

PROCR



02/04/2025
Développer une interface OCR

Proust Maximilien

Table des matières

PROCR	1
1) Présentation du projet	3
2) Présentation de l'OCR.....	5
2.1) Easy_OCR.....	5
2.2) Doctr.....	5
2.3) Azure OCR.....	6
2.4) Tesseract.....	6
2.5) Pyzbar	7
2.6) Regex	7
2.7) Les limites.....	9
3) L'application	10
3.1) Page d'accueil.....	10
3.2) La page de login.....	10
3.2) La page d'accueil après login	11
3.3) L'import de fichier.....	12
3.4) Base de données	13
3.5) Monitoring.....	13
3.6) Statistiques	13
3.7) La documentation	14
4) Les points d'amélioration	14
Source :	15

1) *Présentation du projet*

L'objectif du projet est de développer une application permettant la reconnaissance optique de caractères (OCR). Un client a exprimé le besoin d'évoluer dans sa procédure de pré-traitement de factures. Pour cela, nous avons donc mis en place une application permettant de stocker dans une base de données les factures. Ensuite, ce client a exprimé le besoin d'étendre les fonctionnalités au-delà du simple stockage de ses factures. Pour cela, nous avons ajouté des fonctionnalités de monitoring et de statistiques permettant un suivi personnalisé de sa base de données afin que cela devienne un vrai outil pour la boîte.

En terme plus techniques, nous avons effectué les tâches suivantes :

- Intégration d'un service OCR avec la bibliothèque Tesseract et des paramètres que nous spécifierons plus tard.
- Intégration de ce service OCR avec FastAPI, en back et front-end.
- Monitoring de l'application avec suivi de l'intégration des factures dans la base de données.
- Statistiques afin d'observer différents paramètres concernant les factures, les clients et les articles dans la base de données.

Pour réaliser ce projet, j'ai effectué un schéma fonctionnel que vous retrouverez à la page suivante.

