



Installation, configuration et administration.

Manuel d'utilisation

version 1.0 Beta

Juillet 2020.

MiniBlocklyChain est une marque déposée de OpenQbit Inc, sous une licence d'utilisation libre et commerciale. Conditions d'utilisation à l'adresse suivante : www.OpenQbit.com

Contenu

1.	Introduction	3
2.	Qu'est-ce qu'un réseau public ou privé du système blockchain?	4
3.	Qu'est-ce que la programmation en bloc ?.....	4
4.	Qu'est-ce que Termux ?	5
5.	Qu'est-ce que la Mini BlocklyChain ?	5
6.	Architecture des processus dans la Mini BlocklyChain	9
7.	Schéma de fonctionnement de la BlocklyChain (Mini BlocklyChain).....	12
8.	Qu'est-ce que le Mini BlocklyCode ?	13
9.	Installation du réseau de communication Mini BlocklyChain.....	14
10.	Synchronisation dans les nœuds du système (téléphone mobile) minutes et secondes....	33
11.	Configuration du stockage dans Termux.....	40
12.	Installation du réseau "Tor" et installation de "Syncthing".	41
13.	Installation de la base de données "Redis" et du serveur SSH (Secure Shell).	42
14.	Configuration du serveur SSH sur le téléphone mobile (smartphone).....	43
15.	Configuration du réseau "Tor" avec le service SSH (Secure Shell).....	50
16.	Configuration du système Peer to Peer avec synchronisation manuelle.	53
17.	Ambientes Blockly (App Inventor, AppyBuilder y Thunkable).	65
18.	Qu'est-ce que la preuve de quantum (PQu) ?.....	66
19.	Définition et utilisation des blocs dans la Mini BlocklyChain.....	69
20.	Utilisation de blocs pour la base de données SQLite (version MiniSQLite)	95
21.	Définition et utilisation des blocs de sécurité.....	99
22.	Réglage des paramètres de sécurité dans Mini BlocklyChain.....	109
23.	Annexe "Création des bases de données KeyStore & PublicKeys".	113
24.	Annexe "Commandes RESTful SQLite GET/POST".....	126
25.	Annexe "Code Java SQLite-Redis Connector".	132
26.	Annexe "Mini BlocklyChain pour les développeurs".....	135
27.	Annexe "Contrats intelligents BlocklyCode".	147
28.	Annexe "Informatique quantique à OpenQbit".....	153
29.	Annexe "Blocs étendus pour la base de données SQLite".....	157
30.	Annexe "Exemple de création d'un système de Mini BlocklyChain".....	157
31.	Annexe "Intégration avec les environnements Ethereum et Bitcoin".....	182
32.	Licences et utilisation des logiciels.....	201

1. Introduction.

La chaîne de blocs est généralement associée aux bitcoins et autres devises cryptées, mais ce ne sont que la partie visible de l'iceberg, car elle n'est pas seulement utilisée pour la monnaie numérique, mais peut être utilisée pour toute information pouvant avoir une valeur pour les utilisateurs et/ou les entreprises. Cette technologie, dont les origines remontent à 1991, lorsque Stuart Haber et W. Scott Stornetta ont décrit les premiers travaux sur une chaîne de blocs cryptographiquement sécurisés, n'a pas été remarquée avant 2008, date à laquelle elle est devenue populaire avec l'arrivée des bitcoins. Mais actuellement, son utilisation est demandée dans d'autres applications commerciales et devrait se développer à moyen terme sur plusieurs marchés, comme les institutions financières ou l'Internet des objets, entre autres secteurs.

La blockchain, mieux connue sous le terme de blockchain, est un enregistrement unique, convenu d'un commun accord, réparti sur plusieurs nœuds (appareils électroniques tels que les PC, les smartphones, les tablettes, etc. Dans le cas des crypto-monnaies, on peut considérer qu'il s'agit du livre comptable où chacune des transactions est enregistrée.

Son fonctionnement peut être complexe à comprendre si nous entrons dans les détails internes de sa mise en œuvre, mais l'idée de base est simple à suivre.

Il est stocké dans chaque bloc :

1.- un certain nombre d'enregistrements ou de transactions valables,

2.- des informations concernant ce bloc,

3.- son lien avec le bloc précédent et le bloc suivant grâce au hachage de chaque bloc –un code unique qui serait comme l'empreinte digitale du bloc.

Par conséquent, **chaque bloc a une place spécifique et immuable dans la chaîne**, car chaque bloc contient des informations provenant du hachage du bloc précédent. La chaîne entière est stockée sur chaque nœud du réseau qui constitue la chaîne de blocs, de sorte qu'**une copie exacte de la chaîne est stockée sur tous les participants du réseau**.

Lorsque de nouveaux enregistrements sont créés, ils sont d'abord vérifiés et validés par les nœuds du réseau, puis ajoutés à un nouveau bloc qui est lié à la chaîne.

Maintenant, imaginez que ce réseau d'appareils qui communiquent entre eux, a la capacité d'interagir sans l'intervention d'une personne, c'est-à-dire que le grand avantage de la chaîne de blocage est qu'il peut prendre des décisions autonomes, ce qui bénéficie en temps de réponse du service aux utilisateurs, disponible 24 heures sur 24, minimise les coûts dans les

entreprises et a tout d'abord un niveau de sécurité déjà testé, pour cette raison et d'autres est devenu si populaire en usage dans les différents secteurs publics et privés.

2. Qu'est-ce qu'un réseau public ou privé du système blockchain?

Réseau public. - C'est un réseau d'ordinateurs ou d'appareils mobiles qui communiquent entre eux et conservent l'anonymat. On ne sait pas qui interagit dans ce réseau de manière formelle dans ses transactions, dans ce type de réseau, toute personne ou entreprise peut interagir et se connecter à tout moment car vous n'avez pas besoin d'autorisation pour vous connecter, un exemple est la chaîne de blocage de Bitcoin, n'importe qui peut y entrer pour acheter ou vendre. Normalement, ce type de réseaux est orienté ou dirigé vers l'achat et la vente de biens monétaires numériques ou son synonyme "cryptomonnaies", par exemple : DogCoin, Ethereum, LiteCoin, BitCoin, Waves, etc.

Réseau privé. - Il s'agit d'un réseau d'ordinateurs ou d'appareils mobiles qui communiquent entre eux. Toutefois, contrairement aux réseaux publics, les réseaux privés nécessitent l'autorisation préalable d'une entité (entreprise ou personne) pour se connecter et faire partie de ce type de réseau. Normalement, les réseaux privés en chaîne sont utilisés dans les entreprises ou les sociétés pour effectuer des transactions ou des opérations avec différents types d'informations qui peuvent avoir une valeur tangible sous forme de documents, de processus, d'autorisations et/ou de décisions commerciales appliquées et supervisées par la chaîne, par exemple : le secteur financier, le secteur des assurances, le gouvernement, entre autres.

3. Qu'est-ce que la programmation en bloc ?

Blockly est un **langage de programmation visuel** composé d'un simple ensemble de commandes que nous pouvons combiner comme s'il s'agissait des pièces d'un puzzle. C'est un outil très utile pour ceux qui veulent **apprendre à programmer** de manière intuitive et simple ou pour ceux qui savent déjà programmer et qui veulent voir le potentiel de ce type de programmation.

Blockly est une forme de programmation où il n'est pas nécessaire d'avoir des connaissances dans un quelconque langage informatique, c'est parce qu'il s'agit simplement de joindre des blocs graphiques comme si nous jouions au lego ou à un puzzle, il suffit d'avoir un peu de logique et c'est tout !

Tout le monde peut créer des programmes pour les téléphones mobiles (smartphones) sans se frotter aux langages de programmation difficiles à comprendre, il suffit d'assembler des blocs de manière graphique, de façon simple, facile et rapide à créer.

4. Qu'est-ce que Termux ?

Termux est un émulateur de terminal Android et une application en environnement Linux qui fonctionne directement sans avoir besoin de routage ou de configuration. Un système de base minimal est automatiquement installé.

Nous utiliserons Termux pour sa stabilité et sa facilité d'installation et de gestion, cependant, vous pouvez utiliser un environnement installé de Ubuntu Linux pour Android.

Dans cet environnement Linux, vous aurez le "noyau" des processus de communication de la MiniBlocklyChain.

5. Qu'est-ce que la Mini BlocklyChain ?

La Mini BlocklyChain est une chaîne de blocage entièrement fonctionnelle, une technologie développée pour les téléphones mobiles avec un système d'exploitation **Android** qui sera le nœud qui effectuera les transactions d'envoi et de réception. Nous avons créé la première technologie de chaîne de blocs qui est structurée de manière "modulaire" grâce à la programmation en bloc où toute personne ayant des connaissances minimales et ne sachant pas programmer peut créer et développer des programmes pour les téléphones mobiles et créer sa propre chaîne de blocs en mode réseau public ou privé. Si vous voulez créer votre propre monnaie numérique, vous pouvez le faire ou une solution pour l'utiliser dans une entreprise, les possibilités sont basées sur les besoins de chaque cas réel et la logique commerciale que l'utilisateur adopte ou crée selon ses exigences.

La Mini BlocklyChain est la première chaîne modulaire pour téléphones mobiles où un système transactionnel peut être mis en place en peu de temps.

Avant d'entrer dans la définition et l'utilisation des blocs "module", nous devons avoir les concepts de base des composants de la Mini BlocklyChain pour savoir quand, comment et où les appliquer en fonction du cas réel que nous voulons mettre en œuvre.

Les concepts suivants mettent l'accent sur le type d'utilisateur qu'ils ciblent. Il n'est donc pas important pour les personnes qui n'ont pas de compétences en programmation que les concepts pour les utilisateurs du développement soient bien compris.

Concepts de base :

Nœud. - Chaque appareil mobile (téléphone, tablette, etc.) avec un système d'exploitation Android faisant partie du réseau public ou privé de Mini BlocklyChain est nommé comme un nœud de ce réseau.

Hash. - Il s'agit d'une signature numérique qui est associée à un ensemble de données, une chaîne de caractères, des documents ou une sorte d'information numérique, cette signature

numérique est unique et non répétable ce qui permet d'envoyer ou de recevoir des informations de sorte que le contenu initial des informations envoyées ne peut pas être modifié.

Actif. - Tout type de données ou d'informations numériques qui peuvent être pondérées par une certaine valeur matérielle ou immatérielle pour les personnes ou les entreprises (documents, pièces numériques, musique, vidéo, images, autorisations numériques, etc.)

Transaction. - Échange d'informations entre des nœuds occupant une sorte d'actif.

Transaction UXTO - C'est un type de transaction qui regroupe une ou plusieurs transactions des nœuds et toutes les transactions non dépensées de chaque noeud (UXTO : Unspent Transaction Outputs) peuvent être dépensées comme entrée dans une nouvelle transaction. Ces transactions sont regroupées dans une file d'attente à traiter à chaque instant, qui peut être réglée en fonction des exigences de chaque flux d'information.

Transaction (saisie). - Lorsqu'une file d'attente de transactions UXTO entre, elle est traitée par une transaction d'**entrée** qui classe la transaction comme un dépôt ou une dépense et peut ainsi avoir un solde du résultat final pour chaque utilisateur.

Adresse de la source. - Il s'agit de l'adresse de la personne qui génère ou envoie la transaction à traiter dans la file d'attente SQLite Master. Il s'agit d'un nombre alphanumérique de 64 caractères qui est créé et vérifié par le système.

Adresse de destination. - Il s'agit de l'adresse de la personne qui reçoit la transaction et l'ajoute à son solde ou stocke le bien envoyé. Il s'agit d'un numéro alphanumérique de 64 caractères qui est créé et vérifié par le système.

SQLite Master. - Base de données API REST qui reçoit les transactions et crée une file d'attente à traiter à chaque instant variable ou fixe selon le besoin de l'entreprise.

Transaction DataBase. - Ce sont les transactions qui sont reflétées dans la base de données SQLite qui réserve ou stocke les informations sur les transactions de la Mini BlocklyChain.

AES (Advanced Encryption Standard) - C'est un processus de sécurité qui crypte les données qui sont stockées dans la base de données SQLite de Mini BlocklyChain.

Équilibre. - Après avoir effectué un processus de transaction dans la Mini BlocklyChain, un équilibre de l'opération est obtenu soit comme une valeur tangible (cryptomoney) ou intangible (processus d'affaires entre personnes ou entreprises).

Processus d'affaires. - Il s'agit de tout processus qui implique un certain type de flux d'informations pour obtenir un résultat à un utilisateur final, l'utilisateur final peut être une personne ou une entreprise de divers secteurs du secteur privé ou public.

Clé publique et privée. - Il s'agit d'un type de cryptage des informations où deux clés alphanumériques associées l'une à l'autre sont générées et utilisées pour envoyer des

informations sensibles par le biais de réseaux publics ou privés. La clé privée est utilisée pour crypter les informations et, lors de leur envoi par le réseau, elle ne peut pas être modifiée ou être lisible à première vue, la clé publique est celle qui est partagée et est vue par toute personne ou entreprise et celle-ci aidera à vérifier les informations envoyées, ainsi qu'à ne pouvoir les lire que par le destinataire valide.

Signature numérique. - C'est une signature qui peut être créée avec la clé privée, avec cette signature le destinataire s'assure que l'information n'a pas été altérée et confirme que l'information est valable pour le destinataire.

Adresse numérique (Mini BlocklyChain address). - C'est une adresse qui est composée de caractères alphanumériques uniques à chaque utilisateur de Mini BlocklyChain, cette adresse est utilisée pour effectuer des transactions entre les utilisateurs et qu'il s'agisse de réseau public ou privé.

Numéro magique. - Il s'agit d'un nombre aléatoire défini par des règles de gestion pour chaque transaction UXTO en cours, qui sert à autoriser le nœud à être candidat pour exécuter la transaction UXTO et à créer un nouveau bloc, la Mini BlocklyChain.

Création d'un nouveau bloc. - Un nouveau bloc dans la Mini BlocklyChain est un hachage créé et attaché par le nœud qui est sorti comme le candidat gagnant pour traiter la transaction UXTO.

Protocole de consensus PoW (preuve de travail). - Il s'agit d'un test qui exécute tous les nœuds et le premier nœud qui termine le test avec succès est celui qui a été choisi pour exécuter la transaction UXTO. C'est un algorithme qui est composé de calculs mathématiques pour obtenir un résultat (hash) selon les règles données dans chaque transaction exécutée par Mini BlocklyChain est basé sur l'usure ou la demande des ressources de traitement de l'information des ordinateurs, dans le cas de Mini BlocklyChain a été structuré et modifié pour obtenir un "nombre magique" c'est un nombre pour pouvoir avoir l'autorisation ou le consensus de la majorité des noeuds pour être celui qui peut exécuter la transaction UXTO. Le PoW a un niveau de difficulté maximum de 5 puisqu'il n'est utilisé que pour obtenir le "nombre magique".

Protocole de consensus PoQu (test quantique) - Il s'agit d'un test qui exécute tous les nœuds et qui est composé initialement par le PoW pour obtenir le "nombre magique" et ensuite il exécute un algorithme de probabilité basé sur un QRNG (Quantum Random Number Generator) ; un générateur de nombres aléatoires quantiques, ce processus assure l'égalité de probabilité pour tous les nœuds et ainsi choisir quel nœud sera celui qui exécute la transaction UXTO et la création du nouveau bloc.

L'arbre Merkle. - Il s'agit d'une méthode de sécurité pour assurer à la Mini BlocklyChain que les transactions sont valides ou n'ont pas été modifiées par une entité externe. Un arbre de hachage est une structure arborescente de données binaires ou non binaires dans laquelle

chaque nœud qui n'est pas une feuille est étiqueté avec le hachage de la concaténation des étiquettes ou des valeurs de ses nœuds enfants. Il s'agit d'une généralisation des listes de hachage et des chaînes de hachage.

Redis DB. - Base de données des transactions en temps réel utilisée pour traiter et envoyer aux nœuds les nouvelles transactions qui ont été demandées.

SQLite DB. - Base de données où sont stockés les soldes et les nouveaux blocs pour la Mini BlocklyChain afin de garantir l'intégrité du réseau.

Sentinelle. - Connecteur de sécurité et d'intégrité des données entre Redis et SQLite. Ce connecteur ou programme est chargé d'examiner, de traiter, de valider, de distribuer et de traduire les transactions vers les nœuds.

OpenSSH. - Connecteur de sécurité pour exécuter des tâches dans le système d'exploitation Android.

Termux Shell Terminal. - Programme où vous pouvez trouver des dépendances de tiers pour traiter les transactions de Mini BlocklyChain, dans cette version 1.0 il est utilisé pour une installation facile mais dans les versions futures il sera remplacé par le système BlocklyShell qui est actuellement en cours de développement.

Portefeuille. - Il s'agit du dépôt numérique où sont conservées deux données fondamentales dans toute transaction ; les clés publiques et privées qui seront utilisées respectivement comme adresse source et signature numérique pour toute transaction effectuée, ainsi que le solde des transactions envoyées ou reçues. Il s'agit d'un dépôt où sont conservées les adresses des Mini BlocklyChain, ainsi que les résultats des transactions UXTO (Balance).

Développeur d'applications ou programmeur :

Programmation orientée objet (Java) - La Mini BlocklyChain est fabriquée à Java.

BlocklyCode. - C'est le terme des contrats intelligents qui peuvent être exécutés dans l'environnement Mini BlocklyChain, le BlocklyCode est créé en programmation java.

OpenJDK pour Android. - C'est la suite de JDK et JRE pour Android où les BlocklyCode sont créés et exécutés.

QRNG (Quantum Random Number Generator). - Il s'agit d'un générateur de nombres aléatoires quantiques, la Mini BlocklyChain dispose actuellement de deux API pour la génération de ceux-ci.

rsync. - C'est une application gratuite pour les systèmes de type Unix et Microsoft Windows qui offre une transmission efficace de données incrémentielles, qui fonctionne également avec des données compressées et cryptées.

ffsend. - Il s'agit d'une application permettant de partager des fichiers facilement et en toute sécurité depuis la ligne de commande.

NTP. - Network Time Protocol, est un protocole Internet permettant de synchroniser les horloges des systèmes informatiques par le biais du routage de paquets dans des réseaux à latence variable.

Termux (développement). - Terminal Shell avec un large éventail d'applications et de bibliothèques open source.

Modules (données) en bloc. - Les développeurs peuvent intégrer des modules pour améliorer les fonctionnalités de la Mini BlocklyChain.

Peer to Peer. - Ce terme désigne la communication entre les nœuds de manière directe, c'est-à-dire que la mise à jour des informations ne dépend pas d'un serveur central mais que chaque nœud fonctionne comme un serveur central qui communique entre eux en ayant tous les mêmes informations, ce qui permet d'éviter les points de défaillance.

Synchronisation. - outil de synchronisation de données ou de fichiers entre deux appareils utilisant le type de communication "Peer to Peer".

Red Tor. - Il s'agit d'un réseau de communication distribué à faible latence superposé à l'internet, dans lequel le routage des messages échangés entre les utilisateurs ne révèle pas leur identité, c'est-à-dire leur adresse IP (anonymat à l'échelle du réseau).

Routage mobile. - Processus d'installation d'un logiciel externe sur votre téléphone pour entrer en tant qu'administrateur système dans les systèmes d'exploitation Linux (Android) ; l'utilisateur administrateur est appelé "root" pour pouvoir faire tourner votre téléphone aura accès à tout processus. Il est important de noter que certains fabricants de téléphones portables (Smartphones) affirment que la garantie est perdue en raison d'un défaut quelconque du téléphone portable.

6. Architecture des processus dans la Mini BlocklyChain

La Mini BlocklyChain est constituée de trois processus qui forment son architecture, le premier étant les processus commerciaux, le deuxième les processus de communication et le troisième l'environnement de développement de modules complémentaires et/ou la création de "contrats intelligents" BlockyCode.

Les processus commerciaux sont les blocs qui forment une série de routines pour créer une transaction soit en réseau public, soit en réseau privé. Ce type de processus est la planification de l'entreprise, le comment, le quand, le quoi, le qui, le où et d'autres attributs seront commandés, planifiés et distribués de manière à ce que la fonctionnalité principale

de chaque transaction envoyée, reçue, stockée et/ou rejetée soit réalisée. Le premier processus est composé des types de blocs suivants, divisés en quatre catégories :

1. Blocs de saisie de données
2. Blocs de traitement des données
3. Blocs de sécurité.
4. Blocs de données modulaires (développeurs)
5. Les robots de communication.

Les blocs de saisie sont ceux qui reçoivent la transaction avec un minimum de quatre paramètres d'entrée (**adresse source, adresse destination, actif, données**) et peuvent en avoir plus selon la variable "data", il peut s'agir d'un bloc développé par des tiers ou avec une valeur nulle (NULL).

Les blocs de traitement des données développent la logistique, le calcul, la normalisation des données, les décisions logiques et les contrôles de flux pour chaque transaction. Ces types de blocs sont utilisés pour donner des renseignements au processus commercial et sont principalement basés, comme leur nom l'indique, sur le traitement de tous les types d'informations, ainsi que sur leur conversion à partir de différents types de données.

Les blocs de sécurité sont utilisés pour valider les informations et s'assurer que les transactions n'ont pas été altérées de leur origine à leur destination, ils sont toujours traités avec différents algorithmes de sécurité utilisés dans les technologies de la chaîne de blocs, parmi les outils les plus utilisés sont (signature de hachage, signature numérique, cryptage des données AES, création et validation des adresses numériques, entre autres).

Les blocs de données modulaires, ce sont les blocs qui sont développés par des tiers, rappelons que la Mini BlocklyChain a été créée de manière modulaire pour être enrichie de nouveaux blocs en fonction des besoins de chaque secteur, qu'il soit public ou privé. Pour en savoir plus sur la façon de créer des modules, consultez l'annexe sur les développeurs.

Comme nous l'avons commenté précédemment, l'architecture de Mini BlocklyChain dans sa deuxième composante est constituée de processus de communication, ces processus sont ceux qui sont en charge des canaux de communication via TCP et des Sockets de communication pour donner la flexibilité d'envoyer et de recevoir des transactions soit par les différents moyens qui sont utilisés à l'heure actuelle les plus utilisés sont : Internet mobile, le Wifi fonctionne actuellement pour inclure le moyen de communication Bluetooth.

Les blocs de communication sont essentiellement basés sur l'échange de données avec une sécurité mise en œuvre dans le canal de transfert et ceci est basé sur une communication à travers le protocole SSH (Secure Shell) avec les différentes étapes par lesquelles passe une transaction envoyée ou reçue.

La partie communication est fondamentale puisque ces processus ont pour fonction de mettre à jour les informations dans tous les nœuds via TCP et la connexion appelée "**Peer to OpenQbit.com**

"Peer" ; les blocs qui interviennent dans la communication sont basés sur l'échange d'informations entre les nœuds sans l'intervention de serveurs intermédiaires ; ainsi, chaque nœud est indépendant ; il est possible de créer un réseau de nœuds où les points de défaillance sont minimaux ou presque nuls. De même, cette indépendance de chaque nœud les aide à prendre des décisions individuellement ou globalement en fonction des besoins de l'entreprise. L'architecture "Peer to Peer" se compose de trois parties qui forment le réseau public ou privé que vous décidez de créer. Dans les deux cas, le canal de communication est crypté de nœud à nœud.

Le premier élément de communication entre les appareils mobiles (smartphones) ou le wifi est de fournir aux nœuds ou aux appareils un réseau où ils peuvent être trouvés partout dans le monde, le réseau de communication où la MiniBlocklyChain est montée est le réseau "**Tor**".

Qu'est-ce que le réseau Tor ? - (<https://www.torproject.org>)

"**Tor** (acronyme de Te **Onion Router** - en espagnol) est un projet dont l'objectif principal est le développement d'un réseau de communication distribué à faible latence superposé à l'Internet, dans lequel le routage des messages échangés entre les utilisateurs ne révèle pas leur identité, c'est-à-dire leur adresse IP (anonymat au niveau du réseau) et qui, de plus, maintient l'intégrité et le secret des informations qui transitent".

La deuxième composante et une tâche non moins importante consiste à faire en sorte que tous les nœuds de la MiniBlocklyChain aient les mêmes données à tout moment ou que leurs bases de données et leurs fichiers soient synchronisés pour accomplir cette tâche entre les nœuds sera mise en œuvre "**syncing**".

Qu'est-ce que le réseau Syncing ? - (<https://syncing.net>)

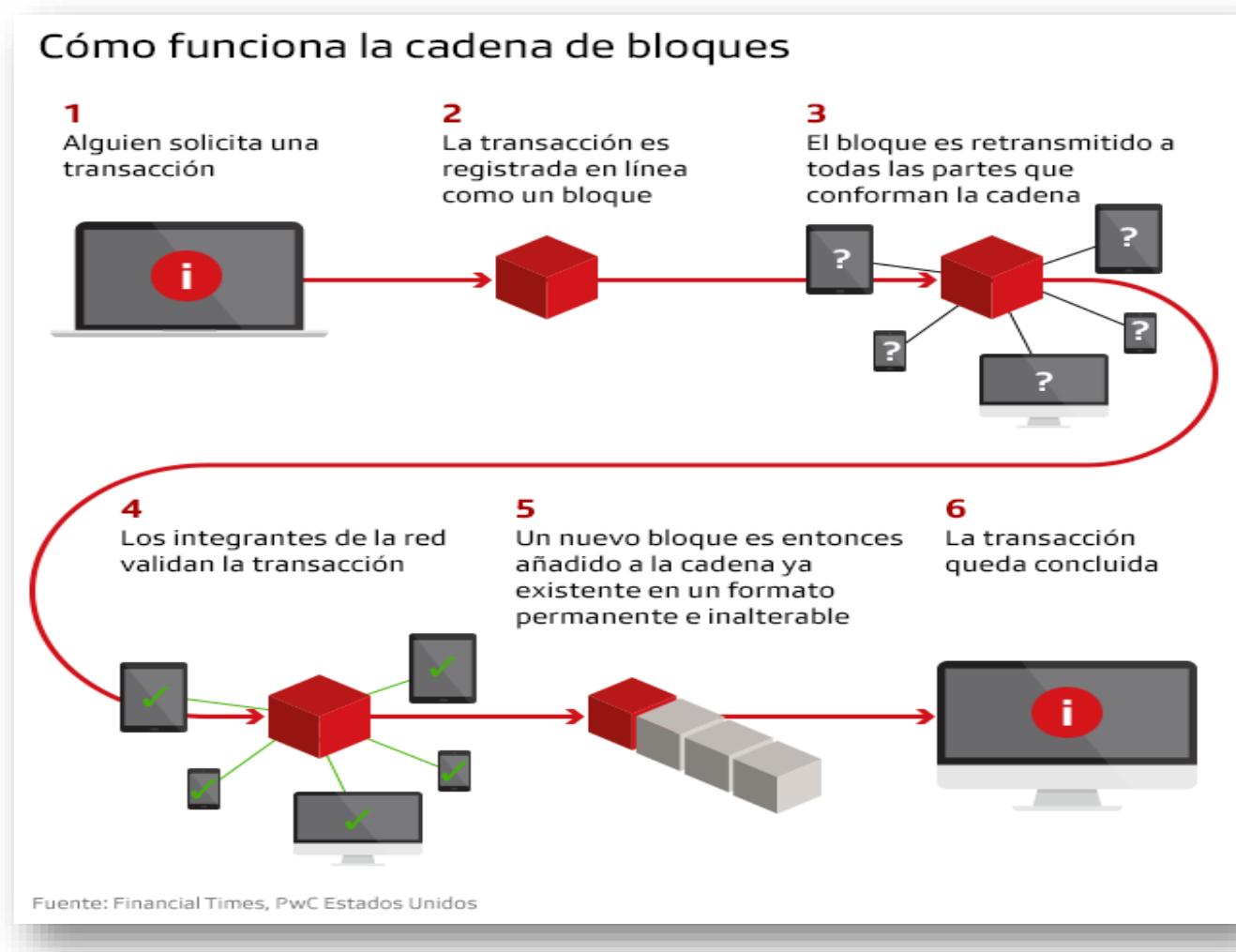
Syncing est une application de synchronisation de fichiers peer-to-peer libre et open source disponible pour Windows, Mac, Linux, Android, Solaris, Darwin et BSD. Vous pouvez synchroniser des fichiers entre des appareils sur un réseau local ou entre des appareils distants sur Internet.

Sur la base des deux composantes de communication précédentes, nous pouvons commencer à développer un réseau de confiance entre les nœuds et avec le niveau de synchronisation des données qui sera l'épine dorsale ou le "œur" des processus commerciaux afin de répondre à tous les besoins de l'utilisateur ou de l'entreprise.

La troisième composante du réseau de communication est la base de données Redis, qui informera tous les nœuds en temps réel des nouvelles transactions reçues et des nouvelles transactions envoyées.

Qu'est-ce que le réseau Redis DB ? - (<https://redis.io>). **Redis** est un moteur de base de données en mémoire, basé sur le stockage dans des tables de hachage (clé/valeur) mais qui peut être utilisé en option comme une base de données durable ou persistante.

7. Schéma de fonctionnement de la BlocklyChain (Mini BlocklyChain).



8. Qu'est-ce que le Mini BlocklyCode ?

Les Mini BlocklyCode sont des programmes créés dans le langage Java et sont stockés dans un référentiel indépendant des nœuds, ils sont normalement appelés contrats intelligents, ces programmes sont conçus avec des conditions logiques car lorsque ces conditions sont remplies, ils sont exécutés automatiquement sans dépendre d'aucune autorisation ou intervention humaine externe.

"Un contrat intelligent est un programme informatique qui facilite, sécurise, fait respecter et exécute les accords enregistrés entre deux ou plusieurs parties (par exemple, des personnes ou des organisations). En tant que tels, ils les aideraient à négocier et à définir de tels accords qui entraîneront la réalisation de certaines actions en raison du respect d'un certain nombre de conditions spécifiques.

Un contrat intelligent est un programme qui vit dans un système non contrôlé par l'une ou l'autre des parties, ou leurs agents, et qui exécute un contrat automatique qui fonctionne comme une phrase "si" de tout autre programme informatique. A la différence qu'il est fait de manière à interagir avec des actifs réels. Lorsqu'une condition préprogrammée, non soumise à un jugement humain, est déclenchée, le contrat intelligent exécute la clause contractuelle correspondante.

Ils visent à offrir une plus grande sécurité que le droit des contrats traditionnel et à réduire les coûts de transaction liés à la passation de contrats. Le transfert de la valeur numérique par un système qui ne nécessite pas de confiance (par exemple, les bitcoins) ouvre la porte à de nouvelles applications qui peuvent faire appel à des contrats intelligents. "

Mini BlocklyCode est destiné aux développeurs ayant une expérience de la programmation en Java. Dans la Mini BlocklyChain, certains types de bibliothèques sont restreints pour des raisons de sécurité du même système, cependant, des bibliothèques peuvent être créées pour créer des transactions pour envoyer ou recevoir une certaine valeur contractuelle créée ou convenue par deux ou plusieurs parties.

Chaque Mini BlocklyChain est conforme aux principes suivants :

- ✓ Toute Mini BlocklyChain créée est basée sur l'exécution de messages uniquement et non d'exécutions sur le système, cependant, ces messages peuvent contenir des autorisations, des approbations de contrats physiques ou des actions physiques.
- ✓ Chaque Mini BlocklyChain créée doit passer par le bloc de communication "auditeur". Ce bloc est chargé de surveiller les codes malveillants ou les codes interdits afin de revoir les phrases ou les ordres non autorisés dans le système.
- ✓ Toutes les Mini BlocklyChain sont exécutées uniquement sur la JVM du nœud source, les programmes ne sont pas exécutés globalement pour la protection du système.

Pour plus de détails, voir l'annexe "Mini BlocklyChain for Developers".

9. Installation du réseau de communication Mini BlocklyChain

1. Installation et configuration du réseau Mini SQLSync

Le réseau Mini SQLSync est chargé de recevoir les transactions des nœuds afin qu'ils puissent les traiter. Ce réseau est constitué d'un réseau principal qui se fait par "Peer to Peer" entre les nœuds et d'un réseau de secours appelé Mini Sentinel RESTful.

Nous commencerons par l'installation du réseau de sauvegarde RESTful Mini Sentinel, ce service est celui qui recevra les transactions des nœuds, ce service est basé sur le stockage des transactions pendant un temps déterminé pour ensuite convertir les informations à travers un connecteur qui injectera une file d'attente de toutes les transactions cryptées vers un service Redis. Plus tard, le service de base de données Redis exécutera en temps réel la libération de la file d'attente des transactions cryptées vers tous les nœuds qui intègrent le réseau Mini BlocklyChain.

Un service ou une conception de type RESTful est un moyen simple et facile de consulter, modifier, supprimer et/ou insérer des informations dans une base de données par le biais d'une URL ou d'une adresse internet. Dans notre cas, nous utiliserons la base de données SQLite en raison de ses caractéristiques qui consistent à regrouper toutes les informations dans un seul fichier. Pour une utilisation des besoins à grande échelle nous pourrons utiliser un autre type de base de données comme MySQL, Oracle, DB2 ou une autre base de données commerciale, simplement nous devrons changer le connecteur de Mini BlocklyChain de SQLite (version gratuite) pour le connecteur de Mini BlocklyChain DB autres (version commerciale).

NOTE IMPORTANTE : la configuration RESTful Mini Sentinel ne traite aucun type de transaction à aucun moment, c'est seulement le service qui met en forme "l'information" afin que les nœuds puissent effectuer le traitement des transactions correctement et dans un temps planifié et synchronisé pour tous les nœuds.

L'installation du service RESTful est basée sur une base de données SQLite. Cette installation se fera dans l'environnement Windows pour des raisons pratiques, mais ce modèle peut être facilement reproduit dans un système d'exploitation de type Linux.

Pour ceux qui souhaitent examiner les performances et le support des requêtes et insertions SQLite, veuillez vous référer à ces tests de performance :

<https://www.sami-lehtinen.net/blog/sqlite3-performance-testing>

<https://sqlite.org/src4/doc/trunk/www/lsmperf.wiki>

Installation du réseau de sauvegarde RESTful Mini Sentinel. Nous installerons et configurerons ce service sur un ordinateur Windows 10, mais le processus a également été installé facilement dans un environnement Linux Ubuntu 18.09.

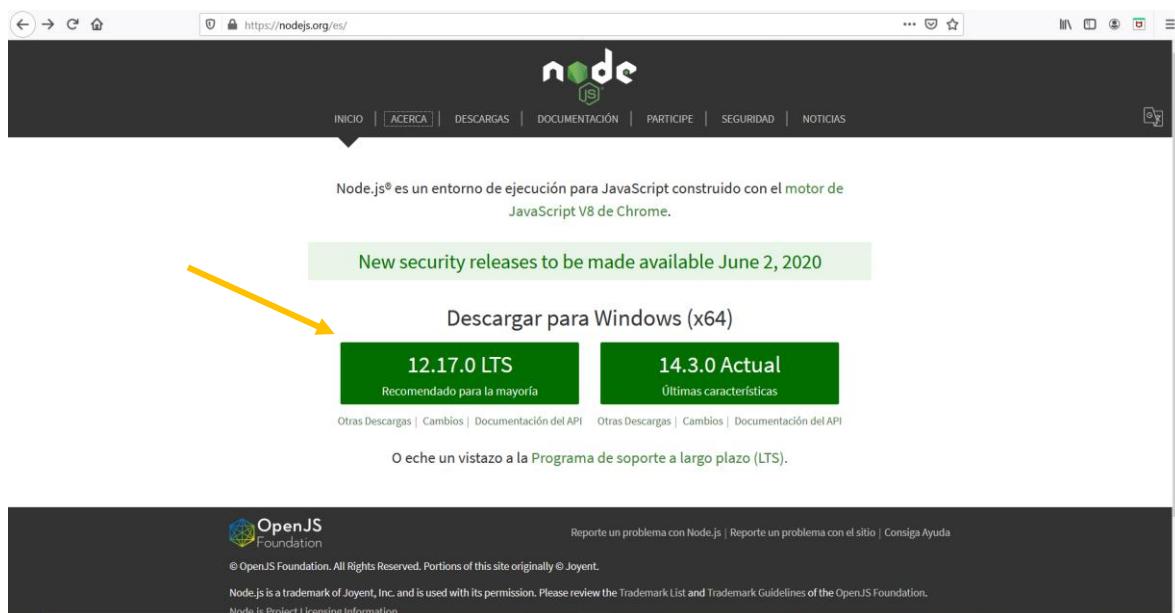
Exigences :

- ✓ Serveur de base de données SQLite en mode RESTful (<https://github.com/olsonpm/sqlite-to-rest>).
- ✓ Connecteur sentinelle SQLite-Redis
- ✓ Serveur de base de données Redis en mode maître-esclave (<https://redis.io/topics/replication>)

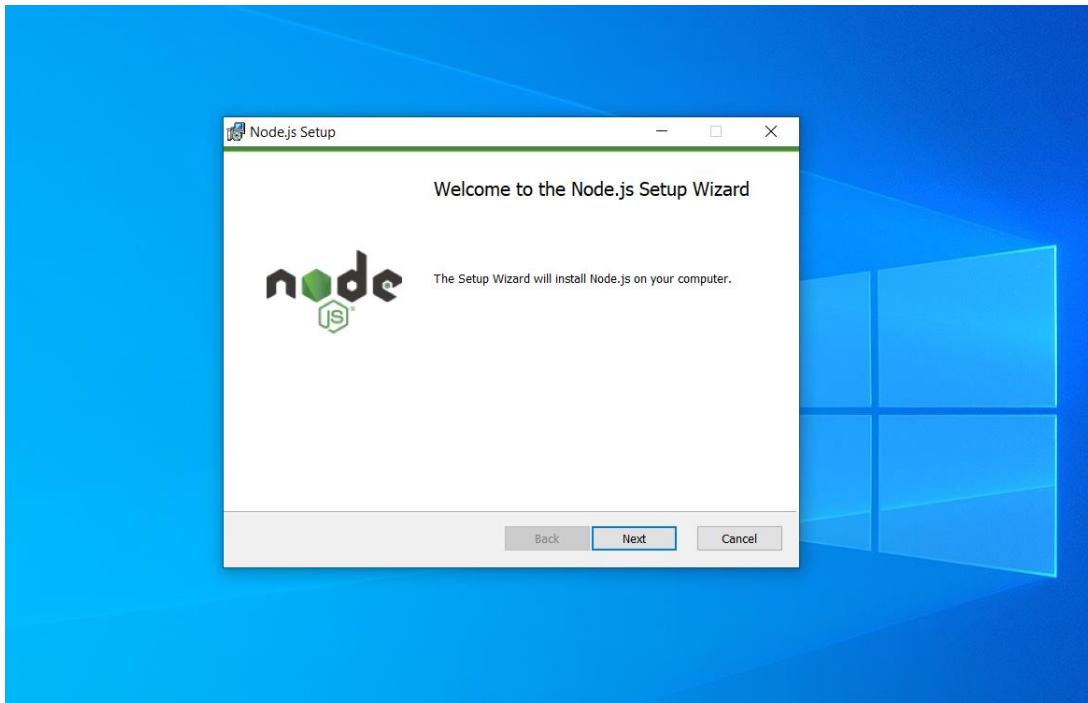
Installation du serveur de base de données SQLite en mode RESTful

Pour l'installation, nous devons commencer par les logiciels suivants : Nodejs, sqlite3 et Git Bash pour Windows.

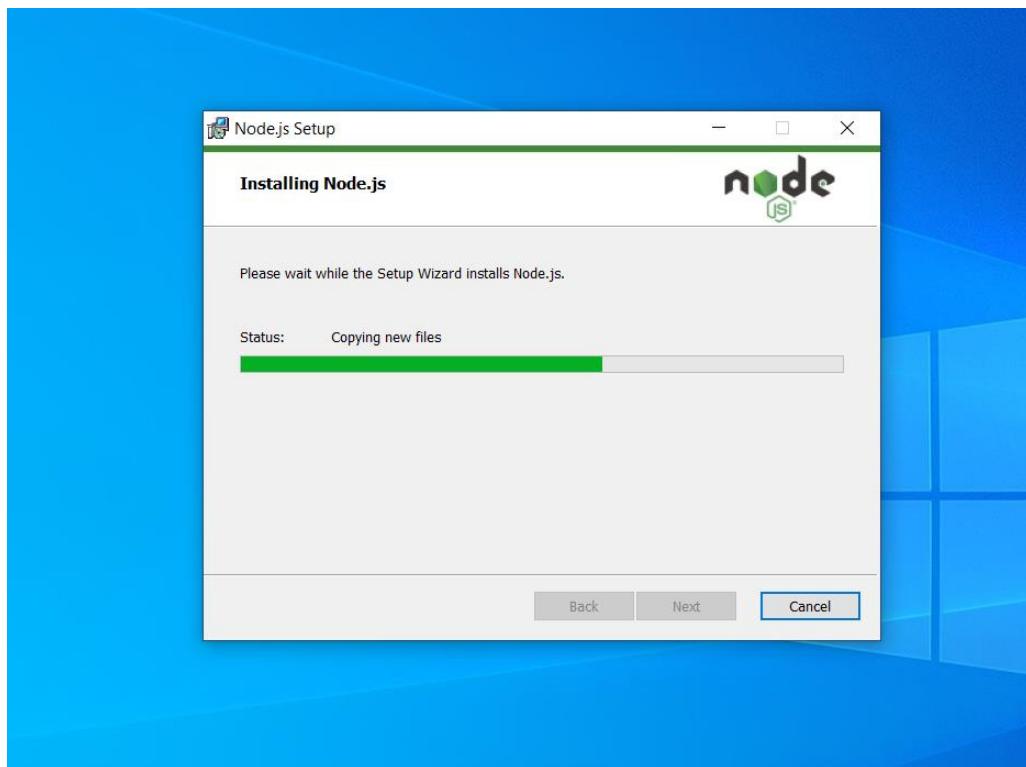
Pour obtenir les Nodejs, nous avons consulté leur site officiel <https://nodejs.org/es/> et avons choisi la version "Recommandé pour le plus grand nombre".

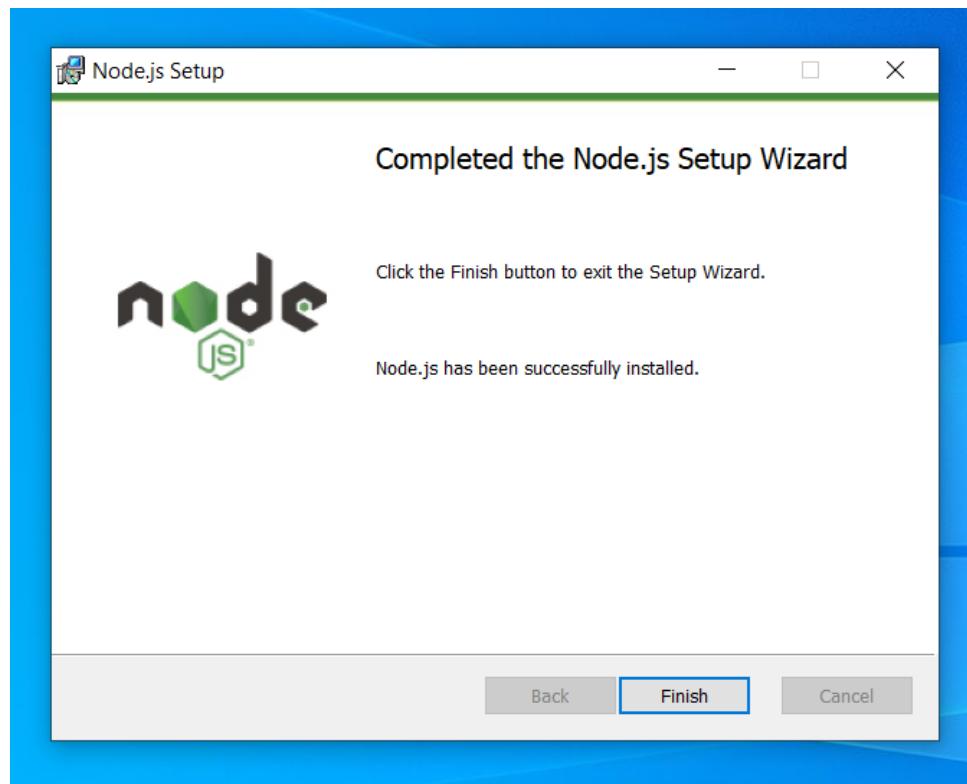


Après avoir téléchargé le fichier avec l'extension .**msi**, nous double-cliquons pour l'installer.



Nous effectuons l'installation par défaut, il suffit de cliquer sur "Suivant" sans choisir les options supplémentaires que vous demandez.



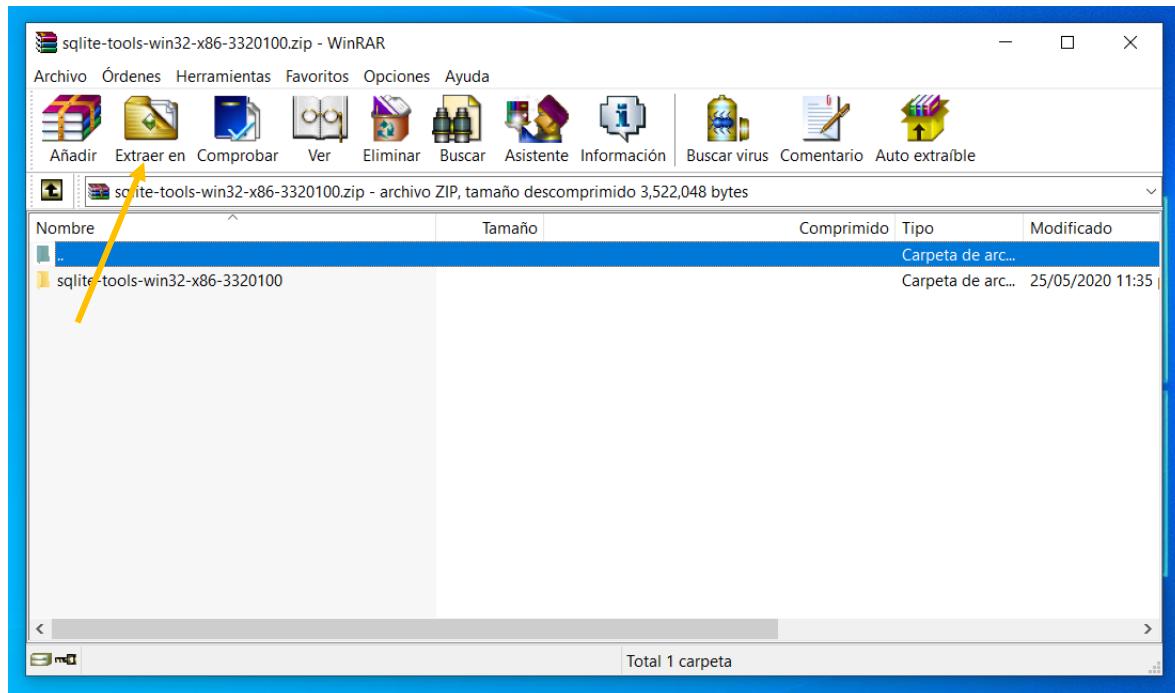


Depuis que nous avons terminé l'installation de Nodejs, nous continuons avec l'installation de la base de données SQLite.

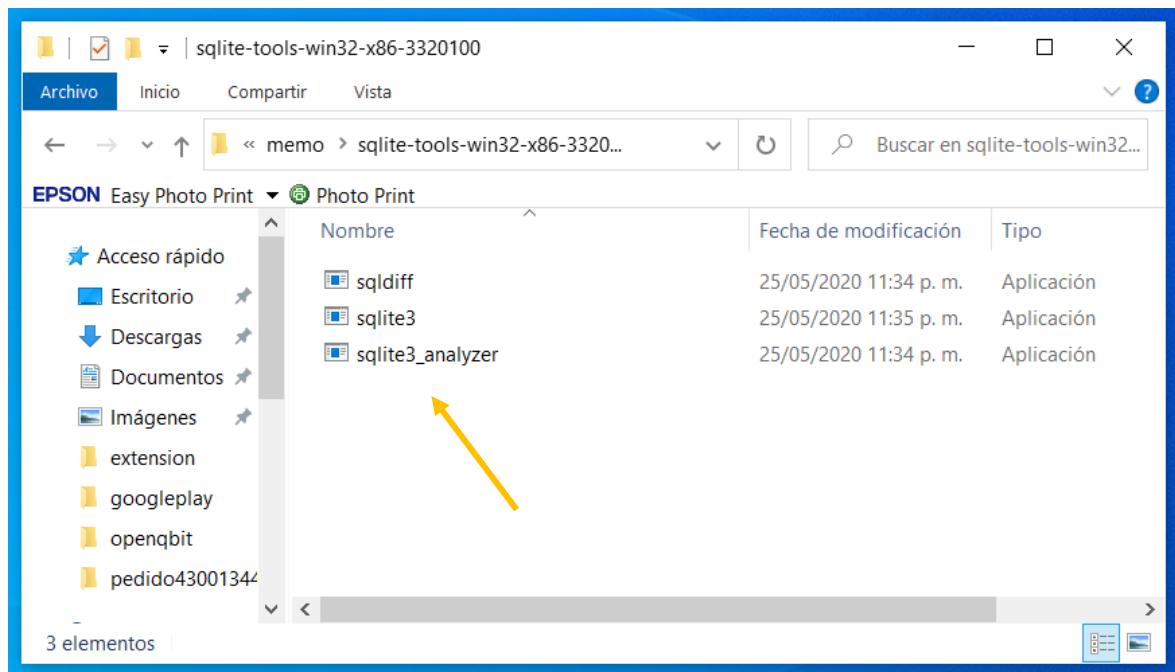
Nous nous rendons sur leur site officiel, <https://www.sqlite.org/index.html>, et cliquons sur la partie où vous "téléchargez".

A screenshot of the SQLite.org website. The header shows navigation links and a URL bar with https://www.sqlite.org/index.html. A yellow arrow points to the "Download" link in the top navigation bar. Below the header, there's a large "SQLite" logo with a feather icon. The main content area has a heading "What Is SQLite?". Text describes SQLite as a C-language library with various features. Another yellow arrow points to the "Download" link in the footer navigation bar. The footer also includes links for "Version 3.32.1 (2020-05-25)", "Download", and "Prior Releases".

Ensuite, nous choisissons l'option où il est indiqué "Binaires précompilés pour Windows" et nous cliquons sur la version du logiciel "**sqlite-tools-win32-x86-3320100.zip**" ; lorsque le téléchargement est terminé, nous procédons à la décompression du fichier de manière simple, nous ouvrons le dossier où le fichier a été téléchargé et nous double-cliquons sur le fichier pour le décompresser.

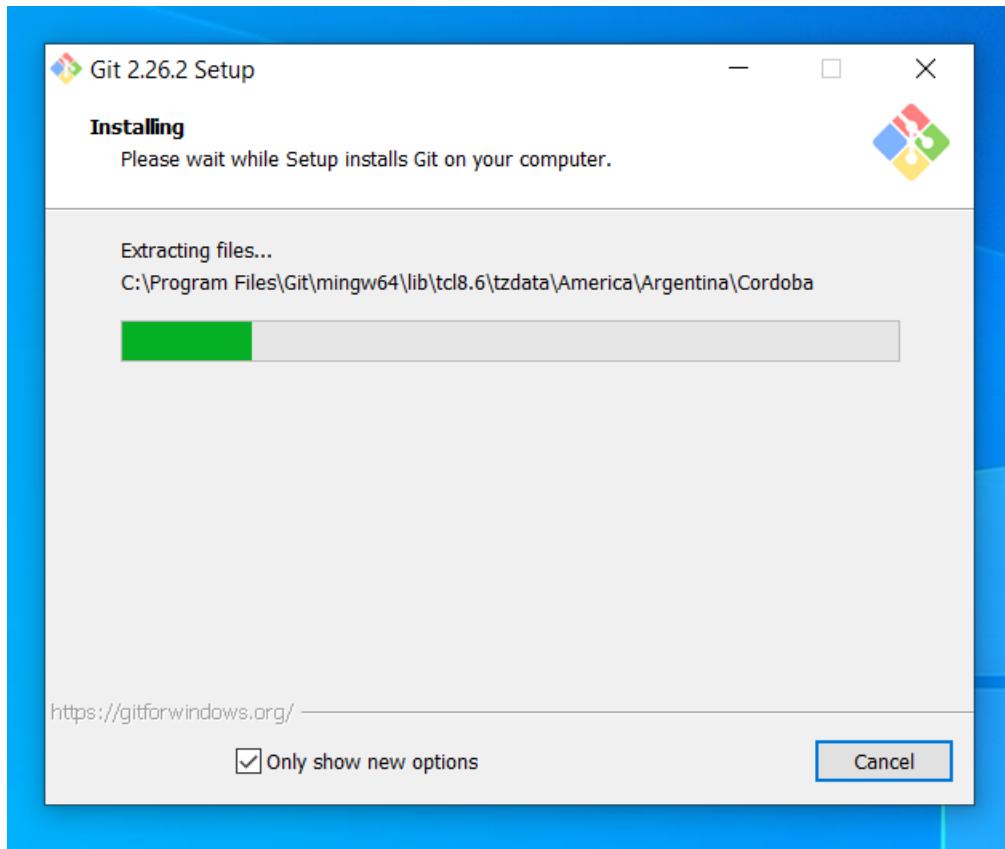


Après décompression, nous aurons trois fichiers, qui sont notre environnement pour créer notre base de données SQLite que nous utiliserons plus tard.



Nous commencerons par installer Git Bash, un émulateur de terminal de type Linux qui nous aidera à faire fonctionner correctement le service de sauvegarde RESTful.

Nous le téléchargerons à partir de leur site officiel, <https://gitforwindows.org/> et procéderons à son installation avec les options par défaut qu'il indique.



Maintenant que nous avons installé nodejs, sqlite et Git Bash sur notre machine Windows 10, nous procédons à l'installation de l'environnement qui supportera notre base de données SQLite en mode RESTful.

Au moment du téléchargement du logiciel qui donnera les fonctionnalités de RESTful à SQLite. Pour cela nous devons nous rendre sur le site suivant, <https://github.com/olsonpm/sqlite-to-rest> dans lequel nous cliquerons sur le bouton vert de droite "Cloner ou télécharger" et choisir l'option "Télécharger ZIP" cela permettra de télécharger le logiciel dans notre ordinateur.

Mini BlocklyChain - DIY - "Do It Yourself

No description or website provided.

automatic-api

Branch: dev New pull request

Find file Clone or download ▾

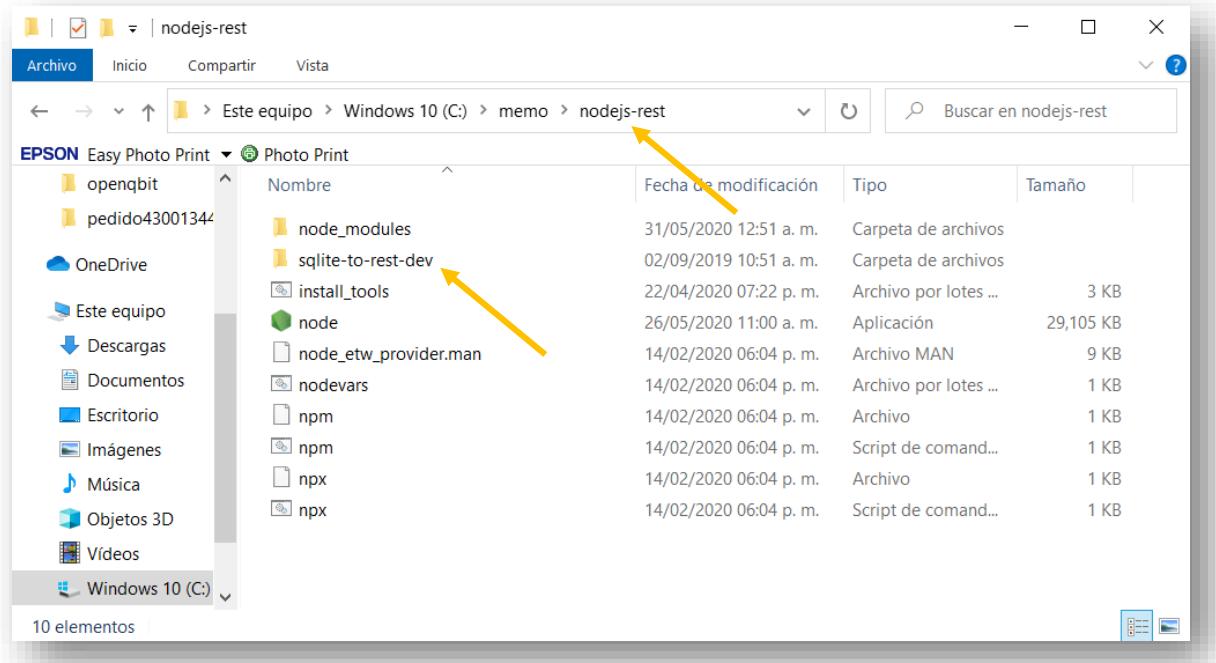
Clone with HTTPS ⓘ
Use Git or checkout with SVN using the web URL.
<https://github.com/olsonpm/sqlite-to-rest>

Open in Desktop Download ZIP

File	Description	Last Commit
bin	add prettier and eslint	9 months ago
cli/commands	add prettier and eslint	9 months ago
docs	add prettier and eslint	9 months ago
lib	add prettier and eslint	9 months ago
tests	add prettier and eslint	9 months ago
.gitignore	add prettier and eslint	9 months ago

Après l'avoir téléchargé sur notre ordinateur, nous le décompressons dans le répertoire où le dossier où nous avons installé le programme **nodejs** et nous obtenons un répertoire portant le nom "**sqlite-to-rest-dev**".

NOTE : Il est important que le répertoire **sqlite-to-rest-dev** se trouve dans le répertoire où **nodejs** a été installé.



Une fois que tout est installé, nous procédons à la configuration de la base de données SQLite que nous utiliserons pour stocker les transactions des nœuds dans l'environnement de sauvegarde RESTful.

Conception des tableaux et structure des données. Commandes à exécuter en ligne dans la liste des commandes CMD

```
sqlite3 op.sqlite3 "CREATE TABLE trans (id entier clé primaire, addro, addrd, valeur) ;";
```

```
sqlite3 op.sqlite3 "CREER TABLE signe (id clé primaire entière, trans_id références trans (id), signe, dièse) ;";
```

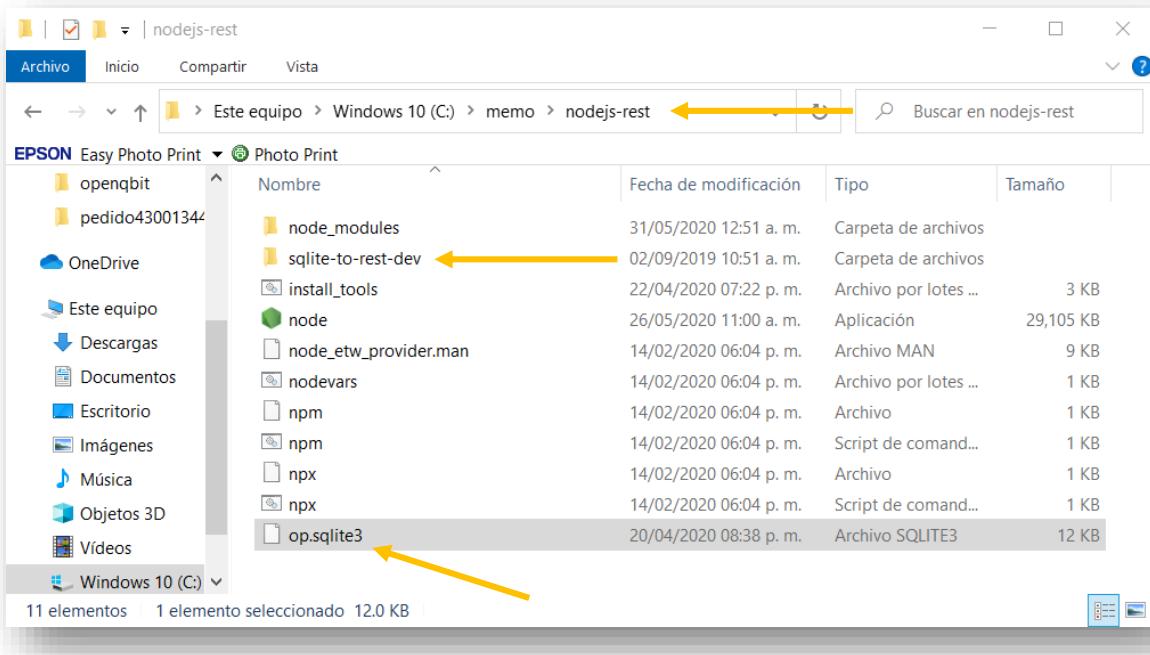
```
C:\memo\sqlite-tools-win32-x86-3320100>sqlite3 op.sqlite3 "CREATE TABLE trans(id integer primary key, addro, addrd, valeur) ;"
C:\memo\sqlite-tools-win32-x86-3320100>sqlite3 op.sqlite3 "CREATE TABLE signe(id integer primary key, trans_id references trans (id), sign, hash);"
C:\memo\sqlite-tools-win32-x86-3320100>dir
El volumen de la unidad C es Windows 10
El número de serie del volumen es: E019-5C05

Directorio de C:\memo\sqlite-tools-win32-x86-3320100

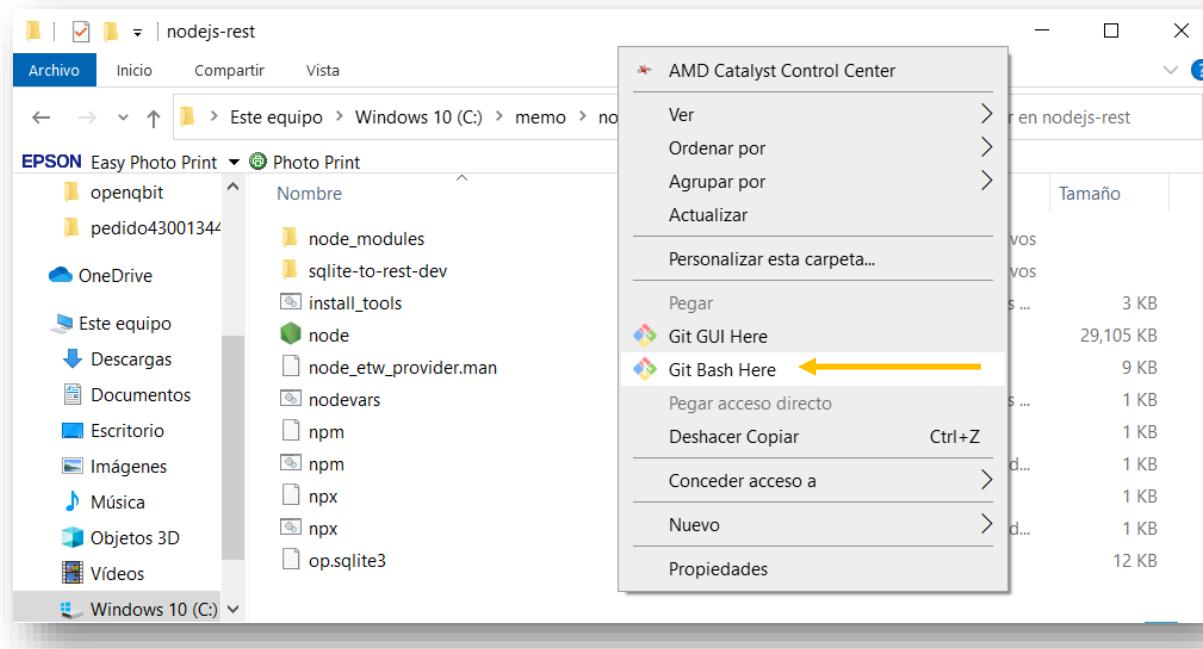
31/05/2020 02:29 a. m.    <DIR>        .
31/05/2020 02:29 a. m.    <DIR>        ..
31/05/2020 02:29 a. m.        12,288 op.sqlite3
25/05/2020 11:34 p. m.      516,608 sqldiff.exe
25/05/2020 11:35 p. m.      972,800 sqlite3.exe
25/05/2020 11:34 p. m.     2,032,640 sqlite3_analyzer.exe
                           4 archivos       3,534,336 bytes
                           2 dirs   137,272,078,336 bytes libres

C:\memo\sqlite-tools-win32-x86-3320100>
```

Après avoir créé la base de données op.sqlite3, nous devons faire une copie dans le répertoire où le **nodejs** a été installé. Dans le répertoire **nodejs** doit également se trouver la copie de "**sqlite-to-rest-dev**".



