ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES - RABAT

Rapport de Projet de programmation C : Gestion d'une société d'Intérim

Réalisé par :

Abir BENDOUDA

Abderrahman BEN SALAH

Encadré par :

Pr. Abdellatif EL FAKER

Année universitaire 2020/2021

Remerciements

Ces remerciements vont tout d'abord au corps professoral et administratif de l'*Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes* (ENSIAS), pour la richesse et la qualité de leur enseignement. Nous adressons nos plus sincères remerciements à notre enseignant et notre encadrant : Pr. EL FAKER Abdellatif pour son suivi et toutes les informations qu'ils nous a donné non seulement durant cette période d'encadrement, mais depuis la première séance du cours de la structure de données, nous sommes vraiment reconnaissants pour la connaissance que nous avons obtenu grâce à lui. Nous sommes très reconnaissantes pour la confiance et le soutien permanent, ainsi que les compétences scientifiques et pédagogiques qui nous a permis de mener à bien ce projet. Nous remercions également les membres de jury qui nous ont honoré en acceptant de juger notre modeste travail et l'enrichir de leurs remarques et critiques. Nous vous prions d'agréer l'expression de toute notre profonde reconnaissance.

Résumé

Les sociétés intermédiaires dans nos jours jouent un rôle très important entre les candidats cherchant une offre d'emploi et les entreprises cherchant le candidat adéquat à cette offre.

La phase du traitement des demandes soit celles des entreprises ou bien celles des candidats devient plus délicate avec l'augmentation du nombre des demandes, et pour faciliter cette tâche, on pensait à produire un programme permettant une gestion plus fluide et plus aisée.
Mots-clés : Gestion, candidats, offre d'emploi, recrutements, entreprises.

Abstract

Intermediary companies these days play a very important role between candidates looking for a job offer and companies looking for the right candidate for that job.
The processing phase of requests, either from companies or from candidates, becomes more delicate with the increase in the number of requests, and to facilitate this task, it was thought to produce a program allowing more fluid and easier management.
Keywords : Management, candidates, job offer, recruitments, companies.

Table des Matières

Table des figures

Liste des abreviations

ENSIAS *Ecole Natio-nale Supérieure d'Informatique et d'Analyse des Systèmes*

Introduction

ManPower est une société intermédiaire entre les candidats cherchant une offre d'emploi et les entreprises cherchant le candidat adéquat à cette offre.

Ce projet C a pour but de produire un programme d'environ 500 lignes de code afin de valider les compétences acquises pendant les cours : « Algorithmique », « Technique de programmation » et « Structures de données ».

Ce programme permet de garantir une bonne gestion de ces appels d'emploi et les cv des candidats. Ce rapport englobera deux parties essentiel. La première partie sera pour l'étude théorique qui décrira le sujet et introduira sa conception. La deuxième partie va être consacré au programmation et partie pratique du projet.

Chapitre 1

Etude théorique

1.1 Description du sujet

Le projet a pour but de réaliser un programme qui permet la gestion du travail de la société ManPower. Cette dernière est un des leaders mondiaux du travail temporaire. Il répond à tous les besoins de recrutement en CDD (Contrat à Durée Déterminée), CDI (Contrat à Durée Indéterminée), CA (Contrat en Alternance), CNE (. . .). Ainsi il propose aux entreprises un service qui consiste à leur dénicher le personnel qualifié qu'elles attendent et aux demandeurs d'emploi des OFFRES ciblées.

1.2 Objectifs

ManPower joue le rôle d'une société intermédiaire entre les candidats qui cherchent un emploi et les entreprises qui proposent des offres d'emploi.

— Les candidats et les entreprises doivent s'enregistrer auparavant et s'acquitter d'une cotisation annuelle.

— On doit avoir un fichier pour gérer les candidats et leur CVs.

— Un CV spécifie au moins un domaine de spécialité en plus des références sur le candidat.