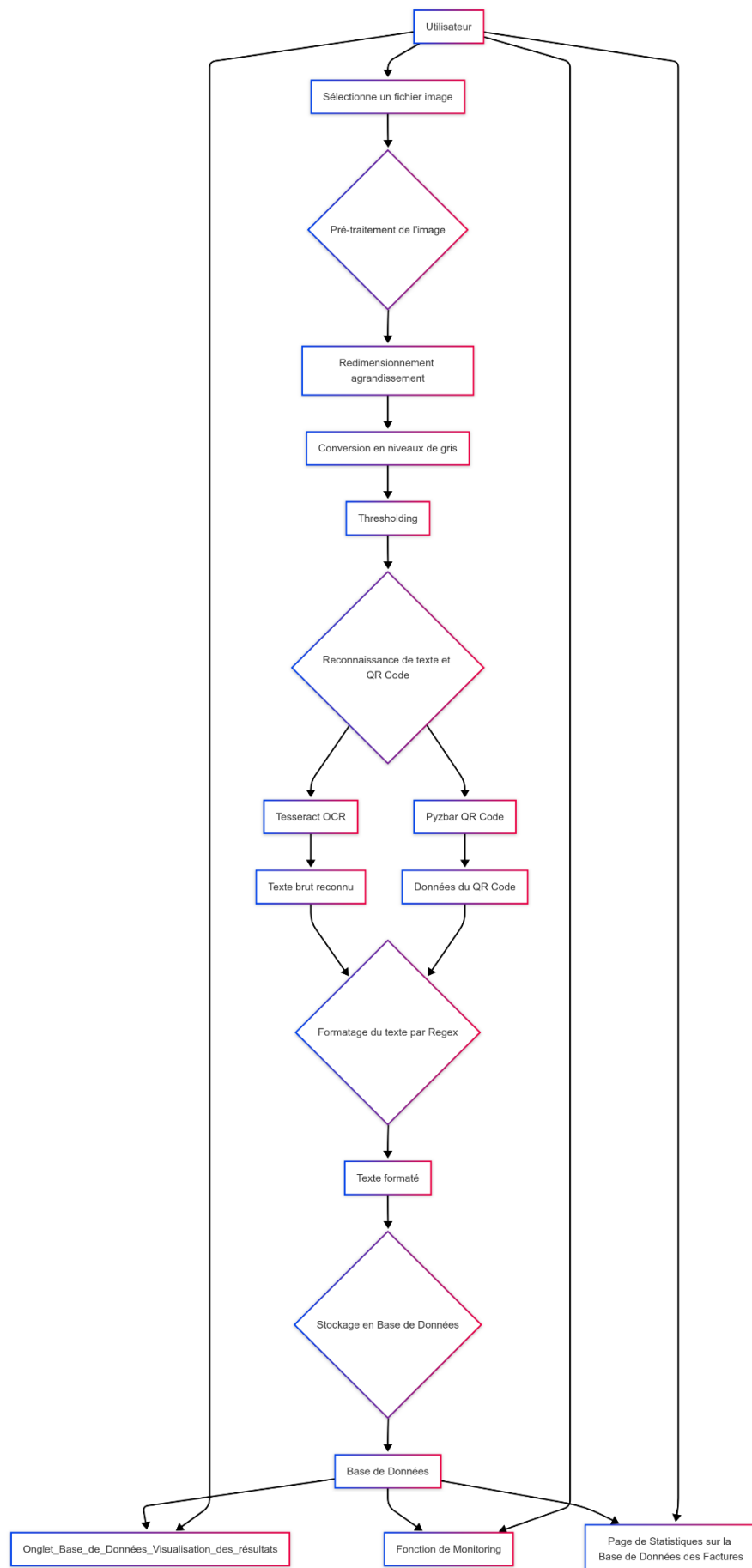


Figure 1 : Schéma fonctionnel de l'application PROCR réalisé avec mermaid

2) Présentation de l'OCR

Pour cette application j'ai testé 4 OCR avant de faire mon choix.

Je vais d'abord vous présenter les 3 OCR et que je n'ai pas choisi en expliquant les raisons du pourquoi. Dans un deuxième temps, je parlerai de la bibliothèque utilisée pour décoder les QR codes et pour finir, je parlerais du processus de sélection et de stockage des informations en base de données.

2.1) Easy_OCR

Dans un premier temps, j'ai effectué des tests avec la bibliothèque easy_ocr. Cette bibliothèque a été créée en 2020 par l'entreprise Jaided AI. Easy_ocr a l'avantage d'être une bibliothèque open-source et est donc gratuite. Malgré des résultats sans aucun paramétrage excepté la langue était plutôt prometteur. Vous pouvez voir ci-dessous les résultats en image de ce test :

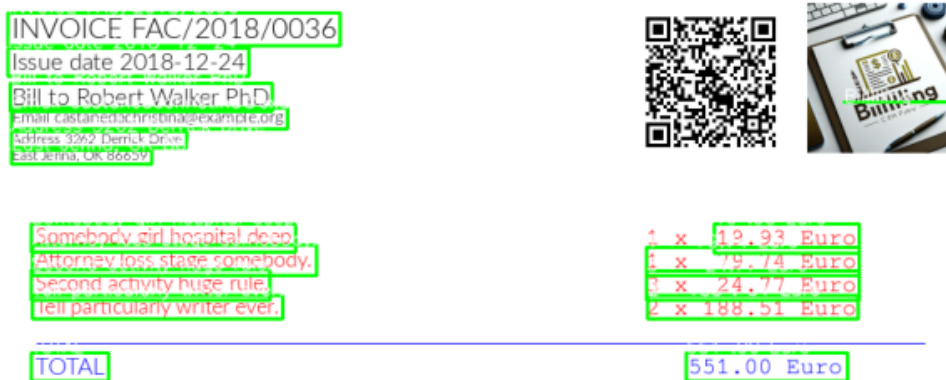


Figure 2 : Test de l'OCR easy_OCR

Vous pouvez observer l'absence de prise en compte des chiffres au niveau des quantités. Cela est un vrai problème et peut-être qu'avec des paramètres cela aurait été efficace mais j'ai préféré tester d'autres technologies.

2.2) Doctr

La deuxième bibliothèque que j'ai testée est Doctr. Cette bibliothèque, également open source proposée par l'entreprise mindee n'a malheureusement donné aucun résultat. Etant donné que je souhaitais faire un premier comparatif sans aucun paramétrage, je suis donc passé à la suite. En effet, je n'avais récupéré aucun texte. Peut-être n'ai-je pas bien compris la doc, ce qui est un premier frein ? Quoi qu'il en soit, j'ai décidé de ne pas perdre de temps supplémentaire avec cette bibliothèque.



