

Application CoCo

IMT-Nord-Europe

DONNE Dylan

DONNE Dylan - IMT-Nord-Europe

Table des Matières

1. Le projet	3
1.1 L'objectif du projet	3
1.2 Participant(s)	3
2. Les choix technologiques	4
2.1 Les relations entre les services	4
2.2 Partie Serveur	6
2.3 Partie Client	9
3. La détection des pièces	12
3.1 Une pièce	12
3.2 Détection de cercle	13
3.3 Détection de contour	15
3.4 Les complications	16
4. Amélioration de la détection	18
4.1 Détection de la couleur	18
4.2 Détection de la taille	19
4.3 Détection des chiffres sur les faces avant	20
5. Amélioration du service	21
5.1 Micro Service de stockage	21
5.2 Micro Service de détection	22
6. Amélioration du client	23
6.1 Restreindre la distance maximale de la photo	23
6.2 Importer des photos depuis la Galerie	25
7. Amélioration du serveur	26
7.1 Amélioration de la disponibilité	26
7.2 Amélioration de la sécurité	27
8. Documentation Utilisateur	28
8.1 Premier pas	28
8.2 Les différents menus	31
8.3 Les conseils d'utilisation	33
8.4 Exemple d'utilisation	38
9. Glossaire	44
9.1 Définitions	44

1. Le projet

1.1 L'objectif du projet

L'application Coins-Counter (CoCo) a pour but de permettre à un utilisateur de calculer la somme des pièces de monnaie présente sur une photo réaliser à partir d'un smartphone.

L'idée principale de ce projet était de mettre en oeuvre les compétences acquises lors des différents cours suivi dans la formation d'Ingénieur en Informatique et Télécommunication (parcours Alternance) de l'IMT Nord Europe.

1.2 Participant(s)

1.2.1 Partie Web Serveur

✓ **DONNE Dylan**

1.2.2 Partie Client

✓ **DONNE Dylan**

1.2.3 Partie Reconnaissance d'image

✓ **DONNE Dylan**

1.2.4 Détails des participants



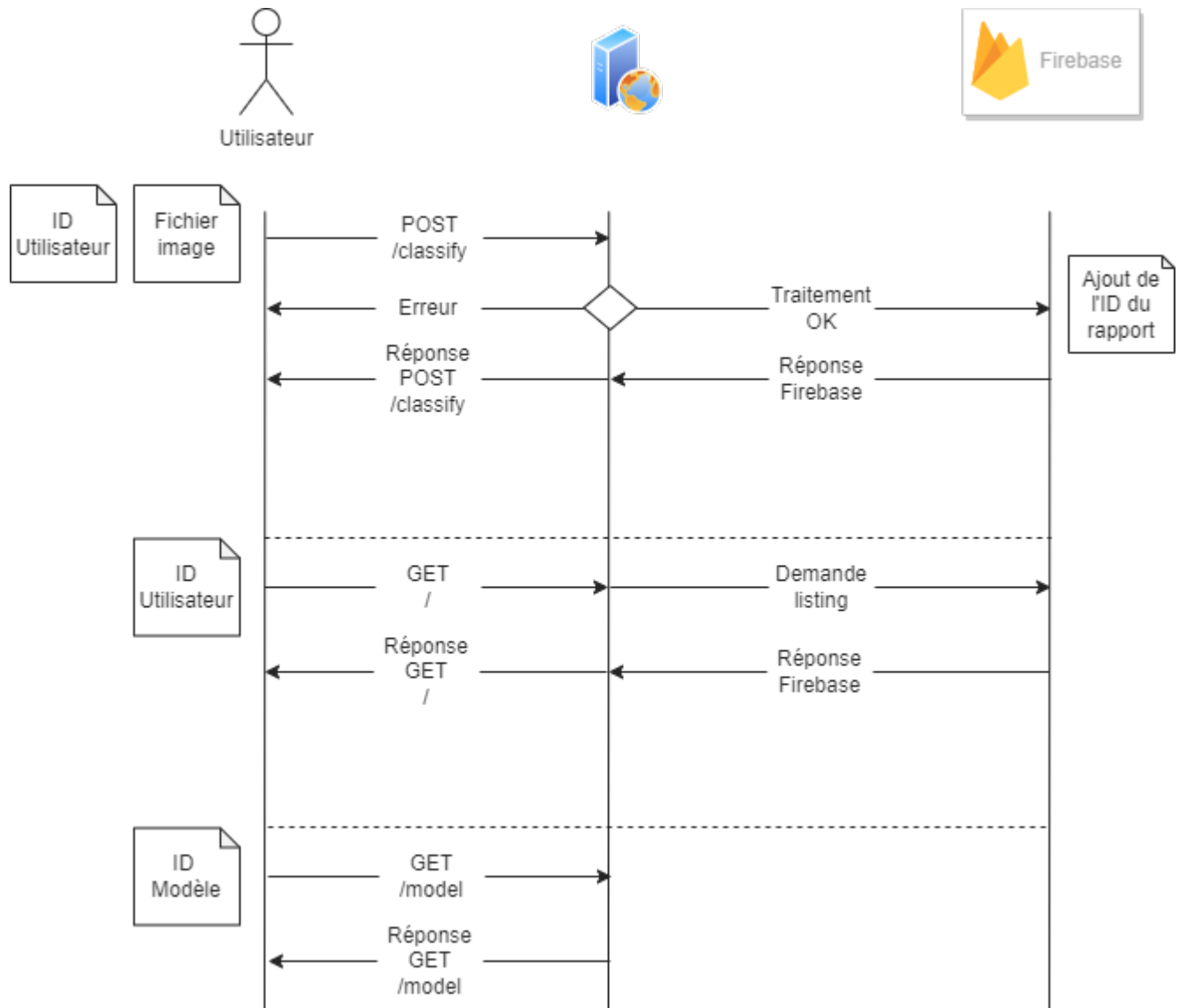
DONNE Dylan

- [Profil GitHub](#)
- [Profil CodinGame](#)

