

Université de Montréal

Département d'Informatique et Recherche Opérationnelle

# PROJET INFORMATIQUE – IFT 3150

## RAPPORT FINAL



**KRYPTNATION**

**Kevin Palmang KOMBATE**

Superviseur: Abdelhakim Senhaji HAFID

# RESUME

La cryptomonnaie est une monnaie alternative (car toutes les cryptomonnaies n'ont de cours légal dans aucun pays) 100% virtuelle, utilisable sur un réseau informatique décentralisée, de pair à pair. Elle est fondée sur les principes de la **cryptographie** et intègre l'utilisateur dans le processus d'émission et de règlement des transactions. Comparées aux monnaies ordinaires maintenues par des institutions financières, les cryptomonnaies sont gérées par un grand livre de comptes ouvert et consultable de tous (la blockchain) qui répertorie l'ensemble des transactions depuis l'origine. De plus en plus de sites webs, magasins et restaurants se mettent à accepter le paiement en cryptomonnaie. Toutefois, bien qu'en croissance, le réseau de paiement des cryptomonnaies est peu développé. Cependant, il y a beaucoup de préoccupations au sujet de la sécurisation contre les hackers et de l'irréversibilité des transactions.

L'objectif principal de ce document est de fournir un bref aperçu sur les concepts de la cryptomonnaie, précisément de l'Ethereum et des précisions sur la conception, l'implémentation et l'installation de l'application mobile Android « **KryptNation** ».

La partie principale de ce rapport se concentre sur les processus, les choix fait lors de la conception et l'implémentation de l'application Android. Le thème principal était la création d'une plateforme informative permettant d'afficher la balance d'un portefeuille Ethereum en termes de nombres tokens, mais aussi la liste des transactions réalisées par ce portefeuille.

Les conclusions tirées de ce rapport de projet sont purement techniques aux choix des différents outils permettant l'implémentation des fonctionnalités telles que décrites plus haut de l'application mobile « **KryptNation** » sur la plateforme Android.

# TABLE DES MATIERES

RESUME.....	2
INTRODUCTION .....	5
A. PLATE-FORME ANDROID.....	8
1. QU'EST-CE QU'ANDROID?.....	8
2. CARACTERISTIQUES .....	8
3. DIFFERENTES VERSIONS D'ANDROID .....	9
4. RENTABILITE .....	10
B. FONCTIONNALITES DE KRYPTNATION.....	11
1. LECTEUR DE QR CODE.....	11
i. <i>Qu'est-ce qu'un QR Code?</i> .....	11
ii. <i>Pourquoi implementer un lecteur de QR Code?</i> .....	12
2. ETHERSCAN .....	12
i. <i>Qu'est-ce qu'ETHERSCAN?</i> .....	12
ii. <i>Comment fonctionne Etherscan?</i> .....	12
C. PROTOTYPE PAPIER .....	15
1. DEFINITION D'UN PROTOTYPE PAPIER.....	15
2. PROTOTYPE PAPIER DE « KRYPTNATION » .....	15
D. NOS CHOIX .....	18
1. LANGAGE DE PROGRAMMATION .....	18
i. <i>Qu'est – ce que Kotlin?</i> .....	18
ii. <i>Pourquoi avoir choisi Kotlin?</i> .....	18
iii. <i>Android Studio et Kotlin</i> .....	19
2. VIEWMODEL.....	21
i. <i>Qu'est ce- qu'un ViewModel?</i> .....	21
ii. <i>Integration du viewModel dans KryptNation</i> .....	22
3. REQUETE HTTP (ANDROID) – LIBRAIRIE RETROFIT .....	22

<i>i.</i>	<i>Qu'est-ce que Retrofit?.....</i>	22
<i>ii.</i>	<i>Integration de Retrofit dans KryptNation .....</i>	23
4.	LECTEUR DE QR CODE – LIBRAIRIE ZXING .....	25
<i>i.</i>	<i>Qu'est-ce que Zxing?.....</i>	25
<i>ii.</i>	<i>Implementation de Zxing dans « KRYPTNATION » .....</i>	25
5.	ANDROID ARCHITECTURE ROOM .....	27
<i>i.</i>	<i>Qu'est-ce qu'Android Architecture Room?.....</i>	27
<i>ii.</i>	<i>Implementation du Room dans « KryptNation ».....</i>	28
6.	FRAGMENT VIEW .....	29
<i>i.</i>	<i>Qu'est-ce qu'un Fragment view?.....</i>	29
<i>ii.</i>	<i>Cycle de vie d'un Fragment.....</i>	29
<i>iii.</i>	<i>Fragment View Dans « KryptNation » .....</i>	30
7.	ETHERSCAN API .....	31
E.	DIFFICULTES RENCONTREES.....	32
F.	VERSION FINALE .....	33
G.	SI C'ETAIT A REFAIRE .....	36
H.	CONCLUSION .....	36
ANNEXE – MANUEL D’UTILISATION .....		37
1.	INSTALLATION DE L’APPLICATION MOBILE .....	37
<i>Prerequis :.....</i>		37
<i>Etape 1 : Télécharger Android Studio .....</i>		37
<i>Etape 2 : Ouvrir le projet avec Android Studio .....</i>		38
<i>Etape 3 : Rebuild le Projet avec Android Studio .....</i>		39
<i>Etape 4 : Brancher votre téléphone .....</i>		39
<i>Etape 5 : Installation de l’application .....</i>		40
2.	UTILISATION DE L’APPLICATION .....	41

# INTRODUCTION

La blockchain ou chaîne de blocs est une technologie de stockage et de transmission d'informations, transparente sécurisée, et fonctionnant sans organe central de contrôle. Par extension, une blockchain constitue une base de données qui contient l'historique de tous les échanges effectués entre ses utilisateurs depuis sa création. Cette base de données est sécurisée et distribuée (elle est partagée par ses différents utilisateurs, sans intermédiaire, ce qui permet à chacun de vérifier la validité de la chaîne). Il existe des blockchains publiques, ouvertes à tous, et des blockchains privées, dont l'accès et l'utilisation sont limitées à un certain nombre d'acteurs.

Une blockchain publique peut donc être assimilée à un grand livre comptable public, anonyme et infalsifiable. Comme l'écrit le mathématicien **Jean-Paul Delahaye**, il faut s'imaginer

**« Un très grand cahier, que tout le monde peut lire librement et gratuitement, sur lequel tout le monde peut écrire, mais qui est impossible à effacer et indestructible »<sup>1</sup>**

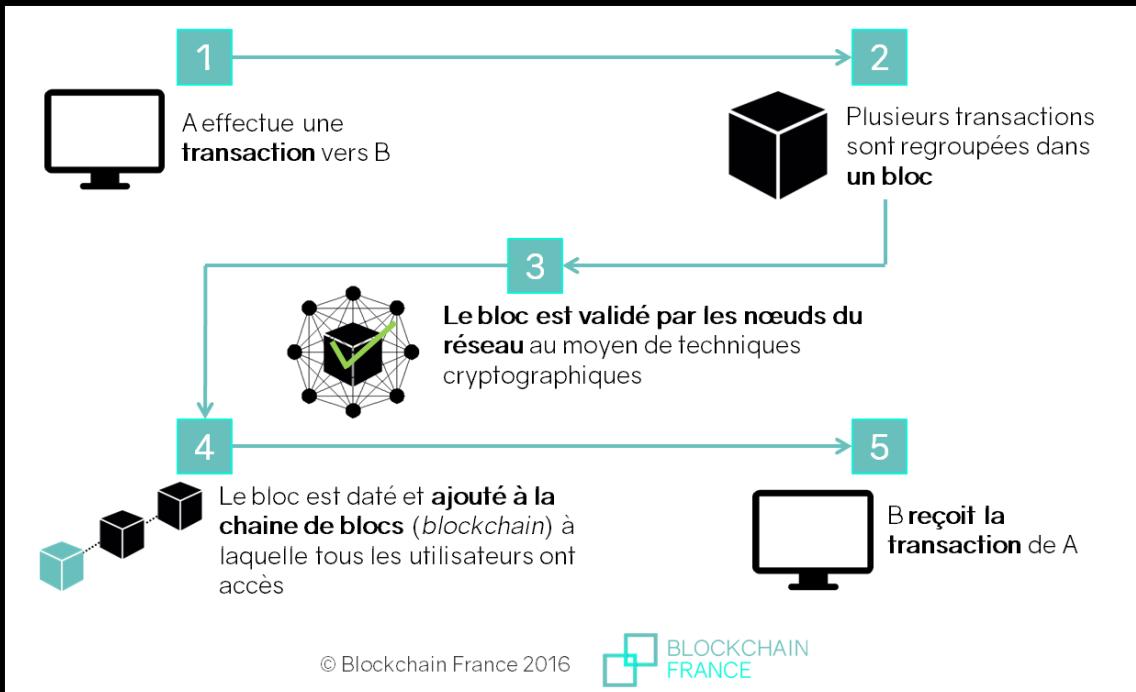
La première blockchain est apparue en 2008 avec la monnaie numérique bitcoin, développée par un inconnu se présentant sous le pseudonyme **Satoshi Nakamoto**. Aujourd'hui, de nombreux acteurs (entreprises, gouvernements etc...) envisagent l'utilisation de la technologie blockchain pour d'autres cas que la monnaie numérique.

Toute blockchain publique fonctionne nécessairement avec une monnaie ou un token (jeton) programmable. Les transactions effectuées entre les utilisateurs sont regroupées par blocs. Chaque bloc est validé par les nœuds du réseau appelés « **les mineurs** », selon des techniques qui dépendent du type de blockchain. Une fois le bloc validé, il est horodaté et ajouté à la chaîne de blocs. La transaction est alors visible par le récepteur ainsi que l'ensemble du réseau (Voir Figure 1). Ce processus prend un certain temps selon la blockchain (environ une dizaine de minutes pour **Bitcoin**, et 15 secondes pour **Ethereum**).

Le caractère décentralisé de la blockchain, couplé avec sa sécurité et sa transparence, promet des applications bien plus larges que le domaine monétaire. On peut classer l'utilisation de la blockchain en trois catégories :

---

<sup>1</sup> <https://blockchainfrance.net/découvrir-la-blockchain/c-est-quoi-la-blockchain/>



**Figure 1**

- Les applications pour le transfert d'actifs (utilisation monétaire, mais pas uniquement : titres, votes, actions, obligations...).
- Les applications de la blockchain en tant que registre : elle assure ainsi une meilleure traçabilité des produits et des actifs.
- Les smarts contracts : il s'agit de programmes autonomes qui exécutent automatiquement les conditions et termes d'un contrat, sans nécessiter d'intervention humaine une fois démarré.

Alors qu'un contrat standard décrit les termes d'une relation (exécutoire par la loi), un contrat intelligent applique une relation avec le code cryptographique. En d'autres mots, les « smarts contrats » sont des programmes qui s'exécutent exactement comme ils ont été configurés par leurs créateurs.<sup>2</sup> D'abord conçu en 1993, l'idée a été initialement décrite par l'informaticien et cryptographe Nick Szabo comme une sorte de distributeur automatique numérique.

<sup>2</sup> <https://courscryptomonnaies.com/ethereum#infos>

Dans un exemple simple, les utilisateurs d’Ethereum peuvent envoyer 5 Éther à un ami à une certaine date en utilisant un « smart contrat ». L’utilisateur crée un « smart contrat » et envoie les données à ce contrat afin qu’il puisse exécuter la commande souhaitée. L’Ethereum est une plateforme spécialement conçue pour exécuter des « smarts contrats ».

Ethereum permet aux développeurs de programmer leurs propres contrats intelligents, ou « agents autonomes » comme les appelle le livre blanc d’Ethereum (description originale du projet Ethereum de **Vitalik Buterin** avant d’entamer le développement du projet). Le langage est « **Turing-complet** », ce qui signifie qu’il prend en charge un ensemble plus large d’instructions de calcul.