Figure 3 : Test de l'OCR Doctr

2.3) Azure OCR

Le dernier OCR non retenu est celui d'azure. Cet OCR n'est pas open source et est donc payant, proposé par Microsoft. Les résultats sont excellents et cela sans aucun paramétrage. Quelques erreurs subsistent mais pourraient être corrigés avec des paramétrages. J'ai décidé de ne pas le conserver car je souhaitais malgré tout une solution gratuite pour le client. Cette bibliothèque pourrait tout à fait être utilisée pour des fichiers qui ne sont pas identiques, ce qui ne correspond pas au besoin de notre client. Vous retrouverez ci-dessous le résultat de l'OCR sans paramétrage.

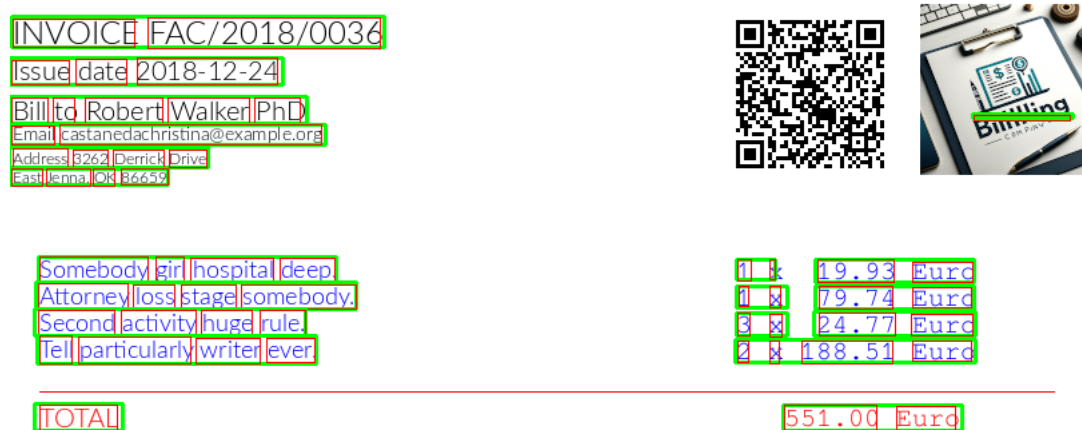


Figure 4 : Test de l'OCR d'azure

2.4) Tesseract

Enfin, l'OCR retenu est la bibliothèque Tesseract. Cette bibliothèque, créée entre 1985 et 1994 aux Etats-Unis. Cette bibliothèque a bien évidemment évolué avec le temps jusqu'à sa dernière version stable en 2021. Cette bibliothèque est open-source, ce qui est un énorme avantage pour notre client et les résultats sont également très bon, ce qui m'a poussé à la conserver. Entre Tesseract et Easy_OCR, le choix ne fut pas forcément facile à faire étant donné la très grande proximité entre les résultats de ces deux OCR.

Tesseract 4 va utiliser un moteur OCR basé sur un réseau neuronal (LSTM) axé sur la reconnaissance de lignes, tout en prenant en charge l'ancien moteur OCR de Tesseract 3 qui lui va permettre reconnaître les motifs de caractères. Cette compatibilité entre Tesseract 3 et 4 est permise grâce au moteur OCR -oem 0. Tesseract reconnaît plus de 100 langues par défaut et prend en charge divers formats d'image, notamment PNG, JPEG et TIFF. Enfin, les formats de sortie peuvent être du texte brut, de l'HTML, du PDF, TSV, ALTO et PAGE.

J'ai donc entamé une démarche d'optimisation des résultats avec la bibliothèque Tesseract. Je suis d'abord parti sur un masque de la photo et du QR code. En effet, la bibliothèque Tesseract me trouvait des caractères dans ces images (ce qui est logique) mais je ne souhaitais aucun bruit. Donc j'ai ajouté

un masque sur ces photos afin d'éviter du bruit inutile qui pourrait perturber la suite des événements. Dans un second temps, j'ai converti l'image en nuance de gris. Pour la conversion en nuance de gris, j'ai effectué un grayscale accompagné d'un thresholding pour récupérer une intensité de pixels en gris et éviter des bruits de fond supplémentaire. En effet, une facture en noire et blanc a des biens meilleurs résultats qu'une photo en couleur. La récupération des caractères était de plus optimale que ce que j'avais sans cette nuance. Enfin, la facture a la particularité d'être de très mauvaise qualité, notamment au niveau des Emails et des adresses qui sont en petit caractère. Cela pose problème pour la lecture et afin d'améliorer sensiblement la reconnaissance, j'ai effectué un redimensionnement de la facture en multipliant la taille de celle-ci par deux et donc la police par deux également.

