



Installatie, configuratie & administratie.

Gebruikershandleiding

versie 1.0 Beta

Juli 2020.

MiniBlocklyChain is een gedeponiert handelsmerk van OpenQbit Inc., onder een gratis gebruik- en commerciële licentie. Gebruiksvoorwaarden op: www.OpenQbit.com

Inhoud

1. Inleiding.....	3
2. Wat is een publiek of privaat netwerk in de blokketenregeling?	4
3. Wat is Blockly-programmering?.....	4
4. Wat is Termux?.....	5
5. Wat is Mini BlocklyChain?	5
6. Procesarchitectuur in Mini BlocklyChain	10
7. Functiediagram van de BlocklyChain (Mini BlocklyChain).	13
8. Wat is Mini BlocklyCode?.....	14
9. Mini BlocklyChain-communicatienetwerkinstallatie	15
10. Synchronisatie in systeemknooppunten (mobiele telefoon) minuten en seconden.....	35
11. Opslagconfiguratie binnen Termux.....	43
12. "Tor" netwerkinstallatie en "Syncthing" installatie.....	44
13. Installatie van "Redis" database en SSH (Secure Shell) server.....	45
14. SSH-serverconfiguratie op mobiele telefoon (smartphone).	46
15. "Tor" netwerkconfiguratie met SSH (Secure Shell) service.	53
16. Peer to Peer systeemconfiguratie met handmatige synchronisatie.....	56
17. Ambientes Blockly (App Inventor, AppyBuilder y Thunkable).	67
18. Wat is het bewijs van Quantum (PQu)?.....	68
19. Definitie en gebruik van blokken in Mini BlocklyChain.....	71
20. Gebruik van blokken voor SQLite database (MiniSQLite versie).....	97
21. Definitie en gebruik van veiligheidsblokken.	100
22. Instellen van veiligheidsparameters in Mini BlocklyChain.....	111
23. Bijlage "Oprichting van KeyStore & PublicKeys-databases".	115
24. Bijlage "RESTful SQLite GET/POST-commando's".	129
25. Bijlage "Java Code SQLite-Redis Connector".....	135
26. Bijlage "Mini BlocklyChain voor ontwikkelaars".	138
27. Bijlage "BlocklyCode Slimme Contracten".	150
28. Bijlage "OpenQbit Quantum Computing".	157
29. Bijlage "Uitgebreide blokken voor SQLite database".....	162
30. Bijlage "Voorbeeld van het creëren van een Mini BlocklyChain systeem".....	162
31. Bijlage "Integratie met Ethereum & Bitcoin-omgevingen".....	187
32. Licentie en gebruik van software.	206

1. Inleiding.

De blokketten wordt over het algemeen geassocieerd met Bitcoin en andere crypto valuta, maar deze zijn slechts het topje van de ijsberg, omdat het niet alleen wordt gebruikt voor digitaal geld, maar kan worden gebruikt voor alle informatie die een waarde kan hebben voor gebruikers en/of bedrijven. Deze technologie, die haar oorsprong vindt in 1991, toen Stuart Haber en W. Scott Stornetta het eerste werk aan een keten van cryptografisch beveiligde blokken beschreef, werd pas in 2008 opgemerkt, toen het populair werd met de komst van de bitcoin. Maar op dit moment wordt het gebruik ervan in andere commerciële toepassingen gevraagd en de verwachting is dat het op middellange termijn zal groeien in verschillende markten, zoals financiële instellingen of het internet van de dingen aan het ivd, naast andere sectoren.

De blokketen, beter bekend onder de term blokketen, is een enkel, overeengekomen record dat verdeeld is over verschillende knooppunten (elektronische apparaten zoals pc's, smartphones, tablets, etc.) in een netwerk. In het geval van crypto-valuta's kunnen we het zien als het boekje waarin elk van de transacties wordt geregistreerd.

De werking ervan kan complex zijn om te begrijpen als we ingaan op de interne details van de uitvoering ervan, maar het basisidee is eenvoudig te volgen.

Het wordt opgeslagen in elk blok:

1.- een aantal geldige records of transacties,

2.- informatie over dat blok,

3.- het verband met het vorige blok en het volgende blok door de hash van elk blok –*een unieke code die zou zijn als de vingerafdruk van het blok*.

Daarom heeft **elk blok een specifieke en onbeweeglijke plaats binnen de keten**, aangezien elk blok informatie bevat uit de hasj van het vorige blok. De hele keten wordt opgeslagen op elk netwerkknoppen dat de blokketen vormt, dus **een exacte kopie van de keten wordt op alle deelnemers aan het netwerk opgeslagen**.

Als er nieuwe records worden aangemaakt, worden deze eerst gecontroleerd en gevalideerd door de netwerkknoppen en vervolgens toegevoegd aan een nieuw blok dat aan de keten is gekoppeld.

Stel nu dat dit netwerk van apparaten die onderling communiceren, heeft de mogelijkheid om te communiceren zonder tussenkomst van een persoon, dat wil zeggen het grote voordeel van de blokketen is dat het kan autonome beslissingen te nemen, die voordelen in de reactietijd van de dienst aan gebruikers, beschikbaar 24 uur per dag, minimaliseert de

kosten in het bedrijfsleven en in de eerste plaats heeft een niveau van veiligheid al getest, voor deze en andere redenen is zo populair geworden in gebruik in de verschillende publieke en private sector.

2. Wat is een publiek of privaat netwerk in de blokketenregeling?

Openbaar netwerk. - Het is een netwerk van computers of mobiele apparaten die met elkaar communiceren en anoniem blijven, het is niet bekend wie er in dit netwerk op een formele manier met elkaar communiceert in hun transacties, in dit type netwerk kan elke persoon of elk bedrijf communiceren en op elk moment verbinding maken omdat u geen rechten nodig heeft om verbinding te maken, een voorbeeld is de blokketen van Bitcoin, iedereen kan binnenkomen om te kopen of te verkopen. Normaal gesproken zijn dit soort netwerken gericht of gericht op de aankoop en verkoop van digitale monetaire activa of het synoniem "cryptomonies", voorbeelden: DogCoin, Ethereum, LiteCoin, BitCoin, Waves, enz.

Privé-netwerk. - Het is een netwerk van computers of mobiele apparaten die met elkaar communiceren. In tegenstelling tot openbare netwerken hebben particuliere netwerken echter voorafgaande toestemming nodig van een of andere entiteit (bedrijf of persoon) om verbinding te kunnen maken en deel uit te maken van dit type netwerk. Normaal gesproken worden particuliere blokketennetwerken in bedrijven of ondernemingen gebruikt om transacties of operaties uit te voeren met verschillende soorten informatie die een tastbare waarde kunnen hebben in de vorm van documenten, processen, autorisaties en/of bedrijfsbeslissingen die worden toegepast en gecontroleerd door blokketen, voorbeelden: financiële sector, verzekeringssector, overheid, enz.

3. Wat is Blockly-programmering?

Blockly is een **visuele programmeertaal** die bestaat uit een eenvoudige set commando's die we kunnen combineren alsof het de stukjes van een puzzel zijn. Het is een zeer nuttig hulpmiddel voor degenen die willen **leren programmeren** op een intuïtieve en eenvoudige manier of voor degenen die al weten hoe ze moeten programmeren en het potentieel van dit soort programmering willen zien.

Blockly is een vorm van programmeren waarbij je geen enkele achtergrond in welke computertaal dan ook nodig hebt, dit is omdat het gewoon het aansluiten van grafische blokken is alsof we lego of een puzzel spelen, je moet gewoon wat logica hebben en dat is het!

Iedereen kan programma's voor mobiele telefoons (smartphones) maken zonder te knoeien met die programmeertalen die moeilijk te begrijpen zijn, gewoon blokken op een grafische manier in elkaar zetten op een eenvoudige, gemakkelijke en snelle manier om te maken.

4. Wat is Termux?

Termux is een Android terminal emulator en een Linux-omgevingstoepassing die direct werkt zonder de noodzaak van routing of configuratie. Er wordt automatisch een minimaal basissysteem geïnstalleerd.

We zullen Termux gebruiken voor zijn stabiliteit en eenvoudige installatie en beheer, maar u kunt gebruik maken van een geïnstalleerde omgeving van Ubuntu Linux voor Android.

In deze Linux omgeving heb je de "kern" van de communicatieprocessen van de MiniBlocklyChain.

5. Wat is Mini BlocklyChain?

Mini BlocklyChain is een volledig functionele blockchain is een technologie die is ontwikkeld voor mobiele telefoons met OS (Operating System) **Android** die de knooppunten zullen zijn voor het verzenden en ontvangen van transacties. We hebben de eerste blokketentechnologie gecreëerd die op een "modulaire" manier is gestructureerd door middel van Blockly-programmering, waarbij iedereen met minimale kennis en zonder te weten hoe te programmeren programma's voor mobiele telefoons kan maken en ontwikkelen en zijn eigen blokketen kan creëren, hetzij in de openbare, hetzij in de particuliere netwerkmodus. Als u uw eigen digitale valuta wilt creëren, kunt u dit doen of een oplossing vinden om deze in een bedrijf te gebruiken, de mogelijkheden zijn gebaseerd op de behoeften van elk echt geval en de bedrijfslogica die de gebruiker aanneemt of creëert volgens zijn eisen.

Mini BlocklyChain is de eerste modulaire blokketen voor mobiele telefoons waarbij in korte tijd een transactiesysteem kan worden geïmplementeerd.

Voordat we de definitie en het gebruik van "module" blokken invoeren, moeten we de basisconcepten van de Mini BlocklyChain componenten hebben om te weten wanneer, hoe en waar ze moeten worden toegepast volgens het echte geval dat we willen implementeren.

De volgende concepten leggen de nadruk op het type gebruiker waar ze zich op richten, dus het is niet belangrijk voor mensen die geen programmeervaardigheden hebben dat de concepten voor de gebruikers van de ontwikkeling volledig worden begrepen.

Basisbegrippen:

Knooppunt. - Elk mobiel apparaat (telefoon, tablet, etc.) met een Android-besturingssysteem dat deel uitmaakt van het openbare of privé-netwerk van Mini BlocklyChain wordt genoemd als een knooppunt van dit netwerk.

Hash. - Het is een digitale handtekening die wordt geassocieerd met een reeks gegevens, een reeks karakters, documenten of een soort digitale informatie, deze digitale handtekening is uniek en niet te herhalen, wat helpt om informatie te verzenden of te ontvangen, zodat de oorspronkelijke inhoud van de verzonden informatie niet kan worden gewijzigd.

Actief. - Elk type digitale gegevens of informatie dat kan worden gewogen met een zekere materiële of immateriële waarde voor mensen of bedrijven (documenten, digitale munten, muziek, video, beelden, digitale autorisaties, enz.)

Transactie. - Uitwisseling van informatie tussen knooppunten die een of ander goed in beslag nemen.

UXTO Transactie - Het is een type transactie dat één of meerdere transacties van de knooppunten verpakt en alle niet-bestede transacties van elk knooppunt (UXTO: *Unspent Transaction Outputs*) kunnen worden uitgegeven als input in een nieuwe transactie. Deze transacties worden gegroepeerd in een wachtrij die elke keer moet worden verwerkt en die kan worden geregeld volgens de vereisten van elke informatiestroom.

