SFL2 - TRAÇABILITE RFID

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dossier technique du projet – Partie Personnelle

Session 2019

Pierre GORGE



**Table des matières**

[I. Situation dans le projet 1](#_Toc9362301)

[A. Rappel des tâches de l’étudiant 1](#_Toc9362302)

[B. Déroulement du projet 1](#_Toc9362303)

[C. Contraintes de développement 2](#_Toc9362304)

[II. Conception et mise en œuvre 3](#_Toc9362305)

[A. Fonctionnement du NFC sous Android Studio 3](#_Toc9362306)

[B. Détection du NFC par le téléphone 4](#_Toc9362307)

[a. Création de la variable « nfcAdapter » 4](#_Toc9362308)

[b. Vérification de l’utilisation du NFC 5](#_Toc9362309)

[c. Le téléphone ne possède pas de NFC 6](#_Toc9362310)

[C. Lecture du tag RFID par NFC 6](#_Toc9362311)

[a. Détection du tag et autorisation de l’afficher 6](#_Toc9362312)

# Situation dans le projet

## Rappel des tâches de l’étudiant

Dans ce projet de traçabilité RFID, j’avais pour objectif, avec mon collègue Julio César Gomez Reyes, de créer une application Android, utilisable sur la plupart des téléphones Android, ayant pour but l’emprunt d’objets via le scan d’un tag RFID. Nous avons pour cela dû nous approprier les librairies existantes pour l’utilisation de la technologie NFC sur les smartphones Android.

## Déroulement du projet

Dans un premier temps, nous avons réfléchi aux différents moyens de mettre en œuvre le projet. Nous avons très vite déduit que de faire deux applications séparées étaient certes plus facile à mettre en place mais moins cohérent et pratique pour le service demandé. Ainsi, nous avons décidé de ne créer qu’une seule application et de travailler main dans la main avec mon collègue Julio César Gomez Reyes.

Nous nous sommes donc concentrés ensemble avec Julio, sur la conception de l’interface de l’application, pour ensuite me pencher sur l’implémentation d’une lecture d’un tag par le smartphone. J’ai aussi travaillé sur la création d’une partie QR Code pour une adaptabilité aux téléphones démunit de NFC.