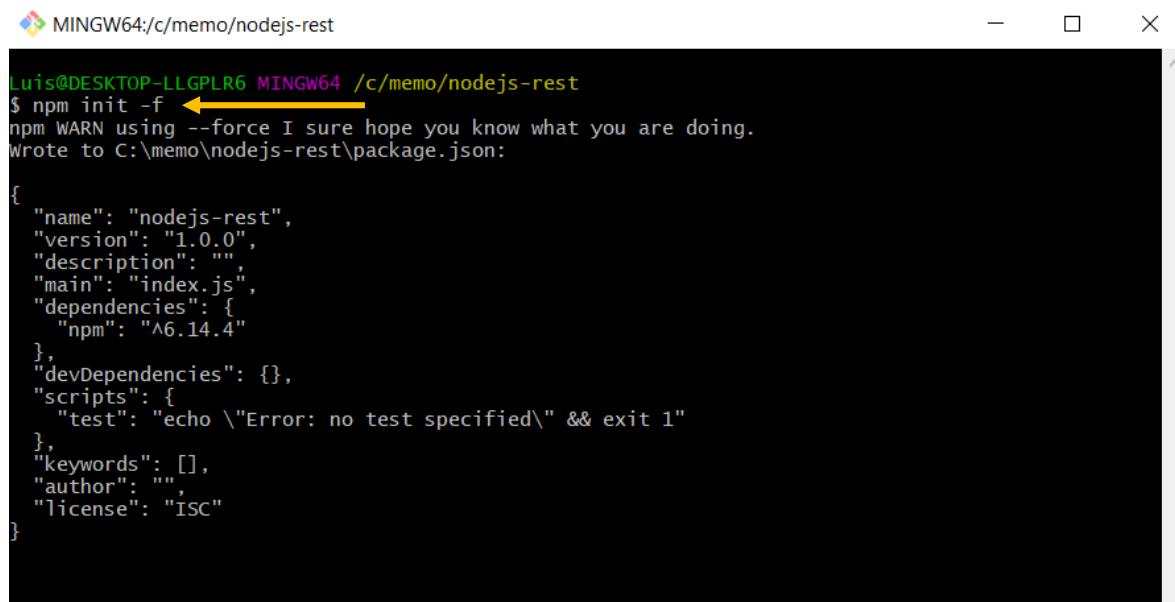
Nous nous plaçons dans le dossier où se trouve l'installation de **nodejs** et où le logiciel "**sqlite-to-rest-dev**" doit également se trouver. Pointez sur le dossier avec le pointeur et



faites un clic droit pour afficher le menu et choisissez l'endroit où il est écrit "**Git Bash**" pour ouvrir un terminal.

Dans le nouveau terminal Git Bash ouvert, nous exécutons les commandes suivantes :

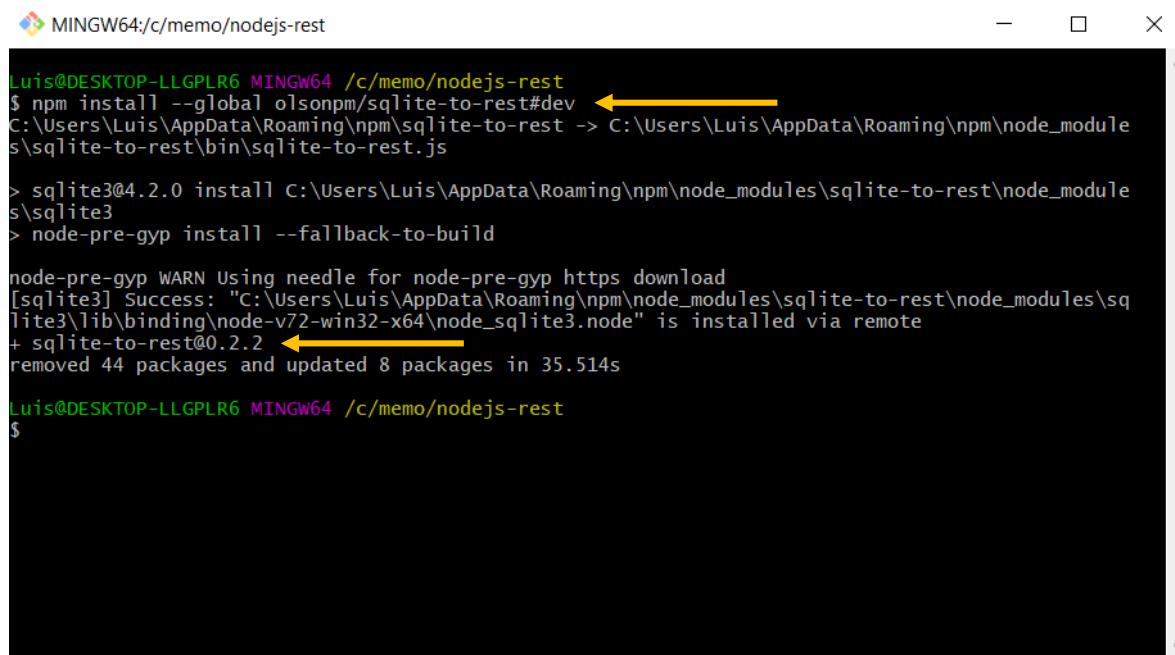
\$ npm init -f



```
Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$ npm init -f
npm WARN using --force I sure hope you know what you are doing.
Wrote to C:\memo\nodejs-rest\package.json:

{
  "name": "nodejs-rest",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "npm": "^6.14.4"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

\$ npm install --global olsonpm/sqlite-to-rest#dev



```
Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$ npm install --global olsonpm/sqlite-to-rest#dev
C:\Users\Luis\AppData\Roaming\npm\sqlite-to-rest -> C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\bin\sqlite-to-rest.js

> sqlite3@4.2.0 install C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\node_modules\sqlite3
> node-pre-gyp install --fallback-to-build

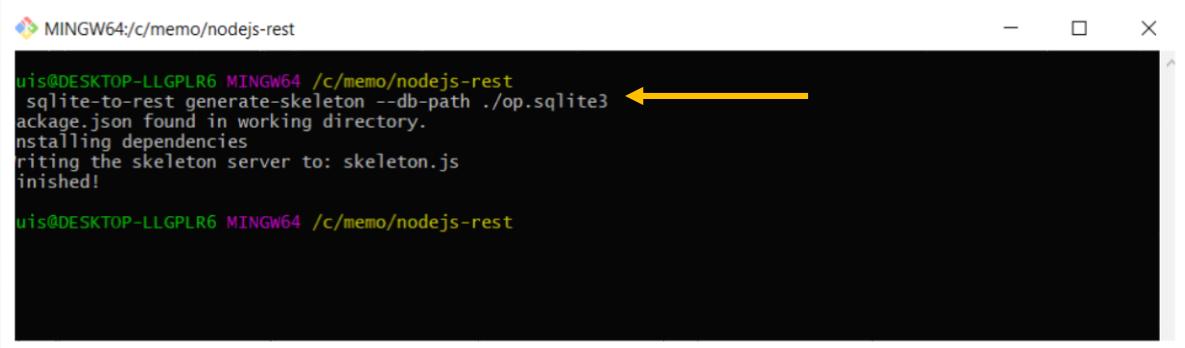
node-pre-gyp WARN Using needle for node-pre-gyp https download
[sqlite3] Success: "C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\node_modules\sqlite3\lib\binding\node-v72-win32-x64\node_sqlite3.node" is installed via remote
+ sqlite-to-rest@0.2.2
removed 44 packages and updated 8 packages in 35.514s

Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$
```

Après l'exécution de la commande, l'installation du paquet "**sqlite-to-rest**" apparaîtra.

Nous avons généré l'environnement RESTful pour SQLite avec la base **op.sqlite3** que nous avons créée auparavant.

Nous exécutons la commande : **\$ sqlite-to-rest generate-skeleton --db-path ./op.sqlite3**



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
sqlite-to-rest generate-skeleton --db-path ./op.sqlite3 ←
package.json found in working directory.
Installing dependencies
Writing the skeleton server to: skeleton.js
finished!

uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
```

Nous avons lancé le service RESTful SQLite sur le port 8085 par défaut. Nous exécutons la commande suivante : **\$ node skeleton.js**



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
node skeleton.js ←
listening on port: 8085
```

Nous avons testé le service RESTful en ajoutant des données et en interrogeant des données.



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s -H "Content-Type: application/json" -d '{"addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/transactions
"id": 6, "addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999" ←
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s http://localhost:8085/transactions ←
"id": 1, "addr": "CO", "addrd": "Boulder", "value": "Avery"}
"id": 2, "addr": "WI", "addrd": "New Glarus", "value": "New Glarus"}
"id": 3, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id": 4, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id": 5, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id": 6, "addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}
```

Pour tester le service SQLite RESTful, nous utilisons les commandes suivantes :

Pour insérer des données dans la table trans qui se trouve dans la base de données op.sqlite3 :

```
$ curl -s -H "Content-Type : application/json" -d '{"addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/trans
```

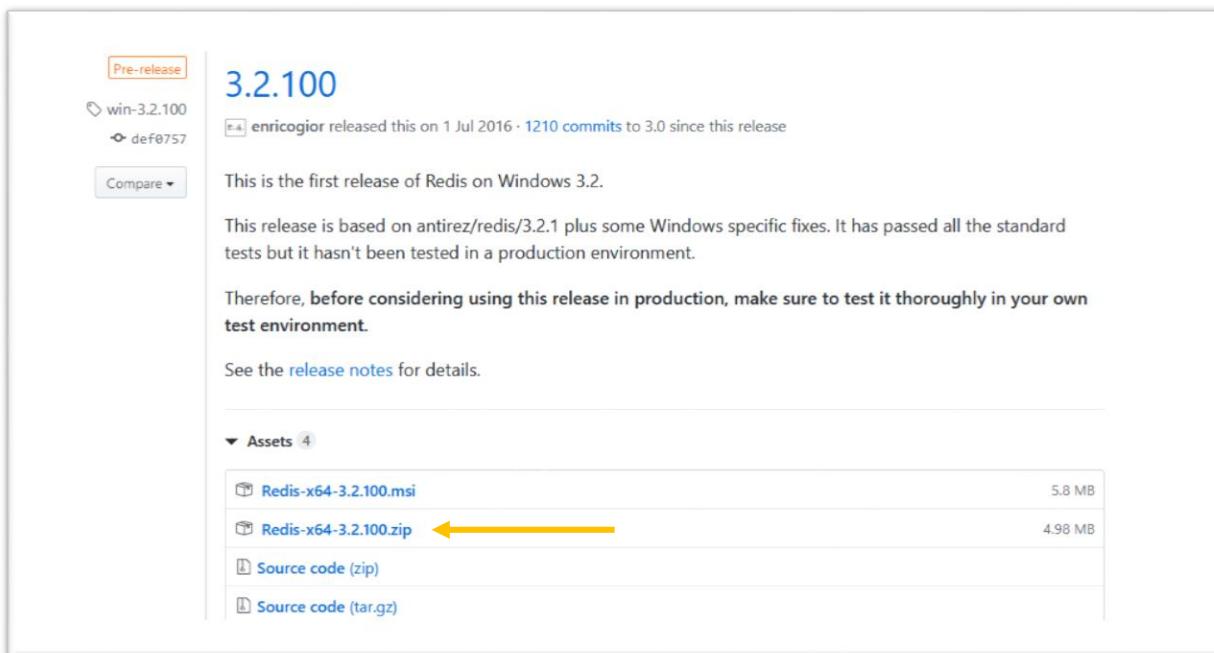
Pour interroger toutes les données de la table trans :

```
$ curl -s -H 'range : rows=0-2' http://localhost:8085/trans
```

Pour savoir comment utiliser en détail tous les services activés par RESTful (interrogation, insertion, mise à jour et/ou suppression de données), consultez l'**annexe "Commandes SQLite GET/POST reposantes"**.

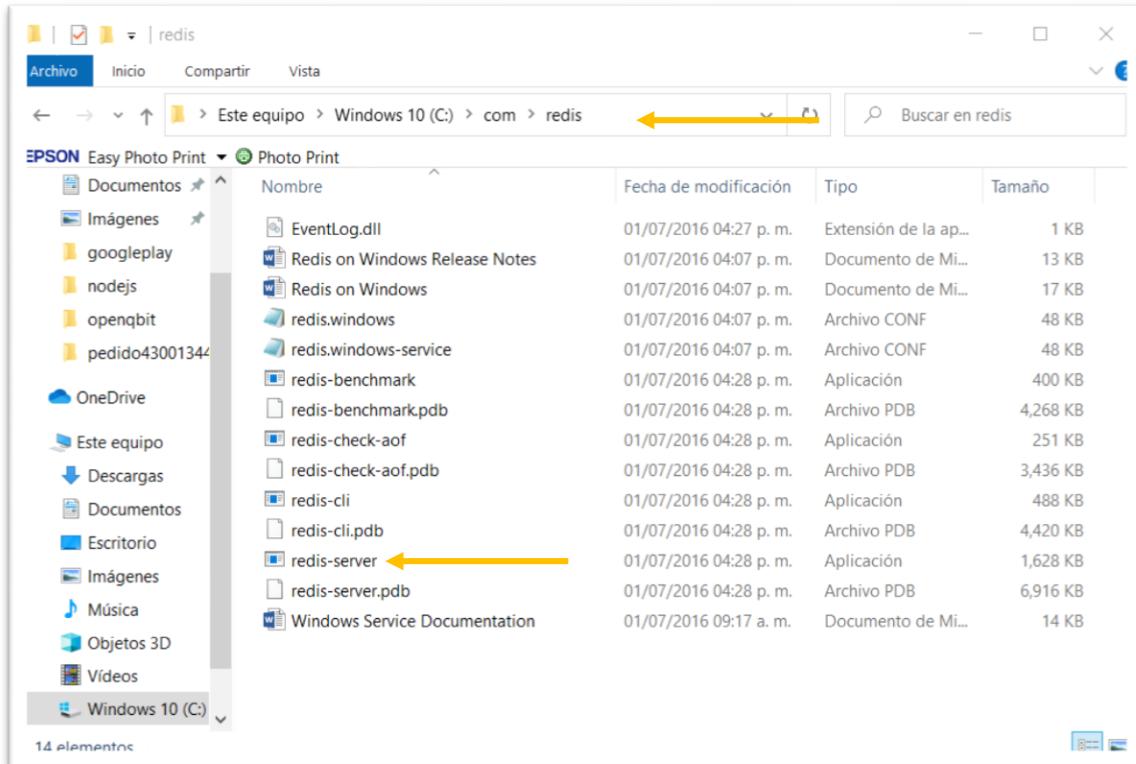
NOTE : Dans l'installation précédente, toutes les requêtes sont effectuées dans l'URL avec l'adresse "localhost", mais lorsque le serveur est exposé avec une IP publique (internet) ou privée (Wifi), il fonctionnera sans aucun problème. Nous le testerons lors des tests de communication entre le service SQLite RESTful et les nœuds du réseau de communication qui forment la Mini BlocklyChain.

Installation de la base de données redis pour le service de réseau de sauvegarde, pour télécharger le logiciel redis pour Windows il faut aller sur le site, <https://github.com/microsoftarchive/redis/releases/tag/win-3.2.100> et choisir le paquet ZIP.



Pour localiser l'installation nous allons créer un répertoire appelé "redis" dans Windows et télécharger le fichier avec une extension ZIP, nous le décompressons dans le répertoire créé précédemment, nous avons déjà l'installation redis terminée.

Nous testons l'installation en faisant tourner le serveur en double-cliquant sur la commande "redis-server".



Après avoir exécuté la commande "redis-server", nous verrons le serveur fonctionner dans un terminal de commande CMD Windows sur le port 6379 par défaut :

```
C:\com\redis>redis-server.exe
[10192] 31 May 13:50:45.340 # Warning: no config file specified, using the default config. In order to specify a config
file use C:\com\redis\redis-server.exe /path/to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379 ←
PID: 10192

http://redis.io

[10192] 31 May 13:50:45.355 # Server started, Redis version 3.2.100
[10192] 31 May 13:50:45.355 * The server is now ready to accept connections on port 6379
```

Dans ce test, nous avons une version de Redis 3.2.100 pour Windows 10, dans le cas du système d'exploitation Linux, nous avons la version 6.0.4 (sortie en mai 2020) ; cependant, pour notre configuration Mini BlocklyChain, la version Windows a suffisamment de fonctionnalités dont nous avons besoin.

Pour arrêter l'exécution nous faisons la combinaison de touches Ctrl + C. Nous procédon à la configuration de la base de données Redis 3.2.100 pour Windows 10 avec une configuration entre Redis et les nœuds du réseau de communication Mini SQLSync, le type **Maître (Master)-Esclave (Slave)** est distribué comme suit :

Le maître est le serveur Redis pour Windows 10 que nous sommes en train de configurer et qui répliquera les données en temps réel et synchronisées avec les mêmes informations vers tous les nœuds (téléphones mobiles). Ces nœuds auront un serveur Redis installé en mode **esclave**.

À ce stade, nous commençons à voir comment fonctionne le réseau de sauvegarde que nous installons et configurons. Cette configuration nous servira à communiquer avec tous les nœuds en temps réel, elle nous aidera à communiquer avec tous les nœuds du réseau lorsqu'il y a une nouvelle file d'attente de transactions à traiter par les nœuds, dans une égalité dans le transfert d'informations à tous les nœuds de sorte que tout nœud a la même probabilité de pouvoir traiter les informations de la "file d'attente de transactions".

Nous commençons la configuration du connecteur **SQLite-Redis Sentinel**.

Ce connecteur est un programme développé en langage Java et comme son nom l'indique, il relie les bases de données SQLite et Redis (**Master**).

La fonction du connecteur est de transmettre les informations (transactions) de SQLite à Redis (**Master**) et cela envoie la "file d'attente des transactions" aux nœuds (**Slaves**).

Pour plus de détails sur le code Java du connecteur **SQLite-Redis Sentinel**, voir l'annexe "Connecteur de code Java SQLite-Redis".

Configuration du fichier **redis.conf** de la base de données Redis (**Master**) pour Windows 10.

Ajoutez les modifications ou directives suivantes au fichier, enregistrez les modifications et démarrez le serveur Redis

Commencez par trouver le paramètre **tcp-keepalive** et réglez-le à 60 secondes comme suggéré par les commentaires. Cela aidera Redis à détecter les problèmes de réseau ou de service :

tcp-keepalive 60

Trouvez la directive **requirepass** et configuez-la avec une phrase de passe forte. Si votre trafic Redis doit être protégé contre les tiers, cela permet de s'authentifier auprès de Redis. Comme Redis est rapide et ne note pas les tentatives de phrases de passe limites, choisissez une phrase de passe forte et complexe pour vous protéger contre les tentatives de force brute :

requirepass type_your_network_master_password

exemple :

requirepass FPqwedsLMdf76ass7asddf2g45vBN8ty99

Enfin, il y a quelques paramètres optionnels que vous voudrez peut-être ajuster en fonction de votre scénario d'utilisation.

Si vous ne voulez pas que Redis mette automatiquement les clés les plus anciennes et les moins utilisées lorsqu'il se remplit, vous pouvez désactiver le retrait automatique des clés :

maxmemory-policy noevasion

Pour des garanties de durabilité accrues, vous pouvez activer la persistance des fichiers par le biais d'un add-on. Cela permettra de minimiser les pertes de données en cas de défaillance du système, au détriment de fichiers plus volumineux et de performances légèrement plus lentes :

appendice oui

nom de l'annexe "redis-staging-ao.aof

Procédez à l'enregistrement des modifications et redémarrez le service Redis pour Windows 10, arrêtez avec les touches Ctrl + C et relancez la ligne de commande CMD de Windows :

C :\redis_redis_directory redis_server

Dans notre exemple, nous pouvons voir que nous avons un nœud (**esclave**) connecté.

```

:\\com\\redis> redis-server redis.conf
1:C 31 May 2020 23:44:56.633 # o000o000o000 Redis is starting o000o000o000
1:C 31 May 2020 23:44:56.634 # Redis version=5.0.7, bits=64, commit=00000000, modified=0,
id=51, just started
1:C 31 May 2020 23:44:56.634 # Configuration loaded
1:M 31 May 2020 23:44:56.635 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 51

http://redis.io

1:M 31 May 2020 23:44:56.648 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.51:May 31 May 2020 23:44:56.648 # Server initialized
1:M 31 May 2020 23:44:56.649 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 31 May 2020 23:44:56.669 * DB loaded from append only file: 0.020 seconds
1:M 31 May 2020 23:44:56.669 * Ready to accept connections
1:M 31 May 2020 23:49:57.013 * 10 changes in 300 seconds. Saving...
1:M 31 May 2020 23:49:57.031 * Background saving started by pid 82
2:C 31 May 2020 23:49:57.052 * DB saved on disk
1:M 31 May 2020 23:49:57.133 * Background saving terminated with success
1:M 31 May 2020 23:50:24.600 * Replica 192.168.1.68:6379 asks for synchronization
1:M 31 May 2020 23:50:24.602 * Full resync requested by replica 192.168.1.68:6379
1:M 31 May 2020 23:50:24.602 * Starting BGSAVE for SYNC with target: disk
1:M 31 May 2020 23:50:24.619 * Background saving started by pid 83
3:C 31 May 2020 23:50:24.642 * DB saved on disk
1:M 31 May 2020 23:50:24.670 * Background saving terminated with success
1:M 31 May 2020 23:50:24.689 * Synchronization with replica 192.168.1.68:6379 succeeded

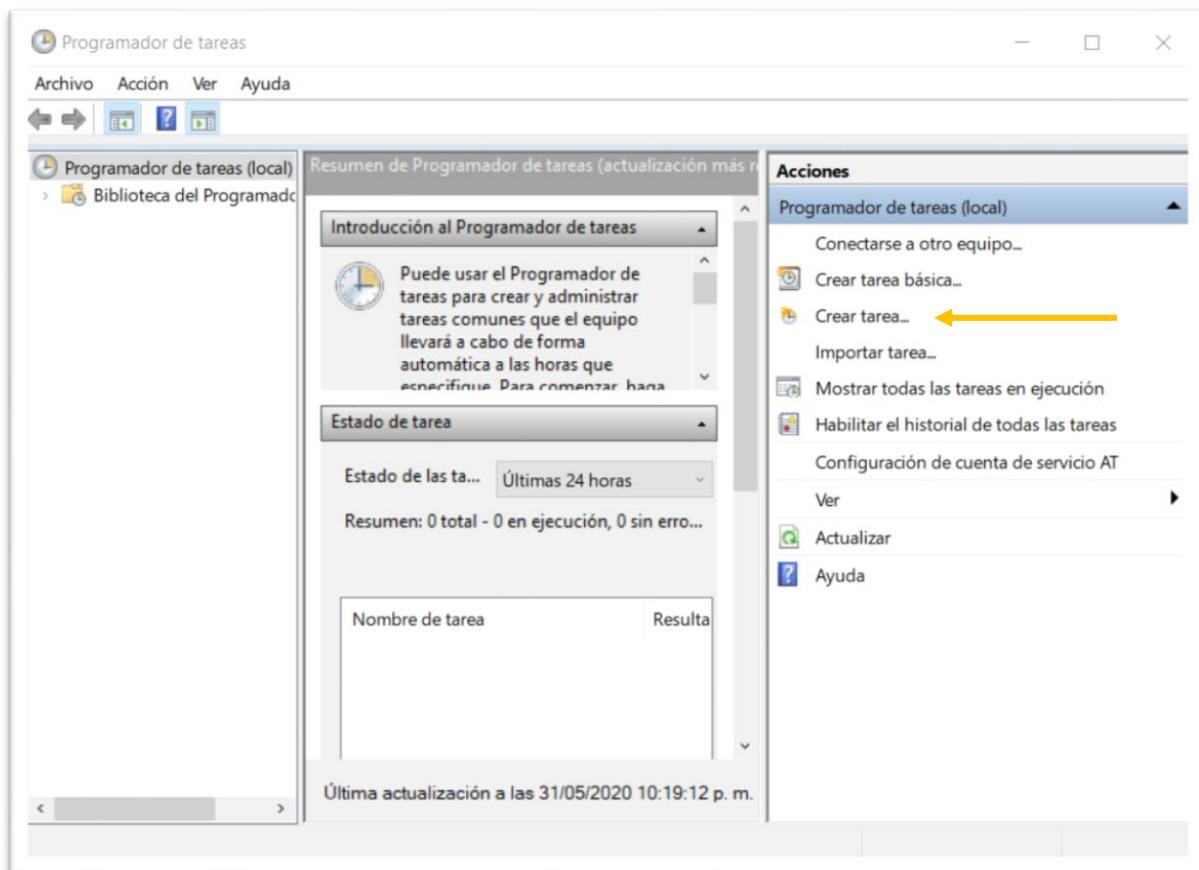
```

Nous pouvons voir que nous avons synchronisé un nœud avec l'IP 192.168.1.68 dans le port par défaut 6379.

Nous devons maintenant programmer dans le système d'exploitation Window 10 la tâche d'exécuter automatiquement le connecteur **SQLite-Redis Sentinel**. Nous le faisons avec l'outil Windows 10, c'est-à-dire que nous l'exécutons en bas à gauche en tapant "Planificateur de tâches".



Nous allons créer une nouvelle tâche "Créer une tâche" où nous inclurons l'exécution du



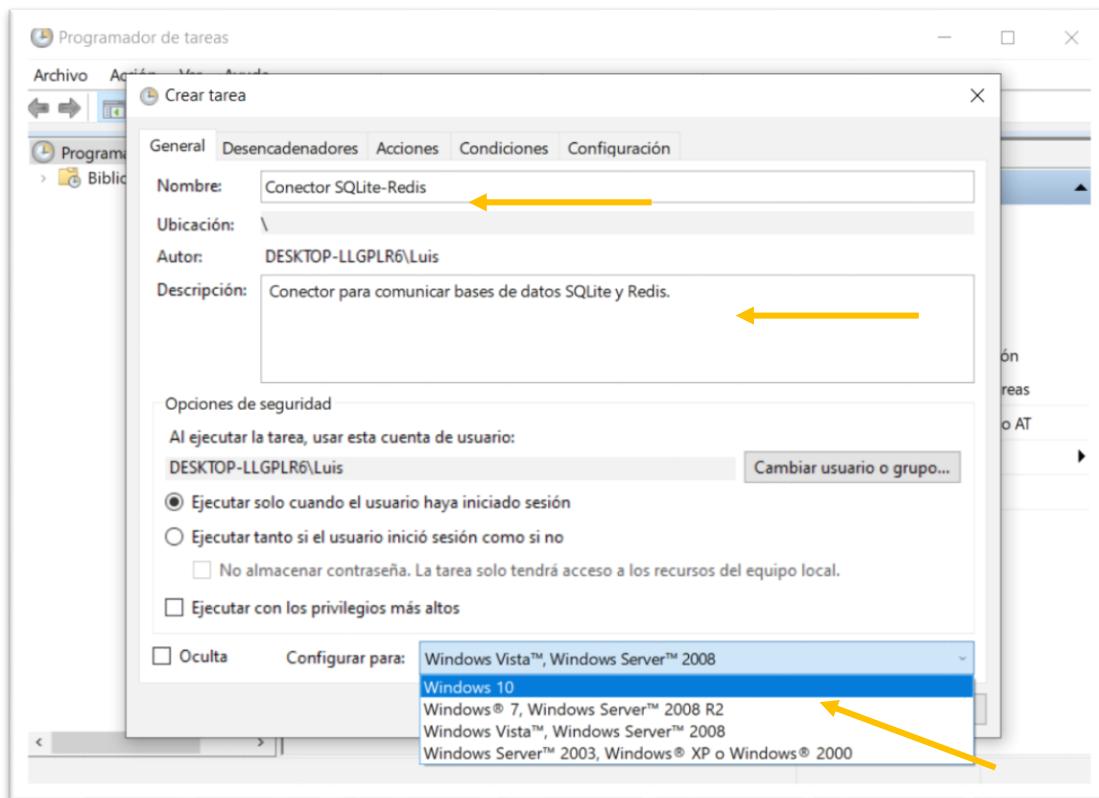
connecteur.

En cliquant sur "Créer une tâche", une fenêtre supplémentaire s'ouvre dans laquelle nous devons indiquer les paramètres suivants dans l'onglet "Général" :

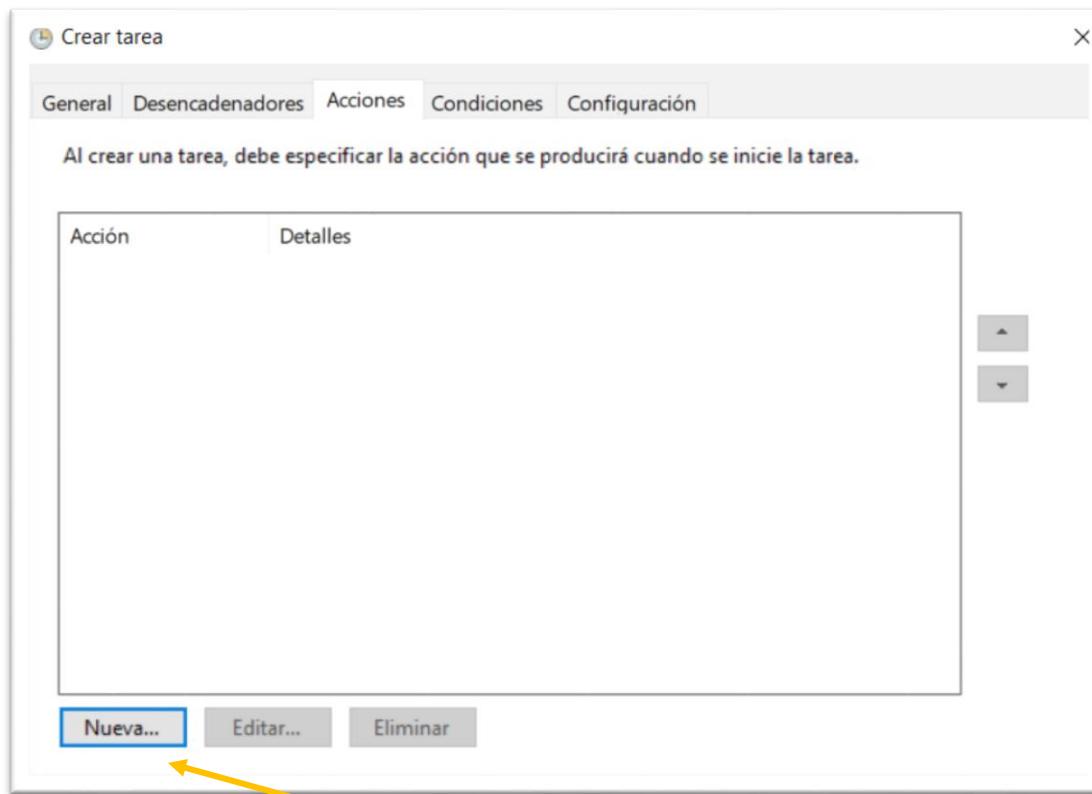
Nom : task_name

Description : facultatif

Configurer pour : select_operating_system_windows_version



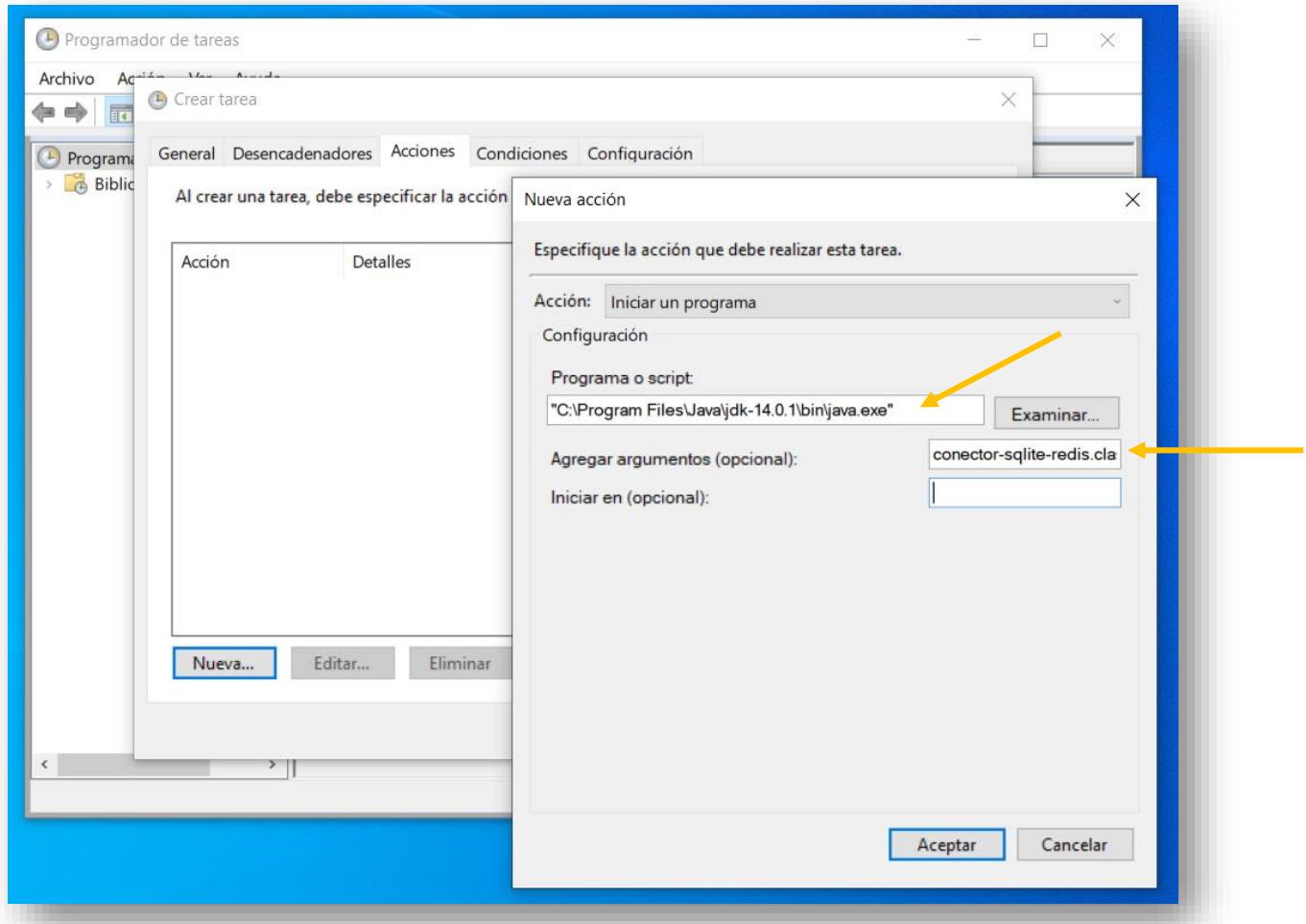
Changez en cliquant sur l'onglet "Actions" et cliquez sur le bouton "Nouveau" :



Nous donnons les paramètres suivants :

Programme ou script : connector_path

Ajout d'arguments : nom du connecteur



Les paramètres ci-dessus peuvent varier en fonction de l'emplacement du connecteur. L'idée principale est l'exécution du programme **connector-sqlite-redis-v1.class** comme il est normalement fait en ligne de commande :

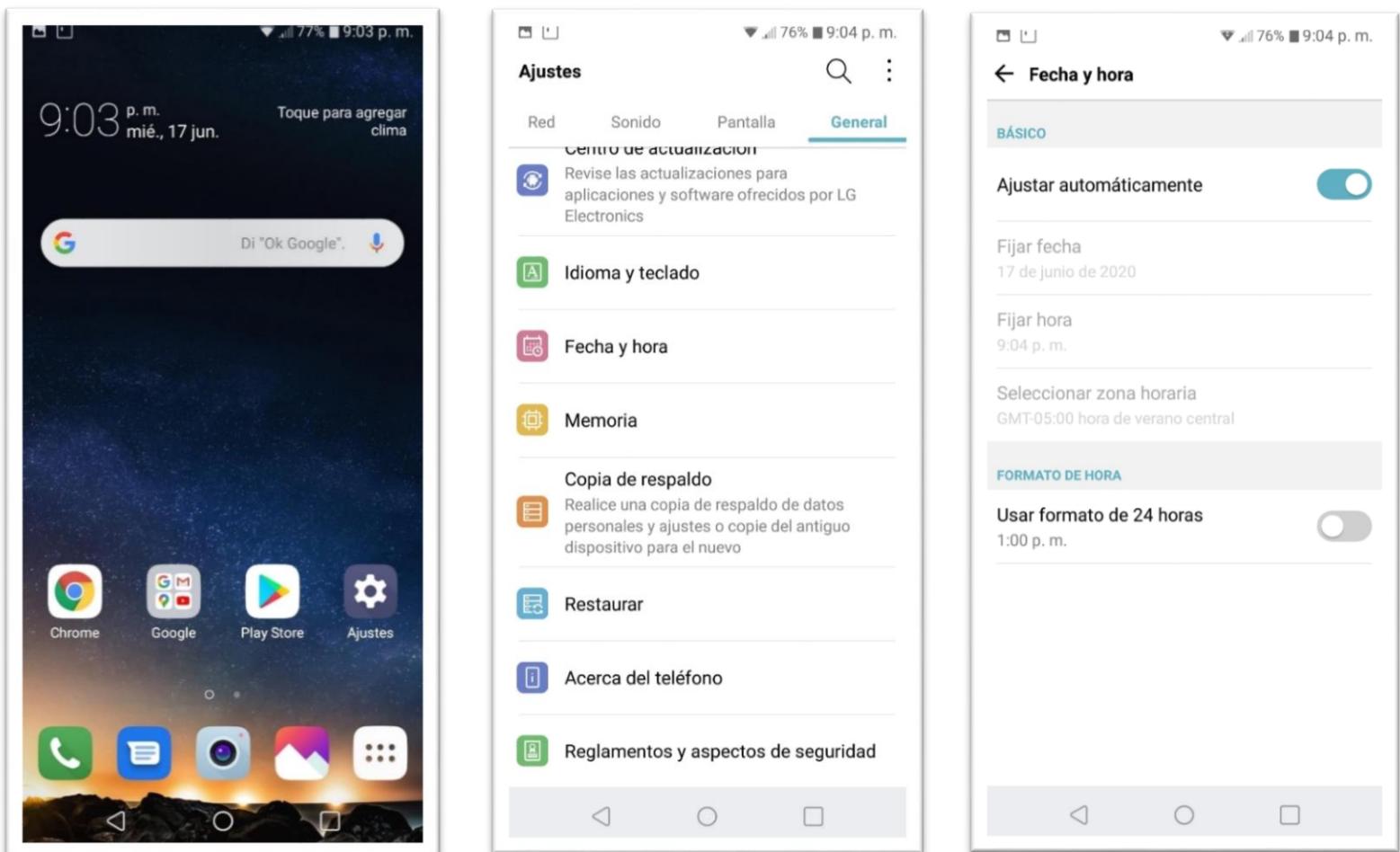
`C:\jdk_directory java connector-sqlite-redis-v1`

Pour finir, nous devons choisir l'onglet "Déclencheurs", dans lequel nous donnerons les paramètres de quand (jour, heure, minutes) nous voulons que la tâche soit exécutée, ces paramètres sont basés sur les règles de gestion que le système Mini BlocklyChain devra être créé.

10. Synchronisation dans les nœuds du système (téléphone mobile) minutes et secondes.

Il est très important que tous les nœuds soient synchronisés principalement dans la partie des minutes et des secondes. Étant donné que, lorsque l'envoi ou la publication de la file d'attente des transactions est effectué à un moment déterminé, tous les nœuds doivent être synchronisés, car cela dépendra du gestionnaire de tâches qui sera établi dans le système local avec l'outil CRON pour être exécuté en même temps dans tous les nœuds, cela signifie que tous les nœuds ont la même probabilité de pouvoir obtenir le droit d'être choisis pour traiter la file d'attente des transactions et de pouvoir générer le nouveau bloc pour l'ajouter à la chaîne de blocs du système. Pour synchroniser les nœuds, nous avons deux options.

La première option de la synchronisation du nœud, nous pourrons le faire de manière simple. Dans notre appareil est par l'option interne qui comprend le système Android, nous devrons aller à la partie de **Paramètres > Date et heure > Ajuster automatiquement**



Avec la configuration précédente, nous aurons synchronisé en minutes et secondes tous les nœuds du système, quel que soit le pays du monde déjà synchronisé, en fonction de chaque zone géographique, et peut-être que l'heure changera, mais en fonction des minutes et secondes qui devraient être synchronisées, cela nous suffit pour exécuter le processus de tâches sur une base programmée avec l'outil cron dans tous les nœuds pour qu'il s'exécute à chaque instant en minutes, c'est-à-dire que nous pouvons créer une tâche dans crontab pour qu'elle s'exécute toutes les 10 ou 30 minutes en fonction de la conception de chaque système.

Cela sera utile lors de l'utilisation du réseau de communication "Peer to Peer". Dans le cas de l'utilisation du réseau de secours, ce processus ne sera pas appliqué puisque la distribution de la file d'attente des transactions se fait dans un modèle client-serveur et que le serveur est celui qui contrôle l'outil cron.

Référence : <https://appinventor.mit.edu/explore/blogs/karen/2016/08.html>

La deuxième option consiste à utiliser une API externe où nous exécuterons la commande Curl via l'extension (**ConnectorSSHClient**).

Le lieu où nous utiliserons les services externes du NTP (Network Time Protocol) est :

<http://worldtimeapi.org/>

Nous allons maintenant étudier un moyen d'obtenir l'heure des serveurs NTP qui sont situés dans le monde entier et qui nous aidera à faire en sorte que tous les nœuds puissent effectuer une requête à un moment donné à la même date et à la même heure.

Exemple d'une requête avec extension (**ConnectorSSHClient**).

```
$ curl "http://worldtimeapi.org/api/timezone/America/Mexico_City"
```

Nous avons fait la connexion avec le terminal Termux :



Nous exécutons le commandement Curl :



Nous devons tenir compte du fait que le résultat de la commande Curl sera au format JSON, quelque chose de similaire au résultat suivant :

```
abbreviation: "CDT"
client_ip: "200.77.16.151"
datetime: "2020-06-18T14:16:57.750466-05:00"
day_of_week: 4
day_of_year: 170
dst: true
dst_from: "2020-04-05T08:00:00+00:00"
dst_offset: 3600
dst_until: "2020-10-25T07:00:00+00:00"
raw_offset: -21600
timezone: "America/Mexico_City"
unixtime: 1592507817
utc_datetime: "2020-06-18T19:16:57.750466+00:00"
utc_offset: "-05:00"
week_number: 25
```

Le résultat pourrait également être au format JSON sans les données formatées en ou JSON sous forme linéaire comme indiqué ci-dessous :

```
{"abbreviation" : "CDT", "client_ip" : "200.77.16.151", "datetime" :
"2020-06-18T14:19:07.216800-05:00", "day_of_week":4, "day_of_year":170,
"dst":true, "dst_from" : "2020-04 05T08:00:00+00:00", "dst_offset":3600,
"dst_until" :"2020-10-25T07:00:00+00:00", "raw_offset": -21600, "timezone"
: "America/Mexico_City", "unixtime":1592507947, "utc_datetime" : "2020-
06-18T19:19:07.216800+00:00", "utc_offset" : "-05:00", "week_number":25}
```

Dans les deux cas, nous devrons filtrer les informations à l'aide d'une extension JSON existante comme "JSONTOOLS" ou utiliser les filtres d'App Inventor dans le traitement de texte pour obtenir uniquement l'heure, le jour ou les secondes en fonction des besoins de chaque système. Après avoir traité le résultat, une comparaison logique peut être faite et, sur la base de cette comparaison, nous pouvons exécuter une tâche déjà programmée avec le service "cron" dont nous verrons plus tard la configuration dans chaque nœud.

Référence de l'extension JSONTOOLS :

<https://thunkableblocks.blogspot.com/2017/07/jsontools-extension.html>

Nous avons maintenant examiné deux options pour synchroniser l'heure des nœuds. Nous continuerons à configurer le service "cron" sur chaque nœud.

Configuration de la programmation automatisée des tâches à exécuter avec le service CRON sur les systèmes Android (nœuds du système).

Nous devons d'abord comprendre comment fonctionne le programmeur automatique.

Le service cron est normalement préinstallé sur tous les systèmes et toutes les tâches à exécuter automatiquement sont programmées dans un fichier appelé crontab.

Nous éditons le fichier crontab avec **\$ crontab -e**, nous utiliserons l'éditeur **vi**, à l'intérieur nous utilisons le format :

```
# m h dom mon dow commande utilisateur
```

où :

- **m** correspond à la minute dans laquelle le script sera exécuté, la valeur va de 0 à 59
- **h** l'heure exacte, le format est de 24 heures, les valeurs vont de 0 à 23, soit 0 minuit.
- **dom fait** référence au jour du mois, par exemple vous pouvez spécifier 15 si vous voulez courir tous les 15
- **dow** signifie le jour de la semaine, il peut être numérique (0 à 7, où 0 et 7 sont le dimanche) ou les 3 premières lettres du jour en anglais : mon, tue, wed, thu, fri, sat, sun.
- **L'utilisateur** définit l'utilisateur qui exécutera la commande, il peut être root, ou un autre utilisateur tant qu'il a les autorisations pour exécuter le script.
- la **commande fait** référence à la commande ou au chemin absolu du script à exécuter, par exemple : /home/user/scripts/update.sh, si vous appelez un script, il doit être exécutable

Pour que les choses soient claires, quelques exemples de tâches cron expliquées :

```
15 10 * * * utilisateur /home/user/scripts/update.sh
```

Vous lancerez le script update.sh à 10h15 tous les jours

```
15 22 * * * utilisateur /home/user/scripts/update.sh
```

Vous lancerez le script update.sh à 22h15 tous les jours

```
00 10 * * 0 root apt-get - et mise à jour de l'utilisateur root
```

Vous effectuerez une mise à jour chaque dimanche à 10 heures.

```
45 10 * * sun root apt-get - et mise à jour
```

L'utilisateur racine effectuera une mise à jour chaque dimanche (soleil) à 10h45.

Nous avons enregistré les modifications dans l'éditeur et nous avons ainsi terminé la configuration du service cron. Si le cron n'est pas installé dans le système, vous pouvez le faire avec la commande suivante :

\$ apt install cron

2. Installation et configuration des nœuds de réseau - Téléphones mobiles.

Commençons par le réseau de communication pour les nœuds qui utiliseront la Mini BlocklyChain.

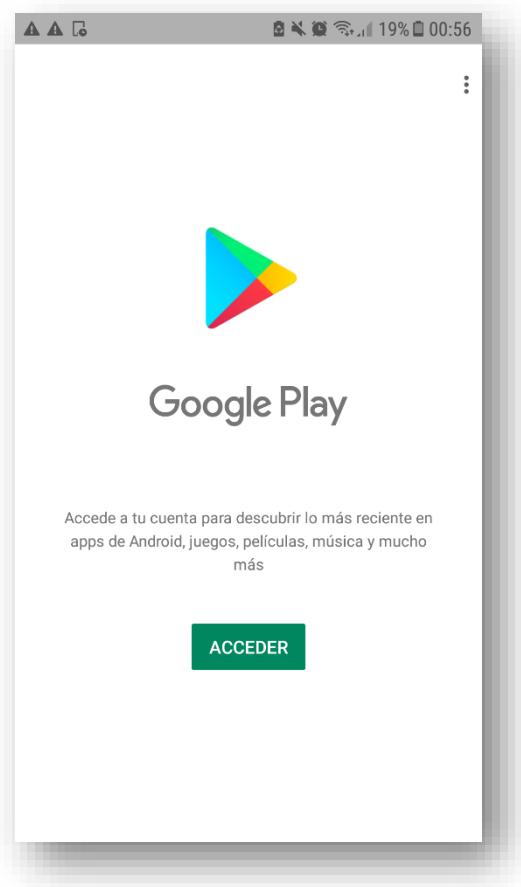
Tout d'abord, nous avons besoin d'un environnement Linux. Comme tout système Android est basé sur Linux pour la sécurité et la flexibilité des outils, nous utiliserons le terminal "Termux" qui contient cet environnement où nous installerons le réseau de communication.

Termux est un émulateur de Linux dans lequel nous allons installer les paquets nécessaires pour créer notre réseau de communication entre les nœuds.

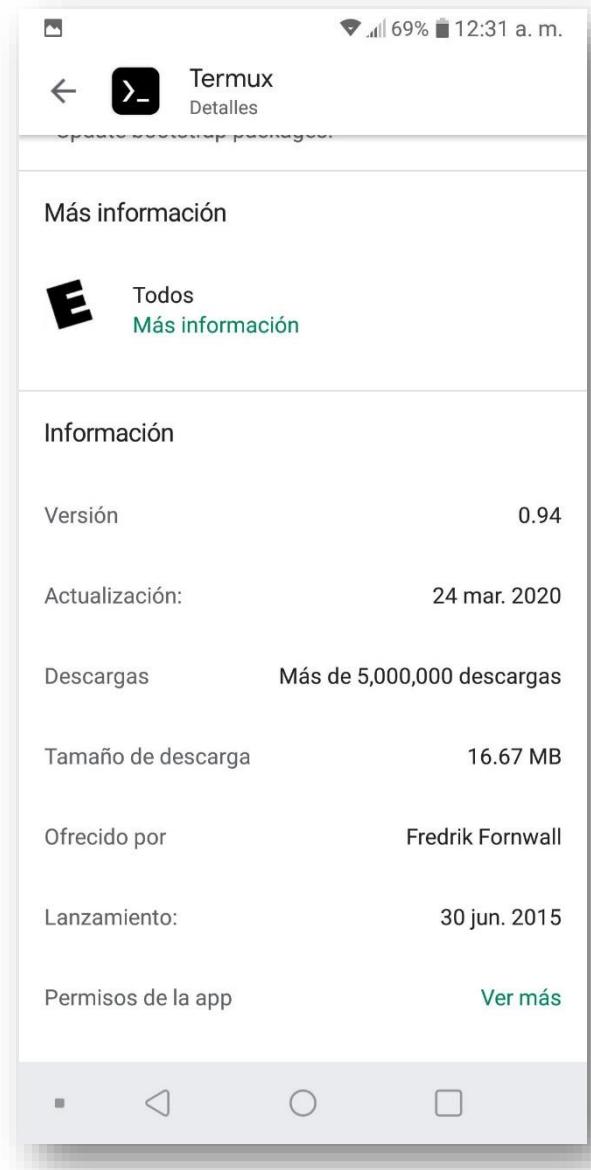
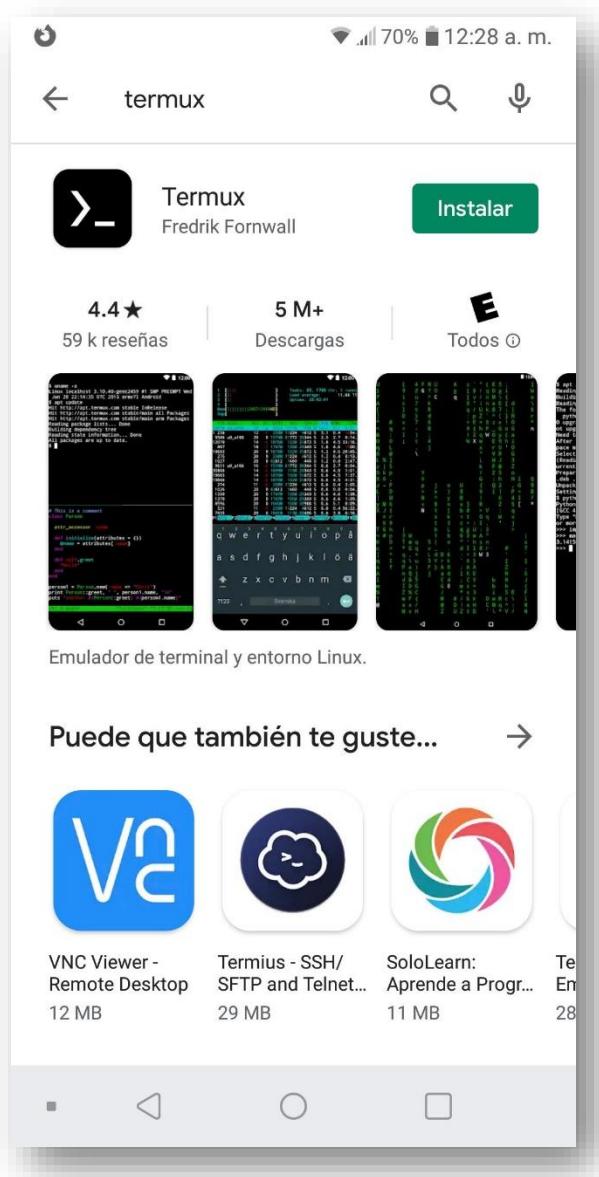
L'un des principaux avantages de l'utilisation de Termux est que vous pouvez installer des programmes sans avoir à "faire tourner" le téléphone portable (Smartphone), ce qui garantit qu'aucune garantie du fabricant n'est perdue à cause de cette installation.

Installation de Termux.

Depuis votre mobile, rendez-vous sur l'application Google Play (play.google.com).



Recherchez par l'application "Termux", sélectionnez-la et lancez le processus d'installation.



Début de l'application Termux.

Après le démarrage, nous devrons exécuter les deux commandes suivantes pour effectuer les mises à jour de l'émulateur du système d'exploitation Linux :

Mise à jour de \$ apt

Mise à niveau appropriée

Confirmez toutes les options O(Oui)...

Termux

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$ 
```

A screenshot of the Termux application interface on a mobile device. The title bar shows 'Termux'. The status bar indicates battery level (69%), signal strength, and time (12:31 a.m.). The main terminal window displays the welcome message and basic usage instructions for package management and repository subscription. A yellow arrow points to the command '\$ apt update' in the bottom right corner of the terminal window.

Home \$ apt update

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$ apt update
```

A screenshot of the Termux application interface on a mobile device. The title bar shows 'Termux'. The status bar indicates battery level (69%), signal strength, and time (12:32 a.m.). The main terminal window shows the command '\$ apt update' being typed. A yellow arrow points to the command '\$ apt update' in the bottom right corner of the terminal window.

\$ apt upgrade

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$ apt upgrade
```

A screenshot of the Termux application interface on a mobile device. The title bar shows 'Termux'. The status bar indicates battery level (69%), signal strength, and time (12:32 a.m.). The main terminal window shows the command '\$ apt upgrade' being typed. A yellow arrow points to the command '\$ apt upgrade' in the bottom right corner of the terminal window.

11. Configuration du stockage dans Termux.

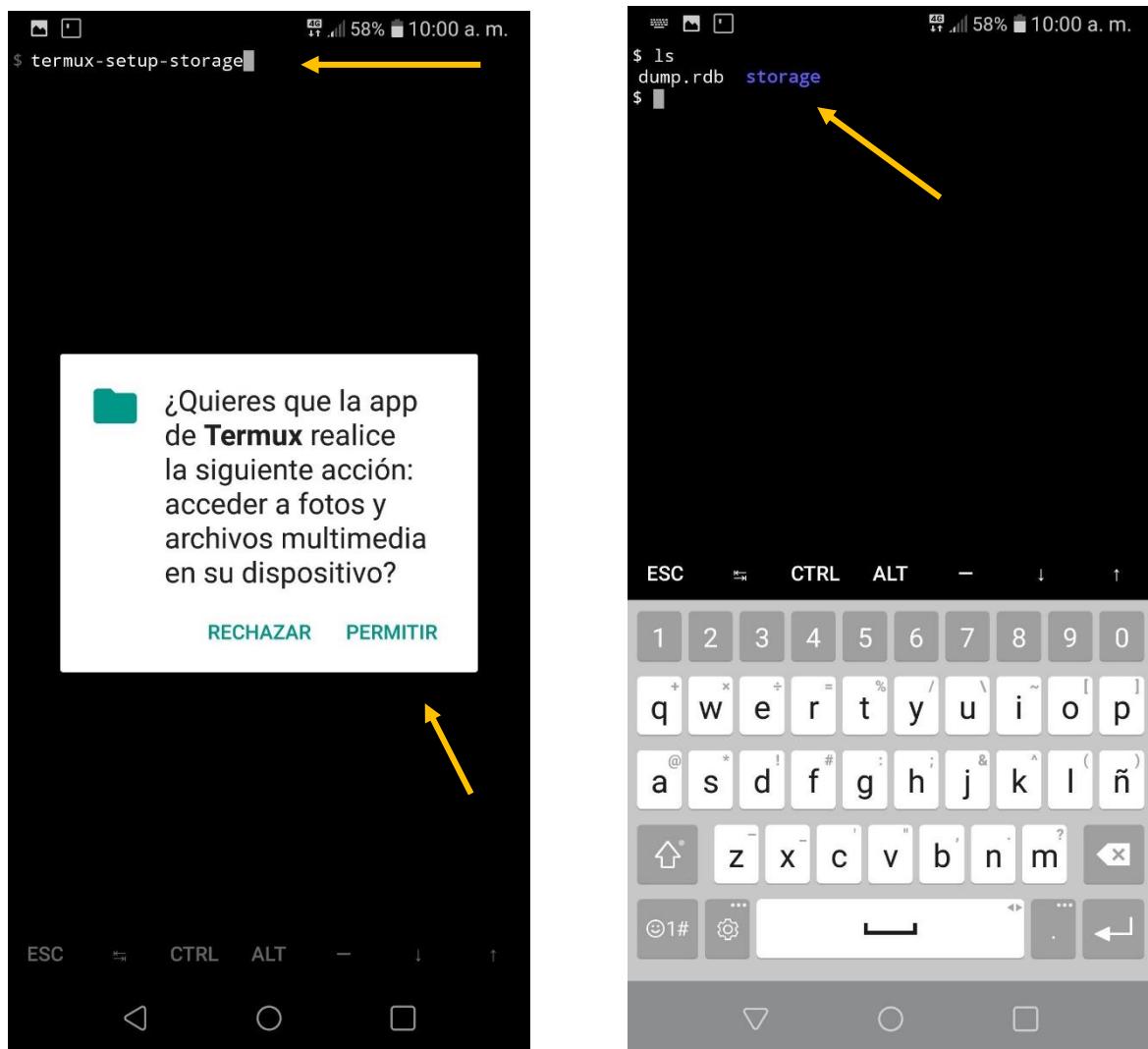
Une fois que vous aurez mis à jour et amélioré le système Termux, nous commencerons à configurer la façon de visualiser le stockage interne du téléphone dans le système Termux. Cela vous aidera à pouvoir échanger des informations entre Termux et nos informations dans le téléphone.

Cela peut être fait simplement et rapidement en exécutant la commande suivante sur un terminal Termux.

`$ termux-setup-storage`

Lorsque vous exécutez la commande précédente, une fenêtre apparaît pour vous demander de confirmer la création d'un **stockage** virtuel (répertoire) dans Termux. Nous vérifions en donnant l'ordre :

`ls`



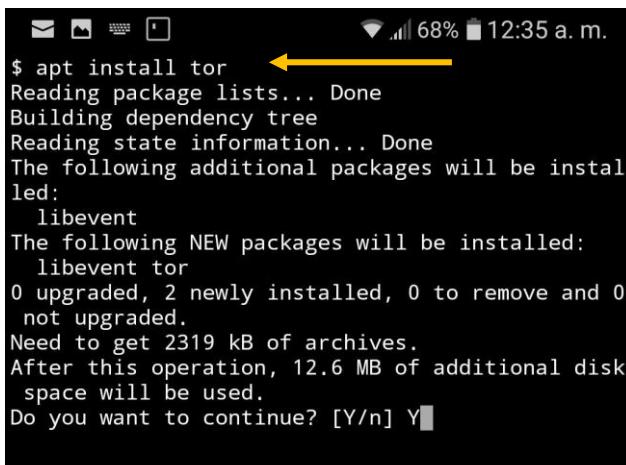
12. Installation du réseau "Tor" et installation de "Syncthing".