Transactie (invoer). - Het is een soort transactie die gebaseerd is op UXTO-transacties. Wanneer een UXTO-transactie in de wachtrij komt, wordt deze verwerkt door een **invoertransactie** die de transactie classificeert als een storting of een uitgave en zo een saldo van het eindresultaat voor elke gebruiker kan hebben.

Bronadres. - Dit is het adres van de persoon die de transactie genereert of verzendt om te worden verwerkt in de SQLite Master-wachtrij. Het is een alfanumeriek getal van 64 tekens dat door het systeem wordt aangemaakt en geverifieerd.

Bestemmingsadres. - Dit is het adres van de persoon die de transactie ontvangt en deze aan zijn saldo toevoegt of het verzonden goed bewaart. Het is een alfanumeriek getal van 64 tekens dat door het systeem wordt aangemaakt en gecontroleerd.

SQLite Master. - API REST-database die de transacties ontvangt en een wachtrij aanmaakt om te worden verwerkt elke bepaalde variabele of vaste tijd, afhankelijk van de bedrijfsbehoefte.

DataBase transactie. - Zijn de transacties die worden weergegeven in de SQLite database die Mini BlocklyChain transactie-informatie reserveert of opslaat.

AES (Advanced Encryption Standard) - Het is een beveiligingsproces dat de gegevens die zijn opgeslagen in de SQLite database van Mini BlocklyChain versleutelt.

Balans. - Na het maken van een transactieproces in de Mini BlocklyChain wordt een balans van de operatie verkregen als materiële waarde (cryptomoney) of immateriële waarde (bedrijfsprocessen tussen mensen of bedrijven).

Bedrijfsproces. - **Het** is elk proces dat een soort informatiestroom met zich meebrengt om een resultaat te verkrijgen naar een eindgebruiker, de eindgebruiker kan een persoon of bedrijf zijn uit verschillende sectoren in de private of publieke sector.

Openbare en particuliere sleutel. - Het is een soort informatie-encryptie waarbij twee alfanumerieke sleutels die aan elkaar gekoppeld zijn, worden gegenereerd en gebruikt om gevoelige informatie via openbare of particuliere netwerken te versturen. De privé-sleutel wordt gebruikt om informatie te versleutelen en bij het verzenden via het netwerk kan deze op het eerste gezicht niet worden gewijzigd of leesbaar zijn, de publieke sleutel is degene die wordt gedeeld en wordt gezien door een persoon of bedrijf en deze zal helpen om de verzonden informatie te verifiëren, maar ook om deze alleen te kunnen lezen door de geldige geadresseerde.

Digitale handtekening. - Het is een handtekening die kan worden aangemaakt met de privé-sleutel, met deze handtekening zorgt de ontvanger ervoor dat de informatie niet is gewijzigd en bevestigt hij dat de informatie geldig is voor de ontvanger.

Digitaal adres (Mini BlocklyChain-adres). - Het is een adres dat bestaat uit alfanumerieke tekens die uniek zijn voor elke gebruiker van Mini BlocklyChain, dit adres wordt gebruikt om transacties uit te voeren tussen gebruikers en of het nu gaat om een publiek of privaat netwerk.

Magisch nummer. - **Dit** is een willekeurig getal dat wordt gedefinieerd door de bedrijfsregels voor elke lopende UXTO-transactie en dat dient om het knooppunt te machtigen om een kandidaat te zijn om de UXTO-transactie uit te voeren en een nieuw blok de Mini BlocklyChain te creëren.

Creatie van een nieuw blok. - Een nieuw blok in Mini BlocklyChain is een hasj die is gecreëerd en bevestigd door het knooppunt dat uitkwam als de winnende kandidaat om de UXTO-transactie te verwerken.

PoW (Proof of Work) consensusprotocol. - Het is een test die alle knooppunten uitvoert en het eerste knooppunt dat de test met succes afrondt is degene die gekozen is om de UXTO-transactie uit te voeren. Het is een algoritme dat is samengesteld uit wiskundige berekeningen om een resultaat (hash) te verkrijgen volgens de regels gegeven in elke transactie uitgevoerd door Mini BlocklyChain is gebaseerd op de slijtage of de vraag van de informatieverwerking middelen van computers, in het geval van Mini BlocklyChain is gestructureerd en gewijzigd om een "magisch getal" te verkrijgen dit is een getal om

toestemming of consensus van de meerderheid van de knooppunten te kunnen hebben om de UXTO-transactie uit te kunnen voeren. De PoW heeft een maximale moeilijkheidsgraad van 5 omdat deze alleen wordt gebruikt om het "magische getal" te verkrijgen.

PoQu consensusprotocol (kwantumtest) - Het is een test die alle knooppunten uitvoert en die in eerste instantie wordt samengesteld door het PoW om het "magische getal" te verkrijgen en later voert het een waarschijnlijkheidsalgoritme uit op basis van een QRNG (Quantum Random Number Generator) een quantum random number generator, dit proces zorgt voor de gelijkheid van waarschijnlijkheid voor alle knooppunten en dus kiezen welk knooppunt de UXTO-transactie en de creatie van het nieuwe blok zal uitvoeren.

Merkleboom. - Dit is een beveiligingsmethode om Mini BlocklyChain te verzekeren dat transacties geldig zijn of niet zijn gewijzigd door een externe entiteit. Een hasj merkboom is een binaire of niet-binaire gegevensboomstructuur waarin elk knooppunt dat geen blad is, wordt gelabeld met de hasj van de aaneenschakeling van de labels of waarden van de kinderknooppunten. Ze zijn een veralgemeening van haslijsten en hasjsnaren.

Redis DB. - Real time transactiedatabase die wordt gebruikt om de gevraagde nieuwe transacties te verwerken en naar de knooppunten te sturen.

SQLite DB. - Database waarin de balansen en nieuwe blokken voor Mini BlocklyChain worden opgeslagen om de integriteit van het netwerk te waarborgen.

Sentry. - Beveiligings- en data-integriteitsconnector tussen Redis en SQLite. Deze connector of programma is verantwoordelijk voor het beoordelen, verwerken, valideren, distribueren en vertalen van de transacties naar de knooppunten.

OpenSSH. - Beveiligingsaansluiting voor het uitvoeren van taken binnen het Android-besturingssysteem.

Termux Shell Terminal. - Programma waar u afhankelijkheden van derden kunt vinden om de transacties van Mini BlocklyChain te verwerken, in deze versie 1.0 wordt het gebruikt voor eenvoudige installatie, maar in toekomstige versies zal het worden vervangen door het BlocklyShell systeem dat momenteel in ontwikkeling is.

Portemonnee. - Het is de digitale opslagplaats waar twee fundamentele gegevens worden bewaard in elke transactie; de publieke en private sleutels die zullen worden gebruikt als respectievelijk het bronadres en de digitale handtekening voor elke uitgevoerde transactie, alsook het saldo van de verzonden of ontvangen transacties. Het is een opslagplaats waar de Mini BlocklyChain-adressen worden bewaard, evenals de resultaten van de UXTO-transacties (Balance).

Applicatieontwikkelaar of programmeur:

Object georiënteerde programmering (Java) - Mini BlocklyChain is gemaakt in Java.

BlocklyCode. - Het is de looptijd van de intelligente contracten die kunnen worden uitgevoerd in de Mini BlocklyChain omgeving, de BlocklyCode is gemaakt in java-programmering.

OpenJDK voor Android. - Het is de suite van JDK en JRE voor Android waar de BlocklyCode wordt gemaakt en uitgevoerd.

QRNG (Quantum Random Number Generator). - Het is een quantum random number generator, Mini BlocklyChain heeft momenteel twee API's voor de generatie van deze.

rsync. - Het is een gratis toepassing voor Unix- en Microsoft Windows-systemen die een efficiënte overdracht van incrementele gegevens biedt, die ook met gecomprimeerde en gecodeerde gegevens werkt.

valsemunterij. - Het is een applicatie om eenvoudig en veilig bestanden te delen vanaf de commandoregel.

NTP. - Network Time Protocol, is een internetprotocol voor het synchroniseren van de klokken van computersystemen door middel van pakketrouting in netwerken met variabele latentie.

Termux (ontwikkeling). - Shell-terminal met een breed scala aan open source-toepassingen en bibliotheken.

Blokkerige modules (gegevens). - Ontwikkelaars kunnen modules integreren om de functies van Mini BlocklyChain te verbeteren.

Peer to Peer. - Deze term verwijst naar de communicatie tussen knooppunten op een directe manier, dat wil zeggen dat de informatie-update niet afhankelijk is van een centrale server, maar elk knooppunt werkt als een centrale server die tussen alle knooppunten communiceert met dezelfde informatie, wat helpt om storingspunten te voorkomen.

Synchronisatie. - hulpmiddel voor het synchroniseren van gegevens of bestanden tussen twee apparaten met behulp van het communicatietype "Peer to Peer".

Red Tor. - Het gaat hier om een gedistribueerd communicatienetwerk met weinig vertraging dat op het internet wordt overlapt, waarbij de routering van de tussen gebruikers uitgewisselde berichten hun identiteit, d.w.z. hun IP-adres (netwerkbrede anonimiteit), niet onthult.

Mobiele beworteling. - Installatie proces van externe software op uw telefoon in te voeren als een systeembeheerder in Linux-besturingssystemen (Android) de beheerder gebruiker wordt genoemd "root" om te kunnen draaien uw telefoon zal toegang hebben tot elk proces. Het is belangrijk om op te merken dat sommige fabrikanten van mobiele telefoons (smartphones) zeggen dat de garantie verloren gaat door een fout in de mobiele telefoon.

6. Procesarchitectuur in Mini BlocklyChain

Mini BlocklyChain wordt gevormd door drie processen die de architectuur ervan vormen, het eerste zijn de bedrijfsprocessen, het tweede zijn de communicatieprocessen en het derde is de ontwikkelingsomgeving voor aanvullende modules en/of de creatie van BlocklyCode "intelligente contracten".

De bedrijfsprocessen zijn de blokken die een reeks van routines vormen om een transactie te creëren in een openbaar of privé netwerk, dit type proces is de planning van het bedrijf, het hoe, wanneer, wat, wie, waar en meer attributen zullen worden besteld, gepland en gedistribueerd zodat de belangrijkste functionaliteit van elke verzonden, ontvangen, opgeslagen en/of afgekeurde transactie wordt bereikt. Het eerste proces bestaat uit de volgende soorten blokken die zijn onderverdeeld in vier categorieën:

1. Gegevensinvoerblokken
2. Gegevensverwerkingsblokken
3. Beveiligingsblokken.
4. Modulaire datablokken (ontwikkelaars)
5. Communicatiebots.

De data-invoerblokken zijn de blokken die de transactie ontvangen met een minimum van vier invoerparameters (**bronadres, bestemmingsadres, actief, data**) en kunnen meer hebben, afhankelijk van de variabele "data", dit kan een blok zijn dat ontwikkeld is door derden of met een nulwaarde (NULL).