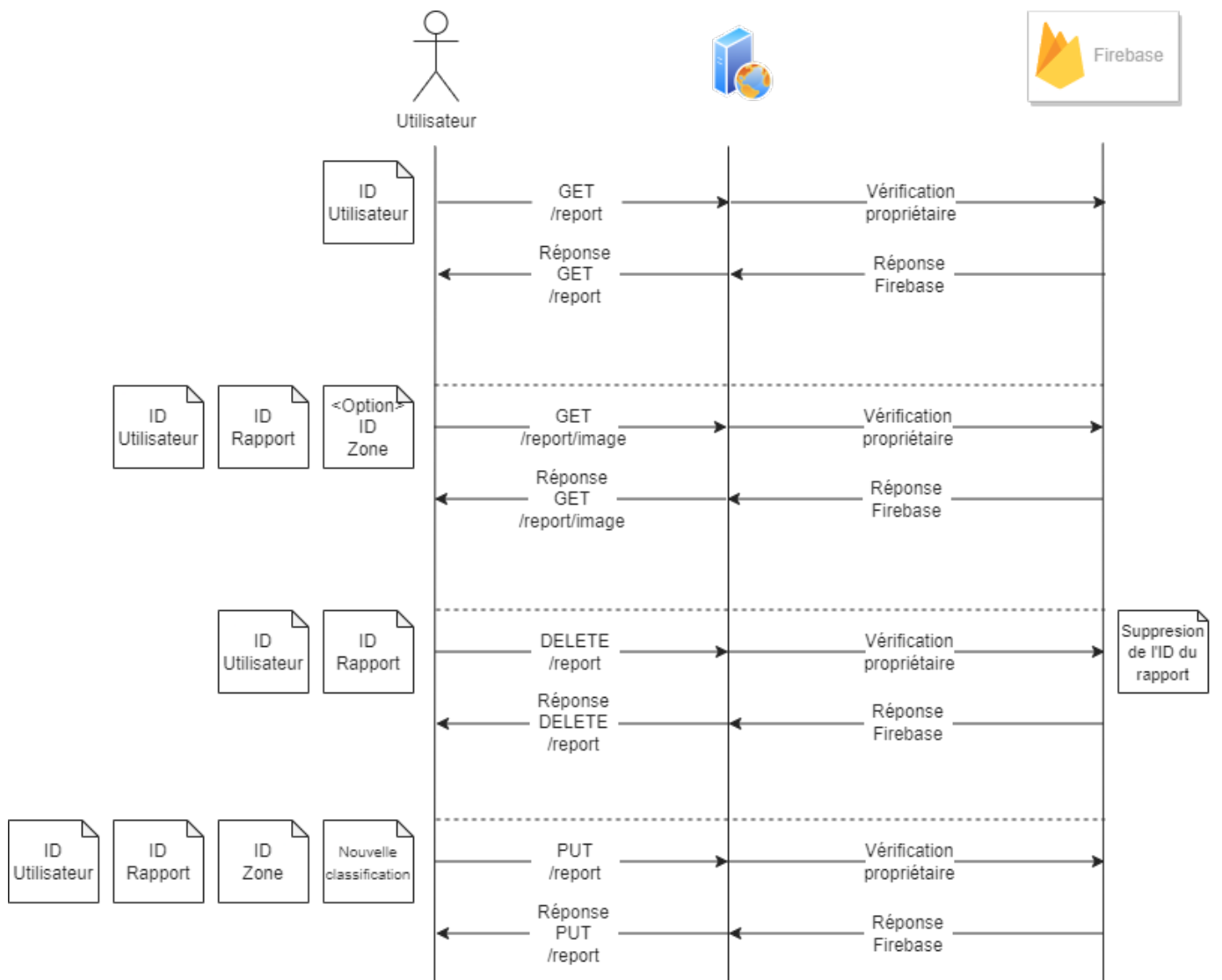
2. Les choix technologiques

2.1 Les relations entre les services

2.1.1 Les routes spécifiques

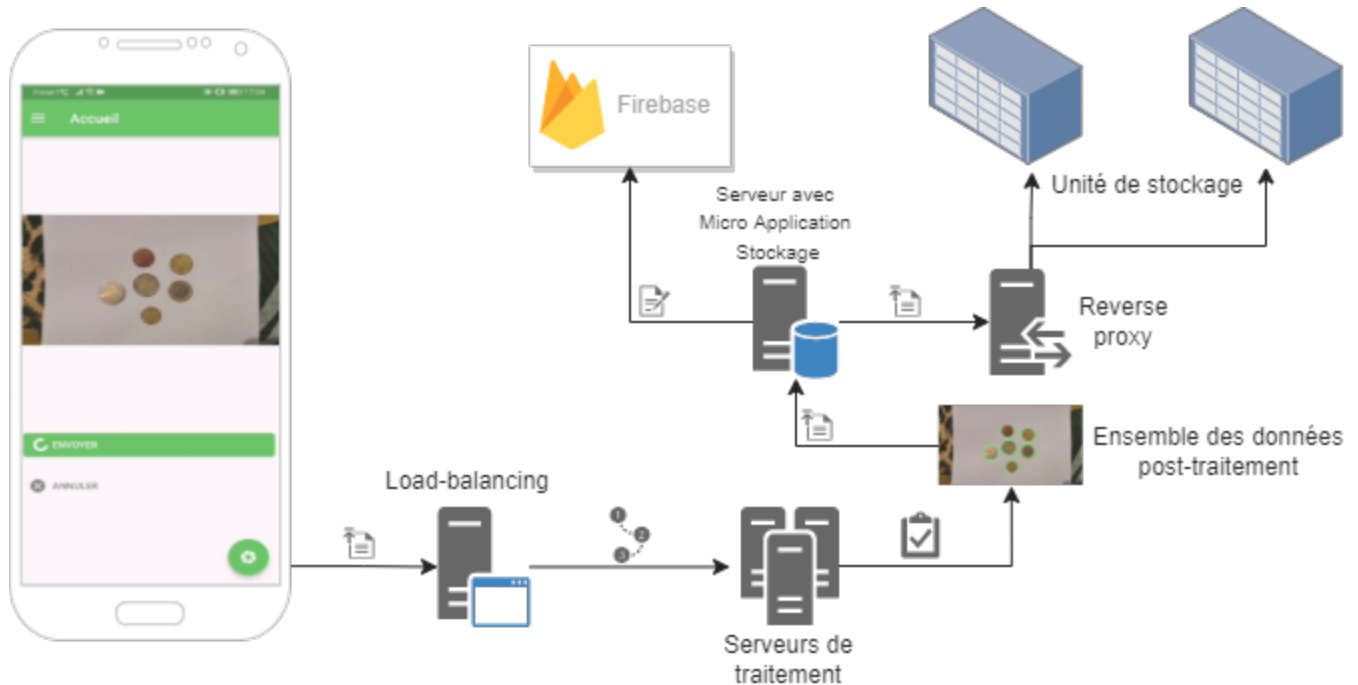


2.1.2 Les routes concernant les rapports



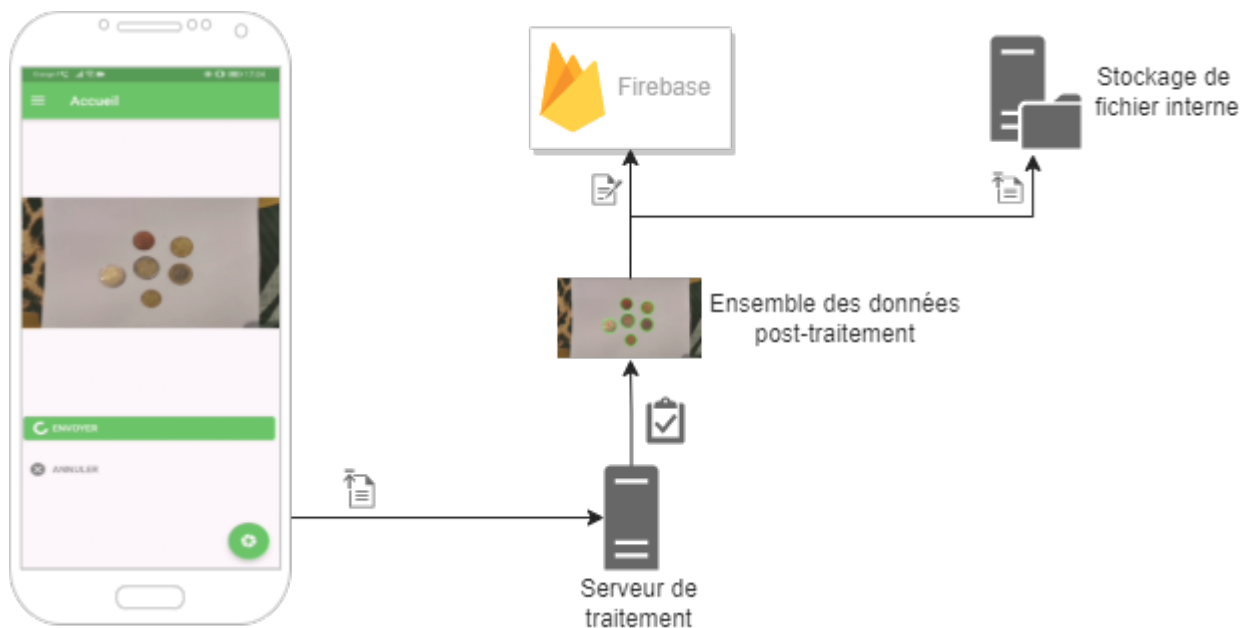
2.2 Partie Serveur

2.2.1 Architecture imaginée



L'architecture ci-dessus a été imaginée pour une application en production, afin d'avoir une disponibilité maximale, un load-balancing ainsi qu'un ensemble de serveurs seraient mis à disposition. Pour garantir une maintenabilité et une sécurité optimale, les unités de stockages sont séparés et misent derrière un reverse proxy.

2.2.2 Architecture mise en place



Par manque de moyen et de temps, une architecture plus simple et plus réaliste a été mis en oeuvre, voici sa représentation.

Les outils utilisés

Pour la partie serveur, le choix s'est porté sur le framework [Flask](#). La reconnaissance d'image fonctionnant en Python, la liaison entre les requêtes du serveur et le système d'analyse était simple à mettre en place.

Firebase est utilisé pour la partie cliente, il permet d'authentifier et de stocker les données (identifiants des rapports d'analyse) dans un cloud, et de les récupérer.

LE MODULE D'INTELLIGENCE ARTIFICIELLE

L'API contient le module d'intelligence artificielle qui était de base destiné aux serveurs de traitement, la liaison est donc direct. Ce module fonctionne grâce à [OpenCV](#), [TensorFlow](#) et [Keras](#).

La détection des pièces est réalisé à l'aide de la fonction [HoughCircles\(\)](#) du module [OpenCV](#), une fois les pièces détectées (cercle(s) détecté(s) dans l'image), elles sont découpées et envoyées au module de Classification qui procède à l'analyse avec [Keras](#). Une fois la classification faite, le service récupère et formate les données proprement avant l'envoi au client, puis sauvegarde le rapport généré dans le stockage grâce à l'identifiant unique du rapport et [Firebase](#).

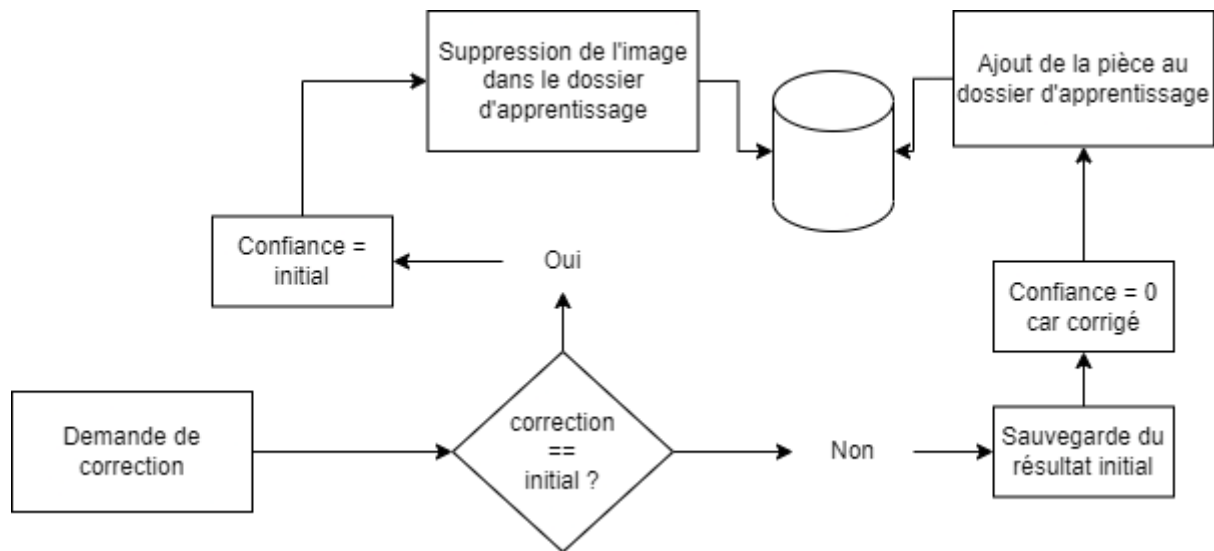
Le module [Firebase](#) est utilisé par l'API uniquement pour stocker les identifiants des rapports pour un utilisateur précis, ce dernier est identifié dans la requête par son Identifiant d'Utilisateur Firebase.

Fonctionnalité

L'API a été conçu pour fonctionner en symbiose avec le client CoCo réalisé dans le cadre du projet, elle comporte notamment diverses fonctionnalités, par manque de temps certaines ne sont pas disponibles sur l'API en production:

- ✓ Envoi d'une demande de traitement
- ☐ Envoi d'une demande de traitement depuis la Galerie
- ✓ Edition et Correction des résultats
- ✓ Une notation effectuée sur les réussites et erreurs
- ✓ Une gestion de modèle d'intelligence artificielle
- ✓ Historique des demandes
- ✓ Demande de synchronisation
- ✓ Suppression de demande de traitement
- ✓ Sauvegarde des images et données de traitement
- ☐ Seconde couche de vérification sur la détection
- ✓ Apprentissage autonome paramétrable

Le serveur est capable d'apprendre automatiquement selon les erreurs retournées par un humain via l'application, lorsque le nombre d'erreurs devient élevé (initialement configuré à 500), il réalise un nouvel apprentissage pour essayer de corriger ses erreurs. Lorsqu'une correction est effectuée par un utilisateur sur l'application, le serveur comprend donc qu'il a fait une erreur et ajoute donc cette image dans un dossier contenant toutes les images pour le prochain apprentissage, s'il y a une nouvelle correction sur la même erreur, celle-ci se voit déplacé de dossier ou supprimé si ce n'était finalement pas une erreur.



Schéma

2.3 Partie Client

2.3.1 Les outils utilisés

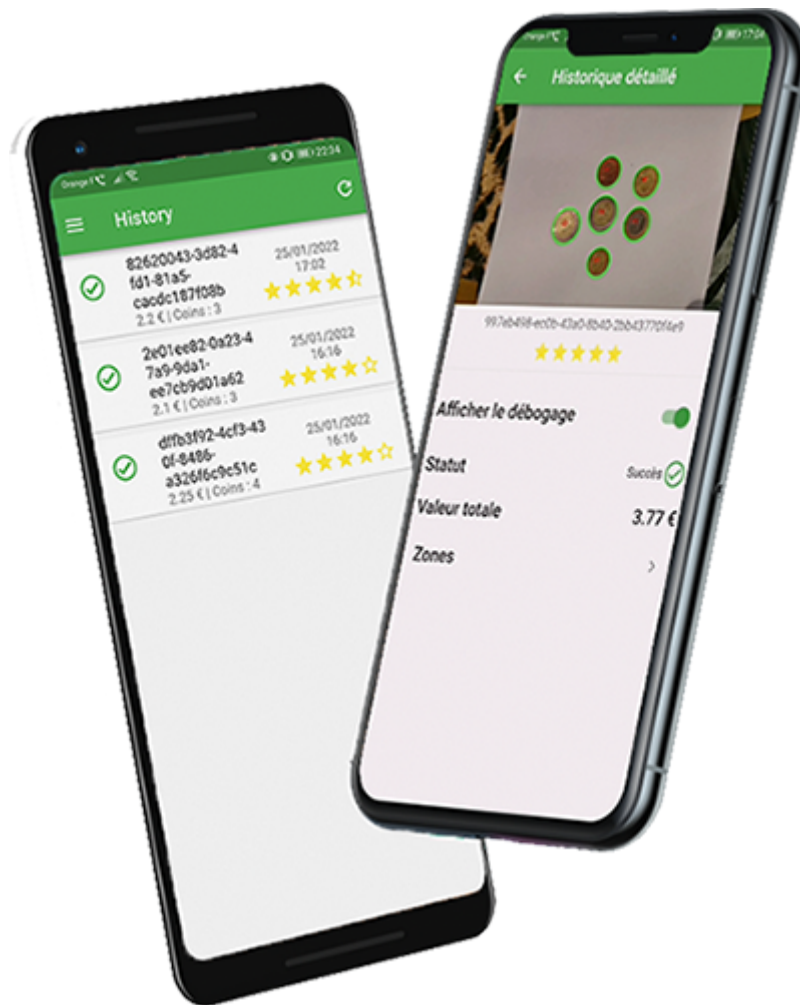
Pour la partie client, le choix s'est porté sur [Flutter](#), qui permet un déploiement multi-platform. L'application est conçu pour supporté plusieurs langages dès lors qu'elles sont implantés grâce au package [Flutter Localization](#).