2.5) Pyzbar

Une fois l'ensemble de ces paramétrages effectués pour Tesseract, il a fallu implémenter une autre bibliothèque. En effet, bien que Tesseract est une très bonne bibliothèque, elle ne peut pas analyser les QR code présent sur ma facture. J'ai donc utilisé la bibliothèque Pyzbar. Pyzbar est une bibliothèque open source écrite en C et conçu pour scanner et interpréter des codes-barres à partir de diverses sources (caméras, fichiers, vidéos). L'avantage de cette bibliothèque est son fonctionnement sur l'ensemble des systèmes d'exploitation. Pyzbar va utiliser un wrapper (un pont entre un langage de programmation et une bibliothèque) qui va permettre de scanner des QR codes et codes-barres. Il va chercher dans le fichier une image, va la traiter et appeler Zbar pour la reconnaissance de ces images et ainsi les décoder par la suite pour fournir les informations à l'utilisateur.

Cette bibliothèque étant parfaitement fonctionnel, je n'ai pas cherché d'autres QR-OCR.

2.6) Regex

Une fois choisi mes deux outils OCR, l'un permettant de transcrire le texte dans mon image en texte brut et l'autre en décodant le QR code, je pouvais passer à l'analyse des informations recueillies. Lorsque je regarde les éléments de la facture, énormément d'informations sont importantes et doivent être stockés. La première étape a été de réfléchir à la façon dont j'allais stocker ces données. Je suis parti sur une création de trois tables. La première table concerne les factures. Les informations que je souhaitais conserver à propos des factures sont les suivantes :

- Le nom de la facture, indispensable et est même la clé primaire de ma table, permettant d'individualiser chacune d'entre elles.
- La date de la facture, permettant un suivi temporel des factures au cours du temps, permettant d'observer des métriques sur les ventes par années, mois, semaines...
- Le total de la facture, permettant un suivi indispensable afin de calculer des métriques autour du chiffre d'affaires, panier moyen...

- L'email de l'acheteur, permettant un lien avec une autre table, l'email est une clé unique, en effet il est impossible d'avoir deux mails identiques. Cela permet un suivi individualisé du client.

La deuxième table que j'ai créée était pour l'utilisateur. En effet, il est indispensable d'avoir des informations sur nos clients afin de pouvoir faire des analyses de marché permettant d'augmenter le chiffre d'affaires ou de cibler la clientèle à contacter.

Pour cela, dans cette table j'ai recueilli les éléments suivants :

- L'email de l'acheteur, comme mentionné précédemment, permettant le lien avec les factures et est ma clé primaire et unique.
- Le nom de la personne, information supplémentaire à l'email.
- Le genre, bien que peu éthique de nos jours, peu permettre de cibler une clientèle particulière en fonction des articles.
- Le numéro et la rue de l'acheteur, permettant entre autres l'analyse de la zone de chalandise.
- La ville et le code postal, de même que le numéro et l'acheteur, permettent de faire des analyses poussées de la zone de chalandise.
- La date d'anniversaire de l'acheteur, permettant ainsi de proposer des offres spéciales pour l'anniversaire du client par exemple.

J'ai ensuite décidé de créer une dernière table à destination des articles. Cela permet en effet à notre client de pouvoir analyser plus précisément ses ventes et d'ainsi pouvoir calculer des métriques indispensables au bon fonctionnement de son magasin. J'ai recueilli les éléments suivants :

- Le nom des factures, permettant de relier la facture aux articles et de vérifier par ailleurs si l'OCR a bien fonctionné via un calcul de somme.
- Le nom de l'article, qui avec les factures forment une clé primaire. Le nom de l'article permet de suivre via le nom du produit les ventes de celui-ci.
- La quantité vendue, évidemment indispensable également.
- Le prix de l'article, indispensable pour calculer des métriques.

Une fois que j'ai imaginé sur papier ses tables, il ne me restait donc qu'à effectuer des Regex afin de récupérer les informations voulues et uniquement les informations souhaitées !