De dataprocesblokken ontwikkelen de logistiek, de berekening, de datanormalisatie, de logische beslissingen en de flowchecks voor elke transactie. Deze soorten blokken worden gebruikt om informatie te geven aan het bedrijfsproces en zijn voornamelijk gebaseerd op de verwerking van alle soorten informatie, evenals de conversie van verschillende soorten gegevens.

De beveiligingsblokken worden gebruikt om de informatie te valideren en ervoor te zorgen dat de transacties niet zijn gewijzigd van hun oorsprong tot hun bestemming, ze worden altijd verwerkt met verschillende beveiligingsalgoritmes die worden gebruikt in blokketentechnologieën, een van de meest gebruikte tools zijn (onder andere hash signature, digitale handtekening, AES-data-encryptie, creatie en validatie van digitale adressen).

De modulaire datablokken, dit zijn de blokken die door derden worden ontwikkeld, herinneren eraan dat Mini BlocklyChain op een modulaire manier werd gecreëerd om verrijkt te worden met nieuwe blokken volgens de behoeften van elke sector, of het nu gaat om publieke of private. Om meer te weten te komen over het maken van modules kunt u de Developers Annex raadplegen.

Zoals we eerder hebben opgemerkt de architectuur van Mini BlocklyChain in zijn tweede component zijn de processen van de communicatie, deze processen zijn degenen die verantwoordelijk zijn voor de communicatiekanalen via TCP en Sockets of communication om flexibiliteit te geven van het verzenden en ontvangen van transacties, hetzij door de verschillende middelen die worden gebruikt op het moment dat de meest gebruikte zijn: Mobiel internet, Wifi werkt momenteel aan de opname van het communicatiemedium Bluetooth.

De communicatieblokken zijn in principe gebaseerd op de uitwisseling van gegevens met beveiliging die in het overdrachtskanaal is geïmplementeerd en dit is gebaseerd op een communicatie via het SSH (Secure Shell) protocol met de verschillende stadia die een verzonden of ontvangen transactie doorloopt.

Het communicatiegedeelte is fundamenteel omdat deze processen de functie hebben om de informatie in alle knooppunten via TCP te actualiseren en de verbinding genaamd "**Peer to Peer**" de blokken die in de communicatie ingrijpen zijn gebaseerd op de uitwisseling van informatie tussen de knooppunten zonder tussenkomst van intermediaire servers, zodat elk knooppunt het onafhankelijk kan maken om een netwerk van knooppunten te creëren waar de storingspunten minimaal of bijna nihil zijn. Deze onafhankelijkheid van elk knooppunt helpt hen ook om individueel of in zijn geheel beslissingen te nemen in functie van de behoeften van het bedrijf.

De "Peer to Peer" architectuur bestaat uit drie delen die het publieke of private netwerk vormen dat u besluit te creëren, in beide gevallen is het communicatiekanaal van knooppunt tot knooppunt versleuteld.

De eerste component voor communicatie tussen mobiele apparaten (smartphones) of wifi is het voorzien van de nodes of apparaten van een netwerk waar ze overal ter wereld te vinden zijn, het communicatienetwerk waar MiniBlocklyChain is gemonteerd is het "**Tor**-netwerk".

Wat is het Tor-netwerk? - (<https://www.torproject.org>)

"**Tor** (acroniem voor Te **Onion Router** - in het Spaans) is een project met als hoofddoel de ontwikkeling van een gedistribueerd communicatienetwerk met weinig vertraging dat op het internet is gesuperponeerd, waarbij de routering van de tussen gebruikers uitgewisselde berichten hun identiteit, d.w.z. hun IP-adres (anonimitet op netwerkniveau), niet onthult en dat bovendien de integriteit en geheimhouding van de informatie die erdoorheen reist, in stand houdt".

De tweede component en niet minder belangrijke taak is om alle knooppunten in MiniBlocklyChain op elk moment dezelfde gegevens te laten hebben of hun databases en bestanden te laten synchroniseren om deze taak uit te voeren tussen de knooppunten zal "**syncthing**" worden geïmplementeerd.

Wat is het Syncting netwerk? - (<https://syncthing.net>)

Synchronisatie is een gratis open source peer-to-peer file syncing applicatie die beschikbaar is voor Windows, Mac, Linux, Android, Solaris, Darwin en BSD. U kunt bestanden synchroniseren tussen apparaten op een lokaal netwerk of tussen apparaten op afstand via het internet.

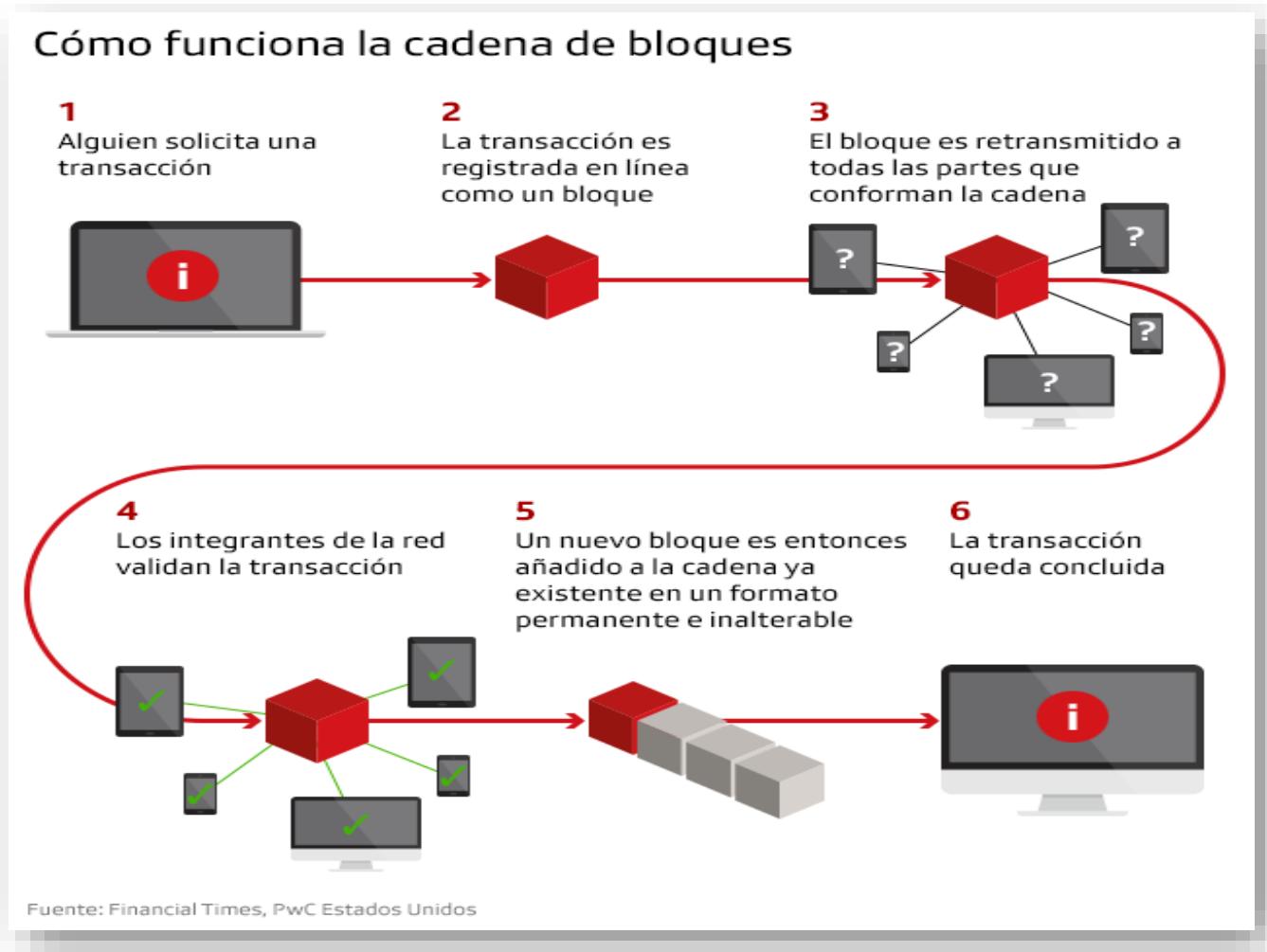
Op basis van de twee voorgaande communicatiecomponenten kunnen we beginnen met het ontwikkelen van een vertrouwensnetwerk tussen knooppunten en met het niveau van datasynchronisatie dat de ruggengraat of "kern" van de bedrijfsprocessen zal zijn om aan elke behoefte van de gebruiker of het bedrijf te voldoen.

Het derde onderdeel van het communicatiennetwerk is de Redis-database, die alle knooppunten in realtime op de hoogte stelt van nieuwe transacties die worden ontvangen en nieuwe transacties die worden verzonden.

Wat is het Redis DB-netwerk? - (<https://redis.io>)

Redis is een in-memory database engine, gebaseerd op opslag in hashtabellen (sleutel/waarde) maar die optioneel kan worden gebruikt als een duurzame of permanente database.

7. Functiediagram van de BlocklyChain (Mini BlocklyChain).



8. Wat is Mini BlocklyCode?

Mini BlocklyCode zijn programma's gemaakt in de Java taal en worden opgeslagen in een repository onafhankelijk van de knooppunten, ze worden normaal gesproken genoemd als intelligente contracten, deze programma's zijn ontworpen met logische voorwaarden voor wanneer aan deze voorwaarden wordt voldaan ze automatisch worden uitgevoerd zonder afhankelijk te zijn van enige autorisatie of externe menselijke tussenkomst.

"Een slim contract is een computerprogramma dat geregistreerde overeenkomsten tussen twee of meer partijen (bijvoorbeeld personen of organisaties) faciliteert, beveiligt, afdwingt en uitvoert. Als zodanig zouden zij hen helpen bij het onderhandelen over en het vaststellen van dergelijke overeenkomsten die ertoe zullen leiden dat bepaalde acties worden ondernomen als gevolg van het feit dat aan een aantal specifieke voorwaarden wordt voldaan.

Een slim contract is een programma dat in een systeem leeft dat niet door een van beide partijen of hun agenten wordt gecontroleerd en dat een automatisch contract uitvoert dat werkt als een if-then-zin van een ander computerprogramma. Met het verschil dat het wordt gedaan op een manier die in wisselwerking staat met de echte activa. Wanneer een voorgeprogrammeerde toestand, die niet aan een menselijk oordeel is onderworpen, in werking treedt, voert het intelligente contract de overeenkomstige contractclausule uit.

Zij hebben tot doel meer zekerheid te bieden dan het traditionele contractenrecht en de transactiekosten in verband met het sluiten van contracten te verminderen. De overdracht van digitale waarde via een systeem dat geen vertrouwen vereist (bijv. bitcoins) opent de deur naar nieuwe toepassingen die gebruik kunnen maken van intelligente contracten. "

Mini BlocklyCode is gericht op ontwikkelaars met ervaring in java-programmering. In Mini BlocklyChain zijn sommige soorten bibliotheken beperkt voor de veiligheid van hetzelfde systeem, maar bibliotheken om transacties te creëren voor het verzenden of ontvangen van een bepaalde contractuele waarde die door twee of meer partijen is gecreëerd of overeengekomen.

