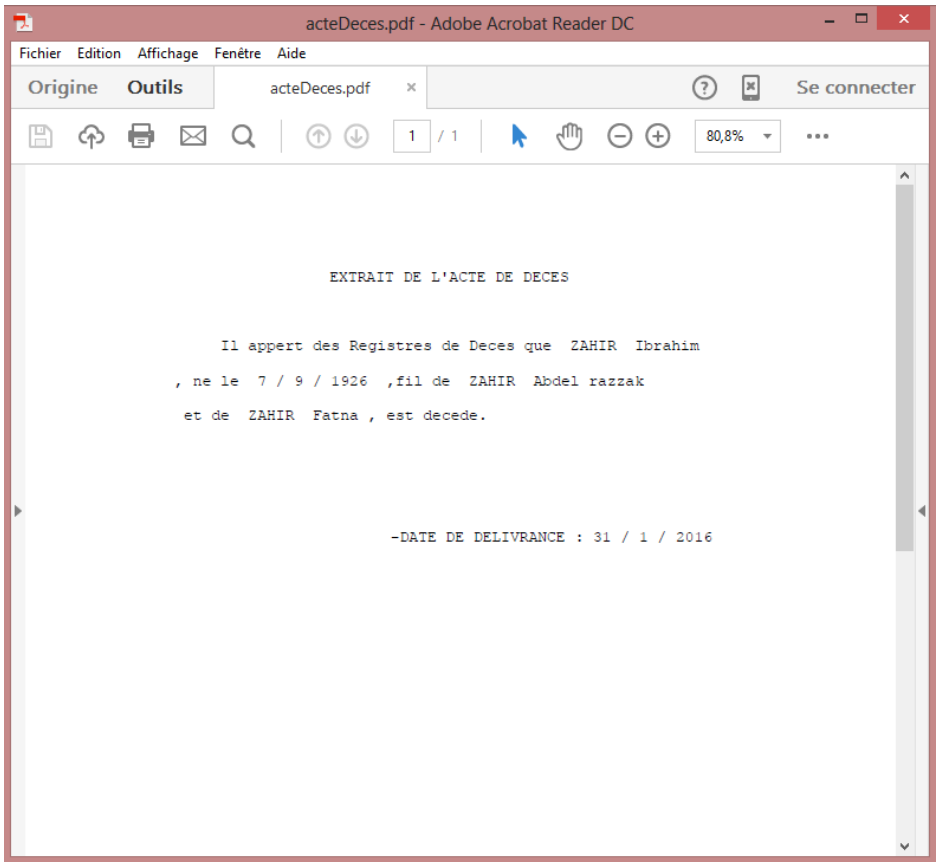
```
- - - - - - - -  - - - - DECLARATION DE DIVORCE - - - - - - - -  - - - -

- Saisissez l'identifiant d'une personne : 271
- Saisissez l'identifiant de son conjoint : 452
```

Et pour manipuler tout ce qui concerne le décès, on choisit la gestion des décès :

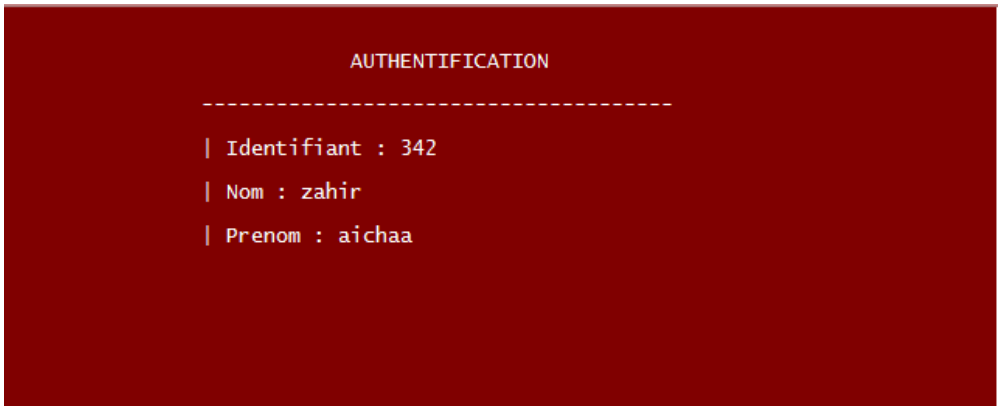
```
- - - - - - - - - - - G E S T I O N   D U   D E C E S - - - - - - - - - - -

                [1] . Declaration de deces
                [2] . Imprimer l'acte de deces
                [3] . Revenir au menu principal.
                [4] . Quitter.

        -- Faites votre choix :
```