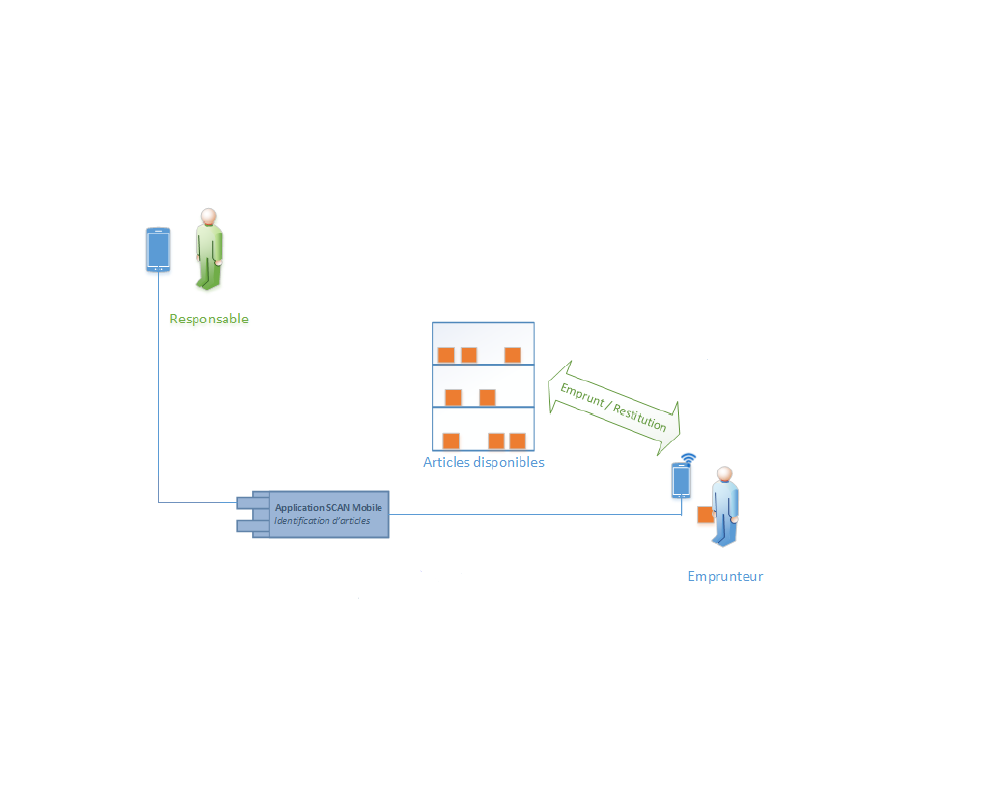


Schéma de la partie concernant l’étudiant 3

Pour cela, j’ai dû me plonger dans les librairies utilisant la technologie NFC et dans une utilisation précise et complexe du logiciel Android Studio.

|  |
| --- |
| Une image contenant clipart  Description générée automatiquementAndroid Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS et Linux3. |

La librairie NFC est très alambiquée et la documentation est étonnamment peu précise. Il m’a fallu beaucoup de temps pour comprendre comment l’utiliser et pour commencer à la maitriser. En effet, au bout de deux mois, j’ai réussi à implémenter une lecture de NFC par le téléphone. Cette tâche m’a demandé beaucoup de temps car très complexe à mettre en place. J’ai aussi travaillé sur la solution du QR Code mais qui n’a pour le moment pas encore aboutie. Une fois la lecture du tag réussi, je me suis concentré sur la création de ma partie de l’interface. Elle a évolué dans le temps avec les différents ajouts fait sur l’application au cours du projet.

La réalisation de l’application s’est déroulée en trois grandes étapes (spécifications, analyse, conception) qui suivent le diagramme de Gantt présent en annexe.

## Contraintes de développement

Le logiciel Android Studio était une totale découverte et une véritable contrainte sur le projet. Nous maîtrisons le langage Java, ce qu’utilise Android Studio, cependant l’architecture du logiciel étant très particulière et propre à lui, il a fallu un temps assez important pour trouver ses marques et savoir comment l’utiliser de manière efficace.

Les librairies NFC ont été une difficulté, au même niveau que le logiciel qui les utilise. La documentation n’est pas très exhaustive, ni même explicite. Il m’a fallu beaucoup de temps, de travail et de patience pour trouver comment les utiliser correctement et les implémenter.

Les téléphones ont aussi été sources de contraintes. En effet, ils sont développés sous Android mais les constructeurs mobiles l’ont tous modifié pour avoir leur propre version et certaines de ces versions créer quelques problèmes avec l’application. Par exemple, il faut impérativement désactiver le débogage USB sur le téléphone de Maxime pour que l’application fonctionne correctement.

# Conception et mise en œuvre

## Fonctionnement du NFC sous Android Studio

Sur Android Studio, toutes les classes et méthodes sont répertoriés dans des librairies. Ce sont des classes qui contiennent toutes les classes et méthodes sur une fonctionnalité donnée. Il nous faut beaucoup de librairies différentes pour notre application. Notamment, pour la connexion à la base de données, pour l’API Google Maps et pour le NFC.

Exemples de librairies utilisées pour le NFC

On remarque que l’on utilise un « import » pour ajouter les librairies sur notre projet. En effet, les librairies sont stockées sur Android Studio mais elles ne sont pas toutes automatiquement implémentées sur le projet que l’on créer, cela permet d’éviter de surcharger le projet et de le rendre plus lent.

|  |
| --- |
| *Une image contenant intérieur  Description générée automatiquementNote :* Android Studio est développé par Google mais reste un logiciel libre, il est donc possible de créer de librairies soit même et de les rendre disponible sur internet. Ainsi, on peut se retrouver avec des besoins pour certaines fonctionnalités ne se trouvant pas sur le logiciel nativement (Même certaines fonctionnalités Android tel que Google Maps ne se retrouve pas nativement sur le logiciel). A ce moment-là, on se dirige vers un dossier du projet appelé « build.gradle », auquel on ajoute une ligne « Implementation » suivi d’un lien qui permet à l’IDE (Environnement de Développement Intégré) de télécharger la librairie et ainsi permettre au développeur de l’utiliser.  Ajout de librairies « externes » |

L’implémentation du NFC se déroule en plusieurs parties et se trouve dans la classe « NFCScan » créer spécialement pour gérer tous les éléments se trouvant sur la page du scan du tag RFID.

