Installation ESUP-NFC-TAG-SERVER

Pré-requis

- Java (openjdk **17**, 11 ou 8) : le mieux est de l'installer via le système de paquets de votre linux.
- Maven (dernière version 3.x) : le mieux est de l'installer via le système de paquets de votre linux - http://maven.apache.org/download.cgi
- Postgresql 9 ou > : le mieux est de l'installer via le système de paquets de votre linux.
- Tomcat (Tomcat 8.5 ou 9 - Tomcat 10 non supporté) - https://tomcat.apache.org/download-90.cgi
- Apache + libapache2-mod-shib2 : https://services.renater.fr/federation/docs/installation/sp
- Git

Configuration Apache Shibboleth

L'authentification repose sur Shibboleth. Apache doit être configuré pour faire du mod_shib.

Une fois le SP Shibboleth et Apache configurés usuellement (voir : https://services.renater.fr/federation/docs/installation/sp), en plus de l'usuel /secure, il faut sécuriser /manager, /admin et /nfc en ajoutant ceci à la conf apache (à adapter cependant en fonction des versions d'Apache et mod_shib) :

```
<Location /secure>
AuthType shibboleth
ShibRequestSetting requireSession 1
require shib-session
ShibUseHeaders On
</Location>


<Location /manager>
AuthType shibboleth
ShibRequestSetting requireSession 1
require shib-session
ShibUseHeaders On
</Location>


<Location /admin>
AuthType shibboleth
ShibRequestSetting requireSession 1
require shib-session
ShibUseHeaders On
</Location>

<Location /nfc>
AuthType shibboleth
ShibRequestSetting requireSession 1
require shib-session
ShibUseHeaders On
</Location>
```

Attention, il ne faut pas sécuriser le dossier racine "/" pour laisser l'accès libre à certains web service ainsi qu'a l'affichage mobile

Configuration PostgreSQL

- pg_hba.conf : ajout de

```
host all all 127.0.0.1/32 password
```

- redémarrage de postgresql
- psql

```
create database esupnfctag;
create USER esupnfctag with password 'esup';
grant ALL ON DATABASE esupnfctag to esupnfctag;
ALTER DATABASE esupnfctag OWNER TO esupnfctag;
```

Paramétrage mémoire JVM :

Pensez à paramétrer les espaces mémoire JVM :

```
export JAVA_OPTS="-Xms1024m -Xmx1024m -XX:MaxPermSize=256m"
```

Pour maven :

```
export MAVEN_OPTS="-Xms1024m -Xmx1024m -XX:MaxPermSize=256m"
```

Sources : https://github.com/EsupPortail/esup-nfc-tag-server

```
cd /opt
git clone https://github.com/EsupPortail/esup-nfc-tag-server
```

Configurations

Versioning de la configuration

D'une manière générale il est conseillé de faite des commits GIT locaux pour sauvegarder vos modification de configuration.

Par exemple, lors que vous avez fini la configuration du fichier "src/main/resources/META-INF/spring/applicationContext-custom.xml" vous pouvez executer les commandes suivantes depuis la racine de vos sources (/opt/esup-sgc):

```
git add src/main/resources/META-INF/spring/applicationContext-custom.xml
git commit -m "config prod univ-ville-fr"
```

La modification du fichier applicationContext-custom.xml sera versionnée ce qui permet de concerver les traces de toutes les modification (procédure à faire à chaque modification). De plus lors d'une mise à jour d'ESUP-SGC (git pull) la configuration ne sera pas écrasée.

Au niveau de l'application, les fichiers de configuration à modifier sont les suivants :

- src/main/webapp/META-INF/context.xml pour le paramétrage des connexions à la base de données
- src/main/resources/logback.xml pour la configuration des logs
- src/main/resources/META-INF/spring/applicationContext-custom.xml pour le paramétrage des services que doit intégrer l'application EsupNfcTagServer ; c'est le fichier de paramétrage important de l'application et ce sont ces paramétrages qui sont décrits dans ce qui suit.

`3 services distincts correspondent à l'implémentation des apis définis dans org.esupportail.nfctag.service.api :`

- NfcAuthConfig
- TagIdCheckApi
- AppliExtApi

NfcAuthConfig

NfcAuthConfig correspond à la façon dont la carte est authentifiée/identifiée.

Les 2 implémentations proposées actuellement sont l'authentification par lecture du simple CSN (Card Serial Number) d'une part, et l'authentification par Mifare Desfire AES d'autre part. Ces implémentations sont très liées à EsupNfcTagDroid (ie l'APK, la partie mobile) aussi une nouvelle implémentation d'un NfcAuthConfig nécessitera(it) sans doute la modification de EsupNfcTagDroid.