Elke Mini BlocklyChain voldoet aan de volgende voorwaarden:

- ✓ Elke Mini BlocklyChain die wordt aangemaakt is gebaseerd op het uitvoeren van alleen berichten en niet op executies op het systeem, maar deze berichten kunnen wel autorisaties, fysieke contractgoedkeuringen of fysieke acties bevatten.
- ✓ Elke Mini BlocklyChain die wordt aangemaakt, moet het communicatieblok "auditor" doorlopen. Dit blok is belast met het controleren van kwaadaardige code of verboden code om zinnen of ongeoorloofde orders in het systeem te controleren.
- ✓ Alle Mini BlocklyChain wordt alleen uitgevoerd op de JVM van het bronknooppunt, programma's worden niet wereldwijd uitgevoerd voor systeembeveiliging.

Voor meer details zie de bijlage "Mini BlocklyChain voor ontwikkelaars".

9. Mini BlocklyChain-communicatienetwerkinstallatie

1. Installatie en configuratie van het Mini SQLSync-netwerk

Het Mini SQLSync netwerk is verantwoordelijk voor het ontvangen van de transacties van de nodes zodat ze die transacties kunnen verwerken. Dit netwerk wordt gevormd door een hoofdnetwerk dat via "Peer to Peer" tussen de nodes en een backup netwerk genaamd Mini Sentinel RESTful.

We beginnen met de installatie van het RESTful Mini Sentinel back-up netwerk, deze service is degene die de transacties zal ontvangen van de knooppunten, deze service is gebaseerd op het opslaan van de transacties voor een bepaalde tijd om later de informatie om te zetten via een connector die een wachtrij van alle gecodeerde transacties zal injecteren naar een Redis service. Later zal de Redis-database service in real time de vrijgave van de gecodeerde transactiewachtrij naar alle knooppunten die het Mini BlocklyChain-netwerk integreren, uitvoeren.

Een RESTful type dienst of ontwerp is een eenvoudige en gemakkelijke manier om informatie te raadplegen, te wijzigen, te verwijderen en/of in te voegen in een database via een URL of internetadres, in ons geval zullen we de SQLite database gebruiken vanwege de kenmerken van het feit dat het alle informatie in één bestand is. Voor een gebruik van eisen met behoeften van grote schaal zullen we in staat zijn om een ander type van de database te gebruiken als MySQL, Oracle, DB2 of een andere commerciële database, gewoon moeten we de connector van Mini BlocklyChain van SQLite (gratis versie) voor de connector van Mini BlocklyChain DB anderen (commerciële versie) te wijzigen.

BELANGRIJKE OPMERKING: RESTful Mini Sentinel-configuratie verwerkt op geen enkel moment enige vorm van transacties, het is alleen de service die "de informatie vormgeeft" zodat de knooppunten de transactieverwerking goed en in een geplande en gesynchroniseerde tijd voor alle knooppunten kunnen uitvoeren.

De RESTful service installatie is gebaseerd op een SQLite database. Deze installatie wordt in een Windows omgeving gedaan voor praktische doeleinden, maar dit model kan eenvoudig worden gerepliceerd in een Linux type besturingssysteem.

Voor degenen die de prestaties en ondersteuning van SQLite queries en inserties willen beoordelen, verwijzen we naar deze prestatietesten:

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

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

RESTful Mini Sentinel backup netwerk installatie. We zullen deze dienst installeren en configureren in een Windows 10 computer, maar het proces is ook eenvoudig te installeren in een Ubuntu 18.09 Linux omgeving.

Eisen:

- ✓ SQLite database server in RESTful modus (<https://github.com/olsonpm/sqlite-to-rest>).
- ✓ SQLite-Redis Sentinel Connector
- ✓ Redis-databaseserver in Master-Slave-modus (<https://redis.io/topics/replication>)

Installatie van SQLite database server in RESTful modus

Voor de installatie moeten we beginnen met de volgende softwarepakketten, Nodejs, sqlite3 en Git Bash voor Windows.

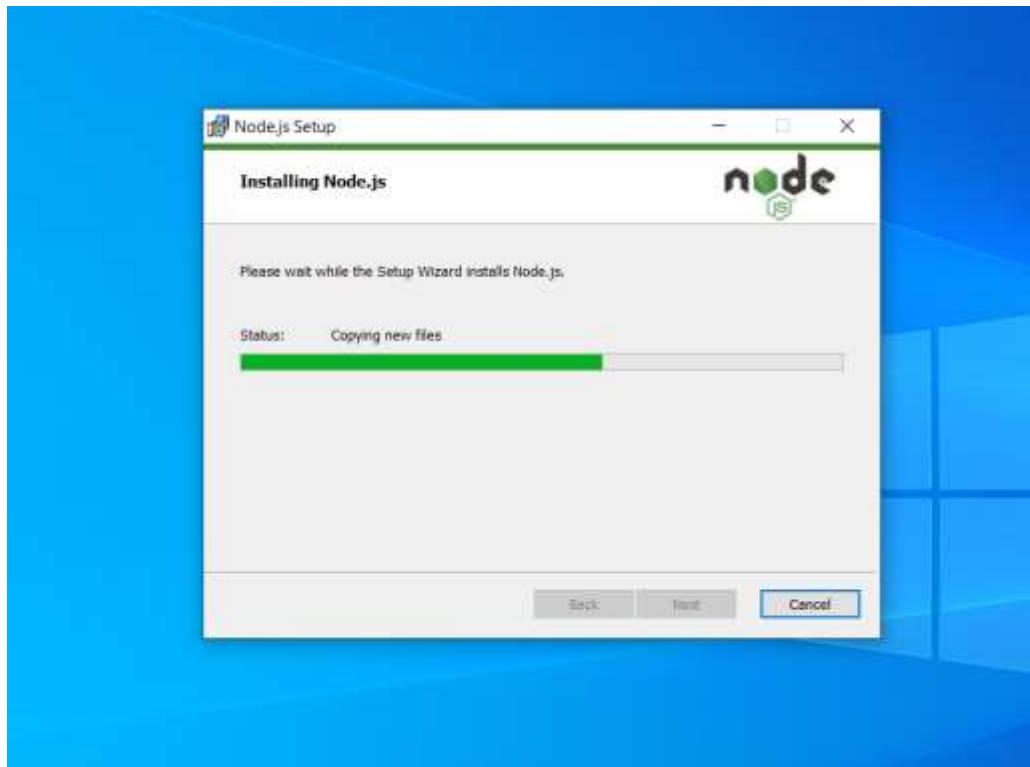
Om Nodejs te verkrijgen hebben we hun officiële site <https://nodejs.org/es/> geraadpleegd en hebben we de "Recommended for Most" versie gekozen.

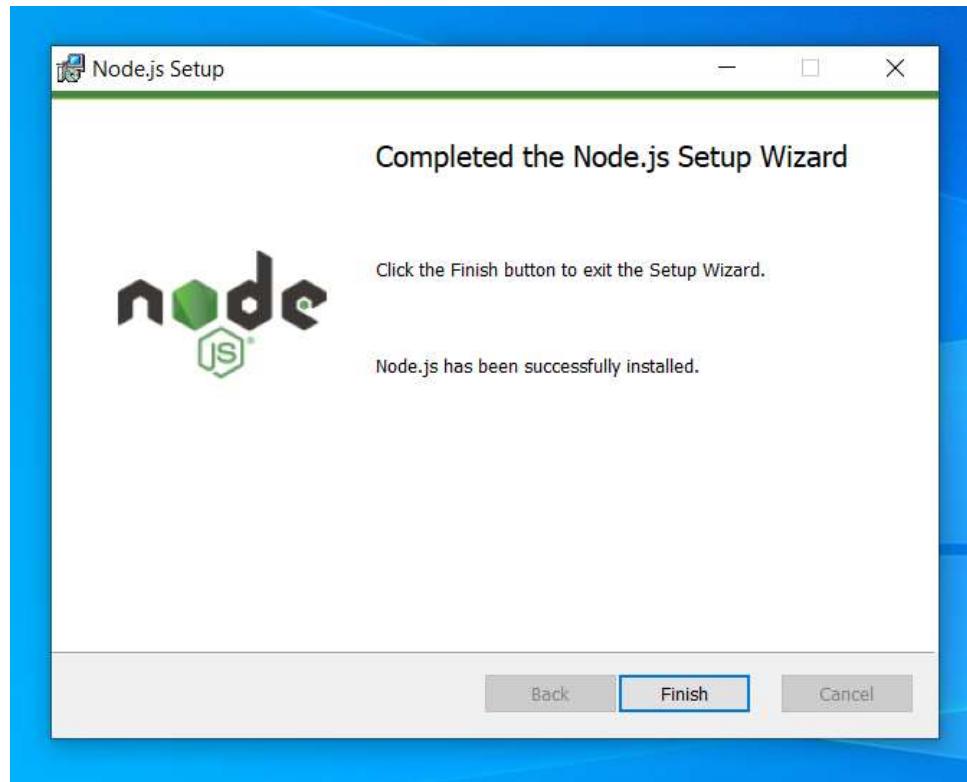


Na het downloaden van het bestand met de extensie .msi dubbelklikken we om het te installeren.



Wij voeren de installatie standaard uit, klik op "Volgende" zonder extra opties te kiezen.



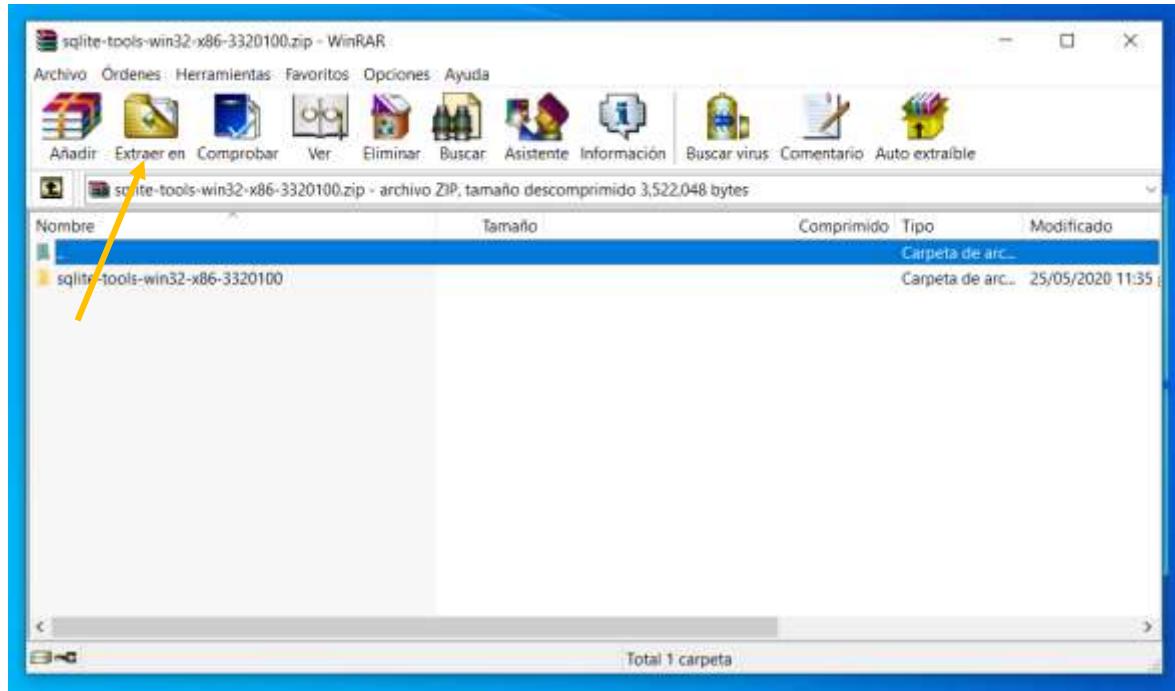


Sinds we klaar zijn met de installatie van Nodejs gaan we verder met de installatie van de SQLite database.

