Berney Alec & Forestier Quentin & Herzig Melvyn

SYM – 07.12.2021

Codes-barres, iBeacons et NFC

Labo 3



Table des matières

[Introduction 2](#_Toc89181362)

[Démarrage 2](#_Toc89181363)

[Remarques préalables 2](#_Toc89181364)

[Manipulations 2](#_Toc89181365)

[Balises NFC 3](#_Toc89181366)

[Utilisation 3](#_Toc89181367)

[Mise en place 3](#_Toc89181368)

[Question 3](#_Toc89181369)

[Codes barre 4](#_Toc89181370)

[Utilisation 4](#_Toc89181371)

[Mise en place 5](#_Toc89181372)

[Question 5](#_Toc89181373)

[IBeacons 6](#_Toc89181374)

[Utilisation 6](#_Toc89181375)

[Mise en place 6](#_Toc89181376)

[Question 6](#_Toc89181377)

# Introduction

Ce laboratoire est une introduction à l’utilisation des données environnementales avec Android. Ces données sont :

* Les balises NFC
* Les codes barre
* Les iBeacons

# Démarrage

L’application démarre sur la page principale :



Chaque bouton crée une nouvelle activité qui met en scène les différentes données environnementales citées précédemment.

## Remarques préalables

Ce laboratoire est une introduction aux données environnementales. En conséquence, certain aspect de la programmation Android ne sont pas développés par manque de temps. Par exemple, les activités n’utilisent pas les principes des view-models.

# Manipulations

Comme présenté préalablement, ce laboratoire consiste en 3 manipulations majeures. Nous avons donc réalisé une activité par manipulation. Au cours des points suivants, nous allons décrire de manière incrémentale ce que nous avons mis en place pour arriver à l’état final de ce laboratoire.

## Balises NFC

todo

### Utilisation

Todo

### Mise en place

Todo

### Question

Todo

## Codes barre

### Utilisation

Au lancement le layout ressemble à ceci :

Une image contenant texte

Description générée automatiquement

Figure Activité code barre - démarrage

La zone supérieure est la zone enregistrée par la caméra. Lorsqu’elle détecte un code barre ou un QR code, elle affiche la frame sur laquelle la détection a eu lieue ainsi que la valeur du scan.



Figure Activité code barre - scan réussi

### Mise en place

**Démarrage et permissions**

L’activité est démarrée à partir de la MainActivity. Avant le démarrage de l’activité, au clic sur le bouton pour l’ouvrir, nous vérifions si l’application a la permission d’utiliser la caméra.

codeBarreButton.setOnClickListener**{** *// Check for permissions, if acquired, starts CodeBarreActivity* checkPermission(Manifest.permission.*CAMERA*, CAMERA\_PERMISSION\_CODE)  
**}**

La fonction checkPermission, pour une permission donnée, effectue une demande à l’utilisateur pour valider l’accès à la permission demandée. Si la permission est déjà accordée, l’activité est démarrée.

private fun checkPermission(permission: String, requestCode: Int) {  
 if (ContextCompat.checkSelfPermission(  
 this@MainActivity, permission) == PackageManager.*PERMISSION\_DENIED*)

{  
  
 *// Requesting the permission* ActivityCompat.requestPermissions(this@MainActivity,   
 *arrayOf*(permission), requestCode)  
 } else {  
 if(requestCode == CAMERA\_PERMISSION\_CODE) {  
 startActivity(Intent(this, CodeBarreActivity::class.*java*))  
 }  
 }  
}

Dans le cas où la permission est demandée à l’utilisateur, nous avons override la méthode de callback utilisée lorsqu’une permission est accordée ou refusée. Si la permission est accordée, l’activité est démarrée.

override fun onRequestPermissionsResult(requestCode: Int,  
 permissions: Array<String>,  
 grantResults: IntArray) {  
  
 super.onRequestPermissionsResult(requestCode,

permissions,

grantResults)  
 if (requestCode == CAMERA\_PERMISSION\_CODE) {  
 if (grantResults.*isNotEmpty*() &&

grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 startActivity(Intent(this, CodeBarreActivity::class.*java*))  
 }  
 }  
}

**CodeBarreActivity**

L’activité mise en place est simple. Elle fait intervenir trois composants :

* un DecoratedBarcodeView (barcodeView) pour afficher le décodeur de code barre
* une ImageView (imageView) pour afficher l’image capturée pour décodage
* une TextView (result) pour afficher la valeur décodée.

A la création de l’activité, nous récupérons des références sur chacun des éléments puis nous initialisons le lecteur de codes-barres.

val formats: Collection<BarcodeFormat> =  
 Arrays.asList(BarcodeFormat.*QR\_CODE*, BarcodeFormat.*CODE\_39*)  
barcodeView.*barcodeView*.*decoderFactory* = DefaultDecoderFactory(formats)  
barcodeView.initializeFromIntent(*intent*)  
barcodeView.decodeContinuous(callback)

Lorsque le barcodeView décode une image il appelle un objet callback pour gérer le résultat du décodage. C’est ce que fait la dernière ligne de code précédente.

