Chi-Hoon Lee
Anthony Palmigiano
Thibault Neulat
Louis Palayret

PLAN DE TEST VCARD

Date: 11/01/2015
Plan de Test Version 1.0

Table des matières

OBJECTIF DES TESTS : VERIFIER LA BONNE REPONSE DE L'APPLICATION A CHAQUE SPECIFICATION

ENVIRONNEMENTS DE TEST

Système d'exploitation : Windows 8.1, Windows 7, OSX 10.10
Version node.js : v0.10.33
Résultat des tests : Issue Tracker Google Code :

- remonter les anomalies via Type Defect
- demande d'amélioration via Type-Enhancement

SCENARIO DE TEST

SCN1 - Vérifier que le format de sortie soit .CSV

Donnée(s) :

- 1 fichier contact .vcf

Exécution de la commande avec un des fichiers de contact fourni : node parser.js < file name>.
L'exécution indique *"Aucun doublon trouvé, Le fichier CSV a été créé"*
Après vérification un fichier .CSV a bien était crée.

Test réussi.

SCN2 - Vérifier que l'exécution de la commande ne supprime pas le fichier existant

Donnée(s) :

- 1 fichier contact .vcf
- 1 fichier .csv déja existant

Après la création précédente d'un fichier .csv via la première exécution de commande, nouvelle exécution de la commande (avec un des fichiers de contact fourni) : node parser.js < file name>.
On vérifie qu'une ligne a été ajoutée au fichier .csv déja crée et que celui-ci n'est pas recrée.

Test réussi.

SCN3 - Vérifier que le format d'import soit *.vcf*

Donnée(s) :

- 1 fichier README.txt

Exécution de la commande avec un fichier *.txt*.

L'exécution indique "*Aucun doublon trouvé, Le fichier CSV a été créé".*

Une erreur devrait être affiché comme le format d'import ne correspond pas, hors la commande indique que tout c'est passé correctement. Après vérification dans le fichier *.csv,* aucune ligne n'a été rajoutée.

Test échoué.

SCN4 - Test synchronisation

Donnée(s) :

- 1 fichier contact .vcf

L'intitulé des colonnes devrait être affiché en début du fichier de sortie, hors il arrive qu'il soit écrit après certains contacts.

Le problème est du à l'utilisation de la fonction fs.appendFile au lieu de fs.appendFileSync

Test échoué.

SCN5 - Vérification de la fonction createContact

Donnée(s) :

- 1 fichier contact .vcf

On exécute la commande avec un fichier contact ne contenant pas de nom.

Il y a une erreur dans la fonction createContact, plus précisément dans le code ci-dessous :

```
if (key[i] != undefined && key[i] == "FN" ) {
        nbrContact++;
        tmp1[i] = value[i].split(' ');
        fName[nbrContact] = tmp1[i][0];
        lName[nbrContact] = tmp1[i][1].replace("\r", "");
        fullName[nbrContact] = lName[nbrContact] + " " + fName[nbrContact];
}
```

Ici, on remplace le retour de ligne "\r" de valeur "tmp1[i][1]" par blanc, mais si le contact n'a pas de nom, ça ne marche pas.
Donc la ligne "lName[nbrContact] = tmp1[i][1].replace("\r", "");" ne marche pas pour les cas des contacts n'ayant pas de nom ou de prénom.

Test échoué.

SCN6 - Vérification en cas d'informations manquantes

Donnée(s) :

- 1 fichier contact .vcf

On exécute la commande avec un contact contenant des cases vides (par exemple nom d'organisation non défini),
Le fichier *.csv* de sortie indique "undefined" pour les champs manquants.

Test partiellement échoué.

SCN7 - Format .csv non respecté par rapport au cahier des charges

Donnée(s) :

- 1 fichier contact .vcf

Dans le Cahier des Charges, la forme backus-naur de sortie suivante était attendue :

<import> ::= <nom> <EOL>
<prenom> <EOL>
<organisations> <EOL>
<fonctions> <EOL>
<telephone> <EOL>
<mobile> <EOL>
<courriel> <EOL>

Il a donc été spécifié que le format de sortie devait être le suivant :
nom;prénom;organisation;fonction;téléphone;mobile;courriel

Hors, le format de sortie du fichier est :
prénom;nom;organisation;fonction;mobile;téléphone;courriel

Test échoué.

SCN8 - Test de la redondance

Donnée(s) :

- 1 fichier contact *redondance.vcf*

Le nombre de cas de redondance traité est insuffisant.

Dans le cas où 2 contacts on le même nom mais l'un possède un numéro, l'autre nom, le programme plante.

****
*

Il y a deux contacts (numéros 2 et 1) portants le même nom : "Jérémy Barat"

```
/parser.js:300
            if ((phoneOf(x)[0][0] == phoneOf(u)[0][0] && phoneOf(x)[1]
                                    ^
TypeError: Cannot read property '0' of undefined
  at checkName (parser.js:300:59)
  at imax (parser.js:48:5)
  at fs.js:271:14
  at Object.oncomplete (fs.js:107:15)
```

Test échoué.