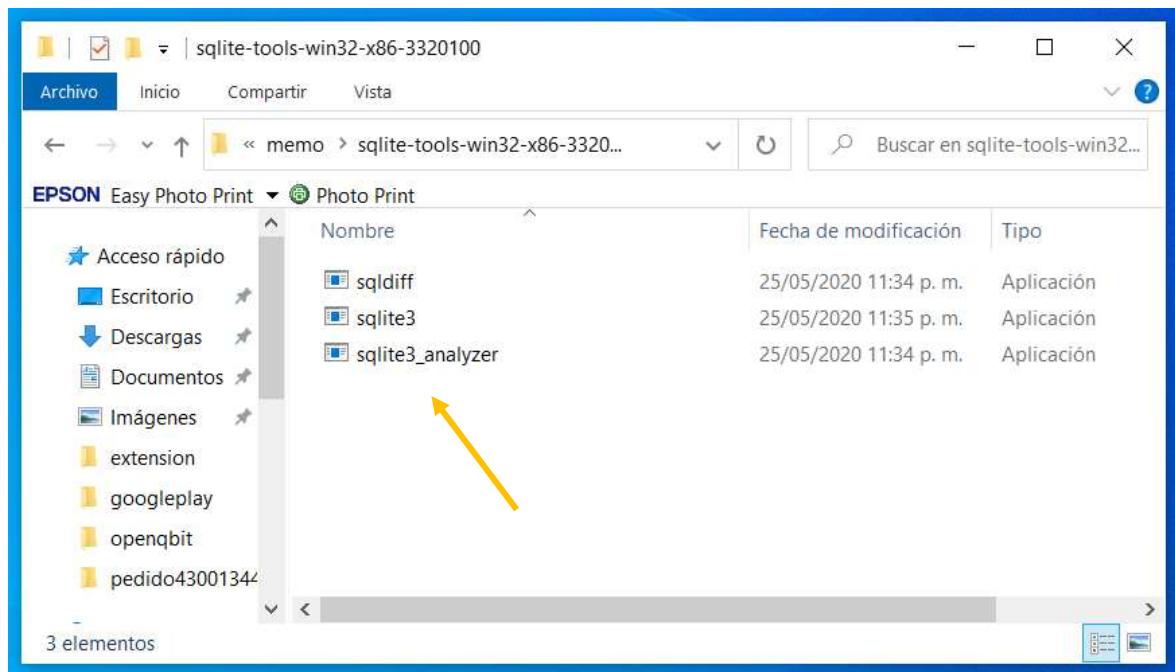
We gaan naar hun officiële site, <https://www.sqlite.org/index.html> en klikken op het gedeelte waar je "download".

A screenshot of the SQLite.org website. The URL in the address bar is https://www.sqlite.org/index.html. A yellow arrow points to the "Download" button in the top navigation menu. Below the menu, there's a section titled "What Is SQLite?" with a brief description and a link to "More Information...". Another section discusses the SQLite file format and its widespread use. At the bottom, there's a "Latest Release" section with a link to "Version 3.32.1 (2020-05-25)" and download links for "Download" and "Prior Releases".

Vervolgens kiezen we de optie waarin staat "Voorgecompileerde binaries voor Windows" en klikken we op de softwareversie "sqlite-tools-win32-x86-3320100.zip" als het downloaden klaar is gaan we over tot het decomprimeren van het bestand op een eenvoudige manier, we openen de map waar het bestand is gedownload en dubbelklikken op het bestand om het te decomprimeren.

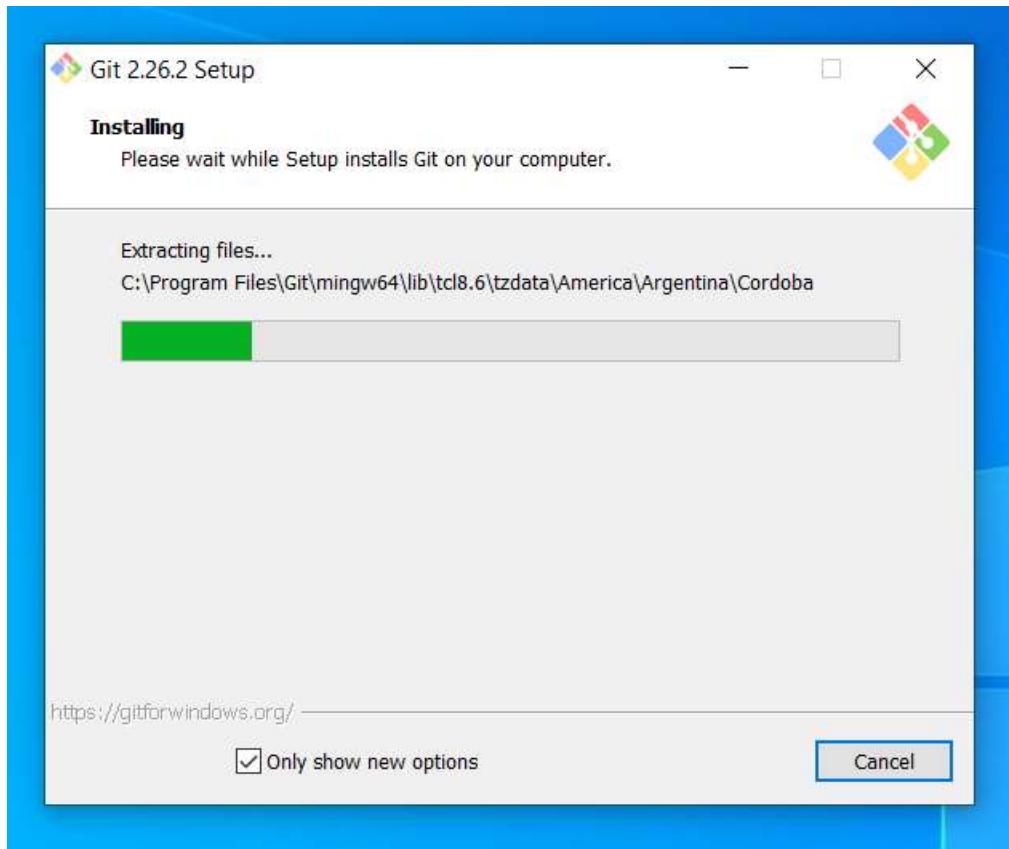


Na het decomprimeren hebben we drie bestanden, dit zijn onze omgeving om onze SQLite database te creëren die we later zullen gebruiken.



We beginnen met het installeren van Git Bash, een Linux-achtige terminal emulator die ons zal helpen om de RESTful back-up service goed te laten draaien.

We zullen het downloaden van hun officiële site, <https://gitforwindows.org/> en doorgaan met het installeren ervan met de standaard opties die het aangeeft.



Nu we nodejs, sqlite en Git Bash hebben geïnstalleerd op onze Windows 10 machine gaan we over tot het installeren van de omgeving die onze SQLite database in RESTful modus zal ondersteunen.

Op het moment van het downloaden van de software die de functionaliteit van RESTful aan SQLite zal geven. Hiervoor moeten we naar de volgende site gaan, <https://github.com/olsonpm/sqlite-to-rest> hierin zullen we op de groene rechter knop "Clone or download" klikken en de optie "Download ZIP" kiezen deze zal de software in onze computer downloaden.

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

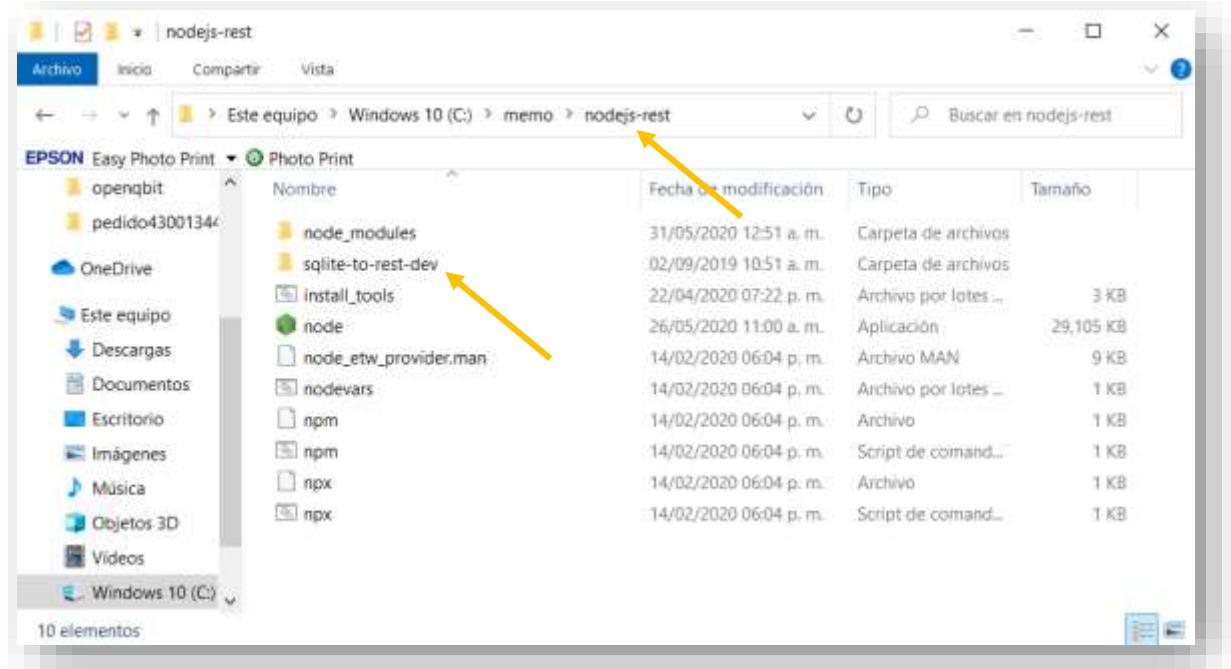
Open in Desktop • Download ZIP

9 months ago • 9 months ago • 9 months ago

File	Description	Modified
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

Na het downloaden in onze computer gaan we over tot het decomprimeren in de directory of de map waar we het **nodejs** programma hebben geïnstalleerd en hebben we een directory met de naam "**sqlite-to-rest-dev**".

