Documentation

Présentation

Ce projet propose un système d'automatisation qui permet d'extraire un ensemble de contacts depuis une image. L'utilisateur n'a pas à se soucier des programmes utilisés, une chaîne d'automatisation est créée et nécessite simplement le dépôt de l'image source et d'un fichier de configuration.

Etapes

- **Extraction du texte brut**, qui récupère simplement l'ensemble des caractères et des mots reconnus sur l'image;
- **Extraction des informations**, qui récupère les informations demandées par l'utilisateur;
- **Conversion** des données au format souhaité, indiqué par l'utilisateur.

Sources

Les codes sources sont fournis dans l'archive `.tgz`, dans laquelle cette documentation se situait. Vous y trouverez les fichiers suivants :

- `start`, qui permet de démarrer la chaîne de conteneurs
- `stop`, qui permet de l'arrêter
- `envoyer`, qui envoie un fichier image et un fichier de configuration au début de la chaîne
- `recevoir`, qui récupère le rendu à la fin de la chaîne

Après extraction, **la première chose à faire est d'ajouter les droits d'exécution** à tous les scripts :

```
chmod +x start stop envoyer recevoir
```

Ces 4 scripts sont ceux avec lesquels un technicien ou un utilisateur doit interragir. Ils contiennent les commandes essentielles et complètes pour la gestion de la chaîne. D'autres fichiers sources, internes au fonctionnement, se situent dans le dossier `src/` :

- `tesseract.sh`, script de la première étape, qui extrait les informations en texte brut et envoie le résultat dans un fichier `output.txt`
- `extraction.php`, script de la deuxième partie, qui extrait les informations demandés depuis `output.txt` vers `output.csv`
- `conversion.sh`, script de la troisième partie, qui transforme `output.csv` au format souhaité
- `modele.php`, modèle utilisé par la troisième partie pour générer un HTML

Fichier de configuration

Le fichier texte de configuration doit absolument se nommer `config.txt`.

Champs

Les champs permettent de préciser quels types d'information récupérer. Les champs recherchés possible sont :

- nom
- prenom
- email
- telephone
- adresse

L'ordre entré par les utilisateurs reste le même sur le résultat, ainsi si `prenom` est entré avant `nom`, le résultat aura lui aussi d'abord le prénom puis le nom.

Les informations complémentaires

Pour les champs **nom**, **prenom** et **adresse**, il est nécessaire de renseigner des informations complémentaires, comme par exemple :

```
nom : "last_name"
prenom : "first_name"
adresse : "address"
```

Format de l'image source :

```
{
    "last_name" : "Deschamps"
    "first_name" : "Didier"
    "address" : "Paris"
}
```

Ces informations complémentaire permettent au programme de mieux détecter les données souhaitées.

Cas du tableau

Si l'image fournie en entrée est un tableau alors l'utilisateur doit spécifier le type et le nombre **total** de colonnes. Par exemple :

```
tableau
colonnes : 3
```

> NB : pour les tableaux, tous les champs ont besoin d'informations complémentaires.

Format de sortie

Les formats en sortie proposés sont **PDF**, **HTML** et **CSV**. Le format de sortie doit être précisé comme suit :

```
format : "PDF"
```

Règles sur le fichier de configuration

Pour éviter d'obtenir des erreurs en sortie, vous pouvez suivre les règles suivantes :

1. Mettre les informations entre guillemets
2. Ne pas rajouter d'accents sur les noms des champs
3. Utiliser le nom de la colonne pour le format tableau

Fonctionnement interne

Les 3 conteneurs suivants sont utilisés. L'usage de ces 3 conteneurs est indispensable :

- tesseract
- php-cli74
- weasyprint

Au début, on crée un volume nommé `sae103`, puis on importe les différentes images docker que l'on va être amené à utiliser par la suite. Après ça, on envoie les données (scripts bash et php) dans un conteneur temporaire créé au préalable (ce conteneur `sae103-tmp` sert de liaison entre les conteneurs et les volumes) afin de les envoyer dans le volume `sae103`. Une fois ceci fait on supprime ce conteneur temporaire. Chaque conteneur est lancé en mode détaché et est lié au volume `sae103`

Exemple d'utilisation complète

Les commandes, dans l'ordre, à lancer pour l'utilisation de la chaîne sont les suivantes :

Démarrage de la chaîne :

```
./start
```

Envoie d'une image, puis du fichier de configuration. **Le fichier de configuration doit être mis en dernier**, car il est l'élément déclancheur de la chaîne :

```
./envoyer dossier/image.png
./envoyer dossier/config.txt
```

Enfin, la récupération des éléments produits par la chaîne, dans le dossier `out_dir` :

```
./recuperer
```

Répeter autant de fois que nécessaire l'envoi et la récupération des fichiers produits. Pour finalement arrêter la chaîne :

```
./stop
```

Cas de test

La documentation est fournie avec 2 cas de tests pour comprendre l'utilisation et le

fonctionnement de la chaîne.

Cas n°1

Le premier test s'effectue avec un tableau, se trouvant de le dossier `test/tableau/`. Il possède un fichier `config.txt`, ainsi qu'une image `tableau.png`

Lorsque l'utiisateur souhaite lancer le test il lui suffit d'entrer les commande suivante (une fois la chaîne lancée) :

```
./envoyer test/tableau/tableau.png
./envoyer test/tableau/config.txt
```

Puis pour récupérer les fichiers produit :

```
./recuperer
```

Cas n°2

Le second cas s'effectue avec un fichier JSON. Il se trouve dans le dossier `test/json/`. De même que pour le premier test, il possède une image (`sample.png`) et un fichier `config.txt`.

Pour exécuter le test il faut lancer la commande suivante (une fois la chaîne lancée) :

```
./envoyer test/json/sample.png
./envoyer test/json/config.txt
```

Puis pour récupérer les fichiers :

```
./recuperer
```