## Détection du NFC par le téléphone

### Création de la variable « nfcAdapter »

Avant toute chose, le téléphone doit détecter si le NFC est activé. Si l’utilisateur oublie de l’activer il faut que le téléphone puisse prévenir et amener l’utilisateur à l’activer.

Pour cela, après avoir initialisé les variables, on commence par créer une méthode « onCreate » où l’on va donner une valeur à tous les boutons et les fonctions du téléphone à activer.

Une image contenant capture d’écran

Description générée automatiquement

*Note :* Les méthodes « onCreate » sont celles qui sont appelés en premier au lancement de l’application. Les variables sont créées et pourront alors être utilisées par l’application. C’est très intéressant (voir nécessaire) à utiliser car cela permet au système de recréer les variables si jamais on retourne sur le menu d’accueil de notre téléphone. Si cette classe n’est pas utilisé, les variables seraient créées une seule fois, un retour au menu du téléphone sans fermer l’application reviendrait à faire disparaître les variables et y revenir ne les ferait pas réapparaitre. L’application planterai à ce moment-là.

On se concentre sur la variable « nfcAdapter » et on lui donne comme valeur la fonction NFC par défaut du téléphone :

Une fois que cette valeur est assignée, « nfcAdapter » possède la valeur du NFC sur le téléphone, c’est-à-dire que cette valeur peut savoir si le NFC est activé ou non.

### Vérification de l’utilisation du NFC

Une image contenant capture d’écran

Description générée automatiquementAinsi, on créer une méthode pour vérifier que le NFC est bien activé au moment où l’on veut scanner un tag RFID. Cette méthode s’appelle « onResume » et se présente de cette manière :

*Note :* Les méthodes « onResume » sont utiles quand les applications sont mises en pause en revenant à l’accueil par exemple. Au moment de revenir sur l’application, cette méthode sera appelée avant la méthode « onCreate » vu précédemment. Cela permet d’établir un ordre dans le relancement des méthodes et évite que certaines méthodes qui dépendent les unes des autres ne se lancent en même temps et créer des bugs.

Le « if » permet de vérifier si le NFC est activé. En arrivant sur la page de lecture du tag, si le NFC n’est pas activé (le « if ») alors on rentre dans cette boucle.

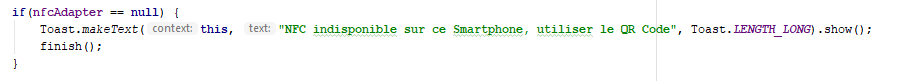
Une image contenant capture d’écran

Description générée automatiquementDans cette boucle, on créer un nouvel « AlertDialog.Builder » qui s’affichera sur la page sur laquelle on se trouve « (context : NFCScan.this) »

Ce morceau de texte nous permet de personnaliser notre message d’alerte. Ici on décide avec « builder.setTitle », « builder.setMessage » et « builder.setPositiveButton » respectivement, du titre, du message et d’un bouton « ok » sur le message d’alerte.

Une fois que l’on appuie sur le bouton « ok », on a fait en sorte de démarrer une nouvelle page qui est celle des paramètres NFC. Ce qui permet à l’utilisateur d’activer son NFC et de revenir sur l’application.

### Le téléphone ne possède pas de NFC

 Un morceau de code a aussi été rajouté pour bloquer l’accès à la page de lecture de tag via NFC si le téléphone ne possède pas de NFC. Certains téléphones un peu anciens ne possède effectivement pas cette technologie, ainsi, on a créé une condition dans laquelle « nfcAdapter » est nul :

La fonction « Toast.makeText » permet de créer une bulle de texte qui apparait pendant environ cinq seconde en bas de l’écran. Il s’agit d’une bulle de texte système utilisé pour afficher des messages qui ne nécessite pas de grande boite de dialogue ou d’alerte tel que celle utilisée pour l’activation du NFC. On ne peut pas interagir avec elles.

