Feuille de présence apprenti NFC

Introduction :

L'objectif est de réaliser un système permettant de pouvoir s'emmarger sans avoir recours à une feuille papier.
Nous avons donc pour cela utilisé un reader tikitag (capable de lire des tag NFC), une beaglebones black (pour transmettre les données au server) et un server AWS.

Configuration et dévellopement :

Pour commencer nous avons chercher à nous renseigner sur ce qui existait comme librairie sur tikitag (sachant que c'est une technologie qui à évolué et maintenant se nome touchatag).
Nous avons trouvé un librairie sur : http://membres-liglab.imag.fr/donsez/pub/tikitag/tikitagreader.zip
à partir de celle ci nous avons pu détecter le reader tikitag et lire des tags. Même si il a fallu pour ça installer sur la machine quelques librairies lié à nfc et les libs usb.
Nous avons donc fait deux programmes :
Un sender.java et recever.java pour tester notre application.
Le principe était simple. Récupérer l'identifiant d'un tag NFC dans un string et l'envoyer sur un server de test mqtt avec mosquitto. Ensuite avec le receiver nous verifions que la string était récupéré dans la file d'attente.
Une fois que tout à bien fonctionné, nous étions d'accords sur les concepts pour pouvoir nous séparer le travail. Un qui s'occupé de récupérer les données sur le server via openhab et l'autre qui s'occupait de configurer la beaglebone pour pouvoir utiliser le fichier sender.java.
Pour la beaglebone il a donc fallu réussir à créer un jar avec maven qui permettait de garder les dépendences dans le target jar.
Ensuite il a fallu configurer l'environnement comme nous l'avions fait pour notre machine personnel : installer les librairies nfc et usb (+java)
et enfin réussir à connecter la beaglebone à l'internet via le port USB.

Déploiement sur la vm :

L'objectif des composants déployés sur la vm sont de réceptionner les données envoyées par le composant déployé sur la beaglebone et de traiter ces données sur la vm. Pour remplir ces objectifs nous avons décidé d'utiliser openhab. Nous avons choisi cette technologie d'une part parce qu'il est capable de souscrire à une file d'attente de mosquitto, et d'autre part parce qu'il offre la possibilité de traiter les données notamment en les sauvegardant dans une base de données et en affichant les informations sans une page web avec le format que nous configurons.

Pour souscrire à une file d'attente il faut configurer plusieurs fichiers d'openhab. Le premier est le fichier de configuration générale localisé à la racine du dossier de configuration d'openhab. Dans ce fichier on renseigne l'adresse du broker mosquitto ou les données sont envoyés.

```
mqtt:mosquitto.url=tcp://212.72.74.21:1883
```

Ensuite il faut instancier un item qui sera mis à jour avec la valeur des données reçu. Pour faire cela il faut créer un item dans le ficher de configuration des item, localisé dans le dossier item du dossier

configuration, et lui dire qu'il souscrit à la file d'attente ou les données sont envoyées. L'item que nous avons créer est une string.

```
Group All
Group emargement (All)
Group emargement2 "Emargement" <video> (emargement)
String identifiant "Identifiant : [%s]" <identifiant> (emargement2)
{mqtt="<[mosquitto:NomFileAttente:state:default]"}
```

Les données réceptionnées dans le premier item sont les identifiants des personnes qui ont badgés. Ces données étant peu exploitable pour un etre humain on a choisi d'instancier un deuxième item qui contient le nom de la personne à qui appartient l'identifiant. Pour cela nous avons ajouter un item dans le fichier de configuration des item.

```
String identifiant2 "Identifiant : [%s]" <identifiant> (emargement2) {}
```

Cet item doit etre mis à jour à la reception d'une donnée. Sachant que le premier item se met à jour automatiquement à la reception d'une donnée, nous avons configuré un fichier de régle, dans le dossier rules du dossier configuration, avec une régle simple pour que lorsque le premier item est mis à jour, le second item prend le nom de personne identifié par l'identifiant.

```
rule MappingIdentifiant
when
        Item identifiant received update
then
        if(identifiant.state == "045FC8990B0280") {
                var String test = String::format( "%s", "max")
                postUpdate(identifiant2, test)
        }else{
                var String test = String::format( "%s", "Babasssssss")
                postUpdate(identifiant2, test)
        }
end
```

Enfin nous avons afficher le second item sur le dashboard fourni par openhab. Pour cela nous avons configurer le fichier localisé dans le dossier sitemaps du dossier configurations d'openhab.

```
sitemap demo label="Main Menu"{
        Frame label="Present : " {
                Text item2=identifiant2
        }
}
```

Le resultat est visible dans le navigateur à l'adresse de la vm amazon avec le port 8080.