Ainsi, nous avons définit l’objet callback de la manière suivante :

private val callback: BarcodeCallback = object : BarcodeCallback {  
 override fun barcodeResult(result: BarcodeResult) {  
 if (result.*text* == null || result.*text* == lastText.*text*) {  
 *// Prevent duplicate scans* return  
 }  
 lastText.*text* = result.*text* beepManager.playBeepSoundAndVibrate()  
  
 *//Added preview of scanned barcode* imageView.setImageBitmap(result.getBitmapWithResultPoints(Color.*YELLOW*))  
 }  
  
 override fun possibleResultPoints(resultPoints: List<ResultPoint>) {}  
}

Nous avons simplement override la méthode barecodeResult pour mettre à jour l’ImageView avec la frame qui a été capturée et pour afficher la valeur du décodage dans la TextView.

Finalement, les méthodes onResume, onPause et onKeyDown ont été override pour appeler les méthodes du même nom sur le barecodeView, typiquement pour arrêter l’enregistrement lorsque l’activité est mise en pause afin de ne pas consommer des ressources inutilement.

### Question

**Quelle est la quantité maximale de données pouvant être stockée sur un QR-code ? Veuillez expérimenter, avec le générateur conseillé5 de codes-barres (QR), de générer différentes tailles de QR-codes.**

En jouant avec le générateur, pour une correction d’erreur LOW, nous avons généré un QR code pour 2953 caractères

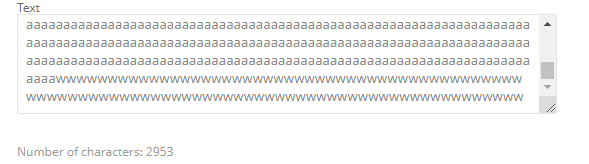
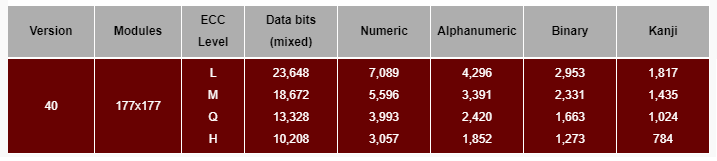


Figure exemple de création de code QR - qté de caractères

Ce qui est curieux, lorsque nous regardons la [spécification](https://www.qrcode.com/en/about/version.html) des QR codes, nous apercevons que le 2953 précédent correspond à la quantité de données binaires pour la version la plus grosse des QR code.



En conclusion, en vue de la manipulation, nous définissons la taille maximale des QR code à 2953 caractères textuels. Attention, la quantité varie en fonction de taux de la correction d’erreur.

**Pensez-vous qu’il est envisageable d’utiliser confortablement des QRcodes complexes (par exemple du contenant >500 caractères de texte, une vCard très complète ou encore un certificat Covid) ?**

En règle générale oui. Toutefois, faisons la nuance, car cela dépend de nombreux facteurs :

Premièrement la correction d’erreur influe. Au format vignette standard, un texte de 1000 caractères en LOW sera scannable tandis que le même texte en HIGH ne sera pas scannable car trop détaillé.

|  |  |
| --- | --- |
| Figure texte de 1000 caractères en LOW ECC | Figure texte de 1000 caractères en HIGH ECC |

En admettant que la taille du QR code soit au format « vignette » traditionnelle (environ 300px sur 300px) et que l’ECC soit LOW, nous avons pu monter jusqu’à 1000 caractères. Aux alentours de 500 caractères aucun problème de scan. Cependant, aux alentours de 1000 caractères, il faut parfois s’y prendre 2 à 3 fois avant de décoder correctement le code.

Au-delà de 1000 caractères le scanner n’arrive presque plus à décoder le code QR dans son ensemble, il en décode des fragments.

En conclusion, oui il est possible de travailler confortablement avec des QR codes avec plus de 500 caractères. Pour preuve, les certificats Covid sont déjà traités au format QR code dans l’application Covid Cert. Toutefois, il faut faire attention à ce que le nombre de modules (en largeur et hauteur) ne soit pas trop élevé pour permettre aux caméras de correctement les distinguer. Si les modules sont trop petits, il faut veiller à afficher le QR code aussi grand que possible pour faciliter l’analyse des caméras. En outre, il faut également prendre en considération l’éclairage et la qualité des caméras.

**Il existe de très nombreux services sur Internet permettant de générer des QR-codes dynamiques. Veuillez expliquer ce que sont les QR-codes dynamiques. Quels sont les avantages et respectivement les inconvénients à utiliser ceux-ci en comparaison avec des QR-codes statiques. Vous adapterez votre réponse à une utilisation depuis une plateforme mobile.**

Les QRcodes statiques encodent directement l’information dans le code QR, ainsi pour modifier le contenu, le code devra être regénéré et le pattern sera différent.

Les QRCodes dynamiques encodent une URL courte qui, une fois scannée, redirige l’utilisateur vers une URL avec son contenu. Ainsi pour modifier le contenu, on ne modifie pas l’URL et donc nous ne modifions pas le QRcode.

Avantages :

* Permet de changer le contenu sans avoir besoin de regénérer le QR code.
* Les QR codes dynamiques sont généralement plus courts que les QR codes statiques, ils sont donc plus facilement scannables.
* Possibilité de monitorer l’accès au contenu.

Inconvénients :

* Nécessitent une connexion à internet pour fonctionner.
* Augmentation de l’utilisation du réseau. Comme en général l’URL utilisée est une URL courte, cela implique une redirection. De plus les données ne sont plus décodées mais téléchargées. Ainsi nous consommons plus de ressources du mobile pour récupérer le contenu.

## IBeacons

todo

### Utilisation

todo

### Mise en place

Todo

### Question

todo