Rapport Projet TOO-CAI

« Better Food » (2022-2023)

L3 INFORMATIQUE

16 décembre 2022

Créé par : Florian ROBIN et Vincent PALACIO

Rapport Projet TOO-CAI

« Better Food » (2022-2023)

I-Présentation du projet

Le but du projet est de prendre une photo d'un code barre issu d'un produit alimentaire présent dans le commerce à partir de la caméra d'un ordinateur afin de récupérer les informations associées à ce produit. Pour cela nous utilisons le site Web world.openfoodfacts.org qui possède une API au format JSON. Ce site répertorie toutes les informations liées aux produits alimentaires qui sont enregistrées dans le commerce comme le nutri-score d'un produit, les ingrédients, l'origine du produit, etc…

Pour réaliser ce projet nous disposons du plusieurs librairies pour déchiffrer le code barre comme quaggaJS, quagga2, javascript-barcode-reader, ces librairies sont en javascript, elles sont utilisables à partir d'une photo prise par la caméra de l'ordinateur par exemple. La photo est prise grâce à la technologie WebRTC, côté serveur java est utilisé pour communiquer les informations entre le client et le serveur afin de récupérer les informations voulues sur le produit scanné.

II- Les différents choix du projet

1) Partie client :

Pour la partie client nous avons fait le choix d'utiliser la technologie webRTC en javascript afin de prendre une photo avec la caméra de l'ordinateur. Nous avons donc pris l'exemple fourni « Selfie.ts.zip ». A partir de ce modèle nous avons utilisé la librairie Quagga2 qui est accessible sur gitHub, cette librairie est légèrement différente de QuaggaJS car elle est écrite en javascript et typescript. (https://github.com/ericblade/quagga2).

Ensuite nous avons utilisé l'exemple de code présent sur l'API QuaggaJS afin de pouvoir détecter un code barre et le stocker. Exemple ci-dessous :

Examples

The following example takes an image `src` as input and prints the result on the console. The decoder is configured to detect *Code128* barcodes and enables the locating-mechanism for more robust results.

```
Quagga.decodeSingle({
    decoder: {
        readers: ["code_128_reader"] // List of active readers
    },
    locate: true, // try to locate the barcode in the image
    src: '/test/fixtures/code_128/image-001.jpg' // or '' + dat
}, function(result){
    if(result.codeResult) {
        console.log("result", result.codeResult.code);
    } else {
        console.log("not detected");
    }
});
```

Nous avons donc modifié le code barre à lire qui est de type EAN car nous voulons le type de code barre utilisé dans le commerce. Nous avons également implanté la technologie Websocket afin de permettre de communiquer le code barre sur le serveur java. Pour cela nous nous sommes basés sur l'exemple fourni sur les webSockets ainsi que l'API. C'est donc le dossier « SelfieS.ts » qui contient le Html, le typescript ainsi que le javascript.

2) Partie serveur :

Cette partie concerne uniquement le java, deux fichiers sont présents : « OpenFoodFact.java » et « WebSocket.java ». Le premier fichier permet d'interroger la base de données via un lien Web de OpenFoodFact en récupérant le nutri-score au format JSON. Une fonction « Nutriscore » est alors créée, elle permet d'accéder au produit voulu grâce au code barre qui est envoyé depuis le client vers le serveur. Ensuite un « parser » est mis en place afin de parcourir toutes les données liées au produit pour trouver l'information que nous voulons afficher. Une fois que « nutriscore_grade » est trouvé, la fonction renvoie la lettre associée.

Dans le second fichier on utilise « tyrus-server » qui permet la création d'un serveur en java ainsi que la technologie des webScokets. Une classe « Server_Endpoint » est dédié pour la configuration de la connexion des webscokets, celle-ci configure la connexion, la communication entre les clients et le serveur ainsi que la fermeture de la connexion mais aussi la gestion des erreurs liées au serveur. C'est ici que le code barre est récupéré par la java, ensuite nous appelons la fonction OpenFoodFact pour trouver le nutri-score avec le code barre en paramètre puis la réponse est envoyé sur le serveur afin que le client puisse afficher le nutri-score sur la page Web.

De plus ce fichier contient aussi le « main » du programme qui permet de lancer le serveur et de le stopper quand on appuie sur une touche dans la console java.

III- Fonctionnement de l'application

Pour faire fonctionner l'application il suffit :

- Compiler et exécuter le programme java.
- La page web se lance, il faut cliquer sur l'image et autoriser l'utilisation de la caméra.
- Positionner le code barre de manière lisible au niveau de la caméra et prendre la photo en cliquant sur l'image.
- Si le nutri-score existe et que le code barre a bien été lu, il s'affiche sur la page Web.
- Fermer le serveur en appuyant sur « entrée » dans la console java.

- Fermer la page web.

Voici une capture d'écran montrant la partie côté client quand le code barre est détecté et que le nutri-score existe :

Selfie

le nutriscore du produit est :
"a"

le code barre est :
3250392603588

IV- Conclusion

Nous avons donc appris à utiliser les webSockets, un parser, la technologie WebRTC ainsi que différentes API pour réaliser ce projet. Nous avons également compris comment échanger des informations entre un client et un serveur, concept que nous utilisons tous les jours en allant sur des sites Web. Ces connaissances nous seront utiles dans la suite de nos études. De plus c'est un projet concret qui nous a permis de voir comment répondre à une problématique en utilisant les informations déjà fournies et disponibles sur internet ainsi que le cours.