SCN9 - Nom des colonnes

Donnée(s) :

- 1 fichier contact

Dans le format de sortie du fichier .csv, le nom des colonnes n'est pas présent dans le fichier de sortie.

Test échoué.

SCN10 - Argument

Donnée(s) :

- 1 fichier contact

Lorsque l'on exécute la commande, le fichier de sortie porte le nom *csvFile.csv,* il devrait être possible de choisir nous même le nom du fichier de sortie.

Test échoué.

ORDRE D'EXECUTION DES TESTS

SCN1 -> SCN2 -> SCN3 -> SCN4 -> SCN5 -> SCN6 -> SCN7 -> SCN8 -> SCN9 -> SCN10

BILAN DES TESTS

SCN1 - Vérifier que le format de sortie soit *.CSV*

Scénario conforme.

SCN2 - Vérifier que l'exécution de la commande ne supprime pas le fichier existant

Scénario conforme.

SCN3 - Vérifier que le format d'import soit *.vcf*

Le fichier .csv n'est pas modifié si le fichier d'import n'est pas *.vcf.* Cependant aucune erreur n'est affichée dans la console.
Il faudrait rajouter un message indique que le fichier d'entrée n'est pas au format *.vcf*.

Donc dans la fonction *fs.readFile()* , on a ajouté des conditions *if* et *else* suivantes :

```
if(myArgs[1].substr(myArgs[1].length-4, myArgs[1].length-5)!=".vcf")
{
        console.log("Format de fichier invalide, veuillez insérer un fichier .vcf");
}
else
{

  /* appel fonctions */
  parse_key_value();
  createcontact();
  cleanTel();
  checkPhone();
  checkName();
  writeCSV();

}
```

Le programme appelle toutes les fonctions uniquement quand le fichier inséré est en format .vcf, et pour les fichiers de différents formats, le programme affiche un message d'erreur.

SCN4 - Test synchronisation

La fonction *fs.appendFile* a été remplacée par *fs.appendFileSync.*

SCN5 - Vérification de la fonction createContact

Comme la ligne "lName[nbrContact] = tmp1[i][1].replace("\r", "");" ne marche pas pour les cas des contacts n'ayant pas de nom ou de prénom, la solution serait de traiter tous ces cas , voici donc la version corrigée du code :

```
if (key[i] != undefined && key[i] == "FN" ) {
        nbrContact++;
        tmp1[i] = value[i].split(' ');

                if(tmp1[i][1] == undefined)
                        {
                                lName[nbrContact]="";
                                fName[nbrContact] = tmp1[i][0].replace("\r", "");
                                fullName[nbrContact] = tmp1[i][0].replace("\r", "");
                        }
                else if(tmp1[i][0] == undefined)
                        {
                                fName[nbrContact]="";
                                lName[nbrContact] = tmp1[i][1].replace("\r", "");
                                fullName[nbrContact] = tmp1[i][1].replace("\r", "");
                        }
                else
                        {
                                fName[nbrContact] = tmp1[i][0];
                                lName[nbrContact] = tmp1[i][1].replace("\r", "");
                                fullName[nbrContact] = lName[nbrContact] + " " +
fName[nbrContact];
                        }
}
```

La première condition(if) vérifie s'il n'y a pas du nom, la deuxième condition(else if) s'il n'y a pas du prénom, et la troisième(else) s'il y a nom et prénom.

SCN6 - Vérification en cas d'informations manquantes

Le fichier *.csv* de sortie indique "undefined" pour les champs manquants, on peut donc remplacer tous les "undefined" par des caractères vides en ajoutant la fonction :

```
function deleteUndefined(data)
{
        if(data==undefined || data == "undefined"){
                return data = "";
        }
        else return data;
}
```

Cette fonction sera appliquée lors de l'exécution de la fonction : writeCSV().

SCN7 - Format .csv non respecté par rapport au cahier des charges :

Pour pallier au problème de format de sortie non respecté et l'absence d'informations quant aux colonnes, il est nécessaire de changer l'ordre des champs et ajouter le nom aux colonnes, pour obtenir :
"nom;prénom;organisation;fonction;téléphone;mobile;courriel"

SCN8 - Test de la redondance

Des cas de redondance n'ont pas été traités, il aurait été judicieux de rajouter des cas, notamment celui où 2 contacts on le même nom mais l'un possède un numéro, l'autre non. Dans le cas cité ci-dessus, le programme plantait. Une modification de ce dernier à été mise en place de façon à ce qu'il finisse quand même son execution, sans pour autant créer un fichier CSV.

SCN9 - Nom des colonnes

Le nom des colonnes a été rajouté.

SCN10 - Argument

Le programme accepte maintenant un argument supplémentaire, et permet à l'utilisateur de choisir le nom du fichier de sortie. Il n'est pas nécessaire de fournir l'extension du fichier (par défaut .csv), celle-ci est automatiquement ajoutée.

L'appel du programme se fait donc désormais de la manière suivante :

node parser.js <nom fichier d'entrée> <nom fichier de sortie>