— On doit maintenir l'ensemble des offres d'emploi proposées par les entreprises. Chaque

offre est valable à partir d'une date jusqu'à ce qu'elle trouve prenant.

- Chaque offre maintient une liste des candidats potentiels classés par ordre de mérite.
- Les données des structures utilisées doivent être enregistrer dans des fichiers.

1.2.1 Le modèle de la structure Candidat

- Index (Code unique)
- Nom
- Prénom
- Adresse mail
- Numéro de téléphone
- Adresse de résidence
- Domaine de spécialité
- Nombre de projets effectués
- Nombre de stages effectués
- Nombre d'années d'expériences
- Nombre de points

1.2.2 Le modèle de la structure Offre

- offre index (Code unique)
- Nom
- Date
- Domaine de spécialité demandée

1.2.3 Le modèle de la structure Recrutement

- recrutement index (Code unique)
- index candidat (Code unique)
- index offre (Code unique)

— Date

— Domaine de spécialité demandée

Chapitre 2

Réalisation et mise en oeuvre

2.1 Contraintes de programmation

Le programme propose un menu comme celui-ci :

- **Gestion des candidats :**
  - Ajouter un candidat
  - Modifier les donnes d'un candidat
  - Supprimer un candidat
  - Afficher la liste des candidats par ordre alphabétique
  - Afficher l'ensemble des offres susceptibles d'intéresser le candidat
- **Gestion des offres :**
  - Ajouter une nouvelle offre (d'une entreprise)
  - Modifier une offre (disponibilité par exemple)
  - Lister les offres disponibles selon plusieurs critères (entreprises, spécialités, . . .)
  - Lister les offres disponibles par ordre alphabétique (spécialités)
  - Afficher la liste des candidats potentiels pour cette offre
- **Gestion des recrutements :**
  - Ajouter un recrutement d'un type donné (CDD, CDI, . . .) une fois un candidat répond à une offre.

- Annuler un recrutement en mettant à jour la disponibilité de l'offre et/ou du candidat
- Afficher la liste des recrutements réalisés par la société dans un intervalle de temps donné

— **Quitter le programme** (par ESC)

— **Sécuriser les accès** à l'aide de mots de passe

2.2 Outils utilisés :

LaTex :

LaTeX est un système de synthèse de haute qualité, il comprend des fonctions destinées à la production de documents techniques et scientifiques. LaTeX est le standard de facto pour la communication et la publication de documents scientifiques. LaTeX est disponible sous forme de logiciel gratuit.

FIGURE 2.1 – L'outil de rédaction de rapport latex .

La documentation professionelle nécessite la manipulation du logiciel de traitement de texte LaTex. Travailler avec ce dernier est inévitable tôt ou tard, donc nous avons voulu exploiter cette opportunité et explorer LaTex.

CodeBlocks :

Code Blocks est un IDE gratuit en C, C ++ et Fortran conçu pour répondre aux besoins les plus exigeants de ses utilisateurs. Il est conçu pour être très évolutif et entièrement configurable.

FIGURE 2.2 – Editeur et compilateur de texte .

2.3 Difficultés rencontrées :

Parmi les difficultés qu'on avait rencontré on site **la contrainte du temps** Nous pensons que le temps était suffisant mais trop serré. Chercher à écrire un **clean code** est un défi pour tous les développeurs. Nous essayons d'assurer une clarté maximale. En fait, nous avons commenté le code de manière pertinente. De plus, nous avons divisé le code en plusieurs fichiers ".c" et ".h". On peut pas parler de la programmation ou bien du codage d'un programme sans parler **des erreurs de compilations** , ces erreurs sont indispensables partout dans le codesource.

2.4 Explications des fonctions et code source :

2.4.1 Gestion Candidats :

CreerCandidat :

```
candidat *CreerCandidat(int id,char *nom,char *prenom,char *domaine,int etat,int nb_exp);
```

FIGURE 2.3 – CreerCandidat.