D’où la création de **KryptNation** qui est une application Android qui permet d’afficher la balance d’un portefeuille (wallet) Ethereum en termes de nombre de tokens, mais aussi la liste des transactions effectuées dans le portefeuille. Dans le but de faciliter aux utilisateurs de la cryptomonnaie plus précisément de l’Ethereum la gestion de leur portefeuille (wallet).

# A. PLATE-FORME ANDROID

## 1. QU'EST-CE QU'ANDROID?

Android est un système d'exploitation mobile basé sur le noyau Linux et développé actuellement par Google. Lancé en juin 2007, à la suite du rachat par Google, le système avait d'abord été conçu pour les smartphones et tablettes tactiles, puis s'est diversifié dans les objets connectés et ordinateurs comme les télévisons (Android TV), les voitures (Android Auto), les ordinateurs (Android-x86) et les smartwatch (Android Wear).



Android est tout d'abord un projet Open Source du nom AOSP (Android Open Source Project) qui est ensuite utilisé par Google pour développer Android, et réutilisé par les autres constructeurs tels que **Samsung**, **LG** et autres avec notamment les services Google.

## 2. CARACTERISTIQUES

Android est défini comme étant une pile de logiciels, c'est-à-dire un ensemble de logiciels destinés à fournir une solution clé en main pour les appareils mobiles (smartphones et tablettes tactiles). Cette pile comporte un système d'exploitation (comprenant le noyau **Linux**), les applications clés telles que le navigateur Web, le carnet d'adresses ainsi que les logiciels intermédiaires entre le système d'exploitation et les applications. L'ensemble est organisé en 5 couches distinctes :

- Le noyau **Linux** avec ses pilotes.
- Des bibliothèques logicielles telles que **WebKit/Blink**, **OpenGL ES**, **SQLite**, ou **FreeType**.
- Un environnement d'exécution et des bibliothèques permettant d'exécuter des programmes prévus pour la plate-forme Java.
- Un Framework – kit de développement d'application.

- Un lot d'applications standard qui comprend un environnement bureau, un carnet d'adresses, un navigateur Web et un téléphone.

Le noyau **Linux** utilisé pour les fondations d'Android, fournit les services classiques des systèmes d'exploitation tels que l'utilisation des périphériques, accès aux réseaux de télécommunication, manipulation de la mémoire des processus. Certains outils tels que le **X Window System**, **GNU** ainsi que certain fichier de configuration qui se trouvent d'ordinaire dans les distributions **Linux** ne sont pas inclus dans Android.

### **3. DIFFERENTES VERSIONS D'ANDROID**

Les différentes versions d'Android ont toutes des noms de dessert ou plus généralement de sucreries (souvent seulement en anglais)

VERSION	NOM DE CODE	VERSION API	DISTRIBUTION
2.3.3 – 2.3.7	Gingerbread	10	0,2%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	0,3%
4.1.x, 4.2.x et 4.3	Jelly Bean	16, 17, 18	3,6%
4.4	KitKat	19	9,1%
5.0 et 5.1	Lollipop	21, 22	20,4%
6.0	Marshmallow	23	23,5%
7.0 et 7.1	Nougat	24, 25	30,8%
8.0 et 8.1	Oreo	26, 27	12,1%

**Figure 2**

Nous pouvons remarquer que les versions les plus âgées tendent naturellement à disparaître au profit des nouvelles et c'est tant mieux, ne serait-ce que pour des raisons de sécurité informatique mais aussi parce que les nouvelles moutures sont dotées de fonctions plus abouties et des nouvelles options. (Voir Figure 2 et Figure 3)

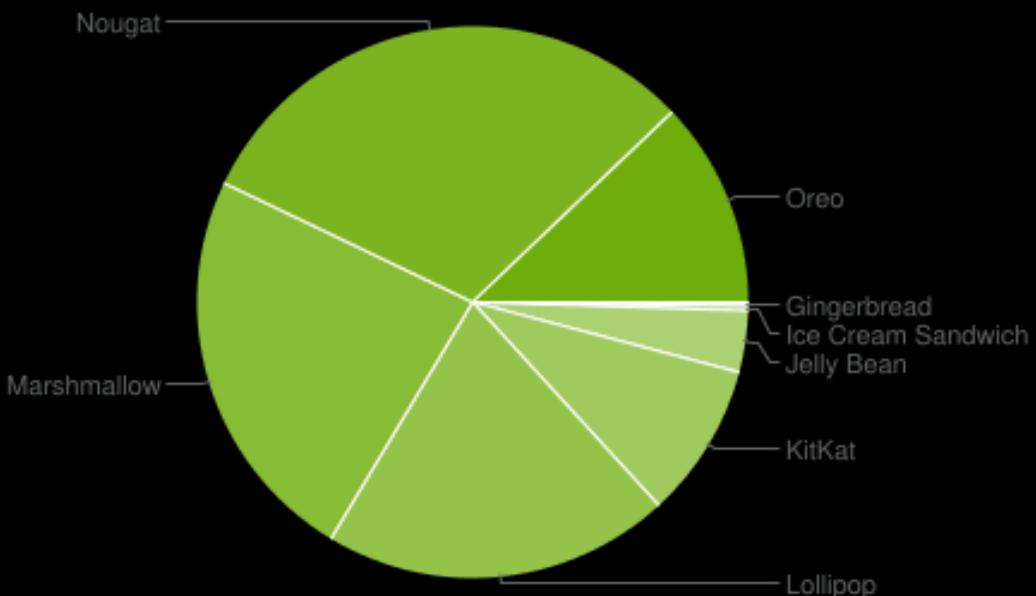


Figure 3

#### 4. RENTABILITE

Pour les développeurs d'applications, la rentabilité qu'offre Android est moins nette, surtout si on la compare avec l'offre de son concurrent **l'App Store** sur iOS. Si la plateforme Android a aujourd'hui plus de succès, si le nombre d'appareils est plus important, il est cependant plus difficile de transformer ce potentiel en ventes. Il semble que ce soit parce que les utilisateurs d'Android sont plus récents alors que les utilisateurs d'iOS sont une base établie, que le marché Android est plus libre alors que chaque compte iOS est directement associé à un moyen de paiement, alors que le système Android a une plus grande diversité qu'iOS.<sup>3</sup>

<sup>3</sup> <http://www.iphon.fr/post/applications-les-developpeurs-preferent-app-store-android-market>

## B. FONCTIONNALITES DE KRYPTNATION

L'application mobile Android « KryptNation » implémente plusieurs fonctionnalités telles que :

- Un lecteur de QR Code pour scanner un code QR qui représente une adresse de portefeuille Ethereum afin d'accéder aux informations relatives au portefeuille.
- Implémentation d'ETHERSCAN API ce qui permet de recevoir toutes les informations concernant un portefeuille via son adresse.

### 1. LECTEUR DE QR CODE

#### I. QU'EST-CE QU'UN QR CODE?

Un QR Code est un code-barres à deux dimensions qui permet d'encoder des données. Il s'agit le plus généralement d'un lien vers une page Internet (URL), mais dans notre cas d'une adresse portefeuille Ethereum. En scannant le QR Code avec un smartphone il est possible, par exemple, d'accéder rapidement à une page Internet. D'autres actions sont également possibles comme enregistrer les informations d'une carte de visite directement dans sa liste de contacts ou encore ajouter un évènement à son agenda.

D'un point de vue fonctionnel, le QR Code permet d'étendre l'information présente sur tout support physique et de donner une dimension numérique à cette information.



Figure 4 – Exemple de QR CODE

## **II. POURQUOI IMPLEMENTER UN LECTEUR DE QR CODE?**

Capable de contenir une quantité de données bien supérieure (jusqu'à 7089 caractères), il s'utilise en étant scanné via l'appareil photo/caméra d'un smartphone ou d'une tablette. Ensuite, il est simplement analysé grâce à une application dédiée (dans notre cas Etherscan API). Vu le nombre de caractères que possède une adresse portefeuille Ethereum, il est plus convenable de générer un QR Code soit directement à la création du portefeuille ou par le biais de certaines plateforme web (par exemple: <https://bitcoinqrcodegenator.win/ethereum.html>).

## **2. ETHERSCAN**

### **I. QU'EST-CE QU'ETHERSCAN?**

Etherscan est ce qu'on appelle un **BlockExplorer** pour Ethereum. Un BlockExplorer est un moteur de recherche qui permet à ses utilisateurs de chercher facilement des transactions qui transitent sur la blockchain Ethereum.

La blockchain Ethereum dispose d'un grand livre public (comme une base de données décentralisée, regroupant toutes les transactions effectuées sur la blockchain Ethereum) qu'Etherscan.io indexe et met ensuite à disposition de tous. Leur mission est de « **Faciliter la transparence de la Blockchain en indexant et en rendant toutes les transactions consultables sur Ethereum Blockchain de la manière la plus transparente et facile possible** ».

### **II. COMMENT FONCTIONNE ETHERSCAN?**

Pour consulter un portefeuille, vous devez indiquer la clé publique (l'adresse correspondante) dans la barre de recherche située en haut à droite. (Voir Figure 5<sup>4</sup>)

---

<sup>4</sup> <https://cryptoast.fr/tutoriel-etherscan/>

Etherscan  
The Ethereum Block Explorer

1 LOGIN Search by Address / Txhash / Block / Token / Ens GO

2 HOME BLOCKCHAIN ACCOUNT TOKEN CHART MISC

Address 0x46a99925d914d746168e9016b030d84d0077a6ad 3

Home / Normal Accounts / Address

Sponsored Link: Searching for Mass-Adopting ICO for Real-World Problems? Look No Further - [Join BIRDCHAIN ICO Now!](#)

**Overview** 4

ETH Balance: 0.003403495 Ether

ETH USD Value: \$1.91 (@ \$560.00/ETH)

No Of Transactions: 6 txns

Misc 5

Address Watch Add To Watch List

Token Balances View (\$0.00) 1

Transactions 6 Latest 6 txns

TxHash	Block	Age	From	To	Value	[TxFee]
0xbd5ea4576034cfb...	5286101	1 day 20 hrs ago	0x46a99925d914d74...	OUT DeveryToken	0 Ether	0.000199645
0x596c8be652a598c...	5286061	1 day 20 hrs ago	0x46a99925d914d74...	OUT INDEX_1	0.292 Ether	0.00020835
0x974cfe0e1f35918e...	5286047	1 day 20 hrs ago	0x65e2c5175e2e618...	IN 0x46a99925d914d74...	0.29210599 Ether	0.000063
0xe8a52b7e808ae67...	5272236	4 days 4 hrs ago	0x46a99925d914d74...	OUT INDEX_1	0.19567068 Ether	0.0012945
0xfd6da6fac6430474...	5272191	4 days 4 hrs ago	0xfb5e05ee2f03fcef...	IN 0x46a99925d914d74...	0.17968238 Ether	0.000063
0xb037d2888cf64f2e...	5271905	4 days 5 hrs ago	0x2e41d73b3bd87f1...	IN 0x46a99925d914d74...	0.0209883 Ether	0.000084

[ Download CSV Export ]

Figure 5

Voici en détails la correspondance de chaque partie de la **Figure 5** :

- **1/** La barre de recherche où vous devez indiquer une adresse publique de portefeuille Ethereum, un Hash de transaction (numéro de transaction), un block, un Token ...
- **2/** Le menu Etherscan qui vous permet de revenir de revenir sur la page d'accueil, d'accéder aux transactions, blocs, données de smart contrats (blockchain). On peut également accéder à tous les contrats et les adresses Ethereum. La partie **Token** permet de visualiser des tokens et leurs transferts. La partie **Chart** fournit des statistiques intéressantes telles que les frais de transactions, le nombre de transactions etc... **Misc** permet lui d'accéder à de nombreuses fonctionnalités comme des APIs (que j'ai utilisé pour mon projet), des calculateurs de minage etc...
- **3/** Cela correspond à l'adresse que vous êtes actuellement en train de consulter.
- **4/ Overview** est un résumé de l'adresse, le nombre d'Éthers contenu (**ETH Balance**), leur valeur en dollars (**ETH USD Value**) et le nombre de transactions (**No Of Transaction**).
- **5/ Misc** vous permet d'ajouter cette adresse à votre liste (vous devez disposer d'un compte Etherscan), ce qui permet de regrouper toutes ses adresses sur un même compte. Il permet également de voir la valeur des tokens que vous stockez sur votre adresse lorsque vous cliquez dessus (**Token Balances**).
- **6/** Cette partie corresponds au tableau regroupant les dernières transactions Ethereum (**Transactions**), les transferts de tokens (**Token Transfers**) et les commentaires (**Comments**).