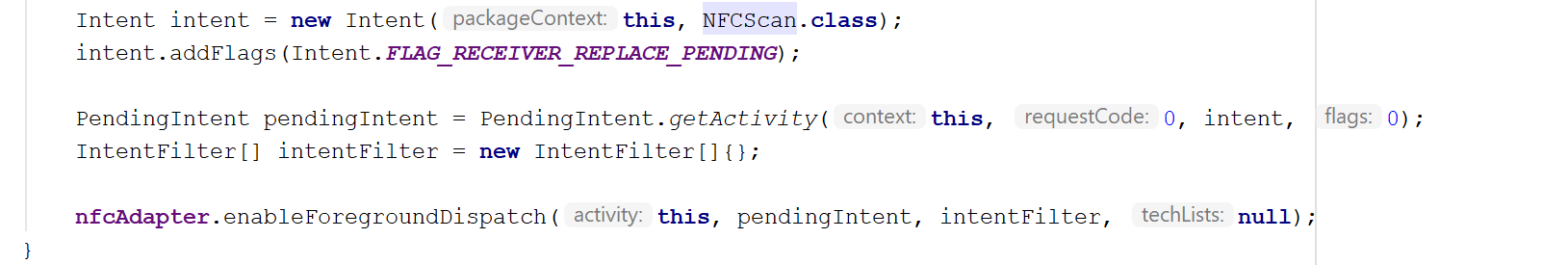
La méthode « finish() ; » quant à elle, termine la page ou « Activity » que l’on essaye d’atteindre. Ainsi, un téléphone qui ne possède pas de NFC ne pourra pas arriver sur la page de lecture du tag.