Cette dernière a été conçu pour fonctionner en symbiose avec le serveur CoCo réalisé dans le cadre du projet, elle comporte notamment diverses fonctionnalités, par manque de temps certaines ne sont pas disponible sur l'application en production:

- ✔ Connexion et Inscription
- ✔ Traduction à la volé
- ✔ Page Paramètres
- ✔ Modification de l'adresse du serveur API
- ✔ Page Tutoriel
- ✔ Envoi d'une demande de traitement
- ☐ Envoi d'une demande de traitement depuis la Galerie
- ✔ Edition et Correction des résultats
- ✔ Une notation effectué sur les réussites et erreurs
- ✔ Demande d'information sur le modèle utilisé
- ✔ Historique des demandes
- ✔ Demande de synchronisation
- ✔ Sauvegarde en local
- ✔ Suppression de demande de traitement

2.3.2 Développement de l'application

Le développement de l'application a été réalisé en utilisant le framework [Flutter](#).



La partie client est composée de plusieurs pages, qui sont les suivantes:

- ✓ Page de connexion et d'inscription
- ✓ Page de paramètres
 - Paramètres générales de l'application
 - Paramètres de l'API
- ✓ Page de tutoriel
- ✓ Page d'accueil (envoi de demande de traitement)
- ✓ Page d'historique des demandes
 - Page de détail d'une demande
 - Page de détail de l'analyse
 - Page de correction des résultats
 - Page d'information sur le modèle utilisé

Design et UX

Le design de l'application a été réalisé de manière à ce que l'utilisateur puisse naviguer facilement dans l'application tout en gardant un maximum de contrôle sur son utilisation.

De nombreuses fonctionnalités ont été ajoutées à l'application, comme la gestion de la langue, la gestion de la synchronisation, la gestion de l'historique, la gestion du choix de serveur, etc.

Toutes ces fonctionnalités ont été pensées pour être facilement accessibles à l'utilisateur, et ainsi permettre une utilisation optimale de l'application. L'implémentation de ces dernières a requis de nombreuses recherches et de tests. Pour avoir une expérience utilisateur optimale, des gestes habituels ont été ajoutés, comme la suppression depuis l'historique en restant appuyé.

Connexion et inscription

Pour la partie client, la connexion et l'inscription sont réalisées en utilisant le package [Firebase](#). Qui permet de gérer les utilisateurs, les mots de passe, les notifications, etc. Dans le cadre de l'application, les utilisateurs sont identifiés par leur adresse email.

Traduction à la volé

Pour la partie client, la traduction à la volé est réalisée en utilisant le package [Flutter Localizations](#). Ainsi, les traductions sont réalisées en utilisant des fichiers de traduction via le package [intl](#), qui permet de gérer les langues. La configuration de la langue est définie par défaut dans le fichier de configuration `l10n.yaml`.

Grâce à cette fonctionnalité, l'application peut être traduite à la volé ce qui permet d'avoir une application multi-langues.

Sauvegarde en local

Le client dispose d'une fonctionnalité de sauvegarde en local, qui permet de sauvegarder les données de l'application. Cette fonctionnalité est réalisée en utilisant le package [sqflite](#).

Grâce à cette fonctionnalité, l'application obtient un fonctionnement indépendant, les données de l'application sont disponibles à tout moment, y compris en cas de coupure de connexion.

Compression

Toutefois, il est important de noter que les données sauvegardées sont compressées, ce qui permet de réduire la taille de la base de données au détriment du visuel.

3. La détection des pièces

3.1 Une pièce



Pièce de 2 €

La pièce ci-dessus a été prise sous deux environnements lumineux différents, la différence entre ces deux images démontre la complexité d'une analyse de ces images.

3.1.1 Caractéristiques

Une pièce peut être défini de plusieurs manières :

- Selon la face
 - Pile :
 - ✓ Sa valeur numéraire
 - Face :
 - ✓ Son symbole
- De manière générale
 - ✓ Son matériaux
 - ✓ Sa couleur
 - ✓ Sa taille
 - ✓ Sa forme

3.2 Détection de cercle

3.2.1 Méthode de détection de cercle

La détection des pièces est réalisé à l'aide de la fonction `HoughCircles()` du module `OpenCV`.

Pour utiliser cette méthode, l'image nécessite un prétraitement de type `grayscale` et `blur`.

```
gray = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)
gray = cv2.equalizeHist(gray)
blur = cv2.GaussianBlur(gray, (19, 19), 0)
```

La fonction `equalizeHist()` permet de normaliser l'histogramme de gris, ce qui permet de réduire les variations d'intensité de l'image causé par la luminosité.

```
circles = cv2.HoughCircles(
    blur, # source image
    cv2.HOUGH_GRADIENT, # type of detection
    1,
    50,
    param1=100,
    param2=30,
    minRadius=5, # minimal radius
    maxRadius=180, # max radius
)
```

Grâce à la fonction `HoughCircles()`, on récupère les cercles détectés sous forme de tableau de coordonnées (x, y, r) qui représente le centre du cercle (x, y) et sa taille (radius). Avec ces éléments on peut ensuite déterminer les pièces détectées et les découper pour obtenir les images de pièces séparées.



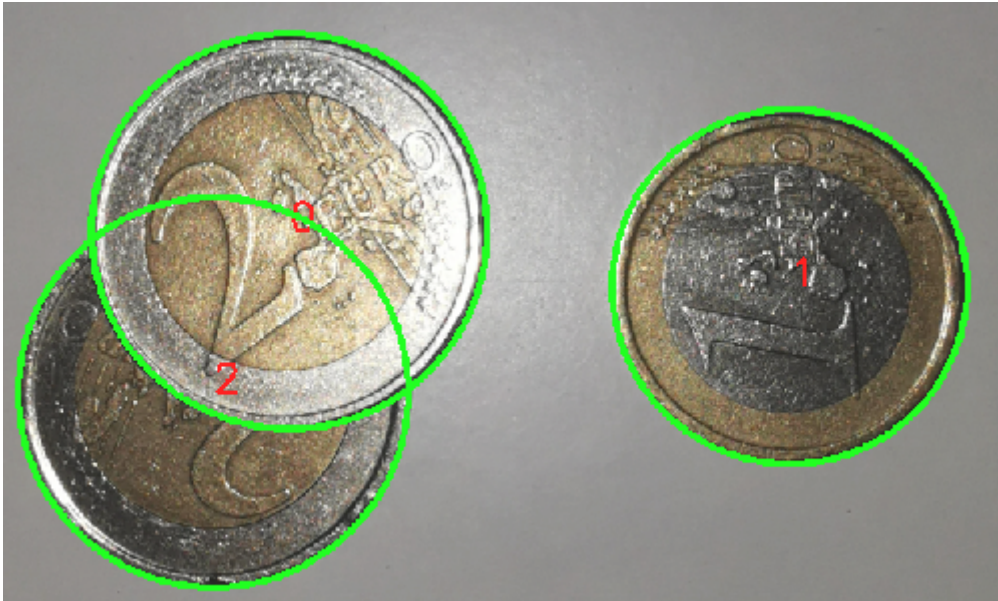
Détection réussie

i Détection

La méthode de détection de cercle est en réalité une méthode de détection d'arcs de cercle.

Ce qui permet notamment la détection de pièces qui sont coupées ou obstruées par des autres pièces ou un décors.

La détection de pièces obstruées est un cas complexe, la méthode `HoughCircles()` permet de détecter les arcs de cercle, contrairement à la méthode de détection de contours qui pourrait détecter des contours simples et considérés comme une seule et même pièce.



Détection complexe réussie

3.3 Détection de contour

3.3.1 Méthode de détection de contour

La détection des pièces est réalisé à l'aide de la fonction `findContours()` du module `OpenCV`.

Pour utiliser cette méthode, l'image nécessite un prétraitement de type `grayscale`, `blur` et un `threshold`.

```
gray = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)
gray = cv2.equalizeHist(gray)
blur = cv2.GaussianBlur(gray, (15, 15), 0)
# Application d'un seuil pour obtenir une image binaire
thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 1)
kernel = np.ones((3, 3), np.uint8)
# Application d'érosion et d'ouverture pour supprimer les contours de petites pièces
closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations=1)
```

La fonction `equalizeHist()` permet de normaliser l'histogramme de gris, ce qui permet de réduire les variations d'intensité de l'image causé par la luminosité.

```
contours, hierarchy = cv2.findContours(closing.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

Grâce à la fonction `findContours()`, on récupère les contours détectés sous forme de tableau de coordonnées point (x, y). L'aire des contours est calculée à l'aide de la fonction `contourArea()`. Une pièce est détectée si l'aire est supérieure à une certaine valeur, mais celle-ci dépend de la taille de l'image ainsi que de la distance de l'appareil photo avec le sujet (test réalisé avec une aire ≥ 10.000 et ≤ 50.000). Pour dessiner les contours, on utilise la fonction `fitEllipse()` qui nécessite un tableau de coordonnées point avec au moins 5 points. Avec ces éléments on peut ensuite déterminer les pièces détectées et les découper pour obtenir les images de pièces séparées.



Détection réussie

3.4 Les complications

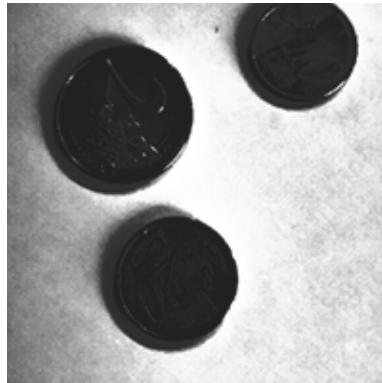
3.4.1 L'impact de la lumière sur le traitement

La lumière est une source de problèmes pour la détection des pièces, car elle peut être trop forte ou trop faible, ce qui peut entraîner des détections de pièces fausses. Une des solutions proposées est de réduire la lumière de l'image en appliquant une [égalisation de l'histogramme](#) sur le niveau gris puis un filtre de [blur](#).

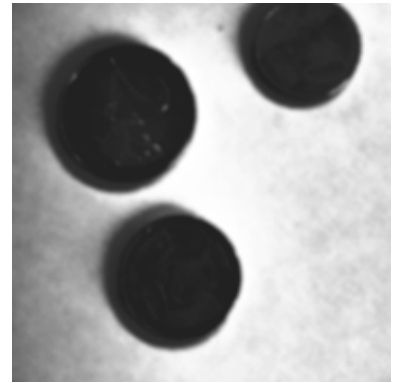
Original



Égalisation

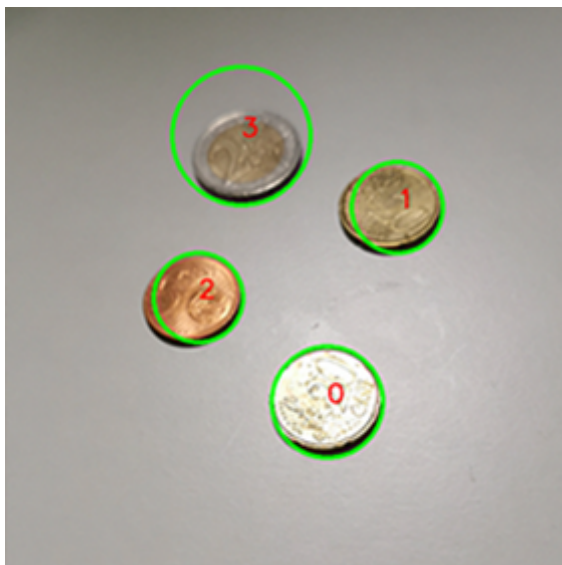


Blur



3.4.2 L'impact de l'inclinaison sur le traitement

L'inclinaison est une source de problèmes pour la détection des pièces, car elle peut être trop forte ou trop faible, ce qui peut entraîner des détections de pièces fausses ou de pièces non détectées. Une des solutions proposées est de réduire l'inclinaison de l'image en appliquant une [rotation](#) de l'image, malheureusement cette solution n'est pas optimale car elle nécessite une rotation de l'image et un rééchantillonnage de l'image, qui peut être un peu long et qui peut être un peu coûteux en ressources.



Exemple
d'inclinaison

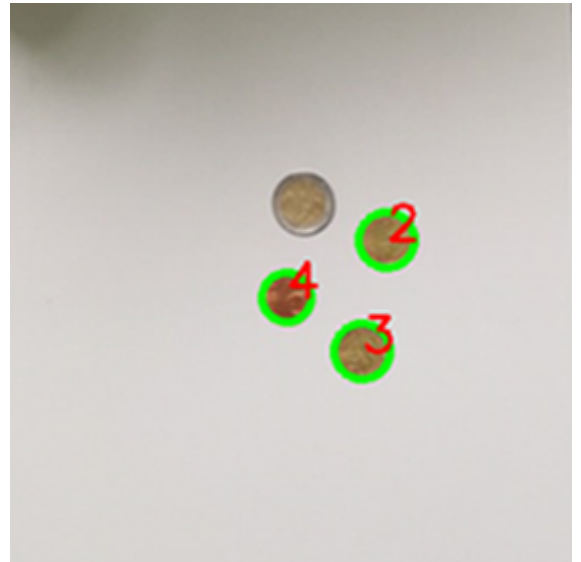


3.4.3 L'impact de la distance sur le traitement

La distance est une source de problèmes pour la détection des pièces, car elle peut être trop forte ou trop faible, ce qui peut entraîner des détections de pièces fausses ou de pièces non détectées. Une des solutions proposées est de réduire la distance de l'image en appliquant une [mise à l'échelle](#) de l'image.



Exemple de distance



3.4.4 Conclusion

La détection des pièces est une étape importante dans le traitement d'une image par l'application. Si les pièces sont détectées correctement, il est possible de les traiter et de les classer avec un meilleur taux de réussite. Dans le cas échéant, il est possible de les traiter et de les classer avec un moins bon taux de réussite.

4. Amélioration de la détection

4.1 Détection de la couleur

4.1.1 Méthode de détection

La détermination de la couleur est réalisée à l'aide du module K-Means de [Sklearn](#), avec une méthode de clustering K-Means. Toutefois, il est important de noter que la méthode de clustering K-Means est un algorithme non-supervisé, ce qui signifie qu'il n'est pas possible de prédire la couleur d'une pièce.

Il est donc important de prédire la couleur de la pièce en utilisant une base de données de couleurs prédéfinies. Mais cette méthode n'est pas très efficace, car elle est soumise à une forte influence de la lumière venant de l'environnement. Le matériau utilisé pour la pièce est également source de problèmes, en majeure partie dû à son coefficient de réflexion.

Il faut d'abord préparer l'image, en échangeant les couleurs BGR par RGB. Dans le cas illustré, on a une image de couleur BGR, mais il faut la convertir en RGB, et récupérer le centre de l'image pour éviter les problèmes de couleur venant du support.