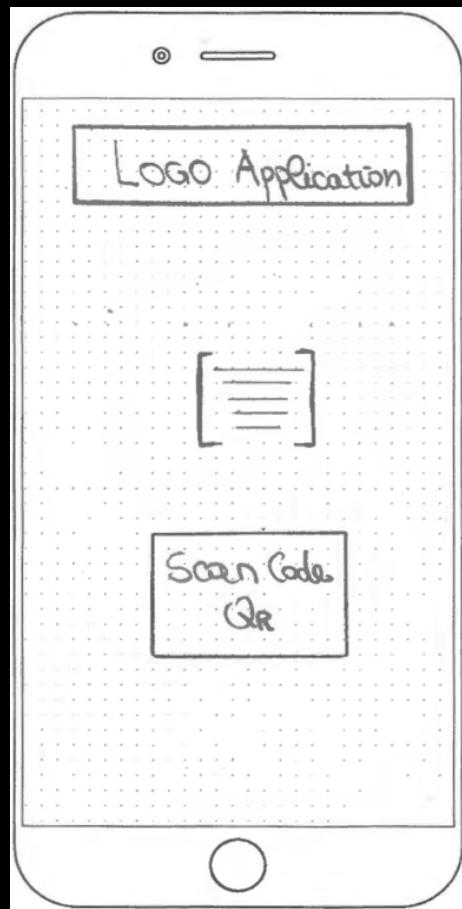
## C. PROTOTYPE PAPIER

### 1. DEFINITION D'UN PROTOTYPE PAPIER

En interface homme-machine et en ergonomie informatique, le prototype papier est une technique encore très utilisée dans la conception centrée sur l'utilisateur. Cette technique aide les parties prenantes d'un projet (concepteurs, développeurs etc..) à concevoir un logiciel (dans notre cas « **KryptNation** »). Il s'agit le plus souvent des idées brutes jetées sur le papier, faites d'esquisses et de dessins, représentant l'agencement et les zones de contenus (ou zoning).

### 2. PROTOTYPE PAPIER DE « KRYPTNATION »

Le prototypage papier est probablement la technique la moins coûteuse et la plus rapide pour réaliser un prototype. Elle nous a été utile mon superviseur et moi afin d'identifier un bon nombre de problèmes d'utilisabilité et le rapprocher des attentes et besoins des utilisateurs. Nous nous sommes finalement entendus sur le prototype papier suivant :



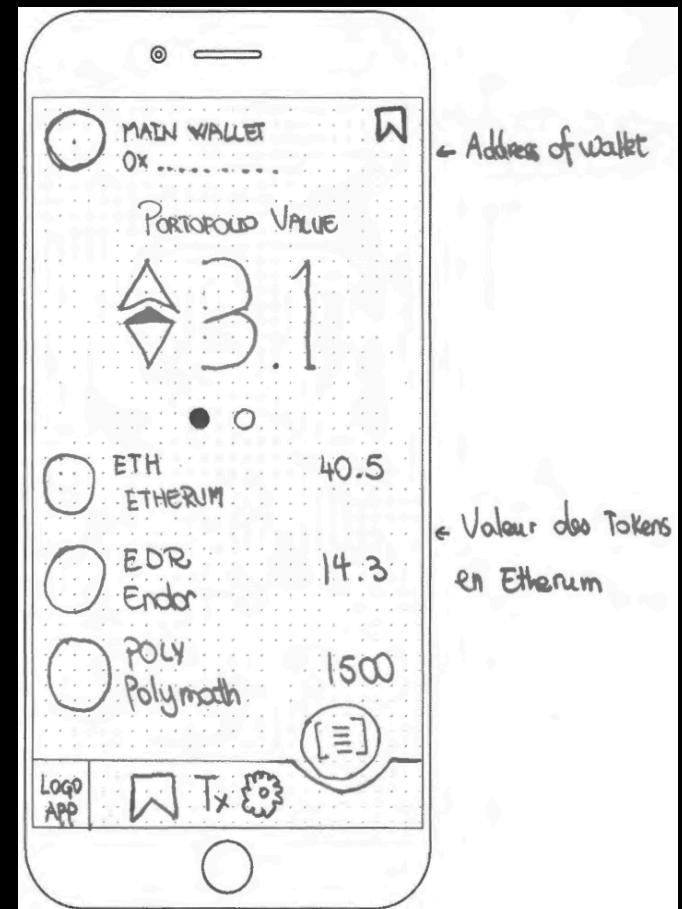
**activity\_scan**

Layout permettant de scanner le code QR de son portefeuille adresse et qui implémente le lecteur de code QR.



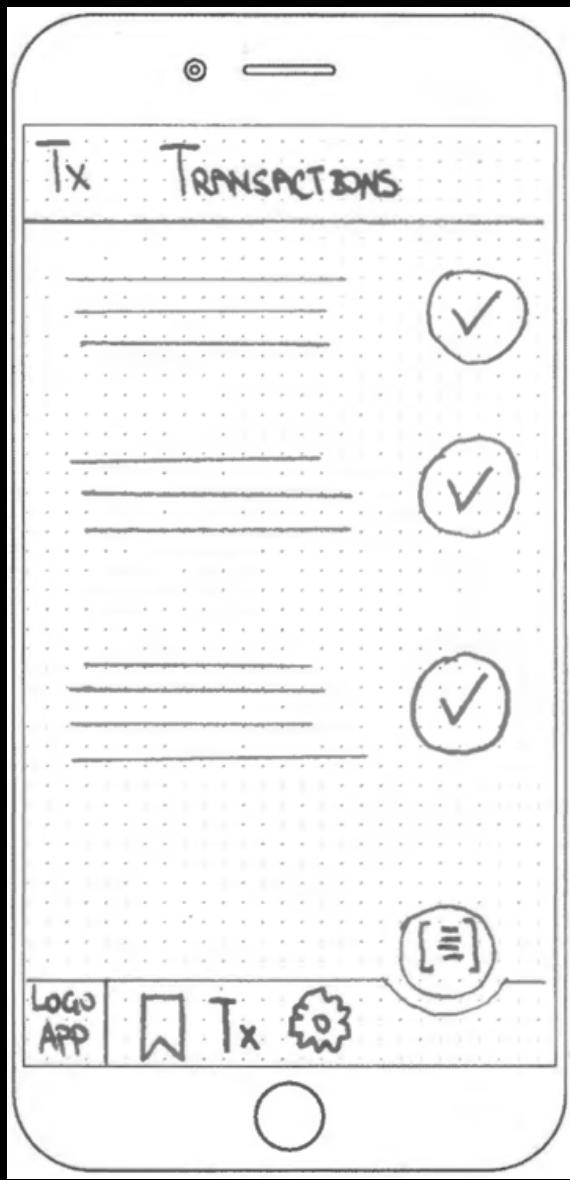
**activity\_edit**

Renommer les adresses du portefeuille Ethereum qui sont par défaut pas très facile à retenir.



**activity\_main**

Après avoir scanner le code QR du portefeuille le Layout principal apparait avec toutes les informations (valeurs des tokens etc...)



### fragment\_transactions && fragment\_bookmarks

- Fragment affichant toutes les transactions du portefeuille courante.
- Fragment affichant tous les portefeuilles mis en favoris sur ce téléphone.

## D. NOS CHOIX

Pendant l'implémentation de l'application « KryptNation » nous avons fait plusieurs choix importants qui nous ont aidé à mener à bien le projet.

### 1. **LANGUAGE DE PROGRAMMATION**

#### I. QU'EST – CE QUE KOTLIN?

Kotlin est un langage de programmation orienté objet et fonctionnel, avec un typage statique qui permet de compiler la machine virtuelle Java et JavaScript. Son développement provient principalement d'une équipe de programmeurs chez **JetBrains** basée à Saint -Pétersbourg en Russie (son nom vient de l'île de Kotline près de Saint-Pétersboug).<sup>5</sup>

D'après Google, Kotlin est un langage « **expressif, concis, extensible, puissant et agréable à lire et à écrire** », et « **il a des fonctionnalités de sécurité intéressantes en termes de nullabilité et d'immutabilité** ».<sup>6</sup>

#### II. POURQUOI AVOIR CHOISI KOTLIN?

Pour la réalisation de ce projet j'ai dû faire plusieurs choix, on devait choisir un langage de programmation parmi plusieurs (par exemple Kotlin, Java et C++). Notre choix était d'utiliser le langage de programmation **Kotlin** sur Android Studio pour le développement de notre application mobile « KryptNation ».

Kotlin est un excellent langage de programmation permettant de développer des applications Android, apportant tous les avantages d'un langage moderne à la plateforme Android sans introduire de nouvelles restrictions :

---

<sup>5</sup> [https://fr.wikipedia.org/wiki/Kotlin\\_\(langage\)](https://fr.wikipedia.org/wiki/Kotlin_(langage))

<sup>6</sup> <https://blog.talanlabs.com/developpement-android-kotlin-java/>

- **Compatibilité :** Kotlin est entièrement compatible avec JDK 6, ce qui garantit que les applications développées sous Kotlin peuvent fonctionner sur d'anciens appareils Android sans aucun problème. L'outillage Kotlin est entièrement pris en charge dans Android Studio et compatible avec le système de construction Android.
- **Performance :** Une application Android développée avec Kotlin s'exécute aussi vite qu'une application équivalente développée avec Java, grâce à une structure de bytecode très similaire.<sup>7</sup>
- **Temps de compilation :** Kotlin prend en charge la compilation incrémentielle efficace, donc, même s'il y a des frais supplémentaires pour les versions propres, les versions incrémentielles sont généralement aussi rapides ou plus rapides qu'avec Java.<sup>8</sup>

### III. ANDROID STUDIO ET KOTLIN

Pour créer votre première application Kotlin sous Android Studio vous devez :

- Téléchargez une installation Android Studio, qui inclut le support Kotlin prêt à l'emploi via le site web <https://developer.android.com/studio/>.
- Suivez le didacticiel « **Getting Started with Android et Kotlin** » pour créer votre première application Kotlin (Voir Figure 6).
- Pour une introduction plus approfondie, consulter la documentation de référence de Kotlin sur le site web <https://kotlinlang.org/docs/reference/kotlin-doc.html>.