Cette fonction permet de créer un candidat, en remplissant les données de ce dernier (nom, prénom, domaine de spécialité, nombre d'année d'expérience). Il retourne le candidat créé.

CreerElementCandidat :

```
elementcandidat *CreerElementCandidat(candidat *c);
```

FIGURE 2.4 – CreerElementCandidat .

Cette fonction permet de créer un élément candidat. Il retourne l'élément candidat.

AjouterCandidat :

```
void AjouterCandidat(listecandidat **L,elementcandidat *newElement);
```

FIGURE 2.5 – AjouterCandidat .

Cette fonction permet d'ajouter l'élément candidat créé auparavant dans la liste des candidats.

ExisteCandidat :

Cette fonction permet de chercher l'existence d'un candidat dans la liste des candidats, en le cherchant par son code. Il retourne 1 s'il existe et 0 sinon.

```
int ExisteCandidat(listecandidat *l,int code);
```

FIGURE 2.6 – ExisteCandidat .

GetCandidat :

```
candidat *GetCandidat(listecandidat *l, int code);
```

FIGURE 2.7 – GetCandidat .

Cette fonction permet de récupérer un candidat de la liste des candidats, en le cherchant par son code. Il retourne le candidat cherché.

ModifierCandidat :

```
void ModifierCandidat(listecandidat *l,int code,char *nom,char *prenom,char *domaine,int etat,int nb_exp);
```

FIGURE 2.8 – ModifierCandidat .

Cette fonction permet de modifier les données d'un candidat de la liste des candidats, en le cherchant par son code. Il ne retourne rien.

DeleteCandidat :

Cette fonction permet de supprimer un candidat de la liste des candidats, en le cherchant par son code. Il retourne -1 si la liste est vide rien, 1 si le candidat est trouvé et supprimé et 0 sinon.

```
int DeleteCandidat(listecandidat **l,int code);
```

FIGURE 2.9 – DeleteCandidat .

AfficherCandidat :

```
void AfficherCandidat(candidat *c);
```

FIGURE 2.10 – AfficherCandidat .

Cette fonction permet d'afficher les données d'un candidat. Il ne retourne rien.

AfficherTousCandidats :

```
void AfficherTousCandidats(listecandidat *AllClient);
```

FIGURE 2.11 – AfficherTousCandidats .

Cette fonction permet d'afficher tous les candidats de la liste. Il ne retourne rien.

AfficherOffresCandidat :

Cette fonction permet d'afficher tous les offres disponibles intéressant un candidat non recruté, en comparant le domaine de chaque offre disponible de la liste des offres avec le domaine du candidat. Il ne retourne rien.

```
void AfficherOffresCandidat(listeoffre *l,candidat *c);
```

FIGURE 2.12 – AfficherOffresCandidat .

AfficherCandidatsOffre :

```
void AfficherCandidatsOffre(listecandidat *l,offre *o);
```

FIGURE 2.13 – AfficherCandidatsOffre .

Cette fonction permet d'afficher tous les candidats non recrutés et adéquats à une offre, en comparant le domaine de de chaque candidat non recruté de la liste des candidats avec le domaine de l'offre. Il ne retourne rien.

DeListeAFichierCandidat :

```
void DeListeAFichierCandidat(listecandidat *L);
```

FIGURE 2.14 – DeListeAFichierCandidat .

Cette fonction permet de stocker tous les candidats de la liste dans un fichier.

DeFichierAListeCandidat :

Cette fonction permet de récupérer tous les candidats du fichier et les mettre dans une liste des candidats.

```
void DeFichierAListeCandidat(listecandidat **L);
```

FIGURE 2.15 – DeFichierAListeCandidat .

2.4.2 Gestion Offres :

CreerOffre :

```
offre *CreerOffre(int index,char *entreprise,SDate *d,char *DP,int Dispo);
```

FIGURE 2.16 – CreerOffre.