```
$ apt install tor
```

```
$ apt install syncthing
```

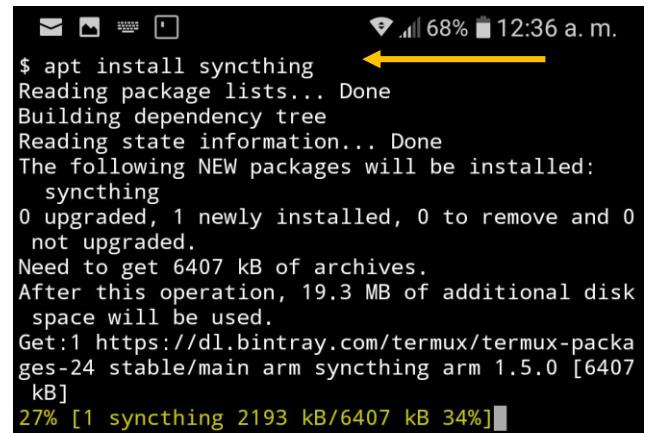
Acceptez l'installation en tapant la lettre Y majuscule dans les deux cas si vous le souhaitez...

\$ apt install tor

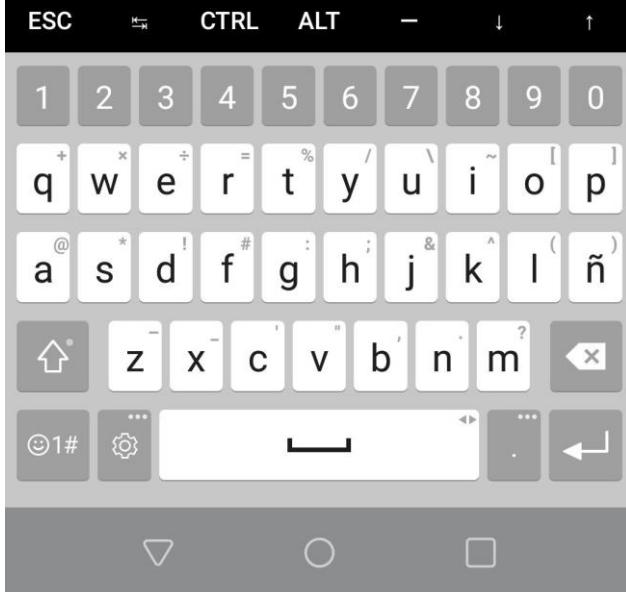


```
$ apt install tor
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libevent
The following NEW packages will be installed:
libevent tor
0 upgraded, 2 newly installed, 0 to remove and 0
not upgraded.
Need to get 2319 kB of archives.
After this operation, 12.6 MB of additional disk
space will be used.
Do you want to continue? [Y/n] Y
```

\$ apt install syncthing



```
$ apt install syncthing
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
syncthing
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 6407 kB of archives.
After this operation, 19.3 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packages-24/stable/main arm syncthing arm 1.5.0 [6407 kB]
27% [1 syncthing 2193 kB/6407 kB 34%]
```



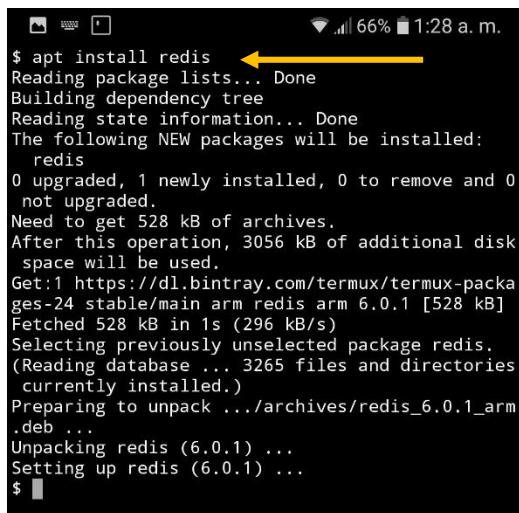
13. Installation de la base de données "Redis" et du serveur SSH (Secure Shell).

```
$ apt install redis
```

```
$ apt install openssh
```

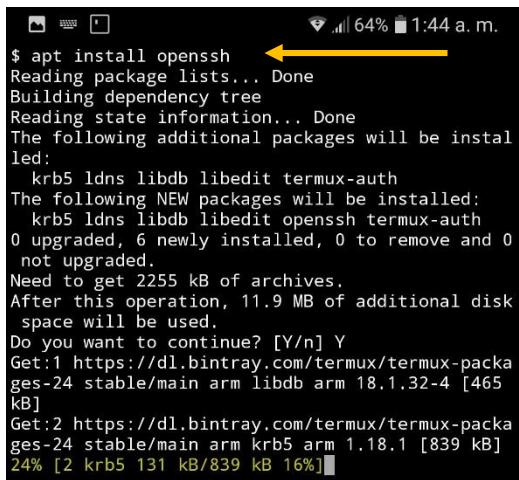
```
$ apt install sshpass
```

\$ apt install redis



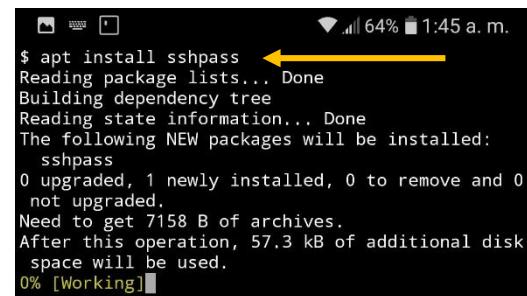
```
$ apt install redis
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  redis
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 528 kB of archives.
After this operation, 3056 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm redis arm 6.0.1 [528 kB]
Fetched 528 kB in 1s (296 kB/s)
Selecting previously unselected package redis.
(Reading database ... 3265 files and directories
currently installed.)
Preparing to unpack .../archives/redis_6.0.1_arm
.deb ...
Unpacking redis (6.0.1) ...
Setting up redis (6.0.1) ...
$
```

\$ apt install openssh



```
$ apt install openssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be instal
led:
  krb5-libs libdb5 libedit termux-auth
The following NEW packages will be installed:
  krb5-libs libdb5 libedit openssh termux-auth
0 upgraded, 6 newly installed, 0 to remove and 0
not upgraded.
Need to get 2255 kB of archives.
After this operation, 11.9 MB of additional disk
space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm libdb5 arm 18.1.32-4 [465
kB]
Get:2 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm krb5-libs arm 1.18.1 [839 kB]
24% [2/krb5 131 kB/839 kB 16%]
```

\$ apt install sshpass



```
$ apt install sshpass
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sshpass
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 7158 B of archives.
After this operation, 57.3 kB of additional disk
space will be used.
0% [Working]
```

Nous avons terminé l'installation du réseau de communication, nous continuons la configuration des paquets : Tor, Syncthing et Redis DB.

14. Configuration du serveur SSH sur le téléphone mobile (smartphone).

Nous allons permettre au serveur SSH du téléphone portable de se connecter de notre PC au portable et de pouvoir travailler de manière plus rapide et plus confortable, cela servira également à vérifier que le service du serveur SSH du portable fonctionne correctement puisque nous l'utiliserons dans le réseau de communication de la Mini BlocklyChain.

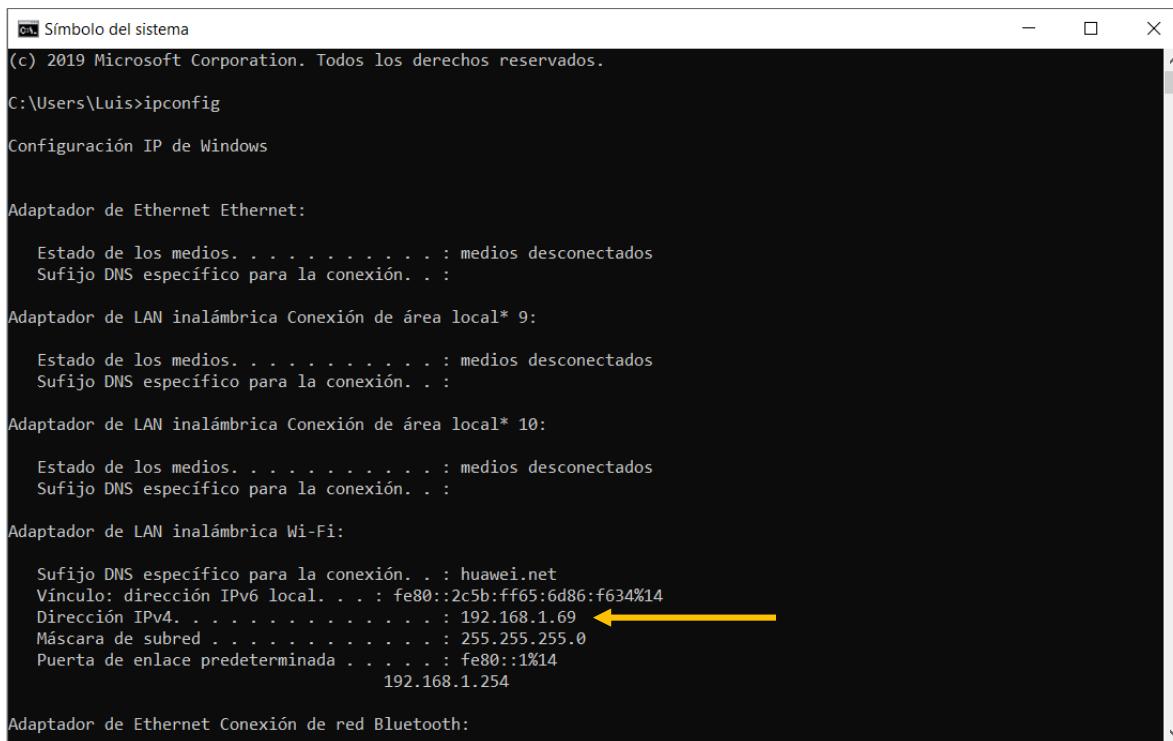
La première chose à faire est de connecter le mobile et le PC au **même réseau WiFi pour** qu'ils puissent se voir. Les IP ou adresses doivent être similaires à 192.168.XXX.XXX ; les valeurs XXX sont des nombres variables qui sont attribués de manière aléatoire dans chaque ordinateur.

Cet exemple a été testé sur un téléphone portable LG Q6 et un PC avec Windows 10 Home.

Vérifiez l'IP ou l'adresse que le PC a connecté au WiFi ; il faut ouvrir un terminal dans Windows.

Dans le panneau inférieur où se trouve la loupe de recherche, écrivez cmd et appuyez sur la touche Entrée. Un terminal s'ouvre et on y inscrit la commande :

C:\Nom_de_l'utilisateur> ipconfig



```

Símbolo del sistema
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:
    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 9:
    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 10:
    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Wi-Fi:
    Sufijo DNS específico para la conexión. . . : huawei.net
    Vínculo: dirección IPv6 local. . . : fe80::2c5b:ff65:6d86:f634%14
    Dirección IPv4. . . . . : 192.168.1.69
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . : fe80::1%14
                                         192.168.1.254

Adaptador de Ethernet Conexión de red Bluetooth:

```

Il nous montrera l'adresse IP attribuée au PC dans ce cas est 192.168.1.69 mais il est fort probable qu'elle soit différente dans chaque cas.

REMARQUE : l'adresse où il est indiqué "adresse IPv4" doit être prise, à ne pas confondre avec la passerelle.

Maintenant, dans le cas du téléphone portable dans le terminal Termux, nous devons taper la commande suivante pour connaître le nom de notre utilisateur que nous utiliserons pour nous connecter au serveur SSH qui a notre téléphone, nous exécutons la commande suivante:

\$ whoami

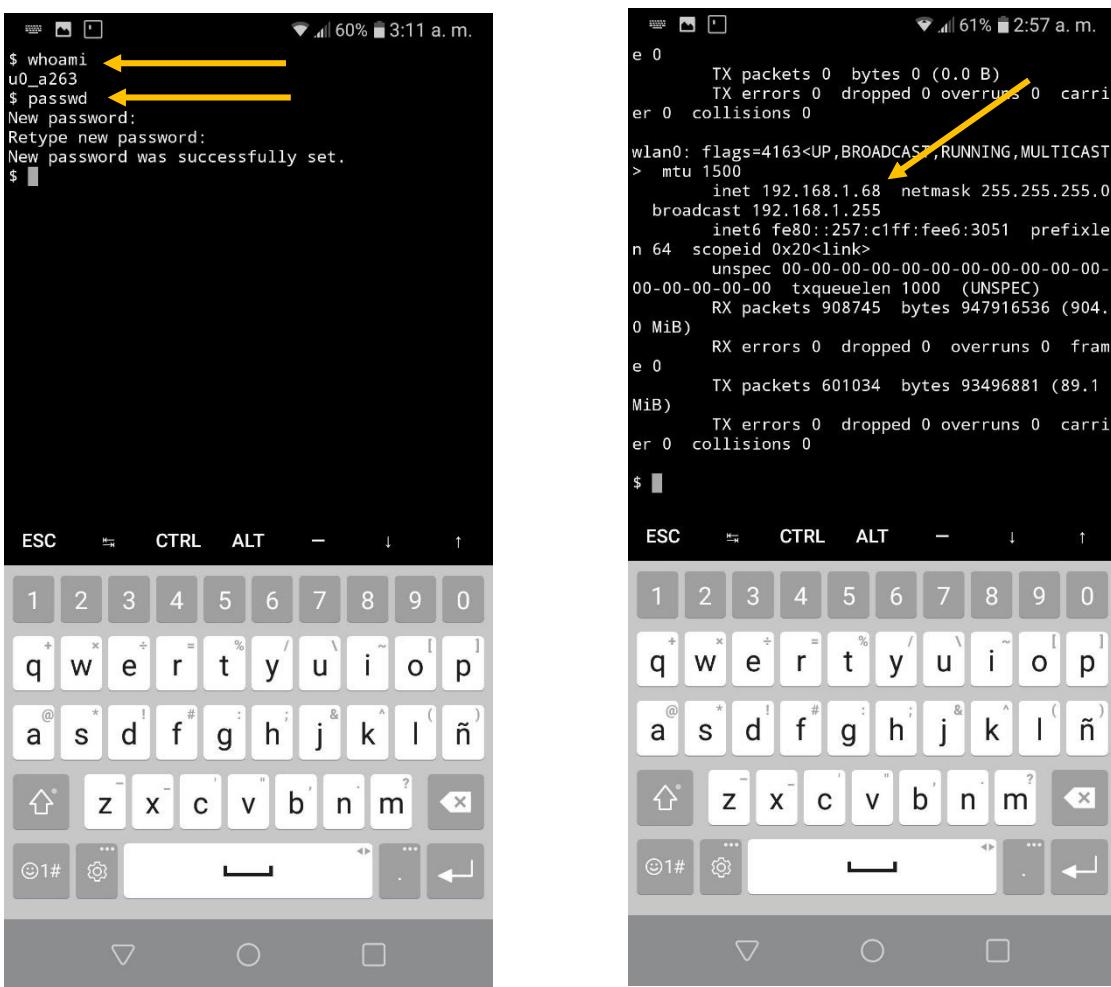
Plus tard, nous devons donner un mot de passe à cet utilisateur et nous devons donc exécuter la commande suivante :

\$ passwd

Il nous demandera de taper un mot de passe et d'appuyer sur Entrée, il nous demande à nouveau le mot de passe que nous confirmons et appuyons sur Entrée, s'il a été correctement "**Nouveau mot de passe a été défini avec succès**" en cas de marquage d'une erreur est possible que le mot de passe n'ait pas été tapé correctement. Répétez la procédure.

Et puis pour savoir quelle IP nous avons dans Termux nous tapons la commande suivante, l'IP est après le mot "**inet**" :

\$ ifconfig -a



Il est maintenant temps de lancer le service de serveur SSH sur votre téléphone pour que vous puissiez recevoir des sessions depuis votre PC. Nous exécutons la commande suivante dans le terminal Termux, cette commande ne donne aucun résultat.

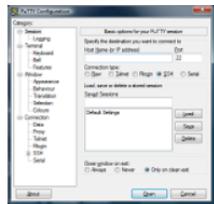
\$ sshd



Nous allons maintenant devoir installer sur le PC un programme qui communiquera avec le serveur SSH du téléphone à partir du PC.

Il faut aller sur <https://www.putty.org>

Sélectionnez où se trouve le lien "Vous pouvez télécharger PuTTY ici".

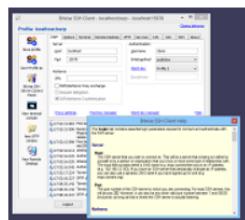


Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).

Below suggestions are independent of the authors of PuTTY. They are *not* to be seen as



Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported prof supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).

Choisissez la version 32 bits, peu importe que votre système soit en 64 bits, il fonctionnera parfaitement.

Download PuTTY: latest release

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirror](#)
Download: **Stable** · [Snapshot](#) | [Docs](#) | [Changelog](#)

This page contains download links for the latest released version of PuTTY. Currently this is 0.73, released on 2019-09-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternative Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date, so check the [development snapshots](#), to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

MSI ('Windows Installer')

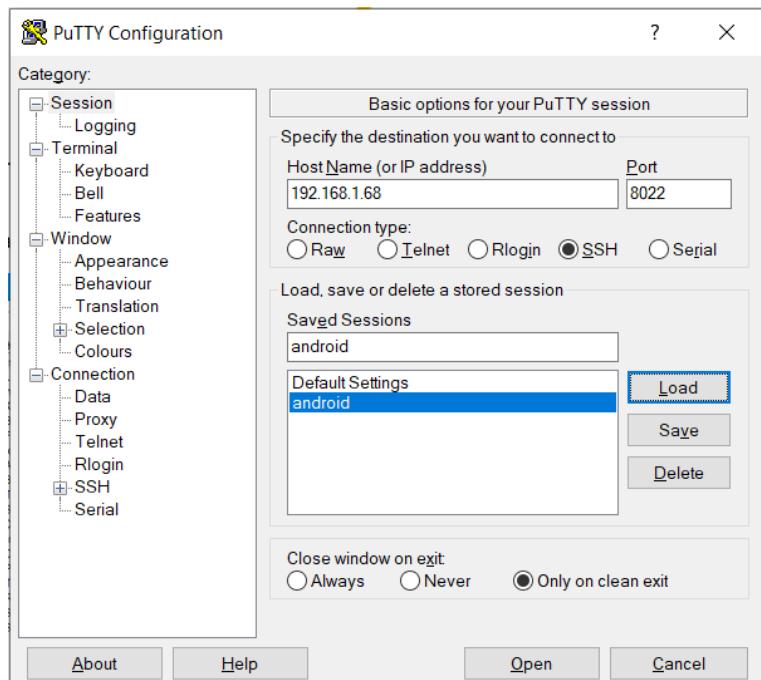
32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.73.tar.gz	(or by FTP)	(signature)
----------	-----------------------------------	------------------------------	-----------------------------

Une fois qu'il a été téléchargé sur votre PC, lancez-le et installez-le avec les options par défaut. Ensuite, lancez l'application PuTTY.

Dans cette session, nous allons entrer les données de notre serveur Openssh que nous avons installé dans le téléphone portable.



Entrez l'adresse IP du téléphone portable.

Nom d'hôte ou adresse IP :

192.168.1.68 (exemple IP)

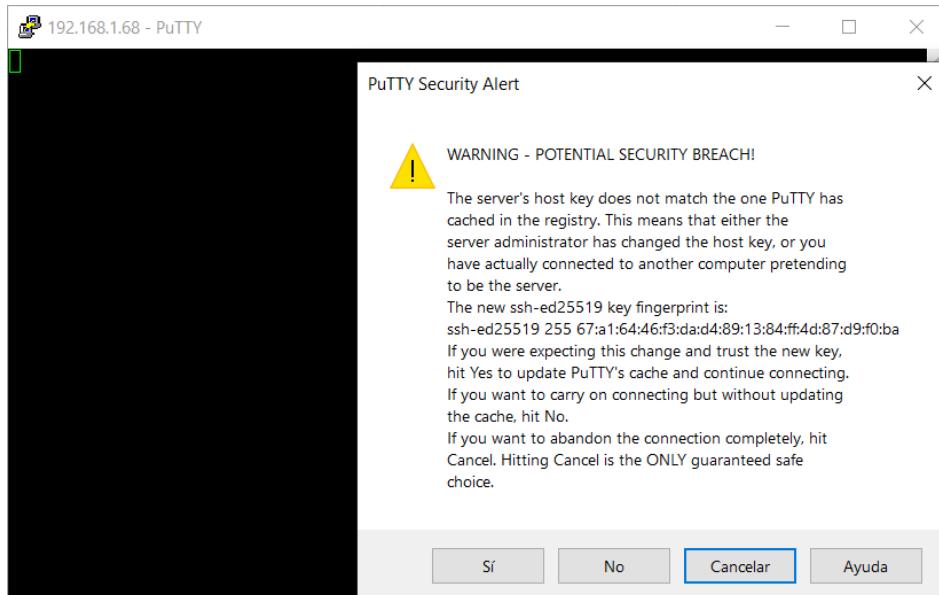
Port :

8022 (port par défaut du serveur SSH mobile).

Nous pouvons donner un nom à la session dans "Sessions enregistrées" et cliquer sur le bouton "Enregistrer".

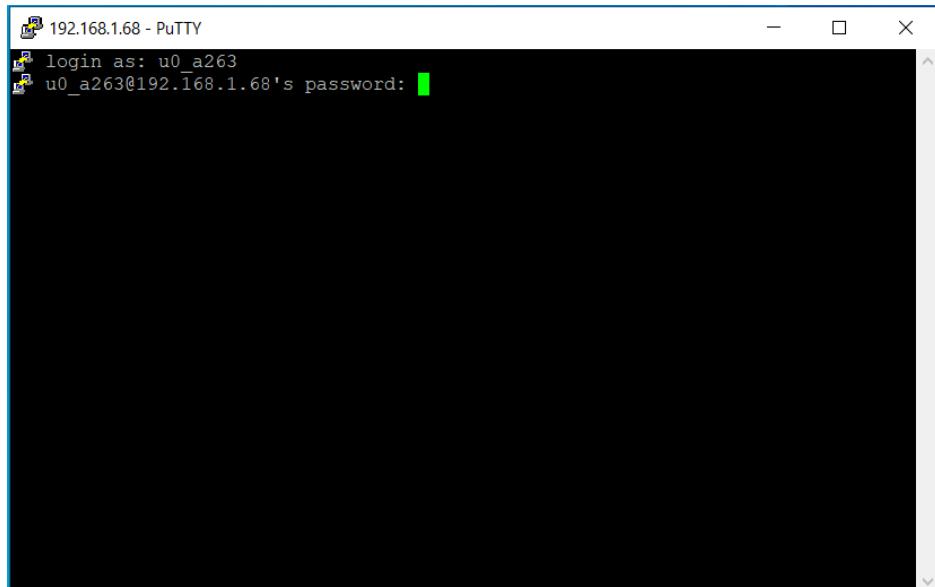
Plus tard, dans la partie inférieure, nous appuyons sur le bouton "Ouvrir" pour ouvrir une connexion au serveur.

Sur le PC, lorsque vous vous connectez pour la première fois, il vous sera demandé de confirmer la clé de cryptage des informations en cliquant sur le bouton "Oui".

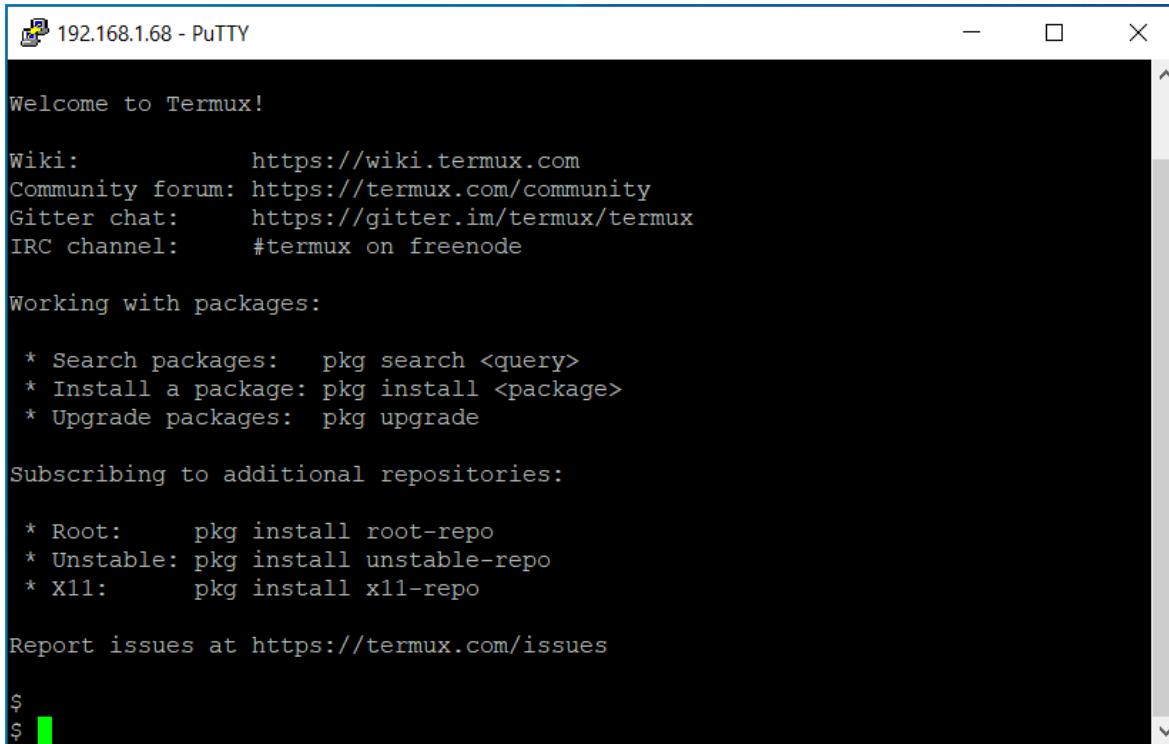


Plus tard, on nous demandera le nom de l'utilisateur avec lequel nous allons nous connecter. Nous utiliserons les informations que nous avons obtenues précédemment (utilisateur et mot de passe).

Dans le **Login comme** : nous devons entrer notre utilisateur et donner Entrée, puis nous demanderons le mot de passe à nouveau donner le bouton Entrée.



Si les données étaient correctes, nous serions dans une session SSH (Secure Shell) effectuée depuis le PC (Client) sur le téléphone (Serveur SSH).



```
Welcome to Termux!

Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:

* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues

$
```

NOTE IMPORTANTE : N'oubliez pas que l'IP (adresse) du PC et l'IP (adresse) du téléphone mobile connecté au même WiFi changeront probablement chaque fois que nous nous déconnecterons et nous reconnecterons. Nous devons donc vérifier les adresses de chaque appareil, ce qui garantira le succès de la connexion entre les appareils via le serveur SSH du téléphone et le PC (Client).

Jusqu'à présent, nous n'avons pu nous connecter qu'au même réseau WiFi, mais si nous déplaçons notre téléphone en dehors du même réseau que le PC, nous ne pourrons pas nous connecter parce qu'il y a différents réseaux impliqués dans le passage par d'autres appareils de communication plus compliqués. Nous résoudrons ce problème lorsque nous mettrons en place le réseau "Tor".

N'oubliez pas que cette connexion n'est faite que pour vérifier le service du serveur que nous avons installé sur le téléphone et pour avoir un environnement de travail plus confortable avec une session à distance d'un PC vers le téléphone.

15. Configuration du réseau "Tor" avec le service SSH (Secure Shell).

Avec la session à distance depuis le PC, nous commencerons à configurer le réseau "Tor" avec le service SSH activé.

L'importance d'avoir le réseau "Tor" est de donner aux appareils la propriété de pouvoir communiquer partout dans le monde via Internet sans être dans le même réseau WiFi, peu importe où nous sommes, nous pourrons nous connecter et former le réseau "Peer to Peer" entre les nœuds qui est une fonctionnalité essentielle de l'architecture Mini BlocklyChain.

Le réseau "Tor" a beaucoup de flexibilité dans sa configuration puisqu'il a plusieurs paramètres dans son fichier de configuration "torrc" ; ce fichier est dans le chemin (`$_PREFIX/etc/tor/torrc`) ; dans notre cas avec la session Termux depuis notre PC nous le saurons en tapant la commande suivante :

```
écho $_PREFIX
```

```
/data/data/com.termux/files/usr
```

Par conséquent, le fichier que nous devons éditer se trouvera dans le chemin d'accès :

```
$_PREFIX/etc/tor/torrc
```

 qui est égal à `/data/data/com.termux/files/usr/etc/tor/torrc`

Nous procédons à l'édition du fichier de configuration à l'aide de l'éditeur de ligne de commande "vi" en exécutant la commande suivante :

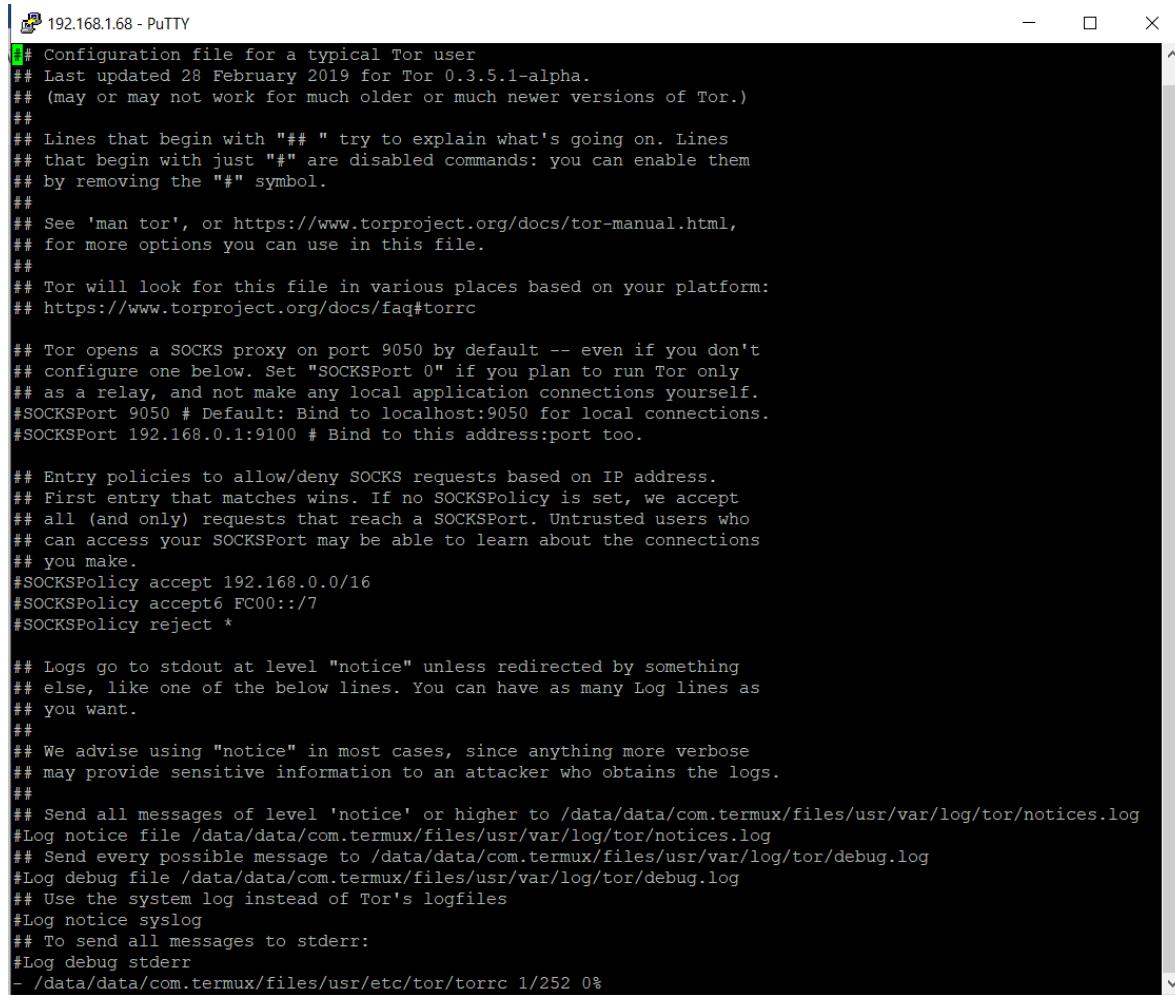
```
vi /data/data/com.termux/files/usr/etc/tor/torrc
```

Il va éditer ce fichier pour nous, à titre d'exemple :



```
192.168.1.68 - PuTTY
$ vi /data/data/com.termux/files/usr/etc/tor/torrc
```

Fichier "torrc" édité :



```
# Configuration file for a typical Tor user
## Last updated 28 February 2019 for Tor 0.3.5.1-alpha.
## (may or may not work for much older or much newer versions of Tor.)
##
## Lines that begin with "## " try to explain what's going on. Lines
## that begin with just "#" are disabled commands: you can enable them
## by removing the "#" symbol.
##
## See 'man tor', or https://www.torproject.org/docs/tor-manual.html,
## for more options you can use in this file.
##
## Tor will look for this file in various places based on your platform:
## https://www.torproject.org/docs/faq#torrc

## Tor opens a SOCKS proxy on port 9050 by default -- even if you don't
## configure one below. Set "SOCKSPort 0" if you plan to run Tor only
## as a relay, and not make any local application connections yourself.
#SOCKSPort 9050 # Default: Bind to localhost:9050 for local connections.
#SOCKSPort 192.168.0.1:9100 # Bind to this address:port too.

## Entry policies to allow/deny SOCKS requests based on IP address.
## First entry that matches wins. If no SOCKSPolicy is set, we accept
## all (and only) requests that reach a SOCKSPort. Untrusted users who
## can access your SOCKSPort may be able to learn about the connections
## you make.
#SOCKSPolicy accept 192.168.0.0/16
#SOCKSPolicy accept6 FC00::/7
#SOCKSPolicy reject *

## Logs go to stdout at level "notice" unless redirected by something
## else, like one of the below lines. You can have as many Log lines as
## you want.
##
## We advise using "notice" in most cases, since anything more verbose
## may provide sensitive information to an attacker who obtains the logs.
##
## Send all messages of level 'notice' or higher to /data/data/com.termux/files/usr/var/log/tor/notices.log
#Log notice file /data/data/com.termux/files/usr/var/log/tor/notices.log
## Send every possible message to /data/data/com.termux/files/usr/var/log/tor/debug.log
#Log debug file /data/data/com.termux/files/usr/var/log/tor/debug.log
## Use the system log instead of Tor's logfiles
#Log notice syslog
## To send all messages to stderr:
#Log debug stderr
- /data/data/com.termux/files/usr/etc/tor/torrc 1/252 0%
```

Dans ce fichier "torrc", nous devrons ajouter ou utiliser les lignes que le fichier possède en effectuant les modifications suivantes, trois lignes qui sont les suivantes :

Syntaxe : **SOCKSPort <numéro de port de l'application>**

Exemple : **SOCKSPort 9050**

La variable **SOCKSPort** nous indique que cette prise de communication sur le protocole TCP-IP utilisera par défaut l'appareil mobile (téléphone) et que le port 9050 sera ouvert ou utilisé.

Syntaxe : **HiddenServiceDir <Répertoire où la configuration de l'application sera sauvegardée>**

Exemple : **HiddenServiceDir** /data/data/com.termux/files/

La variable **HiddenServiceDir** nous indique que ce sera le répertoire où sera stockée la configuration du service qui sera utilisé à travers le réseau Tor. Dans ce répertoire vous trouverez le fichier de configuration et de sécurité du service, et dans ce répertoire vous trouverez un fichier appelé **hostname**.

Nous pouvons voir que l'adresse assignée par le réseau Tor pour le service spécifique créé dans notre cas est la création d'un service SSH qui sera mis en œuvre sur le réseau Tor, le répertoire que nous nommons comme **hidden_ssh** contiendra le fichier de **nom d'hôte**. Ce répertoire n'a pas besoin d'être créé, car lorsque nous démarrons le service du réseau Tor, il sera créé automatiquement.

Pour voir l'adresse fournie par le réseau Tor, nous pouvons utiliser la commande "more". Avant d'utiliser cette commande, nous devons terminer la configuration du fichier "torrc" et enregistrer le service du réseau Tor de manière à ce que le répertoire **hidden_ssh** et les fichiers à l'intérieur de celui-ci soient créés, comme c'est le cas pour le fichier **hostname**.

plus /data/data/com.termux/files/

La commande ci-dessus donnera une adresse avec une chaîne alphanumérique avec une extension .onion similaire à :

uqwthf6ojdmoybyzvudoq3x2weq7k7rjitblhba3pjbausk2nvjmx3wer.onion

Enfin, nous devons ajouter la variable **HiddenServicePort** au fichier de configuration "torrc". Ce paramètre indique au réseau Tor quel port l'application pour laquelle nous nous inscrivons sera utilisée sur le réseau Tor.

Syntaxe : **HiddenServicePort** <Port de départ> <IP local ou public> : <Port de départ>

O

HiddenServicePort <Port de départ>

Exemples :

HiddenServicePort 22 127.0.0.1:8022

O

HiddenServicePort 8022

Avec ce qui précède, nous avons terminé la configuration du réseau Tor pour les services génériques et le service SSH (Secure Shell). Ce service sera utilisé pour la communication de mise à jour de la base de données SQLite où nous allons sauvegarder les processus de validation de notre système Mini BlocklyChain.

Nous poursuivons la configuration du réseau de communication de la Mini BlocklyChain.

16. Configuration du système Peer to Peer avec synchronisation manuelle.

Une architecture "Peer to Peer" est fondamentale dans un système de technologie de chaîne de blocs car cette architecture donne deux points centraux dans les processus de communication du système, l'un est de fournir la même information mise à jour (synchronisée) à tout moment dans tous les nœuds, peu importe si les nœuds sont dans un réseau privé (Wifi) ou un réseau public (Internet) ou un hybride de ces deux et un deuxième point de ce type d'architecture est qu'ils ne dépendent pas d'un intermédiaire (serveur) pour transférer, mettre à jour ou consulter l'information entre les nœuds. Tout cela est dû au fait que la communication se fait directement entre les nœuds, dans notre cas Mini BlocklyChain la communication se fait directement entre les téléphones qui forment le réseau sans intermédiaire.

Pour cette tâche, nous utiliserons l'outil open source Syncthing qui fonctionne sur la base d'une architecture "Peer to Peer".

La configuration de Syncthing pour les appareils (téléphones mobiles) sera vue manuellement et automatiquement. Le formulaire manuel est appliqué en synchronisation avec un maximum de 5 nœuds, dans le cas où le nombre de configurations automatiques est optimal, le processus se concentre sur l'enregistrement des nœuds avec lesquels nous voulons effectuer la synchronisation des informations sélectionnées.

Les conditions à remplir pour commencer sont les suivantes :

1. Avoir initié le service du réseau Tor.
2. (optionnel - recommandé) Si vous avez installé une application capable de lire les codes QR, nous vous suggérons l'application Android de l'inventeur de l'application, qui est facile et simple à installer depuis Google Play. Plus loin dans ce manuel, nous l'utiliserons dans la section "Définition et utilisation des blocs dans la Mini Blocklychain".
3. Avoir initié le service de Syncthing.



Nous montrons l'application App Inventor que nous utiliserons pour lire les codes QR qui nous serviront à l'avenir pour l'enregistrement des nœuds dans Syncthing.

L'exemple suivant a été réalisé entre deux appareils mobiles (téléphones) en utilisant les modèles suivants :

Appareil mobile n° 1 : modèle LG Q6

Appareil mobile n°2 : modèle Samsung

Nous allons commencer par démarrer les services du réseau Tor et les services de synchronisation en utilisant les commandes suivantes, ouvrir deux terminaux à l'intérieur de Termux, et exécuter chaque commande séparément :

\$ tor

Après avoir tapé la commande de démarrage du programme du réseau Tor, vous devez vérifier que l'exécution a été réussie, en vérifiant qu'elle a été faite à 100%.

En exécutant le programme Tor sur un terminal Termux, nous vérifions qu'il fonctionne à 100%.

\$ tor

```

$ tor
May 23 23:00:30.932 [notice] Tor 0.4.3.5 running
on Linux with Libevent 2.1.11-stable, OpenSSL 1
.1.1g, Zlib 1.2.11, Liblzma 5.2.5, and Libzstd N
/A.
May 23 23:00:30.934 [notice] Tor can't help you
if you use it wrong! Learn how to be safe at ht
ps://www.torproject.org/download/download#warnin
g
May 23 23:00:30.939 [notice] Read configuration
file "/data/data/com.termux/files/usr/etc/tor/to
rrc".
May 23 23:00:30.970 [notice] I think we have 8 C
PUS, but only 6 of them are available. Telling T
or to only use 6. You can override this with the
NumCPUs option
May 23 23:00:30.973 [notice] Opening Socks liste
ner on 127.0.0.1:9050
May 23 23:00:30.974 [notice] Opened Socks liste
ner on 127.0.0.1:9050
May 23 23:00:30.000 [notice] Parsing GEOIP IPv4
file /data/data/com.termux/files/usr/share/tor/g
eoip.
May 23 23:00:32.000 [notice] Parsing GEOIP IPv6
file /data/data/com.termux/files/usr/share/tor/g
eoip6.
May 23 23:00:34.000 [notice] Bootstrapped 0% (st
arting): Starting
May 23 23:00:34.000 [notice] Starting with guard
context "default"
May 23 23:00:35.000 [notice] Bootstrapped 5% (co
nn): Connecting to a relay
May 23 23:00:36.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 23 23:00:36.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 23 23:00:36.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 23 23:00:36.000 [notice] Bootstrapped 20% (o
nehop_create): Establishing an encrypted directo
ry connection
May 23 23:00:36.000 [notice] Bootstrapped 25% (r

```

ESC ⌂ CTRL ALT - ↓ ↑

■ ◀ ○ □

```

ps://www.torproject.org/download/download#warnin
g
May 24 01:33:32.982 [notice] Read configuration
file "/data/data/com.termux/files/usr/etc/tor/to
rrc".
May 24 01:33:33.007 [notice] I think we have 8 C
PUS, but only 6 of them are available. Telling T
or to only use 6. You can override this with the
NumCPUs option
May 24 01:33:33.010 [notice] Opening Socks liste
ner on 127.0.0.1:9050
May 24 01:33:33.010 [notice] Opened Socks liste
ner on 127.0.0.1:9050
May 24 01:33:33.000 [notice] Parsing GEOIP IPv4
file /data/data/com.termux/files/usr/share/tor/g
eoip.
May 24 01:33:34.000 [notice] Parsing GEOIP IPv6
file /data/data/com.termux/files/usr/share/tor/g
eoip6.
May 24 01:33:35.000 [notice] Bootstrapped 0% (st
arting): Starting
May 24 01:33:37.000 [notice] Starting with guard
context "default"
May 24 01:33:38.000 [notice] Bootstrapped 5% (co
nn): Connecting to a relay
May 24 01:33:38.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 24 01:33:39.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 24 01:33:39.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 24 01:33:39.000 [notice] Bootstrapped 75% (e
nough_dirinfo): Loaded enough directory info to
build circuits
May 24 01:33:39.000 [notice] Bootstrapped 90% (a
p_handshake_done): Handshake finished with a rel
ay to build circuits
May 24 01:33:39.000 [notice] Bootstrapped 95% (c
ircuit_create): Establishing a Tor circuit
May 24 01:33:42.000 [notice] Bootstrapped 100% (d
one): Done

```

ESC ⌂ CTRL ALT - ↓ ↑

■ ◀ ○ □

Ensuite, nous exécutons la commande de synchronisation :

Synchronisation des \$.

Après avoir exécuté cette commande, il ouvrira une page d'administration dans le navigateur de notre téléphone ; si elle ne s'ouvre pas automatiquement, nous pouvons aller dans n'importe quel navigateur que nous avons installé là où nous naviguons normalement sur Internet et nous pouvons mettre le l suivant :

<http://127.0.0.1:8384>

Il ouvrira l'écran d'administration de l'outil qui nous aidera à synchroniser nos informations entre tous les nœuds (téléphones) du système.



```
$ syncthing
[monitor] 04:02:07 INFO: Default folder created
and/or linked to new config
[start] 04:02:07 INFO: syncthing v1.5.0 "Fermium
Flea" (go1.14.2 android-arm) builder@6bdf862223
8a 2020-05-11 08:38:11 UTC
[start] 04:02:07 INFO: Generating ECDSA key and
certificate for syncthing...
[start] 04:02:08 INFO: Default folder created an
d/or linked to new config
[start] 04:02:08 INFO: Default config saved. Edi
t /data/data/com.termux/files/home/.config/synct
hing/config.xml to taste (with Syncthing stopped
) or use the GUI
[OWEPS] 04:02:08 INFO: My ID: OWEPSNA-6RZI657-SK
VOU65-V44BC5Z-CXZOJI4-C3JC7HM-IPY4V35-JQAKPAW
[OWEPS] 04:02:09 INFO: Single thread SHA256 perf
ormance is 12 MB/s using crypto/sha256 (12 MB/s
using minio/sha256-simd).
[OWEPS] 04:02:11 INFO: Hashing performance is 11
.17 MB/s
[OWEPS] 04:02:11 INFO: Migrating database to sch
ema version 1...
```



Métrica	Datos
Velocidad de descarga	0 B/s (0 B)
Velocidad de subida	0 B/s (0 B)
Estado Local (Total)	0 0 ~0 B
Oyentes	3/3
Descubrimiento	4/5
Tiempo de funcionamiento	1m
Versión	v1.5.0, android (ARM)

Otros dispositivos

- Cambios recientes
- Añadir un dispositivo

Comme l'écran d'administration "Synchronisation" est ouvert, nous procérons à l'enregistrement du ou des nœuds dont nous voulons synchroniser les informations. C'est à ce stade que nous allons occuper le programme qui lit les codes QR.

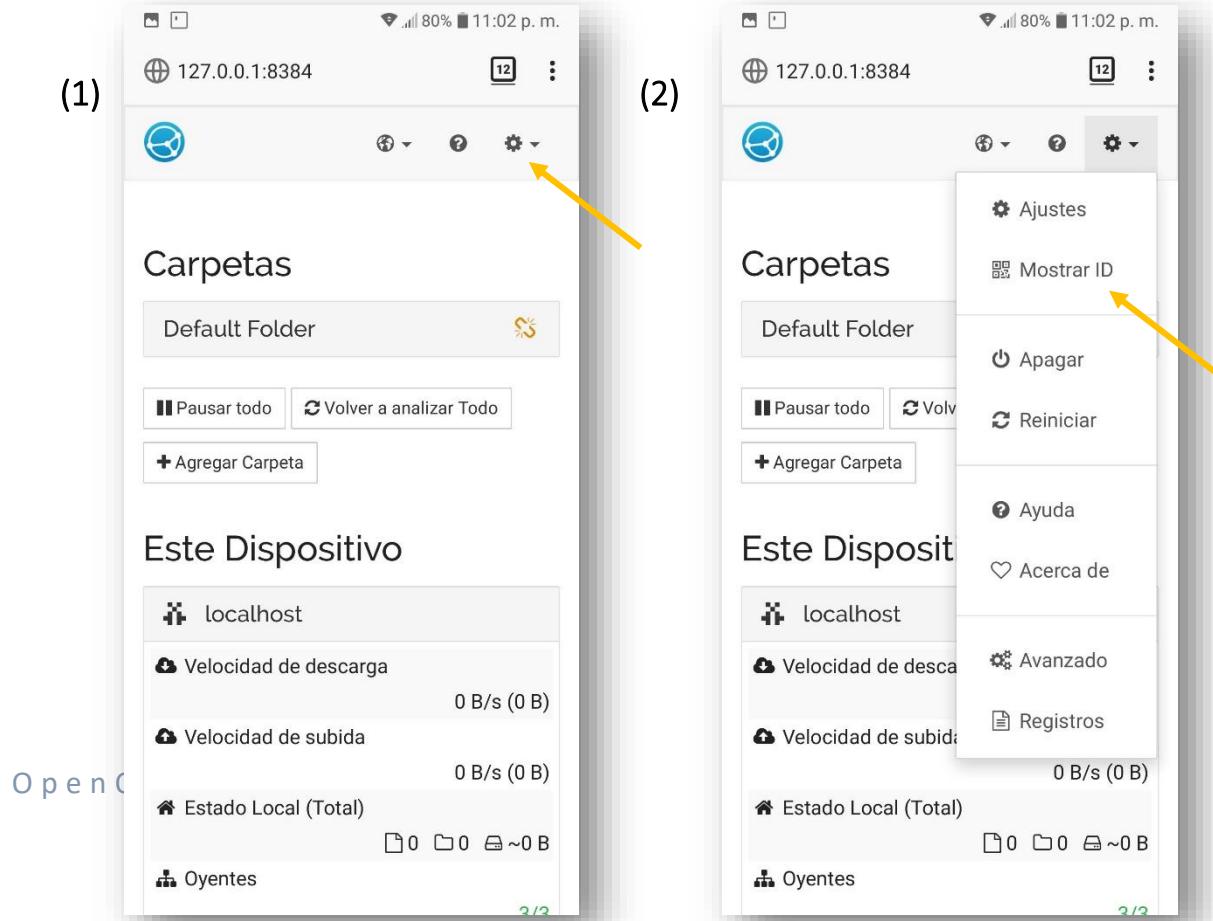
Le programme de synchronisation, lorsqu'il démarre pour la première fois, crée un identifiant unique du téléphone qui est composé d'un groupe de huit ensembles de caractères alphanumériques en majuscules, cet identifiant (ID) est celui que nous allons enregistrer dans le ou les nœuds dont nous voulons synchroniser les informations.

Dans notre cas, le numéro d'identification du téléphone LG Q6 devra être enregistré sur le téléphone Sansumg et le numéro d'identification du téléphone Sansumg devra être enregistré sur le LG Q6. Ils doivent se trouver dans les deux téléphones pour fonctionner correctement.

Nous effectuerons les étapes de l'enregistrement du téléphone mobile Sansumg sur le téléphone LG Q6.

Tout d'abord (1) en haut de l'écran d'administration (navigateur internet) du téléphone Sansumg avec synchronisation nous allons cliquer sur l'onglet menu en haut à droite.

Dans la deuxième étape (2), un menu apparaît, dans lequel on clique sur "Show ID" du Sansumg.

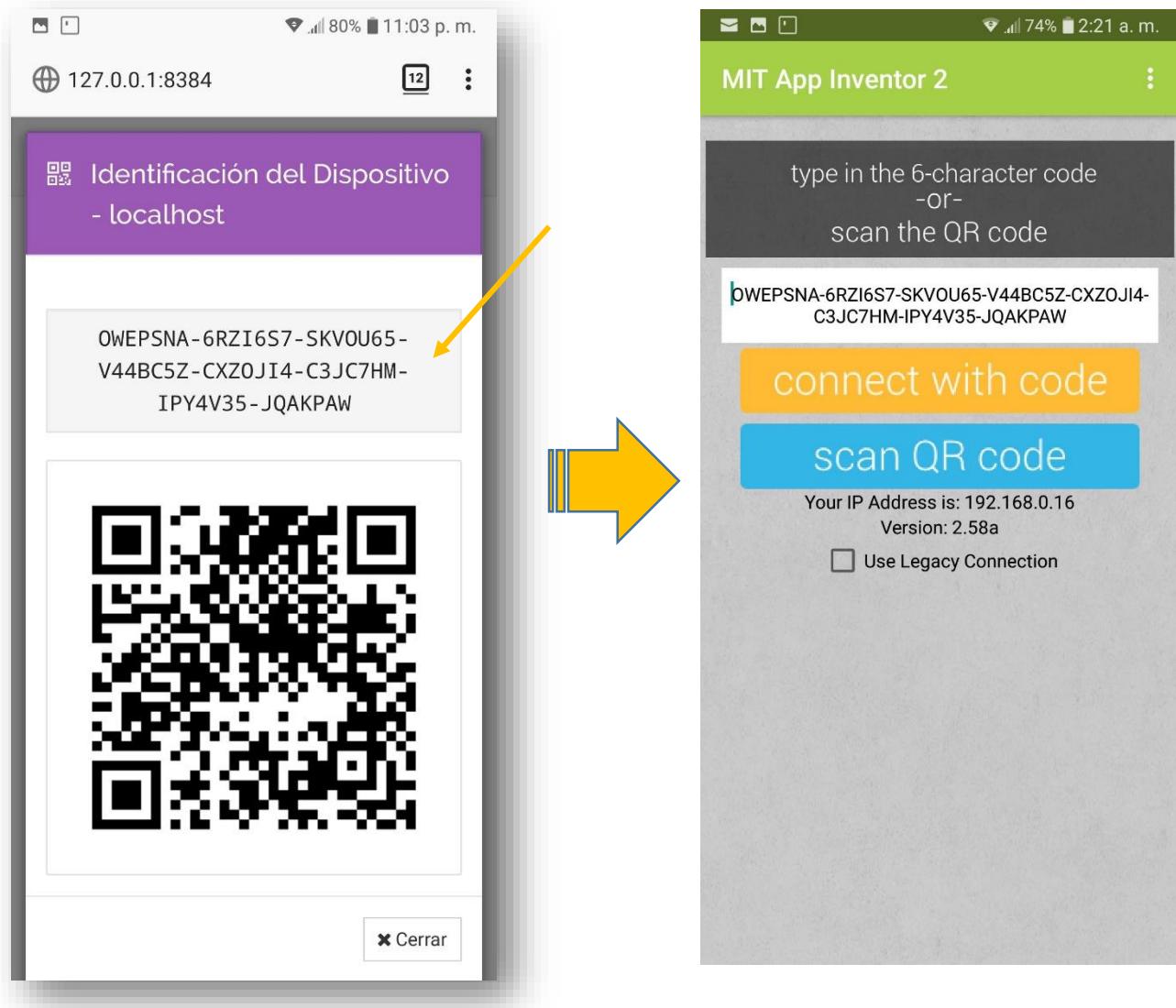


Lorsque vous cliquez sur "Show ID", l'écran suivant apparaît, qui est un code QR du téléphone Sansumg ; dans notre cas, l'ID qui identifie le téléphone est

OWEPSNA-6RZI6S7-SKVOU65-V44BC5Z-CXZOJI4-C3JC7HM-IPY4V35-JQAKPAW

Ensuite, dans le téléphone LG Q6, le programme qui nous aidera à capturer le code QR de ce Sansumg est celui que nous suggérons à partir de l'application App Inventor (optionnel), bien que dans le cas où nous ne l'avons pas, nous pouvons aussi simplement l'introduire manuellement lorsque nous démarrons l'enregistrement dans le LG Q6, cependant, pour éviter toute erreur nous suggérons de l'utiliser.

Utilisation de l'inventeur du programme App installé dans le mobile LG Q6 pour capturer le code QR du téléphone Sansumg distant que nous voulons synchroniser.



Ensuite, nous copions dans le presse-papiers le code QR dans le programme de l'inventeur de l'application en cliquant sur le code capturé, nous choisissons "SELECT ALL" → "COPY".

Avec ces informations, nous procédons à l'enregistrement des Sansumg dans le LG Q6. Dans l'écran d'administration, nous passons à la partie inférieure où il est indiqué "Autres appareils" et nous cliquons sur le bouton "Ajouter un appareil", nous introduisons l'ID du Sansumg et nous lui donnons un nom d'enregistrement.

Ensuite, dans l'onglet supérieur "Partager", nous cliquons et dans celui-ci nous marquons l'option inférieure dans le dossier "Default" avec ceci nous partagerons un répertoire qui a été créé par défaut appelé Sync dans notre appareil.

Ensuite, en haut, nous cliquons sur l'onglet "Avancé" et sélectionnons l'option de compression des données sous "Toutes les données".

Enfin, on clique sur le bouton de sauvegarde inférieur, on termine l'enregistrement dans un nœud, ce même processus doit être fait dans le ou les téléphones distants que l'on veut synchroniser.

Device Info (Left Screen):

- ID del Dispositivo: /IQXYQ-ILHCOID-BUPFIWU-MJJUNQD
- Nombre del Dispositivo: SANSUMG

Advanced Sharing Options (Middle Screen):

- Presentador: Añadir dispositivos desde el introductor a nuestra lista de dispositivos, para las carpetas compartidas mutuamente.
- Aceptar automáticamente: Crear o compartir automáticamente carpetas que este dispositivo anuncia en la ruta por defecto.

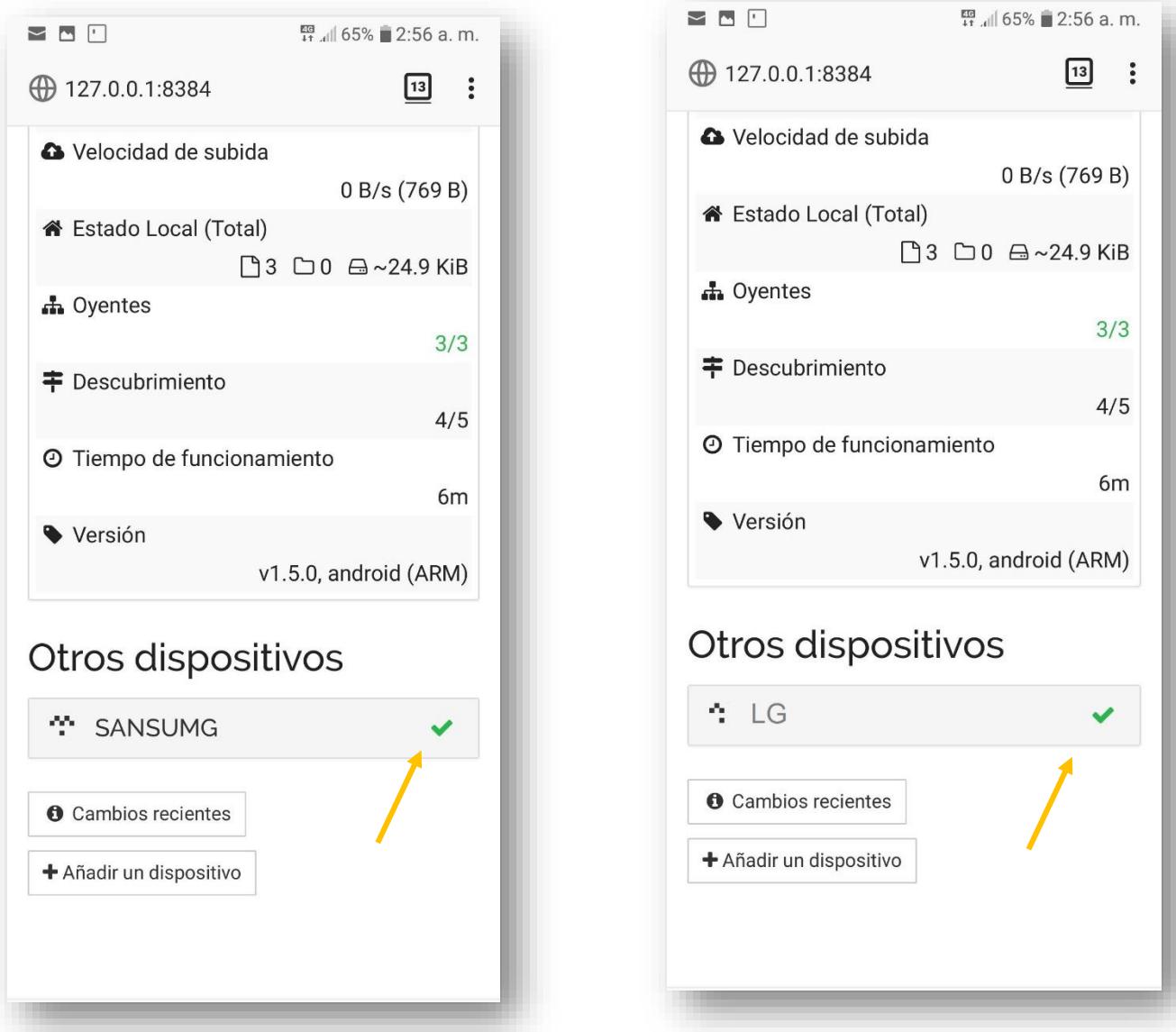
Compression Settings (Right Screen):

- Direcciones: dynamic
- Compresión: Todos los datos

Lors de la sauvegarde et de l'enregistrement dans les deux téléphones, nous attendrons une période maximale de 30 secondes pendant laquelle les appareils seront localisés et la connexion entre les appareils apparaîtra aussi bonne en donnant une confirmation en vert.

Dispositif n° 1 LG Q6

Dispositif n° 2 Sansumg



Toutes les informations du répertoire **Sync** situé dans le chemin d'accès **/data/data/com.termux/file/home/Sync** seront synchronisées, toute modification sera copiée et synchronisée.

Configuration du système Peer to Peer avec Syncthing automatique pour utilisation dans la Mini BlocklyChain.

Nous allons maintenant procéder à la configuration de manière automatique, alors qu'auparavant, le processus manuel était utile si nous gérons un nombre minimum de nœuds. Cependant, dans le cas d'un grand nombre de nœuds, ce serait inefficace et nous disposerons plus tard de l'outil SyncThingManager pour le configurer de manière automatique à l'aide de commandes en ligne automatisées.

Configuration de la base de données Redis pour les nœuds en mode (esclave).

Configuration du fichier `/data/data/com.termux/files/usr/etc/redis.conf` de la base de données Redis (**Slave**) pour les téléphones mobiles Android.

Ajoutez les modifications ou directives suivantes au fichier, enregistrez les modifications et démarrez le serveur Redis.

Tout d'abord, trouvez et supprimez le caractère # de la ligne **slaveof**. Cette directive prend l'adresse IP et le port que vous utilisez pour contacter en toute sécurité le serveur principal de Redis, séparés par un espace. Par défaut, le serveur Redis écoute le 6379 sur l'interface locale, mais chacune des méthodes de sécurité du réseau change la valeur par défaut d'une manière ou d'une autre pour les autres.

Les valeurs que vous utilisez dépendront de la méthode que vous avez utilisée pour protéger le trafic de votre réseau :

Réseau isolé : utilisez l'adresse IP du réseau isolé et le port Redis (6379) du serveur maître (par exemple, esclave de l'adresse IP simple 6379).

stunnel ou spiped : utilisez l'interface locale (127.0.0.1) et le port configuré pour faire passer le trafic par un tunnel

PeerVPN : Utilisez l'adresse IP VPN du serveur maître et le port Redis normal.

Le général le changerait :

slaveof ip_contact_server master_contact_port

Exemple : **esclave de** 192.168.1.69 6379

masterauth votre_mot_de_passe_de_réseau

Exemple : **masterauth** sdfssdfsd12WqE34Rfgthtdf

requirepass your_network_slave_password

Exemple : **requirepass** asdsjdsh34sds67sdFGbbnh

Sauvegardez et exécutez le serveur depuis le terminal Termux avec la commande suivante.

\$ redis-server redis.conf

Après l'exécution, nous pouvons voir comment il se synchronise avec le serveur Windows 10 (**Master**).

Fichier de configuration redis.conf \$ redis-server redis.conf

```
$ pwd
/data/data/com.termux/files/usr/etc
$ ls
alternatives      inputrc    redis.conf
apt              krb5.conf   ssh
bash.bashrc       motd       tls
bash_completion.d profile    tmux.conf
dump.rdb          profile.d wgetrc
$
```

```
32672:S 31 May 2020 23:50:24.130 # Server initialized
32672:S 31 May 2020 23:50:24.131 * Loading RDB produced by version 6.0.1
32672:S 31 May 2020 23:50:24.131 * RDB age 27 seconds
32672:S 31 May 2020 23:50:24.131 * RDB memory usage when created 0.39 Mb
32672:S 31 May 2020 23:50:24.132 * DB loaded from disk: 0.001 seconds
32672:S 31 May 2020 23:50:24.132 * Ready to accept connections
32672:S 31 May 2020 23:50:24.132 * Connecting to MASTER 192.168.1.69:6379
32672:S 31 May 2020 23:50:24.136 * MASTER <-> REPLICIA sync started
32672:S 31 May 2020 23:50:24.159 * Non blocking connect for SYNC fired the event.
32672:S 31 May 2020 23:50:24.166 * Master replied to PING, replication can continue...
32672:S 31 May 2020 23:50:24.236 * Partial resynchronization not possible (no cached master)
32672:S 31 May 2020 23:50:24.266 * Full resync from master: 8ea52fe3c02ae241292f0dcbb823b8febcb9b784:0
32672:S 31 May 2020 23:50:24.349 * MASTER <-> REPLICIA sync: receiving 578 bytes from master to disk
32672:S 31 May 2020 23:50:24.353 * MASTER <-> REPLICIA sync: Flushing old data
32672:S 31 May 2020 23:50:24.353 * MASTER <-> REPLICIA sync: Loading DB in memory
32672:S 31 May 2020 23:50:24.354 * Loading RDB produced by version 5.0.7
32672:S 31 May 2020 23:50:24.354 * RDB age 0 seconds
32672:S 31 May 2020 23:50:24.354 * RDB memory usage when created 1.84 Mb
32672:S 31 May 2020 23:50:24.355 * MASTER <-> REPLICIA sync: Finished with success
```

Installation d'un outil de gestion de la synchronisation pour les nœuds. Nous procérons à l'installation de **SyncthingManager**.

Nous installons d'abord ce dont vous aurez besoin pour que SyncthingManager fonctionne correctement.

\$ apt install Python

\$ pip3 install -upgrade pip

\$ npm install syncthingmanager

L'outil SyncthingManager nous aidera à synchroniser "peer to peer" de manière automatique et non manuelle pour les nouveaux nœuds et les nœuds existants.