---

<sup>7</sup> <https://code-examples.net/fr/docs/kotlin/docs/reference/android-overview>

<sup>8</sup> <https://code-examples.net/fr/docs/kotlin/docs/reference/android-overview>

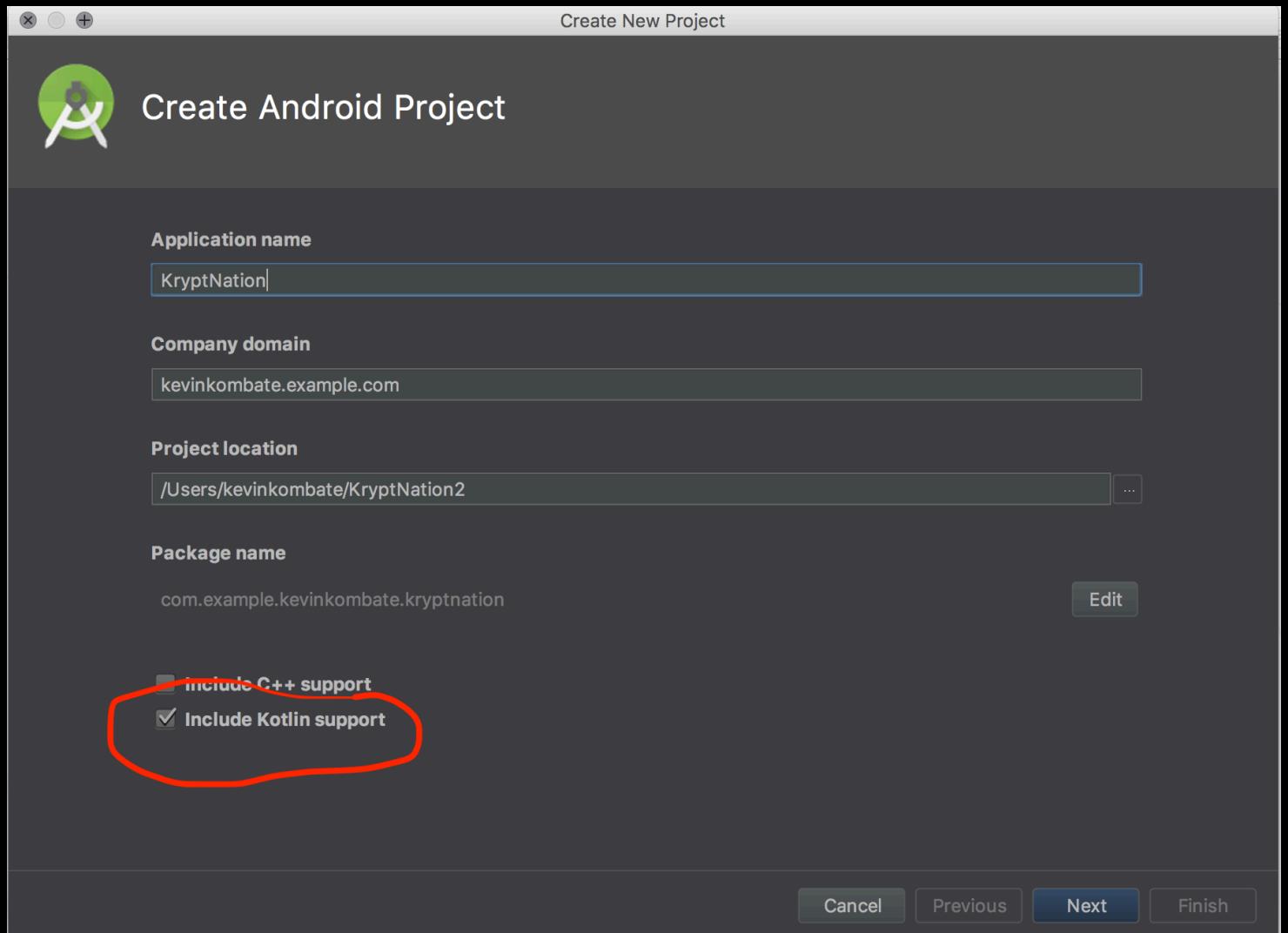


Figure 6

## 2. VIEWMODEL

### I. QU'EST CE- QU'UN VIEWMODEL?

Un controller UI (User Interface) tel qu'un **Fragment** ou une **Activité** peut être détruit ou recréée par le framework Android. Cela se fait en réponse à des évènements provoqués ou non par l'utilisateur. L'exemple le plus simple auquel on peut penser est le changement d'orientation. Si on ne le prévoit pas, on perd toutes données associées. On doit donc :

- Soit systématiquement récupérer les données;
- Soit utiliser les méthodes offertes par Android pour sauvegarder des données telles que « **onSaveInstanceState ()** »

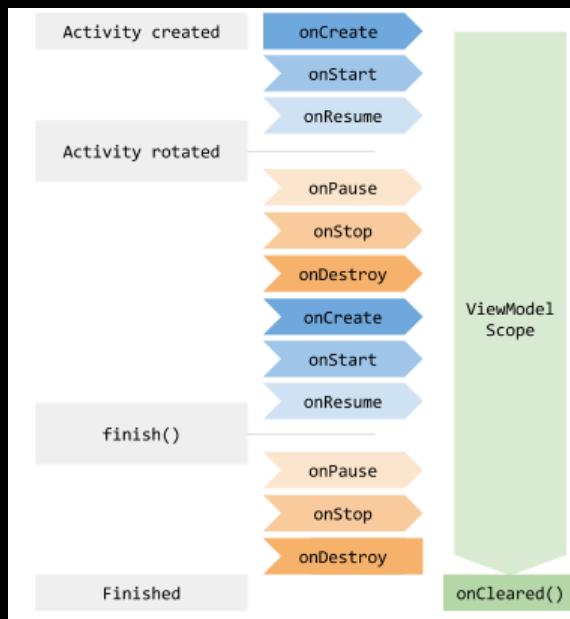


Figure 6 – Cycle de vie d'un ViewModel

Le ViewModel est « lifecycle-aware ». Si son « `lifecycleOwner` » est une activité, il ne sera détruit qu'au moment où cette dernière sera finie et détruite. De plus, son exécution étant décorrélée de l'activité, on peut instancier un seul même ViewModel, partagé par plusieurs UI Components.<sup>9</sup>

<sup>9</sup> <http://blog.ippon.fr/2018/05/31/introduction-aux-android-architecture-components/>

## **II. INTEGRATION DU VIEWMODEL DANS KRYPTNATION**

Certaines méthodes de mon projet tels que « **onCreate** » et « **onStart** » utilisent le ViewModel pour stocker et gérer les données liées à l'interface utilisateur d'une manière consciente du cycle de vie. (Voir Figure 7).

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_edit)
    supportActionBar?.hide() // Pour cacher la toolbar
    ethereumAddress = intent.getStringExtra( name: "ETHEREUM_ADDRESS")
   是从MainActivity = intent.getBooleanExtra( name: "FROM_MAIN_ACTIVITY", defaultValue: false)
    wallet_address.text = ethereumAddress
    addToBookmarkCheckbox.setOnCheckedChangeListener { buttonView, isChecked ->
        if(isChecked) {
            renameEditTextInputLayout.visibility = VISIBLE
        } else {
            renameEditTextInputLayout.visibility = GONE
        }
    }
}
```

**Figure 7 – Code du projet « KryptNation »**

## **3. REQUETE HTTP (ANDROID) – LIBRAIRIE RETROFIT**

### **I. QU'EST-CE QUE RETROFIT?**

Une application Android a souvent besoin d'une source de donnée externe, par exemple un serveur avec lequel échanger des données. Le serveur met à disposition une liste d'URL par lesquelles les applications vont pouvoir accéder aux données. Le format de données échangées se fait le plus souvent en **JSON** ou en **XML**. On dit alors que le serveur fournit un **webservice**, dans notre cas une **API REST**.

Retrofit est donc une librairie permettant de réaliser des appels webservices REST de façon très simple.

## II. INTEGRATION DE RETROFIT DANS KRYPTNATION

Comme toute librairie, il était nécessaire de l'importer dans notre projet en utilisant Gradle. Pour cela, il a suffi d'ajouter les lignes suivantes aux dépendances : (Voir Figure 9 et 10)



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:logging-interceptor:3.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
    implementation 'com.google.zxing:core:3.3.0'
    implementation 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
    implementation 'com.android.support:design:27.1.1'
    implementation 'com.android.support:support-vector-drawable:27.1.1'
    implementation 'com.android.support:recyclerview-v7:27.1.1'
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:0.22.5"
    implementation "io.reactivex.rxjava2:rxjava:2.0.0"
    implementation 'com.afollestad.material-dialogs:core:0.9.6.0'

    implementation 'android.arch.persistence.room:runtime:1.1.1'
    kapt 'android.arch.persistence.room:compiler:1.1.1'

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Figure 8 – Importation de la librairie Retrofit

```

private val retrofit: Retrofit by lazy {
    val httpClient = OkHttpClient.Builder()
    if (BuildConfig.DEBUG) {
        val logging = HttpLoggingInterceptor()
        logging.level = HttpLoggingInterceptor.Level.BODY
        httpClient.addInterceptor(logging)
    }
    ^lazy Retrofit.Builder()
        .baseUrl(baseUrl: "https://api.etherscan.io/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(httpClient.build())
        .build()
}

```

Figure 9

```

package com.example.kevinkombate.kryptnation.api

import com.example.kevinkombate.kryptnation.model.Balance
import com.example.kevinkombate.kryptnation.model.Result
import com.example.kevinkombate.kryptnation.model.Transaction
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Query
💡
interface Etherscan ApiService {

    @GET("value: "api")
    fun singleAccountBalance(@Query("value: "apikey") apiKey: String,
                            @Query("value: "module") module: String,
                            @Query("value: "action") action: String,
                            @Query("value: "tag") tag: String,
                            @Query("value: "address") address: String): Call<Result<String>>

    @GET("value: "api")
    fun transactions(@Query("value: "apikey") apiKey: String,
                    @Query("value: "module") module: String,
                    @Query("value: "action") action: String,
                    @Query("value: "tag") tag: String,
                    @Query("value: "address") address: String,
                    @Query("value: "page") page: Int,
                    @Query("value: "offset") offset: Int,
                    @Query("value: "sort") sort: String): Call<Result<List<Transaction>>>
}

```

Figure 10

## **4. LECTEUR DE QR CODE – LIBRAIRIE ZXING**

### **I. QU'EST-CE QUE ZXING?**

Zxing est une librairie de traitement d'images à code-barres implémentée en Java (aussi utilisable avec Kotlin dans Android Studio), avec des ports vers d'autres langages. Cette librairie met l'accent sur l'utilisation de la caméra intégrée sur les téléphones mobiles et de décoder les codes-barres sur l'appareil, sans communiquer avec un serveur.

Les formats pouvant être décodés sont :

- UPC-A et UPC-E
- EAN-8 et EAN-13
- Code 39
- Code 128
- QR Code
- Data Matrix
- PDF 417
- ITF

### **II. IMPLEMENTATION DE ZXING DANS « KRYPTNATION »**

Pour l'implémentation du lecteur de QR Code j'ai utilisé la librairie Zxing, ce qui est une excellente librairie pour le scanner et génération de QR Code. Pour cela, il a suffi d'ajouter les lignes suivantes aux dépendances : (voir Figure 11)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:logging-interceptor:3.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
    implementation 'com.google.zxing:core:3.3.0'
    implementation 'com.journeyapps:zxing-android-embedded:3.2.0@aar' ←
    implementation 'com.android.support:design:27.1.1'
    implementation 'com.android.support:support-vector-drawable:27.1.1'
    implementation 'com.android.support:recyclerview-v7:27.1.1'
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:0.22.5"
    implementation "io.reactivex.rxjava2:rxjava:2.0.0"
    implementation 'com.afollestad.material-dialogs:core:0.9.6.0'

    implementation 'android.arch.persistence.room:runtime:1.1.1'
    kapt 'android.arch.persistence.room:compiler:1.1.1'

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Figure 11