```
# Read image
original = cv2.imread('coin.png')

# Convert to RGB
image = cv2.cvtColor(original, cv2.COLOR_BGR2RGB)
# Resize
image = cv2.resize(image, (600, 600), interpolation = cv2.INTER_AREA)
# Crop center to avoid border effects
image = image[275:325, 275:325]
# Flat to 1D array
image = image.reshape(image.shape[0]*image.shape[1], 3)
```

On lance ensuite l'algorithme K-Means sur l'image, avec un nombre de clusters égal à 1, pour ne récupérer qu'un seul cluster et donc qu'une seule couleur.

```
# Compute K-Means
clf = KMeans(n_clusters = n_clusters)
colors = clf.fit_predict(image)
```

Pièce



Extraction



Résultat

#b79f7d



4.2 Détection de la taille

4.2.1 Méthode de détection

Grâce à l'algorithme de détection de cercles, on peut récupérer la taille de la pièce, plus précisément son rayon. Avec ces données on peut ensuite comparer la taille de la pièce avec la taille de la pièce en laquelle on a la plus grande confiance.

Détection par contour non compatible

L'algorithme de détection de contours, ne permet pas de prédire la taille de la pièce, mais permet de récupérer plus ou moins les contours de la pièce.

Le majeur problème de cette méthode est qu'elle repose sur la fiabilité de la classification, si la classification est fausse, le résultat impactera les autres détections.

Voici l'idée derrière l'algorithme:

```
# On cherche la pièce avec la meilleur confiance
coins = [] # Retour du classifieur
best_coin = rechercher_best_coin(coins)

# On ajoute la taille de cette pièce à sa catégorie
categories[best_coin.label].size = best_coin.size

# On stock les données actuelles
best_category = category[best_coin.label]
category = best_category

# Méthode pour calculer l'écart de taille
def diff(coinA, coinB):
    return abs(coinA.size - coinB.size)

# On cherche ensuite les tailles des autres pièces inférieures avec une marge de 10%
threshold = best_coin.size * 0.1
lowest = 0.0
lowest_coin = None
# On boucle sur les class qui précèdent la meilleure
for i in range(categories.index(category) - 1, -1, -1):
    for coin in coins:
        # On vérifie que la pièce est bien plus petite
        if coin.size < category.size - threshold:
            continue

        # On cherche la plus petite différence
        if coin.size < lowest:
            lowest = coin.size
            lowest_coin = coin

    # On ajoute la taille de la pièce à sa catégorie
    category[categories.index(category)-1].size = lowest_coin.size
    coins.remove(lowest_coin)
    category = category[lowest_coin.label]

# Le procédé est le même pour les pièces supérieures,
# la boucle change de valeur pour 'range(categories.index(category) + 1, len(categories))'