Cette fonction permet de créer une offre, en remplissant les données de cette dernière (nom de l'entreprise, domaine demandé, date d'expiration, disponibilité). Il retourne l'offre créée.

CreerElementOffre :

```
elementoffre *CreerElementOffre(offre *o);
```

FIGURE 2.17 – CreerElementOffre.

Cette fonction permet de créer un élément offre. Il retourne l'élément offre.

AjouterOffre :

Cette fonction permet d'ajouter l'élément offre créer auparavant dans la liste des offres.

```
void AjouterOffre(listeoffre **l,elementoffre *newElement);
```

FIGURE 2.18 – AjouterOffre.

ExisteOffre :

```
int ExisteOffre(listeoffre *l,int code);
```

FIGURE 2.19 – ExisteOffre.

Cette fonction permet de chercher l'existence d'une offre dans la liste des offres, en le cherchant par le code. Il retourne 1 s'il existe et 0 sinon.

GetOffre :

```
offre *GetOffre(listeoffre *L,int code);
```

FIGURE 2.20 – GetOffre.

Cette fonction permet de récupérer une offre de la liste des offres, en le cherchant par le code. Il retourne l'offre cherchée.

ModifierOffre :

Cette fonction permet de modifier les données d'une offre de la liste des offres, en le cherchant par le code. Il ne retourne rien.

```
void ModifierOffre(listeoffre *L,int code,int J , int M , int A,char *entreprise_nom, char *DP,int Dispo);
```

FIGURE 2.21 – ModifierOffre.

DeleteOffre :

```
int DeleteOffre(listeoffre **l,int code);
```

FIGURE 2.22 – DeleteOffre.

Cette fonction permet de supprimer une offre de la liste des offres, en le cherchant par son code. Il retourne -1 si la liste est vide rien, 1 si l'offre est trouvée et supprimée et 0 sinon.

AfficherOffre :

```
AfficherOffre (offre *o);
```

FIGURE 2.23 – AfficherOffre.

Cette fonction permet d'afficher les données d'une offre. Il ne retourne rien.

AfficherTousOffres :

Cette fonction permet d'afficher tous les offres de la liste. Il ne retourne rien.

```
void AfficherTousOffres(listeoffre *All);
```

FIGURE 2.24 – AfficherTousOffres.

AfficherOffresCriteres :

```
void AfficherOffresCriteres(listeoffre *l, char *nom, char *specialite);
```

FIGURE 2.25 – AfficherOffresCriteres.

Cette fonction permet d'afficher tous les offres d'une entreprise dans un domaine précis, en comparant le domaine et le nom de l'entreprise de chaque offre de la liste des offres avec le domaine et le nom de l'entreprise recherchés. Il ne retourne rien.

CandidatsPotenOffre :

```
void CandidatsPotenOffre(listecandidat *l,offre *o);
```

FIGURE 2.26 – CandidatsPotenOffre.

Cette fonction permet d'afficher tous les candidats non recrutés et adéquats à une offre et ayant le nombre d'année d'expérience supérieur e à 2 ans, en comparant le domaine de de chaque candidat non recruté de la liste des candidats avec le domaine de l'offre ainsi vérifier que ces candidats ont plus de 2 ans d'expérience. Il ne retourne rien.

DeListeAFichierOffre :

Cette fonction permet de stocker tous les offres de la liste dans un fichier.

```
void DeListeAFichierOffre(listeoffre *L);
```

FIGURE 2.27 – DeListeAFichierOffre.

DeFichierAlisteOffre :

```
void DeFichierAlisteOffre(listeoffre **L);
```

FIGURE 2.28 – DeFichierAlisteOffre.

fonction permet de récupérer tous les offres du fichier et les mettre dans une liste des offres.

2.4.3 Gestion Recrutements :

CreerRecrutement :

```
recrutement *CreerRecrutement(int rec_index,int offre_index,int ca_index,char *type_Contrat,SDate *d);
```

FIGURE 2.29 – CreerRecrutement.