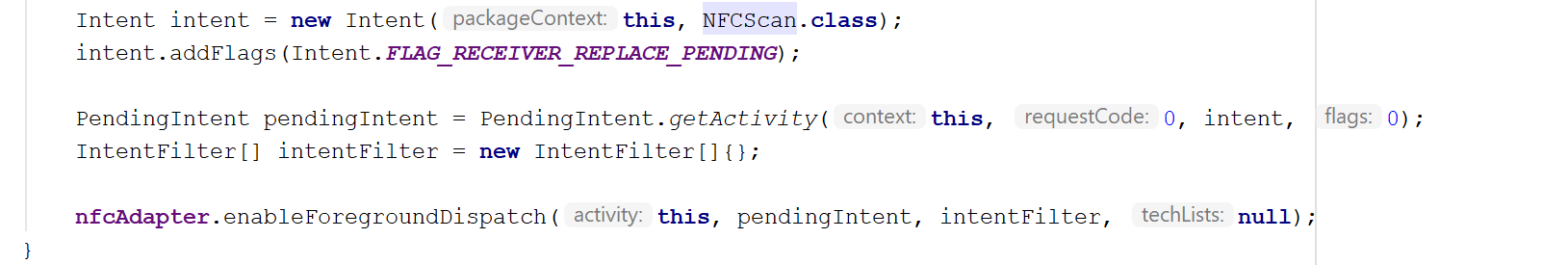
## Lecture du tag RFID par NFC

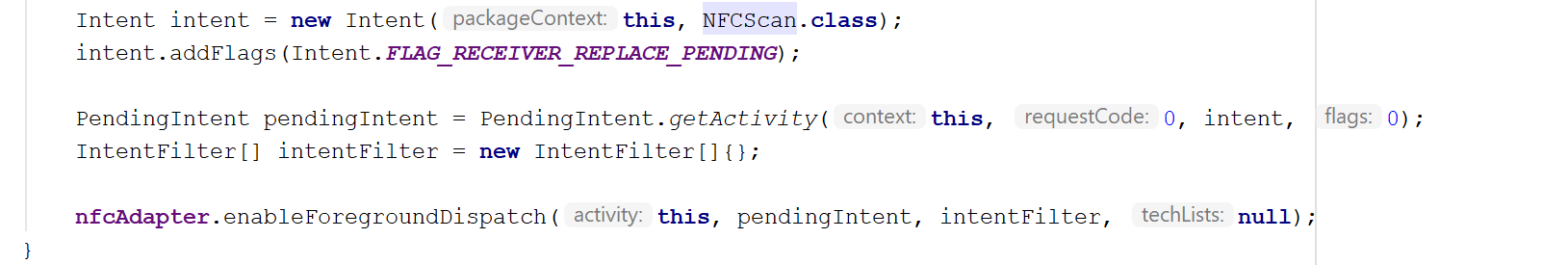
### Détection du tag

Cette partie est très complexe et fait appel à beaucoup de méthodes différentes. Nous allons détailler chaque méthode pour essayer de comprendre le fonctionnement d’une lecture de tag RFID.

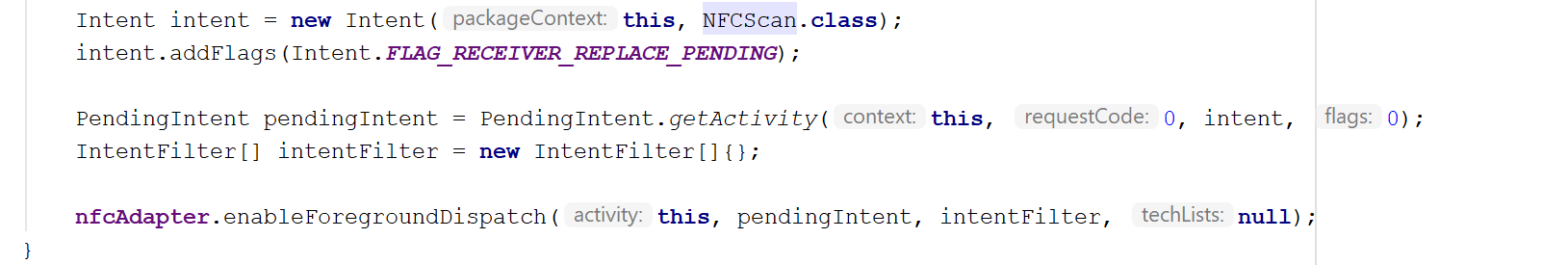
En premier lieu, il faut que l’application soit prioritaire sur la lecture du tag présenté. En effet, les différentes versions de l’OS Android des constructeurs mobiles possèdent déjà un lecteur de tag intégré. Il n’est cependant pas très poussé et ne permet que de détecter un tag, pas forcément d’en lire le contenu. Il reste malgré tout prioritaire et il faut faire en sorte que quand l’application est lancée, ce soit elle qui est la priorité sur la lecture d’un tag.

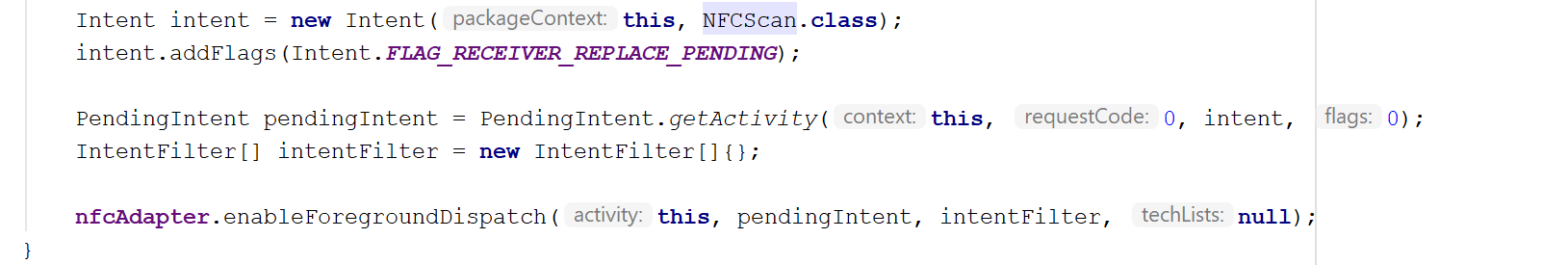
On utilise donc la méthode « enableForgroundDispatcher ». Cette méthode permet littéralement de prendre la priorité sur d’autres activités, notamment le lecteur de l’OS, et de faire en sorte que ce soit l’application dans lequel il est appelé qui récupère la lecture du tag.