OPMERKING: Het is belangrijk dat de **sqlite-to-rest-dev** directory zich in de directory bevindt waar **nodejs** is geïnstalleerd.



Nadat alles is geïnstalleerd gaan we verder met de configuratie van de SQLite database die we zullen gebruiken om de transacties van de nodes op te slaan in de RESTful backup omgeving.

Tabelontwerp en gegevensstructuur. Commando's die online moeten worden uitgevoerd in de CMD-commandolijst

sqlite3 op.sqlite3 "CREATE TABLE trans (id gehele primaire sleutel, addro, addrd, waarde);".

sqlite3 op.sqlite3 "CREATE TABLE teken (id gehele primaire sleutel, trans_id referenties trans (id), teken, hash);"

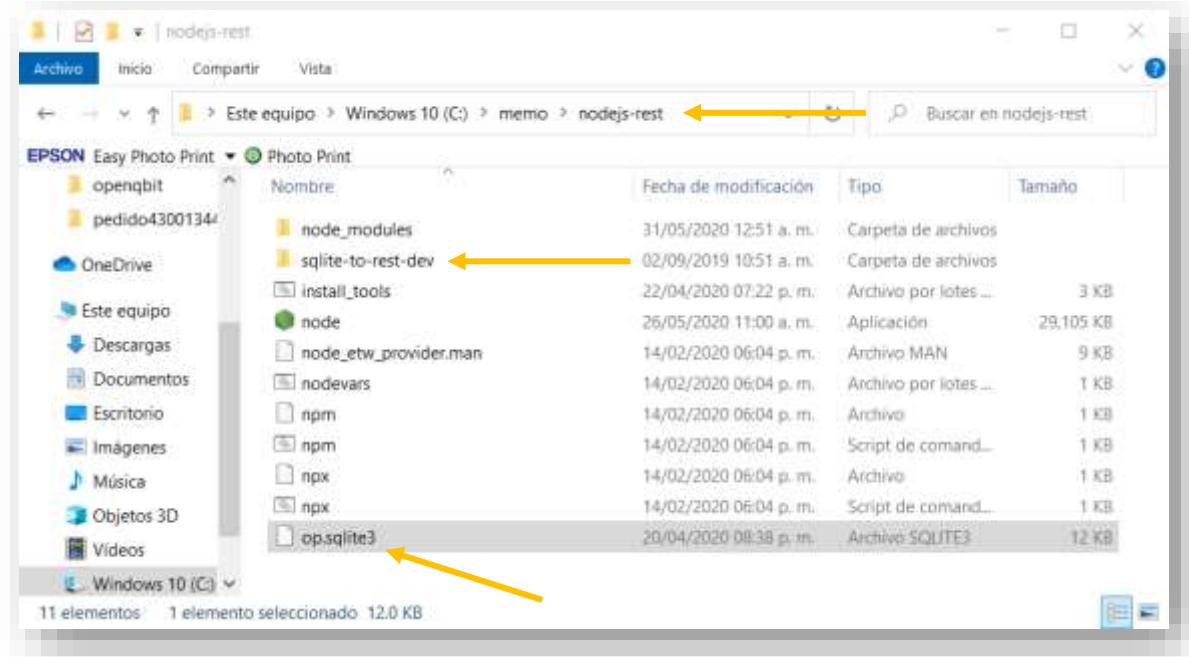
```
C:\memo>sqlite3 op.sqlite3 "CREATE TABLE trans(id integer primary key, addro, addrd, waarde);"
C:\memo>sqlite3 op.sqlite3 "CREATE TABLE sign(id integer primary key, trans_id references trans (id), sign, hash);"
C:\memo>sqlite-tools-win32-x86-3320100>dir
El volumen de la unidad C es Windows 10
El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

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

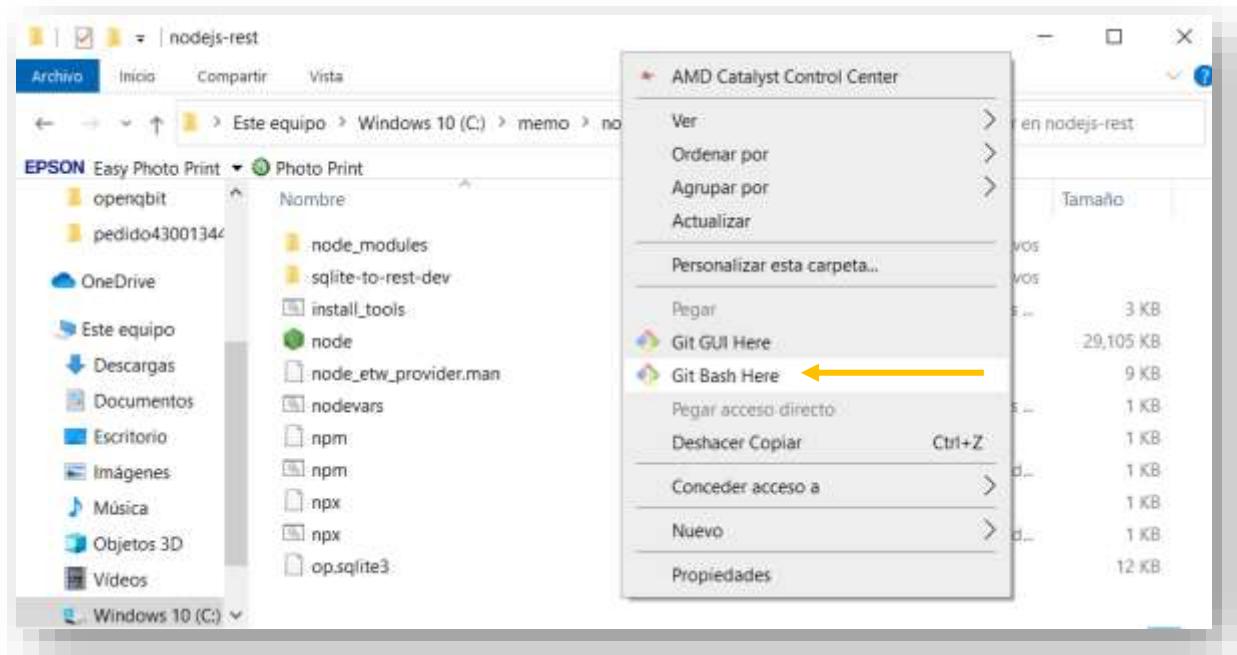
31/05/2020 02:29 a. m.    <DIR>
31/05/2020 02:29 a. m.    <DIR>..
31/05/2020 02:29 a. m.        12,288 op.sqlite3
25/05/2020 11:34 p. m.      516,608 sqldiff.exe
25/05/2020 11:35 p. m.      972,800 sqlite3.exe
25/05/2020 11:34 p. m.     2,032,640 sqlite3_analyzer.exe
                          4 archivos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

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

Na het aanmaken van de database op.sqlite3 moeten we een kopie maken in de directory waar de nodejs zijn geïnstalleerd. In de **nodejs directory** moet ook de kopie van "sqlite-to-rest-dev" staan.



We plaatsen ons in de map waar de **nodejs** installatie is en waar de "**sqlite-to-rest-dev**" software ook zou moeten zijn. Wijs de map aan met de aanwijzer en klik met de



rechtermuisknop om het menu weer te geven en kies waar er "Git Bash" staat om een terminal te openen.

In de nieuwe open Git Bash terminal voeren we de volgende commando's uit:

\$ 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
$
```

Na de uitvoering van het commando zal de installatie van het "sqlite-to-rest" pakket verschijnen.

We hebben een RESTful omgeving voor SQLite gegenereerd met de **op.sqlite3** basis die we eerder hebben gecreëerd.

We voeren het commando uit: **\$ sqlite-to-rest generator-skeleton --db-path ./op.sqlite3**



```
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!
$
```

We zijn begonnen met RESTful SQLite service op standaard poort 8085. We voeren het volgende commando uit: **\$ node skeleton.js**



```
MINGW64/c/memo/nodejs-rest
$ node skeleton.js
$ listening on port: 8085
```

We hebben de RESTful-service getest met het toevoegen van gegevens en het opvragen van



```
MINGW64/c/memo/nodejs-rest
$ curl -s -H "Content-Type: application/json" -d '{"addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/trans
{"id":6,"addr": "QWERTY1234","addrd": "ASDFG4567","value": "999"}  
$ curl -s http://localhost:8085/trans
[{"id":1,"addr": "CO","addrd": "souldee","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"}]
```

gegevens.

Om de SQLite RESTful service te testen gebruiken we de volgende commando's:

Om gegevens in te voegen in de tabel trans die zich binnen de database op.sqlite3 bevindt:

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

Om een query te maken van alle gegevens in de trans-tabel:

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

Bekijk de bijlage "Rustige SQLite GET/POST-commando's" om te zien hoe alle RESTful-services (query's, invoegen, updaten en/of verwijderen van gegevens) in detail kunnen worden gebruikt.

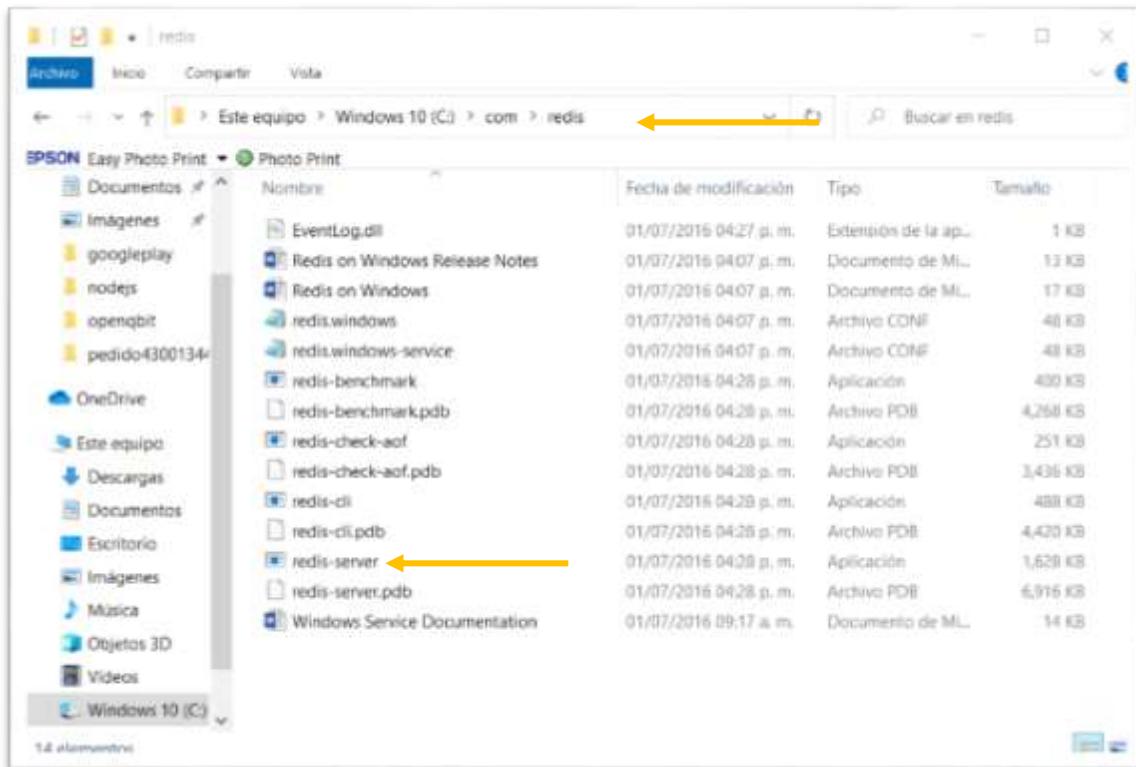
OPMERKING: In de vorige installatie werden alle vragen gesteld in de URL met het adres "localhost", maar wanneer de server wordt blootgesteld aan een publiek IP (internet) of privé IP (Wifi) zal het zonder enig probleem werken. We zullen dit testen wanneer we de communicatietesten doen tussen de SQLite RESTful service en de communicatienetwerkknooppunten die de Mini BlocklyChain vormen.