Implémentations actuellement disponibles :

- `CsnAuthConfig : implémentation de la lecture csn`
- `DesfireAuthConfig : implémentation de la récupération d'un identifiant issu de la lecture d'un fichier chiffré Mifare Desfire AES`

Les 2 autres services TagIdCheckApi et AppliExtApi sont indépendants quant à eux de EsupNfcTagDroid et de nouvelles implémentations de ces services peuvent être envisagées, même si le recours à leurs implémentations sous forme de Web Service REST doit être naturellement privilégiées.

`Exemple AuthConfig`

Pour une configuration d'authentification de carte par CSN :

```
<bean id="csnAuthConfig" class="org.esupportail.nfctag.service.api.impl.CsnAuthConfig">
<property name="description" value="Authentification CSN"/>
</bean>
```

Pour une configuration d'authentification de carte par Mifare Desfire AES :

```
<bean id="desfireAuthConfig" class="org.esupportail.nfctag.service.api.impl.DesfireAuthConfig">
        <property name="desfireKeyNumber" value="01"/>
        <property name="desfireAppId" value="A123F1"/>
        <property name="desfireAppName" value="test-app"/>
        <property name="readFileCommand" value="90BD000007000000016000000"/>
        <property name="desfireKey" value="/var/local/key"/>
        <property name="description" value="Authentification DESFIRE"/>
</bean>
```

TagIdCheckApi

TagIdCheckApi correspond à la façon dont on récupère l'identification d'un individu depuis un identifiant de carte (identifiant CSN ou issu de la lecture d'un fichier Desfire). Cette identification de l'individu consiste en son eppn, nom, prénom.

Implémentations actuellement disponibles :

- `TagIdCheckRestWs : recupération des identifants via webservices REST`
- `TagIdCheckSql : recupération des identifants via une base de données`
- `TagIdCheckLdap : recupération des identifants via un annuaire LDAP`

Exemple TagIdCheck

```
<bean id="tagIdCheckApiCarteCulture" class="org.esupportail.nfctag.service.api.impl.TagIdCheckRestWs">
        <property name="tagIdCheckUrl" value="https://carte-culture.univ-rouen.fr/nfc-ws/tagIdCheck"/>
        <property name="description" value="via Carte Culture"/>
</bean>
```

- `tagIdCheckUrl : adresse du webservice permettant de retrouver une personne en fonction de sont identifiant de carte (csn ou idp2s)`

`Pour tester l'application rapidement, ajouter ce tagIdCheck (qui retournera toujours un résultat):`

```
<bean id="tagIdCheckApiDummy" class="org.esupportail.nfctag.service.api.impl.TagIdCheckDummyWs">
        <property name="description" value="TagIdCheckDummy"/>
</bean>
```

AppliExtApi

AppliExtApi correspond à l'application cible finale du badgeage :

- il retourne d'une part les "locations" sur lesquels peut éventuellement badger l'utilisateur qui se trouve derrière le téléphone : on utilise l'eppn (issu de l'authentification shibboleth) pour identifier cette personne.
- d'autre part, il propose des méthodes correspondant au badgeage sur la "location" de la personne détentrice de la carte ; on utilise ici l'eppn (issu ici du tagIdCheckApi lui même issu du csn ou identifiant desfire issu du nfcAuthConfig) pour identifier cette personne détentrice de la carte.

Implémentations actuellement disponibles :

- `AppliExtDummy : permet de tester l'application, elle renvoit un lieu (Dummy Location)`
- `AppliExtRestWs : s'appuie sur des webservices REST pour fournir les lieux et generer les badgeages`

`Les services utilisables avec la carte peuvent être configurés dans le fichier` **`src/main/resources/META-INF/spring/applicationContext-custom.xml`**`.`

Exemple AppliExtDummy

```
<bean id="exempleDummyExtApi" class="org.esupportail.nfctag.service.api.impl.AppliExtDummy">
          <property name="description" value="Salle de test"/>
          <property name="locationsNames">
                  <util:list>
                          <value>Salle de test</value>
                  </util:list>
          </property>
       <property name="eppnFilter" value=".*"/>
</bean>
```

- Il est possible de filtrer les eppn des personnes autorisées à voir l'application Dummy