# On boucle ensuite sur la liste des coins et on ajoute les tailles les plus proches à chacune
```

Par manque de temps, l'optimisation de la méthode ainsi que son implémentation n'ont pas été réalisées.

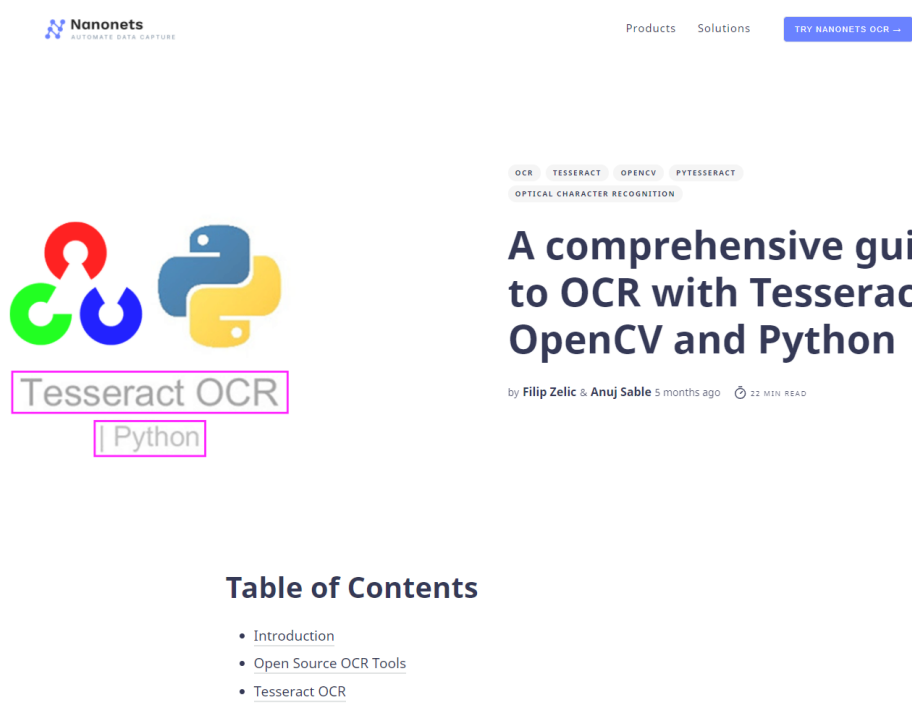
4.3 Détection des chiffres sur les faces avant

4.3.1 Recherches

La détection des chiffres sur les faces avant est une méthode de détection plus précise que toutes les autres énumérées. Cependant son implementation est très complexe, pour pouvoir lire la valeur numéraire de la pièce, il faut détecter les chiffres sur les faces avant, mais l'image celle-ci doit être sur une rotation permettant de lire la valeur numérique par le biais de la [Reconnaissance Optique de Caractères](#).

Aucun point commun entre les différentes pièces n'a été trouvé, ce qui rend impossible de faire une méthode de rotation pour lire les valeurs numériques.

Beaucoup de recherches ont eu lieu autour du sujet, notamment l'article "[OCR avec Tesseract](#)" sur le site de Nanonets.com qui est très intéressant.



Schéma

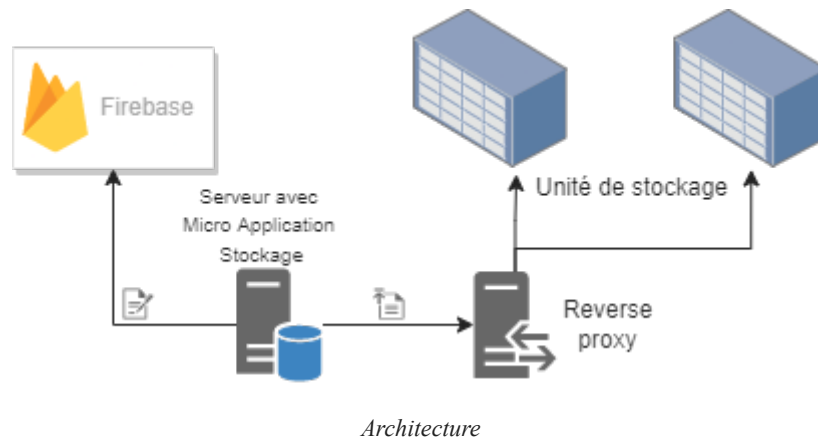
Malheureusement, cette fonctionnalité ne sera pas disponible, par manque de temps et de preuve solide, un prototype n'a pas pu être réalisé.

5. Amélioration du service

5.1 Micro Service de stockage

5.1.1 Architecture du micro service

Voici un rappel de l'architecture imaginée pour le micro service de stockage :



Unités de stockage

Les unités de stockage ont pour but de stocker un maximum de données, afin de ne pas surcharger le disque dur du serveur de traitement. Ils permettent de stocker l'intégralité des rapports d'analyse fournis par les serveurs de traitement.

Un rapport est constitué de plusieurs éléments.

- ✓ Un fichier JSON contenant les informations sur le rapport
- ✓ Un dossier "zones" contenant les différentes pièces détectées
- ✓ Deux fichiers images concernant l'image original ainsi que l'image contenant les détections

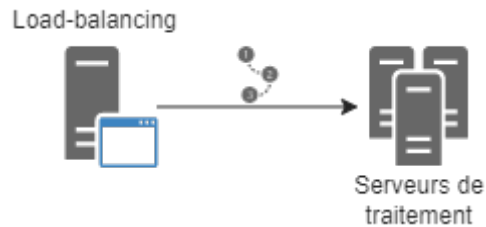
Toutes ces données sont essentielles pour l'apprentissage du modèle de manière autonome.

Le tout serait protégé par un reverse proxy, afin de ne pas exposer le contenu directement sur l'internet.

5.2 Micro Service de détection

5.2.1 Architecture du micro service

Voici un rappel de l'architecture imaginée pour le micro service de détection :

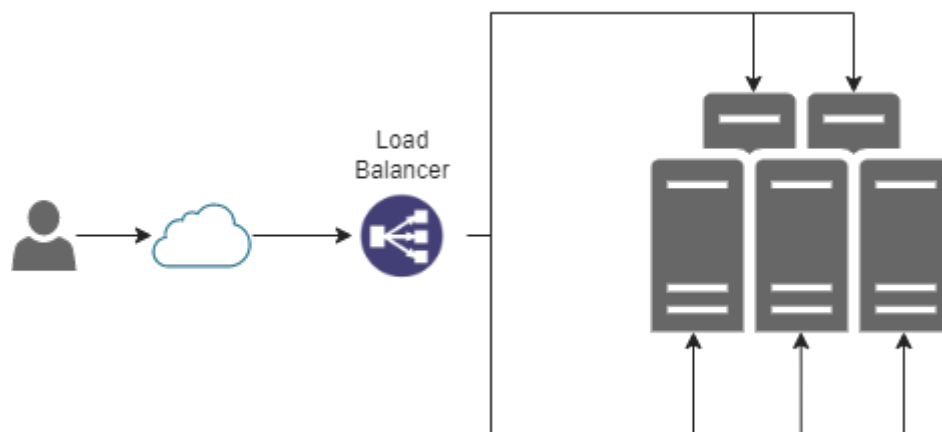


Architecture

Unités de traitement

Le but de cette unité est de construire un parc de serveurs destinés à traiter les images, car certains algorithmes de détection sont très lourds. Le but est d'utiliser des serveurs de traitement multi-core pour améliorer les performances, avec un Load Balancer pour gérer les requêtes en première ligne afin de répartir équitablement les travaux.

Un serveur avec une haute configuration serait également réservé pour les entraînements autonome du modèle.



Load Balancing

6. Amélioration du client

6.1 Restreindre la distance maximale de la photo

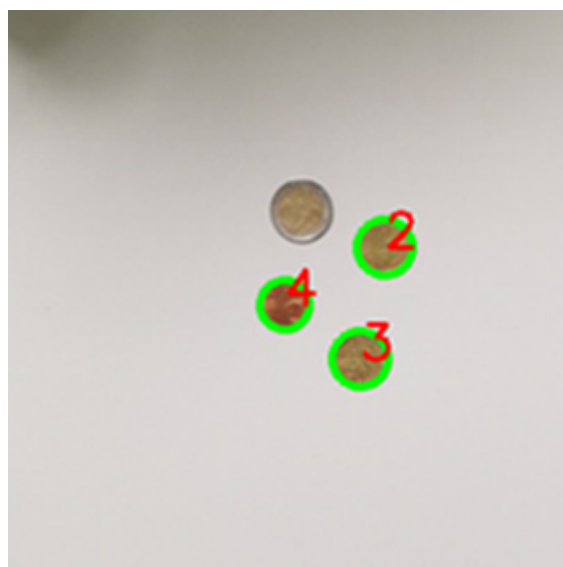
6.1.1 L'impact de la distance sur la détection

La distance entre le sujet et l'objectif est importante pour la détection.

La méthode `houghCircle` est plutôt précise, mais elle est soumise à une forte influence de la distance entre le sujet et l'objectif, il en est de même pour la détection des contours.



Exemple de distance



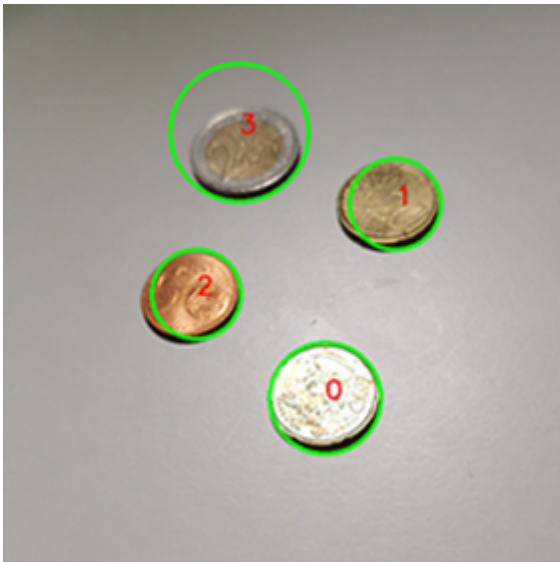
Solution envisagée

Restreindre la distance maximale de la photo à une distance de 20 cm, en utilisant la technologie disponible dans les appareils récents.

6.1.2 L'impact de l'angle sur la détection

L'angle de prise de vue est important pour la détection, si il est trop grand, la détection est moins précise.

La méthode `houghCircle` est plutôt précise, mais elle est soumise à une forte influence de l'angle de prise de vue, La détection des contours sera plus efficace dans cette situation.

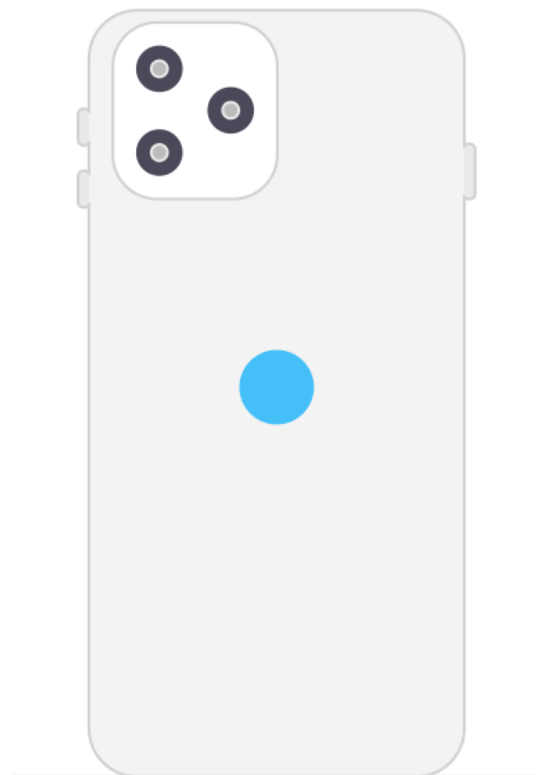


*Exemple
d'inclinaison*



Solution envisagée

Restreindre l'angle maximale de la photo à 15° , en utilisant le gyroscope de l'appareil. Cela empêche certains type de photo (murale), mais il est possible de rajouter des exceptions.

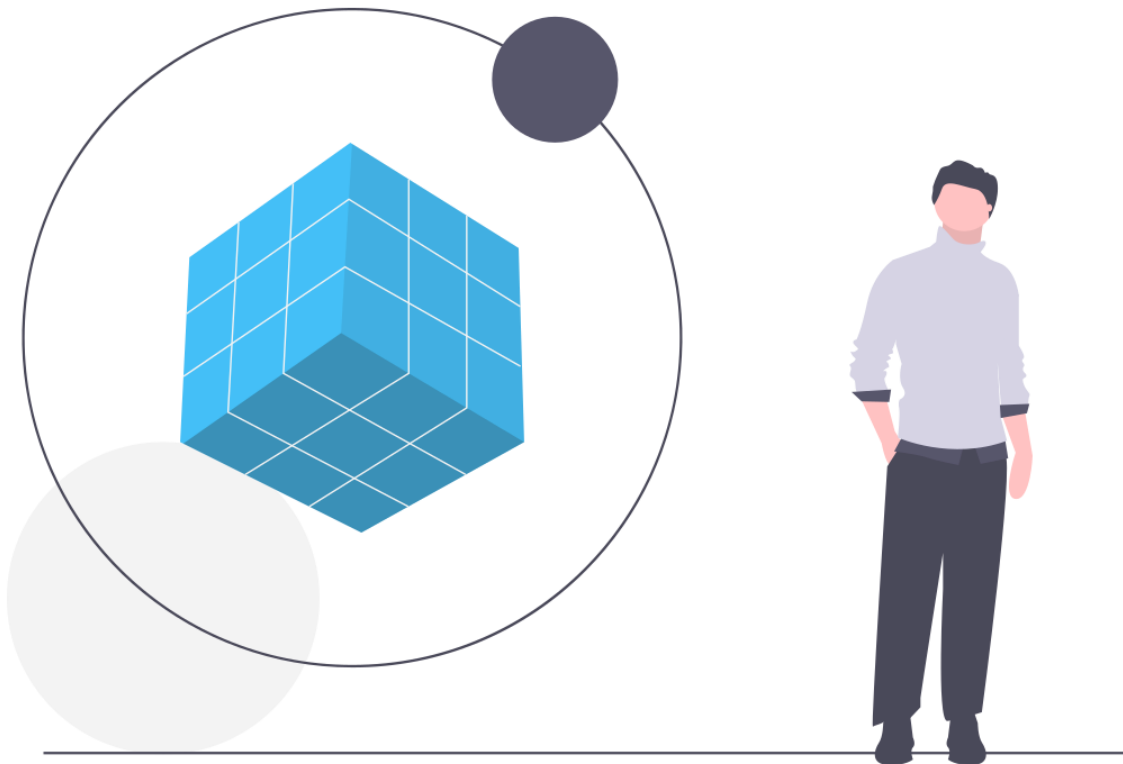


6.2 Importer des photos depuis la Galerie

6.2.1 Problématique