Le format récupéré du QR code et de l'OCR sont sous texte brut. Il m'était donc facile de récupérer les informations que je souhaitais via des systèmes de retour à la ligne présent dans mon OCR. Je m'en suis énormément servi afin de réussir à récupérer les informations souhaitées. Bien que purement technique, je vous mets ci-dessous à titre informatif les regex utilisés par Tesseract.


```

nom_facture = re.findall(r'FAC/\d{4}/\d+', text)
date_facture = re.findall(r'date (\d{4}-\d{2}-\d{2})', text)
nom_personne = re.findall(r'Bill to ([^\n]+)', text)
email_personne = re.findall(r'Email ([^\n]+)', text)
rue_num_personne = re.findall(r'Address ([^\n]+)', text)
ville_personne = [v.strip() for v in re.findall(r'Address [^\n]+\n([\w\s-]+), \w{2} \d{5}', text)]
code_postal_personne = re.findall(r', (\w{2} \d{5})', text)

```

Ci-dessous, les regex que j'ai utilisé afin de récupérer les informations du QR code :

```

nom_facture = re.findall(r'FAC/\d{4}/\d+', regex)
date_facture = re.findall(r'DATE:(\d{4}-\d{2}-\d{2})', regex)
genre = re.findall(r'CUST:(\w)', regex)
date_anniversaire = re.findall(r'birth (\d{4}-\d{2}-\d{2})', regex)

```

Ces informations, une fois récupérées sont transférées dans la base de données selon les tableaux établis précédemment.

2.7) Les limites

Après avoir transférés les factures, j'ai pu observer la table de plus près afin de trouver les limites du système, car oui, malgré de nombreux paramétrages, il y a des limites à mon OCR qui nécessiterait d'être perfectionné, tout comme les factures, qui mériteraient quelques changements.

La première limite, concerne les erreurs de retranscription des chiffres dans la quantité et/ou le total. Après analyse, je me retrouve avec 1,7% d'erreur soit 85 erreurs sur 5115 factures analysées. C'est un total tout à fait honorable mais insatisfaisant pour le client. Il faudra trouver une solution pour y remédier. La deuxième limite concerne toujours la retranscription d'erreurs dans les mails. Après observation à l'œil, certains caractères sont difficilement retrouvés par l'OCR et se transforme en d'autres caractères. Il m'est impossible de quantifier cela mais c'est un réel problème ! La troisième et dernière limite concerne le formatage ville et code postal qui n'est pas toujours bien reconnu par le regex. Il y a plusieurs raisons pour cela mais la première est le mauvais formatage dû à l'émetteur de la facture. Il faudrait donc que celui-ci soit beaucoup plus vigilant afin d'éviter ce genre d'erreur qui correspond tout de même à environ 12% d'erreurs dans la base de données. Enfin, la taille de la police constitue la dernière limite. On a besoin d'une facture avec une taille de police acceptable et non des petits bouts de texte quasiment illisible par l'OCR. Dans l'ensemble, je reste satisfait par cela et l'évolution de nos pratiques (côté dev et côté client) pourraient permettre de réduire à néant les erreurs.

3) L'application

3.1) Page d'accueil

La construction de la page d'accueil est très simple. L'objectif n'est pas d'en faire trop. Si l'utilisateur souhaite une démo, il clique dessus et cela ouvre sa boîte mail pour me contacter directement. L'autre bouton est le login, permettant à l'utilisateur de se connecter. Ci-dessous, vous trouverez la page d'accueil.



Figure 5 : Page d'accueil de l'application PROCR

3.2) La page de login

La aussi très simple, l'utilisateur doit rentrer son mail ainsi que son mot de passe. S'il n'a pas de compte, il suffit de s'inscrire sur la page renvoyée. Une utilisation très simple. Pour ce système, j'ai utilisé l'authentification avec bcrypt ainsi que le mot de passe haché avec la bibliothèque passlib. Les données utilisateurs sont enregistrés dans la base de données mais le mot de passe est crypté et je ne peux donc pas le voir. Ce système d'authentification est très efficace. J'ai pu ajouter un système de tokenisation qui permet de limiter le temps de connexion que j'ai fixé à 30 minutes avant de redevoir se logger mais la tokenisation permet également l'accès aux pages suivantes. En effet, il est impossible d'avoir accès aux autres pages hors tokenisation.