Exemple AppliExtRestWs

```
<bean id="esupSgcExtApi" class="org.esupportail.nfctag.service.api.impl.AppliExtRestWs">
       <property name="isTagableUrl" value="https://esup-sgc.univ-ville.fr/wsrest/nfc/isTagable"/>
       <property name="validateTagUrl" value="https://esup-sgc.univ-ville.fr/wsrest/nfc/validateTag"/>
       <property name="locationsUrl" value="https://esup-sgc.univ-ville.fr/wsrest/nfc/getLocations"/>
<!--           <property name="displayUrl" value="https://esup-sgc.univ-ville.fr/wsrest/nfc/verso"/> -->
       <property name="description" value="Web Service Carte Culture"/>
       <property name="backgroundColor" value="rgb(121, 119, 116)"/>
       <property name="header" value="https://carte-culture.univ-rouen.fr/resources/images/logo.jpg"/>
</bean>
```

- isTagable : adresse du webservice permettant de contrôler si un badge est valide
- validateTagUrl : adresse du webservice permettant de confirmer un badgeage
- locationsUrl : adresse du webservice retournant la liste des « lieux » disponible pour l'utilisateur courant (utilisateur du lecteur de carte)
- displayUrl (non requis) : adresse du webservice permettant l'affichage d'information après la validation du tag

Base de données

src/main/resources/META-INF/persistence.xml

afin que les tables soient préalablement créées, notamment la table big_file sur lequel on souhaite mettre le trigger lo_manage, vous devez démarrer l'application une fois ; en n'oubliant pas ensuite, pour ne pas écraser la base au redémarrage, de modifier src/main/resources/META-INF/persistence.xml : create-> update - cf ci-dessous.

src/main/resources/META-INF/spring/database.properties

```
database.driverClassName=org.postgresql.Driver
database.url=jdbc\:postgresql\://localhost\:5432/esupnfctag
database.username=esupnfctag
database.password=esup
```

Logs

src/main/resources/logback.xml

modifier le fichier pour paramétrer l'adresse mail d'envoi et le chemin du fichier de log

Obtention du war pour déploiement sur tomcat ou autre :

```
cd /opt/esup-nfc-tag-server
mvn clean package
```

Déploiement

On copie/colle le répertoire webapp packagé ainsi dans le tomcat :

```
rm -rf /opt/tomcat-esup-nfc-tag-server/webapps/ROOT && cp -rf /opt/esup-nfc-tag-server/target/esupNfcTagServer-2.0.0-SNAPSHOT /opt/tomcat-esup-nfc-tag-server/webapps/ROOT
```

On arrête le tomcat avant et on le redémarre ensuite.

Lancement de la mise à jour de la base de données

```
mvn exec:java -Dexec.args="dbupgrade"
```

Droits utilisateur

Le rôle ROLE_ADMIN est necessaire pour gérer l'application

Il est à paramétrer dans application-context-security.xml. L'authentification/identification se fait en shibboleth, le credentialsRequestHeader est à paramétrer en fonction de vos attributs shibboleth; ainsi que le authUserDetailsService.

Paramétrage via l'IHM

A cette adresse : https://esupnfctag.univ-ville.fr/manager/applications

Il faut tout d'abord ajouter une application sur laquelle un péri?érique pourra se connecter. Le formulaire d'ajout propose les choix suivants :

- Configuration NFC (CSN, Desfire ou config spécifique SGC)
- Application externe (voir Implementation du webService AppliExtRestWs)
- Contrôle du tagId (voir Implémentation du Web Service TagIdCheck)
- Valeur par défaut pour la validation sans confirmation (si oui, les périphériques qui se brancheront sur cette application ne demanderont pas de validation manuelle lors du badgeage)

- Puis en mode modification il est possible de désactiver une application. Cela masque l'application mais elle reste utilisable.

Pour un test, choisir Authentification CSN, Dummy !, le tagIdCheck qui convient (voir implémentation du tagIdCheck)

Test de l'application sans EsupNfcTagDroid et sans EsupNfcTagArduino

L'idée est de tester le process de badgeage en simulant les appels HTTP du badgeur EsupNfcTagDroid et/ou EsupNfcTagArduino

En utilisant cette adresse :

https://esupnfctag.univ-ville.fr/nfc-index?apkVersion=1-2099-01-01-00-00-00-dev&imei=123456

L'application va vérifier les différent service autorisés pour la personne connectée et créer une entrée dans Device (dans notre cas Dummy location) puis rediriger vers le « live » de ce lieu.

http://esupnfctag.univ-ville.fr/live?numeroId=6847041179388220887

Il est alors possible de simuler un badgeage csn via une commande curl ex :

```
curl -X POST -H "Content-type:application/json" -d '{"csn":"045371d2fd3a80","numeroId":"6847041179388220887"}' http://esupnfctag.univ-ville.fr/csn-ws
```

Une fenetre de validation doit apparaître sur le « live »