Au cas de décès, l'agent peut bien récupérer l'acte de décès d'une personne :

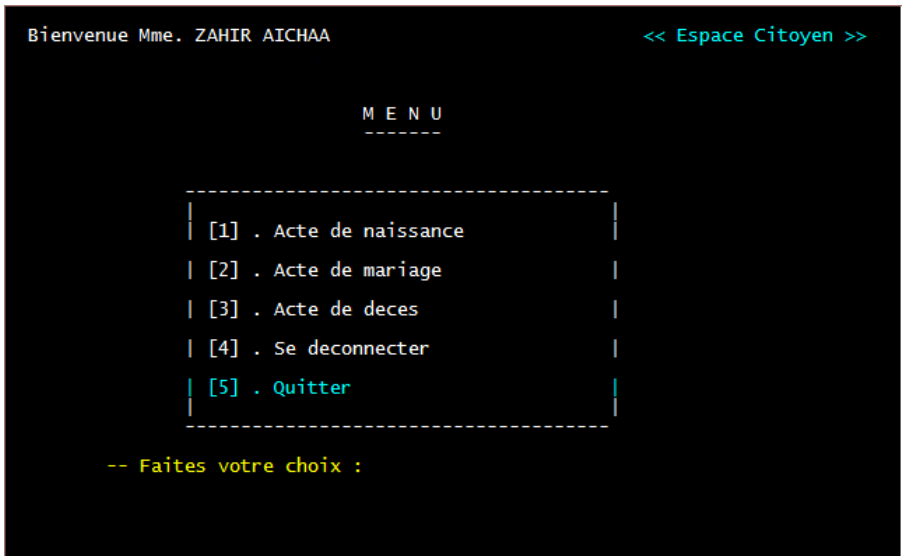
```
- - - - - - - - - - -   IMPRIMER ACTES DE DECES   - - - - - - - - - - - -

- Saisissez l'identifiant de personne : 111

- L'actes de deces est bien imprime, vous pouvez le recuperer.
```



N'importe quel utilisateur peut accéder à son espace citoyen en indiquant son identifiant, son nom et son prénom :



Lorsque les données sont correctes, un menu s'apparait qui permet à l'utilisateur d'avoir ses actes d'état civil à savoir l'acte de naissance, l'acte du mariage ainsi que l'acte de décès :


Bienvenue Mme. ZAHIR AICHAA
<< Espace Citoyen >>
M E N U
[1] . Acte de naissance
[2] . Acte de mariage
[3] . Acte de deces
[4] . Se deconnecter
[5] . Quitter
-- Faites votre choix :

Conclusion et Perspective

L'objectif visé à travers ce rapport est de présenter l'application réalisée qui consiste à concevoir et à développer une application en langage C qui permet de gérer des actes d'état civil numérisés et l'impression de ceux-ci.

Au cours de ce travail, on a consacré, dans un premier temps, nos réflexions à l'étude de l'existant et la spécification des besoins. Cette étude nous a permis de déterminer les grands axes que nous allons suivre pour concevoir notre solution. Une phase de conception détaillée à précéder l'implémentation de ce projet. En effet, on a pu développer une application qui répond aux exigences soulignés pendant l'analyse et la conception.

Le fait de développer une application fonctionnelle, codée en langage C est quelque chose de vraiment motivant, il nous a fallu améliorer nos techniques de documentation, prendre le temps pour les décisions importantes et bien sur tout faire pour favoriser le travail d'équipe et donc notre efficacité.

Le cahier des charges est finalement atteint, mêmes si certaines fonctionnalités n'ont pas pu être implémentées, faute de temps par exemple l'implémentation d'une interface graphique.

Enfin, nous avons éprouvé du plaisir de travailler ensemble, chacun s'étant impliqué dans ce projet. Le travail en équipe, lorsqu'il est efficace, permet de pousser un groupe vers le haut, et au final, chacun retrouve dans l'application une partie de son travail, ce qui constitue la satisfaction du travail accompli.

Annexe

La fonction VérifierDate ( )

La fonction ChercherPersonne ( )

La fonction nomPereMere ( )

La fonction addNewPersonne ( )

La fonction sauvegarder ( )

La fonction charger ( )