```
$ apt install python
$ pip3 install --upgrade pip
$ pip3 install syncthingmanager
```

17. Ambientes Blockly (App Inventor, AppyBuilder y Thunkable).

App Inventor est un environnement de développement logiciel créé par Google Labs pour créer des applications pour le système d'exploitation Android. L'utilisateur peut, visuellement et à partir d'un ensemble d'outils de base, relier une série de blocs pour créer l'application. Le système est gratuit et peut être facilement téléchargé sur le web. Les applications créées avec App Inventor sont très faciles à créer car aucune connaissance d'un langage de programmation n'est requise.

Tous les environnements actuels qui utilisent la technologie de Blockly tels que AppyBuilder et Thunkable entre autres ont leur version gratuite, leur mode d'utilisation peut se faire via Internet sur leurs différents sites ou il peut également être installé à la maison.

Les blocs qui composent l'architecture Mini BlocklyChain ont été testés dans App inventor et AppyBuilder mais, en raison de l'optimisation de leur code, ils devraient fonctionner sur les autres plateformes.

Versions en ligne :

App Inventor.

<https://appinventor.mit.edu/>

AppyBuilder.

<http://appybuilder.com/>

Possédable.

<https://thunkable.com/>

Version à installer sur votre ordinateur (PC) :

<https://sites.google.com/site/aprendeappinventor/instala-app-inventor>

Environnement pour les développeurs de blocs Blockly.

<https://editor.appybuilder.com/login.php>

18. Qu'est-ce que la preuve de quantum (PQu) ?

PoQu. - "Proof of Quantum" est un algorithme de consensus développé pour la Mini BlocklyChain, ce test est une variante du Test of Work (PoW) qui fonctionne comme suit.

Le Test of Quantum (PoQu) au démarrage est exécuté avec le même algorithme que le "Test of Work" (PoW) qui consiste à mettre le processeur de l'appareil (PC, serveur, tablette ou téléphone portable) au travail pour obtenir une chaîne de caractères qui est un puzzle mathématique appelé "hash".

N'oubliez pas qu'un "hachage" est un algorithme ou un processus mathématique qui, lors de l'introduction d'une phrase ou d'un type d'information numérique tel qu'un fichier texte, un programme, une image, une vidéo, un son ou tout autre type d'information numérique, nous donne comme résultat un caractère alphanumérique qui représente la signature numérique qui le représente de manière unique et non répétitive des données, l'algorithme de hachage est unidirectionnel, ce qui signifie que lorsque vous saisissez une donnée pour obtenir sa signature "hachage", son processus inverse ne peut pas être effectué, ayant une signature "hachage" nous ne pouvons pas savoir quelles informations ont été obtenues ; cette propriété nous donne un avantage de sécurité pour traiter les informations que nous envoyons sur Internet. Comment cela fonctionne-t-il ? Imaginez que vous envoyez n'importe quel type d'information par des canaux non sécurisés et que vous l'accompagnez de son "hachage source" respectif, le récepteur, lorsqu'il reçoit l'information, peut obtenir le "hachage" de l'information reçue ; nous l'appellerons "hachage destination" et nous le vérifierons avec le "hachage source" ; si les deux "hachages" sont identiques, nous pouvons confirmer que l'information n'a pas été altérée dans le canal qui a été envoyé, c'est juste un exemple où ce type de processus de sécurité de l'information est actuellement utilisé.

Actuellement, il existe différents types d'algorithmes ou de processus de hachage qui diffèrent par leur niveau de sécurité. Les plus utilisés ou les plus connus sont : MD5, SHA256 et SHA512.

Exemple de SHA256 :

Nous avons une chaîne ou une phrase comme suit: "Mini BlocklyChain" est modulaire. Si nous appliquons un hachage SHA256 à la chaîne précédente, cela nous donnera le hachage suivant.

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2afd827e8804db8

La chaîne alphanumérique ci-dessus est la signature qui représente la phrase dans l'exemple ci-dessus

Par exemple, nous pouvons utiliser le site sur Internet : <https://emn178.github.io/online-tools/sha256.html>

Dans le cas de l'algorithme "Test Work" (PoW), il fonctionne en utilisant la puissance de calcul pour obtenir un hachage prédéfini.

Imaginons que nous ayons le précédent "hash" que nous avons pris de la chaîne "Mini BlocklyChain is modular".

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2af827e8804db8

A ce "hachage" à son début on met le paramètre de difficulté qui est simplement de mettre des zéros "0" au début, c'est-à-dire que si on dit que la difficulté est de 4 il aura "**0000**" + "hachage" à cela on l'appellera "hachage de semence".

0000 f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db8

Maintenant, en tenant compte du fait que nous connaissons les informations d'entrée qui constituent la chaîne : "La Mini BlocklyChain est modulaire", nous ajoutons à la fin de la chaîne un nombre commençant à zéro "0" et nous en retirons son hachage, nous l'appellerons "hachage nonce" :

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db80

On a du haschisch nonce :

7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8

Ensuite, nous effectuons une comparaison du nouveau "hash nonce" avec le "hash seed" ; s'ils sont égaux, le nœud qui trouve le premier l'égalité gagnera l'exécution du traitement de la transaction en cours. Comme nous pouvons le voir, ce processus est basé sur la probabilité et la puissance de calcul du dispositif, ce qui donne au test de "Preuve du travail" une équité consensuelle pour tous les nœuds.

Si le "hachage de semence" ne coïncide pas avec le "hachage nonce", la difficulté est augmentée de un et le "hachage nonce" est à nouveau supprimé, le nombre qui est augmenté est appelé le nombre "nonce", il est comparé avec le "hachage de semence" jusqu'à ce qu'ils coïncident ou soient identiques.

Comme on peut le voir, le nombre "nonce" ou augmentation est celui qui permettra d'obtenir le "hash" de l'égalité.

Basé sur l'algorithme du "Test of Work" (PoW), l'algorithme du Test de Quantum (PoQu) est basé sur l'obtention du nombre "nonce" comme le fait le PoW et en utilisant un niveau de difficulté minimum allant de 1 à 5, cela sert uniquement à l'appareil mobile pour obtenir le droit d'être candidat pour gagner le consensus.

Le test quantique (PoQu), est activé lorsque le téléphone mobile a terminé le PoW minimum et gagne le passage pour obtenir un nombre de probabilité dans le système QRNG.

Le QRNG (Quantum Random Number Generator) est un générateur de nombres aléatoires quantiques, ce système est basé sur la génération de vrais nombres aléatoires basés sur la mécanique quantique est le système le plus sûr aujourd'hui pour générer de tels nombres. Pour plus de détails, voir l'annexe "Calculs quantiques avec OpenQbit".

La Mini BlocklyChain peut mettre en œuvre les deux types de concession minimum PoW et PoQu.

Le test PoQu est basé sur l'obtention du nombre "nonce" ; ce nombre dans le test PoQu est connu sous le nom de "Magic Number" ; le système "Peer to Peer" confirmera si le nombre est correct et un nombre aléatoire sera alors obtenu avec le pool de serveurs QRNG. Ce nombre aléatoire sera enregistré dans tous les nœuds, une liste sera créée contenant **((Node Sum /2)) +1** et de cette liste sera choisi celui qui a le plus haut pourcentage de probabilité d'être le candidat gagnant du consensus (PoQu) et celui-ci exécutera la file d'attente des transactions en cours.

L'algorithme PoQu utilise également les tests du **NIST** (National Institute of Standards and Technology) pour nous assurer que les nombres aléatoires dans le QRNG sont vraiment des nombres aléatoires.

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>

Dans Mini BlocklyChain, nous avons mis en place un bloc pour PoW et un bloc pour PoQu.

Ces blocs utilisent un type de hachage : SHA256 pour une utilisation gratuite, pour une utilisation commerciale vous avez un SHA512 et d'autres types de hachage selon les besoins.

Pour plus de détails sur le concept de HASH, voir :

https://es.wikipedia.org/wiki/Funcion_hash

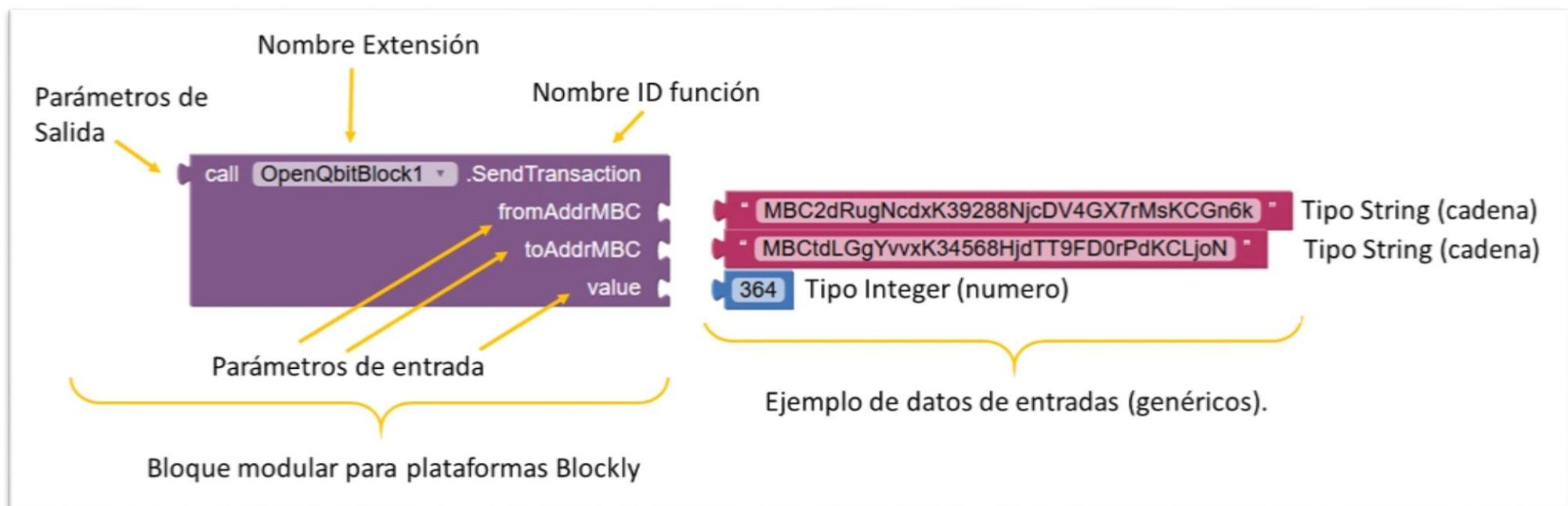
REMARQUE : le Test of Work (PoW) utilisé dans les téléphones mobiles ne peut utiliser qu'une difficulté maximale de 5 car le traitement mathématique de ces appareils n'est pas dédié comme les serveurs ou les PC. Nous utilisons uniquement l'algorithme PoW pour obtenir la possibilité d'obtenir votre laissez-passer ou la permission d'entrer dans le système du générateur quantique de nombres aléatoires (QRNG) et avec lui pour exécuter l'algorithme du générateur quantique de nombres aléatoires (PoQu).

Sur les téléphones portables, n'utilisez pas une difficulté maximale de 5 car le système peut se bloquer et ne pas répondre correctement.

19. Définition et utilisation des blocs dans la Mini BlocklyChain

Nous commencerons par expliquer la répartition des données que tous les blocs auront, leur syntaxe d'utilisation et leur configuration.

Dans l'exemple suivant, nous pouvons voir un bloc modulaire et ses paramètres d'entrée et de sortie, ainsi que les types de données d'entrée, ces données peuvent être de type String (chaîne de caractères) ou Integer (entier ou décimal). Nous montrons comment il est utilisé et le configurons pour son bon fonctionnement.



Chaque bloc de module aura sa description et sera nommé au cas où il aurait une ou plusieurs dépendances obligatoires ou facultatives d'autres blocs utilisés comme paramètres d'entrée, le processus d'intégration sera annoncé.

Bloc pour créer une liste temporaire de chaînes de caractères dans un tableau prédéfini appelé en interne "chaîne" - (**AddHash**).

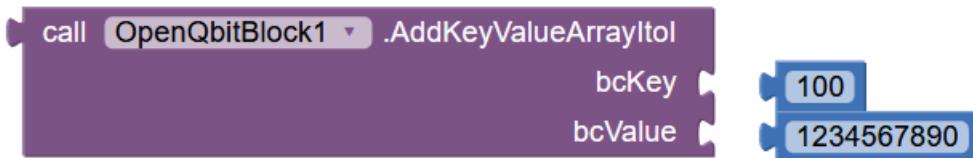


Paramètres d'entrée : **bloc** <string>

Paramètres de sortie : Non applicable.

Description : Bloc pour stocker un tableau temporaire de hachages ou de chaînes de caractères dans un tableau prédéfini appelé "chaîne" interne.

Bloc pour créer des tableaux de valeurs clés (**Integer-Integer**) - (**AddKeyValueArrayItol**)

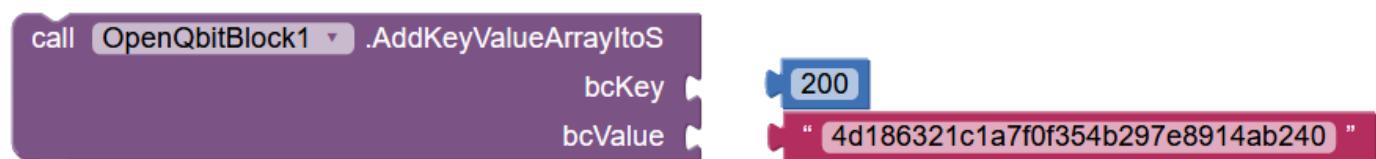


Paramètres d'entrée : **bcKey** <Integer>, **bcValue** <Integer>

Paramètres de sortie : renvoie les valeurs saisies à l'entrée.

Description : Il s'agit d'un arrangement temporaire de clé-valeur, il est prédéfini avec le nom interne "Itol_UTXO" ; il est utile pour traiter les transactions UTXO.

Bloc pour créer des tableaux de valeurs clés (chaîne de caractères entiers) - (**AddKeyValueArrayItoS**)

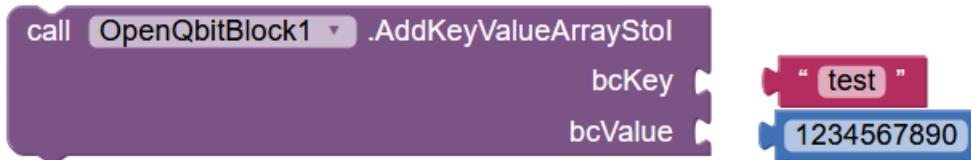


Paramètres d'entrée : **bcKey** <Integer>, **bcValue** <String>

Paramètres de sortie : renvoie les valeurs saisies à l'entrée.

Description : Il s'agit d'un arrangement temporaire de clé-valeur, il est prédéfini avec le nom interne "ItoS_UTXO" ; il est utile pour traiter les transactions UTXO.

Bloc pour créer des tableaux de valeurs clés (**String-Integer**) - (**AddKeyValueArrayStol**)



Paramètres d'entrée : **bcKey <String>**, **bcValue <Integer>**

Paramètres de sortie : renvoie les valeurs saisies à l'entrée.

Description : Il s'agit d'un arrangement temporaire de clé-valeur, il est prédéfini avec le nom interne "**Stol_UTXO**" ; ceci est utile pour traiter les transactions UTXO.

Bloc pour créer des tableaux de valeurs clés (**String-String**) - (**AddKeyValueArrayStoS**)



Paramètres d'entrée : **bcKey <String>**, **bcValue <String>**

Paramètres de sortie : renvoie les valeurs saisies à l'entrée.

Description : Il s'agit d'un arrangement temporaire de clé-valeur, il est prédéfini avec le nom interne "**StoS_UTXO**" ; ceci est utile pour traiter les transactions UTXO.

Bloc pour la génération de nombres quantiques aléatoires décimaux - (**ApiGetQRNGdecimal**)



Paramètres d'entrée : **qté < Entier>**

Paramètres de sortie : donne la quantité "qté" de nombres décimaux quantiques aléatoires entrés dans les numéros d'entrée sont dans la gamme de 0 et 1 dans le format JSON.

Exemple :

```
qté = 5 ; sortie : {"résultat" : [0,5843012986202495, 0,7746497687824652, 0,05951126805960929, 0,1986079055812694, 0,03689783439899279]}
```

Description : API du générateur quantique de nombres aléatoires (QRNG)

Bloc pour la génération de nombres quantiques aléatoires décimaux - (**ApiGetQRNGdécimal**)



Paramètres d'entrée : **qty <Intérieur>**, min <Intérieur>, max <max>

Paramètres de sortie : donne la quantité "qté" d'entiers quantiques aléatoires entrés dans l'entrée ; les nombres sont compris entre min et max dans le format JSON.

Exemple :

qty = 8, min = 1, max = 100 ; sortie : {"résultat" : [3, 53, 11, 2, 66, 44, 9, 78]}

Description : API du générateur quantique de nombres aléatoires (QRNG)

Bloc de hachage "SHA256" pour les chaînes de caractères (**AppliquerSha256**).



Paramètres d'entrée : **entrée <Corde>**

Paramètres de sortie : calcule le hachage "SHA256" d'une chaîne de caractères.

Exemple :

Entrée = "La Mini BlocklyChain est modulaire"

sortie : f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db8

Description : Fonction permettant de supprimer le hachage "SHA256". *Sha* ou *SHA* font référence : Secure Hash Algorithm, un ensemble de fonctions de hachage conçu par l'Agence de sécurité nationale des États-Unis. Le SHA256 utilise un algorithme de 256 bits.

Bloc pour nettoyer la "chaîne" de tableaux internes prédéfinis (**ClearBlockList**).

call **OpenQbitBlock1** .**ClearBlockList**

Paramètres d'entrée et de sortie : Sans objet

Description : Supprimez tous les éléments qui ont la "chaîne" interne d'arrangement temporel.

Bloc pour nettoyer le tableau interne prédéfini (**Integer-Integer**) - (**ClearItol**).

call **OpenQbitBlock1** .**ClearItol**

Paramètres d'entrée et de sortie : Sans objet

Description : Supprimer tous les éléments qui ont l'arrangement temporaire interne "Itol_UTXO".

Bloc pour nettoyer le tableau interne prédéfini (**Integer-String**) - (**ClearItoS**).

call **OpenQbitBlock1** .**ClearItoS**

Paramètres d'entrée et de sortie : Sans objet

Description : Supprimer tous les éléments qui ont l'arrangement temporaire interne "ItoS_UTXO".

Bloc pour nettoyer le tableau interne prédéfini (**String-Integer**) - (**ClearStol**).

call **OpenQbitBlock1** .**ClearStol**

Paramètres d'entrée et de sortie : Sans objet

Description : Supprimer tous les éléments qui ont l'arrangement temporaire interne "Stol_UTXO".

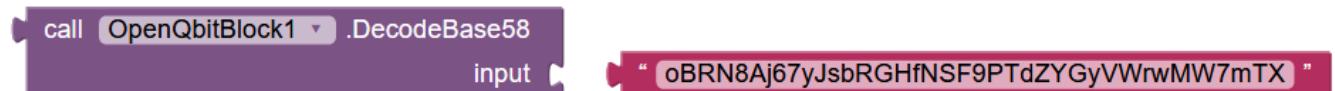
Bloc pour nettoyer le tableau interne prédéfini (**String-String**) - (**ClearStoS**).

call **OpenQbitBlock1** .**ClearStoS**

Paramètres d'entrée et de sortie : Sans objet

Description : Supprimer tous les éléments qui ont l'arrangement temporaire interne "StoS_UTXO".

Bloc pour décoder une chaîne vers la Base10. (**DecodeBase58**)



Paramètres d'entrée : **input<String>**

Paramètres de sortie : Il fournit la chaîne originale qui a été utilisée dans le bloc (**EncodeBase58**).

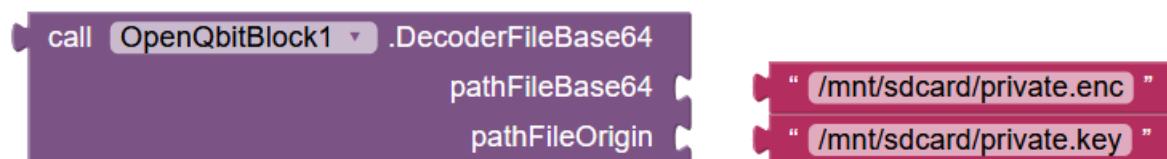
Exemple :

Entrée = oBRN8Aj67etJsbRGHfNSF9PTdZYGetVWrwMW7mTX

Résultat : "Mini BlocklyChain est modulaire".

Description : Convertit une chaîne Base58 au texte original donné dans le bloc (**EncodeBase58**)

Bloc pour décoder un fichier avec l'algorithme Base64 (**DecoderFileBase64**).



Paramètres d'entrée : **pathFileBase64 <String>**, **pathFileOrigin <String>**

Paramètres de sortie : Fichier source qui a été saisi dans le bloc (**EncoderFileBase64**)

Description : un fichier Base64 est converti en fichier original qui a été inséré dans le bloc (**EncoderFileBase64**).

Bloc pour savoir si le tableau interne prédéfini "chaîne" est vide. (**Liste de blocage vide**)



Paramètres d'entrée : Non applicable.

Paramètres de sortie : Retourne "Vrai" si vide ou retourne "Faux" si vous avez des données.

Description : Bloc permettant de demander si la "chaîne" de tableaux internes temporaires prédéfinis comporte des éléments.

Bloc pour encoder une chaîne de caractères en Base58. ([EncodeBase58](#)).



Paramètres d'entrée : `input<String>`

Paramètres de sortie : Il fournit la chaîne originale qui a été utilisée dans le bloc ([EncodeBase58](#)).

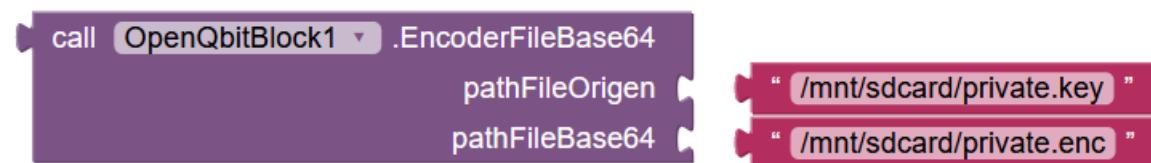
Exemple :

Entrée = "La Mini BlocklyChain est modulaire".

Notre contribution : oBRN8Aj67yJsbRGHfNSF9PTdZYGyVWrwMW7mTX

Description : Convertit une chaîne de poitrine en une chaîne en Base58. L'algorithme Base58 est un groupe de schémas de codage binaire à texte utilisés pour représenter de grands nombres entiers sous forme de texte alphanumérique, introduit par Satoshi Nakamoto pour être utilisé avec Bitcoin.

Bloc pour l'encodage d'un fichier avec l'algorithme Base64 ([EncoderFileBase64](#)).



Paramètres d'entrée : `pathFileOrigin <String>`, `pathFileBase64 <String>`

Paramètres de sortie : Fichier encodé en Base64.

Description : Convertit un fichier source de n'importe quel format en un fichier Base64. Les noms de fichiers peuvent être arbitraires et choisis par l'utilisateur.

Générateur de blocs d'adresses d'utilisateurs (**GenerateAddrBitcoin**).



Unité obligatoire : Bloc (**ApiGetQRNGinteger**),

Dépendances (facultatif) : extension **OpenQbitFileEncription**, extension **OpenQbitFileDecryption** et extension **OpenQbitSQLite**.

Paramètres d'entrée : **qrng** < dépendance obligatoire>

Paramètres de sortie : adresse de transaction alphanumérique de 34 caractères et adresse d'utilisation du keyStore

Description : Bloc permettant de créer une nouvelle adresse de transaction générique Bitcoin pour l'utilisateur et le générateur de clé privée (signature numérique pour l'envoi de transactions) et de clé publique (adresse publique pour effectuer des transactions). Ce générateur de clés est essentiellement le générateur d'adresses à utiliser dans un portefeuille numérique ou communément appelé "Wallet".

Ce bloc est utilisé en conjonction avec les blocs du générateur quantique de nombres aléatoires (QRNG) et les deux paramètres de sortie doivent être insérés dans le KeyStore.

KeyStore est une base de données qui stocke des clés privées au format hexadécimal :

Exemple d'une adresse hexadécimale stockée dans le KeyStore

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

Cette base est utilisée uniquement par l'utilisateur local et les informations sont cryptées, ce processus se fait par l'utilisation de l'extension OpenQbitSQLite et des extensions de cryptage des informations OpenQbitAESEncryption et OpenQbitAESDecryption.

Pour créer un KeyStore, voir l'annexe "Création d'un KeyStore". Les adresses générées utilisent le même algorithme d'adresse Bitcoin, l'identificateur d'adresse Bitcoin initial étant "1".

Les adresses générées par le bloc (**GenerateAddrBitcoin**) sont composées de 34 caractères alphanumériques et de 1 de l'identifiant Bitcoin comme suit :

12dRugNcdxK39288NjcDV4GX7rMsKCGn6k

Bloc générateur d'adresses utilisateur (**GenerateAddrEthereum**).



Dépendance obligatoire : Obtenez un jeton à l'adresse suivante : <https://accounts.blockcypher.com/signup>

Dépendances (facultatif) : extension **OpenQbitFileEncryption**, extension **OpenQbitFileDecryption** et extension **OpenQbitSQLite**.

Paramètres d'entrée : **token** < dépendance obligatoire - Chaîne de caractères>

Paramètres de sortie : l'adresse de transaction de 40 caractères alphanumériques ne comprend pas l'indicateur Ethereum initial "0x" qui nous donnerait les 42 caractères d'une adresse commune. Il rend également la clé publique et la clé privée.

Exemple :

```
{
  "privé": "227ac59f480131272003c2d723a7795ebd3580acaab62b5c537989e2ce4e08ef",
  "public":
  "04e2d55ebccd32a7384e096df559cc36b856c64a16e5b402e10585dc3ea055672aafa84df8
  a859531570a650a8ab1e7a22949100efa1aa5f072c035551cac1ce",
  "adresse": "14e150399b0399f787b4d6fe30d8b251375f0d66"}
```

Description : Bloc permettant de créer une nouvelle adresse de transaction pour l'utilisateur et le générateur de clé privée (signature numérique pour envoyer des transactions) et de clé publique (adresse publique pour effectuer des transactions). Ce générateur de clé est une API externe et vous devez avoir une connexion Wifi ou mobile, c'est en gros le générateur d'adresses pour l'utiliser dans un portefeuille numérique ou communément appelé "Wallet".

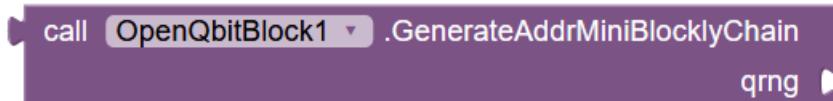
Vous pouvez créer aKeyStore est une base de données qui stocke des clés privées au format hexadécimal, voir l'annexe "Création de KeyStore".

Exemple d'une adresse hexadécimale stockée dans le KeyStore

0x14e150399b0399f787b4d6fe30d8b251375f0d66

La clé privée est utilisée uniquement par l'utilisateur local et les informations sont cryptées, ce processus se fait par l'utilisation de l'extension OpenQbitSQLite et des extensions de cryptage des informations OpenQbitAESEncryption et OpenQbitAESDecryption.

Bloc générateur d'adresses utilisateur (**GenerateAddrMiniBlocklyChain**).



Unité

obligatoire : Bloc (**ApiGetQRNGinteger**),

Dépendances (facultatif) : extension **OpenQbitFileEncription**, extension **OpenQbitFileDecryption** et extension **OpenQbitSQLite**.

Paramètres d'entrée : **qrng** < dépendance obligatoire>

Paramètres de sortie : adresse de transaction en 36 caractères alphanumériques et adresse d'utilisation du keyStore. Méthode des blocs d'examen (**PairKeysMBC**).

Description : Bloc permettant de créer une nouvelle adresse de transaction pour l'utilisateur et le générateur de clé privée (signature numérique pour envoyer des transactions) et de clé publique (adresse publique pour effectuer des transactions). Ce générateur de clés est en fait le générateur d'adresses à utiliser dans un portefeuille numérique ou communément appelé "Wallet".

Ce bloc est utilisé en conjonction avec les blocs du générateur quantique de nombres aléatoires (QRNG) et les deux paramètres de sortie doivent être insérés dans le KeyStore.

KeyStore est une base de données qui stocke des clés privées au format hexadécimal :

Exemple d'une adresse hexadécimale stockée dans le KeyStore

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

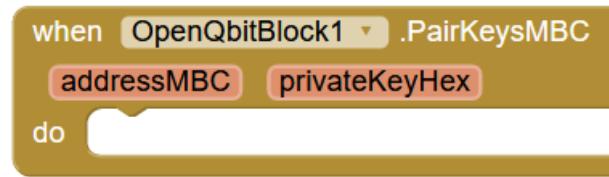
Cette base est utilisée uniquement par l'utilisateur local et les informations sont cryptées, ce processus se fait par l'utilisation de l'extension OpenQbitSQLite et des extensions de cryptage des informations OpenQbitAESEncryption et OpenQbitAESDecryption.

Pour créer un KeyStore, voir l'annexe "Création d'un KeyStore". Les adresses générées utilisent le même algorithme d'adresse bitcoin, la seule différence est que l'identifiant d'adresse bitcoin qui est "1" ou "3" est remplacé par l'identifiant Mini BlocklyChain "MBC".

Les adresses générées par le bloc (**GenerateAddrMiniBlocklyChain**) sont de 36 caractères alphanumériques et sont composées de 33 caractères alphanumériques et de 3 lettres majuscules de l'identifiant de la Mini BlocklyChain "MBC" comme suit :

Mbc2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

La méthode des blocs (**PairKeysMBC**) est une sortie de bloc (**GenerateAddrMiniBlocklyChain**).



Paramètres d'entrée : Non applicable.

Paramètres de sortie : **addressMBC <String>**, **privateKeyHex <String>**.

Description : Adresse publique de l'utilisateur au format Mini BlocklyChain et clé privée au format hexadécimal.

exemple :

adresseMBC : MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

privateKeyHex :

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

Générateur de blocs d'adresses d'utilisateurs (**GenerateKeyValuePair**).



Unité obligatoire : Bloc (**ApiGetQRNGinteger**),

Dépendances (facultatif) : extension **OpenQbitFileEncryption**, extension **OpenQbitFileDecryption** et extension **OpenQbitSQLite**.

Paramètres d'entrée : **qrng < dépendance obligatoire>**

Paramètres de sortie : fournit la clé publique et la clé primaire pour l'utilisateur local

Description : génère des clés publiques et primaires à l'aide de l'algorithme ECC (1) et au format hexadécimal.

- (1) La cryptographie à courbe elliptique (ECC) est une variante de la cryptographie asymétrique ou à clé publique basée sur les mathématiques des courbes elliptiques.

Bloquer pour signer les transactions (**GenerateSignature**).

```
call OpenQbitBlock1 .GenerateSignature
```

Dépendance requise : blocage (**GenerateKeyPais**), blocage (**SenderLoadKeyPair**), blocage (**ReciepientLoadKeyPair**). Préfacez ces blocs avant de les utiliser.

Paramètres d'entrée : < Dépendances de contrôle obligatoire>

Paramètres de sortie : Pas de sortie.

Description : il génère une signature numérique de l'expéditeur appliquée au bien envoyé dans la transaction au destinataire, cette signature sera confirmée lorsque la transaction est traitée par un nœud du réseau, avec cette signature le système Mini BlocklyChain garantit que le bien n'a pas été modifié dans son envoi et qu'il est conforme à la relation expéditeur-destinataire et n'est pas détourné ou appliqué à une autre adresse numérique.

Bloc pour obtenir le solde total des actifs d'un utilisateur (**GetBalance**).

```
call OpenQbitBlock1 .GetBalance  
fromAddrMBC " MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k "
```

Dépendance obligatoire : Blocage (**ConnectorTransactionTail**).

Paramètres d'entrée : **fromTransTail** <Chaîne de tableau>, < Dépendances obligatoires des tableaux>

Paramètres de sortie : vérifie les dépenses d'actifs entrantes et sortantes pour chaque transaction.

Description : Vérifie le solde pour approuver si l'utilisateur a des actifs à envoyer ou si les actifs qu'il reçoit sont ajoutés à son solde.

Bloquer pour obtenir le q22 du téléphone portable. (**GetDeviceID**)

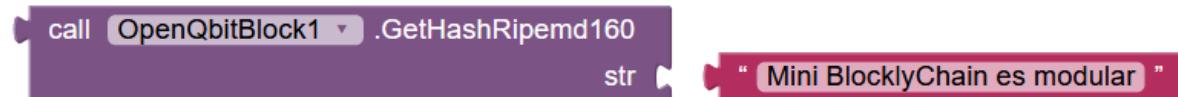
```
call OpenQbitBlock1 .GetDeviceID
```

Paramètres d'entrée : **Non applicable**

Paramètres de sortie : il fournit l'identifiant interne du téléphone mobile.

Description : Fournit l'identifiant IMEI de téléphone portable unique à chaque appareil.

Bloc pour supprimer le hachage RIPEMD-160 d'une chaîne. (**Ripemd160**)



Paramètres d'entrée : **str <String>**

Paramètres de sortie : calcule le hachage "RIPEMD-160" d'une chaîne de caractères.

Exemple :

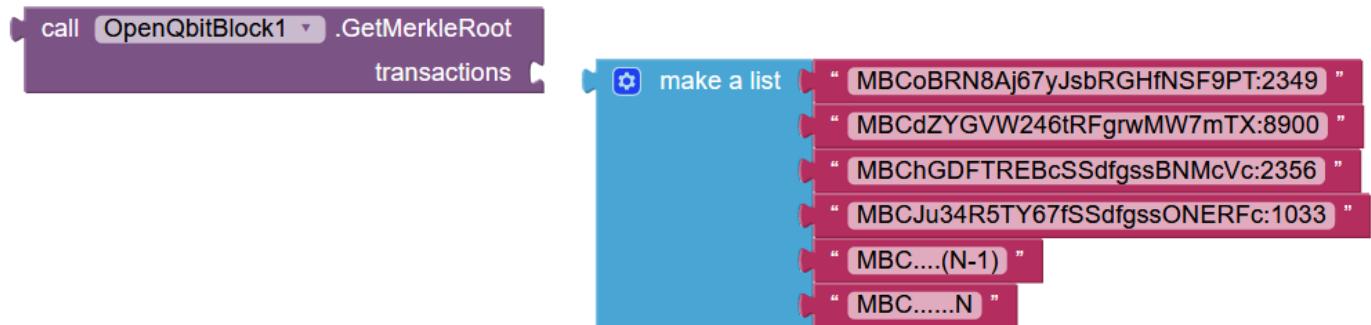
Entrée = "La Mini BlocklyChain est modulaire"

sortie : ae29436e4b8ea8ed6143f3f92380dfa2f4f47336

Description : Obtenir le hachage "RIPEMD-160". Ce hachage est utilisé dans le processus de génération d'une adresse valide pour Bitcoin et Mini BlocklyChain.

RIPEMD-160 (acronyme de RACE Integrity Primitives Evaluation Message Digest) est un algorithme de 160 bits de synthèse de messages (et une fonction de hachage cryptographique).

Bloc pour calculer l'arbre de Merkler. (**GetMerkleRoot**)



Paramètres d'entrée : **transactions <Array String>**

Paramètres de sortie : Il délivre un hachage de type "SHA256".

Exemple :

Entrée = file d'attente des transactions, un arrangement de toutes les transactions à traiter.

sortie : b4a44c42b6070825f763cd118d6ab49a8e80bbb7cdc0225064f8e042b94196bd

Description : Obtenir le hachage "SHA256" en utilisant l'algorithme de l'arbre de Merkle

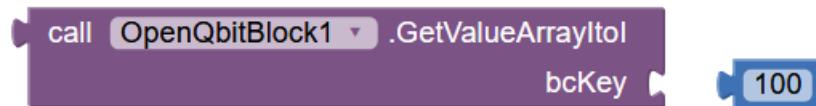
Un arbre de hachage Merkle est une structure d'arbre de données binaire ou non binaire dans laquelle chaque nœud qui n'est pas une feuille est étiqueté avec le hachage de la concaténation des étiquettes ou des valeurs (pour les nœuds feuilles) de ses nœuds enfants. Il s'agit d'une généralisation des listes de hachage et des chaînes de hachage.

Il permet de lier un grand nombre de données distinctes à une seule valeur de hachage, le hachage du nœud racine de l'arbre. Il s'agit d'une méthode sûre et efficace pour vérifier le contenu de grandes structures de données. Dans ses applications pratiques, le hachage du nœud racine est généralement signé pour garantir son intégrité et que la vérification est totalement fiable. Démontrer qu'un nœud de feuille fait partie d'un arbre de hachage donné nécessite une quantité de données proportionnelle au logarithme du nombre de nœuds dans l'arbre.

Actuellement, la principale utilisation des arbres Merkle est de sécuriser les blocs de données reçus d'autres pairs dans les réseaux peer-to-peer, pour s'assurer qu'ils sont reçus sans dommage et sans être altérés.

Il permet de vérifier qu'une file d'attente contenant les transactions à traiter n'a pas été modifiée et de s'assurer de son intégrité pour la dispersion dans tous les nœuds du réseau Mini BlocklyChain. Tous les nœuds doivent exécuter cet algorithme pour garantir l'intégrité de chaque transaction qui sera appliquée.

Bloc pour obtenir une valeur à partir du tableau prédéfini "Itol_UTXO" (**GetValueArrayItol**).

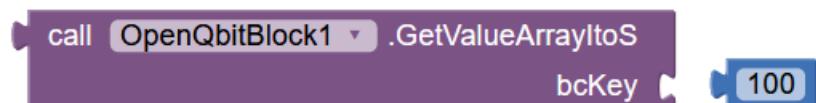


Paramètres d'entrée : **bcKey** <Integer>

Paramètres de sortie : renvoie la valeur <Intégrateur> stockée dans l'étiquette avec le numéro donné en entrée.

Description : Il s'agit d'une demande à l'arrangement temporaire de valeur clé, qui est prédéfini avec le nom interne "Itol_UTXO" ; ceci est utile pour traiter les transactions UTXO.

Bloc pour obtenir une valeur à partir du tableau prédéfini "Itos_UTXO" (**GetValueArrayItos**).



Paramètres d'entrée : **bcKey** <Integer>

Paramètres de sortie : renvoie la valeur <String> stockée dans l'étiquette avec le numéro donné en entrée.

Description : Il s'agit d'une demande à l'arrangement temporaire de valeur clé, qui est prédéfini avec le nom interne "**ItoS_UTXO**" ; ceci est utile pour traiter les transactions UTXO.

Bloc pour obtenir une valeur à partir du tableau prédéfini "Stol_UTXO" (**GetValueArrayStol**).



Paramètres d'entrée : **bcKey** < String >

Paramètres de sortie : renvoie la valeur <Intégrateur> stockée dans l'étiquette avec le prénom comme entrée.

Description : Il s'agit d'une demande à l'arrangement temporaire de valeur clé, qui est prédéfini avec le nom interne "**Stol_UTXO**" ; ceci est utile pour traiter les transactions UTXO.

Bloc pour obtenir une valeur à partir du tableau prédéfini "StoS_UTXO" (**GetValueArrayStoS**).



Paramètres d'entrée : **bcKey** < String >

Paramètres de sortie : renvoie la valeur <String> stockée dans l'étiquette avec le prénom en entrée.

Description : Il s'agit d'une demande à l'arrangement temporaire de valeur clé, qui est prédéfini avec le nom interne "**StoS_UTXO**" ; ceci est utile pour traiter les transactions UTXO.

Validateur de bloc de la chaîne de blocs. (**IsChainValid**)

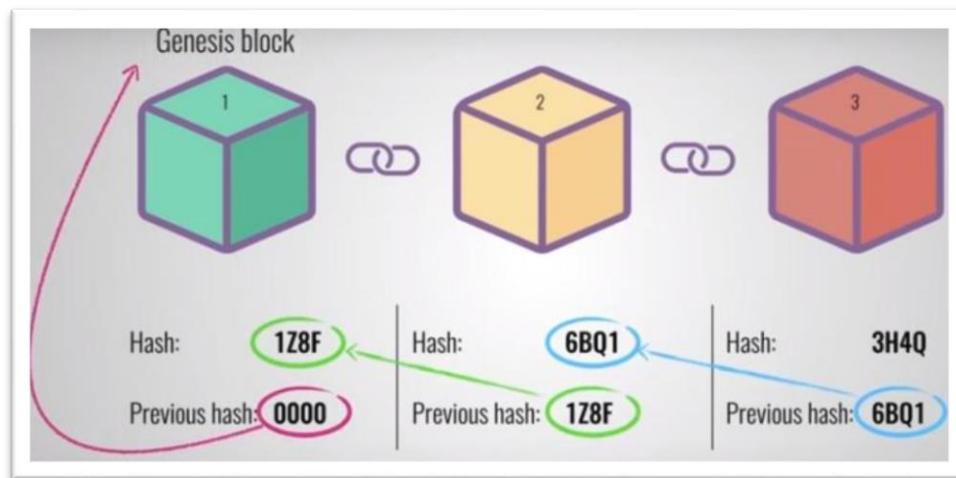


Dépendance requise : blocage (**GenerateKeyPairs**), blocage (**SenderLoadKeyPair**), blocage (**RecipientLoadKeyPair**), blocage (**GenerateSignature**). Préfacez ces blocs avant de les utiliser.

Paramètres d'entrée : Non applicable.

Paramètres de sortie : Livraison "True" si la validation de la chaîne de bloc est correcte ou "False" en cas d'échec de la validation.

Description : Il nous fournit la validation des composants qui ont été précédemment insérés dans le système de chaîne de blocs, cette validation est la plus importante de tout le système. Comment fonctionne la validation de la chaîne de blocs ou comment elle est communément connue de manière générique (BlockChain). Elle est la partie centrale de tout système.



Comme nous le voyons dans l'image précédente, chaque bloc ajouté dans le système de stockage est lié au bloc précédent par les algorithmes de hachage.

Par exemple, lors de la création d'un nouveau bloc à ajouter, le hachage représentant les nouvelles informations est obtenu en prenant le hachage du nouveau bloc d'informations + le hachage précédent. Avec ce type de maillon, toute modification minimale dans le stockage des chaînes de blocs sera détectée, ce qui permet une sécurité des données très élevée et efficace.

Voici un exemple de la manière dont une chaîne de blocs est stockée dans une base de données SQLite, nous verrons comment le hachage précédent est lié au nouveau hachage du dernier bloc (dernière file d'attente des transactions traitées) qui a été inséré. Chaque hachage dans les colonnes "prevhash" et "newhash" représente les informations de la file d'attente des transactions qui ont été traitées à ce moment.

SQLite Expert Personal 5.3 (x64)

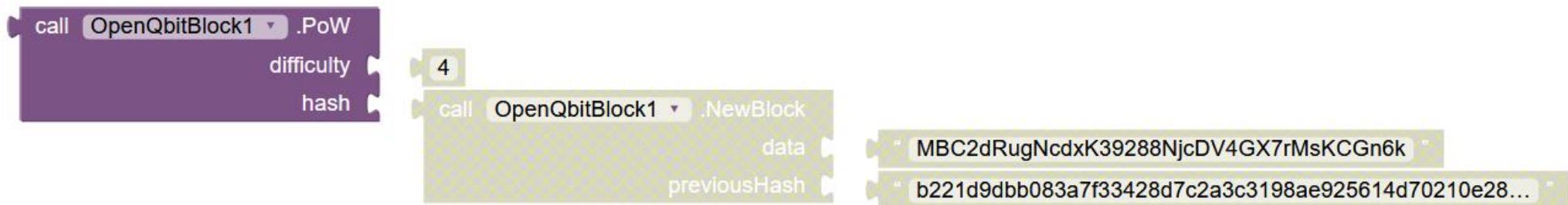
File View Database Object SQL Transaction Tools Help

Database: miniblock Table: nblock File: C:\memo\thunkable\miniblock.db

rowid	id	prevhash	newhash	opera	trans	balan
1	1	0767c864cef0334f27473902eb9868e7	bdc9065d20a4cd037bb1a7538486403e	2	0	0
2	2	bdc9065d20a4cd037bb1a7538486403e	6619f4809d73a267a4b9ac554bb4523a	1	5	5
3	3	6619f4809d73a267a4b9ac554bb4523a	4d186321c1a7f0f354b297e8914ab240	1	5	5

Le hash précédent est lié au nouveau hash.
Ils sont toujours les mêmes.

Bloc pour effectuer le test de travail de PoW et pour exploiter de nouveaux blocs. (**Pow**)



Dépendance obligatoire : **NewBlock**. Préfacez ce bloc avant de l'utiliser.

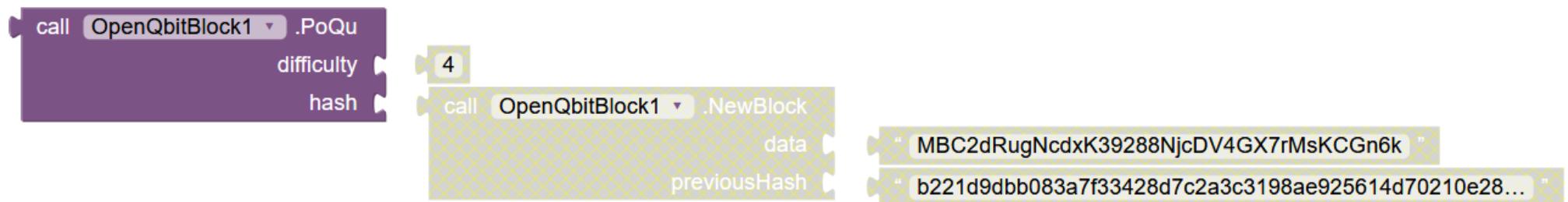
Paramètres d'entrée : **difficulté** <Intégration>, hachage <Dépendance obligatoire>

Paramètres de sortie : Donne en sortie un hachage "SHA256" composé du #Nombre de "zéros" donné dans la difficulté du paramètre d'entrée.

Description : Ce bloc effectue une activité appelée "exploitation minière" ; un nouveau bloc est ce que nous avons précédemment décrit comme le processus de Preuve du travail (PoW), voir la section Qu'est-ce que la Preuve du travail (PQu) ?

Dans le cas de la Mini BlocklyChain, la première personne qui résout le puzzle a la possibilité de poursuivre le processus afin d'être choisie pour être le gagnant de la file d'attente des transactions en attente de traitement.

Bloc pour effectuer le test de travail de PoW et pour exploiter de nouveaux blocs. (**PoQu**)



Dépendance obligatoire : **NewBlock**. Préfacez ce bloc avant de l'utiliser.

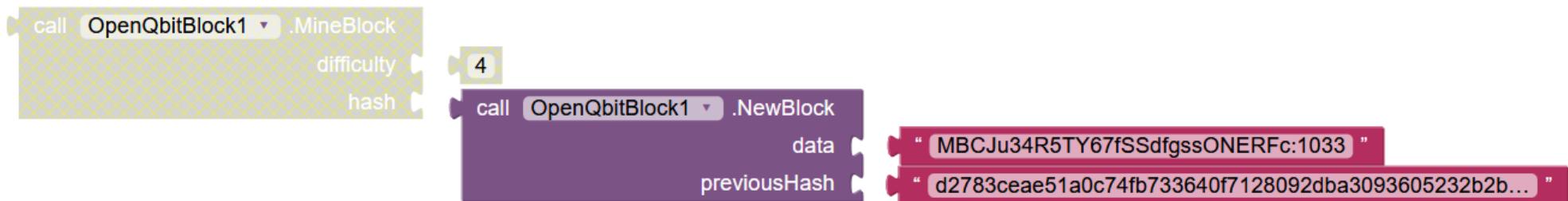
Paramètres d'entrée : **difficulté** <Intégration>, hachage <Dépendance obligatoire>

Paramètres de sortie : donne le nombre "Magic Number" et le nombre quantique aléatoire qui se situe dans la plage de 0 et 1 type décimal à 16 chiffres, exemple (0. 5843012986202495) et un hachage "SHA256" composé avec le #Nombre de "zéros" donné dans la difficulté du paramètre d'entrée.

Description : Ce bloc effectue le processus d'"extraction" d'un nouveau bloc est ce que nous avons précédemment décrit comme le processus de PQ (Preuve du travail), mais ce processus appelle la fonction de nombre quantique aléatoire générée après le calcul du nombre "nonce" approprié pour répondre au niveau de difficulté qui peut être dans la gamme de 1 minimum, 5 maximum. Voir la section Qu'est-ce que la Preuve du quantum (PQu - Proof Quantum) ?

L'extraction ou l'exécution du PoW ou du PoQu est un concept dans lequel les nœuds sont chargés de résoudre un puzzle mathématique. Le nœud qui le résout en premier dans le cas d'un système de chaîne de blocs a la possibilité de poursuivre le processus afin d'être choisi pour être le gagnant de la possibilité de traiter la file d'attente des transactions qui attendent d'être traitées.

Bloc pour la création de **nouveaux** blocs (**NewBlock**).

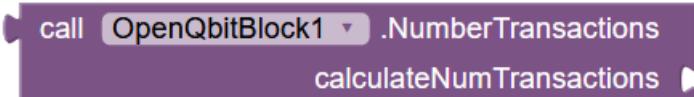


Paramètres d'entrée : **données** <Chaîne>, **previousHash** <Chaîne>

Paramètres de sortie : Il fournit un hachage calculé comme suit : **SHA256 (données (paramètres d'entrée) + previousHash (paramètre d'entrée) + IMEI (paramètre interne) + MerkleRoot (paramètre interne))**.

Description : Ce bloc effectue la création d'un nouveau bloc à traiter. Il donne en sortie un hachage "SHA256" composé de la chaîne de paramètres d'entrée dans le cas de données variant selon la conception de chaque système et le hachage précédent est basé sur le hachage de la chaîne de transaction précédente qui a déjà été traitée et est stockée dans le système de chaîne de bloc Mini BlocklyChain, ces deux paramètres sont variables, il y a deux paramètres internes au système qui sont fixes et qui assurent l'intégrité de l'information et du système, ces deux paramètres internes sont l'identifiant unique du téléphone portable et le hachage de l'arbre de merklet de la file d'attente des transactions qui est en cours de traitement.

Bloc du total des transactions locales du nœud (**NumberTransactions**)



Dépendance obligatoire : Bloc (**AddKeyValueArrayStoS**). Préfacez ce bloc avant de l'utiliser.

Paramètres d'entrée : < Dépendance obligatoire>.

Paramètres de sortie : renvoie le total des transactions locales au nœud.

Description : Fournit le total des transactions qui seront appliquées uniquement aux comptes locaux du nœud.

Bloc pour lire l'adresse du destinataire dans un fichier binaire. (**RecieipientLoadKeyPair**)

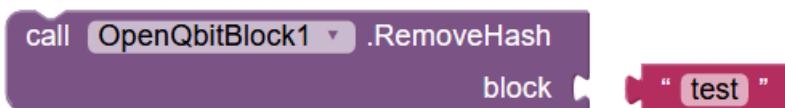


Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie la clé privée et la clé publique dans un format Base64.

Description : obtient l'adresse privée et publique du destinataire à partir d'un fichier binaire comme paramètre d'entrée.

Bloc pour supprimer un élément dans la "chaîne" de l'arrangement temporaire interne (**RemoveHash**).

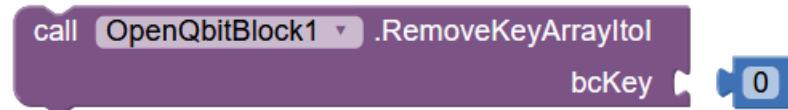


Paramètres d'entrée : **bloc < String >**.

Paramètres de sortie : Non applicable.

Description : obtient l'adresse privée et publique du destinataire à partir d'un fichier binaire comme paramètre d'entrée.

Bloc pour supprimer un élément de l'arrangement temporaire interne "Itol_UTXO" (**RemoveKeyArrayItol**)

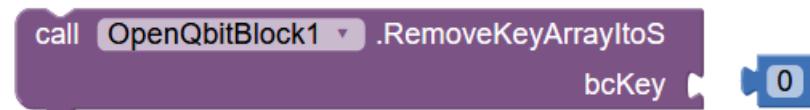


Paramètres d'entrée : **bcKey < Integer >**.

Paramètres de sortie : Non applicable, type d'élément supprimé <Intégrer>.

Description : l'élément associé à l'étiquette numérique de l'arrangement temporaire interne "Itol_UTXO" est supprimé.

Bloc pour supprimer l'élément dans l'arrangement temporaire interne "ItoS_UTXO" (**RemoveKeyArrayItoS**).

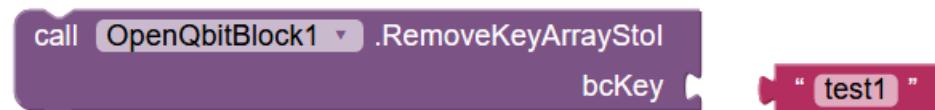


Paramètres d'entrée : **bcKey < Integer >**.

Paramètres de sortie : Sans objet, type d'élément à supprimer <Chaîne>.

Description : l'élément associé à l'étiquette numérique de l'arrangement temporaire interne "ItoS_UTXO" est supprimé.

Bloc pour la suppression d'un élément dans l'arrangement temporaire interne "Stol_UTXO" (**RemoveKeyArrayStol**)

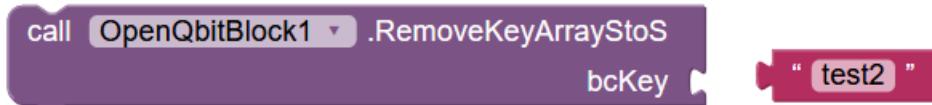


Paramètres d'entrée : **bcKey < String >**.

Paramètres de sortie : Non applicable, type d'élément supprimé <Intégrer>.

Description : l'élément associé à l'étiquette numérique de l'arrangement temporaire interne "Stol_UTXO" est supprimé.

Bloc pour la suppression de l'élément dans l'arrangement temporaire interne "StoS_UTXO" (RemoveKeyArrayStoS)

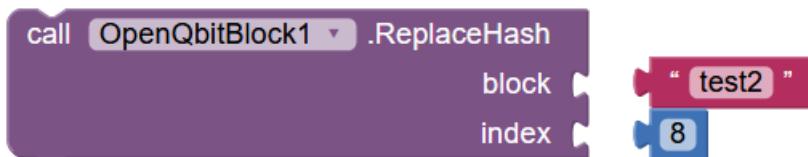


Paramètres d'entrée : **bcKey < String>**.

Paramètres de sortie : Sans objet, type d'élément à supprimer <Chaîne>.

Description : Supprimer l'élément de l'étiquette interne de l'arrangement temporaire "StoS_UTXO".

Bloc pour remplacer une valeur de la "chaîne" de l'arrangement temporaire interne (ReplaceHash).

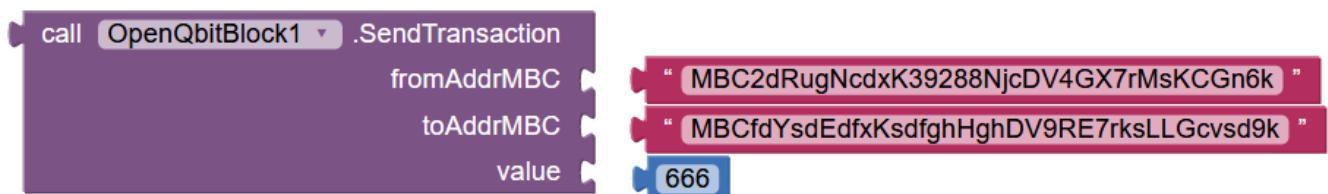


Paramètres d'entrée : **bloc < Chaîne>, index < Entier>**

Paramètres de sortie : Non applicable.

Description : l'élément associé à l'index "index" est remplacé par la valeur de l'étiquette "block" dans l'arrangement interne temporaire "chain".

Blocage pour démarrer (**envoyer**) une nouvelle transaction (SendTrasaction).



Paramètres d'entrée : **fromAddrMBC < String>, toAddrMBC < String>, value < Integer>**

Paramètres de sortie : renvoie la valeur "True" si la transaction a été envoyée avec succès ou "False" si elle n'a pas été envoyée avec succès.

Description : bloc qui transforme l'adresse de l'expéditeur et du destinataire en un format binaire et qui est attaché à la file d'attente des transactions à traiter.

Bloc pour lire l'adresse de l'expéditeur dans un fichier binaire. (**SenderLoadKeyPair**)

 call OpenQbitBlock1 .SenderLoadKeyPair

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie la clé privée et la clé publique dans un format Base64.

Description : permet d'obtenir l'adresse privée et publique de l'expéditeur à partir d'un fichier binaire comme paramètre d'entrée.

Bloc pour obtenir le numéro d'élément de la "chaîne" de l'arrangement temporaire (**SizeBlockList**).

 call OpenQbitBlock1 .SizeBlockList

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie le numéro d'élément de la "chaîne" de tableau interne.

Description : Obtenir le numéro d'élément de la "chaîne" de tableaux internes.

Bloc pour obtenir le numéro d'élément de l'arrangement temporaire "Itol_UTXO" (**SizeItol**)

 call OpenQbitBlock1 .SizeItol

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie le numéro d'élément du tableau interne "Itol_UTXO".

Description : Obtenir le numéro d'élément du tableau interne "Itol_UTXO".

Bloc pour obtenir le numéro d'élément de l'arrangement temporaire "ItoS_UTXO" (**SizeItoS**)

 call OpenQbitBlock1 .SizeItoS

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie le numéro d'élément du tableau interne "ItoS_UTXO".

Description : Obtenir le numéro d'élément du tableau interne "ItoS_UTXO".

Bloc pour obtenir le numéro d'élément de l'arrangement temporaire "StoI_UTXO" (**SizeStoI**)

 call **OpenQbitBlock1** .**SizeStoI**

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie le numéro d'élément du tableau interne "StoI_UTXO".

Description : Obtient le numéro d'élément du tableau interne "StoI_UTXO".

Bloc pour obtenir le numéro d'élément de l'arrangement temporaire "StoS_UTXO" (**SizeStoS**)

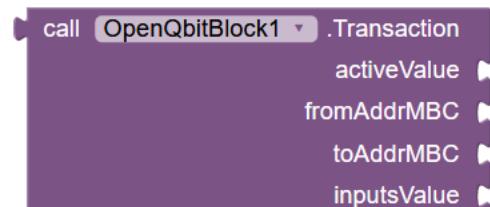
 call **OpenQbitBlock1** .**SizeStoS**

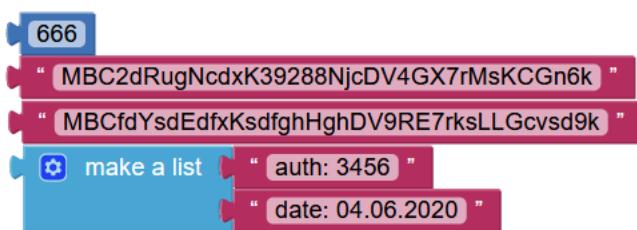
Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie le numéro d'élément du tableau interne "StoS_UTXO".

Description : Obtient le numéro d'élément du tableau interne "StoS_UTXO".

Bloge de la création d'une transaction avec des valeurs supplémentaires (**Transaction**).

 call **OpenQbitBlock1** .**Transaction**
 activeValue
 fromAddrMBC
 toAddrMBC
 inputsValue


 666
 " MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k "
 " MBCfdYsdEdfxKsfghHghDV9RE7rksLLGcvsd9k "
 make a list
 " auth: 3456 "
 " date: 04.06.2020 "

Paramètres d'entrée : **activeValue < Integer >**, **fromAddrMBC < String >**, **toAddrMBC < String >**, **inputsValue < Array String >**.

Paramètres de sortie : il nous donne les adresses en format binaire et la valeur inputValue convertie en une chaîne "String", et nous donne le hachage "SHA256" de la valeur ActiveValue.

Description : prépare une nouvelle transaction à traiter par les nœuds.

Mini BlocklyChain bloc de validation d'adresse (**ValiderAddMiniBlocklyChain**)

```
call OpenQbitBlock1 .ValidateAddrMiniBlocklyChain  
addrValidate " MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k "
```

Paramètres d'entrée : adresses des utilisateurs au format Mini BlocklyChain.

Paramètres de sortie : renvoie "True" si l'adresse est au bon format ou "False" si l'adresse n'est pas valide.

Description : Valide si l'adresse entrée dans la Mini BlocklyChain est correcte, ce bloc applique un algorithme pour vérifier si l'adresse a été créée en utilisant le mécanisme de création d'adresse à utiliser dans le système Mini BlocklyChain.

Bloc de validation des adresses Bitcoin. (**ValiderAddBitcoin**)

```
call OpenQbitBlock1 .ValidateAddressBitcoin  
addr " 12dRugNcdxK39288NjcDV4GX7rMsKCGn6k "
```

Paramètres d'entrée : **adr < String >**, adresses d'utilisateurs formatées en Bitcoin (accepte les adresses Bitcoin avec l'identifiant initial "1", les adresses avec l'identifiant "3" ne sont pas applicables).

Paramètres de sortie : renvoie "True" si l'adresse est au bon format ou "False" si l'adresse n'est pas valide.

Description : Valide si l'adresse Bitcoin saisie est correcte, ce bloc applique un algorithme pour vérifier si l'adresse a été créée en utilisant le mécanisme de création d'adresse à utiliser dans le système Bitcoin.

Bloc de vérification de la signature numérique pour la transaction en cours. (**VerifySignature**).