Redis database installatie voor backup netwerk service, om de redis software voor Windows te downloaden moeten we naar de site gaan, <https://github.com/microsoftarchive/redis/releases/tag/win-3.2.100> en het ZIP pakket kiezen.

The screenshot shows a GitHub release page for Redis version 3.2.100. At the top, there's a 'Pre-release' button, a 'win-3.2.100' link, and a 'dev8757' link. Below that is a 'Compare' button. The main title is '3.2.100'. It says 'enricogiar released this on 1 Jul 2016 · 1210 commits to 3.0 since this release'. A note states: 'This is the first release of Redis on Windows 3.2. This release is based on antirez/redis/3.2.1 plus some Windows specific fixes. It has passed all the standard tests but it hasn't been tested in a production environment. Therefore, before considering using this release in production, make sure to test it thoroughly in your own test environment.' It also says 'See the [release notes](#) for details.' Under the heading 'Assets', there are four items: 'Redis-x64-3.2.100.msi' (5.8 MB), 'Redis-x64-3.2.100.zip' (4.95 MB) with a yellow arrow pointing to it, 'Source code (zip)' (1.1 MB), and 'Source code (tar.gz)' (1.1 MB).

Om de installatie te lokaliseren maken we een map genaamd "redis" binnen Windows en downloaden we het bestand met een ZIP-extensie, we decomprimeren het in de eerder gemaakte map, we hebben de installatie al voltooid redis.

We testen de installatie door de server te laten draaien door te dubbelklikken op het commando "redis-server".



Na het uitvoeren van het commando "redis-server" zullen we de server zien draaien in een Windows CMD commando terminal op de standaard poort 6379:

```
C:\com\redis\redis-server.exe
[10192] 31 May 13:50:45.348 # 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
```

In deze test hebben we een versie van Redis 3.2.100 voor Windows 10, in het geval van Linux hebben we de versie 6.0.4 (uitgebracht in mei 2020) maar voor onze Mini BlocklyChain configuratie heeft de Windows versie genoeg functionaliteiten die we nodig hebben.

Om de uitvoering te stoppen maken we de toetsencombinatie Ctrl + C. We gaan verder met het configureren van de Redis 3.2.100 database voor Windows 10 met een configuratie tussen Redis en de Mini SQLSync communicatie netwerk nodes, type **Master(Master)-Slave(Slave)** is als volgt verdeeld:

De Master is de Redis-server voor Windows 10 die we aan het configureren zijn en deze zal de data in real time repliceren en synchroniseren met dezelfde informatie naar alle nodes (mobiele telefoons). Deze nodes zullen een Redis-server geïnstalleerd hebben in **Slave-modus**.

Op dit punt beginnen we te zien hoe het backup netwerk dat we installeren en configureren werkt. Deze configuratie zal ons dienen om te communiceren met alle knooppunten in real time, het zal ons helpen om te communiceren met alle knooppunten van het netwerk wanneer er een nieuwe transactiewachtrij wordt verwerkt door de knooppunten, in een gelijkheid in de informatieoverdracht naar alle knooppunten, zodat elk knooppunt dezelfde kans heeft om de "transactiewachtrij" informatie te kunnen verwerken.

We starten de configuratie van de **SQLite-Redis Sentinel-connector**.

Deze connector is een in Java-taal ontwikkeld programma dat, zoals de naam al aangeeft, de SQLite en Redis (**Master**) databases met elkaar verbindt.

De functie van de connector is om de informatie (transacties) van SQLite naar Redis (**Master**) te sturen en dit stuurt de "transactiewachtrij" naar de nodes (**Slaves**).

Voor meer details over de Java-code van de **SQLite-Redis Sentinel-connector** zie de bijlage "SQLite-Redis Java Code Connector".

Configuratie van de Redis (**Master**) database **redis.conf** bestand voor Windows 10.

Voeg de volgende wijzigingen of richtlijnen toe aan het bestand, sla de wijzigingen op en start Redis server

Begin met het vinden van de **tcp-keepalive** instelling en stel deze in op 60 seconden zoals gesuggereerd in het commentaar. Dit zal Redis helpen om netwerk- of serviceproblemen op te sporen:

tcp-onderhoud 60

Zoek de richtlijn voor **de eisenpas** en configureren deze met een sterke passphrase. Terwijl uw Redis-verkeer moet worden beschermd tegen derden, biedt dit authenticatie aan Redis. Aangezien Redis snel is en geen grenswaarde geeft aan passphrase pogingen, kies dan voor een sterke, complexe passphrase om te beschermen tegen pogingen met brute kracht:

vereist type_uw_network_master_wachtwoord

Voorbeeld:

vereisen FPqwedsLMdf76ass7asdd2g45vBN8ty99

Tot slot zijn er enkele optionele instellingen die u mogelijk wilt aanpassen, afhankelijk van uw gebruiksscenario.

Als u niet wilt dat Redis automatisch de oudste en minst gebruikte toetsen invoert bij het vullen, kunt u de automatische sleutelverwijdering uitschakelen:

maxmemory-policy noevasion

Voor een betere duurzaamheidsgarantie kunt u de add-only file persistentie inschakelen. Dit zal helpen om het verlies van gegevens te minimaliseren in het geval van een systeemstoring ten koste van grotere bestanden en iets tragere prestaties:

als bijlage ja

de naam "redis-stage-ao.aof".

Sla de wijzigingen op en start de Redis-service voor Windows 10 opnieuw op, stop met de toetsen Ctrl + C en voer de Windows CMD-opdrachtregel opnieuw uit:

C:\Redis_redis_directory redis_server

In ons voorbeeld kunnen we zien dat er een (**slaven**)knoop is aangesloten.

```

Símbolo del sistema
:\\com\\redis> redis-server redis.conf
1:C 31 May 2020 23:44:56.633 # o000o000o000o Redis is starting o000o000o000o
1:C 31 May 2020 23:44:56.634 # Redis version=5.0.7, bits=64, commit=00000000, modified=0,
1: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 ori
1:inally set to 1024).

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

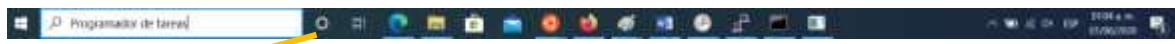
http://redis.io

1:M 31 May 2020 23:44:56.648 # WARNING: The TCP backlog setting of 511 cannot be enforced
1:because /proc/sys/net/core/somaxconn is set to the lower value of 128.51:M 31 May 2020 23:4
1:44:56.648 # Server initialized
1:M 31 May 2020 23:44:56.649 # WARNING overcommit_memory is set to 0! Background save may
1:fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/s
1:ctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to t
1:ke 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 23:50:24.679 * Background saving terminated with success
1:M 31 May 2020 23:50:24.689 * Synchronization with replica 192.168.1.68:6379 succeeded

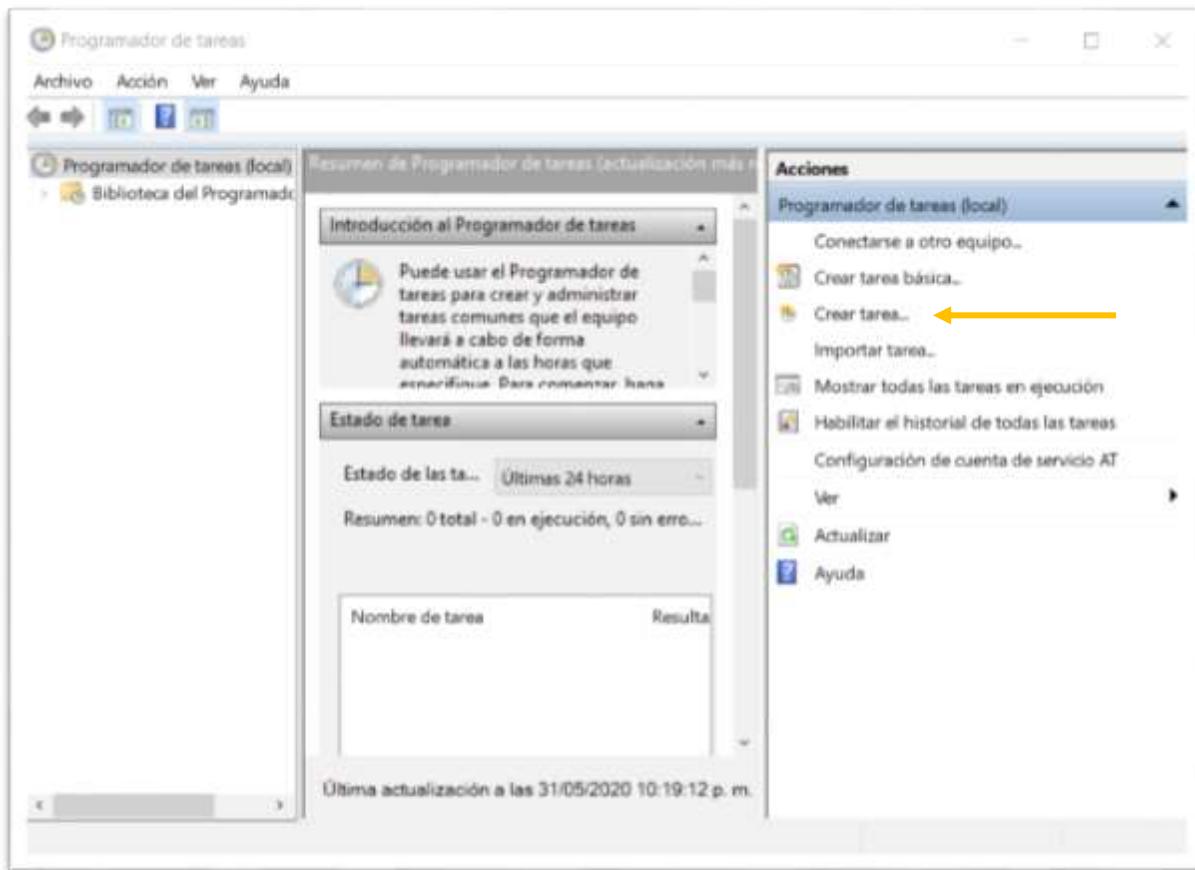
```

We zien dat we een node hebben gesynchroniseerd met het IP 192.168.1.68 in de standaard poort 6379.