A la page suivante, vous trouverez la capture d'écran du login de mon application.



Connexion

Email

maximilien_proust@hotmail.fr

Mot de passe

Se connecter

Pas de compte ? Inscrivez-vous !

Figure 6 : Page d'authentification de l'application PROCR

3.2) La page d'accueil après login

Après le login, vous arriverez sur une page d'accueil classique avec implémentation d'une barre de navigation. Il faut donc faire un choix dans cette barre de navigation pour accéder aux autres pages et ainsi faire ce que l'on souhaite. Afin de fluidifier l'expérience utilisateur, la barre de navigation change de couleur lorsqu'on survol une option. Option supplémentaire : Lorsque vous êtes dans une page de la barre de navigation, celle-ci change de couleur et permet à l'utilisateur de le situer dans l'application. Ci-dessous, vous trouverez cette page :



Figure 7 : Visualisation de la page d'accueil après login avec la barre de navigation

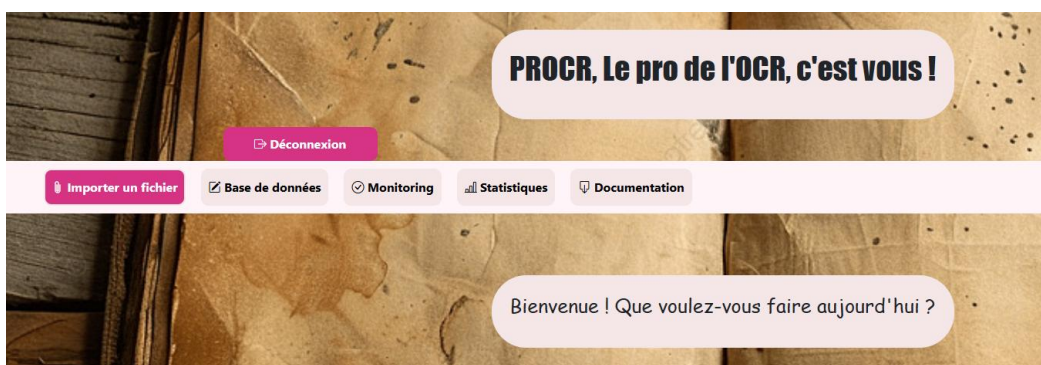


Figure 8 : Visualisation de la barre de navigation avec le changement de couleur au survol

3.3) L'import de fichier

Cette page va en conduire à d'autres qui vont permettre d'effectuer l'OCR. Afin de fluidifier l'expérience utilisateur, pas de texte superflu et juste un bouton « Importer votre fichier » qui ne laisse de place au doute. Il suffit alors de cliquer et vous devez choisir parmi vos fichiers quelles factures vous souhaitez traitées aujourd'hui.

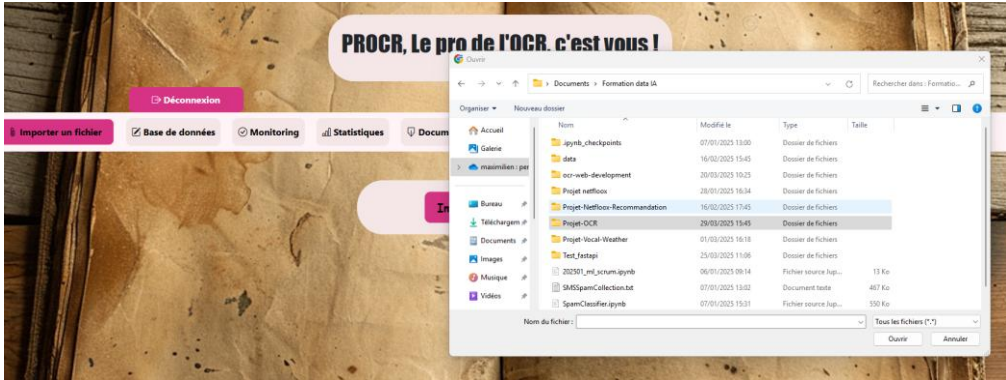


Figure 9 : L'import de fichier

Malheureusement, dans cette première version vous devez traiter les factures une par une. Une fois la facture choisie, vous êtes transféré dans une autre page qui va vous montrer l'image pour vérifier que c'est bien celle-ci que vous souhaitez traiter.