```
override fun onClick(v: View?) {
    val integrator = IntentIntegrator(activity)
    integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES)
    integrator.setPrompt("Scan your Ethereum wallet address")
    integrator.setCameraId(0)
    integrator.setBeepEnabled(false)
    integrator.setBarcodeImageEnabled(false)
    integrator.initiateScan()
}
```

Figure 12 – Implémentation du lecteur de QR Code

## 5. ANDROID ARCHITECTURE ROOM

### I. QU'EST-CE QU'ANDROID ARCHITECTURE ROOM?

La bibliothèque de persistance Room fournit une couche d'abstraction sur SQLite pour permettre un accès à une base de données plus robuste tout en exploitant la puissance de SQLite. Room est une librairie décomposée en trois parties<sup>10</sup>: (voir Figure 13)

- Les **Entités** qui correspondent à des tables en base de données.
- Les **DAOs** qui contiennent les méthodes permettant d'interagir avec la base de données.
- La **Base de données** où on déclare nos DAOs, nos Entités et la version de la base.

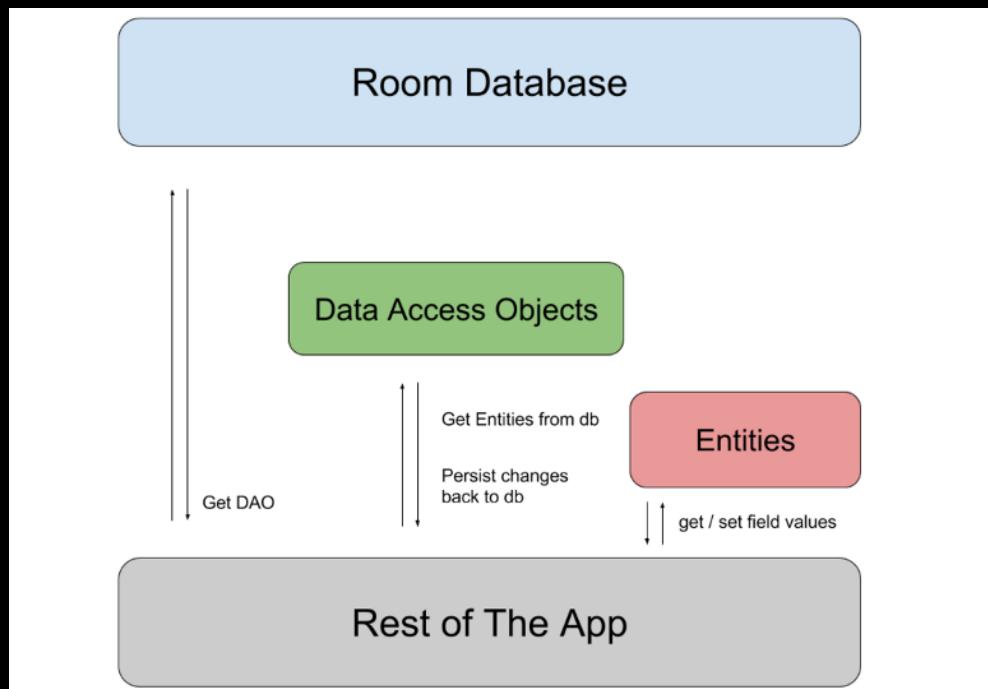


Figure 13

La bibliothèque aide à créer un cache des données de l'application sur un appareil qui l'exécute votre application. Ce cache, qui constitue l'unique source de vérité de l'application, permet aux utilisateurs d'afficher une copie cohérente des informations clés au sein de votre application,

<sup>10</sup> <http://blog.ippon.fr/2018/05/31/introduction-aux-android-architecture-components/>

que les utilisateurs disposent ou non d'une connexion Internet. D'où la nécessité pour moi de son implémentation dans mon projet pour mettre en favoris les différents wallets

## II. IMPLEMENTATION DU ROOM DANS « KRYPTNATION »

L'implémentation du Room dans l'application mobile « KryptNation » a été fait dans le paquage « Database » (Voir les Figure 14 et 15 pour l'implémentation)

```
package com.example.kevinkombate.kryptnation.database

import ...

@Database(entities = [Wallet::class], version = 1)
abstract class AppDatabase: RoomDatabase() {
    abstract fun walletDao()
}
```

Figure 14

```
package com.example.kevinkombate.kryptnation.database

import ...

@Dao
interface WalletDao {

    @Query( value: "SELECT * FROM wallet ORDER BY date DESC")
    fun getAll(): List<Wallet>

    @Query( value: "SELECT * FROM wallet WHERE bookmarked = :isBookmarked")
    fun getBookmarked(isBookmarked: Boolean = true): List<Wallet>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(vararg users: Wallet)

    @Delete
    fun delete(wallet: Wallet)

}
```

Figure 15

## **6. FRAGMENT VIEW**

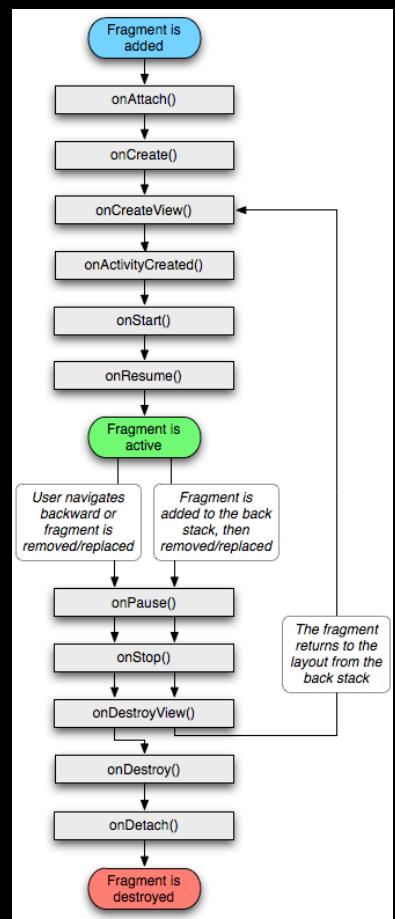
### **I. QU'EST-CE QU'UN FRAGMENT VIEW?**

Un **fragment** représente un comportement ou une partie de l'interface utilisateur dans FragmentActivity. Pour ce projet j'ai combiné plusieurs fragments dans une même activité pour créer une interface utilisateur à plusieurs volets et réutiliser un fragment dans plusieurs activités. Un **fragment** peut être considéré comme une section modulaire d'une activité, qui possède son propre cycle de vie, reçoit ses propres événements d'entrée et qu'on peut ajouter ou supprimer pendant l'exécution de l'activité (un peu comme une "sous-activité" réutiliser dans différentes activités).

### **II. CYCLE DE VIE D'UN FRAGMENT**

Les activités sont régulées par un cycle de vie. Les fragments possèdent des similitudes sur ce point avec les activités. Ils définissent une interface qu'ils contrôlent mais ils ne sont pas associés à un écran. C'est la raison pour laquelle leurs cycles de vie sont semblables mais pas identiques. Ils possèdent les méthodes callback d'une activité, à savoir **onCreate**, **onStart**, **onResume**, **onPause**, **onStop**, **onDestroy**. Ces méthodes ont le même objectif que dans une activité.

(Voir Figure 16 pour cycle de vie d'un fragment).



**Figure 16**

### III. FRAGMENT VIEW DANS « KRYPTNATION »

L'implémentation d'un fragment view a été fait de la manière suivante dans le projet :

```
package com.example.kevinkombate.kryptnation.fragment

import android.support.v4.app.Fragment
import io.reactivex.disposables.CompositeDisposable
open class BaseFragment: Fragment() {

    protected val compositeDisposable: CompositeDisposable by lazy { CompositeDisposable() }

    override fun onDestroyView() {
        compositeDisposable.dispose()
        super.onDestroyView()
    }
}
```

Figure 17 - La classe **BaseFragment** extend la grande classe de **Fragment ()**

```
class BookmarkFragment : BaseFragment() {

    private var listener: OnBookmarkFragmentInteractionListener? = null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                             savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.fragment_bookmark, container, attachToRoot: false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        list.layoutManager = LinearLayoutManager(requireContext())
        updateUI()
        compositeDisposable.add(RxBus.listen(Wallet::class.java).subscribe { updateUI() })
    }
}
```

Figure 18

## 7. ETHERSCAN API

Etherscan API est un service qui permet d'avoir des informations sur la blockchain Ethereum d'un wallet précis. Pour utiliser le service API, il faudrait créer un Api-Key GRATUIT dans la zone ClientPortal-> MyApiKey (du site web <https://etherscan.io/apis>) que vous pouvez ensuite utiliser avec toutes vos demandes API. Il prend en charge à la fois les requêtes GET / POST et une limite de débit de 5 requêtes / sec.

```
object EtherScanApiManager {

    private val apiKey = "H7W179MC2ANBR558RDKTSGAMQAJ8BM5DCU"

    private val retrofit: Retrofit by lazy {
        val httpClient = OkHttpClient.Builder()
        if (BuildConfig.DEBUG) {
            val logging = HttpLoggingInterceptor()
            logging.level = HttpLoggingInterceptor.Level.BODY
            httpClient.addInterceptor(logging)
        }
        ^lazy Retrofit.Builder()
            .baseUrl(baseUrl: "https://api.etherscan.io/")
            .addConverterFactory(GsonConverterFactory.create())
            .client(httpClient.build())
            .build()
    }

    private val etherscan ApiService = retrofit.create(Etherscan ApiService::class.java)

    fun singleAccountBalance(address: String)
        = etherscan ApiService.singleAccountBalance(
            address = address,
            apiKey = apiKey,
            action = "balance",
            module = "account",
            tag = "latest")