L'importation d'images depuis la galerie est un cas complexe, car il faut prendre en compte les différents cas de figure. L'image ne respectera pas forcément les contraintes de la méthode de détection de cercles ou de contours, et peut être difficile à analyser.

Pour y remédier, il faut utiliser une méthode de détection plus précise et plus efficace, capable de détecter les pièces sous n'importe quel angle sans restriction de la distance maximale. Malheureusement, une telle prouesse n'a pas pu être réalisée, par manque de temps et de preuve solide, un prototype n'a pas pu être réalisé.



7. Amélioration du serveur

7.1 Amélioration de la disponibilité

7.1.1 Améliorer la montée en charge

Les traitements réalisés sur les images sont plutôt lourds, mais ils ne sont pas suffisamment précis ce qui oblige parfois des traitements encore plus lourds. Les durées de traitement des images sont donc très importantes, et paralyse l'application.

Comment y remédier ?

LOAD BALANCING

La première solution consiste à mettre en place un système de Load Balancing, qui permet de répartir les travaux sur les différents serveurs, dans le but d'éviter les défauts de performance mais également la paralysie de l'application.

SYSTÈME DE FILE D'ATTENTE

La seconde solution consiste à ajouter un système de queue, qui permettrait d'accepter les requêtes même en cas de défaut de performances.

Bien que cette solution soit très complexe à mettre en place, elle est très efficace, mais n'est pas adaptée à toutes les applications.

Il faudra donc envisager d'utiliser une solution plus complète et plus adaptée à l'application.

LIMITER LES REQUÊTES

La dernière solution consiste à limiter les requêtes, qui permet de limiter le nombre de requêtes en cours, et de limiter le nombre de requêtes en attente.

La limitation aurait lieu sur les requêtes d'un même utilisateur, ce qui permet de donner l'accès à plusieurs utilisateurs en même temps.

7.2 Amélioration de la sécurité

7.2.1 Protection contre les attaques

Apprendre à protéger l'application contre les attaques est une tâche très importante. Notamment dans le cadre d'une application web, qui est donc susceptible de recevoir des requêtes malveillantes, il est important de pouvoir gérer les attaques.

Dans notre application, les requêtes pour des traitements sont parfois lourdes, ce qui entraîne des défauts de performance. Une personne mal intentionnée peut par exemple demander à l'application de traiter une image de 100Mo, ou encore plusieurs images en même temps.

Il y a également une attaque de type "DoS" qui consiste à faire des requêtes inutiles en continuant de les faire.

Solution

La principale solution pour gérer les attaques est de limiter le nombre de requêtes en cours, notamment avec un load balancer correctement configuré. Et de limiter la taille des images dans les demandes de traitement.

7.2.2 Utilisation de certificats SSL

L'utilisation de certificats SSL est une bonne pratique pour garantir la sécurité de l'application. Elle permet de s'assurer que l'application est bien connectée à un serveur qui a un certificat SSL valide.



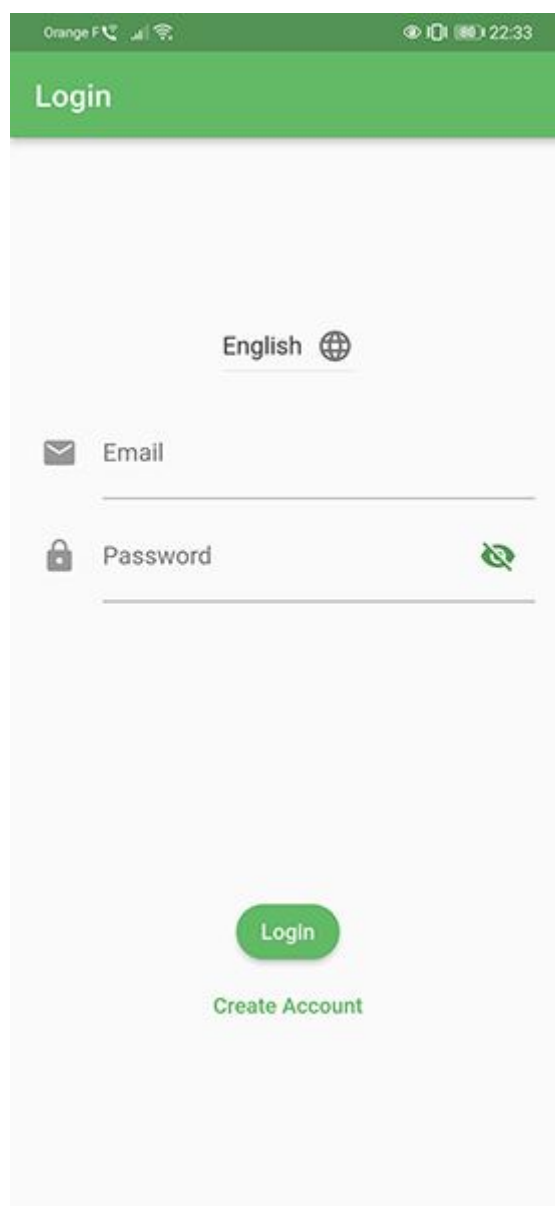
8. Documentation Utilisateur

8.1 Premier pas

L'application est en Anglais mais dispose d'une traduction en Français.

8.1.1 Déjà inscrit ?

Pour vous connecter, il suffit de vous rendre sur la page de connexion. Si vous n'êtes pas encore inscrit, vous pouvez vous inscrire en cliquant sur le bouton "Create Account".

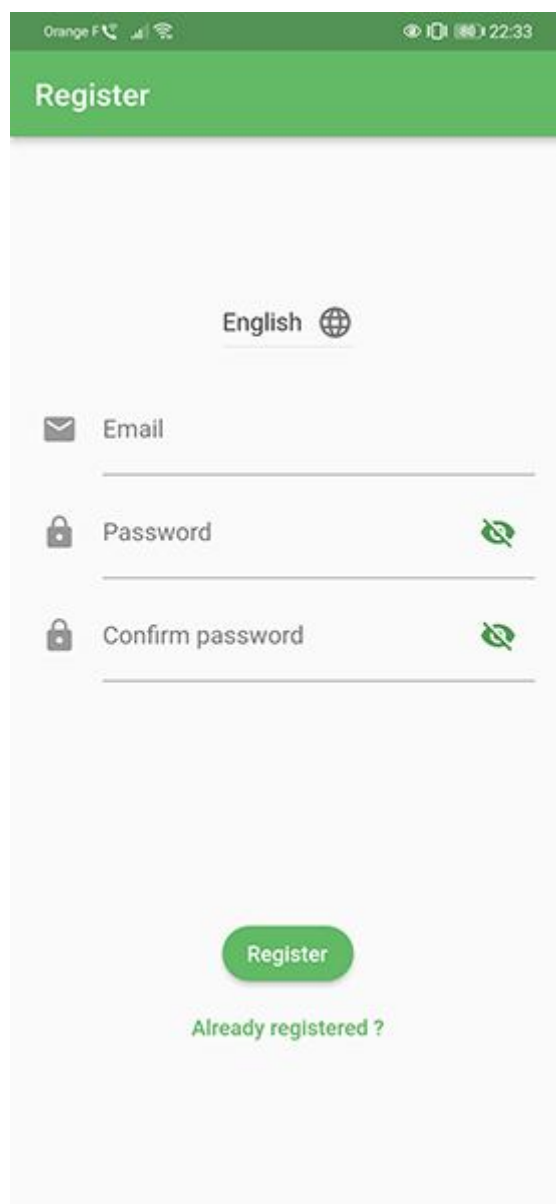


Page d'authentification

8.1.2 Inscription

Cette page est accessible depuis la page de connexion. Il suffit de cliquer sur "Create Account" et de remplir les champs suivants :

Une fois l'inscription terminée, vous serez automatiquement connecté et redirigé vers la page d'aide.

The image shows a mobile application interface for registration. At the top, there is a green header with the word "Register" in white. Below the header, the status bar shows "Orange F" and the time "22:33". The main content area is white and contains a language selector "English" with a globe icon. Below this are three input fields: "Email" with an envelope icon, "Password" with a lock icon and a green eye icon to toggle visibility, and "Confirm password" with a lock icon and a green eye icon. At the bottom, there is a green "Register" button and a link "Already registered ?" in green text.