Cette fonction permet de créer un recrutement, en remplissant les données de ce dernière (id de l'entreprise, ide du candidat recruté , date de recrutement, type de contrat). Il retourne l'offre créée.

CreerElementRecrutement :

Cette fonction permet de créer un recrutement. Il retourne l'élément recrutement.

```
elementrecrutement *CreerElementRecrutement(recrutement *Rec);
```

FIGURE 2.30 – CreerElementRecrutement.

AjouterRecrutement :

```
void AjouterRecrutement(listerecrutement **L,elementrecrutement *newElement);
```

FIGURE 2.31 – AjouterRecrutement .

Cette fonction permet d'ajouter l'élément recrutement créé auparavant dans la liste des recrutements.

ExisteRecrutement :

```
int ExisteRecrutement(listerecrutement *L,int code);
```

FIGURE 2.32 – ExisteRecrutement .

Cette fonction permet de chercher l'existence d'un recrutement dans la liste des recrutements, en le cherchant par son code. Il retourne 1 s'il existe et 0 sinon.

DeleteRecrutement :

Cette fonction permet de supprimer un recrutement de la liste des recrutements, en le cherchant par son code. Il retourne -1 si la liste est vide rien, 1 si l'offre est trouvée et supprimée et 0 sinon.

```
int DeleteRecrutement(listerecrutement **l, int code);
```

FIGURE 2.33 – DeleteRecrutement.

AfficherRecutement :

```
void AfficherRecutement(recrutement *r , listecandidat *lc , listeoffre *lo);
```

FIGURE 2.34 – AfficherRecutement.

Cette fonction permet d'afficher les données recutement (nom du candidat recruté, nom de l'entreprise, type de contrat, date de recrutement). Il ne retourne rien.

AfficherTousRecrutement :

```
void AfficherTousRecrutement(listerecrutement *all,listeoffre *o,listecandidat *c);
```

FIGURE 2.35 – AfficherTousRecrutement.

Cette fonction permet d'afficher tous les recrutements de la liste. Il ne retourne rien.

DeListeAFichierRecrutement :

Cette fonction permet de stocker tous les recrutements de la liste dans un fichier.

```
void DeListeAFichierRecrutement(listerecrutement *L);
```

FIGURE 2.36 – DeListeAFichierRecrutement.

DeFichierAListeRecrutement :

```
void DeFichierAListeRecrutement(listerecrutement **L);
```

FIGURE 2.37 – DeFichierAListeRecrutement.

fonction permet de récupérer tous les recrutements du fichier et les mettre dans une liste des recrutements.

Conclusion

Ce projet ne nous a pas permis uniquement d'améliorer nos connaissances théoriques et pratiques acquises pendant ces années d'études mais aussi de se familiariser avec le monde professionnel, et d'apprendre grâce à la cohésion du groupe à s'adapter et trouver des solutions efficaces dans un temps très réduit aux problèmes rencontrés que ce soient lors de l'installationdes outils utilisés ou lors du développement et codage du programme.

Titre : Projet de programmation C Gestion d'une société d'Intérim

Mots clés : Gestion, candidats, offre d'emploi, recrutements, entreprises.

Résumé : Les sociétés intermédiaires dans nos jours jouent un rôle très important entre les candidats cherchant une offre d'emploi et les entreprises cherchant le candidat adéquat à cette offre. La phase du traitement des demandes soit celles des entreprises ou bien celles des candidats devient plus délicate avec l'augmentation du nombre des demandes, et pour faciliter cette tâche, on pensait à produire un programme permettant une gestion plus fluide et plus aisée.

Title : Programming project C Management of an interim company

Keywords : Management,candidates,job offer,recruitments, companies.

Abstract : Intermediary companies these days play a very important role between candidates looking for a job offer and companies looking for the right candidate for that job. The processing phase of requests, either from companies or from candidates, becomes more delicate with the increase in the number of requests, and to facilitate this task, it was thought to produce a program allowing more fluid and easier management.