```
call OpenQbitBlock1 .VerifySignature
```

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : renvoie "Vrai" si le contrôle est valable ou "Faux" si le contrôle n'est pas valable.

Description : permet de vérifier la signature numérique que l'expéditeur aurait dû voir faire lors de l'envoi de la transaction que vous souhaitez effectuer. Dans ce processus de vérification, on vérifie que la valeur source envoyée n'a pas été altérée dans le canal où la transaction a été envoyée, et on vérifie le destinataire auquel la transaction doit être appliquée

20. Utilisation de blocs pour la base de données SQLite (version MiniSQLite)

Dans cette section, nous verrons comment utiliser les blocs pour effectuer deux opérations principales qui nous intéressent pour la fonctionnalité du système Mini BlocklyChain qui consiste à "INSÉRER" des données dans la chaîne de blocs et à les interroger.

NOTE : Les transactions de la base de données SQLite sont différentes des transactions envoyées par les nœuds pour être appliquées dans le système Mini Blocklychain.

Les transactions dans la base de données sont basées sur un modèle CRUD (Create, Read, Update and Delete) et sont les processus qui peuvent être exécutés avec des données externes ou internes uniquement dans la base de données SQLite. Dans la chaîne de blocs, les systèmes sont utilisés principalement les processus de création et de lecture, car la sécurité est écartée des processus de mise à jour ou de suppression et plus encore là où elle est stockée la chaîne de blocs qui donne la sécurité intégrale du système.

D'autre part, les transactions envoyées par les nœuds font référence à tous les processus qui impliquent l'action d'envoyer un certain type d'actif entre les membres (nœuds) du système Mini BlocklyChain, ce type de transactions comprend divers processus pour son application, ceux-ci peuvent être de la création d'une adresse numérique, une signature numérique, une validation de signature, un processus dans la base de données SQLite, entre autres processus.

Nous commencerons par la définition et l'utilisation des blocs de la base de données SQLite version MiniSQLite ; cette version n'est intégrée que par 8 blocs. Vous disposez d'une version complète pour manipuler les données dans la base de données SQLite, cependant, pour des raisons pratiques, le système Mini BlocklyChain avec la version MiniSQLite est suffisant.

Si vous souhaitez examiner tous les composants, leur utilisation et leur description, voir l'annexe "Blocs étendus pour la base de données SQLite".

Blocs de la version MiniSQLite :

Bloc pour démarrer une sorte de transaction dans la base de données SQLite (`BeginTransaction`)

call OpenQbitQSQLite1 .BeginTransaction

Unité(s) obligatoire(s) : Block (`ImportDatabase`), Block (`OpenDatabase`).

Paramètres d'entrée : à utiliser avant < Dépendance(s) obligatoire(s) >

Paramètres de sortie : Non applicable, il lance une transaction dans une base de données SQLite.

Description : Bloc qui lance un processus dans la base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

Bloc à faire Commit dans SQLite. (**CommitTrasaction**)

call OpenQbitQSQLite1 .CommitTransaction

Dépendance(s) requise(s) : Bloc (**BeginTransaction**), Bloc (**ImportDatabase**), Bloc (**OpenDatabase**).

Paramètres d'entrée : à utiliser avant < Dépendance(s) obligatoire(s) >

Paramètres de sortie : Non applicable, il effectue un **commit d'une** transaction dans une base de données SQLite.

Description : Bloc qui lance un processus de **commit sur** la base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

Bloc pour fermer la base de données SQLite qui a été importée ou exportée (**CloseDatabase**)

call OpenQbitQSQLite1 .CloseDatabase

Unité(s) obligatoire(s) : Block (**ImportDatabase**), Block (**OpenDatabase**).

Paramètres d'entrée : à utiliser avant < Dépendance(s) obligatoire(s) >

Paramètres de sortie : Non applicable, il ferme une base de données SQLite.

Description : Bloc qui ferme la base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

Bloc pour exporter une base de données SQLite (**ExportDatabase**).



Unité(s) obligatoire(s) : Block (**ImportDatabase**), Block (**OpenDatabase**).

Paramètres d'entrée : **fileName <String>** entrez un chemin où vous pouvez trouver une base de données déjà créée avec le format SQLite.

Utilisation avant < Dépendance(s) obligatoire(s) >

Paramètres de sortie : Exportation vers une base de données SQLite.

Description : Bloc qui exporte une base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

Bloc pour l'importation de la base de données SQLite. (**ImportDatabase**)

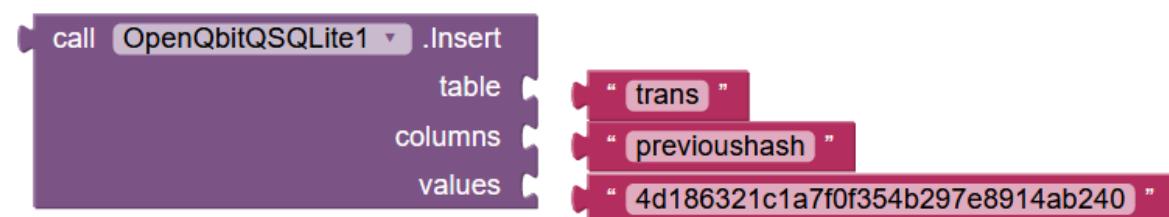


Paramètres d'entrée : **fileName <String>** entrez un chemin où vous pouvez trouver une base de données déjà créée avec le format SQLite.

Paramètres de sortie : lance une base de données SQLite pour exécuter les transactions.

Description : Bloc qui lance un processus dans la base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

Bloc pour insérer des données dans la base de données SQLite. (**Insérer**)



Unité(s) obligatoire(s) : Block (**ImportDatabase**), Block (**OpenDatabase**).

Paramètres d'entrée : **tableau < Chaîne>** , **colonnes < Chaîne>** , **valeurs < Chaîne>** , **Utilisation avant < Dépendance(s) obligatoire(s)**

Paramètres de sortie : Insère une transaction dans une base de données SQLite.

Description : Bloc qui insère des données dans la base de données SQLite, vous devez d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc ouvert de la base de données (**OpenDatabase**).

Bloc pour ouvrir la base de données SQLite (**OpenDatabase**)



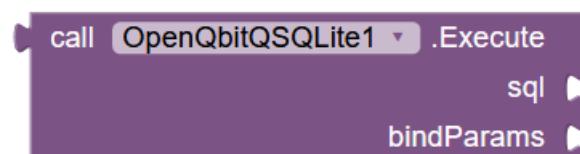
Unité(s) obligatoire(s) : Block (**ImportDatabase**).

Paramètres d'entrée : à **utiliser avant < Dépendance(s) obligatoire(s) >**

Paramètres de sortie : Sans objet, démarrer ou ouvrir une base de données SQLite pour effectuer des transactions.

Description : Bloc qui démarre une base de données SQLite, il doit y avoir avant.

Bloc pour la recherche de données dans SQLite (**Execute**).



Unité(s) obligatoire(s) : Block (**ImportDatabase**), Block (**OpenDatabase**).

Paramètres d'entrée : **sql < String>** , **bindParams < String>** , **Use before < Mandatory Dependency(s)>**

Paramètres de sortie : L'instruction SQL est exécutée pour créer une transaction dans une base de données SQLite.

Description : Bloc qui exécute des instructions SQL dans la base de données SQLite, doit d'abord avoir exécuté le bloc d'importation de la base de données (**ImportDatabase**) et le bloc d'ouverture de la base de données (**OpenDatabase**).

21. Définition et utilisation des blocs de sécurité.

Au cours de cette session, nous examinerons l'utilisation des blocs qui nous fournissent des niveaux de sécurité pour sauvegarder, valider et transférer des transactions à partir des nœuds du réseau Mini BlocklyChain.

Les blocs de sécurité sont basés sur l'extension suivante :

- I. Extension OpenQbitAESEncryption.
- II. Extension de décryptage OpenQbitAESD.
- III. Extension OpenQbitAEToString.
- IV. Extension OpenQbitEncDecData.
- V. Extension OpenQbitFileHash.
- VI. Extension OpenQbitRSA.
- VII. Extension OpenQbitSSHClient (nécessiterait)
- VIII. Extension OpenQbitStringHash.

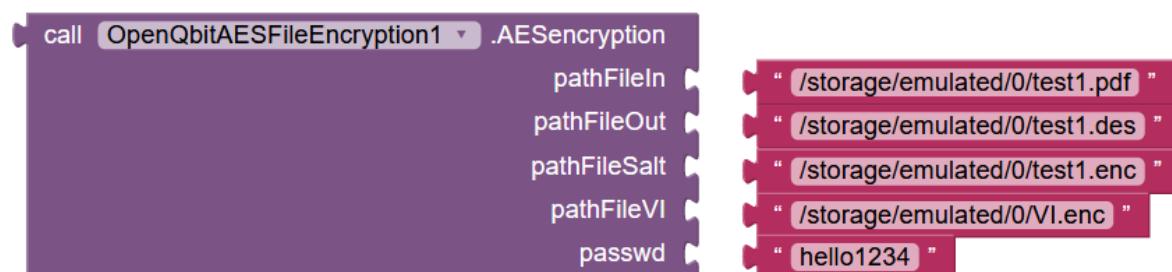
A l'exception de l'extension OpenQbitSSHClient qui est obligatoire, les extensions ci-dessus sont facultatives à utiliser dans la création d'un réseau public ou privé Mini BlocklyChain.

Toutefois, afin de disposer d'un système sécurisé pour vos transactions en fonction de chaque cas, l'utilisation des extensions qui sont facultatives doit être laissée à votre discrétion.

Par exemple, dans le cas de l'utilisation du hachage, nous pouvons utiliser un certain nombre d'options d'algorithme telles que MD5, SHA1, SHA128, SHA256, SHA512 pour une chaîne de caractères ainsi qu'être appliqué à tout type de fichier en fonction du flux d'informations de chaque système créé avec Mini BlocklyChain.

Extension OpenQbitAESEncryption.

Blocage pour crypter le fichier avec la sécurité AES. (**AESEncryption**).



Paramètres d'entrée : **pathFileIn < String>** , **pathFileOut < String>** , **pathFile < String>**, **pathFileVI < String>** et **passwd < String>**. Les noms de fichiers sont arbitraires et à la discréption de chaque conception de système.

Paramètres de sortie : fichier crypté AES saisi dans le paramètre d'entrée **pathFileIn**.

Description : Bloc qui nous donne trois fichiers de sortie dont l'un est le fichier source déjà crypté par l'algorithme AES avec une clé de 256 bits, les deux autres fichiers servent à contrôler l'extension pour décrypter et récupérer le fichier original.

Extension de décryptage OpenQbitAESD.

Bloc pour décrypter le fichier AES. (**AESDecryption**).



Paramètres d'entrée : **pathFileIn < String>** , **pathFileOut < String>** , **pathFile < String>**, **pathFileVI < String>** et **passwd < String>**. Les noms des fichiers sont les mêmes que ceux obtenus à la suite du blocage (**AESEncryption**).

Paramètres de sortie : **Fichier** original décrypté avec AES entré dans le paramètre d'entrée **pathFileIn**, dans ce cas pour décrypter le fichier il est entré dans **pathFileIn** le **fichier** crypté et dans **pathFileOut** il nous donnera le fichier original (décrypté).

Description : Bloc qui nous donne un fichier dans le paramètre **pathFileOut** qui sera la sortie décryptée par l'algorithme AES avec une clé de 256 bits.

Extension OpenQbitAESToString.

Cette extension est à usage unique par session d'appareil, c'est-à-dire qu'elle ne fonctionne que lorsque l'appareil n'est pas redémarré (téléphone portable), car la valeur de cryptage VI est générée temporairement.

REMARQUE : si l'appareil est redémarré, la chaîne cryptée ne peut pas être récupérée.

Bloc pour le cryptage de chaînes de caractères temporaires (**DecrypSecretText**)

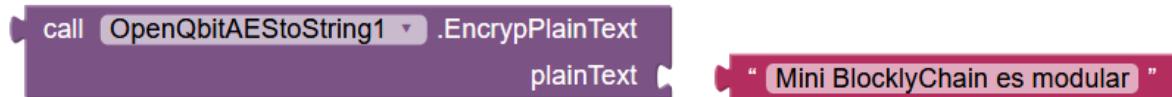


Paramètres d'entrée : **secretText** < String >

Paramètres de sortie : chaîne de caractères originale décryptée avec AES avec une clé de 256 bits

Description : Le bloc qui nous donne une chaîne de caractères, est le paramètre d'entrée qui a été introduit dans le bloc (**EncrypPlainText**).

Bloc pour décoder une chaîne de caractères (**EncrypPlainText**)



Paramètres d'entrée : **plainText** < String >

Paramètres de sortie : chaîne de caractères chiffrée en AES avec une clé de 256 bits.

Description : Bloc qui nous donne une chaîne de caractères alphanumériques chiffrée avec AES à l'aide d'une clé numérique de 256 bits.

Extension OpenQbitEncDecData.

Bloc de cryptage spécialisé pour les bases de données génériques (**EncryptionData**)



Paramètres d'entrée : **plainText** < String >

Paramètres de sortie : chaîne de caractères chiffrée en AES avec une clé de 256 bits.

Description : Bloc qui nous donne une chaîne de caractères alphanumériques chiffrée avec AES à l'aide d'une clé numérique de 256 bits.

Bloc de cryptage spécialisé pour les bases de données génériques (**DescriptionData**)



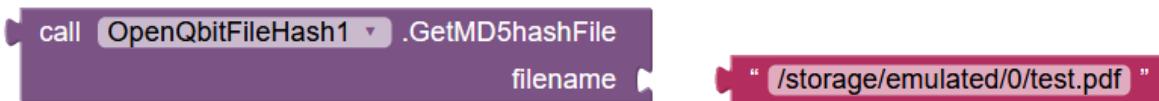
Paramètres d'entrée : **plainText** < String >

Paramètres de sortie : chaîne de caractères chiffrée en AES avec une clé de 256 bits.

Description : Bloc qui nous donne une chaîne de caractères alphanumériques chiffrée avec AES à l'aide d'une clé numérique de 256 bits.

Extension OpenQbitFileHash.

Bloc pour générer des MD5 à partir d'un fichier (**GetMD5hashFile**)

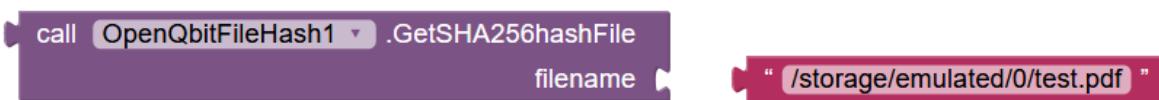


Paramètres d'entrée : **nom de fichier** <string>

Paramètres de sortie : Fournit le fichier de hachage MD5.

Description : Bloc permettant de créer le hachage MD5 du fichier indiqué dans le paramètre d'entrée.

Bloc pour générer SHA256 à partir d'un fichier (**GetSHA256hashFile**)

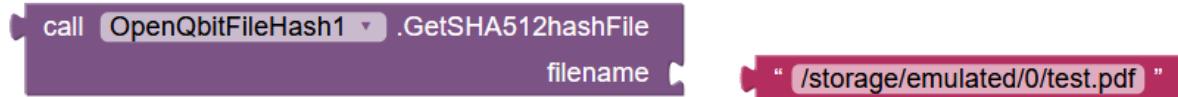


Paramètres d'entrée : **nom de fichier** <string>

Paramètres de sortie : Fournit le hachage d'archive SHA256.

Description : Bloc permettant de créer le hachage SHA256 du fichier indiqué dans le paramètre d'entrée.

Bloc pour générer SHA512 à partir d'un fichier (**GetSHA512hashFile**)

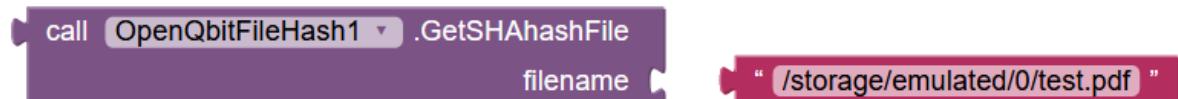


Paramètres d'entrée : **nom de fichier** <string>

Paramètres de sortie : Fournit le hachage d'archive SHA256.

Description : Bloc permettant de créer le hachage SHA256 du fichier indiqué dans le paramètre d'entrée.

Bloc pour générer SHA1 à partir d'un fichier (**GetSHA1hashFile**)



Paramètres d'entrée : **nom de fichier** <string>

Paramètres de sortie : Fournit le hachage des archives SHA1.

Description : Bloc permettant de créer le hachage SHA1 du dé dans le paramètre d'entrée.

Extension OpenQbitRSA.

Bloc pour **décrypter une** chaîne avec RSA (**Decrypt**)



Dépendance(s) requise(s) : Bloquer (**crypter**), bloquer (**OpenFromDiskPrivateKey**), bloquer (**OpenFromDiskPublicKey**).

Paramètres d'entrée : **résultat** < Chaîne de caractères>

Paramètres de sortie : Chaîne de caractères décodée avec RSA.

Description : Bloc qui nous donne une chaîne de caractères alphanumériques déchiffrée à l'aide d'une clé de la taille qui a été utilisée dans le bloc (**GenKeyPair**).

Bloc pour le cryptage de la chaîne avec RSA (**Encrypt**)



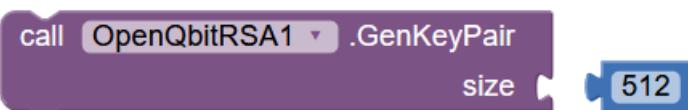
Unité(s) requise(s) : Bloc (**GenKeyValuePair**), Bloc (**SaveFromDiskPrivateKey**), Bloc (**SaveFromDiskPublicKey**)

Paramètres d'entrée : **simple** < Chaîne de caractères>

Paramètres de sortie : chaîne de caractères chiffrée RSA

Description : Bloc qui nous donne une chaîne de caractères alphanumériques déchiffrée à l'aide d'une clé de la taille qui a été utilisée dans le bloc (**GenKeyValuePair**).

Bloc pour **déchiffrer une** chaîne avec RSA (**Decrypt**)

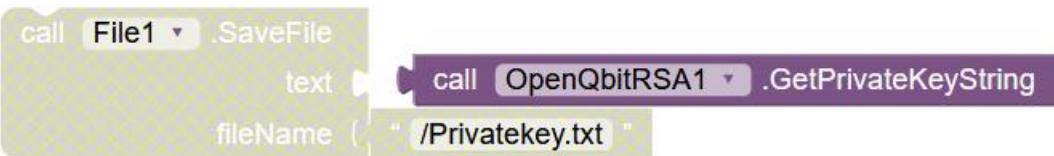


Paramètres d'entrée : **taille** < Entier>

Paramètres de sortie : Non applicable.

Description : Bloc permettant de générer une clé privée et une clé publique en fonction de la taille choisie.

Bloque pour obtenir la clé privée (**GetPrivateKeyString**)



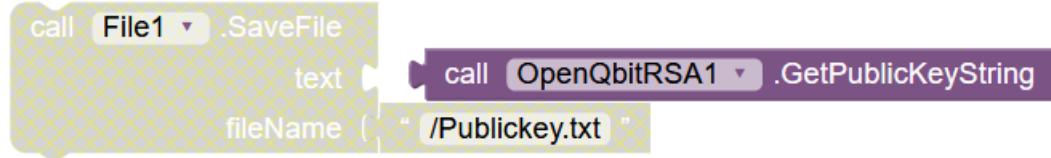
Unité(s) obligatoire(s) : Bloc (**GenKeyValuePair**), Bloc (**GenKeyValuePair**), Bloc (**File**)

Paramètres d'entrée : **Non applicable**.

Paramètres de sortie : Fichier avec chaîne de caractères codée RSA (clé privée)

Description : bloc qui nous donne une chaîne alphanumérique représentant la clé privée cryptée à l'aide de la clé de taille qui a été utilisée dans le bloc (**GenKeyValuePair**).

Blocage pour obtenir la clé privée (**GetPublicKeyString**)



Unité(s) obligatoire(s) : Bloc (**GenKeyPair**), Bloc (**GenKeyPair**), Bloc (**File**)

Paramètres d'entrée : **Non applicable**.

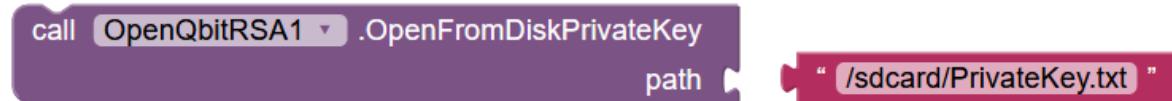
Paramètres de sortie : Fichier avec chaîne cryptée RSA (clé publique)

Description : bloc qui nous donne une chaîne alphanumérique représentant la clé publique chiffrée en utilisant la taille de la clé utilisée dans le bloc (**GenKeyPair**).

NOTE : Dans les blocs précédents (**GetPrivateKeyString**) et (**GetPublicKeyString**) dans les dépendances, nous utiliserons le bloc générique (**File**) de l'application App Inventor de la session palette "Storage".

Cette façon de stocker la clé privée et publique au moyen du bloc (**fichier**) peut nous aider à avoir une meilleure manipulation de l'information.

Blocage pour lire la clé privée d'un fichier (**OpenFromDiskPrivateKey**).



Dépendance(s) requise(s) : Bloquer (**SaveFromDiskPrivateKey**) ou Bloquer (**GetPrivateKeyString**).

Paramètres d'entrée : **chemin d'accès** < Chaîne de caractères>

Paramètres de sortie : Chargement du système Chaîne de caractères codée par clé privée RSA.

Description : Bloc qui nous donne une chaîne de caractères alphanumériques chiffrée de la clé privée stockée dans le chemin de fichier fourni.

Bloc pour lire la clé publique d'un fichier (**OpenFromDiskPublicKey**)



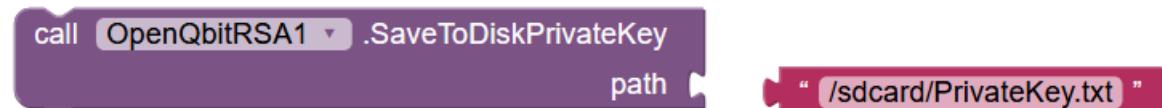
Unité(s) obligatoire(s) : Block (**SaveFromDiskPublicKey**) ou Block (**GetPublicKeyString**).

Paramètres d'entrée : **chemin d'accès** < Chaîne de caractères>

Paramètres de sortie : chargement dans le système d'une chaîne de caractères chiffrés à clé publique RSA.

Description : Bloc qui nous donne une chaîne alphanumérique cryptée de la clé publique stockée dans le chemin de fichier fourni.

Bloc pour la sauvegarde de la clé privée d'un fichier (**SaveToDiskPrivateKey**).



Unité(s) obligatoire(s) : Block (**GenKeyValuePair**).

Paramètres d'entrée : **simple** < Chaîne de caractères>

Paramètres de sortie : Fichier avec chaîne cryptée RSA. (clé privée)

Description : Bloc qui nous donne un fichier avec une chaîne alphanumérique cryptée à l'aide d'une clé privée de la taille utilisée dans le bloc (**GenKeyValuePair**).

Bloc pour la sauvegarde de la clé privée d'un fichier (**SaveToDiskPublicKey**).



Unité(s) obligatoire(s) : Block (**GenKeyValuePair**).

Paramètres d'entrée : **simple** < Chaîne de caractères>

Paramètres de sortie : Fichier avec chaîne cryptée RSA. (clé publique)

Description : Bloc qui nous donne un fichier avec une chaîne alphanumérique cryptée à l'aide d'une clé publique de la taille qui a été utilisée dans le bloc (**GenKeyValuePair**).

Extension OpenQbitSSHClient.

Connector Block SSH Client (**ConnectorMiniBlocklyChain**)

Paramètres d'entrée : nom d'utilisateur <string>, mot de passe <string>, hôte <string>, port<integer>

Paramètres de sortie : Si la connexion avec le serveur ssh du terminal Termux est réussie, il nous donne un message ; "**Connect SSH**", si elle n'est pas réussie, il nous donne un message **NULL**.

Description : Bloc de communication pour connecter la Mini BlocklyChain au terminal Termux, via le protocole de communication SSH (Secure Shell).

Bloc pour l'exécution des commandes dans le terminal Termux Linux



(**CommandLineMiniBlocklyChain**).

Paramètres d'entrée : commande <string>

Paramètres de sortie : données variables, en fonction de la commande ou du programme exécuté.

Description : bloc d'exécution de commandes dans le terminal Termux ; la condition préalable pour établir une connexion avec le bloc (**ConnectorMiniBlocklyChain**) permet d'exécuter toutes sortes de commandes en ligne et/ou d'obtenir des données d'exécution spécifiques à partir de scripts ou de programmes qui ont un CLI (Command-Line Interface) en ligne.

Déconnectez la mini chaîne de blocSSH.

call **OpenQbitSSHClient1** .DisconnectMiniBlockchainSSH

Paramètres d'entrée et de sortie : Sans objet (aucun)

Description : Bloc pour fermer la session SSH.

Extension OpenQbitStringHash.

Bloc pour générer une chaîne MD5 (**MD5ThisString**)

call **OpenQbitStringHash1** .MD5ThisString
string

" Mini BlocklyChain es modular "

Paramètres d'entrée : chaîne de caractères

Paramètres de sortie : Délivre le hachage MD5.

Description : Bloc pour créer le hachage MD5 de la chaîne de caractères donnée dans le paramètre d'entrée.

Bloc pour générer la chaîne de caractères SHA256 (**SHA256ThisString**)

call **OpenQbitStringHash1** .SHA256ThisString
string

" Mini BlocklyChain es modular "

Paramètres d'entrée : chaîne de caractères

Paramètres de sortie : Délivre le hachage SHA256.

Description : Bloc pour créer le hachage SHA256 de la chaîne de caractères donnée dans le paramètre d'entrée.

Bloc pour générer la chaîne SHA512 (**SHA512ThisString**)

call **OpenQbitStringHash1** .SHA512ThisString
string

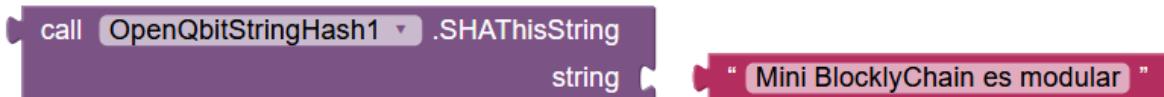
" Mini BlocklyChain es modular "

Paramètres d'entrée : chaîne de caractères

Paramètres de sortie : Délivre le hachage SHA512.

Description : Bloc pour créer le hachage SHA512 de la chaîne de caractères donnée dans le paramètre d'entrée.

Bloc pour générer une chaîne de caractères SHA1 (**SHAThisString**)



Paramètres d'entrée : chaîne de caractères

Paramètres de sortie : Délivre le hachage SHA1.

Description : Bloc permettant de créer le hachage SHA1 de la chaîne de caractères donnée dans le paramètre d'entrée.

22. Réglage des paramètres de sécurité dans Mini BlocklyChain.

Les paramètres de sécurité sont divisés en trois composantes de tout système qui est conçu et appliqué dans les éléments suivants :

- a. Base de données Redis (réseau de communication de secours)
- b. Système de synchronisation entre pairs
- c. Intégrer la sécurité pour protéger la base de données SQLite
- d. Environnement de développement pour l'intégration des modules

a. Base de données Redis (réseau de communication de secours)

Renommer les commandes dangereuses, la fonction supplémentaire de sécurité intégrée de Redis consiste à renommer ou à désactiver certaines commandes considérées comme dangereuses.

Lorsqu'elles sont exécutées par des utilisateurs non autorisés, ces commandes peuvent être utilisées pour reconfigurer, détruire ou supprimer leurs données. Comme le mot de passe d'authentification, le renommage ou la désactivation des commandes est configuré dans la même section **SECURITE** du fichier `/etc/redis/redis.conf`.

Certains des commandements considérés comme dangereux : **FLUSHDB**, **FLUSHALL**, **KEYS**, **PEXPIRE**, **DEL**, **CONFIG**, **SHUTDOWN**, **BGREWRITEAOF**, **BGSAVE**, **SAVE**, **SPOP**, **SREM**, **RENAME** et **DEBUG**. Cette liste n'est pas complète, mais renommer ou désactiver toutes les commandes de cette liste est un bon début pour améliorer la sécurité de votre serveur Redis.

En fonction de vos besoins spécifiques ou de ceux de votre site, vous devez renommer ou désactiver une commande. Si vous savez que vous n'utiliserez jamais une commande qui peut être manipulée, vous pouvez la désactiver. D'autre part, vous pourriez vouloir la renommer.

Pour activer ou désactiver les commandes Redis, rouvrez le fichier de configuration :

```
$ vi /etc/redis/redis.conf
```

Attention : les étapes suivantes pour désactiver et renommer les commandes sont des exemples. Vous devez uniquement choisir de désactiver ou de renommer les commandes qui vous concernent. Vous pouvez consulter la liste complète des commandements et déterminer comment ils peuvent être utilisés à mauvais escient dans redis.io/commands.

Pour désactiver une commande, il suffit de la renommer de manière à ce qu'elle devienne une chaîne vide (symbolisée par une paire de guillemets sans caractère entre eux), comme indiqué ci-dessous :

`/etc/redis/redis.conf`

```
.
.
.
# Il est également possible de tuer complètement une commande en la
renommant
# une chaîne vide :
#
renommer-commander FLUSHDB ""
renommer-commander FLUSHALL ""
renommer-commander DEBUG ""
.
.
```

Pour renommer une commande, donnez-lui un autre nom comme indiqué dans les exemples ci-dessous. Les commandes renommées devraient être difficiles à deviner pour les autres, mais faciles à retenir pour vous.

/etc/redis/redis.conf

```
...  
# renommer-commander CONFIG ""  
renommer-commander SHUTDOWN SHUTDOWN_MENOT  
renommer-commander CONFIG ASC12_CONFIG  
...
```

Enregistrez les modifications et fermez le fichier.

Après avoir renommé une commande, appliquez le changement en redémarrant Redis :

- sudo systemctl restart redis.service

Pour tester la nouvelle commande, entrez dans la ligne de commande Redis :

- redis-cli

Ensuite, effectuez l'authentification :

- auth your_redis_password

Sortie
OK

Comme dans l'exemple précédent, supposons que vous ayez renommé la commande CONFIG en ASC12_CONFIG. Essayez d'abord d'utiliser la commande CONFIG d'origine. Parce que vous l'avez rebaptisé, ça ne devrait pas marcher :

- config get requirepass

Sortie
(erreur) ERR commande inconnue 'config'.
Cependant, la commande renommée peut être appelée avec succès. Il n'est pas sensible à la casse :

- asc12_config get requirepass

Sortie
1) "requirepass
2) "votre_mot_de_passe_redis"

Enfin, vous pourrez clôturer le redécoupage :

- sortie

Notez que si vous utilisez déjà la ligne de commande Redis et que vous redémarrez Redis, vous devrez vous authentifier à nouveau. Sinon, si vous tapez une commande, cette erreur sera affichée :

```
Sortie  
NOAUTH Authentification requise.
```

b. Système de synchronisation entre pairs

Dans l'utilisation du système "Peer to Peer" entre les nœuds, le système comporte les trois éléments suivants appliqués dans le réseau de communication

-Privé : aucune information n'est stockée ailleurs que sur vos ordinateurs. Il n'existe pas de serveur central qui puisse être compromis (illégalement ou illégalement).

-Cryptée : toute communication est sécurisée par le protocole TLS (Transport Layer Security), un protocole cryptographique qui comprend une séquence parfaite pour empêcher toute personne non autorisée d'accéder à vos informations.

-Authentifié : chaque nœud est identifié par un certificat cryptographique fort. Seuls les nœuds que vous avez explicitement autorisés peuvent se connecter à vos informations.

<https://docs.syncthing.net/users/security.html>

En utilisant la commande en ligne SyncthingManager :

<https://github.com/classicsc/syncthingmanager>

c. Intégrer la sécurité pour protéger la base de données SQLite

Lorsque l'on utilise l'extension (**OpenQbitSQLite**) ou dans son cas l'extension (**ConnectorSSHClient**) pour utiliser le CLI SQLite, les deux extensions peuvent être combinées avec l'extension de sécurité AES (**OpenQbitEncDecData**).

Voir l'annexe "Exemple de création d'un système de Mini BlocklyChain".

d. Environnement de développement pour l'intégration des modules

La mise en œuvre de la sécurité dans l'environnement de développement peut se faire selon deux processus :

- Restreindre l'utilisation des bibliothèques OpenJDK.
- Utilisation limitée aux nœuds de système autorisés.

23. Annexe "Création des bases de données KeyStore & PublicKeys".

Un KeyStore est un dépôt de certificats de sécurité, soit des certificats d'autorisation, soit des certificats de clé publique, des mots de passe ou des clés de sécurité génériques telles que les clés privées correspondantes (adresses) d'un utilisateur, qui est utilisé pour créer ou traiter des transactions.

Il s'agit d'une base de données cryptée, de sorte que seul le propriétaire peut l'utiliser. Normalement c'est un type local, cependant, dans le système Mini BloclyChain c'est une base de données sécurisée mais elle est partagée et distribuée dans tous les nœuds, ceci est essentiellement dû à une simple raison, tous les nœuds doivent connaître les adresses de tous les utilisateurs (adresses publiques), les adresses privées sont toujours locales et sont d'usage unique et exclusif de chaque utilisateur, cependant quand on fait une transaction d'entrée (dépôt) ou de sortie (dépense) chaque transaction a toujours au moins trois composantes lors de sa création : adresse d'origine, adresse de destination et bien ou valeur qui est envoyé.

Pour créer notre KeyStore, nous devrons suivre les étapes et les exigences suivantes.

Une première exigence est d'avoir un type de stockage où ils seront stockés, dans notre cas nous utiliserons la base de données SQLite et nous créerons deux KeyStore : un avec les clés privées de l'utilisateur qui sera local et il sera sécurisé "crypté" les informations et une autre base de données qui stockera les clés publiques ; cette base sera partagée dans tous les nœuds du réseau, la base qui sera partagée dans le réseau sera également cryptée, cependant celle-ci aura un mot de passe que seuls les membres (nœuds) du réseau pourront partager.

La deuxième exigence fondamentale est le processus de cryptage des informations, ce qui sera fait avec l'extension spécialisée pour l'utilisation dans une base de données générique appelée (**OpenQbitEncDecData**) qui utilise un algorithme AES.

L'extension (**OpenQbitEncDecData**) est fonctionnelle pour la vérification des éléments unitaires de la chaîne de blocs, cependant, dans le cas où il faut vérifier l'intégrité totale de la chaîne de blocs, elle ne sera pas efficace, donc nous effectuerons également un cryptage d'une copie, mais dans ce cas, ce sera par le biais des extensions : (**OpenQbitAESEncryption**) et (**OpenQbitAESDecryption**) qui fonctionnent sur un fichier, et non sur des données référencées.

NOTE : Le serveur SSH doit être exécuté sur le terminal Termux pour que la base de données **KeyStore** fonctionne correctement. Exécutez la commande dans le terminal :

```
$ sshd
```

Les extensions basées sur l'algorithme AES peuvent être utilisées pour tout type de données ou de fichiers et cet algorithme a été choisi car c'est le seul qui a été prouvé comme étant à l'épreuve des attaques basées sur l'informatique quantique.

Criptosistema	Categoría	Tamaño de clave	Parámetro de seguridad	Algoritmo cuántico estimado que rompa el criptosistema	Nº de qubits lógicos necesarios	Nº de qubits físicos necesarios	Tiempo necesario para romper el sistema	Estrategias de reemplazo cuántico-resilientes
AES-GCM	Cifrado simétrico	128	128	Algoritmo de Grover	2.953	$4,61 \times 10^6$	$2,61 \times 10^{12}$ años	
		192	192		4.449	$1,68 \times 10^7$	$1,97 \times 10^{22}$ años	
		256	256		6.581	$3,36 \times 10^7$	$2,29 \times 10^{32}$ años	
RSA	Cifrado asimétrico	1024	80	Algoritmo de Shor	2.290	$2,56 \times 10^6$	3,58 horas	Migrar a un algoritmo PQC seleccionado por el NIST
		2048	112		4.338	$6,2 \times 10^6$	28,63 horas	
		4096	128		8.434	$1,47 \times 10^7$	229 horas	
ECC Problema del logaritmo discreto	Cifrado asimétrico	256	128	Algoritmo de Shor	2.330	$3,21 \times 10^6$	10,5 horas	Migrar a un algoritmo PQC seleccionado por el NIST
		386	192		3.484	$5,01 \times 10^6$	37,67 horas	
		512	256		4.719	$7,81 \times 10^6$	95 horas	
SHA256	Minado de Bitcoin	N/A	72	Algoritmo de Grover	2.403	$2,23 \times 10^6$	$1,8 \times 10^6$ años	
PBKDF2 con 10.000 iteraciones	Hashing de contraseñas	N/A	66	Algoritmo de Grover	2.403	$2,23 \times 10^6$	$2,3 \times 10^7$ años	Abandonar la autenticación basada en contraseñas

Le tableau ci-dessus fait référence aux Académies nationales des sciences, de l'ingénierie et de la médecine.

<https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects>

Description

La mécanique quantique, le sous-domaine de la physique qui décrit le comportement des très petites particules (les quanta), fournit la base d'un nouveau paradigme informatique. Proposé pour la première fois dans les années 1980 comme moyen d'améliorer la modélisation informatique des systèmes quantiques, le domaine de l'informatique quantique a récemment attiré une grande attention en raison des progrès réalisés dans la construction de dispositifs à petite échelle. Toutefois, des progrès techniques importants seront nécessaires avant qu'un ordinateur quantique pratique puisse être réalisé à grande échelle.

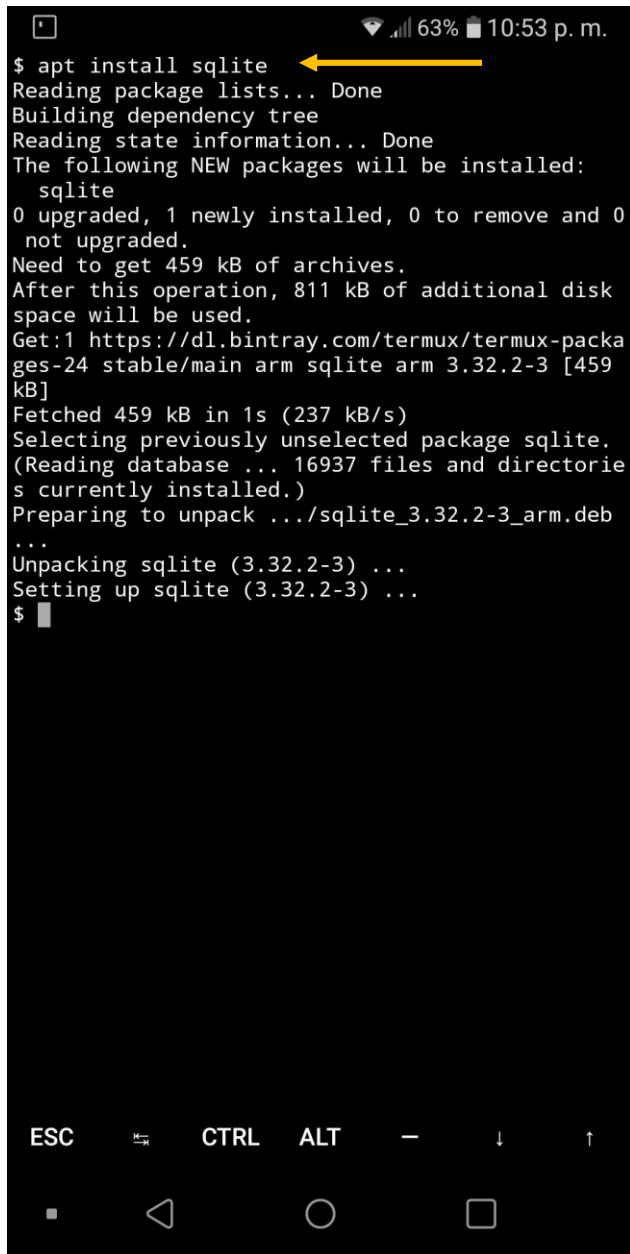
L'informatique quantique : progrès et perspectives présente une introduction à ce domaine, notamment les caractéristiques et les limites uniques de cette technologie, et évalue la faisabilité et les implications de la création d'un ordinateur quantique fonctionnel capable de répondre aux problèmes du monde réel. Ce rapport examine les exigences en matière de

matériel et de logiciel, les algorithmes quantiques, les moteurs des progrès de l'informatique quantique et des dispositifs quantiques, les points de référence associés aux cas d'utilisation pertinents, le temps et les ressources nécessaires et la manière d'évaluer la probabilité de réussite. Installation du gestionnaire de base de données SQLite dans le terminal TERMUX et test du CLI (Command-Line) de la commande **sqlite3** en créant une base de données appelée **keystore.db**

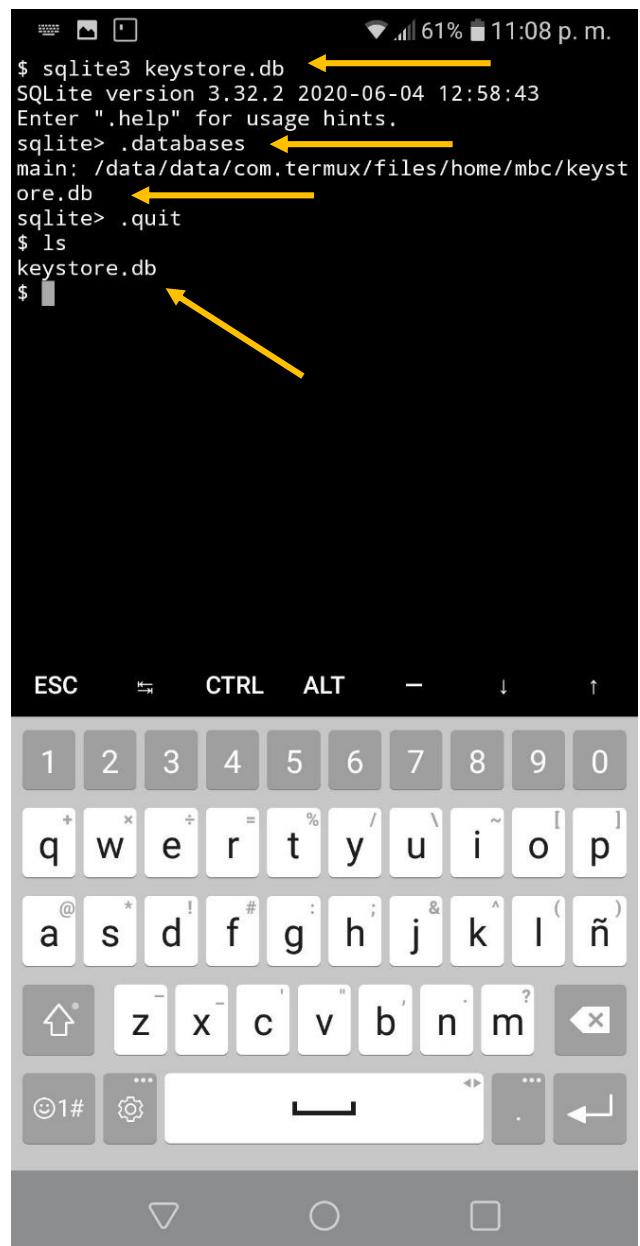
Nous utilisons les commandes :

\$ apt install sqlite

\$ sqlite3 keystore.db



```
$ apt install sqlite
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sqlite
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 459 kB of archives.
After this operation, 811 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm sqlite arm 3.32.2-3 [459
kB]
Fetched 459 kB in 1s (237 kB/s)
Selecting previously unselected package sqlite.
(Reading database ... 16937 files and directorie
s currently installed.)
Preparing to unpack .../sqlite_3.32.2-3_arm.deb
...
Unpacking sqlite (3.32.2-3) ...
Setting up sqlite (3.32.2-3) ...
$
```



```
$ sqlite3 keystore.db
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
sqlite> .databases
main: /data/data/com.termux/files/home/mbc/keyst
ore.db
sqlite> .quit
$ ls
keystore.db
$
```

Ensuite, dans le **gestionnaire sqlite>**, nous exécutons la phrase .databases pour vérifier dans quel chemin se trouve la base de données que nous sommes en train de créer, puis pour quitter et enregistrer la base de données, nous donnons la phrase de . quit.

NOTE : Dans les deux instructions ou commandes du **gestionnaire sqlite>**, vous devez d'abord mettre un point "." **dans** la syntaxe.

Enfin, nous vérifions que la base de données (vide) a déjà été créée en donnant la commande :

ls

Nous procédons à la création d'un tableau dans lequel les clés primaires seront stockées sous trois formats différents : hexadécimal, binaire et l'adresse de référence de l'utilisateur de la Mini BlocklyChain qui est la clé publique de sa clé primaire respective.

Le cryptage des données AES ne sera appliqué qu'aux formats hexadécimal et binaire. Dans le cas de l'adresse de l'utilisateur addrMBC et de l'alias non parce qu'il s'agit de la clé publique qui peut et doit être partagée sur le réseau afin de recevoir des transactions, ainsi que l'alias pour effectuer la recherche par ce champ.

Création d'une table appelée "privatekey" dans la base de données en SQLite (keystore.db).

```
CREATE TABLE privatekey (
    id clé primaire entière
    alias      VARCHAR(50) NON NUL
    addrHexVARCHAR  (65) NON NUL
    addrMBCVARCHAR (65) NON NUL
    addrBinBLOB   PAS NUL
);
```

Exécutons les phrases dans le CLI sqlite pour créer la base de données keystore.db.

Utilisons à nouveau la ligne de commande sqlite3 avec la commande suivante :

\$ sqlite3

Dans ce premier cas, nous ouvrirons la base de **données** déjà créée pour pouvoir y travailler avec la phrase suivante dans le gestionnaire :

```
Sqlite> .open keystore.db
```

Ensuite, nous allons entrer l'instruction SQL "**CREATE TABLE**" pour créer la table de **clés privées**.

Ensute, deux options sont présentées pour créer le même tableau, l'une passe par une seule ligne où sont incluses toutes les instructions SQL des éléments qui l'intègrent. Le deuxième exemple est l'introduction de chaque élément qui intègre la structure de manière segmentée.

```
$ sqlite3
```

```
SQLite version 3.32.2 2020-06-20 15:25:24
```

```
Entrez ".help" pour les conseils d'utilisation.
```

```
Connecté à une base de données en mémoire transitoire.
```

```
Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.
```

```
sqlite> .open keystore.db
```

```
sqlite> CREATE TABLE privatekey (id clé primaire entière AUTOINCREMENT NOT NULL, aka  
VARCHAR(50) NOT NULL, addrHex VARCHAR(65) NOT NULL, addrMBC VARCHAR(65) NOT  
NULL, addrBin BLOB NOT NULL) ;
```

```
sqlite> .quit
```

La création de la même table de **clés privées** est présentée ci-après, mais c'est en introduisant l'instruction SQL de manière segmentée :

```
sqlite> CREATE TABLE privatekey (  
...> id  clé primaire entière AUTOINCREMENT  
...> alias  VARCHAR(50) NON NUL,  
...> addrHex VARCHAR (65) NON NUL,  
...> addrMBC VARCHAR (65) NON NUL,  
...> addrBin BLOB  NOT NULL  
...> ) ;  
sqlite> .tables  
privatekey  
sqlite> .quit
```

Les deux exemples ci-dessus donnent le même résultat. Plus tard, nous exécutons la phrase **.tables** pour vérifier que la table **privatekey** a été créée, à la fin nous donnerons la phrase **.quit** avec ce processus nous avons déjà créé la table **privatekey** dans la base de données SQLite **keystore.db**.

Jusqu'à présent, nous disposons déjà de la structure de la base de données keystore.db et de sa table de clés privées où les clés privées seront stockées de manière cryptée.

Cela devrait montrer quelque chose de similaire dans le nœud (téléphone mobile) avec le terminal TERMUX.



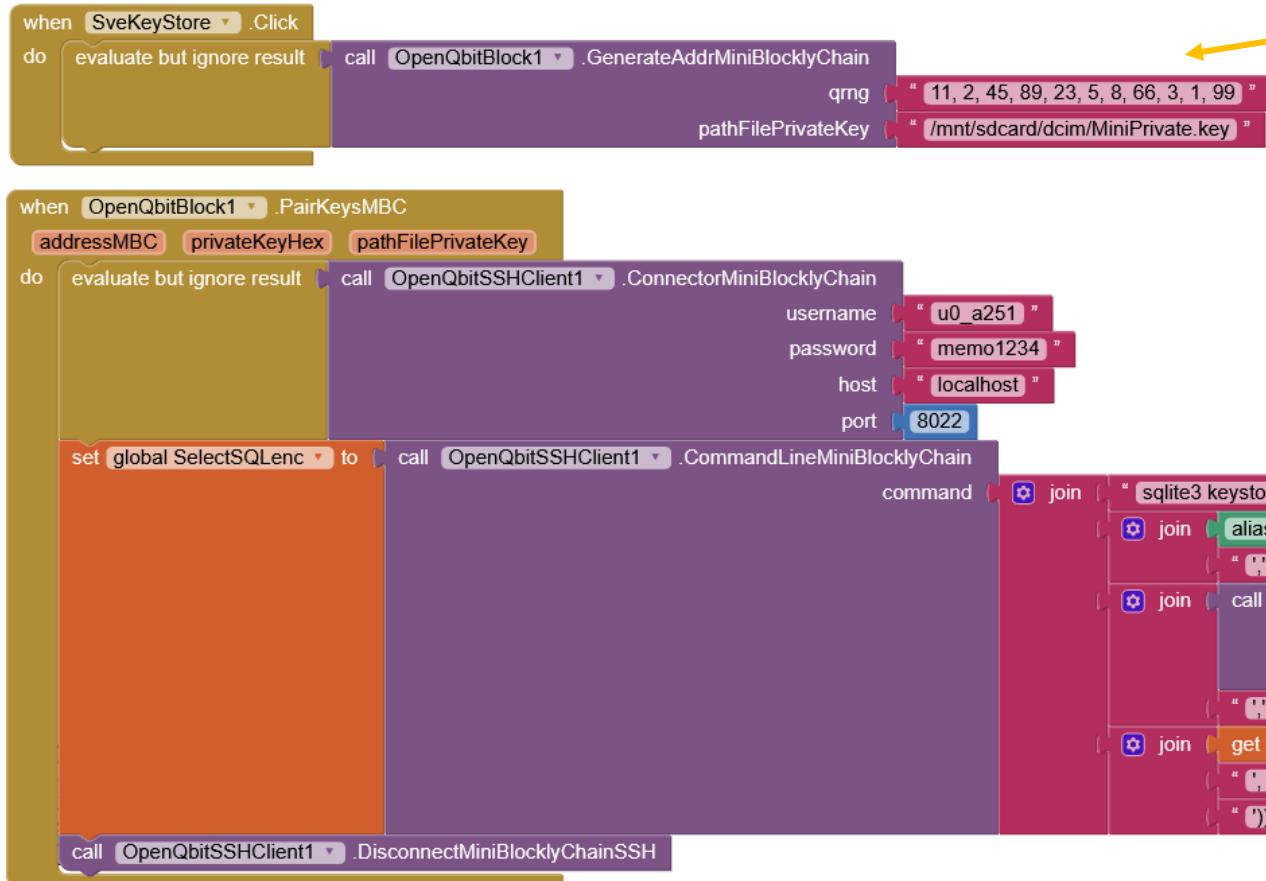
```
$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open keystore.db
sqlite> CREATE TABLE privatekey (
...>     id integer primary key AUTOINCREMENT,
...>     alias VARCHAR(50) NOT NULL,
...>     addrHex VARCHAR(65) NOT NULL,
...>     addrMBC VARCHAR(65) NOT NULL,
...>     addrBin BLOB NOT NULL
...> );
sqlite> .tables
privatekey
sqlite> .quit
$
```

Instructions SQL pour créer une nouvelle table

Nous allons maintenant utiliser certaines extensions de sécurité pour insérer des données de "clé privée" et ensuite consulter ces données.

Nous utiliserons le bloc pour générer une nouvelle adresse utilisateur (**GenerateAddrMiniBlocklyChain**), ce bloc nous donnera l'adresse privée en deux formats (hexadécimal et binaire), ainsi que l'adresse publique en format d'adresse générique MBC. Nous utiliserons également l'extension (**OpenQbitSSHClient**) et l'extension (**OpenQbitEncDecData**) pour voir plus de détails sur ces derniers, consultez la section "Définition et utilisation des blocs de sécurité". Dans le terminal Termux, vous devez faire fonctionner le service SSH.

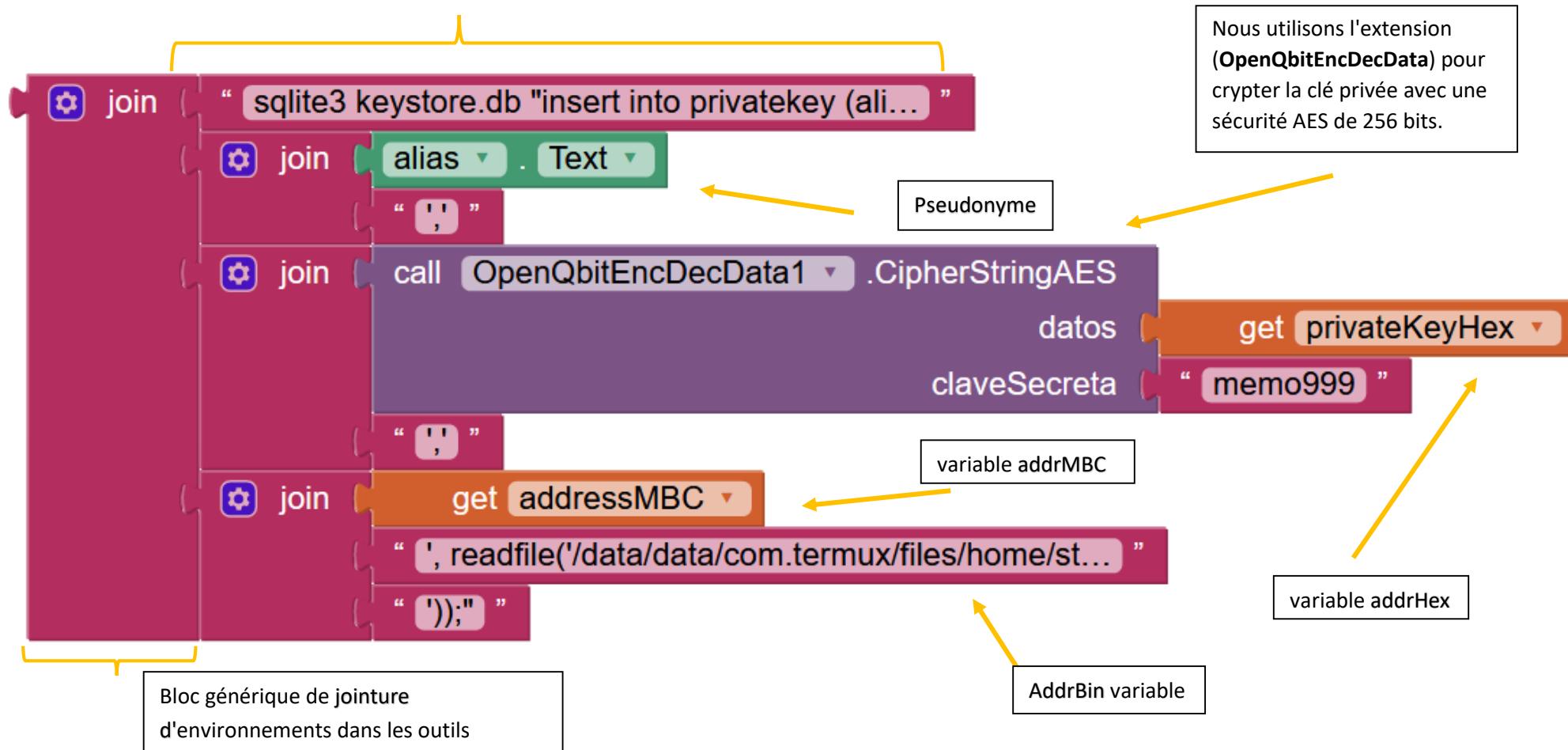
Insérer les données cryptées dans la base de données **keystore.db**



Les séries de nombres aléatoires quantiques générées par le bloc (ApiGetQRNGInteger) vérifient comment formater le résultat JSON, puisque l'entrée du bloc (GenerateAddrMiniBlocklyChain) nécessite une série de nombres seulement séparés par des

La syntaxe de la commande est très importante, nous devons introduire la commande suivante au bon format dans l'environnement Blockly. Les valeurs des valeurs seront toujours les variables, par exemple :

```
sqlite3 keystore.db "insérer dans privateKey (alias, addrHex, addrMBC, addrBin) les valeurs ('memo', 'QWERTY', 'MBC12345',  
readfile('/data/data/com.termux/files/home/storage/dcim/MiniPrivate.key')) ;;".
```



Après l'exécution des blocs, nous aurons un résultat dans la table des clés privées de la base de données keystore.db similaire à ce qui suit :

Nous entrons le sqlite handler dans le terminal Termux, nous ouvrons la base keystore.db et nous exécutons la phrase :

\$ sqlite3

SQLite version 3.32.2 2020-06-20 15:25:24

Entrez ".help" pour les conseils d'utilisation.

Connecté à une base de données en mémoire transitoire.

Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.

sqlite> .open keystore.db

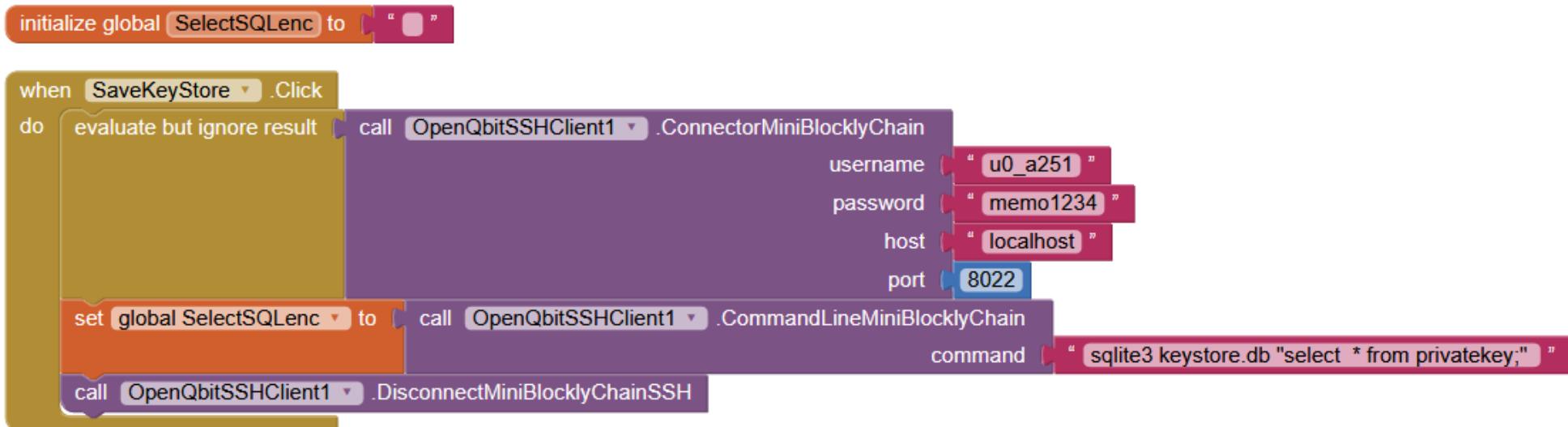
sqlite> sélectionnez * de privatekey ;

```

$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open keystore.db
sqlite> select * from privatekey;
1|mexico|07JBBizTcC0Ce6a8Pwe5aTV41Ql1DKiUQZyPSfJuRbVUZnvIwQJcry4LilBMs5jHavQeMo1iN8rCO87V5Xka2YaKR4fswopeTQmI/Q+ipzI=|2eENpFt2HSuGtXNxoeiwfrp5d6e87Z7y5|0♦♦
sqlite>

Alias : mexique
addrBin (clé privée en binaire)
cryptage
Adresse addrMBC
    
```

CONSULTER toutes les données dans la table des **clés privées de la** base de données SQLite **keystore.db**



CONSULTER les alias des données dans la table des **clés privées de la** base de données SQLite **keystore.db**

sqlite3 keystore.db "sélectionnez un alias de privatekey ;"

Interroger toutes les données de la colonne addrHex cryptées dans la table des **clés privées de la** base de données SQLite **keystore.db**

sqlite3 keystore.db "select addrHex from privatekey ;"

CONSULTER pour récupérer la clé privée addrBin dans la table **privatekey de la** base de données SQLite **keystore.db**

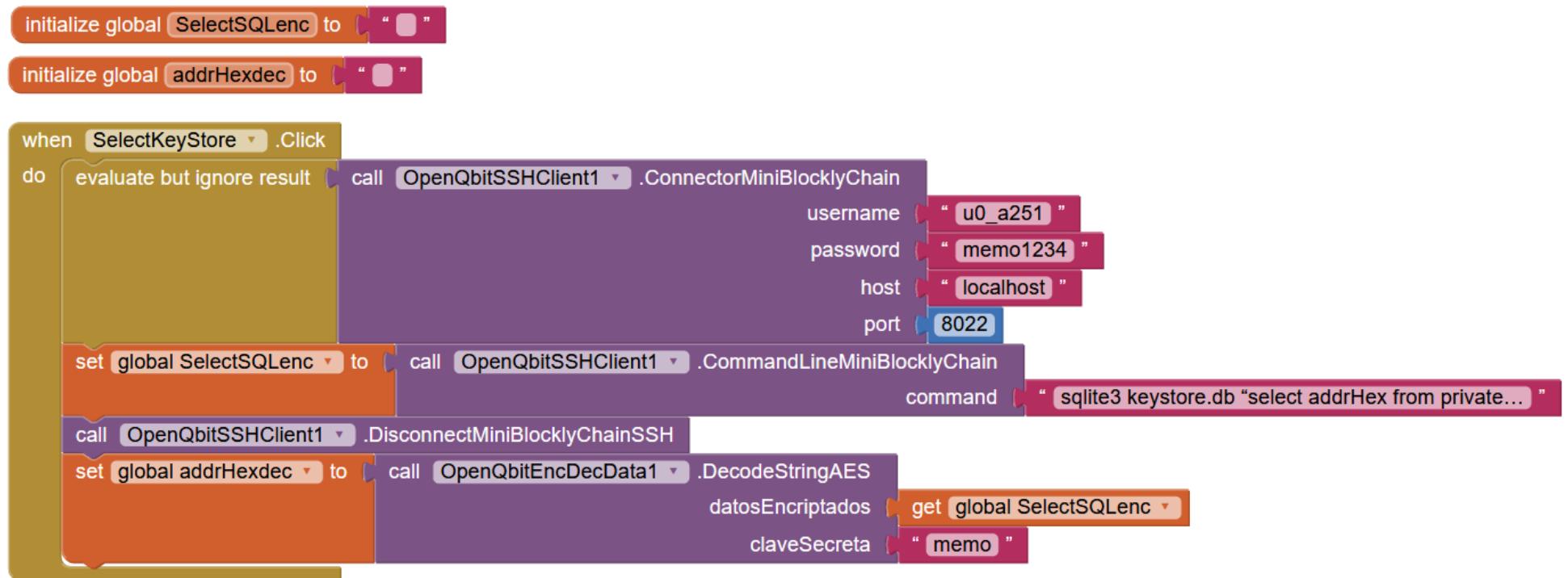
sqlite3 **writefile('PrivateKey.key', addrBin)** FROM privatekey WHERE alias='mexico'; (2)

(1) Ce type de requête sera la principale pour consulter la clé privée pour effectuer le processus de signature des transactions.

Requête pour le décryptage de données par alias dans la colonne addrHex de la table de clés privées de la base de données SQLite keystore.db

Dans ce cas, nous utiliserons le bloc (**DecodeStringAES**) pour décoder les données stockées dans la colonne "addrHex".

sqlite3 keystore.db "select addrHex from privatekey where='mexico';"



Cette consultation nous donne la clé privée en format hexadécimal, c'est la partie fondamentale de toute réception ou envoi de transactions. Il est recommandé à plusieurs reprises de conserver une sauvegarde de cette base de données keystore.db.

Conception de la base de données **publickeys.db** avec table publicaddr :

```
$ sqlite3
```

SQLite version 3.32.2 2020-06-20 15:25:24

Entrez ".help" pour les conseils d'utilisation.

Connecté à une base de données en mémoire transitoire.

Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.

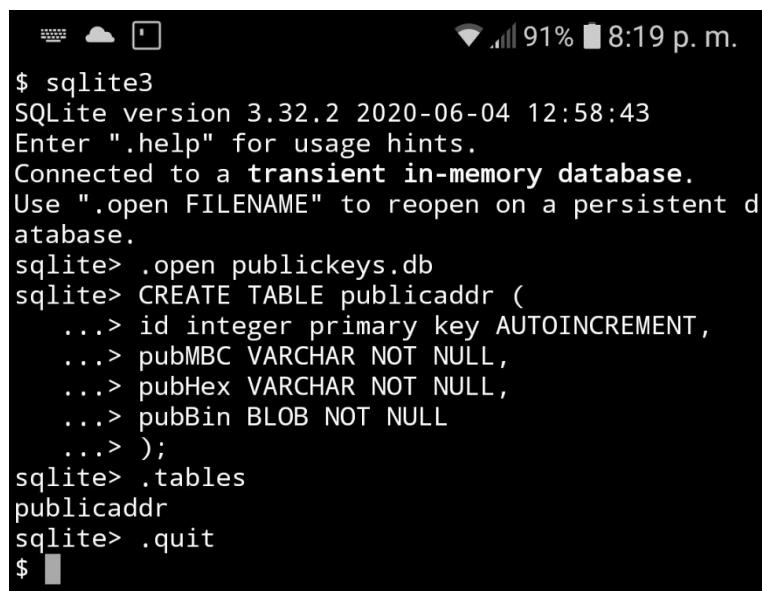
```
sqlite> .open publickeys.db
```

```
sqlite> CREATE TABLE publicaddr (id clé primaire entière AUTOINCREMENT NON NUL,  
pubMBC VARCHAR NON NUL, pubHex VARCHAR NON NUL, pubBin BLOB NON NUL);
```

```
sqlite> .quit
```

Ensuite, la création de la table **publiée** est indiquée par la phrase SQL suivante sous forme segmentée :

```
sqlite> CREER UN TABLEAU publiedaddr (  
...> id clé primaire entière AUTOINCREMENT  
...> pubMBC VARCHAR PAS NUL,  
...> pubHex VARCHAR PAS NUL,  
...> pubBin BLOBE PAS NUL  
...> );  
sqlite> .tables  
publishedaddr  
sqlite> .quit
```



The screenshot shows a terminal window on a mobile device. The status bar at the top indicates signal strength, battery level (91%), and the time (8:19 p.m.). The terminal prompt is '\$'. The user runs the command '\$ sqlite3' which outputs the SQLite version (3.32.2) and connection details. Then, the user creates a new table 'publicaddr' with four columns: 'id' (primary key, auto-increment, integer), 'pubMBC' (VARCHAR, NOT NULL), 'pubHex' (VARCHAR, NOT NULL), and 'pubBin' (BLOB, NOT NULL). Finally, the user lists the tables in the database and exits the SQLite shell with '.quit'.