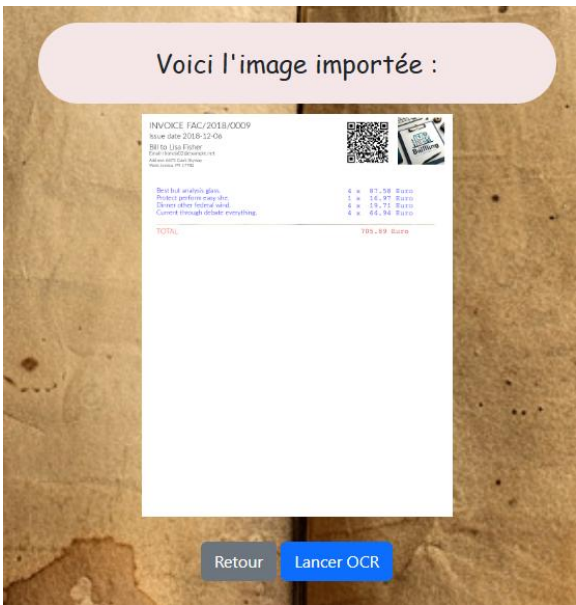


Figure 10 : Visualisation de l'image choisie

Vous pouvez voir sur le cliché ci-dessus qu'il y a un bouton retour qui permet de changer en cas d'erreur de fichier. Vous pouvez également voir le bouton Lancer OCR qui va permettre d'effectuer l'OCR, mais également l'enregistrement en base de données. Lorsque vous appuyez sur ce bouton et qu'il n'y a pas d'erreur, un petit message apparaît « Résultat OCR : Fonctionnel ».

3.4) Base de données

L'onglet suivant dans la barre de navigation est la base de données. La base de données permet de visualiser les 3 tables enregistrés dans l'étape 2. Vous pouvez également chercher des informations sur une facture en particulier, un mois ou une année mais vous pouvez uniquement visualiser. Aucune modification n'est possible, c'est juste un outil de visualisation.



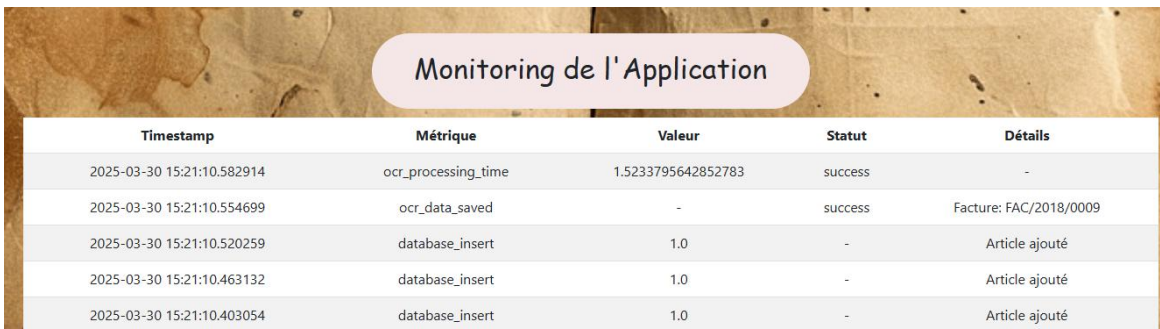
Nom Facture	Nom Article	Quantité	Prix
FAC/2017/0001	Edge so crime share.	4	12.18
FAC/2017/0001	Thank do article especially.	1	67.86
FAC/2017/0001	Incredible delicious article featured	3	787.000

Figure 11 : Base de données

Un petit problème cela dit, j'ai utilisé une template pour le tableau mais cette template est en anglais d'où le Show entries et Search. C'est un problème mineur qui pourrait être régler lors d'une deuxième version plus aboutie.

3.5) Monitoring

L'onglet de Monitoring va permettre de suivre l'entrée des factures. Cet onglet va permettre de suivre si les factures ont bien été ajouté dans les bonnes tables et le temps que cela a mis.