La méthode en elle-même permet donc de lire le tag. C’est son seul rôle. Le reste se fait au niveau de « l’intent ». Cette méthode est fondamentale dans notre projet et notamment dans la lecture, et surtout l’affichage, de notre tag.

« L’intent » est une variable qui reçoit une information d’action et qui va réagir si on l’utilise. On initialise un « new Intent » sur cette classe précisément. Quand un tag va être scanné, l’information « un tag est détecté » est enregistré dans l’Intent.

Cette ligne permet le remplacement du tag si un autre est scanné. Cela évite de scanner deux tags à la suite et que les deux se mélangent. **Si un deuxième tag est scanné, alors il va remplacer le premier.**



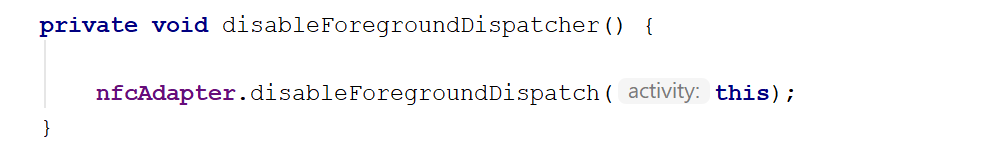
 Le « PendingIntent » n’est pas en lien direct avec l’intent en lui-même. Son but est de permettre l’exécution de l’Intent avec les permissions de notre application. En effet, notre application ne possède pas de lecteur à lui. Il utilise celui de l’OS en place, ici Android. Comme il ne fait pas parti de notre application, **il faut que ce lecteur puisse avoir accès à l’autorisation prioritaire de notre « enableForgroundDisplay ».** Cette ligne permet au lecteur du système d’y accéder.

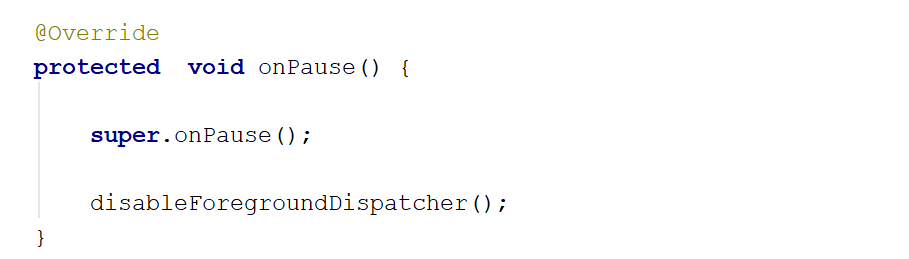
Comme il peut exister plusieurs Intent de déclaré dans l’application, il faut les filtrer. L’ « IntentFilter » permet cela. On déclare quelles classes utilisent des Intents qui doivent être filtré.

Déclaration des IntentFilter dans le Manifest

Une fois déclaré, seuls les Intent créer et appelé dans ces classes, ici « InterfaceNFCQR » et « NFCScan » pourront être récupérer avec un IntentFilter.

Finalement, on précise à la fin que la méthode dépend de « nfcAdapter » avec toutes ses caractéristiques qui lui permet de passer le « IntentFilter »



 On créer aussi une méthode pour détruire cette permission de lecture.

Cette méthode est appelée dans une méthode « onPause » qui se lance quand on sort de l’application pour retourner sur le menu principal de notre téléphone sans pour autant fermer l’application. Quand c’est le cas, si jamais on scan un tag RFID, **on ne veut pas que ce soit notre application qui prenne le relais, mais bien le lecteur du système. On désactive donc temporairement les droits prioritaires de notre application avec la méthode « disableForgroundDispatcher »**