Receipt parser server

Monolidth

Jul 11, 2022

RECEIPT SERVER API

Receipt parser server is a modular, minimal server to parse receipts.

CHAPTER

ONE

UPLOAD API

The upload API is used to upload a given receipt to the receipt parser server. The server return the parsed image (if successful) or an ERROR code.

1.1 Entrypoint

The entrypoint of the upload api is `api/upload`.

1.2 Parameter

Parameter	Type	Default value	Description
legacy_parser	bool	false	Use the legacy parser
grayscale_image	bool	false	Grayscale the image
gaussian_blur	bool	false	Apply the gaussian blur
rotate_image	bool	false	Rotate image

Please note: The parameter *file* and *access_token* is always required. Take a look at the cURL example.

1.3 Return Code

Return code	Event
200	request is valid
403	APITOKEN is invalid
415	image is invalid

1.4 Curl example

```
curl -X POST "https://$IP:$PORT/api/upload?access_token=$API_TOKEN -H  "accept:␣
↪application/json" -H  "Content-Type: multipart/form-data" -F "file=$IMAGE;type=image/
↪jpeg"
```

with the given parameters:

Parameter	Description
IP	The server ip
PORT	The server port
ACCESS_TOKEN	The server access token
IMAGE	The receipt image

CHAPTER

TWO

TRAINING API

The training API is used to upload a given receipt to the receipt parser server. The server return the parsed image (if successful) or an ERROR code.

2.1 Entrypoint

The entrypoint of the upload api is `api/training`.

2.2 Return Code

Return code	Event
200	request is valid
403	APITOKEN is invalid
415	image is invalid

2.3 Parameter

The parameter receipt and access_token is always required. Take a look at the cURL example.

2.4 Curl example

```
curl -X POST "https://$IP:$PORT/api/training?access_token=$ACCESS_TOKEN" -H  "accept:␣
↪application/json" --data '{"company":"$COMPANY_NAME","date":"$DATE","total":"$RECEIPT_
↪TOTAL"}'   -k
```

with the given parameters:

Parameter	Description
IP	The server ip
PORT	The server port
ACCESS_TOKEN	The server access token
RECEIPT	Receipt object as json

the receipt object is submitted via the `--data` flag.

CHAPTER

THREE

DOCKER INSTALLATION GUIDE

The receipt-parser-server image gets built automatically using the Docker Hub. The installation is very simple. First pull the image from Docker hub.

```
docker pull monolidth/receipt-parser:latest
```

3.1 Recommended

The launcher script does take care of various things e.g.

- cleanup old Docker container
- forward IP
- use the pseudo TTY
- forward configuration file

1. Download the launcher script
2. Execute the launcher script

```
wget https://raw.githubusercontent.com/ReceiptManager/receipt-parser-server/master/util/
↪launcher.sh
wget https://raw.githubusercontent.com/ReceiptManager/receipt-parser-server/master/
↪config.yml
bash launcher.sh
```

3.2 Manual

You could also run the Docker image without the launcher script e.g.

```
docker run -i -t -p [YOUR-IP]:8721:8721  monolidth/receipt-parser
```

CHAPTER

FOUR

DEVELOPER INSTALLATION GUIDE

4.1 Clone the repository

First clone the GitHub project.

```
git clone https://github.com/ReceiptManager/receipt-parser-server
```

4.2 Install project dependencies

Please notice that you install following packages with your favorite package manager:

- python
- python-pip
- libmagickwand-dev
- tesseract-ocr-all
- libgl1-mesa-glx
- libmagickwand-dev
- qrencode

```
apt-get install python python-pip libmagickwand-dev tesseract-ocr-all libgl1-mesa-glx␣
↪libmagickwand-dev
```

4.3 Install python dependencies

Now, install all python dependencies using *pip* the following

```
pip install -r requirements.txt
```