    fun transactions(address: String)
        = etherscan ApiService.transactions(apiKey = apiKey,
            action = "txlist",
            module = "account",
            tag = "latest",
            address = address,
            page = 1,
            offset = 100,
            sort = "desc")
}
```

Figure 19

Ma classe « EtherScanApiManager » utilise la clé de l'API, Retrofit pour les requêtes HTTP vers le serveur d'Etherscan. Ces requêtes retournent la valeur du portefeuille, les transactions faites avec le portefeuille.

## E. DIFFICULTES RENCONTREES

Pendant l'implémentation de l'API Etherscan, j'ai rencontré plusieurs problèmes et certaines choses ont dû être revues. Voici en quelques points les difficultés rencontrées lors du développement du projet:

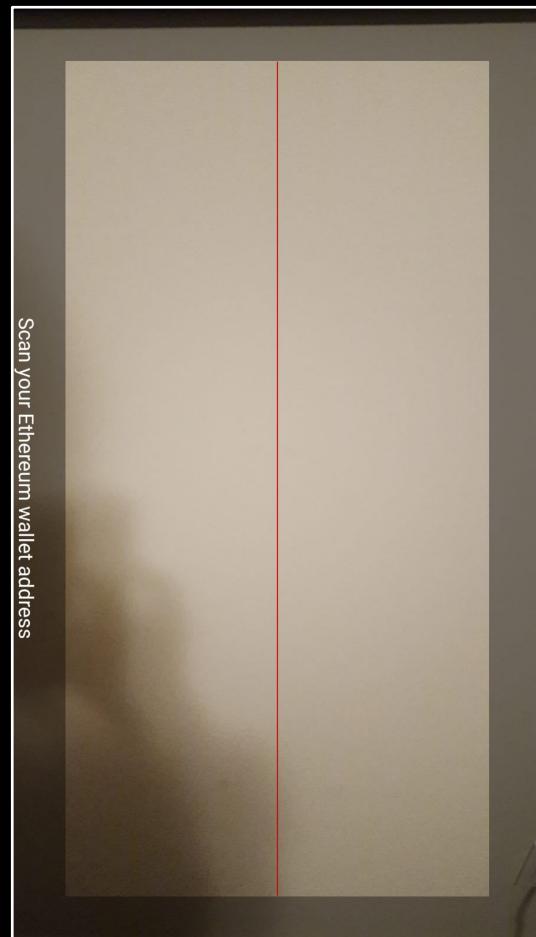
- ETHERSCAN API ne rentrait que la valeur actuelle en Ether, des transactions liées à l'adresse du portefeuille. Comme illustré dans mon prototype papier, j'étais dans l'incapacité d'afficher les tokens.
- Au préalable, j'ai voulu implémenter le QR Scanner Reader avec « Machine Learning KIT » d'Android. Pour une raison inconnue, tout avait bien été implémenté mais, le scanner ne marchait pas en test sur un téléphone Android, ce qui a dû me faire plutôt implémenter cette fonctionnalité avec Zxing Library.
- Comme discuté avec mon superviseur, je voulais implémenter dans mon application la possibilité de créer des wallets Ethereum avec META MASK, mais après de vaine recherche, META MASK n'est pas supporté sur Android, ce qui ne me permettait pas donc d'implémenter cette fonctionnalité.

## F. VERSION FINALE

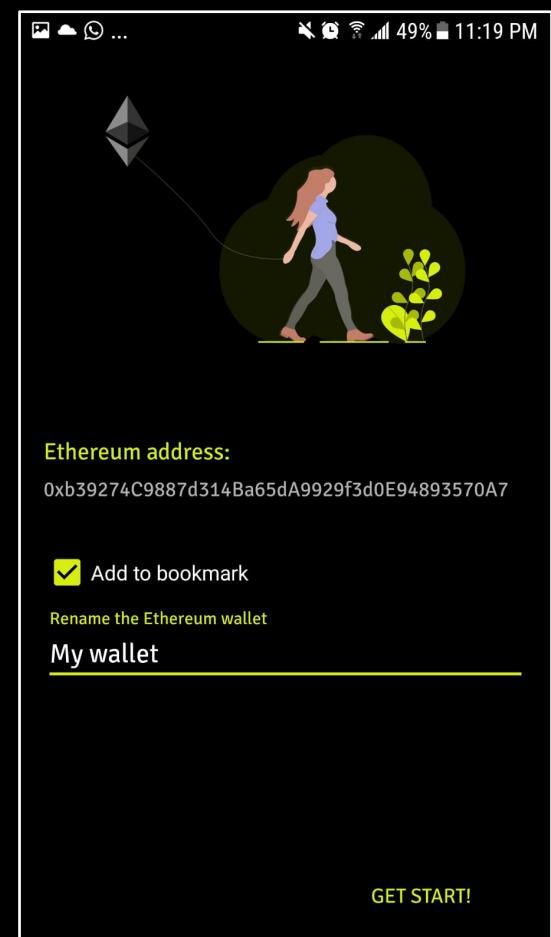
Cette version présente la version finale de l'application conçue sous Android Studio et alimenté par Etherscan API.



activity\_scan : qui demande à l'utilisateur de scanner une adresse d'Ethereum wallet



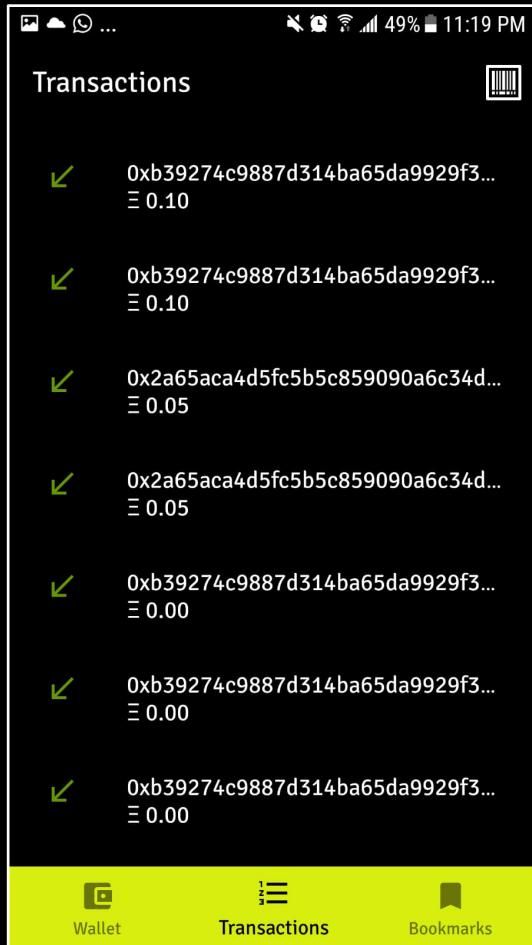
activity\_scan : lecteur de code QR implémenté avec la librairie Zxing



activity\_edit : Après avoir scanné un QR code d'une adresse de portefeuille Ethereum, on peut mettre en favori le portefeuille dans nos favoris et même renommer l'adresse .