```
$ sqlite3  
SQLite version 3.32.2 2020-06-04 12:58:43  
Enter ".help" for usage hints.  
Connected to a transient in-memory database.  
Use ".open FILENAME" to reopen on a persistent database.  
sqlite> .open publickeys.db  
sqlite> CREATE TABLE publicaddr (  
...> id integer primary key AUTOINCREMENT,  
...> pubMBC VARCHAR NOT NULL,  
...> pubHex VARCHAR NOT NULL,  
...> pubBin BLOB NOT NULL  
...> );  
sqlite> .tables  
publicaddr  
sqlite> .quit
```

24. Annexe "Commandes RESTful SQLite GET/POST".

Ensuite, les différentes commandes que nous pouvons utiliser dans le SQLite de repos du réseau de sauvegarde sont indiquées. Ces derniers ont été consultés depuis le développement initial de GITHUB (<https://github.com/olsonpm/sqlite-to-rest>)

Opérations CRUD (Create, Read, Update and Delete) RESTful.

Voici une liste des opérations disponibles mises à disposition par l'API RESTful sous forme de pseudo exemples. Nous supposerons une base "op.sqlite" et deux tables "trans" associées pour les transactions et un autre "signe" pour l'adresse de la source, l'adresse de la destination et la valeur de l'actif avec deux colonnes id INTEGER PRIMARY KEY.

Comme indiqué dans les limitations, veuillez noter que les méthodes (SUPPRIMER et POSTER) ne peuvent affecter qu'une ligne à la fois.

OBTENIR Ceci permet la plus grande variation. Nous verrons plus tard tous les opérateurs de recherche disponibles.

Les rubriques peuvent être spécifiées juste en dessous des URL.

/transDemandes

pour toutes les lignes

/trans?id=1Où

id = 1

/trans

gamme : rangées=0-2Les
trois premières
rangées

/trans

range : rows=-5Les
cinq dernières lignes

/trans

gamme : rows=0-

Autant de lignes que le serveur peut fournir, ce qui, en pratique, correspond au nombre de lignes le moins élevé entre maxRange et le nombre total de lignes.

/trans

gamme : rows=1-

Autant de lignes que le serveur peut fournir, à partir de la ligne 1.

/trans
ordre : nomOrdre
par nom croissant

/trans
ordre : nom descOrdre
par nom descOrdre par nom

/trans
ordre : nom desc, id

Contribués, mais d'abord triés par nom descendant, et dans le cas d'une égalité par identifiant ascendant.

/trans?id>1
Où vous allez > 1

/trans?id>=2&id<5
Où id >= 2 et id < 5

/trans?name_NOTNULL
Lorsque le nom n'est pas nul

/trans?name_ISNULLOù le
nom est nul

/trans?id!=5&name_LIKE 'Spotted%'.
Où id != 5 et le nom est comme "Spotted%" (ignorez les guillemets)

/trans?id>=1&id<10&name_LIKE 'Avery%'.
ordre : nom desc, id range : rows=2-4

Contribution à titre d'exemple.

Obtenir une transaction avec des identifiants entre 1 et 9 inclus, avec un nom comme "Avery%", ordonné d'abord par nom descendant puis par identifiant ascendant, en obtenant la troisième à la cinquième ligne du résultat. Ou en SQL :

DELETE Requiert une chaîne d'interrogation dont toutes les clés primaires sont définies comme étant égales à une valeur. Cela nécessite une suppression maximale d'une seule rangée.

/trans?id=1Supprime le
trans avec id=1

Si, au contraire, la transaction comportait une clé principale composée d'un identifiant et d'un nom.

/trans?id=1&name= 'Avery IPA'.

POST créer

Vous ne devez pas passer une chaîne d'interrogation. Si une chaîne de requête est passée, une mise à jour POST est supposée. Toutes les demandes POST doivent passer le type de contenu de l'en-tête : application / json.

Veuillez noter que le corps doit contenir toutes les colonnes de la CLE PRIMAIRE non annulable et non INTÉGRER. Dans le cas contraire, une réponse de 400 sera envoyée indiquant les champs qui ont été perdus. Les colonnes annulables seront nulles et les colonnes de la CLÉ PRIMAIRE INTÉGRÉE seront automatiquement augmentées selon les spécifications de sqlite3.

Les données JSON seront spécifiées juste en dessous des URL.

/trans

{"id":1, "name" : "Serendipity"}Crée
un trans avec id = 1 et name = 'Serendipity'

/trans

{"id":1}Crée
un trans avec id = 1 et nom = NULL

/trans

{"nom" : "Serendipity"}

Crée une transaction dont l'identifiant est fixé à la valeur suivante augmentée des spécifications sqlite3 TOUCHE PRIMAIRE INTEGRE.

/trans

{ }

Crée une transaction avec un identifiant augmenté et le nom mis à NULL.

Mise à jour POST

Il doit contenir une chaîne de requête. Sans chaîne d'interrogation, la création de POST est supposée. Comme pour la création de POST, le type de contenu de l'en-tête : application / json est requis.

La chaîne d'interrogation doit contenir toutes les clés primaires pour garantir qu'une seule ligne est mise à jour. Si des valeurs incorrectes sont transmises, un 400 sera renvoyé avec les clés incriminées.

Le corps de la demande doit contenir un objet non vide et doit contenir des clés valides correspondant aux noms des colonnes.

Les données JSON seront spécifiées juste en dessous des URL.

/trans?id=1
{"id":2}

Mettez à jour la transaction avec un ID de 1 en la mettant à 2.

/trans?id=1
{"nom" : "MCBza45Rt56cvbgfdR2Swd788kj"}

Mettez à jour la transaction avec l'ID de 1 en fixant son nom ou sa valeur à "MCBza45Rt56cvbgfdR2Swd788kj".

Si la salle des marchés avait plutôt une clé principale composée de l'identifiant et du nom.

/trans?id=1&name=MCBza45Rt56cvbgfdR2Swd788kj
{"nom" :"MCB3ofFG5Hj678MNb09vLdfaasx"}

Mettez à jour la transaction dont l'identifiant est un et l'adresse est MCBza45Rt56cvbgfdR2Swd788kj, en définissant le MCB3ofFG5Hj678MNb09vLdfaasx.

Référence

isSqliteFile

Il suffit de vérifier les 16 premiers octets du fichier pour voir s'il est au format "sqlite 3" suivi d'un octet nul.

isDirectory

Renvoie le résultat de fs.statsSync suivi de .isDirectory

isFile

Renvoie le résultat de fs.statsSync suivi de .isFile

GET consulting operators

Les conditions d'enquête doivent être marquées par des symboles de connexion, par exemple id> 5 & name = MCBza45Rt56cvbgfdR2Swd788kj

Opérateurs binaires (nécessite une valeur après) Man.

=
!=
>=
<=
>
<
COMME

LIKE est spécial parce qu'il doit avoir des citations d'ouverture et de clôture simples. Dans le cas contraire, une erreur de 400 sera générée, indiquant les cas où l'analyse n'a pas pu être achevée et ce qui était attendu. Voir les exemples d'opérations CRUD RESTful.

Opérateurs uniques (doivent suivre le nom d'une colonne)

EST NUL

PAS NUL

Objet de configuration du routeur

isLadenPlainObject

Le but de cet objet est de fournir une configuration générique pour le routeur sqlite. Les propriétés suivantes sont autorisées :

prefix : isLadenString La chaîne de caractères passée à l'option de construction du préfixe du routeur koa. Par exemple, le serveur squelette ne spécifie pas de préfixe, ce qui permet à l'API de la bière d'être atteinte directement depuis la racine du domaine http://localhost:8085/trans. Si le préfixe est "/api", vous devez envoyer les demandes à http://localhost:8085/api/trans.

allTablesAndViews : un objet de configuration tabulaire

Les paramètres spécifiés dans cet objet seront appliqués à toutes les tables et vues, éventuellement remplacés par la propriété tablesAndViews.

tablesAndViews : isLadenPlainObject L'objet passé doit avoir des clés qui correspondent à la colonne de la base de données ou afficher les noms. Dans le cas contraire, un message d'erreur amical sera émis. Les valeurs de chaque tableau et vue doivent être un objet de configuration tabulaire.

Objet de configuration tabulaire

isLadenPlainObject Cet objet représente les paramètres qui peuvent être définis pour les vues ou les tableaux. Il permet les propriétés suivantes :

maxRange : isPositiveNumber

Demande par défaut : 1000

Il s'agit de la portée maximale que votre serveur autorisera les requêtes. Si une demande GET arrive sans en-tête de plage, la spécification suppose que vous voulez la ressource entière. Si le nombre de lignes résultant de l'EEG est supérieur à maxRange, un statut 416 est renvoyé avec l'en-tête personnalisé max-range. La valeur par défaut de l'application est

délibérément conservatrice dans l'espoir que les auteurs fixeront maxRange en fonction de leurs besoins.

Veuillez noter que "Infinity" est un nombre positif valide.

drapeaux : isLadenArray

Actuellement, le seul indicateur accepté est la chaîne "sendContentRangeInHEAD". Lorsqu'elles sont définies, les demandes HEAD renvoient la gamme de contenu disponible sous la forme content-range : * / <max-range>. La raison pour laquelle il est configurable est que le calcul de la plage maximale peut représenter plus de travail que ce qu'il vaut, en fonction de la charge du serveur et de la taille de vos tables.

En-têtes personnalisés

Candidature

order : cet en-tête est uniquement défini pour GET, et peut être considéré comme l'équivalent de sql ORDER BY. Il doit contenir un nom de colonne délimité par des virgules, chacune étant éventuellement suivie d'un espace et des chaînes "asc" ou "desc". Si des valeurs de commande incorrectes sont envoyées, une réponse 400 indiquera lesquelles.

Réponse

Tous ne sont pas nécessairement personnalisés, mais toute utilisation est en dehors des spécifications et nécessite donc une clarification.

GET

max-range : cet en-tête est renvoyé lorsque le nombre de lignes demandées dépasse la maxRange configurée. Notez que la demande peut ne pas spécifier l'en-tête de la plage, mais le nombre de lignes résultant de cette ressource sera quand même vérifié.

contenu : rfc7233 états

Seuls les codes de statut 206 (Contenu partiel) et 416 (Fourchette insatisfaisante) décrivent une signification pour la fourchette de contenu.

Lorsque sqlite-to-rest répond avec un code de statut 200, l'en-tête de la plage de contenu est envoyé avec le format 206 de <début de rang> - <fin de rang> / <nombre de rangs>.

Lorsqu'une demande est envoyée sans en-tête de plage et que le nombre de lignes qui en résulte dépasse maxRange, un 400 est renvoyé avec la plage de contenu définie au format 416 de * / <nombre de lignes>

Notez que cet en-tête peut être renvoyé dans une réponse HEAD.

accept-order : sera renvoyé si l'ordre de l'en-tête de la demande a une syntaxe incorrecte ou des noms de colonne incorrects. Pour plus de détails, voir HEAD -> accept-order ci-dessous.

25. Annexe "Code Java SQLite-Redis Connector".

Un code du lien entre la communication des deux bases de données SQLite-Redis est présenté ci-dessous. Fondamentalement, comme on peut le voir, le connecteur modifie le format SQLite en une chaîne générique de données (transactions) que Redis doit recevoir.

Le processus ci-dessus agit comme un déclencheur de file d'attente de transactions vers tous les nœuds qui sont connectés et qui sont des candidats possibles pour le traitement de la file d'attente de transactions actuelle.

Lorsque la base de données Redis reçoit la file d'attente des transactions par la configuration Maître-Esclave dont elle dispose, elle distribue les informations en temps réel et de manière égale à tous les nœuds.

Un autre point fondamental est que tous les nœuds doivent être synchronisés dans leur horloge, ce qui sera fait comme nous l'avons vu précédemment avec la fonctionnalité de la requête à un pool de serveurs NTP (Network Time Protocol).

Ce connecteur effectue également le premier niveau d'examen de la sécurité des données en calculant l'arbre racine Merkle de toutes les transactions.

Code de base du connecteur SQLite-Redis.

```
import java.sql.* ;
import java.util.* ;
importer des tableaux java.util.Arrays ;
importer java.util.list ;
importer java.util.array ;
import java.security.* ;
importer java.security.MessageDigest ;
import redis.clients.jedis.Jedis ;
classe RediSqlite{
    public String previousHash ;
    public String merkleRoot ;
    public static ArrayList<String> transactions = new ArrayList<String>() ;
    public static ArrayList<String> treeLayer = nouveau ArrayList<String>() ;
    public static String getMerkleRoot() {
        int count = transactions.size() ;
        int item = 1 ;
```

```

Int. impaire = 0 ;
ArrayList<String> previousTreeLayer = transactions ;
while(count > 1) {
    treeLayer = nouveau ArrayList<String>() ;
    for(int i=1 ; i <= count ; i=i+2) {

        si (!lesPar(count) && i == count) impair = 1 ;
        treeLayer.add(applySha256(previousTreeLayer.get(i-item)
previousTreeLayer.get(i-impar)) +)

    }
    point = 1 ;
    Impaire = 0 ;
    count = treeLayer.size() ;
    previousTreeLayer = couche d'arbres ;
}
Chaîne merkleRoot = (treeLayer.size() == 1) ? treeLayer.get(0) : "" ;
retourner merkleRoot ;
}
//Définir si l'arbre Merkle est pair ou impair
booléen statique enPar(int number){
    si (numero%2==0) renvoie vrai ; sinon renvoie faux ;
}

public static String applySha256(String input){
try {
    MessageDigest digest = MessageDigest.getInstance("SHA-256") ;
    //Applique le sha256 à notre contribution,
    byte[] hash = digest.digest(input.getBytes("UTF-8")) ;
    StringBuffer hexString = nouveau StringBuffer() ; // Ceci contiendra le hachage en
hexadécimal
    for (int i = 0 ; i < hash.length ; i++) {
        String hex = Integer.toHexString(0xff & hash[i]) ;
        if(hex.length() == 1) hexString.append('0') ;
        hexString.append(hex) ;
    }
    retourne hexString.toString() ;
}
catch(Exception e) {
    lancer une nouvelle RuntimeException(e) ;
}

```

```
}

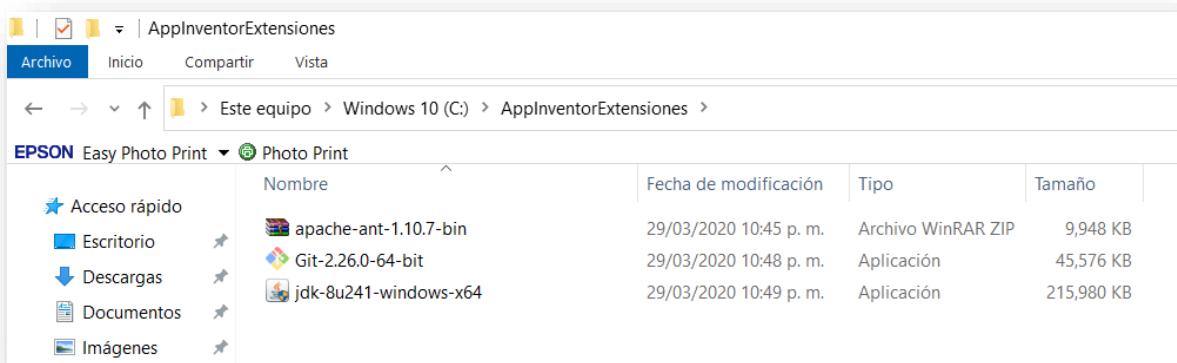
public static void main(String args[]){
    try{
        Connection      con=DriverManager.getConnection("jdbc:sqlite:C://memo/sqlite-tools-
win32-x86-3310100/trans.sqlite3");
        //Connexion au serveur Redis sur localhost
        jedis = nouveaux Jedis ("localhost");
        jedis.auth ("memo1234");
        Statement stmt=con.createStatement();
        String sql = "SELECT * FROM brewery";
        ResultSet rs=stmt.executeQuery(sql);
        while(rs.next()) {
            transactions.add(rs.getString(2));
            jedis.set("LATAM"
                    :"+String.valueOf(rs.getInt(1)), "["+"\"" +rs.getString(2)+"\\""+","+"\"" +rs.getString(3)+"+", "+"\
                    +"\""+rs.getString(4)+"\"");
        }
        jedis.set("LATAM:merkleroot", getMerkleRoot());
        System.out.println("Nombre d'éléments ArrayList: : "+treeLayer.size());
        avec .close();
        catch(Exception e){ System.out.println(e);}

    }
}
```

26. Annexe "Mini BlocklyChain pour les développeurs".

Dans cette annexe, nous examinerons la configuration, l'installation et l'utilisation de base de la manière de générer des modules externes basés sur le langage de programmation Java pour l'environnement Blockly et nous pourrons créer des modules spécialisés pour chaque cas d'entreprise et insérer des fonctionnalités au système Mini BlocklyChain ou ajouter d'autres fonctionnalités à notre application mobile.

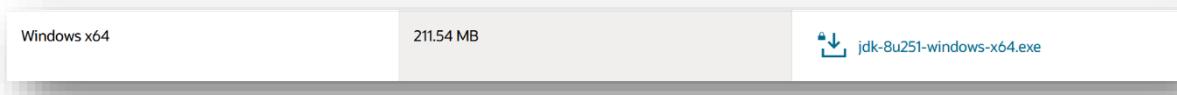
Nous avons créé dans notre système Windows un répertoire appelé "AppInventorExtensions" qui permet de télécharger les logiciels publics suivants.



Nous allons télécharger la dernière version du **JDK (Java Development Kit)**

exemple : jdk-8u251-windows-x64.exe (211 MB)

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>



2 - Bibliothèque **Apache Ant** qui utilise JAVA pour construire des applications, <http://ant.apache.org/bindownload.cgi>, dans mon cas j'ai téléchargé Ant 1.10.8 (Binary Distributions) (apache-ant-1.10.8-bin.zip). Il existe peut-être des versions plus avancées.

1.9.15 release - requires minimum of Java 5 at runtime

- 1.9.15 .zip archive: [apache-ant-1.9.15-bin.zip \[PGP\] \[SHA512\]](#)
- 1.9.15 .tar.gz archive: [apache-ant-1.9.15-bin.tar.gz \[PGP\] \[SHA512\]](#)
- 1.9.15 .tar.bz2 archive: [apache-ant-1.9.15-bin.tar.bz2 \[PGP\] \[SHA512\]](#)

1.10.8 release - requires minimum of Java 8 at runtime

- 1.10.8 .zip archive: [apache-ant-1.10.8-bin.zip \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.gz archive: [apache-ant-1.10.8-bin.tar.gz \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.bz2 archive: [apache-ant-1.10.8-bin.tar.bz2 \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.xz archive: [apache-ant-1.10.8-bin.tar.xz \[PGP\] \[SHA512\]](#)

Old Ant Releases

Older releases of Ant can be found [here](#). We highly recommend to not use those releases but upgrade to Ant's [latest](#) release.

Nous avons installé le Git Bash depuis votre site <https://git-scm.com/download/win>

Downloading Git



You are downloading the latest (**2.27.0**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **9 days ago**, on **2020-06-01**.

[Click here to download manually](#)

Other Git for Windows downloads

[Git for Windows Setup](#)

[32-bit Git for Windows Setup](#).

[64-bit Git for Windows Setup.](#)

[Git for Windows Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

The current source code release is version **2.27.0**. If you want the newer version, you can build it from [the source code](#).

4.- Dézippez "Apache Ant". Lorsque vous avez fini de décompresser, vous pouvez le faire dans un double dossier, par exemple :

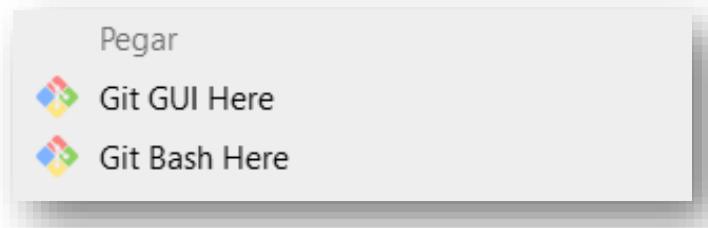
C : AppInventorExtensionsapache-ant-1.10.8-binapache-ant-1.10.8-bin

- Changez le en C : Apache-ant-1.10.8-bin

Nombre	Fecha de modificación	Tipo	Tamaño
bin	01/09/2019 11:43 a. m.	Carpeta de archivos	
etc	01/09/2019 11:43 a. m.	Carpeta de archivos	
lib	01/09/2019 11:43 a. m.	Carpeta de archivos	
manual	01/09/2019 11:43 a. m.	Carpeta de archivos	
CONTRIBUTORS	01/09/2019 11:43 a. m.	Archivo	7 KB
contributors	01/09/2019 11:43 a. m.	Documento XML	33 KB
fetch	01/09/2019 11:43 a. m.	Documento XML	14 KB
get-m2	01/09/2019 11:43 a. m.	Documento XML	5 KB
INSTALL	01/09/2019 11:43 a. m.	Archivo	1 KB
KEYS	01/09/2019 11:43 a. m.	Archivo	94 KB
LICENSE	01/09/2019 11:43 a. m.	Archivo	15 KB
NOTICE	01/09/2019 11:43 a. m.	Archivo	1 KB
patch	01/09/2019 11:43 a. m.	Documento XML	2 KB
README	01/09/2019 11:43 a. m.	Archivo	5 KB
WHATSNEW	01/09/2019 11:43 a. m.	Archivo	250 KB

5.- Nous avons installé Git Bash. Nous avons tout laissé par défaut dans l'installation.

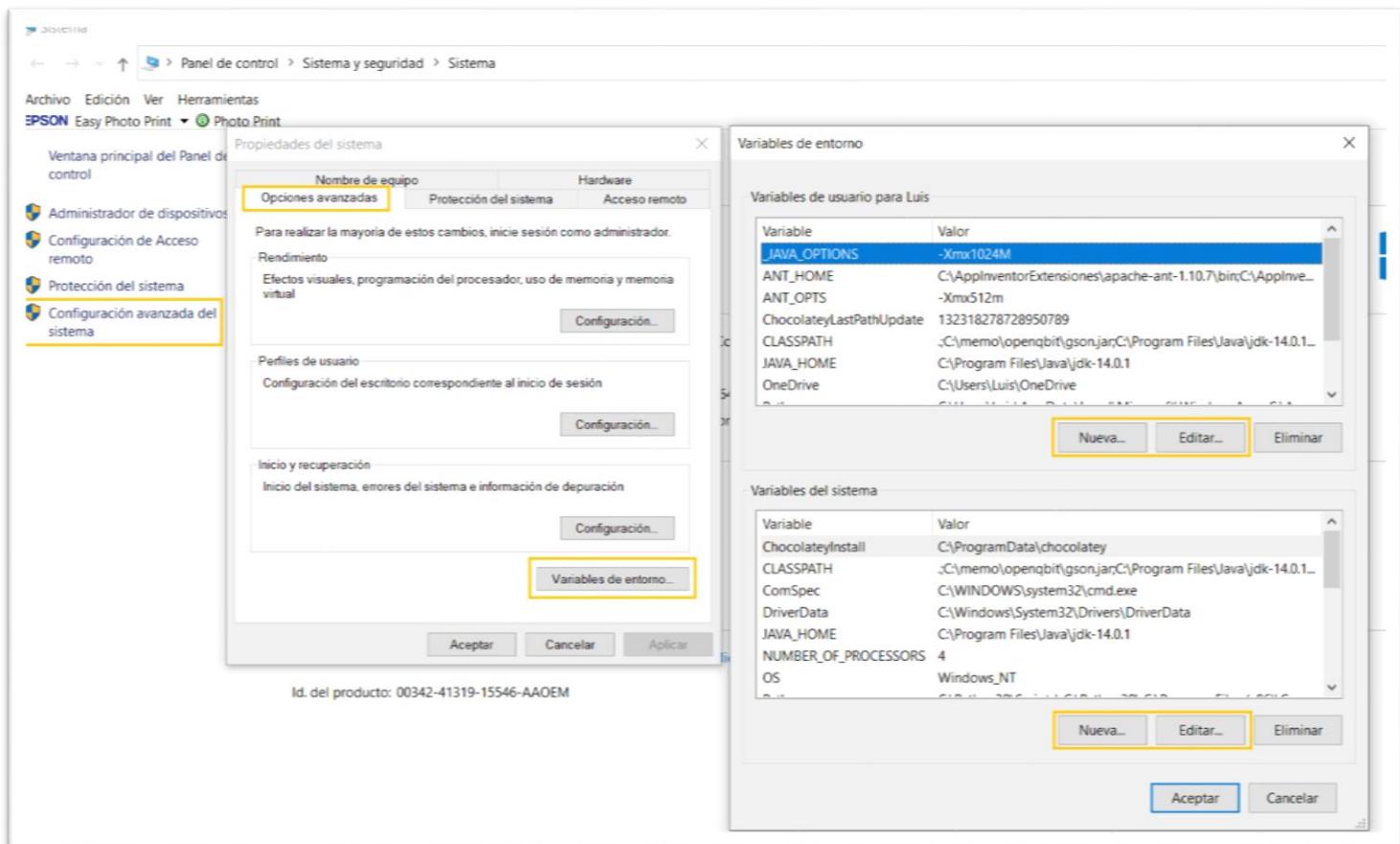
- Plus tard, nous pouvons voir qu'il y a un lien dans le bouton de démarrage de Windows, en cliquant sur le bouton gauche du navigateur de fichiers.



6.- Nous avons installé le kit de développement Java SE. Selon la version de Windows, vous devrez probablement redémarrer votre ordinateur.

Variables de l'environnement système de Windows. Nous allons créer les variables d'environnement. Pour ce faire, nous le ferons :

Panneau de configuration -> Système -> Paramètres système avancés -> Options avancées -> Variables d'environnement.



Selon que l'on souhaite mettre une nouvelle variable ou modifier une variable existante, on appuie sur le bouton "Nouveau..." ou "Modifier...".

Pour ajouter des adresses à celles déjà établies, nous les séparons par un point-virgule ;

Dans la section User Variables pour John, nous avons mis en place ces...

JAVA_OPTIONS nous vous mettons de valeur -Xmx1024m

ANT_HOME nous mettons de la valeur C : \AppInventorExtensions\apache-ant-1.10.8-bin [c'est-à-dire le dossier où nous avons décompressé apache-ant]

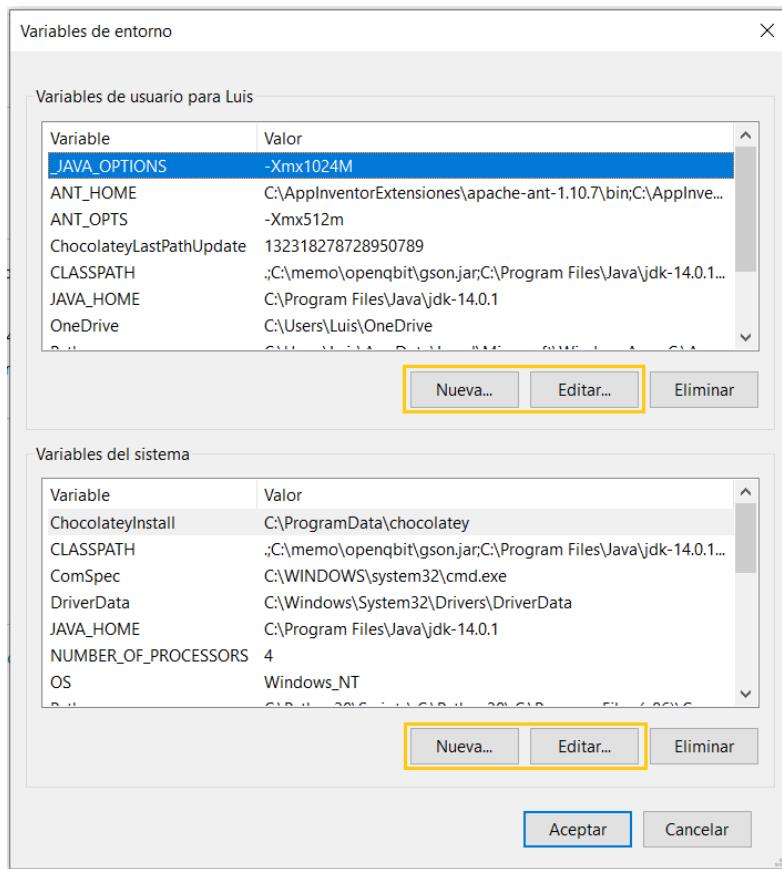
ANT_OPTS nous avons mis de la valeur -Xmx256M

JAVA_HOME est défini à C:³³³Program Files³³³jdk1.8.0_131 [S'il a une autre valeur, changez-la. Notez qu'il s'agit de jdk NOT jdr]

CLASSPATH we put of Value %ANT_HOME%\lib;%JAVA_HOME%\lib

Dans PATH nous avons ajouté ;%ANT_HOME%\bin;%JAVA_HOME%\bin [Notez que ; commence par un point-virgule ; à ajouter à ceux qui sont déjà là].

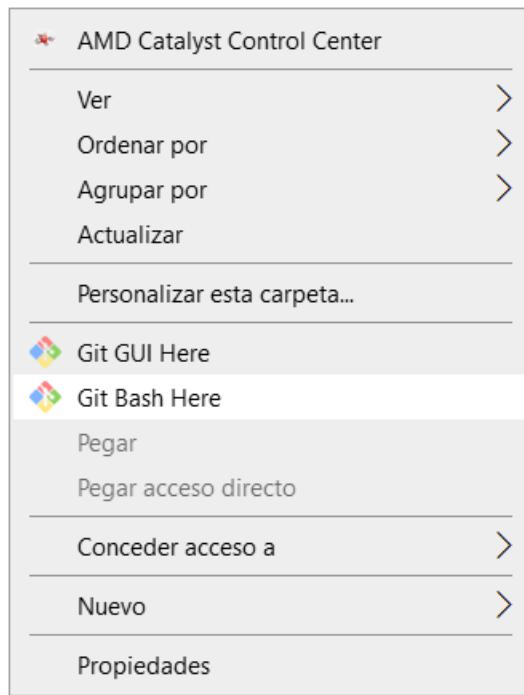
NOTE : Les variables sont séparées les unes des autres par un point-virgule : variable-1 ; variable-n ; variable-n+1



Création du clone de l'App Inventor dans notre ordinateur

Nous allons créer une copie "clone" de App Inventor sur notre serveur (PC), la télécharger directement sur Internet et créer cette copie.

Pour ce faire, nous utiliserons l'application **Git Bash** et cliquerons sur celle-ci pour ouvrir un terminal.



Nous dirigeons le commandement sur le terminal de Git Bash :

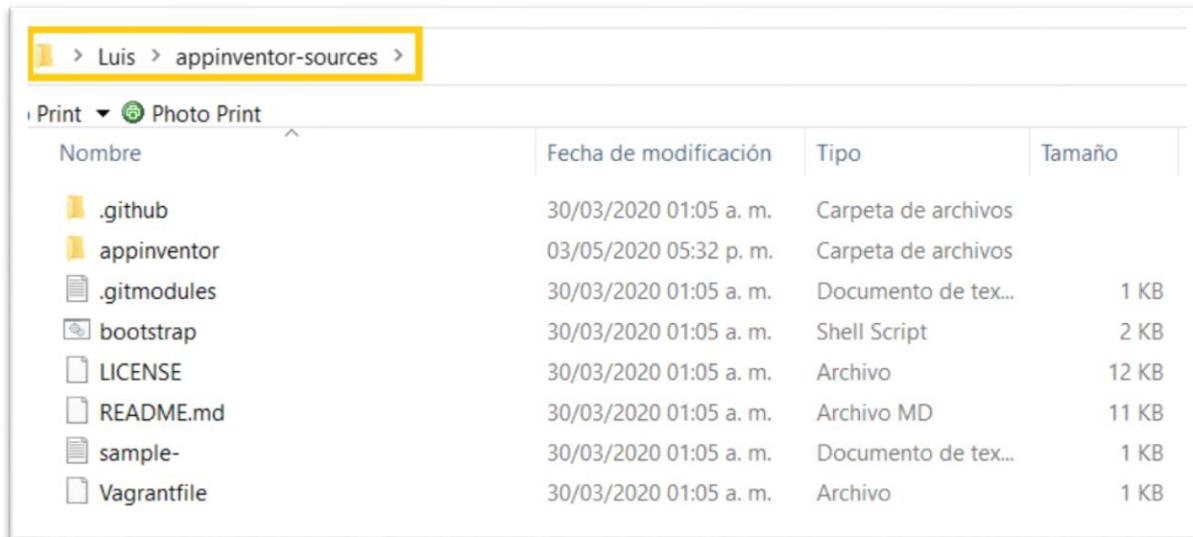
clone de \$ git <https://github.com/mit-cml/appinventor-sources.git>

```
uis@DESKTOP-LLGPLR6 MINGW64 ~
git clone https://github.com/mit-cml/appinventor-sources.git
Cloning into 'appinventor-sources'...
remote: Counting objects: 41191, done.
remote: Total 41191 (delta 0), reused 0 (delta 0), pack-reused 41190
receiving objects: 100% (41191/41191), 553.30 MiB | 494.00 KiB/s, done.
resolving deltas: 100% (21999/21999), done.
Checking out files: 100% (1758/1758), done.
uis@DESKTOP-LLGPLR6 MINGW64 ~
```

Le site où se trouve le dépôt :

<https://github.com/mit-cml/appinventor-sources/>

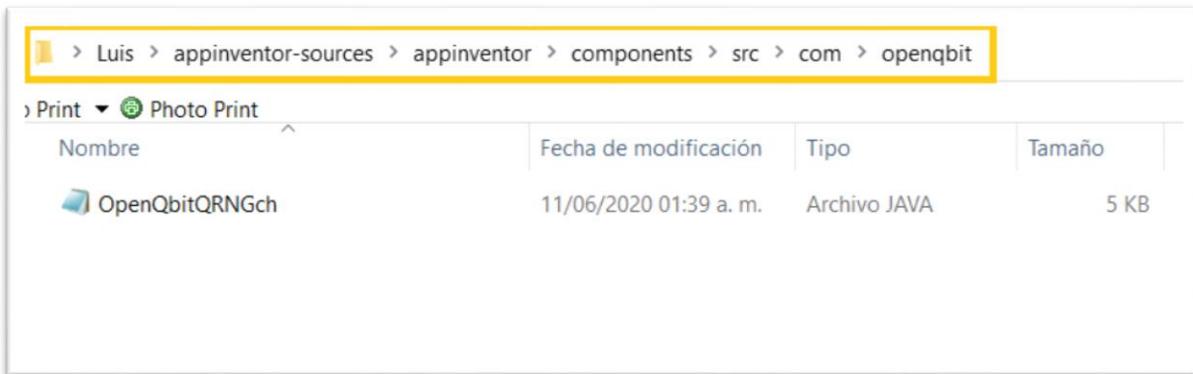
Il créera un dossier appelé appinventor-sources avec la source App Inventor.



Nombre	Fecha de modificación	Tipo	Tamaño
.github	30/03/2020 01:05 a. m.	Carpeta de archivos	
appinventor	03/05/2020 05:32 p. m.	Carpeta de archivos	
.gitmodules	30/03/2020 01:05 a. m.	Documento de tex...	1 KB
bootstrap	30/03/2020 01:05 a. m.	Shell Script	2 KB
LICENSE	30/03/2020 01:05 a. m.	Archivo	12 KB
README.md	30/03/2020 01:05 a. m.	Archivo MD	11 KB
sample-	30/03/2020 01:05 a. m.	Documento de tex...	1 KB
Vagrantfile	30/03/2020 01:05 a. m.	Archivo	1 KB

Nous avons créé le répertoire suivant dans le chemin C : Users - Appinventor-sourcesv-babkup - Appinventor-components -rc-openqbit

Nous avons copié à l'intérieur notre programme de test suivant appelé OpenQbitQRNGch.java



Nombre	Fecha de modificación	Tipo	Tamaño
OpenQbitQRNGch	11/06/2020 01:39 a. m.	Archivo JAVA	5 KB

Création de l'extension.

Respectez les majuscules et les minuscules, ce n'est pas la même chose Bonjour et Bonjour.

Ne mettez pas d'accents. Nous sommes prêts à faire notre première extension. Ce sera le générateur de nombres aléatoires quantiques.

Nous utilisons un éditeur de texte, Notepad++, nous créons un fichier appelé OpenQbitQRNGch.java qui génère des nombres quantiques aléatoires avec le code suivant :

```
// Cette extension est utilisée pour produire l'API du générateur
quantique de nombres aléatoires QRNG Switzerland.

paquet com.openqbit. OpenQbitQRNGch ;
import com.google.appinventor.components.annotations.DesignerComponent ;
import com.google.appinventor.components.annotations.DesignerProperty ;
import com.google.appinventor.components.annotations.PropertyCategory ;
import com.google.appinventor.components.annotations.SimpleEvent ;
import com.google.appinventor.components.annotations.SimpleFunction ;
import com.google.appinventor.components.annotations.SimpleObject ;
import com.google.appinventor.components.annotations.SimpleProperty ;
import com.google.appinventor.components.common.ComponentCategory ;
import com.google.appinventor.components.common.PropertyTypeConstants ;
import com.google.appinventor.components.runtime.util.MediaUtil ;
import com.google.appinventor.components.runtime.* ;
import java.io.* ;

@DesignerComponent(version = OpenQbitQRNGch.VERSION,
    description = "API Quantum Random Number Generator. Quantum random
numbers as a service, QRNGaaS, API web pour le générateur de nombres
aléatoires quantiques de Quantis développé par la société suisse - " +
"Guillermo Vidal - OpenQbit.com",
    catégorie = ComposanteCatégorie.EXTENSION
    nonVisible = vrai,
    iconName = "http://www.pinntar.com/logoQbit.png")
@SimpleObject(externe = vrai)

la classe publique OpenQbitQRNGch étend AndroidNonvisibleComponent
implémente Component {

    public static final int VERSION = 1 ;
    public statique final String DEFAULT_TEXT_1 = "" ;
    conteneur privé ComponentContainer ;
    private String text_1 = "" ;

    public OpenQbitQRNGch(ComponentContainer container) {
        super(container.$form()) ;
        this.container = conteneur ;
    }

    // Établissement des propriétés.
    //@DesignerProperty(editorType =
    PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue =
    OpenQbitQRNGch.DEFAULT_TEXTO_1 + "")
    //@SimpleProperty(description = "API Get Quantum Random Number Generator,
    nombre aléatoire entre 0 et 1. Entrez la variable *quantité* de nombres à
    générer")

    @SimpleFunction(description = "API Get Quantum Random Number Generator,
    random number between 0 and 1. Enter variable integer number *quantity*
    of numbers to generate")
```

```

public String APIGetQRNGdecimal(String qty) throws Exception {
    String url = "http://random.openqu.org/api/rand?size=" + qté ;
    Commande String[] = {"curl", url} ;

    ProcessBuilder process = nouveau ProcessBuilder(commande) ;
    Processus p ;
    p = process.start() ;
        Lecteur BufferedReader = nouveau BufferedReader(nouveau
InputStreamReader(p.getInputStream()) ) ;
        StringBuilder builder = nouveau StringBuilder() ;
        La ligne de caractères = nul ;
        while ((line = reader.readLine()) != null) {
            builder.append(ligne) ;

        builder.append(System.getProperty("line.separator")) ;
    }
    Résultat de la chaîne = builder.toString() ;
    //System.out.print(résultat) ;
    retour du résultat ;

}
@SimpleFunction(description = "API Get Quantum Random Number Generator,
nombre aléatoire entre une plage de nombres min à max. Entrer un nombre
entier variable *quantité* de nombres pour générer une plage de nombres
minimum *min* et maximum *max*")
public String APIGetQRNGinteger(String qty, String min, String max)
throws Exception {
    String url = "http://random.openqu.org/api/randint?size=" + qty +
"&min=" + min + "&max=" + max
    Commande String[] = {"curl", url} ;

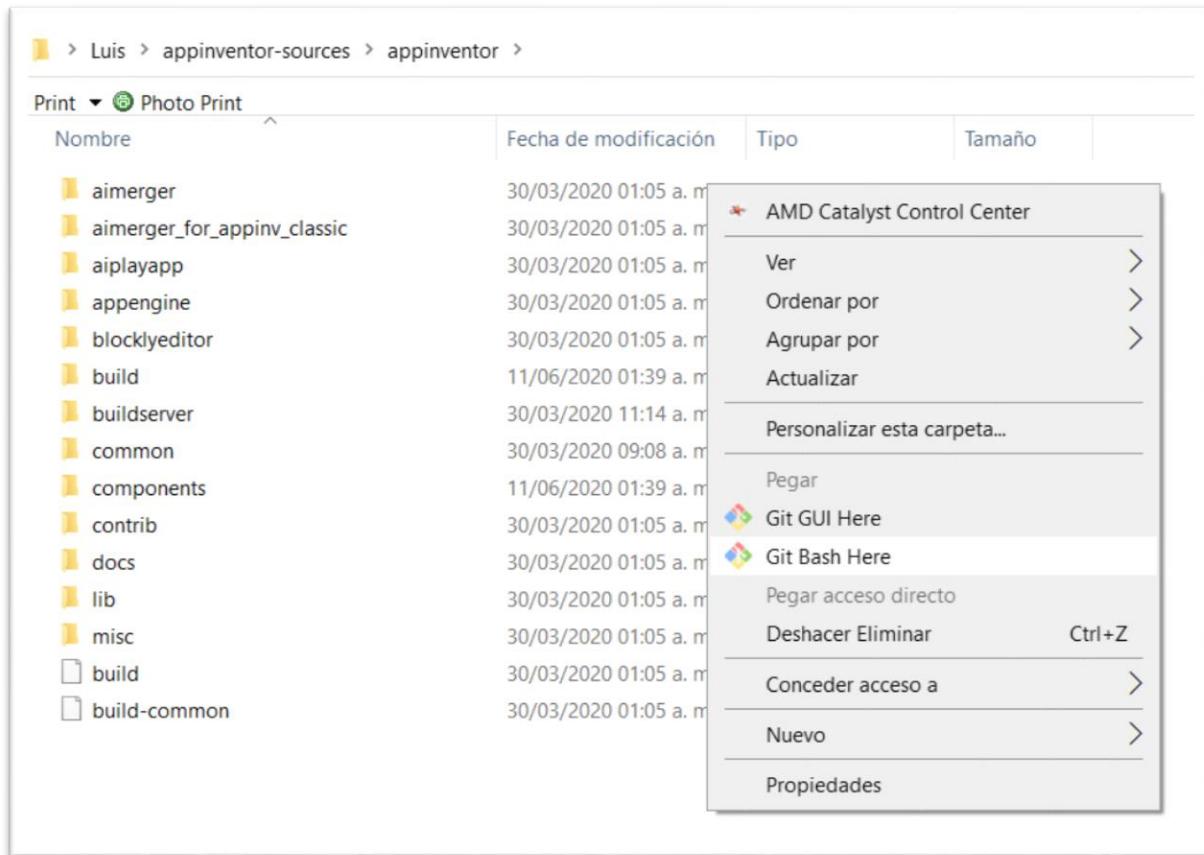
    ProcessBuilder process = nouveau ProcessBuilder(commande) ;
    Processus p ;
    p = process.start() ;
        Lecteur BufferedReader = nouveau BufferedReader(nouveau
InputStreamReader(p.getInputStream()) ) ;
        StringBuilder builder = nouveau StringBuilder() ;
        La ligne de caractères = nul ;
        while ((line = reader.readLine()) != null) {
            builder.append(ligne) ;

        builder.append(System.getProperty("line.separator")) ;
    }
    Résultat de la chaîne = builder.toString() ;
    //System.out.print(résultat) ;
    retour du résultat ;

}
}

```

Nous exécutons le **Git Bash en nous** positionnant sur la voie suivante : C : Luis-appinventeur-sources-appinventeur. Dans ce répertoire, on clique sur le bouton gauche de la souris et on sélectionne le terminal Git Bash.



Le terminal de Git bash sera positionné dans le répertoire suivant :

C:\Users\Luis\appinventor-sources\appinventor (master)

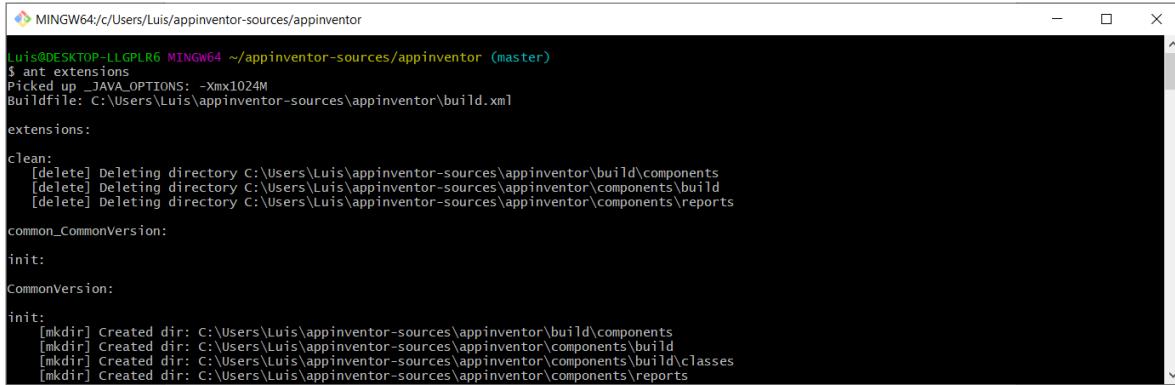
```

MINGW64:/c/Users/Luis/appinventor-sources/appinventor
Luis@DESKTOP-LLGPLR6 MINGW64 ~ /appinventor-sources/appinventor (master)
$ |

```

Dans le terminal Git Bash et exécutez la commande suivante :

\$ ant extensions



```
MINGW64:/c/Users/Luis/appinventor-sources/appinventor
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
$ ant extensions
Picked up _JAVA_OPTIONS: -Xmx1024M
Buildfile: C:\Users\Luis\appinventor-sources\appinventor\build.xml

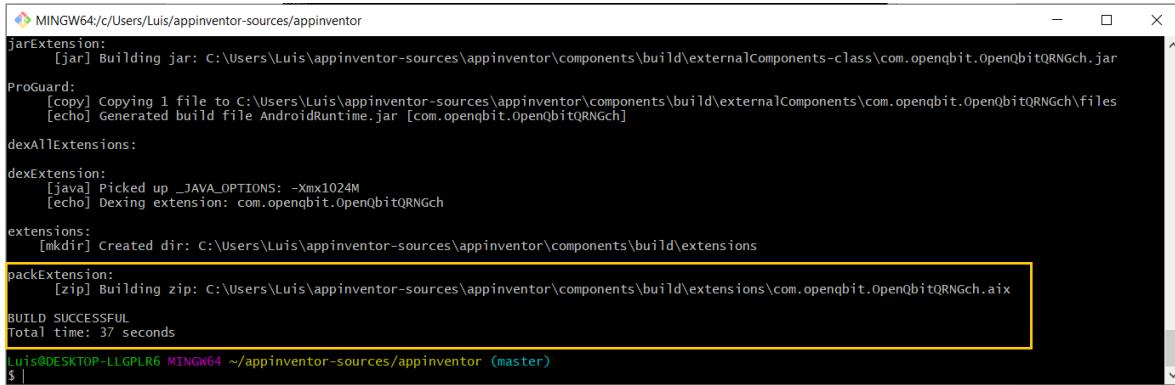
extensions:
clean:
[delete] Deleting directory C:\Users\Luis\appinventor-sources\appinventor\build\components
[delete] Deleting directory C:\Users\Luis\appinventor-sources\appinventor\components\build
[delete] Deleting directory C:\Users\Luis\appinventor-sources\appinventor\components\reports
common_CommonVersion:
init:
CommonVersion:
init:
[mkdir] Created dir: C:\Users\Luis\appinventor-sources\appinventor\build\components
[mkdir] Created dir: C:\Users\Luis\appinventor-sources\appinventor\components\build
[mkdir] Created dir: C:\Users\Luis\appinventor-sources\appinventor\components\build\classes
[mkdir] Created dir: C:\Users\Luis\appinventor-sources\appinventor\components\reports
```

Si tout s'est bien passé, nous l'aurons : CONSTRUIRE AVEC SUCCÈS.

Notre extension a été créée en...

C:\Users\Luis\appinventor-sources\appinventor-components\build\extensions

REMARQUE : le contenu du dossier des extensions est supprimé et recréé chaque fois que nous créons une extension ant.