Timestamp	Métrique	Valeur	Statut	Détails
2025-03-30 15:21:10.582914	ocr_processing_time	1.5233795642852783	success	-
2025-03-30 15:21:10.554699	ocr_data_saved	-	success	Facture: FAC/2018/0009
2025-03-30 15:21:10.520259	database_insert	1.0	-	Article ajouté
2025-03-30 15:21:10.463132	database_insert	1.0	-	Article ajouté
2025-03-30 15:21:10.403054	database_insert	1.0	-	Article ajouté

Figure 12 : Monitoring des factures

3.6) Statistiques

Cet onglet est la page qui va intéresser le client. En effet, elle va présenter des statistiques autour de la base de données permettant à l'utilisateur de pouvoir se servir de ses données chiffrées pour prendre des décisions stratégiques intéressantes. Bien évidemment, cette section reste encore largement améliorable à la demande du client. Les statistiques présentés dans cette version 1 sont des statistiques basiques afin de ne pas surcharger la page. Elle va présenter dans l'ordre :

- Le nombre d'utilisateurs uniques
- Le nombre de facture avec un total nul
- Le panier moyen

- Le chiffre d'affaires total
- Le nombre de factures dans la base de données
- Les clients les plus prolifiques
- Les clients les plus réguliers
- Les villes qui attirent le plus de clients
- Les articles les plus populaires

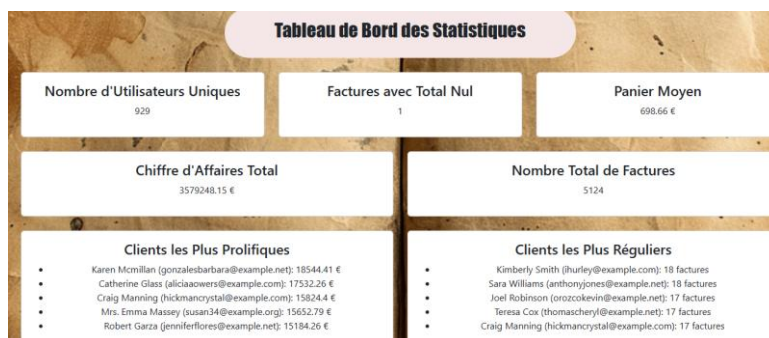


Figure 13 : Statistiques de la base de données

3.7) La documentation

L'onglet de documentation permet d'expliquer à l'utilisateur comment fonctionne l'application pour un non initié à l'informatique.

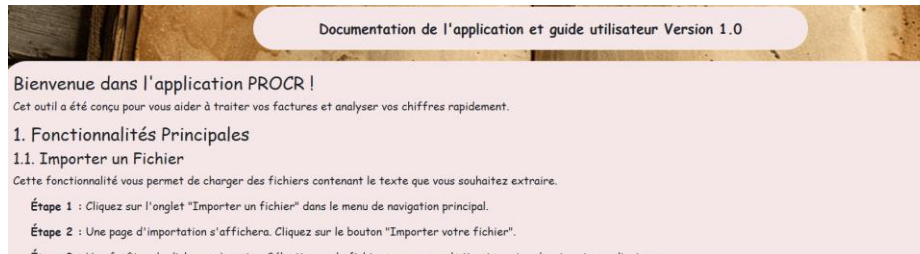


Figure 14 : Documentation de l'application OCR

4) Les points d'amélioration

Evidemment, tout dans l'application n'est pas parfait. Il reste beaucoup à améliorer dans celle-ci. On va commencer par le début avec le login. En effet, dans ma version 1, lorsque vous essayez d'accéder à une page où une authentification est nécessaire, vous n'aurez pas d'écran utilisateur, juste une page avec une phrase écrite en anglais.

Ensuite pour l'OCR, comme listé précédemment, il y a encore des points de progrès à réaliser pour atteindre les 100% de réussite dans le traitement d'image à texte. Une optimisation est tout à fait réalisable mais nécessiterait plus de temps pour que cela soit réalisé. Enfin, une possibilité de modifier la base de données devrait être mise en place afin de pouvoir gérer les erreurs d'OCR et ainsi minimiser les erreurs dans la base de données.

Source :

Mermaid : <https://www.mermaidchart.com/>

Easy_OCR : <https://github.com/JaidedAI/EasyOCR>

Doctr : <https://github.com/mindee/doctr>

Azure OCR : <https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/quickstarts-sdk/image-analysis-client-library-40?tabs=visual-studio%2Clinux&pivots=programming-language-python>

Tesseract : <https://github.com/tesseract-ocr/tesseract>

Pyzbar : <https://pypi.org/project/pyzbar/>

Bcrypt : <https://pypi.org/project/bcrypt/>

Passlib : <https://pypi.org/project/passlib/>