**activity\_main** : affiche le nom du portefeuille, son adresse et la valeur en Ether du portefeuille.



**TransactionFragment** : affiche toutes les transactions envoyés ou reçu en relation avec ce portefeuille.



**BookmarkFragment** : affiche toutes les portefeuilles mis en favoris sur ce téléphone.



On peut toutefois scanner un autre portefeuille à l'aide  
de QR code

## **G. SI C'ETAIT A REFAIRE**

Si je pouvais refaire ce projet, je prendrais plus le temps de mieux m'informer sur ETHERSCAN API et mieux exploiter ses données. Le design serait aussi un point où je mettrai plus de temps à développer.

Un autre point sur lequel je me concentrerai sera de créer un espace client où les utilisateurs pourraient avoir accès à tous leurs portefeuilles mis en favoris sur leur téléphone sur d'autres téléphones. Cela permettrait une meilleure utilisabilité de l'application « KryptNation ».

## **H. CONCLUSION**

En conclusion, la cryptomonnaie (Ethereum) est une découverte technologique et financière qui a suscité énormément d'intérêt ces deux dernières années aussi bien sur les marchés financiers que dans le domaine informatique. Ce projet avait pour but de rendre plus facile aux utilisateurs de la cryptomonnaie Ethereum une meilleure utilisabilité des informations relatives aux wallets. Cette application va permettre aux utilisateurs de la plateforme Android de pouvoir visualiser à temps réel les informations sur les différents wallets. Pour un futur proche, le défi serait d'intégrer META MASK à la plateforme Android afin d'augmenter ses fonctionnalités comme créer un wallet Ethereum. Ce projet fût instructif pourra répondre aux besoins de plusieurs utilisateurs.

# ANNEXE – MANUEL D’UTILISATION

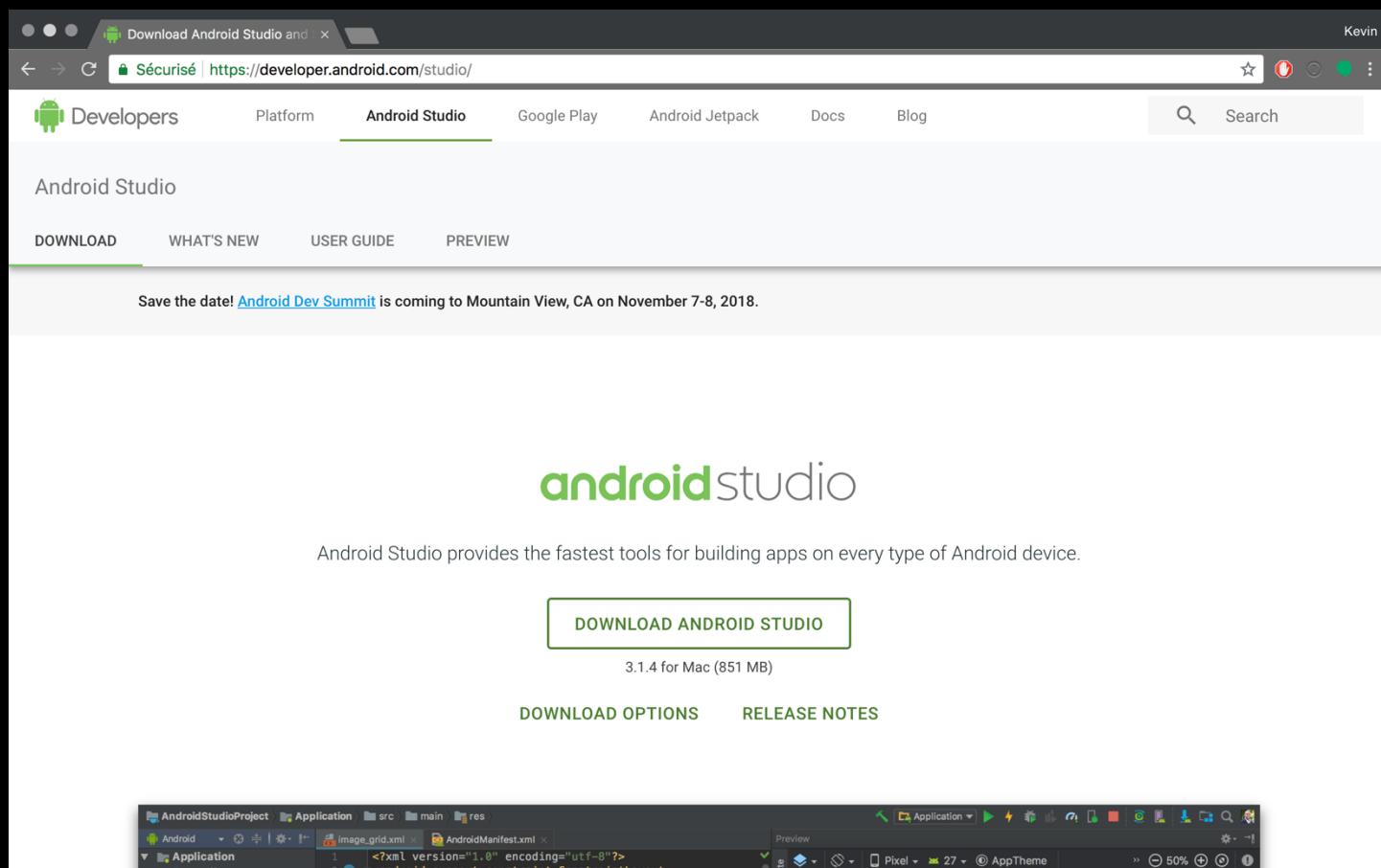
## 1. INSTALLATION DE L’APPLICATION MOBILE

### PREREQUIS :

- Un téléphone ayant un système d’exploitation Android (Version minimum requise API 16 – Jelly Bean).
- Un ordinateur ayant une connexion internet ou Android Studio installé
- Un câble permettant de relier votre téléphone portable (Android) à votre ordinateur.

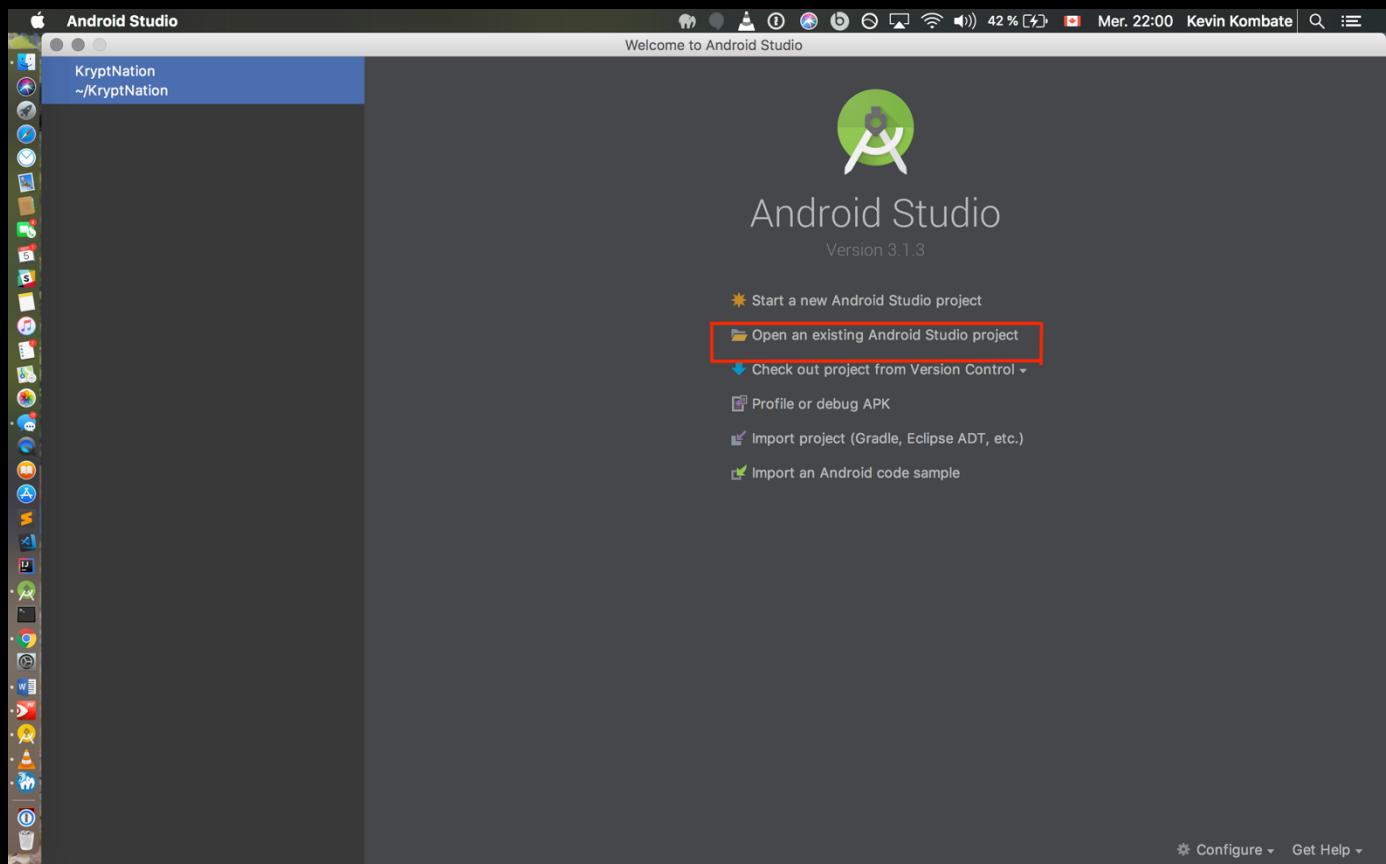
### ETAPE 1 : TELECHARGER ANDROID STUDIO

Saisir l’adresse <https://developer.android.com/studio/> dans votre navigateur et télécharger la dernière version d’Android Studio.

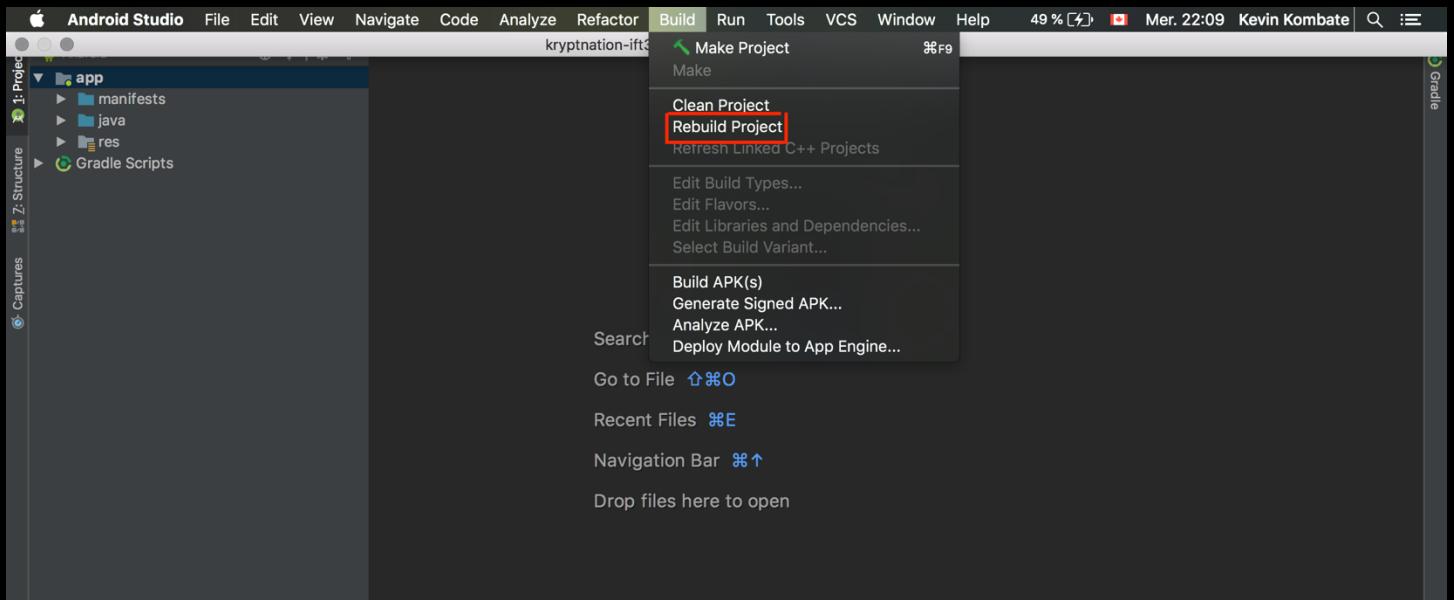