4.4 Generate SSL certificates

Now, generate new SSL certificates. First, generate a new file called `.private_key` and type your favourite password. Please submit at least 8 characters. You can do this using echo like:

```
echo "favorite_password" > .private_key
```

The password is used to generate the root certificate. Generate the cert files using

```
make generate_cert
```

Now, you should see new certificates located in cert folder which is located in the root directory.

```
ls cert
```

The output looks like the following

```
rootCA.key  rootCA.pem  rootCA.srl  server.crt  server.csr  server.csr.cnf  server.key␣
↪v3.ext
```

4.5 Run the server

Now, you are ready to run the Receipt Parser Server.

```
make serve
```

CHAPTER

FIVE

VERIFY INSTALLATION

If you run the Docker image. The output should like similar to:

```
...
INFO:     Started server process [16322]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on https://0.0.0.0:8721 (Press CTRL+C to quit)
```

The API token in printed on the screen. Additionally, you can scan the QR code.

```
Current API token: XXXXXXXX
```

CHAPTER

SIX

SERVER CONFIGURATION

Following keys need to be defined.

Config key	Default value	Description
language	de	Define the tesseract language
https	true	Enable HTTPS
receipts_path	“data/txt”	Path where receipts are stored
markets	**markets:** **store name:** • likely name 1 • likely name 2	Markets name
sum_keys	• summe • gesamtbetrag • gesamt • total • sum • zwischensumme • bar • te betalen	Keys to identify sum
ignore_keys	• rockgeld • rusckgeld • rückgeld • mwst	Keys
sum_format	d+(.s? |,s?|[^a-zA-Zd])d{2}	Regex to identify the receipt total
item_format	([a-zA-Z].+)s(-|)(d,dd)s	Regex to identify the receipt items
date_format	((d{2}.d{2}.d{2,4}) |(d{2,4}/d{2}/d{2})|(d{2}/d{2}/d{4}))	Regex to identify the receipt date

6.1 Add new market names

You can add new market entry below the `markets` key e.g.

```
Store name:
        - likely name 1
        - likely name 2
```

Note: that the store name is returned and the likely names are used to scan the receipt for these names. You can consider the receipt parser output in `data/txt`

In this example, the tesseract output looks like:

```
EWE Rene Müller 0HGCITY
org-Friedrich-Str.9

}
-
L
L
E
/
D “il s

L „é„ 31 Karlsruhe
| | 50 /
R 0/Z1 / 664 87 954
LL UID Nr. : DE326445229
B ) EUR
-- MIO MIO MATE -
| | )
_„*}„%_ PF£N3t% ?5 1108 4
| 6 Sal ,19 .EUR X
f““j“i$“ 2 5Stk x \ 0,15 V
E E O
_; Ge R "*M-w-‘--»»------*_„::_;::::..:‘::.;:_:.;r:::::.::ä..b-ö-«"
; , Rückgeld BAR EHE 0, 32
%---%---i S£Buer % Netto: steuer B[9£15
| HAL En 2,25 0,43 . 268
/ ““‘};l@samtbetrag 2,25 0,43 a Z
f ı TSE-Signatur: M631mP54IvkcwnNk+H7th3&meTdLüö[w
8 0bo5B71skamunHSsZC1Z4q9ds6BRoDNWg
Sa aUfagzEbyt TDVULU2ecc4rUk5/3211shY
```

The output looks horrible but you might noticed that the store name is Rewe but the output is: `EWE Rene Müller 0HGCITY`. Now, add the following market in the `config.yml`.

```
REWE:
 - ewe
```

To identify the market name Rewe but be carefully for duplicate store names. If the store name Rewe exist please only add the likely name `ewe`.

CHAPTER

SEVEN

FOR DOCKER USERS

7.1 Forward config

If you use the Docker image, you can forward the configuration file `config.yml`. If the `config.yml` is in your current directory you can add the following flag

```
-v "$(pwd):/config" -e RECEIPT_PARSER_CONFIG_DIR="/config"
```

If the config file is not in your current working directory. Replace `$(pwd)` with you the configuration folder.