```
MINGW64:/c/Users/Luis/appinventor-sources/appinventor
jarExtension:
[jar] Building jar: C:\Users\Luis\appinventor-sources\appinventor\components\build\externalComponents-class\com.openqbit.OpenQbitQRNGch.jar
ProGuard:
[copy] Copying 1 file to C:\Users\Luis\appinventor-sources\appinventor\components\build\externalComponents\com.openqbit.OpenQbitQRNGch\files
[echo] Generated build file AndroidRuntime.jar [com.openqbit.OpenQbitQRNGch]
dexAllExtensions:
dexExtension:
[java] Picked up _JAVA_OPTIONS: -Xmx1024M
[echo] Dexing extension: com.openqbit.OpenQbitQRNGch
extensions:
[mkdir] Created dir: C:\Users\Luis\appinventor-sources\appinventor\components\build\extensions
packExtension:
[zip] Building zip: C:\Users\Luis\appinventor-sources\appinventor\components\build\extensions\com.openqbit.OpenQbitQRNGch.aix
BUILD SUCCESSFUL
Total time: 37 seconds
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
$ |
```

Allons voir ce dossier :

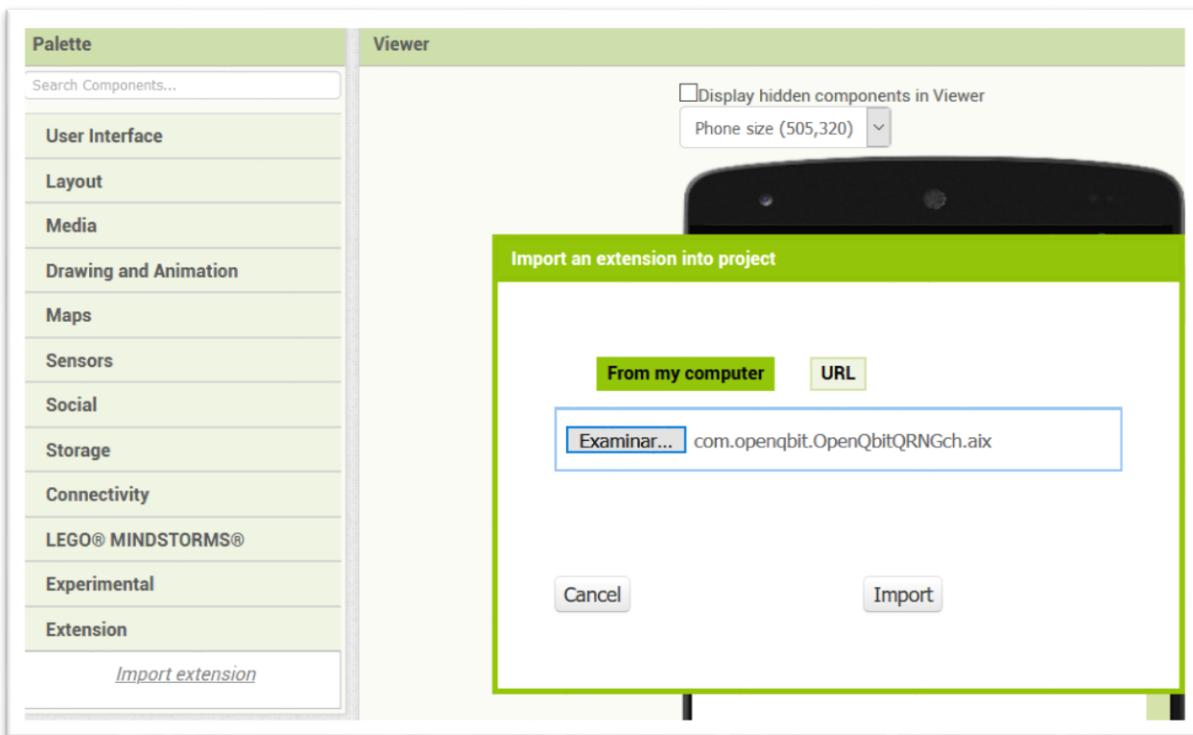
C:\Users\Luis\appinventor-sources\appinventor-components\build\extensions

et copiez le fichier com.openqbit.OpenQbitQRNGch.aix, ce fichier est notre extension.

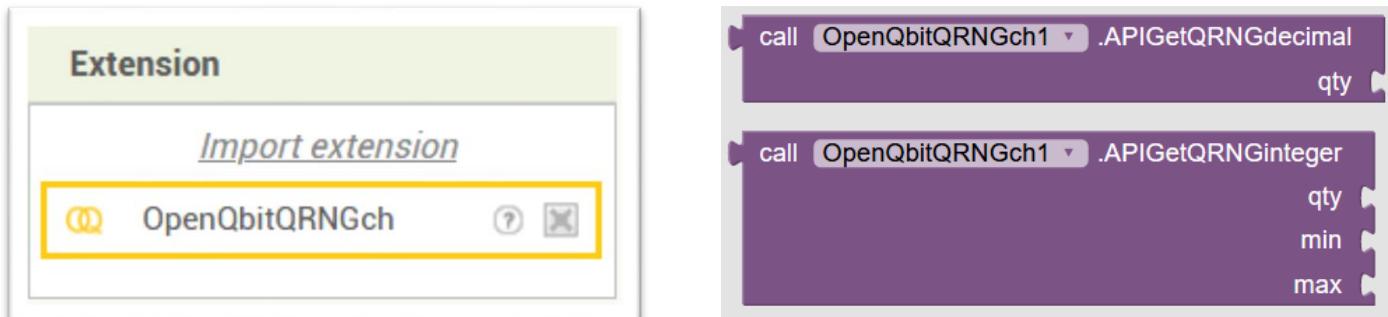
Ayant déjà un compte dans App Inventor ou un autre système Blockly, nous sommes entrés pour ajouter la nouvelle extension et la tester.

Allez au bas de la page où se trouve l'option Extension et cliquez sur "Importer l'extension"

:



On appuie sur le bouton "Importer". Nous avons maintenant les blocs (APIGetQRNGdécimal) et (APIGetQRNGintégrateur) à utiliser :



Nous avons déjà notre première extension créée et fonctionnelle.

27. Annexe "Contrats intelligents BlocklyCode".

Les BlocklyCodes sont des programmes réalisés en langage java. L'extension permettant de créer ce type de programme communément appelé "Smart Contracts" est un moyen de traiter les accords entre utilisateurs (entreprises ou personnes).

Ce bloc (**BlocklyCode**), est implémenté dans un programme qui a déjà des paramètres établis et qui selon le type d'accords qui pourraient être exécutés par le système Mini BlocklyChain de façon automatique lorsque les prémisses qui régissent le "Smart Contract" sont remplies.

Les paramètres à prendre en considération pour les "entrées" proposées ou les paramètres d'entrée sont les suivants

Date d'expiration

Date de référence

Délai d'expiration

Temps de référence

Actif référencé

Total des actifs immobilisés

Actifs variables

Membres contractuels

Données confirmées (String)

Données confirmées (Nom de fichier)

Événement variable

Événement fixe

Validation du/des document(s)

Validation des chaînes

Signature valable

Intervalle défini (entier)

Intervalle décimal

Minimum établi

Maximum établi

Avant de commencer à utiliser le bloc (**BlocklyCode**), nous devons d'abord installer le système qui effectuera l'exécution des "Smart Contracts", ceci est fait en installant dans le terminal de Termux OpenJDK et OpenJRE.

OpenJDK (Open Java Development Kit). - C'est l'outil pour développer des programmes en java, il contient les bibliothèques, le compilateur et la JVM (Java Virtual Machine).

OpenJRE (Open Java Runtime Environment). - C'est l'outil pour exécuter des programmes java uniquement. Il contient la JVM (Java Virtual Machine).

Nous procédons à l'installation de OpenJRE et OpenJDK dans le terminal Termux des nœuds qui forment le système Mini BlocklyChain.

Nous avons installé les salles

\$ apt install - et wget



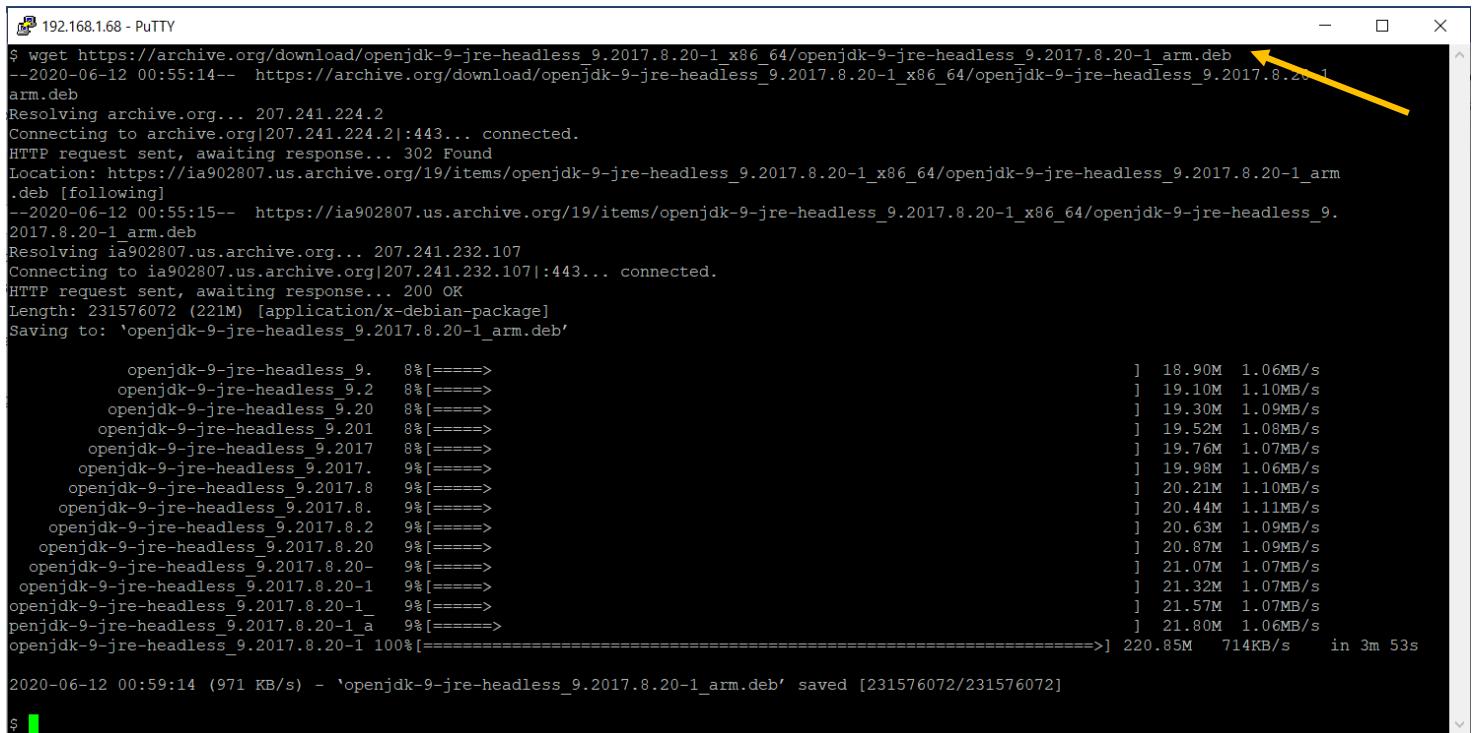
The screenshot shows a Termux terminal window. At the top, it displays the status bar with signal strength, battery level at 100%, and the time 1:07 a.m. Below the status bar, the terminal prompt is '\$ apt install wget'. A yellow arrow points to this prompt. The terminal then shows the output of the command:

```
$ apt install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  wget
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 206 kB of archives.
After this operation, 430 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm wget arm 1.20.3-2 [206 kB]
Fetched 206 kB in 1s (128 kB/s)
Selecting previously unselected package wget.
(Reading database ... 16936 files and directorie
s currently installed.)
Preparing to unpack .../archives/wget_1.20.3-2_a
rm.deb ...
Unpacking wget (1.20.3-2) ...
Setting up wget (1.20.3-2) ...
```

At the bottom of the terminal window, there is a standard Android-style keyboard with a numeric keypad and various function keys like ESC, CTRL, ALT, and navigation keys.

Nous avons téléchargé l'OpenJRE

\$ wget https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb



```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
--2020-06-12 00:55:14-- https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org[207.241.224.2]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 00:55:15-- https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving ia902807.us.archive.org... 207.241.232.107
Connecting to ia902807.us.archive.org[207.241.232.107]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231576072 (221M) [application/x-debian-package]
Saving to: 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb'

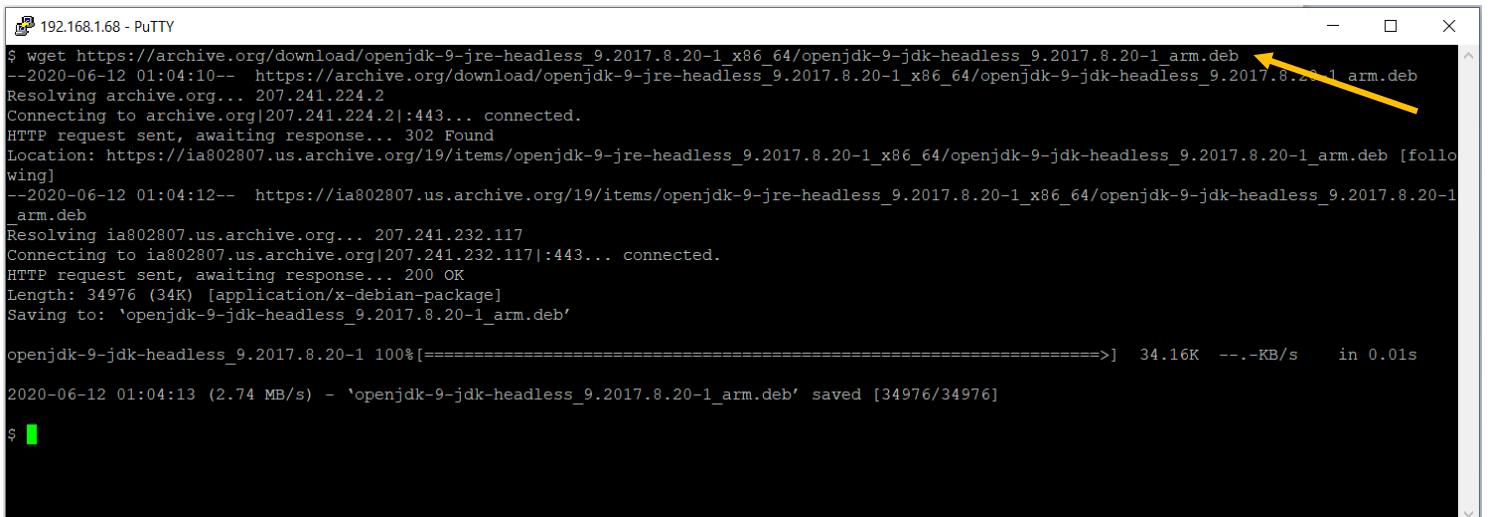
openjdk-9-jre-headless_9. 8%[=====] 18.90M 1.06MB/s
openjdk-9-jre-headless_9.2 8%[=====] 19.10M 1.10MB/s
openjdk-9-jre-headless_9.20 8%[=====] 19.30M 1.09MB/s
openjdk-9-jre-headless_9.201 8%[=====] 19.52M 1.08MB/s
openjdk-9-jre-headless_9.2017 8%[=====] 19.76M 1.07MB/s
openjdk-9-jre-headless_9.2017. 9%[=====] 19.98M 1.06MB/s
openjdk-9-jre-headless_9.2017.8 9%[=====] 20.21M 1.10MB/s
openjdk-9-jre-headless_9.2017.8. 9%[=====] 20.44M 1.11MB/s
openjdk-9-jre-headless_9.2017.8.2 9%[=====] 20.63M 1.09MB/s
openjdk-9-jre-headless_9.2017.8.20 9%[=====] 20.87M 1.09MB/s
openjdk-9-jre-headless_9.2017.8.20- 9%[=====] 21.07M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1 9%[=====] 21.32M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_ 9%[=====] 21.57M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_a 9%[=====] 21.80M 1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1 100%[=====] 220.85M 714KB/s in 3m 53s

2020-06-12 00:59:14 (971 KB/s) - 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb' saved [231576072/231576072]

$
```

Nous avons téléchargé l'OpenJDK

\$ wget https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb



```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
--2020-06-12 01:04:10-- https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org[207.241.224.2]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 01:04:12-- https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving ia802807.us.archive.org... 207.241.232.117
Connecting to ia802807.us.archive.org[207.241.232.117]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34976 (34K) [application/x-debian-package]
Saving to: 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb'

openjdk-9-jdk-headless_9.2017.8.20-1 100%[=====] 34.16K --.-KB/s in 0.01s

2020-06-12 01:04:13 (2.74 MB/s) - 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb' saved [34976/34976]

$
```

Nous effectuons l'installation depuis le terminal Termux d'OpenJDK et OpenJRE :

```
$ apt install - and ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
```



```
$ ls
openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
$ apt install -y ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'openjdk-9-jdk-headless' instead
of './openjdk-9-jdk-headless_9.2017.8.20-1_arm.
deb'
Note, selecting 'openjdk-9-jre-headless' instead
of './openjdk-9-jre-headless_9.2017.8.20-1_arm.
deb'
The following additional packages will be instal
led:
  ca-certificates-java freetype libpng
The following NEW packages will be installed:
  ca-certificates-java freetype libpng
  openjdk-9-jdk-headless
  openjdk-9-jre-headless
0 upgraded, 5 newly installed, 0 to remove and 0
not upgraded.
Need to get 668 kB/232 MB of archives.
After this operation, 376 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ages-24 stable/main arm ca-certificates-java all
20200101 [110 kB]
Get:2 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm libpng arm 1.6.37-2 [190
kB]
Get:3 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm freetype arm 2.10.2 [368
kB]
Get:4 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jre-headless_9.2017.8.20-1_arm.deb open
jdk-9-jre-headless arm 9.2017.8.20-1 [232 MB]
Get:5 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jdk-headless_9.2017.8.20-1_arm.deb open
jdk-9-jdk-headless arm 9.2017.8.20-1 [35.0 kB]

ESC      ←      CTRL      ALT      –      ↓      ↑
          □      ○      □
```



```
Get:5 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jdk-headless_9.2017.8.20-1_arm.deb open
jdk-9-jdk-headless arm 9.2017.8.20-1 [35.0 kB]
Fetched 668 kB in 23s (28.0 kB/s)
Selecting previously unselected package ca-certifi
cates-java.
(Reading database ... 16939 files and directorie
s currently installed.)
Preparing to unpack .../ca-certificates-java_202
00101_all.deb ...
Unpacking ca-certificates-java (20200101) ...
Selecting previously unselected package libpng.
Preparing to unpack .../libpng_1.6.37-2_arm.deb
...
Unpacking libpng (1.6.37-2) ...
Selecting previously unselected package freetype
.
Preparing to unpack .../freetype_2.10.2_arm.deb
...
Unpacking freetype (2.10.2) ...
Selecting previously unselected package openjdk-
9-jre-headless.
Preparing to unpack .../openjdk-9-jre-headless_9
.2017.8.20-1_arm.deb ...
Unpacking openjdk-9-jre-headless (9.2017.8.20-1)
...
Selecting previously unselected package openjdk-
9-jdk-headless.
Preparing to unpack .../openjdk-9-jdk-headless_9
.2017.8.20-1_arm.deb ...
Unpacking openjdk-9-jdk-headless (9.2017.8.20-1)
...
Setting up libpng (1.6.37-2) ...
Setting up freetype (2.10.2) ...
Setting up ca-certificates-java (20200101) ...
Setting up openjdk-9-jre-headless (9.2017.8.20-1
) ...
Setting up openjdk-9-jdk-headless (9.2017.8.20-1
) ...
$ █

ESC      ←      CTRL      ALT      –      ↓      ↑
          □      ○      □
```

Depuis que nous avons terminé l'installation de l'environnement OpenJDK et OpenJRE. Ces installations contiennent la JVM (Java Virtual Machine) qui sera l'environnement qui fera fonctionner le BlocklyCode.

Nous allons maintenant installer un éditeur pour créer un fichier en ligne, nous allons utiliser l'éditeur appelé **vi** et exécuter la commande suivante :

```
$ apt install vim
```

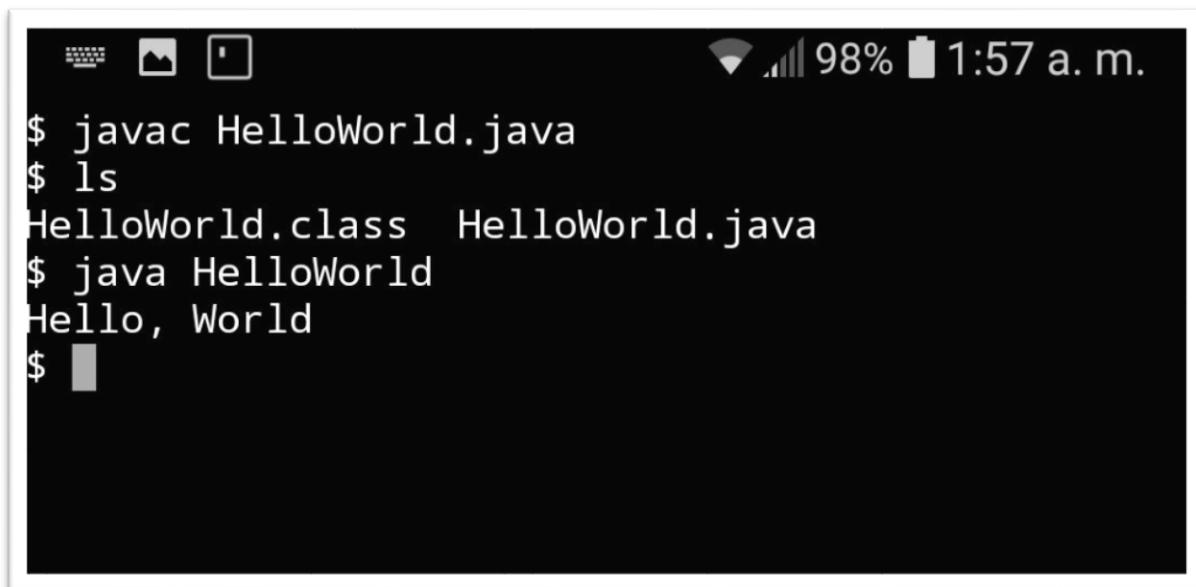
Essayons maintenant de compiler un fichier test et de l'exécuter. Nous créons dans le terminal Termux avec l'éditeur **vi** et nous écrivons le code java d'un "Hello World", et nous le sauvegardons dans le fichier HelloWorld.java

```
classe publique HelloWorld {  
  
    public static void main(String[] args) {  
        // Imprime "Hello, World" sur la fenêtre du terminal.  
        System.out.println("Bonjour, le monde") ;  
    }  
}
```

Ensuite, nous exécutons la commande suivante dans le terminal Termux pour la compilation puis l'exécution du programme :

```
javac HelloWorld.java (après exécution, un fichier de code d'octet HelloWorld.class est créé)
```

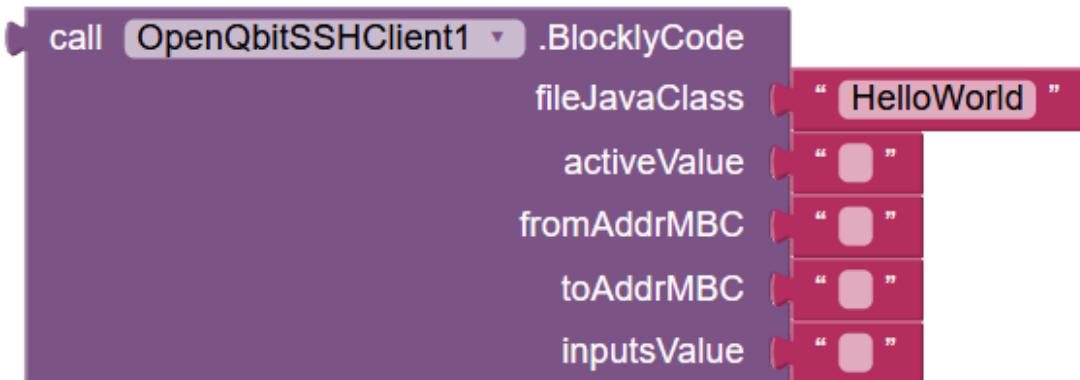
```
$ java HelloWorld (Après avoir diffusé et imprimé l'annonce, Hello World)
```



The screenshot shows a Termux terminal window with a black background and white text. At the top, there are icons for network, battery (98%), and time (1:57 a.m.). The terminal output is as follows:

```
$ javac HelloWorld.java  
$ ls  
HelloWorld.class  HelloWorld.java  
$ java HelloWorld  
Hello, World  
$
```

En utilisant le bloc (**BlocklyCode**), ce bloc est situé dans l'extension (**ConnectorSSHClient**).



Dans le bloc précédent, vous pouvez voir comment sera exécuté le fichier HelloWorld.class, qui a déjà été compilé et positionné dans le répertoire de base des utilisateurs.

Il s'agit d'un bloc dans lequel il est possible d'exécuter un programme déjà compilé en java, qui est communément connu dans l'environnement de développement sous le nom de fichier bytecode sous sa forme binaire.

Ce type de fichier est prêt à être exécuté par la machine virtuelle Java (JVM).

Il y a deux façons de créer un fichier bytecode ou avec l'extension .CLASS, l'utilisateur peut le créer dans son PC par confort ou son semblable peut également être fait dans le téléphone mobile, rappelez-vous que nous avons déjà installé l'environnement pour compiler le JDK (Java Development Kit) auparavant, bien qu'il sera moins efficace parce que la limitation du clavier comptera, aussi en cas de choix de l'environnement Android devra tenir compte du fait que les bibliothèques sont limitées à l'OpenJDK qui a été installé.

Les paramètres d'entrée sont ouverts dans <inputsValue> et les autres paramètres fixes sont ceux de activeValue, deaddrMBC et deAddrMBC.

Dans le cas des tests d'exécution, ils peuvent être laissés vides sans contenu et seul le fichier bytecode sera exécuté sans paramètres.

Le bloc précédent est comme la ligne de commande suivante en cours d'exécution :

\$ java HelloWorld

Les programmes développés pour BlocklyCode seront soumis à chaque environnement de conception particulier et pourront être contrôlés et exécutés par routine avec l'agent "cron" et partagés avec le syncthingmanager.

28. Annexe "Informatique quantique à OpenQbit".

Comment fonctionne l'informatique quantique ?⁽²⁾

La transformation numérique entraîne des changements dans le monde plus rapides que jamais. Croyez-vous que l'ère numérique est sur le point de prendre fin ? La **culture numérique** a déjà été identifiée comme un domaine dans lequel il est urgent de disposer de connaissances ouvertes et de possibilités accessibles d'apprentissage de la technologie afin de combler les lacunes en matière de développement social et économique. L'apprentissage des concepts clés de l'ère numérique deviendra encore plus crucial avec l'arrivée imminente d'une autre nouvelle vague technologique capable de transformer les modèles existants avec une rapidité et une puissance étonnantes : les **technologies quantiques**.

Dans cet article, nous comparons les concepts de base de l'informatique traditionnelle et de l'informatique quantique ; et nous commençons également à explorer leur application dans d'autres domaines connexes.

Que sont les technologies quantiques ?

Tout au long de l'histoire, les êtres humains ont développé la technologie à mesure qu'ils ont compris comment la nature fonctionne grâce à la science. Entre 1900 et 1930, l'étude de certains phénomènes physiques encore mal compris a donné naissance à une nouvelle théorie physique, la **mécanique quantique**. Cette théorie décrit et explique le fonctionnement du monde microscopique, l'habitat naturel des molécules, des atomes ou des électrons. Grâce à cette théorie, non seulement il a été possible d'expliquer ces phénomènes, mais il a également été possible de comprendre que la réalité subatomique fonctionne d'une manière totalement contre-intuitive, presque magique, et que dans le monde microscopique se produisent des événements qui ne se produisent pas dans le monde macroscopique.

Ces **propriétés quantiques** comprennent la superposition quantique, l'intrication quantique et la téléportation quantique.

- **La superposition quantique** décrit comment une particule peut se trouver dans différents états en même temps.
- **L'intrication quantique** décrit comment deux particules aussi éloignées l'une de l'autre que souhaité peuvent être corrélées de telle sorte que, lorsqu'elles interagissent avec l'une d'entre elles, l'autre en est consciente.
- **La téléportation** quantique utilise l'enchevêtrement quantique pour envoyer des informations d'un endroit à un autre dans l'espace sans avoir à le traverser.

Les technologies quantiques sont basées sur ces propriétés quantiques de nature subatomique.

Dans ce cas, la compréhension du monde microscopique par la mécanique quantique nous permet aujourd'hui d'inventer et de concevoir des technologies capables d'améliorer la vie des gens. Il existe de nombreuses et très différentes technologies qui utilisent des phénomènes quantiques et certaines d'entre elles, comme les lasers ou l'imagerie par résonance magnétique (IRM), sont présentes depuis plus d'un demi-siècle. Cependant, nous assistons actuellement à une révolution technologique dans des domaines tels que l'informatique quantique, l'information quantique, la simulation quantique, l'optique quantique, la métrologie quantique, les horloges ou les capteurs quantiques.

Qu'est-ce que l'informatique quantique ? Tout d'abord, il faut comprendre l'informatique classique.



Carácter	Bits
7	111
A	01000001
\$	00100100
:)	0011101000101001

FIGURA 1.
Ejemplos de caracteres en lenguaje binario.

Pour comprendre le fonctionnement des ordinateurs quantiques, il convient d'abord d'expliquer comment fonctionnent les ordinateurs que nous utilisons tous les jours, que nous appellerons dans ce document ordinateurs numériques ou classiques. Ceux-ci, comme le reste des appareils électroniques tels que les tablettes ou les téléphones portables, utilisent les bits comme unités fondamentales de la mémoire. Cela signifie que les programmes et les applications sont encodés en bits, c'est-à-dire en langage binaire de zéros et de uns. Chaque fois que nous interagissons avec l'un de ces dispositifs, par exemple en appuyant sur une touche du clavier, des chaînes de zéros et de uns sont créées, détruites et/ou modifiées à l'intérieur de l'ordinateur.

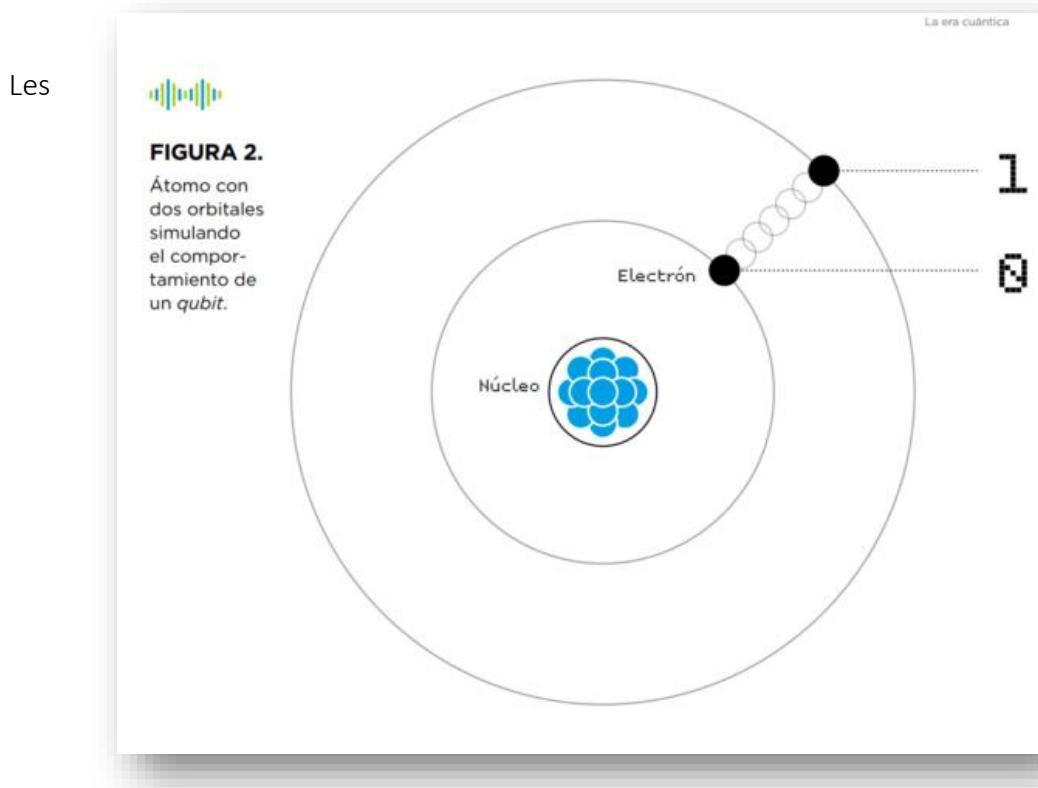
La question intéressante est de savoir ce que sont ces zéros et ces uns physiquement à l'intérieur de l'ordinateur. Les états zéro et un correspondent à un courant électrique qui circule, ou non, à travers des parties microscopiques appelées transistors, qui agissent comme des interrupteurs. Lorsqu'aucun courant ne circule, le transistor est "off" et correspond au bit 0, et lorsqu'il circule, il est "on" et correspond au bit 1.

Plus simplement, c'est comme si les bits 0 et 1 correspondaient à des trous, de sorte qu'un trou vide est un bit 0 et un trou occupé par un électron est un bit 1. C'est pourquoi ces appareils sont appelés électroniques. À titre d'exemple, la figure 1 montre l'écriture binaire

de certains caractères. Maintenant que nous avons une idée du fonctionnement des ordinateurs d'aujourd'hui, essayons de comprendre comment fonctionnent les quanta.

Des bits aux qubits

L'unité fondamentale de l'information dans l'informatique quantique est le bit ou qubit quantique. Les qubits sont, par définition, des systèmes quantiques à deux niveaux - nous en verrons des exemples ici - qui, comme les bits, peuvent être au niveau bas, qui correspond à un état de faible excitation ou d'énergie défini comme 0, ou au niveau haut, qui correspond à un état d'excitation plus élevé ou défini comme 1. Cependant, et c'est là que réside la différence fondamentale avec le calcul classique, les qubits peuvent également se trouver dans n'importe lequel des états intermédiaires infinis entre 0 et 1, comme un état qui est la moitié de 0 et la moitié de 1, ou les trois quarts de 0 et un quart de 1.



algorithmes quantiques, un calcul exponentiellement plus puissant et plus efficace

L'objectif des ordinateurs quantiques est de tirer parti de ces propriétés quantiques des *qubits*, en tant que systèmes quantiques, afin d'exécuter des algorithmes quantiques qui utilisent le chevauchement et l'entrelacement pour fournir une puissance de traitement beaucoup plus importante que les classiques. Il est important de souligner que le véritable changement de paradigme ne consiste pas à faire la même chose que les ordinateurs numériques ou classiques - les actuels - mais plus rapidement, comme on peut le lire dans

de nombreux articles, mais que les algorithmes quantiques permettent d'effectuer certaines opérations d'une manière totalement différente qui, dans de nombreux cas, s'avère plus efficace - c'est-à-dire en beaucoup moins de temps ou en utilisant beaucoup moins de ressources informatiques -.

Voyons un exemple concret de ce que cela implique. Imaginons que nous soyons à Bogota et que nous voulions connaître le meilleur itinéraire pour se rendre à Lima parmi un million de possibilités pour y arriver ($N=1.000.000$). Afin d'utiliser les ordinateurs pour trouver la voie optimale, nous devons numériser 1 000 000 d'options, ce qui implique de les traduire en langage binaire pour l'ordinateur classique et en *qubits pour l'ordinateur* quantique. Alors qu'un ordinateur classique devrait analyser un par un tous les chemins jusqu'à trouver celui qu'il souhaite, un ordinateur quantique tire parti du processus connu sous le nom de parallélisme quantique qui lui permet de considérer tous les chemins en même temps. Cela implique que, alors que l'ordinateur classique a besoin de l'ordre de $N/2$ étapes ou itérations, c'est-à-dire 500 000 tentatives, l'ordinateur quantique trouvera le chemin optimal après seulement des opérations \sqrt{N} sur le registre, soit 1 000 tentatives.

Dans le cas précédent, l'avantage est quadratique, mais dans d'autres cas, il est même exponentiel, ce qui signifie qu'avec n *qubits*, nous pouvons obtenir une capacité de calcul équivalente à 2^n bits. Pour illustrer ce point, il est courant de compter qu'avec environ 270 qubits, nous pourrions avoir plus d'états de base dans un ordinateur quantique - plus de chaînes de caractères différentes et simultanées - que le nombre d'atomes dans l'univers, qui est estimé à environ 280. Un autre exemple est qu'on estime qu'avec un ordinateur quantique de 2000 à 2500 *qubits*, nous pourrions casser pratiquement toute la cryptographie utilisée aujourd'hui (la cryptographie dite à clé publique).

Pourquoi est-il important de connaître la technologie quantique ?

Nous sommes dans un moment de transformation numérique où les différentes technologies émergentes telles que la chaîne de blocs, l'intelligence artificielle, les drones, l'Internet des objets, la réalité virtuelle, les imprimantes 5G, 3D, les robots ou les véhicules autonomes sont de plus en plus présentes dans de multiples domaines et secteurs. Ces technologies, appelées à améliorer la qualité de vie de l'être humain en accélérant le développement et en générant un impact social, progressent aujourd'hui de manière parallèle. Il est rare de voir des entreprises développer des produits qui exploitent des combinaisons de deux ou plusieurs de ces technologies, comme la chaîne de blocs et l'IdO ou les drones et l'intelligence artificielle. Bien qu'elles soient destinées à converger, générant ainsi un impact exponentiellement plus important, le stade initial de développement dans lequel elles se trouvent et la rareté des développeurs et des personnes ayant un profil technique font que la convergence est encore une tâche en suspens.

En raison de leur potentiel perturbateur, les technologies quantiques devraient non seulement converger avec toutes ces nouvelles technologies, mais aussi avoir une influence

transversale sur pratiquement toutes d'entre elles. L'informatique quantique menacera l'authentification, l'échange et le stockage sécurisé des données, ce qui aura un impact majeur sur les technologies où la cryptographie joue un rôle plus important, comme la cybersécurité ou la chaîne de blocage, et un impact négatif mineur, mais qui doit également être pris en compte dans les technologies telles que la 5G, l'IdO ou les drones.

Vous voulez pratiquer l'informatique quantique ?

Des dizaines de simulateurs d'ordinateurs quantiques sont déjà disponibles sur le net avec différents langages de programmation déjà utilisés tels que C, C++, Java, Matlab, Maxima, Python ou Octave. De plus, de nouveaux langages comme Q#, lancé par Microsoft. Vous pouvez explorer et jouer avec une machine quantique virtuelle grâce à des plateformes telles qu'IBM et Rigetti.

Mini BlocklyChain est créé par la société OpenQbit.com qui se concentre sur le développement de la technologie de l'informatique quantique pour différents types de secteurs privés et publics.

Pourquoi Mini BlocklyChain est différent des autres chaînes de blocs, simplement parce que le système a été créé pour être modulaire et inclure l'informatique quantique.

(2) <https://blogs.iadb.org/conocimiento-abierto/es/como-funciona-la-computacion-cuantica/>

29. Annexe "Blocs étendus pour la base de données SQLite".

Pour plus de détails sur l'utilisation correcte et opportune de chaque bloc qui constitue l'extension OpenQbitSQLite, veuillez consulter l'auteur original de l'extension dans le lien suivant.

OpenQbitSQLite est un clone de la conception originale avec de petites modifications pour être utilisé avec un niveau de sécurité AES.

<https://github.com/frdfsnlght/aix-SQLite>

30. Annexe "Exemple de création d'un système de Mini BlocklyChain".

Nous allons mettre en place un système de test permettant de vérifier les fonctionnalités, les avantages et la facilité de création d'un système de Mini BlocklyChain.

Nous commencerons par la conception et le développement du stockage où résideront les informations, que nous avons déjà défini comme une chaîne de blocs. La conception sera basée sur la base de données SQLite et, en fonction de chaque organisation ou conception, elle pourra être facilement modifiée par une autre base de données telle que Microsoft SQL

Server, MySQL, Maria DB, Oracle, DB2, PostgreSQL, entre autres. Bien que, là encore, en raison du processus de transactions et de la concurrence, la base de données SQLite peut être utilisée à tout niveau et pour une application professionnelle ou personnelle.

La création de la base de données appelée **minibc** dans ce cas aura une table qui stockera la chaîne du bloc. Cette base de données sera intégrée dans le code de programme final qui sera installé dans les nœuds, ce lieu de stockage est appelé "assets packaged" est une espèce interne dans les environnements Blockly tels que App Inventor.

```
$ sqlite3 minibc.db
SQLite version 3.32.2 2020-06-20 15:25:24
Entrez ".help" pour les conseils d'utilisation.
Connecté à une base de données en mémoire transitoire.
Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.
sqlite> .quit
```

Conception des champs de la table **nblock**.

```
CREER TABLE nbloc (
    id clé primaire entière AUTOINCREMENT
    , prevhashVARCHAR PAS NUL
    , newhashVARCHAR PAS NUL
    ntransINTEGER NON NUL
    nonceINTEGER NON NUL
    ,qrng      ENTIER NON NUL
);
```

Nous avons créé le tableau des **blocs**.

```
$ sqlite3 minibc.db
SQLite version 3.32.2 2020-06-20 15:25:24
Entrez ".help" pour les conseils d'utilisation.
Connecté à une base de données en mémoire transitoire.
Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.
sqlite> .open minibc.db
```

```
sqlite> CREATE TABLE nblock ( id clé primaire entière AUTOINCREMENT , prevhash VARCHAR NOT NULL , newhash VARCHAR NOT NULL , ntrans INTEGER NOT NULL ,nonce INTEGER NOT NULL ,qrng INTEGER NOT NULL ) ;
```

Nous allons voir quelque chose de similaire :



The screenshot shows a terminal window on an Android device. The status bar at the top indicates battery level (27%), signal strength, and time (12:46 a.m.). The terminal output is as follows:

```
$ sqlite3 minibc.db
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
sqlite> CREATE TABLE nblock (
...>     id integer primary key AUTOINCREMENT
...>     , prevhash VARCHAR NOT NULL
...>     , newhash VARCHAR NOT NULL
...>     , ntrans INTEGER NOT NULL
...>     , nonce INTEGER NOT NULL
...>     , qrng INTEGER NOT NULL);
sqlite> .tables
nblock
sqlite> .quit
$
```

A yellow bracket on the right side of the terminal window groups the entire command for creating the table, with a callout box pointing to it containing the text: "Instruction SQL pour la création de la table nblock."

Ensuite, nous allons créer le hachage initial du système appelé "**Genesis**", qui est basé sur la création d'un nouveau hachage du premier bloc de la chaîne de blocs que nous utiliserons

(**NewBlock**). Ensuite, nous créerons un deuxième hachage que nous appellerons "un" en nous basant sur le hachage "Genesis" car il doit être cohérent avec le premier hachage car le test de validation du hachage sur la chaîne de blocs initiale doit toujours être conforme à : prevhash = newhash.

NewBlock. Pour créer le hachage "Genesis", nous utiliserons les paramètres d'entrée :



Paramètres d'entrée : **données** <Corde>, **previousHash** <Corde>

Paramètre de sortie : Un hachage SHA256.

Dans ce cas, nous utiliserons comme "previousHash" une initiale égale à "0" et dans le cas des données d'entrée, "data" sera le mot "Genensis".

Cela nous donnera un hachage calculé de la combinaison des deux données :

SHA256 (Genèse0) = eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642

La chaîne ci-dessus sera insérée dans la table nblock, cette chaîne doit être cryptée pour cela nous utilisons l'extension (**OpenQbitEncDecData**).



Nous utiliserons l'extension OpenQbitSSHClient pour insérer les données dans la base de données minibc.db, ce serait la déclaration INSERT à la base de données minibc.db de la table nblock.

```

sqlite3 keystore.db "insérer dans le nblock (prevhash, newhash, ntrans, nonce, qrng) les
valeurs ('0', 'eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642',
'1', '0', '0')"; "sqlite3 keystore.db
  
```

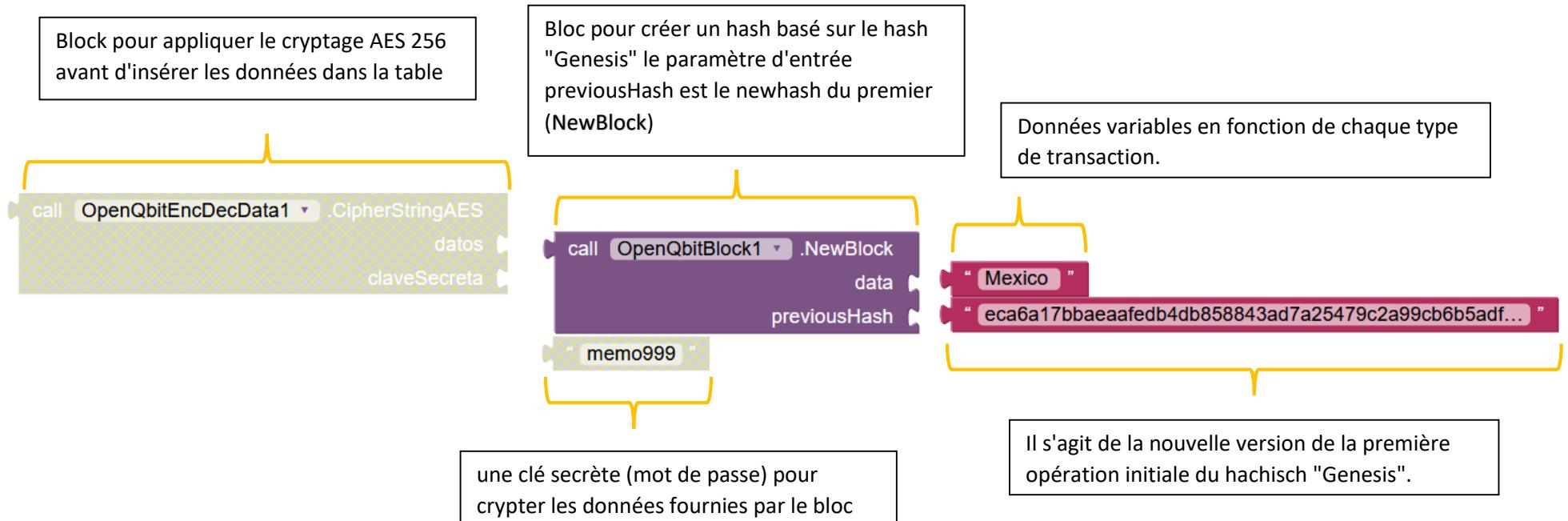
Nous avons créé le hachage "one" avec l'instruction SQL basée sur le hachage "Genensis" :

```
sqlite3 keystore.db "insérer dans nblock (prevhash, newhash, ntrans, nonce, qrng) les  
valeurs ('  
eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642', f6a6b509376  
deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68, '1', '0', '0') ;"
```

Le hash "un" newhash est calculé comme suit :

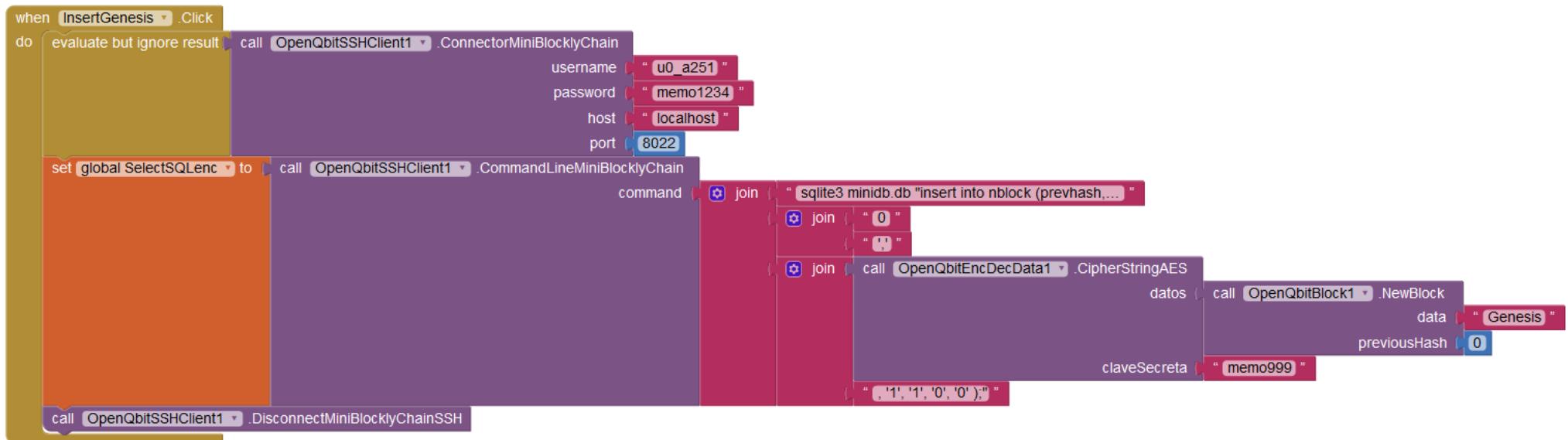
SHA256 (eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642Mexique) =
f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78ab59a68

Deuxième hachage basé sur le premier hachage d'initialisation "Genesis", on doit faire deux INSERTS au début car toute validation initiale de la chaîne de blocs doit respecter l'égalité des champs : prevhash (avant-dernières données) = newhash (dernières données).



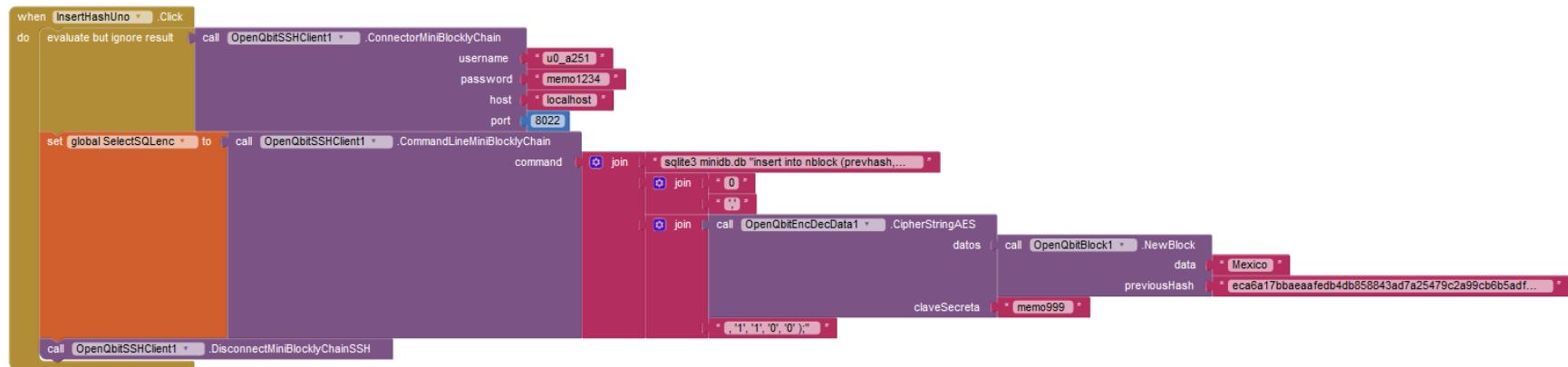
Nous continuons avec la création de l'exécution dans App Inventor des hachis ci-dessus ; hachis "Genesis" et hachis "start".

Nous montrons maintenant comment l'INSERTION du hachage "Genesis" serait intégrée dans la structure initiale de la chaîne de blocs dans le tableau des nblocs du système App Inventor :



Puisque nous avons inséré la première donnée dans le tableau nblock basé sur le hachage "Genesis", nous procédons à la deuxième INSERTION référencée au hachage "one".

Il intégrerait l'INSERTION du hachage "**One**" dans la structure initiale de la chaîne de blocs dans le tableau des nblocs du système App Inventor :



Les deux structures de programmation précédentes (hash "Genesis" et hash "one") devraient être intégrées par le noeud initial du système Mini BlocklyChain, un point important est que la base de données minibc.db avec toute sa structure interne (tables) sera répliquée à travers le réseau Mini BlocklyChain basé sur une architecture "Peer to Peer".

Jusqu'à présent, nous avons achevé la structure de base et fondamentale de la chaîne de stockage des blocs et celle-ci est prête à recevoir de nouveaux blocs provenant de futures transactions provenant ou demandées dans le système.

Nous avons déjà inséré la première donnée de hachage "Genesis" et le hachage "one" dans notre base de données **minibc.db**, maintenant nous allons utiliser les instructions SQL suivantes pour interroger les champs **prevhash** et **newhash** dans la table **nblock**.

Ce point sera fondamental puisque, grâce à la comparaison de ces deux domaines, nous saurons si la chaîne de blocs est cohérente et si elle maintient l'intégrité et la sécurité des transactions. Ce n'est qu'un mécanisme de départ pour revoir la chaîne de blocs car, à l'avenir, nous allons revoir l'utilisation du bloc pour calculer l'arbre de merkle qui sera un autre outil essentiel pour garantir l'intégrité et la sécurité de la chaîne de blocs dans notre système également.

Pour l'instant, nous allons seulement revoir la structure ou les phrases SQL que nous devons utiliser comme nous l'avons fait précédemment avec l'extension (**OpenQbitSSHClient**), avec celles-ci nous pourrons consulter les champs prevhash et newhash. Plus tard, nous pourrons faire la simple comparaison de données égales pour ce contrôle chaque fois que nous ajouterons un nouveau bloc.

Pour prevhash :

```
sqlite3 minibc.db "SÉLECTIONNER LES PRÉVISIONS DE nblock ORDRE PAR id DESC LIMIT 1 ;"
```

Pour le newhash :

```
sqlite3 minibc.db "SELECT newhash FROM nblock ORDER BY id DESC LIMIT 1 ;"
```

Nous allons maintenant nous concentrer sur la manière dont nous allons faire une demande de soumission d'une nouvelle transaction et/ou d'une nouvelle opération à insérer dans la file d'attente des transactions.

L'envoi d'une transaction dans la file d'attente des transactions se fait en deux étapes.

La première condition est que le solde de notre compte dispose de suffisamment d'"actifs" pour couvrir le montant à envoyer. N'oubliez pas que les "actifs" peuvent être non seulement un nombre ou un montant, mais que nous pouvons créer des "actifs" de toute sorte en fonction de chaque système à créer.

Exemples d'actifs :

- ✓ Montant intrinsèque d'un montant "virtuel ou crypto-monnaie" de nouvelle origine.
- ✓ Les données référencées aux données numériques (tous les types de documents)
- ✓ Signature d'authentification de hachage pour les chaînes de caractères ou tous les types de fichiers.
- ✓ Toute transaction ou transfert d'informations numériques ou leur représentation.

Le solde des "actifs" de chaque nœud est obtenu à l'aide du bloc (**GetBalance**).



La deuxième exigence consiste à fournir les paramètres minimums lors de l'envoi d'une transaction, qui sont

- ✓ Adresse de la source. - est l'adresse à partir de laquelle les biens seront expédiés.
- ✓ Adresse de destination. - est l'adresse de l'utilisateur qui recevra les avoirs envoyés.
- ✓ Actif. - Ce sont les données à valeur intrinsèque données dans chaque système qui doivent être envoyées lors de chaque transaction.

Les adresses ci-dessus (origine-destination) correspondent aux clés publiques de chaque utilisateur lors de la création des comptes de chaque utilisateur. N'oubliez pas que lors de la création d'un compte utilisateur, deux types de clés publiques et privées sont créés.

La clé publique est l'adresse qui sera partagée dans tout le système et que tout utilisateur du système peut voir et utiliser pour envoyer ou recevoir des transactions.

La clé privée est l'adresse utilisée localement par chaque nœud et, comme son nom l'indique, elle est à usage privé et permet de créer des signatures numériques pour sécuriser les transactions envoyées. Cette clé ne doit jamais être partagée, elle est à usage local et unique au nœud source.

Afin d'utiliser les paramètres précédents, nous nous appuierons sur deux dépôts où les adresses des "clés privées" qui sont dans la base de données `keystore.db` sont stockées et seront d'usage local de chaque nœud sans partage dans le système et l'autre dépôt où toutes les adresses des "clés publiques" qui sont dans la base de données `publickeys.db` sont stockées sera partagé à travers le protocole "Peer to Peer" dans tous les nœuds du système.

Les bases de données "`keystore.db`" et "`publickeys.db`" ont déjà été créées dans l'annexe "Création de la base de données KeyStore & PublicKeys".

La base de données et le système pour la file d'attente des transactions ont déjà été créés dans la section "Installation et configuration du réseau RESTful Mini Sentinel". Là où nous avons déjà créé une base de données pour les transactions appelée `op.sqlite3`, nous avons deux tableaux : l'un est "trans" qui s'occupe des transactions (**file d'attente des transactions**) et l'autre appelé "sign" où sont stockées les signatures numériques de chaque opération.

Le système SQLite RESTful est le réseau de sauvegarde ; à cette occasion nous l'utiliserons pour la facilité et pour tester le système de sauvegarde, cependant, pour changer et utiliser la même base de données `op.sqlite3` dans un modèle "Peer to Peer" nous n'aurons qu'à utiliser la base de données dans un modèle partagé à nouveau dans le système de synchronisation, ceci nous le verrons plus tard.

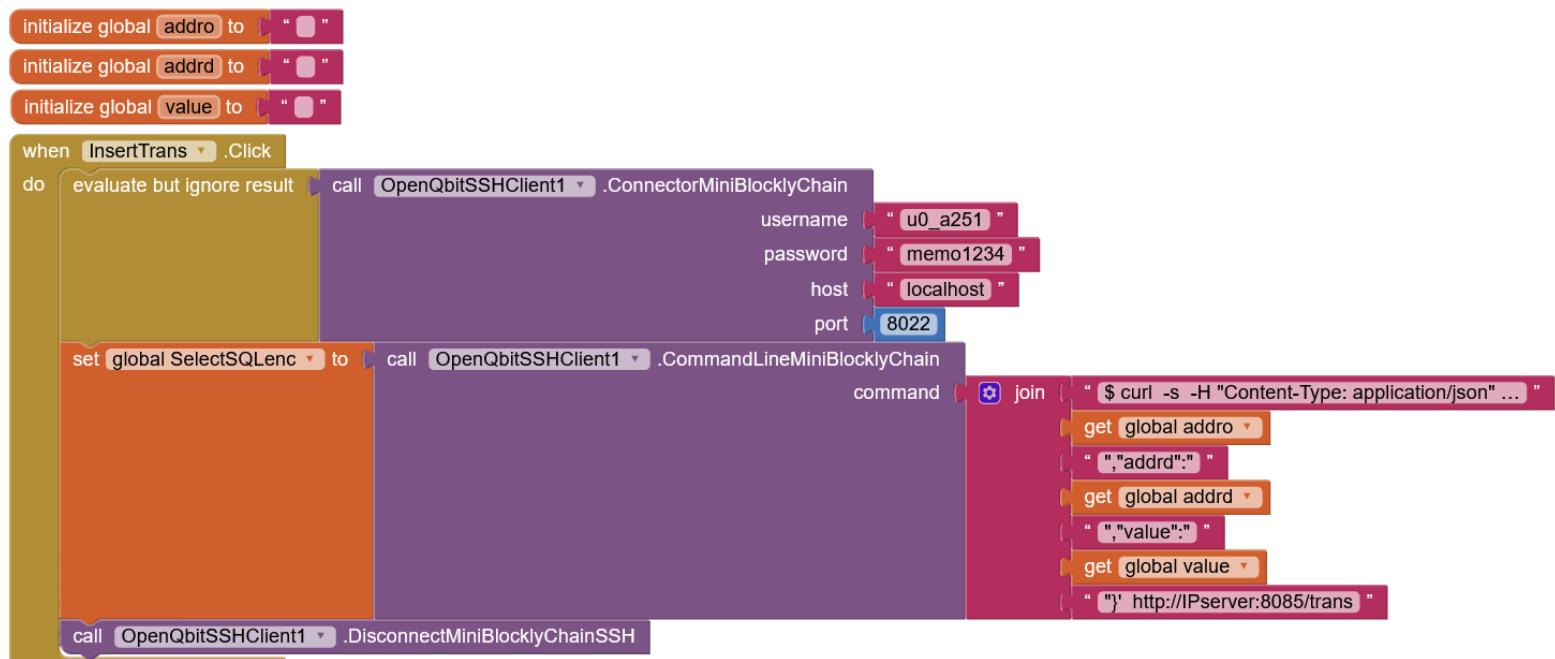
Nous allons commencer à envoyer des opérations à la file d'attente des transactions en utilisant RESTful appliqué à la base de données op.sqlite.

Nous avons créé une nouvelle transaction dans le tableau "trans

```
$ curl -s -H "Content-Type : application/json" -d '{"addr0" :  
"MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9", "addrd" :  
"MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh", "value" : "999"}' http://IPserver:8085/trans
```

```
# Nombre de sorties  
{  
    "id" : 1,  
    "addr0" : "MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",  
    "addrd" : "MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh",  
    "valeur" : "999"  
}
```

Mise en œuvre dans App Inventor :



Les paramètres d'entrée : addro, addrd, value sont des variables qui peuvent être entrées dans l'algorithme et sont extraites de la base de données keystore.db

Voir l'annexe "Création de la base de données KeyStore".

Nous insérons maintenant les signatures numériques de la transaction précédente. Dans le tableau "signe"

```
$ curl -s -H "Content-Type : application/json" -d '{  
  "trans_id":1  
  "signe" : "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",  
  "hash" : "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68"}'  
http://IPserver:8085/sign  
  
