| Icône d’accentuation | Livrable du projet de SGBD DIC1 Informatique |
| --- | --- |

# Travail réalisé par :

Chérif Younouss Ndiaye

Serigne Cheikh Mbacké Thiaw

# **Introduction**

Afin de faciliter l’analyse de flux de donnees au format JSON ou XML, il nous a été demandee de developper une solution d’ingenierie inversee. La dite solution devra permettre de generer un fichier SVG permettant de visualiser un diagramme entite-association refletant la structuration des donnees qu’elle aura en entree

Nous avons utilisee le langage de programmation C afin de pouvoir realiser la solution.

Dans la suite, on va d’abord presenter la structuration du projet et enfin les fonctions qui nous ont permis de realiser les differents traitements relatifs aux differents flux de donnees

# **Structuration du projet**

Le projet comprend essentiellement trois modules :

1. Module JSON : Dans ce module sont effectuées tous les traitements relatifs au flux de données de type JSON
2. Module XML : Dans ce module sont effectuées tous les traitements relatifs au flux de données de type XML
3. Module SVG : Ce module traite de la génération du fichier SVG qui doit comporter le diagramme d’entité-association

**Dans la suite, nous allons voir les différentes fonctions qui caractérisent ces modules.**

# **Description des fonctions utilisées**

# Module JSON

* 1. Validation de données

Pour la validation de données au format JSON, la librairie JSON\_CHECKER a été utilisée.  
Les fonctions suivantes ont été notamment utilisées :

* + - int est\_valide (char\* filename) :

Cette fonction prend en argument le nom du fichier à valider et retourne 0 si la validation est réussie. Sinon, elle arrête le programme par le biais de la fonction exit.

* + - int JSON\_checker\_char(JSON\_checker jc, int next\_char) :

Il s’agit ici d’une fonction de la librairie JSON\_CHECKER qui permet de faire la validation d’une structure JSON caractère par caractère

* + - int JSON\_checker\_done(JSON\_checker jc) :

Il s’agit d’une fonction de la librairie JSON\_CHECKER qu’on appelle après que tous les caractères ont été. Elle retourne alors VRAI si la vérification est réussie.

1. Extractions des données

Pour l’extraction des données au format JSON, la librairie cJSON a été utilisée  
Les fonctions suivantes peuvent y être notées :

* char\* chaine\_json(char\* filename) :

Cette fonction permet de parcourir un fichier et de retourner son contenu sous format d’une chaîne de caractères

* void get\_nombre\_entites\_associations(cJSON\* json, int\* nb\_entites, int\* nb\_associations) :

Cette procédure permet de récupérer le nombre d’entité et le nombre d’associations

* char\*\* get\_attributs\_entite(cJSON\* json, char\*\* attributs, int\* n\_attributs) :

Cette fonction permet de récupérer les attributs d’une entité

* char\*\* get\_attributs\_association(cJSON\* json, char\*\* attributs, int\* n\_attributs) :

Cette fonction permet de récupérer les attributs d’une association

* void test(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) :

Cette fonction permet de parcourir le premier niveau de l’arborescence du fichier JSON

* void test2(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) :

Cette fonction permet de réaliser le parcours récursif de l’arborescence du fichier json

* char\*\* get\_associations(cJSON\* json, cJSON\* parent, char\*\* associations, int\* n\_associations) :

Cette fonction permet de récupérer les associations auxquelles participent l’entité

* void afficher\_entite(s\_entite\* entite) :

Fonction permettant d’afficher une entité pris en argument

* void afficher\_association(s\_association\* association) :

Fonction permettant d’afficher une association

* s\_entite\* creer\_entite(s\_entite\* e, char\* nom, char\*\* attributs, char\*\* associations, int n\_attributs, int n\_associations) :

Fonction qui crée et retourne une entité

* s\_association\* creer\_association(s\_association\* a, char\* nom, char\*\* attributs, int n\_attributs) :

Fonction qui crée et retourne une association

* void test2\_bis(cJSON\* json, s\_entite\*\* tab\_entites, s\_association\*\* tab\_associations) :

Fonction pour faire l’extraction de nœuds de même niveau

* void get\_nombre\_e\_a\_bis(cJSON\* json, int\* nb\_entites, int\* nb\_associations) :

Cette fonction permet de faire le parcours de nœuds de même niveau pour récupérer le nombre d’entités et d’associations.

# Module XML

Pour la validation et l’extraction de données, nous avons utilisé la librairie libxml2.

1. Validation des données

Nous avons utilisés une seule fonction dans cette partie qui est la suivante:

* + - void xmlIsValid(char \*file) :

La fonction lit un fichier en entrée et vérifie si le fichier est en format XML bien formé en validant le DTD et les caractères.  
Sa déclaration est dans xmlValideFile.h

1. Extractions des données

Pour cette partie-ci, les fonctions utilisées sont les suivantes :

* + - void extractionData(char \*filename) :

La fonction lit un fichier en entrée et initialise un nœud racine et appel la fonction printData pour l’extraction de données

* + - void printData(xmlNode \*a\_node) :

C’est au niveau de cette fonction que nous parcourons l’arborescence du fichier XML et que nous extrayons les entités, les attributs,les associations et les relations.

* + - void save\_data(const xmlChar \*data, File \*f) :

Cette fonction enregistre la trace des données extraites dans un fichier nommé saveFile.txt

* + - int nombre\_entite(xmlNode \*root) :

La fonction parcours l’arborescence d’un fichier XML et compte le nombre d’entités définit.

void creerEntite(xmlChar \*name) :

La fonction dessine dans le fichier svg une entité  
Ces fonctions sont déclarées dans extractData.h.

# Module SVG

Pour la création et la génération de fichier svg, nous avons utilisé la librairie cairo.  
Les principales fonctions utilisées sont les suivantes :

* + - void dessiner\_entite(cairo\_t\* cr, double x, double y, double width, double height, char\* nom\_entite, char\* attributs[], int n\_attributs, int left\_right) :

Il s’agit là de la fonction qui permet de dessiner une entité avec une taille donnée à une position donnée.

* + - double percentage(double x, double per\_cent) :

Cette fonction calcule un pourcentage d’un nombre

* + - void dessiner\_association(cairo\_t\* cr, double xc, double yc, double radius, char\* nom) :

Cette fonction permet de dessiner une association à une position donnée

* + - int chercher\_chaine(char\*\* attributs, char\* chaine, int n\_attributs) :

Cette fonction permet de faire la recherche d’une chaîne de caractères dans un tableau de chaîne de caractères passé en paramètre

* + - void generer\_svg(char\* file, char\* svg\_file, int count\_t) :

Cette fonction permet de générer le fichier svg avec le nom (svg\_file) qui lui est passé en argument. Elle permet aussi d’afficher la trace si l’argument count\_t est différent de 0.

# IV.Flux http

Pour récupérer les données provenant d’un serveur, on a utilisé la librairie curl.

Les fonctions suivantes ont par la suite été définis :

* + - char \*request(const char \*url) :

Cette fonction prend en paramètre un url et va récupérer les données qui s’y trouvent. Les données nous sont retournées sous forme de chaîne de caractères.

# **Référence:**

[**https://github.com/cherifyounouss/projet\_sgbd\_dic1/tree/master**](https://github.com/cherifyounouss/projet_sgbd_dic1/tree/master)

[**http://xmlsoft.org/**](http://xmlsoft.org/)

[**https://github.com/DaveGamble/cJSON**](https://github.com/DaveGamble/cJSON)

[**https://github.com/douglascrockford/JSON-c**](https://github.com/douglascrockford/JSON-c)

**https://cairographics.org/**