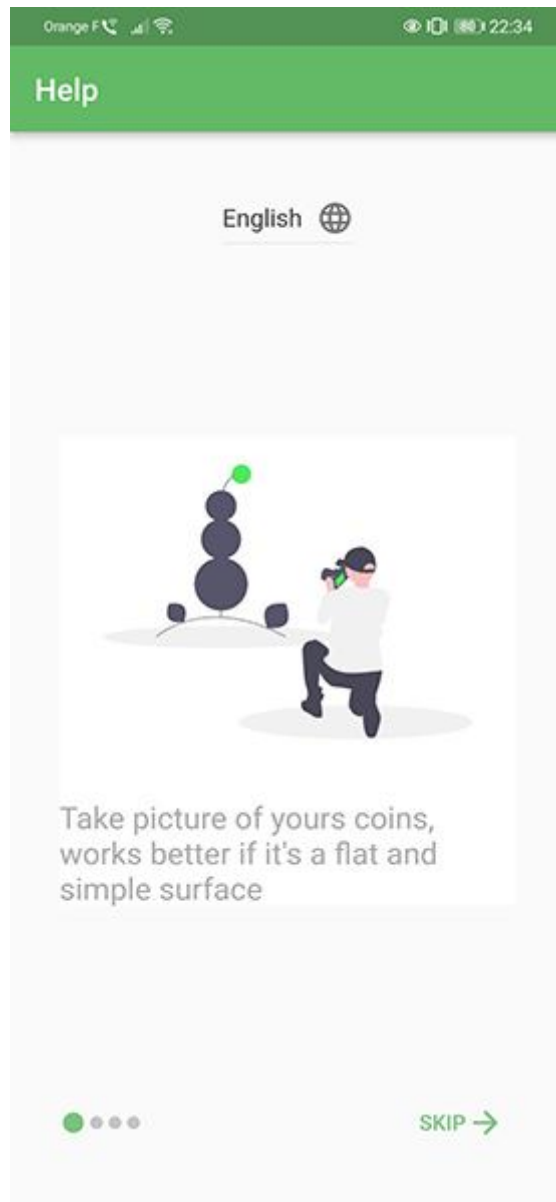
Page d'inscription

Important

Vous serez redirigé vers la page d'aide après l'inscription. Vous pourrez la passer en appuyant sur le bouton dédié.

8.1.3 Aide

La page d'aide s'affiche après avoir rempli le formulaire d'inscription. Il est également possible de revenir sur la page d'aide en cliquant sur le bouton "Help" dans le menu latéral.

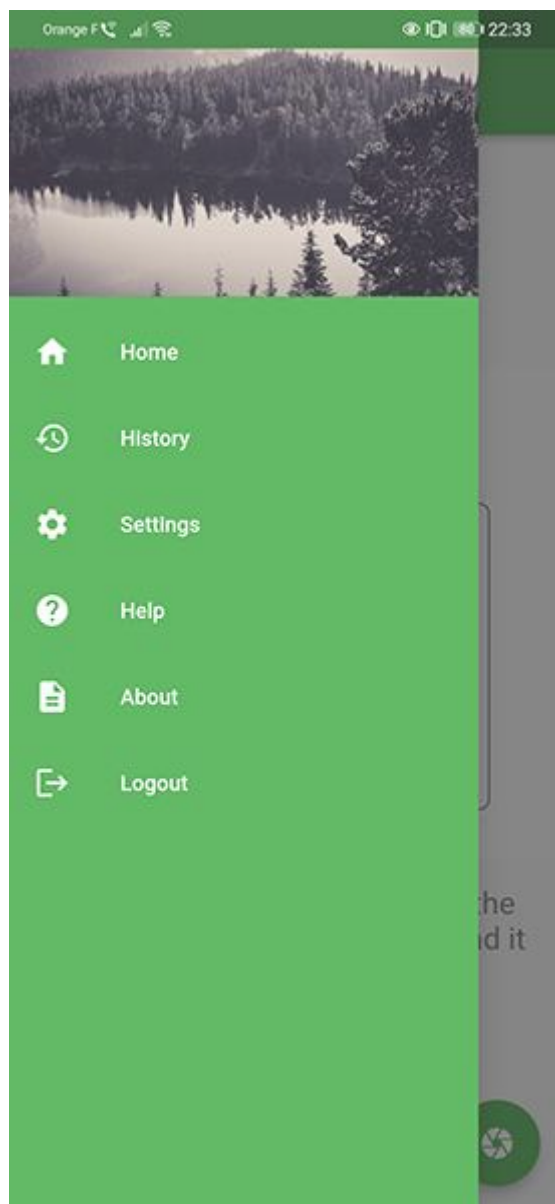


Page d'aide

8.2 Les différents menus

8.2.1 Afficher le menu

Le menu peut être affiché en utilisant le bouton `menu`



Menu de l'application

Home | Accueil

Il s'agit de la page principal de l'application. Vous pouvez y accéder en cliquant sur le bouton "Home | Accueil" dans le menu latéral.

Vous y retrouverez le bouton pour prendre une photo et l'envoyer au serveur.

History | Historique

Il s'agit de la page de l'historique des traitements. Vous pouvez y accéder en cliquant sur le bouton "History | Historique" dans le menu latéral.

Vous y retrouverez les traitements effectués par l'application.

Settings | Paramètres

Il s'agit de la page des paramètres. Vous pouvez y accéder en cliquant sur le bouton "Settings | Paramètres" dans le menu latéral.

Vous y retrouverez les paramètres modifiable de l'application.

Attention

Pensez à paramétrer votre application avant de l'utiliser. Vous devez vous assurer que vos paramètres d'API sont configurées correctement.

Help | Aide

Il s'agit de la page d'aide. Vous pouvez y accéder en cliquant sur le bouton "Help | Aide" dans le menu latéral.

Vous y retrouverez les informations basique sur le fonctionnement de l'application (tutoriel).

About | A propos

Il s'agit de la page des détails de l'application. Vous pouvez y accéder en cliquant sur le bouton "About | A propos" dans le menu latéral.

Cette page n'est pas disponible.

Logout | Déconnexion

Il s'agit de la page de déconnexion. Vous pouvez y accéder en cliquant sur le bouton "Logout | Déconnexion" dans le menu latéral.

Permet de vous déconnecter de l'application.

8.3 Les conseils d'utilisation

8.3.1 Prendre une bonne photo

Pour prendre une bonne photo, Assurez-vous de suivre les consignes suivantes :

- ✓ La caméra est bien autorisée.
 - ✓ La lentille de la caméra est bien nettoyée.
 - ✓ Les pièces sont légèrement espacées et bien positionnées.
 - ✓ La lumière n'est pas trop forte ni trop faible.
 - ✓ La distance entre les pièces et l'objectif est suffisante ($> 10\text{ cm}$ & $< 30\text{ cm}$).
 - ✓ L'angle de la caméra est correct ($< 30^\circ$).
-

8.3.2 Corriger les résultats

Dans l'historique, vous pouvez voir les résultats des traitements. Mais vous pouvez aussi soumettre une correction sur une pièce que vous jugez incorrecte.

L'objectif est de corriger les résultats des traitements pour permettre à notre service de s'améliorer.



Important

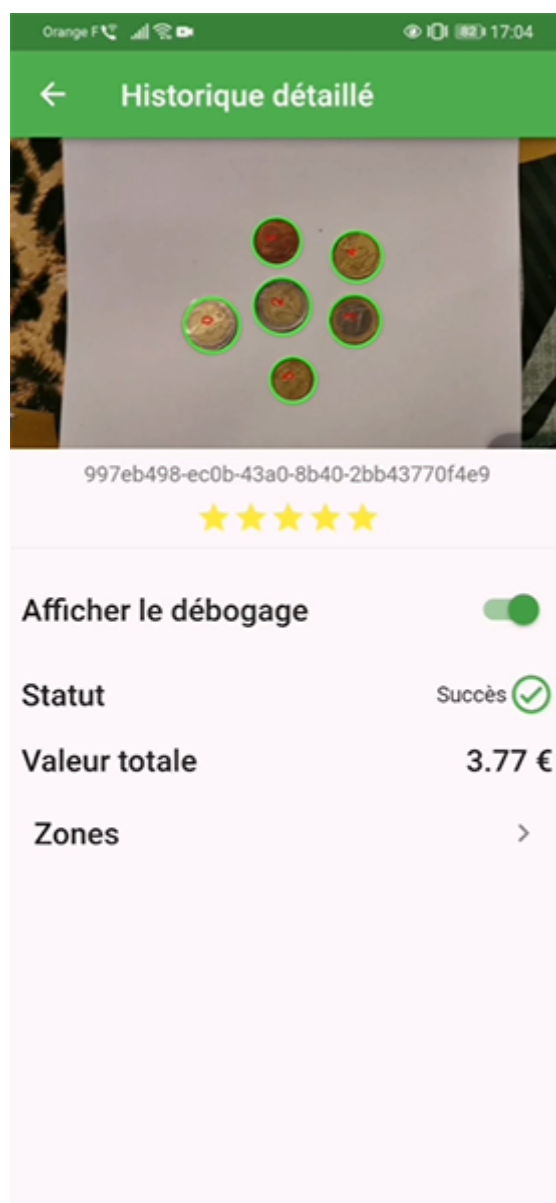
Les corrections sont traitées par notre service. Vous pouvez donc soumettre une correction sur une pièce que vous jugez incorrecte.



Attention

Les abus de corrections peuvent entraîner des sanctions.

La gestion des corrections est la suivante :



Rapport détaillé

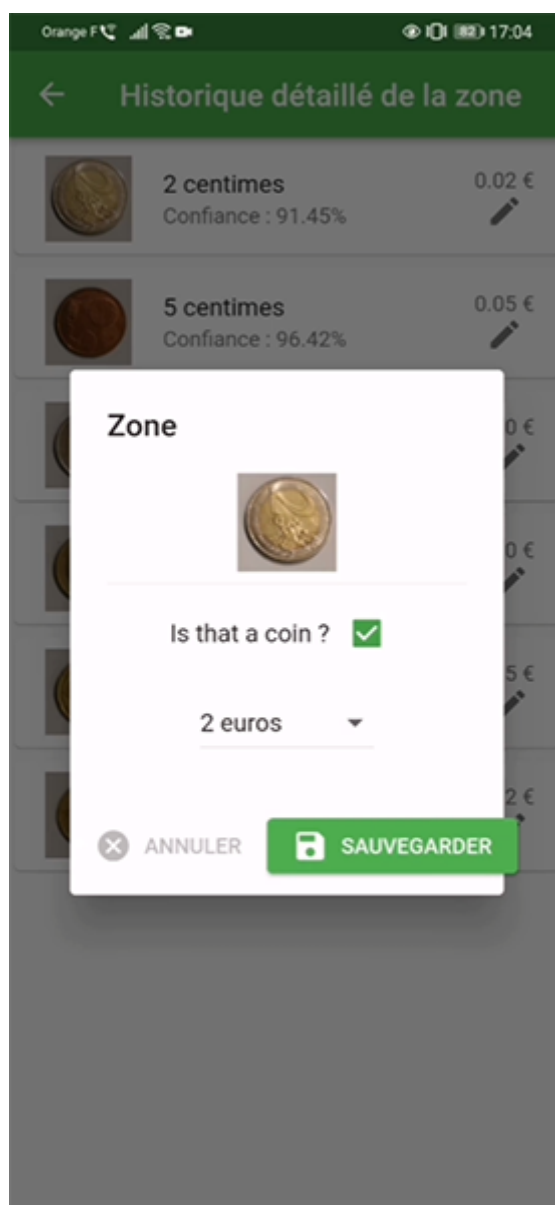
En cliquant sur "Zones" vous pouvez voir les zones détectées par le système et soumettre une correction.



← Historique détaillé de la zone			
	2 centimes Confiance : 91.45%	0.02 €	
	5 centimes Confiance : 96.42%	0.05 €	
	2 euros Confiance : 100.00%	2.0 €	
	1 euro Confiance : 100.00%	1.0 €	
	50 centimes Confiance : 97.39%	0.5 €	
	20 centimes Confiance : 69.23%	0.2 €	

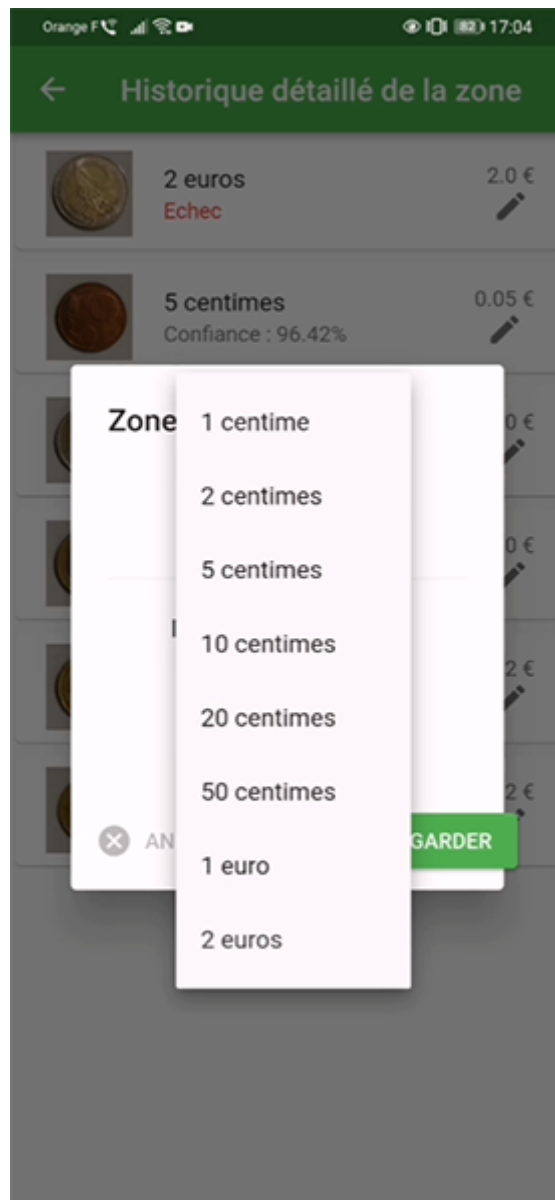
Rapport détaillé des zones

En cliquant sur une zone vous accédez à la correction de cette zone.



Correction d'une zone

Vous pouvez préciser si il s'agit bien d'une pièce, ou corriger la pièce si elle est incorrecte.



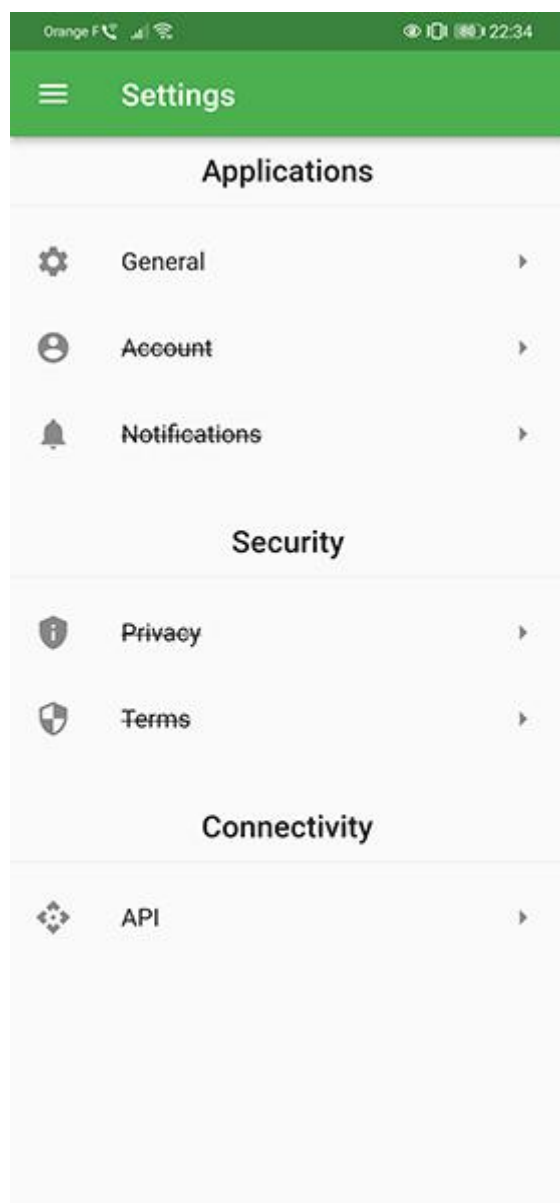
Correction d'une zone détaillé

8.4 Exemple d'utilisation

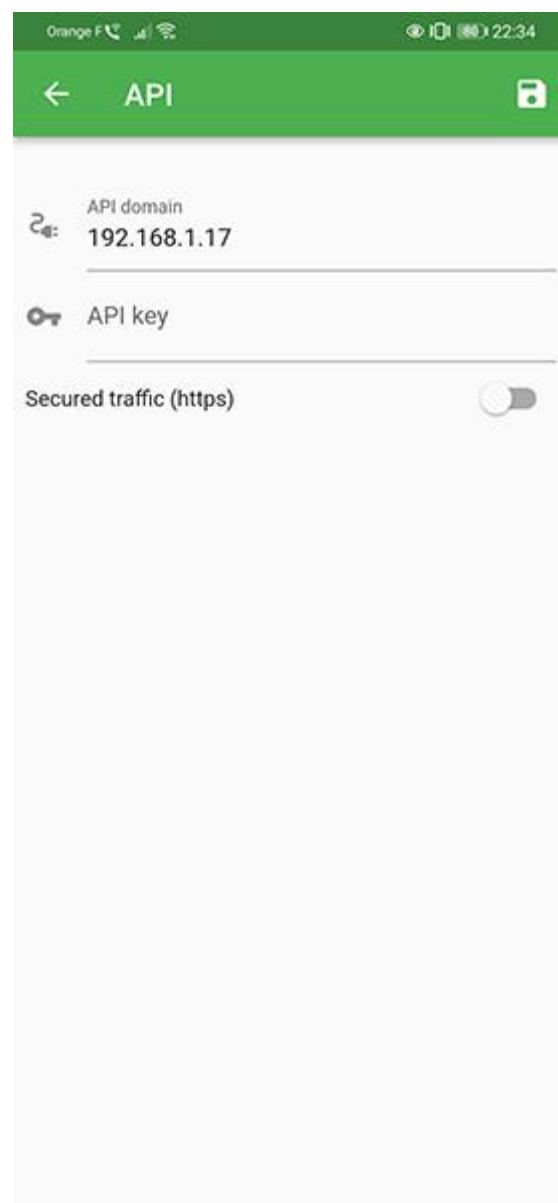
8.4.1 Manipulation de l'application

Avant de commencer, il est nécessaire de configurer l'application.