# Nombre de sorties  
{  
  "id" : 1,  
  "trans_id" : 1,  
  "signe" : "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",  
  "hash" : "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78ab59a68"  
}
```

NOTE : La signature numérique (signe) doit être obtenue avant l'envoi de la déclaration de commande de curl, elle est générée avec le bloc (**GenerateSignature**).

Les paramètres d'entrée : Trans_id, sign, hash sont des variables qui peuvent être entrées dans l'algorithme et la signature numérique de la transaction sera générée avec le bloc (**GenerateSignature**).

Nous allons maintenant voir la procédure de génération d'une signature numérique à appliquer à chaque transaction qui est effectuée.

Génération de la signature numérique pour la transaction. Lorsque nous utilisons le bloc (**GenerateSignature**), nous aurons comme résultat un fichier de type .sig ; ce fichier sera utilisé pour être enregistré dans le champ "sign" du tableau "sign", cette signature sera utilisée lorsque la file d'attente des transactions sera traitée et c'est un autre paramètre pour donner la sécurité entre l'origine et la destination, ainsi que le montant ou le type de transaction entre eux.

Pour traiter des données binaires comme le fichier .sig, nous devrons le convertir en base64 puis le stocker dans la variable de signe de la table "sign". Pour ce faire, nous utiliserons le bloc (**EncoderFileBase64**).

Comment la valeur du champ de hachage est calculée pour la table "signe" de chaque transaction :

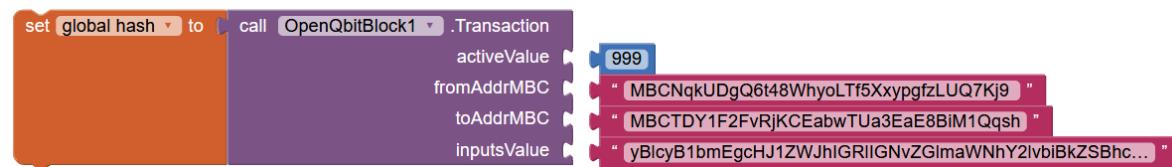
Transaction Hash = SHA256(adresse source + adresse de destination + actif + fichierBase64.sig)

Exemple de hachage de type SHA256 :

SHA256(MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9 + MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh + 999 +).

Sortie : 9d45198faaef624f2e7d1897dd9b3cede6ecc7fbac516ed1756b350fe1d56b4

Pour le calcul du hash, nous devrons nous appuyer sur le Bloc(**Transaction**).



Le paramètre de sortie est le hachage qui sera stocké pour chaque transaction envoyée depuis n'importe quel nœud du système, dans le champ "signe" de la table "signe" de la base op.sqlite

Lors de l'opération précédente, nous nous sommes assurés que seul un hachage unique et non répétable représentera chaque transaction envoyée à la file d'attente et ce hachage représentatif est la totalité des informations transmises.

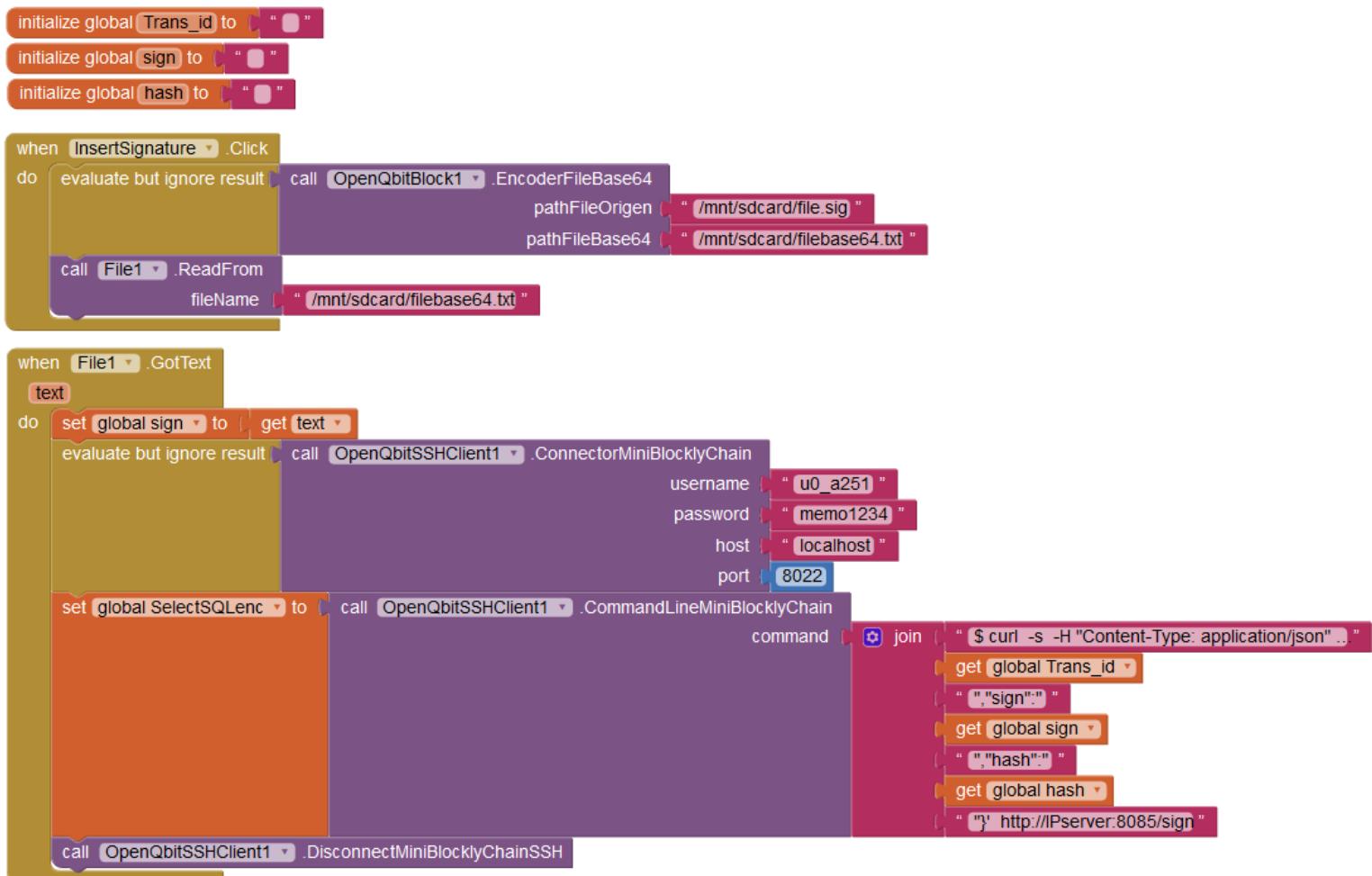
La seule chose qui manque est d'inclure la variable Trans_id, qui peut être un identifiant permettant de différencier le nœud envoyé pour chaque transaction. Dans notre exemple, nous allons mettre la variable Trans_id avec une valeur fixe de "1". Parce que c'est le premier nœud qui est en train d'être configuré dans notre réseau. Cette valeur peut faire varier la syntaxe en fonction de chaque dessin particulier, c'est pourquoi nous avons choisi un identifiant simple pour des raisons de simplicité.

Comme nous avons défini toutes les variables, nous allons créer la structure et la séquence d'exécution des blocs dans App Inventor, pour effectuer l'INSERTION dans le tableau "signe".

NOTE : Il est important de tenir compte du fait que chaque envoi d'une transaction est composé de deux INSERTS dans la base de données op.sqlite3, l'un doit être fait dans la table "trans" et leurs valeurs respectives dans la table "sign".

Lors de la conception de la base de données op.sqlite3, deux tables distinctes ont été créées car, à l'avenir, il est possible de crypter la table "signe" uniquement parce qu'elle contient des informations qui ne doivent pas être envoyées dans le réseau de manière plate, cette option est laissée à l'examen de chaque conception future.

Nous allons créer la structure d'exécution des blocs et des méthodes dans l'App Inventor de l'INSERT dans le tableau "signe".



Jusqu'à présent, nous avons déjà terminé l'insertion dans la table "sign" et dans la table "trans", qui se trouvent dans la base de données **op.sqlite3**, et les données insérées dans ces deux tables seront utilisées pour créer la file d'attente des transactions qui sera envoyée à tous les nœuds pour leur traitement.

Pour l'instant, nous utiliserons le connecteur Java SQLite-Redis, qui permettra de convertir les données de la base de données op.sqlite3 à la base de données Redis. Voir l'annexe "Connecteur de code Java SQLite-Redis".

Après avoir mis en place le connecteur SQLite-Redis, la file d'attente des transactions sera livrée à tous les nœuds du système par le biais du système Redis.

Nous allons commencer à réfléchir à la manière dont nous pouvons recevoir la file d'attente des transactions de manière appropriée pour pouvoir la traiter.

La file d'attente des transactions sera fournie par le service Redis, une base de données en temps réel qui a ses blocs de contrôle respectifs dans l'environnement App Inventor.

Configuration de l'environnement Redis dans le système Blockly d'App Inventor.

Un point important à noter est que les blocs pour le contrôle d'un serveur Redis jusqu'à la rédaction de ce manuel ne sont disponibles que dans le système App Inventor. En cas d'utilisation d'un système Blockly différent de l'App Inventor, vous devrez utiliser le Redis CLI (Command-Line) pour l'exécuter en envoyant des commandes au terminal Termux via l'extension ([OpenQbitSSHClient](#)).

Exemple d'utilisation dans Redis CLI (Command-Line) :

Pour obtenir tous les **labels** ou clés de Redis.

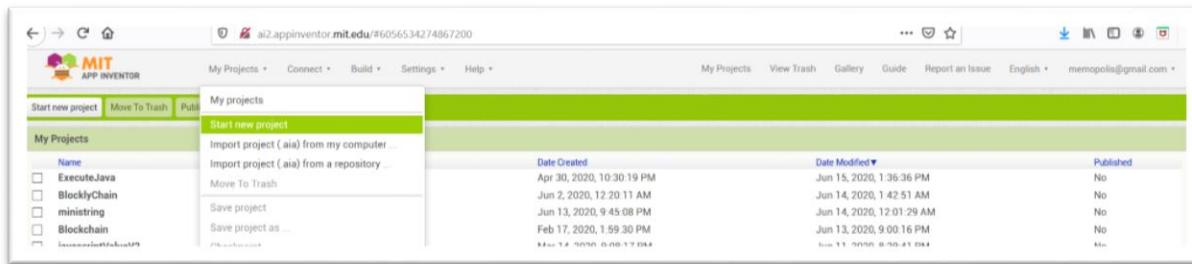
\$ redis-cli KEYS '*' (en anglais)

Pour obtenir la valeur d'une clé.

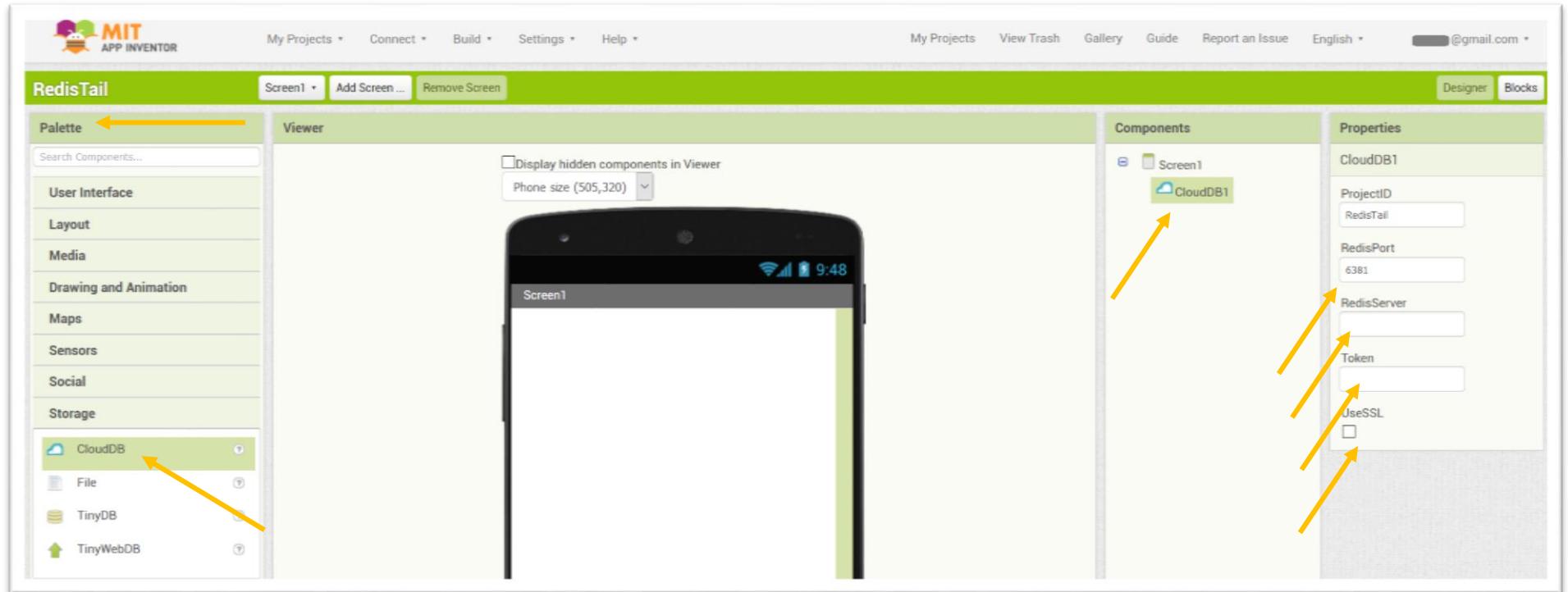
\$ redis-cli GET <key>

Dans notre cas, parce que nous utilisons l'inventeur App, nous allons examiner comment utiliser les blocs pour travailler avec Redis.

A l'intérieur d'App Inventor, nous avons plusieurs blocs à utiliser avec Redis, nous avons commencé à créer un nouveau projet que nous allons réaliser : Mes projets > Démarrer un nouveau projet



Après avoir créé le projet, nous allons en haut à gauche et dans la section "Palette", cliquez sur "Stockage" et faites glisser l'objet "CloudDB" ; cela intégrera toutes les fonctionnalités pour configurer une connexion à un serveur Redis.



La configuration est très simple, il suffit de donner les paramètres suivants pour pouvoir se connecter à notre serveur Redis (Local) qui tourne dans notre noeud (Téléphone portable).

ProjectID. - Il s'agit de l'identifiant qui lancera l'étiquette qui identifie la file d'attente des transactions dans Redis.

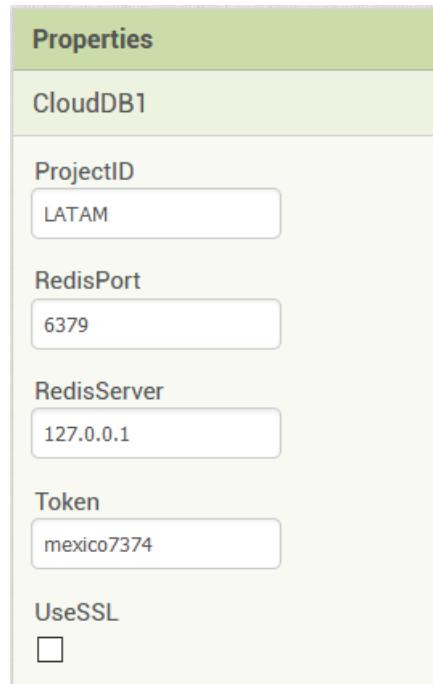
RedisPort. - C'est le port du serveur Redis où nous nous connecterons par défaut est 6379

RedisServer. - Il s'agit du domaine ou de l'IP locale du nœud dans notre cas et celui de tous les nœuds sera 127.0.0.1

Token. - Il s'agit du mot de passe du serveur Redis activé pour la connexion.

Utilisez leSSL. - Cette option nous donne la possibilité d'utiliser des certificats SSL dans notre cas, nous la laissons non activée.

Dans notre modèle et la conception du système, nous aurons les paramètres suivants dans tous les nœuds du réseau.



Il est temps de revoir les blocs qui nous permettent de contrôler et de traiter les données dans un environnement de base de données Redis.

Nous commençons par le bloc méthode (**DataChanged**)



Cette méthode nous donnera deux valeurs à chaque fois qu'il y aura un changement dans le serveur Redis que nous aurons configuré :

tag. - Cette valeur est la balise qui identifie la file d'attente des transactions.

valeur. - Cette valeur contient la liste de toutes les transactions envoyées pour traitement. Cette valeur est une liste de chaînes de caractères de type <Cordes>.

Dans le cas de la "valeur", c'est là que nous commencerons à effectuer notre traitement de l'information comme suit.

Comme vous nous fournissez une chaîne de toutes les transactions de valeur de hachage, nous commencerons par vérifier si cette chaîne est valable et vérifierons si elle n'a pas été modifiée depuis son origine. Pour ce faire, nous utilisons un algorithme très utile pour la vérification de grandes quantités d'informations.

Nous utiliserons l'algorithme de l'arbre de Merkle. L'application de cet algorithme nous donne une sécurité dans l'intégrité des informations que nous recevons (file d'attente des transactions).

Voyons l'exemple suivant d'une file d'attente de transactions dans Redis, nous exécutons le CLI (Command-Line) de Redis depuis un terminal Termux pour vérifier la clé "LATAM:HASH", nous devons avoir déjà exécuté le connecteur SQLite-Redis pour pouvoir consulter cette clé.

```
C:memor>redis-cli
127.0.0.1:6379> obtenir LATAM:HASH
"9d45198faaef624f2e7d1897dd9b3cde6ecca7fbac516ed1756b350fe1d56b4,"
f71c801a5fd25fc303ebc8c616204b4877ffb93006ec6a88bc30acf43ec250f5,"\60f8a3bcac1
ea7d38e86efbc3e3e00480807d23f980391000766e804ed14ecb2 ,
8a6dfe1d38c22e0f9212052efa6136da3edf1fb1b2a3e25224ac3d689124b754]
127.0.0.1:6379>
```

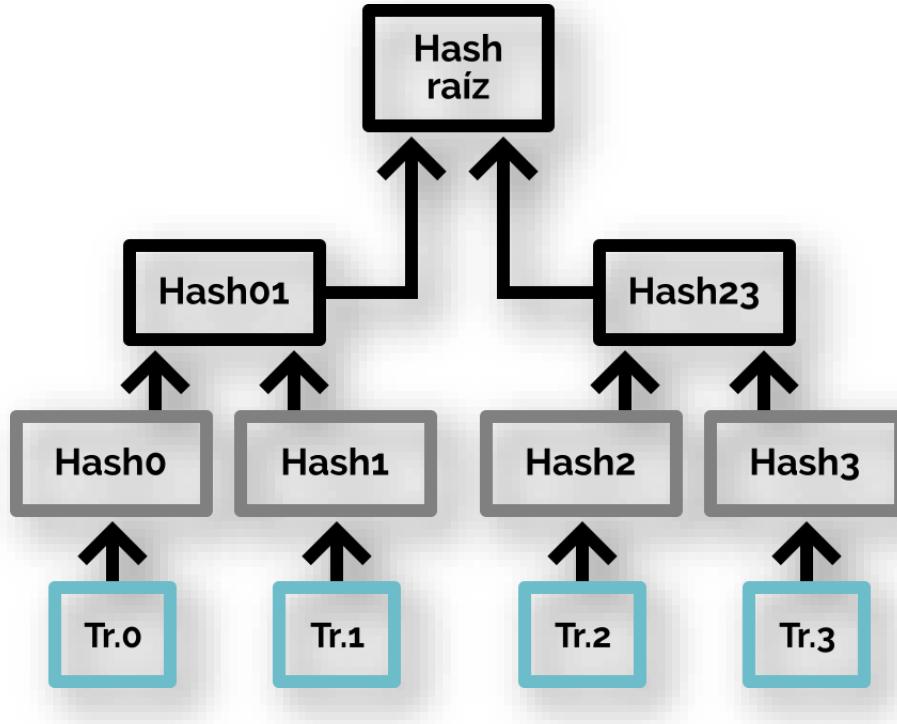
La file d'attente des transactions précédentes ne comporte que trois éléments puisque nous voyons quatre hachages représentant une transaction individuelle qui la composent et ce sont les hachages de chaque transaction qui a été initialement injectée dans la base de données op.sqlite3 dans les tables "trans" et "sign".

Il est maintenant temps de comprendre comment fonctionne l'arbre merkle.

Un arbre de hachage est une structure arborescente de données binaires ou non binaires dans laquelle chaque nœud qui n'est pas une feuille est étiqueté avec le hachage de la concaténation des étiquettes ou des valeurs de ses noeuds enfants. Il s'agit d'une généralisation des listes de hachage et des chaînes de hachage.

Dans notre cas, nous allons faire le calcul suivant pour calculer l'arbre merkle pour quatre éléments ; ceci est fait en calculant le résultat de la prise de paires de données concaténées et obtenir leurs hachages respectifs, les résultats du premier niveau seront appliqués aux résultats du second niveau jusqu'à ce qu'il n'y ait qu'un seul élément final, ceci sera appelé le hachage merkleroot (hachage racine).

Examinons le schéma suivant qui décrit ce processus.



La file d'attente des transactions basées sur le hachage sera retirée via le bloc (`GetMerkleRoot`)



Racine du hachage :
51431822de7c94b90dc06d47b8f6275f315a4976c8479d30c32747fa90325432

Le résultat est ensuite comparé à la clé LATAM:merkleroot du système Redis local et les deux clés doivent correspondre pour vérifier l'intégrité des données.

Un autre point de sécurité fourni par l'arbre de merkle est de confirmer qu'une transaction spécifique est incluse dans la file d'attente des transactions depuis son origine et qu'elle n'a pas été introduite par un moyen de communication externe ou interne de manière frauduleuse.

Dans le cas où la chaîne est composée d'éléments impairs dans sa sommation, le dernier élément se répète pour avoir un arrangement pour et commencer l'exécution de l'algorithme.

Un autre point non moins important est le temps nécessaire pour valider que chaque transaction correspond à son origine-destination et que le bien est celui envoyé par l'adresse d'origine.

C'est là que se trouve le bloc (VerifySignature).



```
call [OpenQbitBlock1] .VerifySignature
```

Avant d'exécuter le bloc précédent, vous devez d'abord télécharger le fichier (file.sig) en format binaire à partir de la table "signe" :

```
sqlite3 op.sqlite3 "select sign from sign where=id_addr;"
```

id_addr : C'est l'identifiant de l'adresse source de la table "trans".

La demande de recherche précédente nous donnera des données au format Base64 de la signature numérique de la transaction en cours, ceci pour la convertir dans son format original (binaire) nous aurons besoin du Block(**DecoderFileBase64**).

Comme nous disposons de notre fichier binaire (file.sig), nous devrons charger dans le système la clé publique de l'origine et la clé publique du destinataire également en format binaire, avoir les quatre données dans leurs formats respectifs sera le moment de lancer la vérification de la signature numérique dans le système.

- ✓ Adresse du domicile à clé publique
- ✓ Adresse de la clé publique du destinataire
- ✓ Actif envoyé.
- ✓ Signature numérique (fichier .sig)

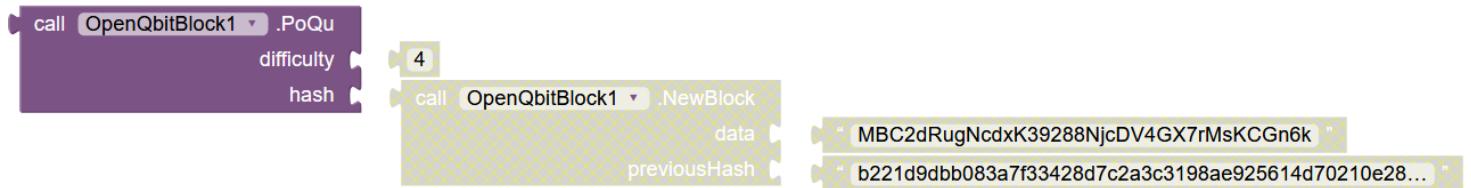
Les clés publiques en format binaire peuvent être téléchargées dans le format approprié (binaire) à partir de la base de données partagée publickeys.db

Ce processus doit être exécuté pour chaque opération individuelle dans la file d'attente des transactions.

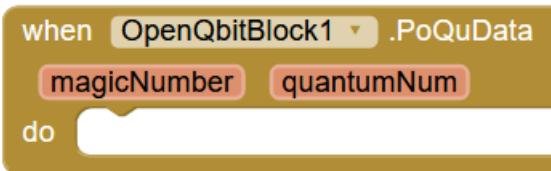
Enfin, nous examinerons comment le système choisit un nœud de manière consensuelle pour pouvoir être choisi afin d'ajouter le bloc suivant à la chaîne de blocs et être celui qui traite la file d'attente des transactions.

Cette façon de choisir le nœud gagnant pour traiter la file d'attente des transactions est basée sur notre algorithme développé pour un système de Mini BlocklyChain.

Comment nous avons mis en œuvre le consensus du PoQu "Preuve de Quantum". Ce processus de consensus est basé sur la génération de nombres aléatoires quantiques et est appliqué en utilisant le bloc (**PoQu**).



D'abord, nous obtenons deux paramètres que le bloc (**PoQu**) nous donne livrés dans la méthode (**PoQuData**) ; les paramètres sont le **magicNumber** et le **quantumNum**.



Le **magicNumber** est le nombre "nonce", est un entier qui est obtenu en faisant un PoW interne avec difficulté pas plus grand que 5, ce nombre est chargé de donner une première exigence au noeud avec le **magicNumber** le noeud pourra faire une exigence pour obtenir un nombre du système de génération de nombres aléatoires quantiques qui est dans la gamme de 0 à 1, ce nombre donnera une probabilité établie aléatoirement pour le noeud en exécution qui sera stockée dans le tableau appelé "vote".

La table de "vote" stockera le **quantumNum** calculé par chaque noeud, ce stockage sera fait avec un certain nombre de noeuds jusqu'à un certain moment établi dans chaque conception de système ; il est recommandé d'avoir des entrées $((N/2) + 1)$ où "N" est la quantité de noeuds disponibles dans le système et elle peut être établie ou contrôlée par une action dans l'outil de gestion des tâches "cron" de chaque noeud.

Un point important est qu'à ce stade, vous devriez déjà avoir établi une synchronisation de l'heure locale de chaque noeud par le biais du système automatique du téléphone portable en cas d'utilisation du réseau "**Peer to Peer**" dans la transmission de la file d'attente des transactions.

La configuration de l'agent cron dans les nœuds. Voir la section "Synchronisation du temps dans les nœuds du système (téléphone portable) en minutes et secondes".

Dans cet exemple, comme nous utilisons le réseau de communication de secours, nous n'avons pas besoin de la synchronisation des minutes et des secondes pour les nœuds car notre exemple occupe un schéma "**client-serveur**" ; ce type de communication n'est que dans le processus de transmission de la file d'attente des transactions. Tous les autres processus entre les nœuds se font par une communication "**Peer to Peer**".

Nous allons maintenant examiner la structure, la conception et la création du tableau "vote" qui sera situé dans la base de données appelée "quorum.db" que nous allons également créer dans cet exemple.

\$ sqlite3

SQLite version 3.32.2 2020-06-20 15:25:24

Entrez ".help" pour les conseils d'utilisation.

Connecté à une base de données en mémoire transitoire.

Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.

sqlite> .open quorum.db

sqlite> CREATE TABLE vote (id entier clé primaire AUTOINCREMENT PAS NUL, node_imei VARCHAR PAS NUL, quantumNum INTEGER PAS NUL, nonce INTEGER PAS NUL);

sqlite> .quit

La création de la même table de **vote** est présentée ci-dessous, mais c'est en introduisant l'instruction SQL sous une forme segmentée :

\$ sqlite3

SQLite version 3.32.2 2020-06-20 15:25:24

Entrez ".help" pour les conseils d'utilisation.

Connecté à une base de données en mémoire transitoire.

Utilisez ".open FILENAME" pour rouvrir sur une base de données persistante.

sqlite> .open quorum.db

sqlite> CREER UN TABLEAU de vote (

...> id clé primaire entière AUTOINCREMENT

...> node_imei VARCHAR NOT NULL,

...> INTEGRE quantique NON NUL,

...> nonce INTEGER NOT NULL

...>);

sqlite> .tables

voter

sqlite> .quit

Nous verrons quelque chose de similaire dans la création de la base "quorum.db" et du vote par correspondance.

```

$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open quorum.db
sqlite> CREATE TABLE vote (
...>     id INTEGER primary key AUTOINCREMENT,
...>     node_imei VARCHAR NOT NULL,
...>     quantumNum INTEGER NOT NULL,
...>     time INTEGER NOT NULL
...> );
sqlite> .tables
vote
sqlite> .quit
$ 

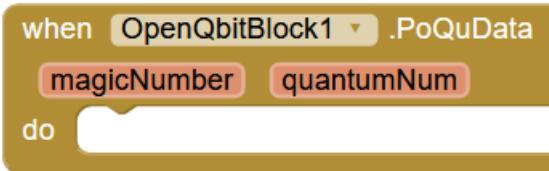
```

Voyons maintenant comment nous obtenons les valeurs du tableau "vote".

node_imei. - Nous **obtenons** cette valeur en utilisant le bloc (**GetDeviceID**).

call OpenQbitBlock1 .GetDeviceID

quantumNum - Cette valeur est l'un des résultats de la méthode (**PoQuData**) qui est obtenue en utilisant le bloc (**PoQu**).



nonce. - Cette valeur est obtenue en ayant déjà exécuté le bloc (**PoQu**) de manière intégrale et est identique au magicNumber de la méthode (**PoQuData**).

Après avoir complété les INSERTS dans le tableau "vote", il est nécessaire de faire une copie de la base de données.

Après un certain temps et en fonction de la conception de chaque système, le service "cron" sera exécuté sur chaque nœud du réseau et aura le prochain SELECT à traiter dans la table "vote" :

CHOISIR node_imei DEPUIS le vote OÙ magicNumber= (CHOISIR max(magicNumber) DEPUIS le vote) ;

Le précédent SELECT renvoie le résultat IMEI avec une probabilité plus élevée, maintenant chaque nœud qui exécute SELECT fera une comparaison de son IMEI avec l'IMEI du résultat et seul le nœud qui correspond créera un fichier avec le numéro de probabilité le plus élevé qui sera répliqué dans le réseau "Peer to Peer" par un fichier au format IMEI.mbc qui contiendra l'IMEI du nœud gagnant.

Le nœud gagnant pourra commencer à traiter la file d'attente des transactions. Basé sur tous les blocs ci-dessus.

Deux points importants sont que, en fonction de la création de chaque système, trois processus devront être revus et personnalisés par chaque concepteur.

1) Lorsque le nœud gagnant commence à traiter la file d'attente des transactions, il faut mettre en œuvre une méthode ou un processus permettant de vérifier que le nœud gagnant est en ligne et que la communication avec le réseau n'a pas été perdue.

2.- Lorsque le traitement de la file d'attente des transactions commence, le nœud gagnant doit lancer deux drapeaux de contrôle indiquant le début du traitement et un autre pour confirmer que le traitement de la file d'attente des transactions est terminé. Ces deux drapeaux doivent être partagés dans le réseau à tous les nœuds, ce qui permettra de localiser

les échecs de connexion ou les défaillances de traitement ou un temps de traitement trop long.

En cas de défaillance de la communication ou d'un autre événement où le nœud gagnant n'a pas pu traiter la file d'attente des transactions, le nœud suivant dans la plage de probabilité immédiate doit être choisi.

Le point précédent peut être contrôlé avec un service qui vérifie que le nœud gagnant est en ligne et peut utiliser le service "cron", le script doit être développé pour chaque cas conçu, cependant, un exemple générique de script shell est montré ci-dessous afin qu'il puisse être modifié selon les besoins de chaque système de Mini Blocklychain.

```
#!/bin/bash
dir="/data/data/com.termux/files/home/Sync/imei" ;
si [ ! "$(ls $directory)" ] ]
puis
sqlite3 quorum.db "MISE À JOUR du vote SET magicNumber=0 WHERE magicNumber=
(SÉLECTIONNER max(magicNumber) DU vote) ;"
sinon
MAX_NUM=$(sqlite3 quorum.db "SÉLECTIONNER max(magicNumber) DU vote ;")
IMEI_quorum=$(sqlite3 quorum.db "SELECT node_id FROM vote WHERE=MAX_NUM")
IMEI_local=$(cat device_imei) // Utiliser le bloc (GetDevice)
si [ IMEI_quorum -eq IMEI_local ]
puis
touchez $MAX_NUM > IMEI.mbc
fi
fi
sortie
```

31. Annexe "Intégration avec les environnements Ethereum et Bitcoin".

Nous allons maintenant voir comment nous pouvons intégrer les deux systèmes de chaînes de blocs les plus connus au monde spécialisés dans les cryptomonnaies tels qu'Ethereum et Bitcoin.

Commençons par l'installation du logiciel qui nous aidera à effectuer toutes les transactions possibles dans l'environnement Ethereum.

Qu'est-ce qu'Ethereum ?

Ethereum est une plateforme open source, décentralisée contrairement aux autres chaînes de blocs, Ethereum peut faire beaucoup plus. Il est programmable, ce qui signifie que les développeurs peuvent l'utiliser pour créer de nouveaux types d'applications.

Ces applications décentralisées (ou "dapps") bénéficient des avantages du montage cryptographique et de la technologie des chaînes de blocs. Ils sont fiables et prévisibles, ce qui signifie qu'une fois "chargés" dans l'Ethereum, ils fonctionneront toujours dans les délais prévus. Ils peuvent contrôler les biens numériques pour créer de nouveaux types d'applications financières. Ils peuvent être décentralisés, ce qui signifie qu'aucune entité ou personne ne les contrôle.

En ce moment, des milliers de développeurs dans le monde entier créent des applications sur Ethereum et inventent de nouveaux types d'applications, dont beaucoup peuvent être utilisées aujourd'hui :

- Portefeuilles en crypto-monnaie qui vous permettent d'effectuer des paiements instantanés et peu coûteux avec l'ETH ou d'autres actifs
- Les applications financières qui vous permettent d'emprunter, de prêter ou d'investir vos actifs numériques
- Des marchés décentralisés, permettant d'échanger des biens numériques, ou même d'échanger des "prédictions" sur des événements du monde réel.
- Des jeux où vous avez des atouts dans le jeu et pouvez même gagner de l'argent réel.
- Elle a des contrats intelligents qui sont des programmes avec des accords à exécuter lorsque les prémisses avec lesquelles elle a été élaborée ou créée sont remplies.

Les contrats intelligents présentent des similitudes avec les **DApps** (applications décentralisées), mais les séparent aussi de certaines différences importantes.

Comme les contrats intelligents, un DApp est une interface qui relie un utilisateur à un service d'un fournisseur par le biais d'un réseau de pairs décentralisé. Mais, alors que les contrats intelligents nécessitent la création d'un nombre fixe de participants, les DApps n'ont pas de limite au nombre d'utilisateurs. En outre, elles ne se limitent pas aux applications financières telles que les contrats intelligents : un DApp peut servir à toutes les fins auxquelles vous pensez.

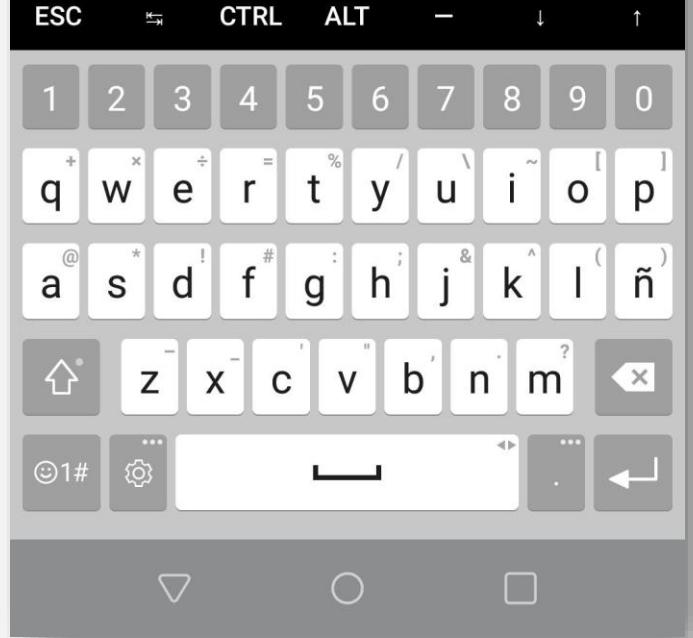
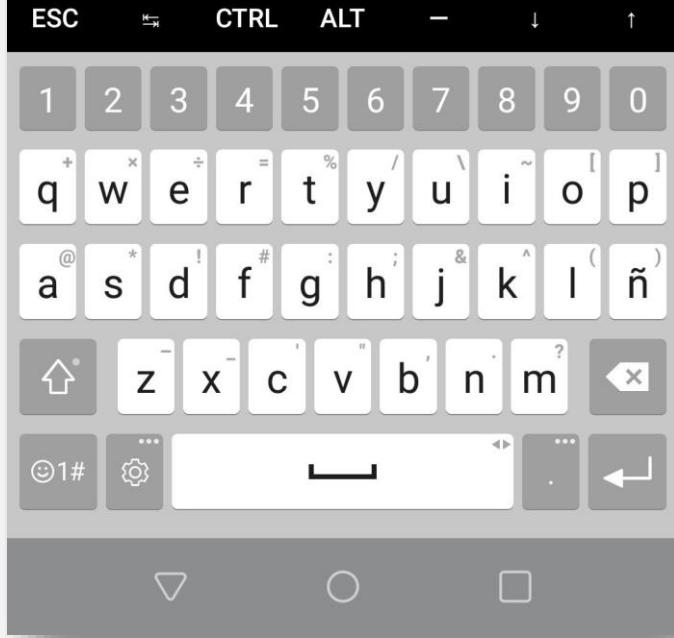
Dans notre cas, nous utiliserons la bibliothèque appelée "Web3j" qui est développée en Java, et qui permet d'interagir avec la chaîne de blocs Ethereum de manière simple et intuitive.

Exécutez la commande suivante pour installer "Web3j" :

```
$ curl -L get.web3j.io | sh
```

```
#####
##### 100.0%
Installing Web3j...
Removing downloaded archive...

Web3j was successfully installed.
To use web3j in your current shell run:
source $HOME/.web3j/source.sh
When you open a new shell this will be performed
automatically.
To see what web3j's CLI can do you can check the
documentation bellow.
https://docs.web3j.io/command\_line\_tools/
```



Ensuite, nous devons créer la variable d'environnement **\$JAVA_HOME** avec le chemin où se trouve l'exécutable "java" car la bibliothèque va rechercher cette variable pour pouvoir être exécutée avec succès.

```
$ JAVA_HOME= /data/data/com.termux/files/usr/bin
```

Une fois la variable créée, il faut aller dans le répertoire où la bibliothèque "Web3j" a été installée en exécutant la commande suivante, un point important est que le répertoire est caché après la commande "cd" on met un point "..." et ensuite le répertoire sans espaces, comme suit :

```
$ cd .web3j
```

Une fois à l'intérieur, nous testons si la bibliothèque fonctionne correctement avec la commande suivante :

```
Version ./web3j
```

Il en résulte quelque chose de très similaire à :

```
$ ls
source.sh web3j web3j-4.5.16
$ ./web3j version

Version: 4.5.16
Build timestamp: 2020-03-06 14:13:49.943 UTC
$
```

Plus tard, nous créerons un portefeuille qui sera utilisé dans l'environnement de la chaîne d'approvisionnement d'Ethereum de la manière suivante :

\$./web3j portefeuille créer

La commande précédente nous donne l'adresse de l'ethereum :

4598fe2fd6afe2508f58343c7d42f2ab492edf34

```
$ ./web3j wallet create

Please enter a wallet file password:
Please re-enter the password:
Please enter a destination directory location [/data/data/com.termux/files/home/.ethereum/testnet/keystore]:
Wallet file UTC--2020-06-27T06-12-23.819752000Z-4598fe2fd6afe2508f58343c7d42f2ab492edf34.json successfully created in: /data/data/com.termux/files/home/.ethereum/testnet/keystore
$
```

Il en résulte un fichier au format JSON contenant l'adresse et les données cryptées qui seront utilisées ultérieurement pour générer la clé privée du compte qui vient d'être créé.

Ce fichier de type JSON sera créé dans un répertoire par défaut appelé **keystore**.

A partir de là, nous pouvons déjà effectuer des opérations en utilisant et/ou en exécutant la commande Web3j.

Nous pouvons le faire en utilisant l'extension (**ConnectorSSHClient**).

Pour savoir comment utiliser les différents paramètres et opérations de la bibliothèque "Web3j", nous pouvons nous aider en consultant la documentation sur son site officiel.

<https://docs.web3j.io/>

Pour l'environnement Bitcoin, nous avons deux options : utiliser le bloc () qui génère le compte Bitcoin et ses clés publiques et privées respectives. Nous pouvons utiliser ces clés pour nous intégrer dans l'environnement Bitcoin en installant la bibliothèque java "Bitcoij".

\$ npm install bitcoinj



```
$ npm install bitcoinj
+ bitcoinj@0.0.0
added 1 package from 1 contributor and audited 2
09 packages in 20.528s
$
```

Pour utiliser cette bibliothèque, nous nous appuierons sur son site officiel.

<https://bitcoinj.github.io/>

Une deuxième option d'intégration dans l'environnement de la chaîne de blocs Bitcoin consiste à installer le paquet Bitcoind pour Termux comme indiqué ci-dessous.

\$ apt install bitcoin



```
$ pkg install bitcoin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bitcoin
0 upgraded, 1 newly installed, 0 to remove and 1
9 not upgraded.
Need to get 3601 kB of archives.
After this operation, 15.0 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm bitcoin arm 0.20.0 [3601
kB]
Fetched 3601 kB in 2s (1253 kB/s)
Selecting previously unselected package bitcoin.
(Reading database ... 19110 files and directo
ries currently installed.)
Preparing to unpack .../bitcoin_0.20.0_arm.deb .
..
Unpacking bitcoin (0.20.0) ...
Setting up bitcoin (0.20.0) ...
$
```

Maintenant, lorsque nous lançons l'agent bitcoind sur le terminal Termux, il crée automatiquement une adresse dans le répertoire :

/data/data/com.termux/files/hme/.bitcoin

Dans le cas de Bitcoin, nous nous appuierons sur la documentation suivante de l'agent "Bitcoind".

Dans ce cas, nous pouvons également utiliser l'extension (**ConnectorSSHClient**) pour exécuter l'agent "bitcoind" en fonction de chaque besoin.

Description des paramètres à utiliser avec bitcoind.

NOM

bitcoind - lancement du démon central Bitcoin

SYNOPSIS

bitcoind [*options*] Démarrer le *démon central de Bitcoin*

DESCRIPTION

Démarrer le démon central de Bitcoin

OPTIONS

- ?

Imprimez ce message d'aide et quittez

-alertnotify=<cmd>

Exécutez la commande lorsqu'une alerte pertinente est reçue ou que nous voyons une bifurcation très longue (%s en cmd est remplacé par le message)

-assumevalid=<hex>

Si ce bloc est dans la chaîne de caractères, supposez que lui et ses ancêtres sont valides et sautez potentiellement votre vérification en écriture (0 pour vérifier tout, par défaut : 0000000000000000f1c54590ee18d15ec70e68c8cd4cfbadb1b4f11697eee, testnet : 000000000000037a8cd3e06cd5edbfe9dd1dbcc5dacab279376ef7fcf2b4c75)

-blocknotify=<cmd>

Exécutez la commande lorsque le meilleur bloc change (%s en cmd est remplacé par le hachage du bloc)

-blockreconstructionextratxn=<n>

Transactions supplémentaires à garder en mémoire pour les reconstructions de blocs compacts (par défaut : 100)

-blocksdir=<dir>

Précisez le répertoire des blocs (par défaut : <datadir>/blocks)

-seulement en bloc

Si vous refusez les transactions des partenaires du réseau. Les transactions de portefeuille ou de CPR ne sont pas affectées. (par défaut : 0)

-conf=<archive>

Précisez le fichier de configuration. Les chemins relatifs seront préfixés par l'emplacement du datadir. (par défaut : bitcoin.conf)

-daemon

Il fonctionne en arrière-plan comme un démon et accepte les commandes

-datadir=<dir>

Préciser le répertoire de données

-dbcache=<n>

Taille maximale du cache de la base de données <n> MiB (4 à 16384, par défaut : 450). En outre, la mémoire mempool inutilisée est partagée pour ce cache (voir **-maxmempool**).

-debuglogfile=<file>

Indiquez l'emplacement du fichier journal de débogage. Les chemins relatifs seront préfixés par l'emplacement d'un datadir spécifique au réseau. (**-nodebuglogfile** pour désactiver ; par défaut : debug.log)

-includeconf=<file>

Spécifier un fichier de configuration supplémentaire, relatif au chemin **-datadir** (ne peut être utilisé qu'à partir du fichier de configuration, pas à partir de la ligne de commande)

-loadblock=<file>

Importer des blocs du fichier externe blk000 ???.dat au démarrage

-maxmempool=<n>

Maintenir la mémoire des transactions en dessous de <n> mégaoctets (par défaut : 300)

-maxorphantx=<n>

Garder en mémoire les transactions déconnectables (par défaut : 100)

-mempoolexpiry=<n>

Ne pas conserver les transactions dans mempool plus de <n> heures (par défaut : 336)

-par=<n>

Définit le nombre de fils de vérification dans le tiret (-6 à 16, 0 = auto, <0 = laisse tous les fils libres, par défaut : 0)

-persistmempool

Si vous sauvegardez le mempool lorsque vous vous arrêtez et le chargez lorsque vous redémarrez (par défaut : 1)

-pid=<file>

Précisez le fichier PID. Les chemins relatifs seront préfixés par l'emplacement d'un datadir spécifique au réseau. (par défaut : bitcoind.pid)

-punza=<n>

Réduire les besoins de stockage en permettant l'élagage (enlèvement) des vieux blocs. Cela permet d'appeler la chaîne d'élagage RPC pour retirer des blocs spécifiques, et permet l'élagage automatique des anciens blocs si une taille cible dans le MIB est fournie. Ce mode est incompatible avec **-txindex** et **-rescan**. Avertissement : Pour inverser ce paramètre, il est nécessaire de retélécharger toute la chaîne de blocs (par défaut : 0 = désactiver l'élagage en bloc, 1 = autoriser l'élagage manuel via RPC, >=550 = élaguer automatiquement les fichiers de blocs pour rester en dessous de la taille cible spécifiée dans la MIB)

-reindex

Reconstruit l'état de la chaîne et l'index de bloc des fichiers blk*.dat sur le disque

-reindex-chainstate

Reconstruire l'état de la chaîne à partir des blocs actuellement indexés. En mode d'élagage ou si les blocs sur le disque peuvent être corrompus, utilisez plutôt le **ré-index** complet.

-sign

Créer de nouveaux fichiers avec les autorisations par défaut du système, au lieu de l'umask 077 (efficace uniquement si la fonctionnalité de portefeuille est désactivée)

-txindex

Maintenir un index complet des transactions, utilisé par l'appel rpc getrawtransaction (par défaut : 0)

-version

Version Print & Go

Options de connexion :

-addnode=<ip>

Ajoutez un nœud de connexion et essayez de garder la connexion ouverte (voir l'aide de la commande RPC de l'"addendum" pour plus d'informations). Cette option peut être spécifiée plusieurs fois pour ajouter plusieurs nœuds.

-banscore=<n>

Seuil pour déconnecter les collègues qui se comportent mal (par défaut : 100)

-En attendant...

Nombre de secondes pour éviter que des collègues mal intentionnés ne se reconnectent (par défaut : 86400)

-bind=<addr>

Attachez-vous à l'adresse donnée et écoutez-la toujours. Utiliser la notation [host]:port pour IPv6

-connection=<ip>

Connectez-vous uniquement au nœud spécifié ; **-noconnect désactive les connexions automatiques** (les règles pour cette paire sont les mêmes que pour **-addnode**). Cette option peut être spécifiée plusieurs fois pour se connecter à plusieurs nœuds.

-découvrez

Découvrez vos propres adresses IP (par défaut : 1 lors de l'écoute et non **-externalip** ou **-proxy**)

-dns

Autoriser les recherches DNS pour **-addnode**, **-seednode** et **-connect** (par défaut : 1)

-dnsseed

Recherche de paires d'adresses via la recherche DNS, si le nombre d'adresses est faible (par défaut : 1 sauf si **-connection** est utilisé)

-enablebip61

Envoyer des messages de rejet par BIP61 (par défaut : 1)

-externalip=<ip>

Précisez votre propre adresse publique

-forcednsseed

Toujours interroger les adresses des collègues via la recherche DNS (par défaut : 0)

-écoutez

Accepter les connexions de l'extérieur (par défaut : 1 si pas de **-proxy** ou **-connexion**)

-listenonionionion

Créer automatiquement le service caché Tor (par défaut : 1)

-maxconnections=<n>

Maintenir au maximum <n> les connexions avec les collègues (par défaut : 125)

-maxreceivebuffer=<n>

Tampon de réception maximum par connexion, <n>*1000 octets (par défaut : 5000)

-maxsendbuffer=<n>

Tampon d'envoi maximum par connexion, <n>*1000 octets (par défaut : 1000)

-la fixation d'une durée maximale

Le maximum autorisé pour l'ajustement de la compensation de temps moyen des paires. La perspective de l'heure locale peut être influencée par les paires avant ou arrière de ce montant. (par défaut : 4200 secondes)

-maxuploadtarget=<n>

Essayez de maintenir le trafic sortant sous la cible donnée (en MiB pendant 24h), 0 = pas de limite (par défaut : 0)

-onion=<<ip:port>

Utiliser un proxy SOCKS5 séparé pour atteindre les pairs à travers les services cachés de Tor, en réglant **-noonion** sur disable (par défaut : **-proxy**)

-onlynet=<net>

N'établissez des connexions sortantes qu'à travers le réseau <net> (ipv4, ipv6 ou oignon). Les connexions entrantes ne sont pas concernées par cette option. Cette option peut être spécifiée plusieurs fois pour permettre l'utilisation de plusieurs réseaux.

-filtres de paires

Supporte le blocage du filtrage et la transaction avec les filtres à fleurs (par défaut : 1)

-permitbaremultisig

Relais non P2SH multisig (par défaut : 1)

-port=<port>

Écoutez les connexions dans "port" (par défaut : 8333, testnet : 18333, regtest : 18444)

-proxy=<ip:port>

Connectez-vous via le proxy SOCKS5, réglez **-noproxy** sur off (par défaut : off)

-proxyrandomize

Randomiser les références pour chaque connexion proxy Cela permet d'isoler le flux Tor (par défaut : 1)

-seednode=<ip>

Connectez-vous à un nœud pour récupérer les adresses de la paire, et déconnectez-vous. Cette option peut être spécifiée plusieurs fois pour se connecter à plusieurs nœuds.

-timeout=<n>

Précisez le délai de connexion en millisecondes (minimum : 1, par défaut : 5000)

-torcontrol=<ip>:<port>

Port de contrôle Tor à utiliser si l'écoute en oignon est activée (par défaut : 127.0.0.1:9051)

-password=<pass>

Mot de passe du port de contrôle de Tor (par défaut : vide)

-upnp

Utilisez UPnP pour attribuer le port d'écoute (par défaut : 0)

-whitebind=<addr>

Lien vers une certaine adresse et les partenaires de la liste blanche qui s'y connectent. Utiliser la notation [host]:port pour IPv6

-whitelist=<adresse IP ou réseau>

Les paires en liste blanche sont connectées à partir de l'adresse IP donnée (par exemple 1.2.3.4) ou du réseau annoté du CIDR (par exemple 1.2.3.0/24). Il peut être spécifié plusieurs fois. Les paires figurant sur la liste blanche ne peuvent pas être interdites par le ministère de la sécurité intérieure

Options de portefeuille :

-type d'adresse

Quel type d'adresses utiliser ("legacy", "p2sh-segwit", ou "bech32", par défaut : "p2sh-segwit")

-éviter les coûts partiels

Regroupez les sorties par direction, en sélectionnant toutes ou aucune, plutôt que de sélectionner par sortie. La confidentialité est renforcée car une adresse n'est utilisée qu'une seule fois (sauf si quelqu'un l'envoie après qu'elle a été dépensée), mais peut

entraîner des taux légèrement plus élevés car la sélection des devises peut être sous-optimale en raison de la limitation ajoutée (par défaut : 0)

-changement de type

Quel taux de change utiliser ("legacy", "p2sh-segwit", ou "bech32"). La valeur par défaut est la même que **-addresstype**, sauf que **-addresstype=p2sh-segwit** utilise la sortie native segwit lors de l'envoi vers une adresse native segwit)

-du portefeuille

Ne pas charger le portefeuille et désactiver les appels RPC depuis le portefeuille

-discardfee=<amt>

Le taux du tarif (en BTC/kB) qui indique sa tolérance à rejeter la modification en l'ajoutant au tarif (par défaut : 0,0001). Note : Une sortie est rejetée si elle est de la poussière à ce taux, mais nous rejetons toujours jusqu'au taux de retransmission de la poussière et un taux de rejet supérieur est limité par l'estimation du taux pour l'objectif à long terme

-fallbackfee=<amt>

Un taux de redevance (en BTC/kB) à utiliser lorsque l'estimation de la redevance comporte des données insuffisantes (par défaut : 0,0002)

-keypool=<n>

Définir la taille de la réserve de clés à <n> (par défaut : 1000)

-mintxfee=<amt>

Les taux (en BTC/kB) inférieurs sont considérés comme un taux zéro pour la création de la transaction (par défaut : 0,00001)

-paytxfee=<amt>

Taux (en BTC/kB) à ajouter aux transactions que vous envoyez (par défaut : 0,00)

-rescan

Rescanner la chaîne de blocs pour rechercher les transactions de portefeuille manquantes au début

-salvagewallet

Tentative de récupération des clés privées d'un portefeuille corrompu dans le coffre

-en gaspillant l'échange d'informations

Dépenser la monnaie non confirmée lors de l'envoi des transactions (par défaut : 1)

-txconfirmtarget=<n>

Si aucun frais de paiement n'est fixé, inclure un montant suffisant pour que les transactions commencent à être confirmées en moyenne dans n blocs (par défaut : 6)

-maintien du portefeuille

Mettre à jour le portfolio au format le plus récent au début

-portefeuille=<chemin>

Précisez le chemin de la base de données des portefeuilles. Vous pouvez le spécifier plusieurs fois pour charger plusieurs portefeuilles. Le chemin est interprété par rapport à <walletdir> s'il n'est pas absolu, et sera créé s'il n'existe pas (comme un répertoire contenant un fichier wallet.dat et des fichiers journaux). Pour la rétrocompatibilité, il acceptera également les noms des fichiers de données existants dans <walletdir>).

-walletbroadcast

Effectuer les opérations de diffusion du portefeuille (par défaut : 1)

-walletdir=<dir>

Précisez le répertoire pour enregistrer les portefeuilles (par défaut : <datadir>/portfolios s'il existe, sinon <datadir>)

-walletnotify=<cmd>

Exécuter la commande lorsqu'une transaction de portefeuille change (%s en cmd est remplacé par TxID)

-walletrbf

Envoyer les transactions avec l'option d'inclure l'ensemble du RBF activé (RPC uniquement, par défaut : 0)

-zapwallettxes=<mode>

Supprimez toutes les transactions du portefeuille et ne récupérez que les parties de la chaîne de blocage par **-rescan au** début (1 = conserver les métadonnées tx, par exemple les informations relatives aux demandes de paiement, 2 = supprimer les métadonnées tx)

Options de notification ZeroMQ :

-zmqpubhashblock=<adresse>

Activer le bloc de publication dans "adresse"

-zmqpubhashblockhwm=<n>

Définir le bloc de publication des messages sortants avec un filigrane élevé (par défaut : 1000)

-zmqpubhashtx=<adresse>

Permet de publier la transaction de hachage dans l'adresse

-zmqpubhashtxhwm=<n>

Définir publier le message de sortie de la transaction de hachage en filigrane élevé (par défaut : 1000)

-zmqpubrawblock=<adresse>

Activer le bloc de publication brut dans "adresse"

-zmqpubrawblockhwm=<n>

Définir le filigrane du message sortant en bloc brut (par défaut : 1000)

-zmqpubrawtx=<adresse>

Activer la publication de la transaction brute dans "l'adresse"

-zmqpubrawtxhwm=<n>

Définir le filigrane élevé du message de sortie des transactions brutes publiées (par défaut : 1000)

Options de débogage et de test :

-debug=<catégorie>

Informations de débogage en sortie (par défaut : **-nodebug**, la mention "catégorie" est facultative). Si <catégorie> n'est pas fourni, ou si <catégorie> = 1, toutes les informations de débogage sont sorties. <catégorie> peut être : net, tor, mempool, http, bench, zmq, db, rpc, estimatefee, addrman, selectcoins, reindex, cmpctblock, rand, prune, proxy, mempoolrej, libevent, coindb, qt, leveldb.

-debugexclude=<catégorie>

Exclure les informations de débogage d'une catégorie. Peut être utilisé en conjonction avec **-debug=1** pour générer des enregistrements de débogage pour toutes les catégories sauf une ou plusieurs catégories spécifiques.

-help-debug

Imprimer le message d'aide avec les options de débogage et de sortie

-logips

Inclure les adresses IP dans la sortie de débogage (par défaut : 0)

-logtimestamps

Préparer la sortie de débogage avec l'horodatage (par défaut : 1)

-maxtxfee=<amt>

Frais totaux maximums (en CTB) pour une utilisation sur une seule transaction de portefeuille ou sur une transaction brute ; s'il est fixé à un niveau trop bas, il peut interrompre des transactions importantes (par défaut : 0,10)

-impression pour la console

Envoyer les informations de trace/débogage à la console (par défaut : 1 quand il n'y a pas de **-daemon**. Pour désactiver l'enregistrement dans le fichier, définissez **-nodebuglogfile**)

-shrinkdebugfile

Réduire le fichier debug.log au démarrage du client (par défaut : 1 si pas **de -debug**)

-uacomment=<cmt>

Ajouter un commentaire à la chaîne de l'agent utilisateur

Options de sélection de la chaîne :

-testnet

Utilisez la chaîne de test...

Options de retransmission des noeuds :

-bytespersigop

Équivalents en octets par sigop dans les transactions de radiodiffusion et d'exploitation minière (par défaut : 20)

-datacarrier

Transactions de relais et de support de données de mine (par défaut : 1)

-datacarrierize

Taille maximale des données dans les transactions des supports de données que nous retransmettons et extrayons (par défaut : 83)

-mempool-replacement

Permettre le remplacement des transactions dans le pool de mémoire (par défaut : 1)

-minrelaytxfee=<amt>

Les frais (en BTC/kB) inférieurs à ce montant sont considérés comme une redevance nulle pour la retransmission, l'extraction et la création de transactions (par défaut : 0,00001)

-whitelistforcerelay

Forcer la retransmission des transactions des partenaires de la liste blanche, même si les transactions étaient déjà dans mempool ou violent la politique locale de retransmission (par défaut : 0)

-whitelistrelay

Accepter les transactions transmises reçues de paires de la liste blanche même si aucune transaction n'est transmise (par défaut : 1)

Bloquez les options de création :

-blockmaxweight=<n>

Fixer le poids maximum du bloc BIP141 (par défaut : 3996000)

-blockmintxfee=<amt>

Fixez le taux de commission le plus bas (en BTC/kB) pour les transactions qui sont incluses dans la création de blocs. (par défaut : 0.00001)

Options du serveur RPC :

-Restaurant

Accepter les demandes REST publiques (par défaut : 0)

-rpcallowip=<ip>

Autoriser les connexions JSON-RPC à partir de la source spécifiée. Ils sont valables pour <ip> un seul IP (par exemple, 1.2.3.4), un réseau/masque de réseau (par exemple, 1.2.3.4/255.255.255.0), ou un réseau/CIDR (par exemple, 1.2.3.4/24). Cette option peut être spécifiée plusieurs fois

-rpcauth=<userpw>

Nom d'utilisateur et mot de passe HMAC-SHA-256 pour les connexions JSON-RPC. Le champ "userpw" a le format suivant : "username" : "SALT" : \$ "HASH". Un script python canonique est inclus dans share/rpcauth. Le client se connecte alors normalement en utilisant la paire d'arguments rpcuser=<USERNAME>/rpcpassword=<PASSWORD>. Cette option peut être spécifiée plusieurs fois

-rpcbind=<addr>[:port]

Lien vers une certaine adresse pour écouter les connexions JSON-RPC. N'exposez pas le serveur RPC à des réseaux peu fiables tels que l'Internet public ! Cette option est ignorée à moins que **-rpcallowip** ne soit également adopté. Le port est optionnel et a priorité sur le **port de plaisance**. Utilisez la notation [host]:port pour IPv6. Cette option peut être spécifiée plusieurs fois (par défaut : 127.0.0.1 et ::1 c'est-à-dire localhost)

-rpccookiefile=<loc>

Emplacement du cookie d'autorisation. Les chemins relatifs seront préfixés par l'emplacement d'un datadir spécifique au réseau. (par défaut : dir de données)

-rpcpassword=<pw>

Mot de passe pour les connexions JSON-RPC

-rpcport=<port>

Écoutez les connexions JSON-RPC dans le "port" (par défaut : 8332, testnet : 18332, regtest : 18443)

-rpcserialversion

Définit la sérialisation de la transaction brute ou l'hexagone de bloc renvoyé en mode non verbal, et non segwit(0) ou segwit(1) (par défaut : 1)

-rpcthreads=<n>

Définir le nombre de threads pour traiter les appels RPC (par défaut : 4)

-rpcuser=<user>

Nom d'utilisateur pour les connexions JSON-RPC

-serveur

Accepter les commandes de la ligne de commande et les JSON-RPC

32.Licences et utilisation des logiciels.

Android

<https://source.android.com/setup/start/licenses>

Termux

<https://github.com/termux/termux-app/blob/master/LICENSE.md>

Nœud

<https://raw.githubusercontent.com/nodejs/node/master/LICENSE>

SQLite

<https://www.sqlite.org/copyright.html>

Git

<https://git-scm.com/about/free-and-open-source>

de la place au repos

<https://github.com/olsonpm/sqlite-to-rest/blob/dev/license.txt>

Redis DB

<https://redis.io/topics/license>

WorldTimeAPI NTP

<http://worldtimeapi.org/pages/faqs#commercial-apps>

Réseau Tor

<https://github.com/torproject/tor/blob/master/LICENSE>

Réseau de synergie

<https://forum.syncthing.net/t/syncthing-is-now-mplv2-licensed/2133>

OpenSSH

<https://www.openssh.com/features.html>

Mastic SSH

<https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

MIT App Inventor 2 Companion et App Inventor Blockly

<https://appinventor.mit.edu/about/termsofservice>

SQLite Expert Personal - gratuit

<http://www.sqliteexpert.com/download.html>

Fourmi apache

<https://ant.apache.org/license.html>

WGET

<https://www.gnu.org/software/wget/>

OpenJDK

<https://openjdk.java.net/legal/>

Extensions externes :

JSONTOOLs

<https://thunkableblocks.blogspot.com/2017/07/jsontools-extension.html>

L'octroi de licences pour les versions open source et commerciales du système Mini BlocklyChain est disponible sur le site officiel <http://www.openqbit.com>

Mini BlocklyChain, MiniBlockly, BlocklyCode, MiniBlockMiniChain, QBlockly son marcas registradas por OpenQbit.

La Mini BlocklyChain est dans le domaine public.

Tout le code et la documentation de Mini BlocklyChain ont été mis à disposition du domaine public par les auteurs. Tous les auteurs de codes et les représentants des entreprises pour lesquelles ils travaillent ont signé des déclarations sous serment consacrant leurs contributions au domaine public et les originaux de ces déclarations sont conservés dans un coffre-fort dans les bureaux principaux d'OpenQbit Mexico. Tout le monde est libre de publier, d'utiliser ou de distribuer les extensions originales de la Mini BlocklyChain (OpenQbit), soit sous forme de code source, soit sous forme de binaires compilés, à toutes fins, commerciales ou non, et par tous moyens.

Le paragraphe précédent s'applique au code et à la documentation livrables dans la Mini BlocklyChain, ces parties de la bibliothèque de la Mini BlocklyChain qui regroupent et livrent effectivement avec une application plus importante. Certains scripts utilisés dans le cadre du processus de compilation (par exemple, les scripts de "configuration" générés par autoconf) peuvent être inclus dans d'autres licences open source. Cependant, aucun de ces scripts de compilation n'entre dans la bibliothèque finale de Mini BlocklyChain, donc les licences associées à ces scripts ne devraient pas être un facteur dans l'évaluation de vos droits de copie et d'utilisation de la bibliothèque Mini BlocklyChain.

Tout le code livrable dans Mini BlocklyChain a été écrit à partir de zéro. Aucun code n'a été repris d'autres projets ou de l'internet ouvert. Chaque ligne de code peut être retracée jusqu'à son auteur d'origine, et tous ces auteurs ont des dédicaces du domaine public dans leur dossier. Par conséquent, la base de code de la Mini BlocklyChain est propre et n'est pas contaminée par du code sous licence d'autres projets à source ouverte, et non par des contributions ouvertes

Mini BlocklyChain est open source, ce qui signifie que vous pouvez faire autant de copies que vous le souhaitez et faire ce que vous voulez avec ces copies, sans limitation. Mais Mini BlocklyChain n'est pas un logiciel libre. Afin de maintenir la Mini BlocklyChain dans le domaine public et de garantir que le code n'est pas contaminé par des contenus propriétaires ou sous licence, le projet n'accepte pas les patchs de personnes inconnues. Tout le code de Mini BlocklyChain est original, car il a été écrit spécifiquement pour être utilisé par Mini BlocklyChain. Aucun code n'a été copié à partir de sources inconnues sur Internet.

La Mini BlocklyChain est du domaine public et ne nécessite pas de licence. Malgré cela, certaines organisations veulent une preuve légale de leur droit à utiliser la Mini BlocklyChain. Les circonstances dans lesquelles cela se produit sont notamment les suivantes :

- Votre entreprise veut être indemnisée pour des plaintes pour violation de droits d'auteur.
- Vous utilisez la Mini BlocklyChain dans une juridiction qui ne reconnaît pas le domaine public.
- Vous utilisez la Mini BlocklyChain dans une juridiction qui ne reconnaît pas le droit d'un auteur de placer son œuvre dans le domaine public.
- Vous voulez avoir un document juridique tangible comme preuve que vous avez le droit légal d'utiliser et de distribuer la Mini BlocklyChain.
- Votre service juridique vous dit que vous devez acheter une licence.

Si l'une des circonstances ci-dessus s'applique à vous, OpenQbit, la société qui emploie tous les développeurs de Mini BlocklyChain, vous vendra une Mini BlocklyChain Title Guarantee. Une garantie de titre est un document juridique qui déclare que les auteurs revendiqués de la Mini BlocklyChain sont les vrais auteurs, et que les auteurs ont le droit légal de dédier la Mini BlocklyChain au domaine public, et qu'OpenQbit se défendra vigoureusement contre les revendications de licence. Tous les produits de la vente des garanties de titre de Mini BlocklyChain sont utilisés pour financer l'amélioration et le soutien continus de Mini BlocklyChain.

Code de contribution

Pour que Mini BlocklyChain reste totalement gratuit et libre de droits, le projet n'accepte pas les patchs. Si vous voulez faire une suggestion de changement et inclure un patch comme preuve de concept, ce serait bien. Cependant, ne soyez pas offensé si nous réécrivons votre patch à partir de zéro. Le type de licence non commerciale ou open source qui l'utilise dans cette modalité et certaines autres similaires sans achat de support, qu'il s'agisse d'une utilisation individuelle ou collective, quelle que soit la taille de l'entreprise, sera régi par les principes juridiques suivants.

Exclusion de garantie. Sauf si la loi applicable l'exige ou si cela est convenu par écrit, le Concédant fournit l'œuvre (et chaque contributeur fournit ses Contributions) "TEL QUEL", **SANS GARANTIES OU CONDITIONS DE QUELQUE NATURE QUE CE SOIT**, expresses ou implicites, y compris, sans limitation, toute garantie ou **condition de TITRE**, d'**ABSENCE DE CONTREFAÇON**, de **VALEUR MARCHANDE** OU d'**ADAPTATION À UN USAGE PARTICULIER**. Vous êtes seul responsable de déterminer l'utilisation ou la redistribution correcte de l'œuvre et d'assumer les risques associés à l'exercice des droits que vous confère la présente licence.

Toute perte financière ou autre subie par l'utilisation de ce logiciel sera supportée par la partie concernée. Tous les litiges juridiques que les parties soumettront à la juridiction des tribunaux de la ville de Mexico, au Mexique.

Pour le support commercial, l'utilisation et la licence, un accord ou un contrat doit être établi entre OpenQbit ou sa société et la partie intéressée.

Les conditions de distribution et de commercialisation peuvent être modifiées sans préavis. Veuillez consulter le site officiel www.openqbit.com pour voir les modifications apportées aux clauses de soutien et de licence non commerciales et commerciales.

Toute personne, utilisateur, entité privée ou publique de toute nature juridique ou de toute partie du monde qui utilise simplement le logiciel accepte sans conditions les clauses établies dans ce document et celles qui peuvent être modifiées à tout moment dans le portail de www.openqbit.co sans avis préalable et peuvent être appliquées à la discréction d'OpenQbit dans un usage non commercial ou commercial.

Toutes les questions et informations sur Mini BlocklyChain doivent être adressées à la communauté App Inventor ou aux différentes communautés du système Blockly telles qu'elles sont : AppBuilder, Trunkable, etc. et/ou au mail opensource@openqbit.com pour la demande de questions peut prendre la réponse de 3 à 5 jours ouvrables.

Soutien à l'utilisation commerciale.

support@openqbit.com

Vente à des fins commerciales.

sales@openqbit.com

Informations juridiques et questions ou préoccupations concernant les licences

legal@openqbit.com