7.2 Forward IP

Additionally, you can forward the Docker IP using:

```
-p $IP:8721:8721
```

CHAPTER

EIGHT

EXAMPLE CONFIG

```
    # Define the tesseract language
language: deu

# Enable https
https: true

# Where the receipts are stored
# Receipts should be simple text files
receipts_path: "data/txt"

# Market names roughly ordered by likelihood.
# Can contain market locations for fuzzy parsing
markets:
  Colruyt:
     - colruyt
     - Colruyt
  Delhaize:
     - Delhaize
     - delhaize
  Penny:
     - penny
     - p e n n y
     - m a r k t gmbh
  REWE:
     - rewe
  Real:
     - real
  Netto:
     - netto-online
  Kaiser's:
     - kaiser
     - kaiserswerther straße 270
  Aldi:
     - aldi
     - friedrichstr 128-133
  Lidl:
     - lidl
  Edeka:
    - edeka
  Drogerie:
```

(continues on next page)

(continued from previous page)

```
    - drogerie
  Kodi:
    - kodi
  Getraenke:
    - Getraenke Tempel
  Tanken:
    - text
    - esso station
    - aral
    - total tankstelle
    - RK Tankstellen
  Migros:
    - genossenschaft migros

sum_keys:
  - summe
    - gesamtbetrag
    - gesantbetrag
    - gesamt
    - total
    - sum
    - zwischensumme
    - bar
    - te betalen
    - rockgeld
    - rusckgeld
    - rückgeld

    ignore_keys:
      - mwst
      - kg x
      - stkx
      - stk


sum_format: '\d+(\.\s?|,\s?|[^a-zA-Z\d])\d{2}'

item_format: '([a-zA-Z].+)\s(-|)(\d,\d\d)\s'

date_format: '((\d{2}\.\d{2}\.\d{2,4})|(\d{2,4}\/\d{2}\/\d{2})|(\d{2}\/\d{2}\/\d{4}))'
```

CHAPTER

NINE

REVERSE PROXY

To use a reverse proxy, you need to disable *HTTPS* in the receipt parser config. Change this line

```
# Enable https
https: true
```

to

```
# Disable https
https: false
```

After, use this example NGINX configuration and replace *DOMAIN* with your domain and *CERT PATH* with your SSL certificate path.

```
server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name [DOMAIN] [DOMAIN];

        # optional
        access_log /var/log/nginx/[DOMAIN].access.log;
        error_log /var/log/nginx/[DOMAIN].log;

        client_max_body_size 0;
        underscores_in_headers on;

        ssl on;
        ssl_certificate [CERT PATH]; # managed by Certbot
        ssl_certificate_key [CERT PATH]; # managed by Certbot

        ssl_stapling on;
        ssl_stapling_verify on;
        include /etc/nginx/snippets/ssl.conf;


        location / {
                proxy_headers_hash_max_size 512;
                proxy_headers_hash_bucket_size 64;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

(continues on next page)

(continued from previous page)

```
                add_header Strict-Transport-Security "max-age=15768000;␣
↪includeSubDomains;";
                add_header Front-End-Https on;

                # whatever the IP of your receipt server server is
                proxy_pass http://localhost:8721;
        }
}

server {
        listen 80;
        listen [::]:80;
        server_name [DOMAIN] [DOMAIN];
        access_log /var/log/nginx/[DOMAIN].access.log;
        error_log /var/log/nginx/[DOMAIN].80.error.log;
        root /usr/share/nginx/html/[DOMAIN]/;

        location ^~ /.well-known/acme-challenge/ {
            allow all;
            default_type "text/plain";
        }
        location ^~ /.well-known/pki-validation/ {
            allow all;
            default_type "text/plain";
        }
        location / {
            return 403;
        }
}
```

Don't forget to reload your NGINX server, after.