## ETAPE 2 : OUVRIR LE PROJET AVEC ANDROID STUDIO

Ouvrir le projet en utilisant « Open an existing Android Studio project ».



## ETAPE 3 : REBUILD LE PROJET AVEC ANDROID STUDIO

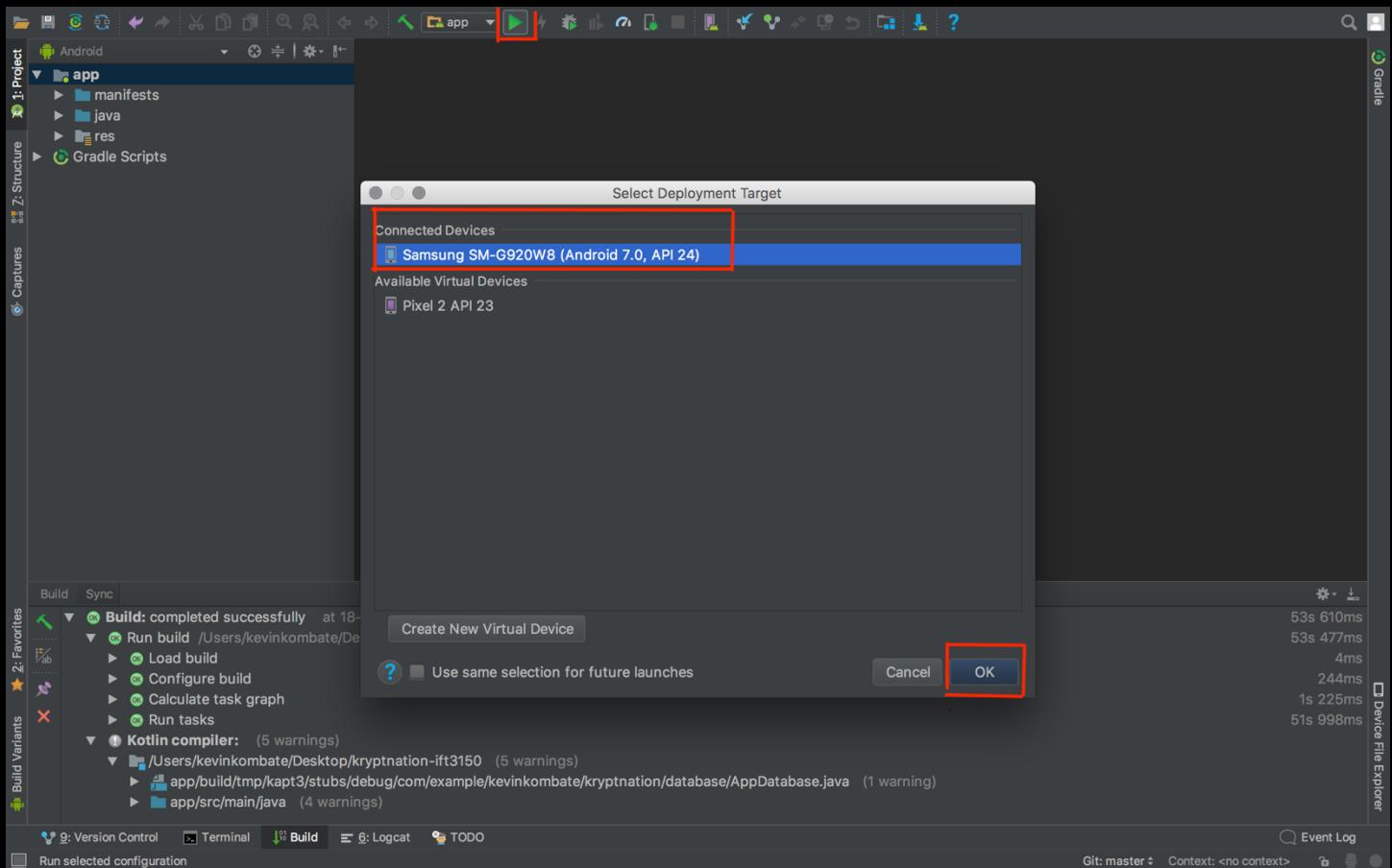


## ETAPE 4 : BRANCHER VOTRE TELEPHONE

Brancher votre téléphone à votre ordinateur via un câble. Assurer vous que vous avez mis votre téléphone en mode développeur (ce qui donne tous les accès afin de pouvoir installer l'application sur votre téléphone).

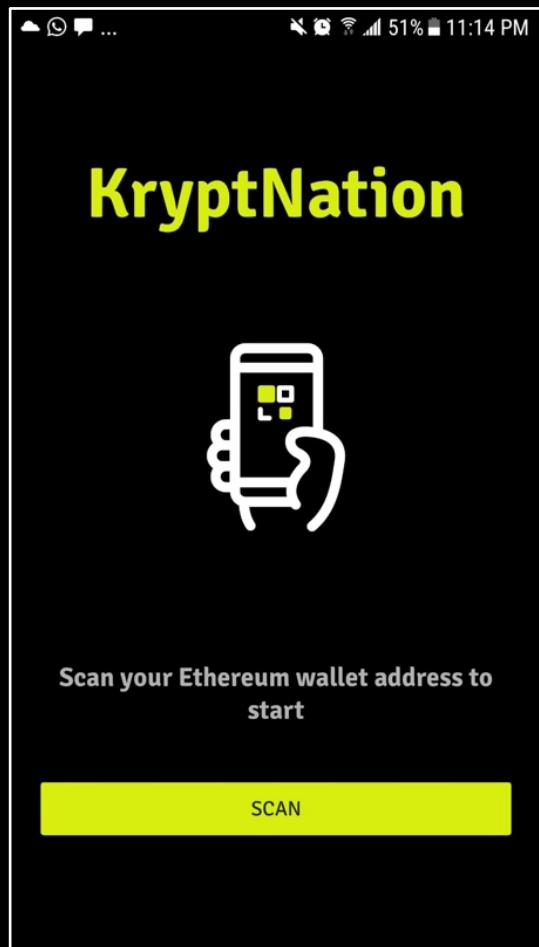
## ETAPE 5 : INSTALLATION DE L'APPLICATION

Pour installer l'application sur votre téléphone, il faut appuyer sur « **Run app** » en haut de l'image, puis ensuite sélectionner votre téléphone qui s'affichera s'il est bien connecté à l'ordinateur et après confirmer votre choix.

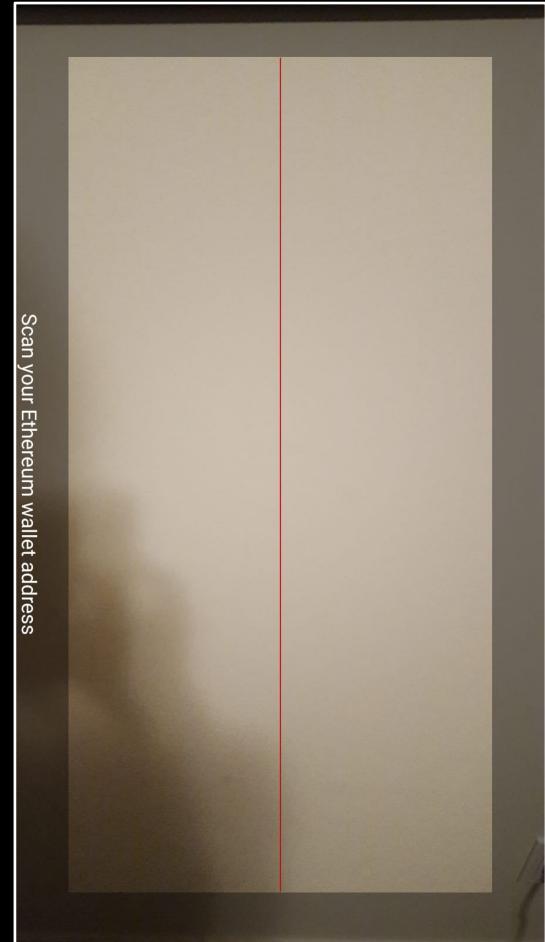


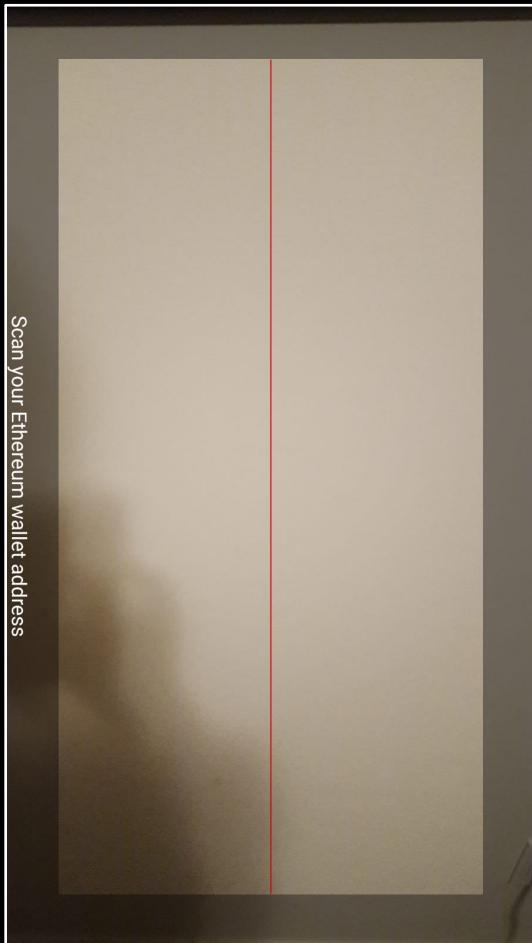
Cela peut prendre quelques minutes, et l'application « **KryptNation** » sera installé sur votre téléphone.

## **2. UTILISATION DE L'APPLICATION**

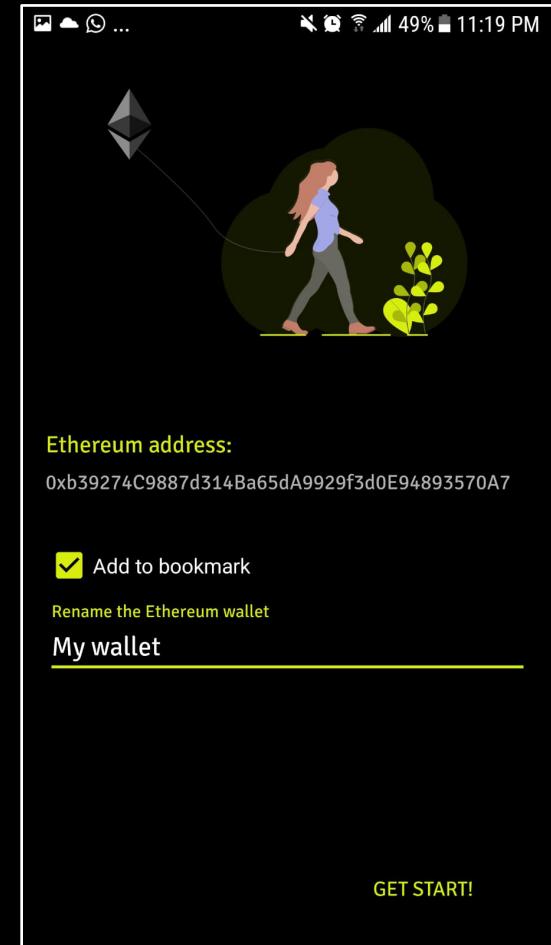


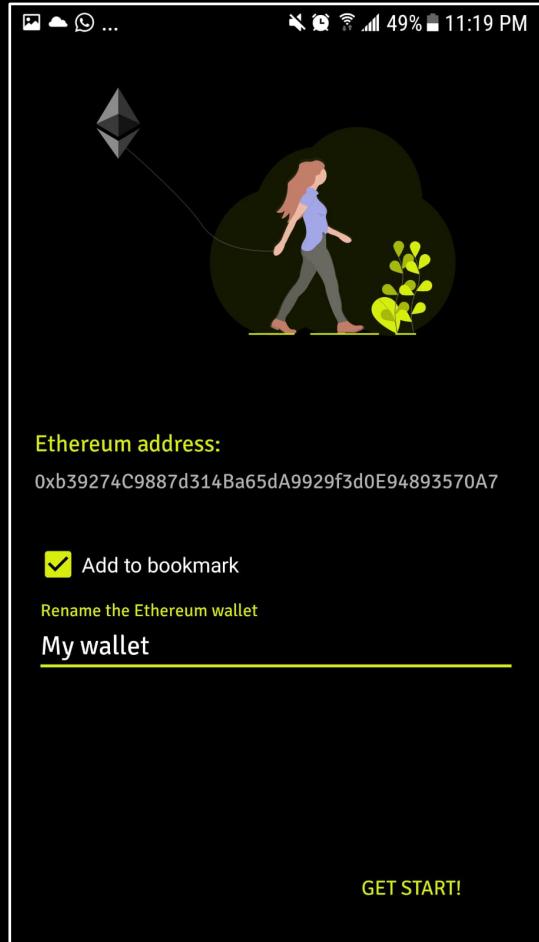
Après avoir appuyé sur le button « SCAN » s'ouvre le lecteur de Code QR qui vous permettra de scanner le code QR d'une adresse de portefeuille Ethereum.



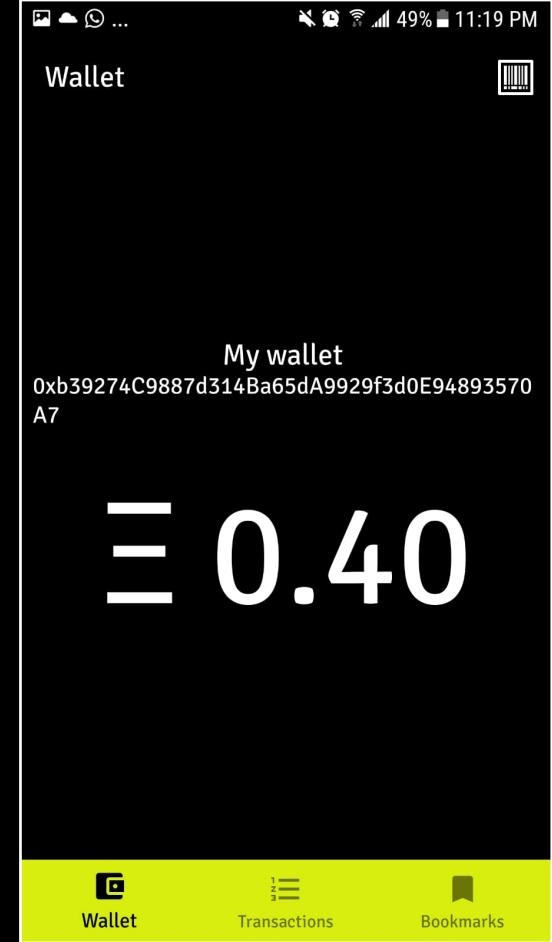


Aussitôt que le lecteur aurait détecté un code QR  
(en relation avec un portefeuille Ethereum)  
s'ouvrira cette activité vous permettant de  
renommer votre portefeuille et l'ajouter aux  
favoris



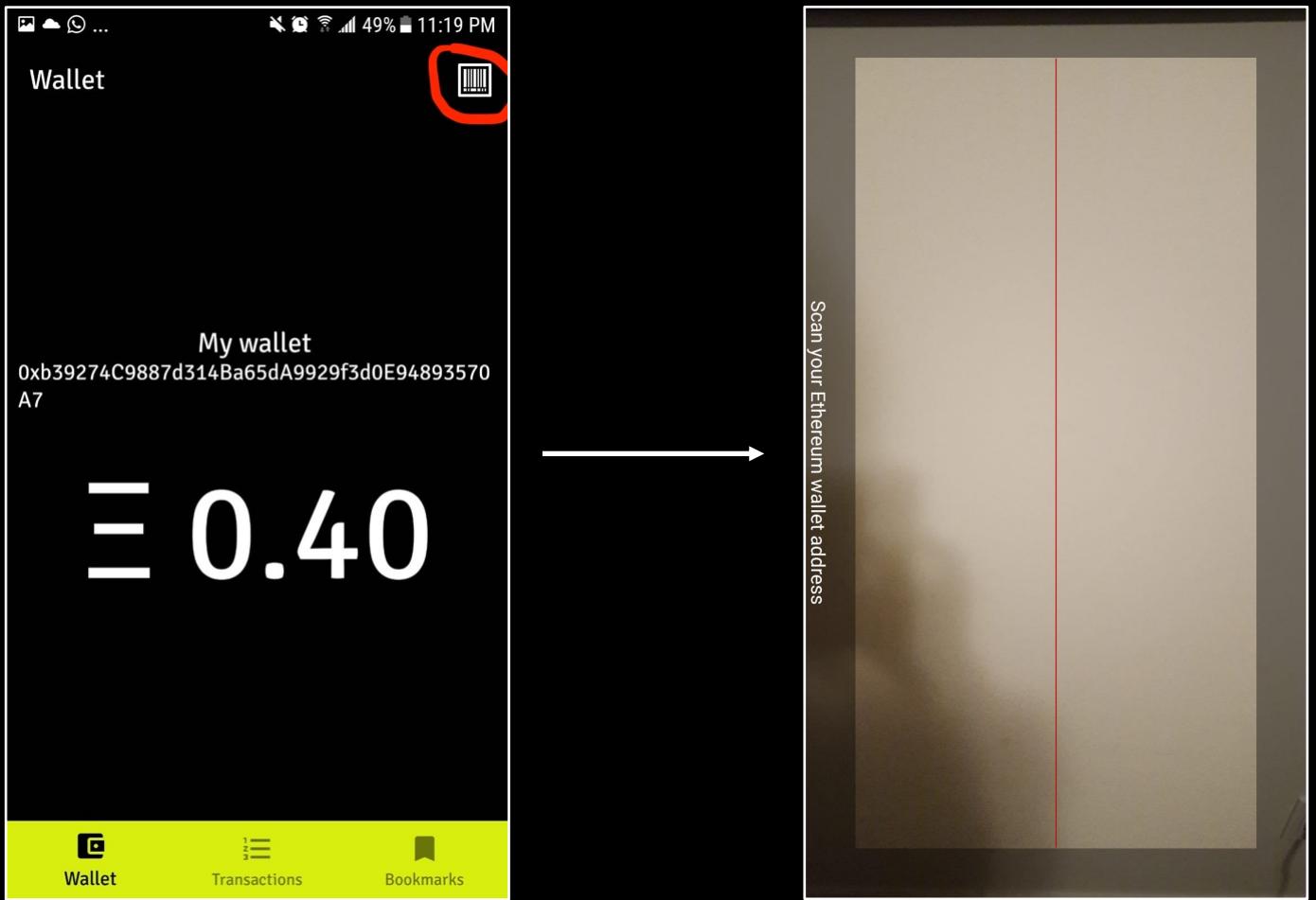


Après avoir appuyé sur “Get Start” s’ouvre l’activité principale qui affiche en bas la barre de navigation, le nom du wallet, l’adresse du portefeuille Ethereum et la valeur en Ether du portefeuille.





En glissant vers la gauche on change de fenêtre en affichant les transactions en relation avec le portefeuille dont vous avez scanné l'adresse précédemment. En glissant vers la gauche encore vous afficherait tous les portefeuilles que vous aviez mis en favoris sur votre téléphone. En glissant vers la droite vous reviendrez vers les transactions ainsi de suite.



Il est possible de scanner d'autres QR code de portefeuille Ethereum en appuyant sur l'image en haut sur votre droite pour ouvrir le lecteur de Code QR.