Pour cela, rendez vous dans la page des paramètres, accessible depuis le menu latéral en cliquant sur le bouton "Settings". Puis dans la section "API" :



Page de paramètres



Important

La clé API n'est pas nécessaire pour pouvoir utiliser l'application. Ce mécanisme n'est pas encore implémenté.

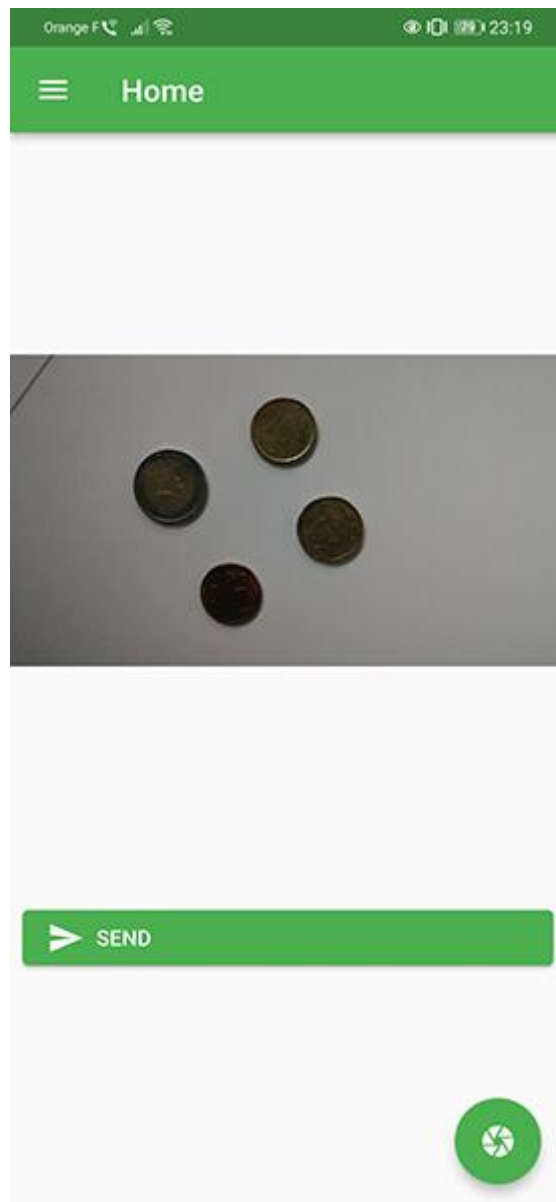
Première étape : Prendre une photo

Pour prendre une photo, rendez vous dans la page "Home" accessible depuis le menu latéral en cliquant sur le bouton "Take a photo"



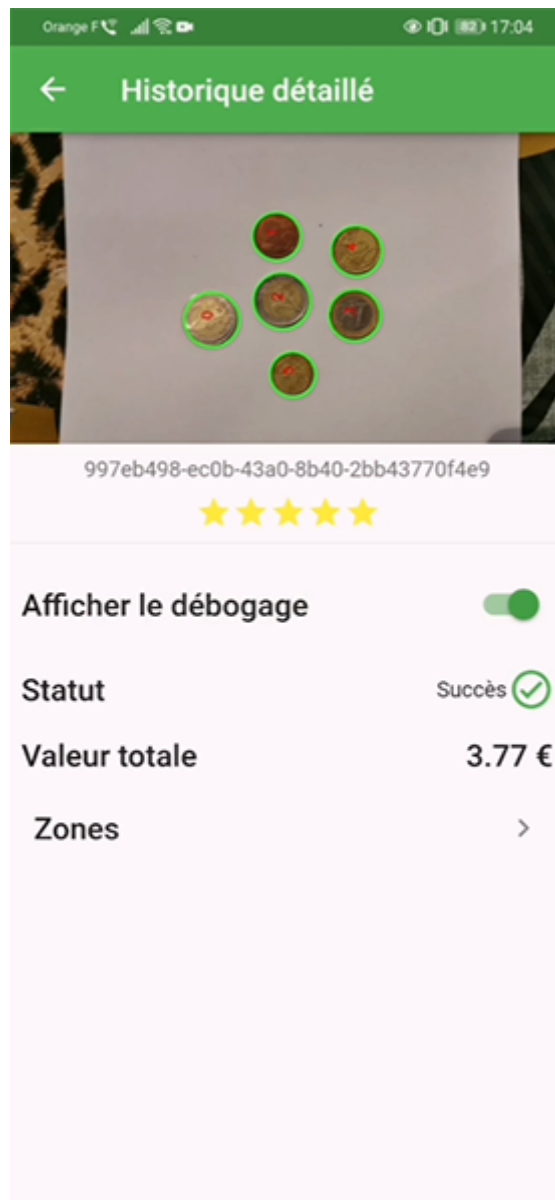
Page d'accueil

Une fois la photo prise, vous accédez à la page d'envoi au serveur.



Page d'envoi

En cliquant sur le bouton "Send", il faudra attendre que la photo soit envoyée au serveur et traitée. Une fois la réponse du serveur reçue, vous accédez à la page de résultat.



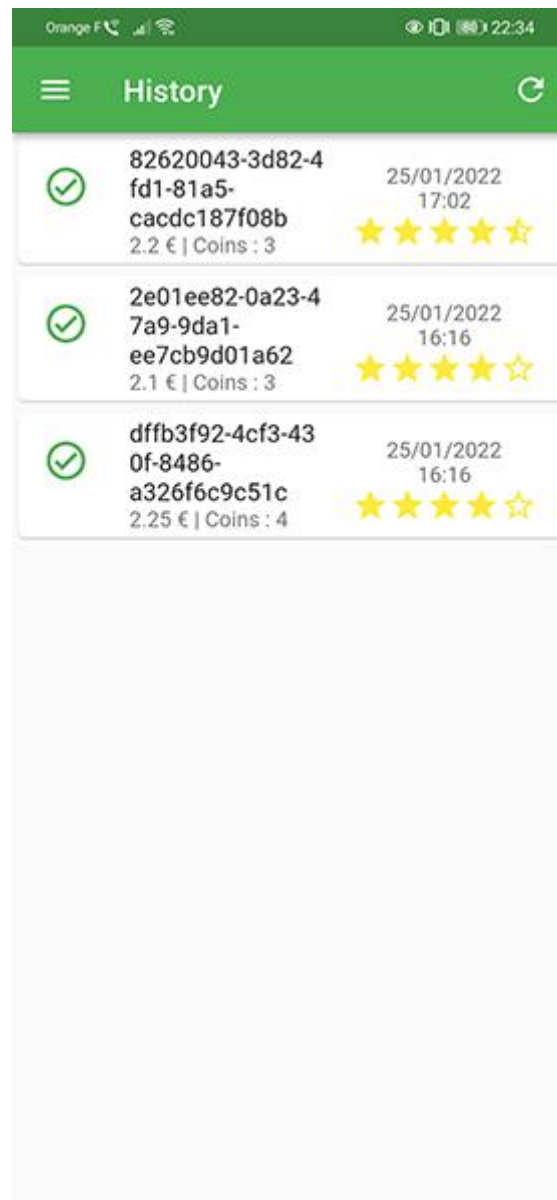
Page de résultat

Correction de résultat

Pour corriger le résultat, veuillez vous référer à la section "Conseils d'utilisation"

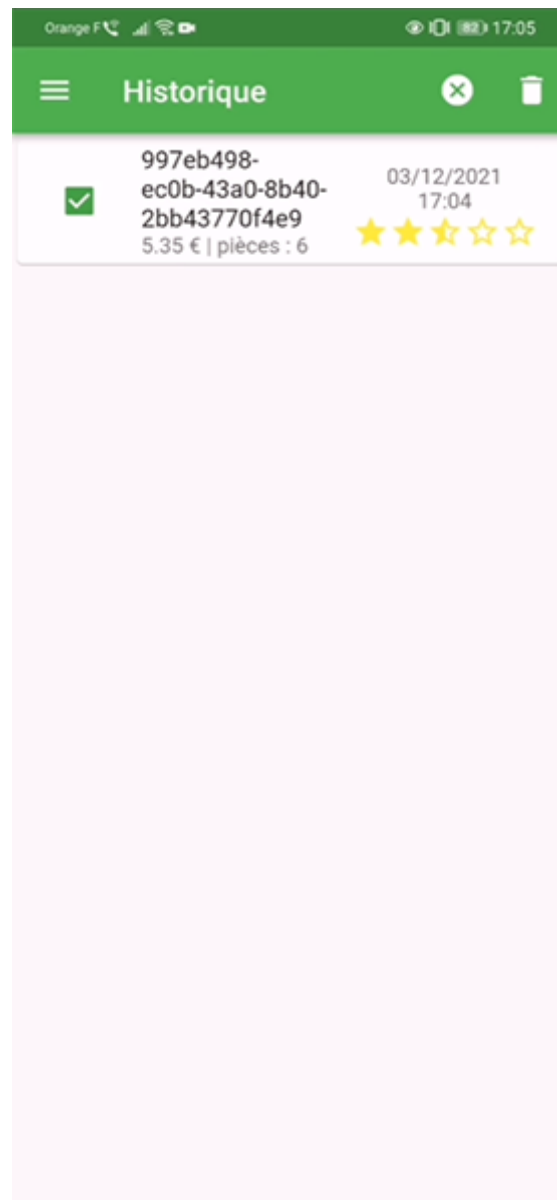
Deuxième étape : Gérer l'historique

Vous pouvez accéder à l'historique des traitements en cliquant sur le bouton "History" dans le menu latéral.



Page de d'historique

Vous y retrouverez les traitements effectués par l'application. Vous pouvez supprimer un traitement en restant appuyé sur un traitement, en faisant cela vous accédez au mode de suppression. Plusieurs traitements peuvent être sélectionnés en même temps.



Page de d'historique - Mode suppression

Pour supprimer les traitements sélectionnés, cliquez sur le bouton "Delete"



. Si vous souhaitez annuler la suppression, cliquez sur le

bouton "Cancel"



9. Glossaire

9.1 Définitions

9.1.1 API

Une API, ou interface de programmation d'application, est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

9.1.2 Firebase

Firebase est un service de gestion de données et de services web. Projet open source, il est développé par Google.

9.1.3 Flutter

Flutter est un SDK pour la création de applications Android et iOS. Projet open source, il est développé par Google.

9.1.4 Flask

Flask est un microframework Python. Projet open source, il est développé par Armin Ronacher.

9.1.5 Reverse Proxy

Un reverse proxy est un serveur HTTP qui permet de faire un traitement sur un autre serveur HTTP non exposé sur l'internet mondiale. Sans reverse proxy, il serait impossible de faire des requêtes HTTP sur ce dernier.

9.1.6 Load Balancer

Un load balancer est un serveur qui permet de répartir des requêtes HTTP sur plusieurs serveurs HTTP.

9.1.7 OpenCV

OpenCV est une librairie de traitement d'image. OpenCV est un projet open source, il est développé par Intel.

9.1.8 TensorFlow

TensorFlow est une librairie open source de machine learning. Il a été développé par Google et est utilisé pour la recherche et la production de logiciels de machine learning.

9.1.9 Keras

Keras est une API de réseau de neurones développée par Google. Elle est utilisée pour la construction de réseaux de neurones. Rédigée en Python, elle permet une implémentation rapide et facile d'algorithmes de machine learning.

9.1.10 K-Means

K-Means est un algorithme de clustering. Il permet de découper des données en groupes de données.

9.1.11 Epochs

Les Epochs sont le nombre de fois que les données d'entrainement sont passées dans le réseau de neurones.

9.1.12 Batch Size

Le Batch Size est le nombre d'entrées d'entrainement qui sont utilisées pour le calcul de la gradient.

9.1.13 Learning Rate

Le Learning Rate est la vitesse d'apprentissage. Il détermine la vitesse à laquelle les poids sont mis à jour.

9.1.14 Le format H5

Le format H5 est un format de fichier utilisé pour stocker les modèles de réseau de neurones.

9.1.15 Le format .pickle

Le format .pickle est un format de fichier utilisé pour stocker les label associé à un modèle de réseau de neurones sous forme binaire.

9.1.16 OCR

La reconnaissance des caractères est une technique de reconnaissance utilisé pour reconnaître des caractères dans des images.

9.1.17 Dataset

Un Dataset est un ensemble de données d'entrainement et de test.

9.1.18 Blur

Le Blur est un effet de flou. Au sein de l'image, les pixels sont déplacés dans un rayon de 3 pixels autour de lui.

9.1.19 Grayscale

Le Grayscale est un effet de conversion de couleur en niveaux de gris.