REPUBLIQUE DU SENEGAL

UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT GENIE INFORMATIQUE

DIC Télécommunications et Réseaux

Rapport de projet Système de gestion de bases de données

Sujet : Reverse Engineering – Ingenierie Inversée

Présentées par

- **CISSE Narou**
- **DIOUF Elisabeth Ndame**

Professeur :

- **Mr Mbacke**

Année universitaire 2018-2019

La description des procédures/fonctions/classes utilisées

I. Vérification de la validité du fichier en input et de son format

les librairies utilisees sont :
-le module **json** permet de créer et de lire des données au format **json permet également sa validations.**
-sys permet de gérer les argument ce qui nous a permi de créer un tableau pour l'implementation de la commande **XJ_Convertor**
-svgwrite **pour créer des dessin svg**

1) Les fonctions qui nous ont permis de faire ces validations sont les suivantes :

a) def parseJSON(data):

Cette méthode de chargement de JSON est utilisée pour charger le fichier JSON à partir d'une chaîne notamment avec **json.loads().** Au niveau de la fonction, on a utilisé la méthode **open()** afin d'ouvrir le fichier et qui renvoie un fichier objet et la fonction **read()** qui lit l'intégralité du texte du fichier sous forme de chaîne unique.

b) def json_validator(myfile):

Au niveau de cette fonction, on vérifie si l'on arrive à charger un fichier. Si oui, on en déduit que le fichier est au bon format c'est-a-dire s'il est valide.

c) def xml_validator(file):

Afin de valider le fichier xml, nous avons verifier si ce fichier est d'abord au bon format si tel est le cas on fait un return true sinon on fait un return false.

II. Extraction des entités , des relations et des attributs de relations et entités

d) XML

Pour extraire les entites du fichier xml, on a utiliser les quatre fonctions suivantes : def getEntites(), def getHeritEntites(), def getStereotypes(), def getRelations()

Dans la fonction **getEntites()** , on utilise une fonction **ET.ElementTree()** qui est définie dans **xml.etree.cElementTree as ET** prend en entrée un fichier, extrait les entités sous formes d'arbre. Au niveau de la fonction, on fait appel de la fonction

getroot() qui donne la racine de l'arborescence ainsi grâce à elle, nous pouvons naviguer et itérer sur les éléments du fichier pour ressortir ainsi les attributs, leurs nom, etc.
def getRelations(), cette fonction permet de faire la relation entre les différentes tables.

e) JSON

Pour l'extraction des entités du fichier JSON qu'on a au préalable validé, on l'a directement fait au niveau de la fonction main en créant un tableau qui contiendra la liste des entités et en utilisant la fonction range() qui renvoie une séquence de nombres, commençant par 0, incrémente de 1 (par défaut) et se termine par un nombre spécifié. Ainsi pour extraire les entités, on parcourera le tableau pour ainsi recupérer l'ensemble des entités.

III. Génération du Fichier de Sortie dans le répertoire en cours

Au niveau de la génération, on a utilise la librairie **svgwrite** où différentes fonctions sont définies pour créer un fichier SVG.
L'objet *Drawing* est le conteneur global de tous les éléments SVG. Il fournit les méthodes pour stocker le dessin dans un fichier ou un objet de type fichier.
Après, on trace les coordonnées des rectangles contenant les informations de l'entité puis on entre le nom de l'entité et ses attributs et faire leurs associations.

IV. Url d'un dépôt sur github

https://github.com/Narou98/projetsgbd

V Démo de la commande sur Linux

```
python XJ_Convertor.py -i xml  -f input.xml -o test.svg
python XJ_Convertor.py -i json  -f input.json -o svgJson.svg
```