Nu moeten we in het Windows 10 besturingssysteem de taak van het automatisch uitvoeren van de **SQLite-Redis Sentinel** connector plannen. Dit doen we met de Windows 10 tool, die we linksonder uitvoeren door "Task Scheduler" te typen.



We zullen een nieuwe taak "Create task" aanmaken waarbij we de uitvoering van de



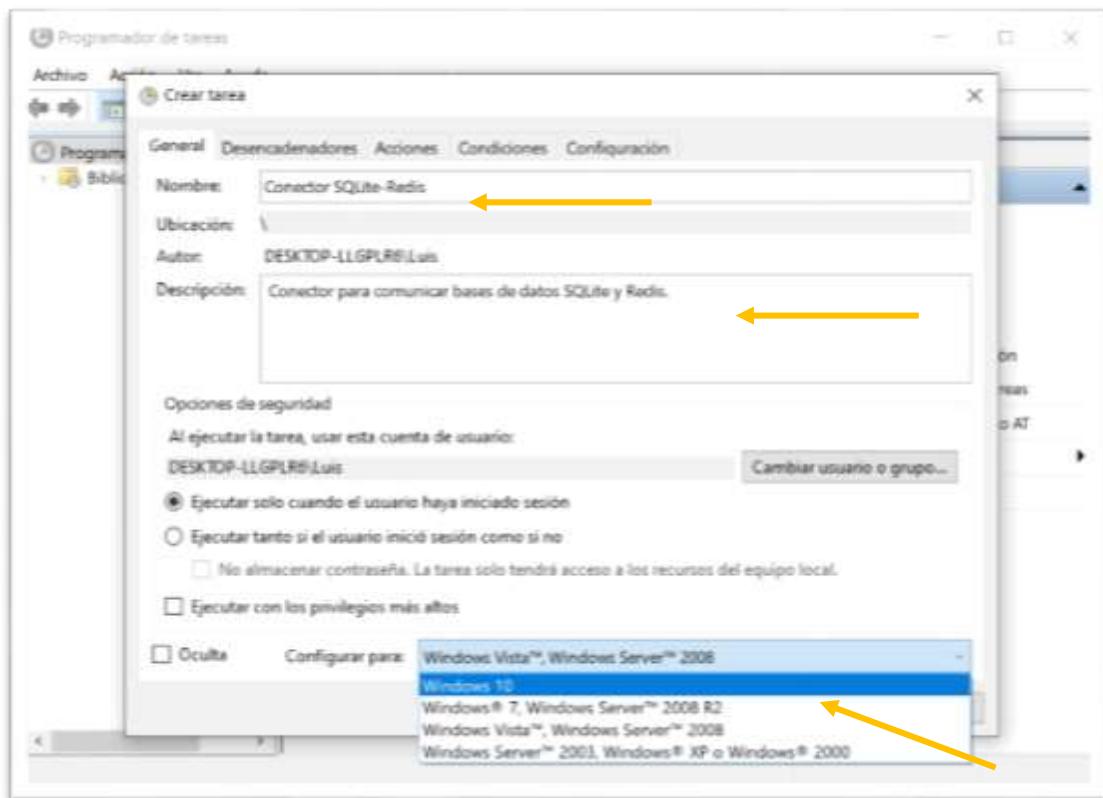
connector zullen opnemen.

Door te klikken op "Taak aanmaken" wordt een extra venster geopend waarin we de volgende parameters in het tabblad "Algemeen" moeten opgeven:

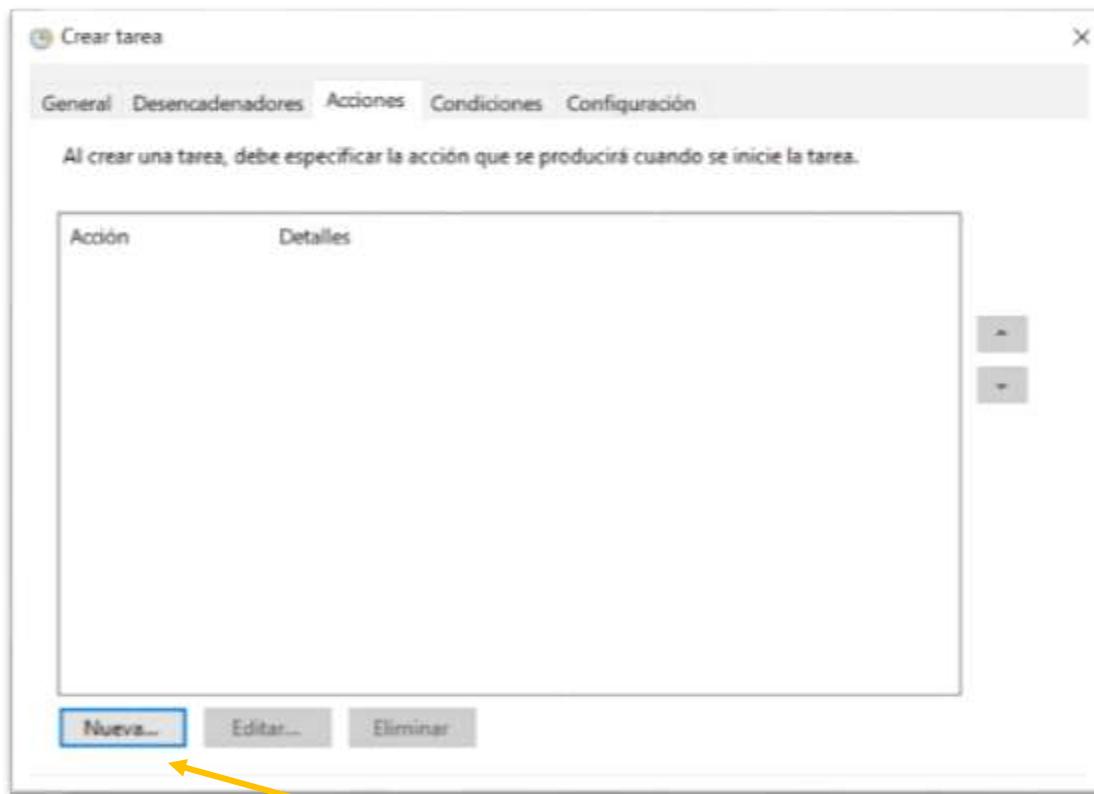
Naam: task_name

Beschrijving: optioneel

Configureren voor: select_operating_system_windows_version



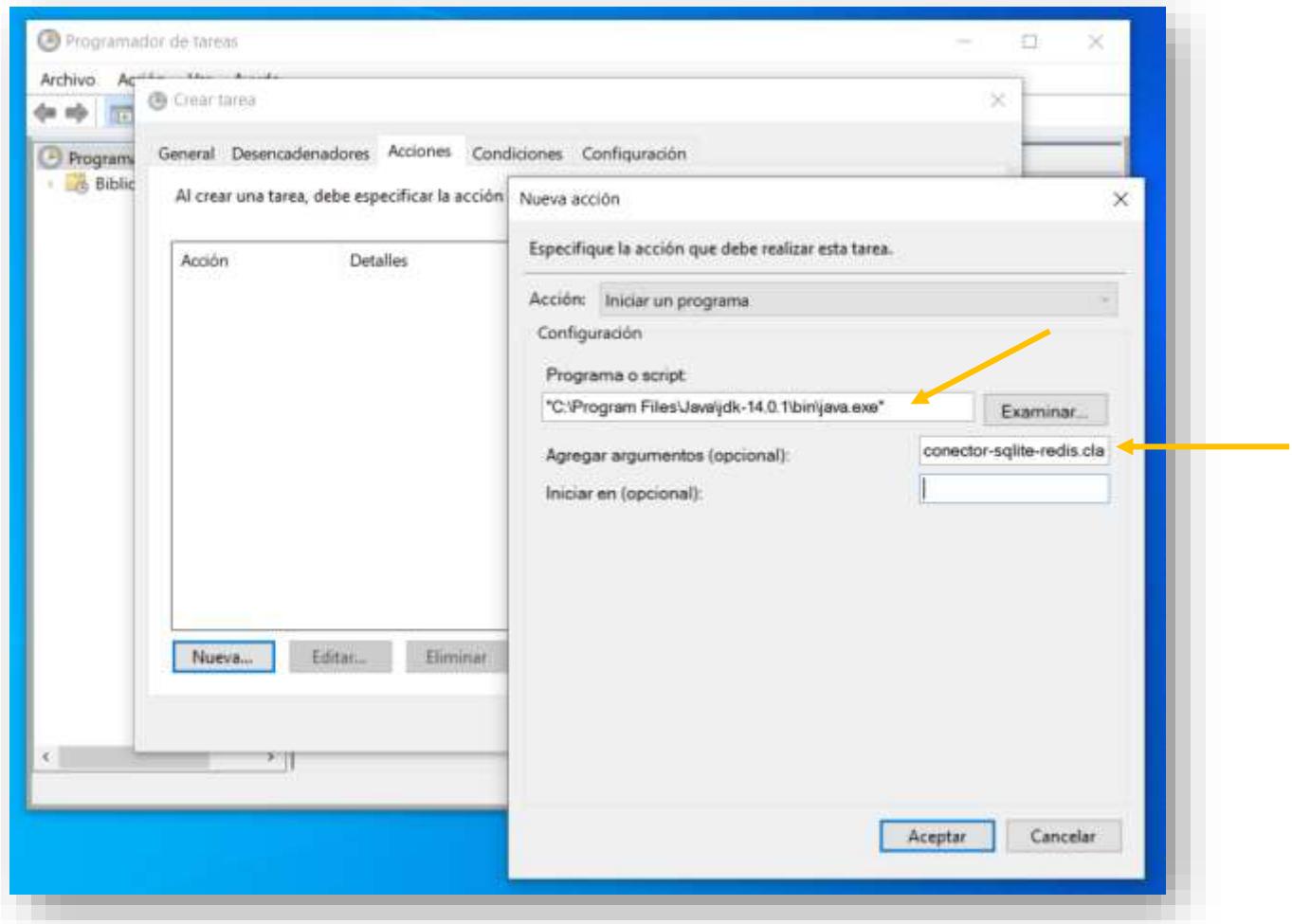
Wijzig door op het tabblad "Acties" te klikken en op de knop "Nieuw" te klikken:



We geven de volgende parameters:

Programma of script: connector_pad

Argumenten toevoegen: naam van de connector



De bovenstaande parameters kunnen variëren afhankelijk van de locatie van de connector. Het belangrijkste idee is de uitvoering van het **connector-sqlite-redis-v1.class** programma zoals dat normaal gesproken op de commandoregel wordt gedaan:

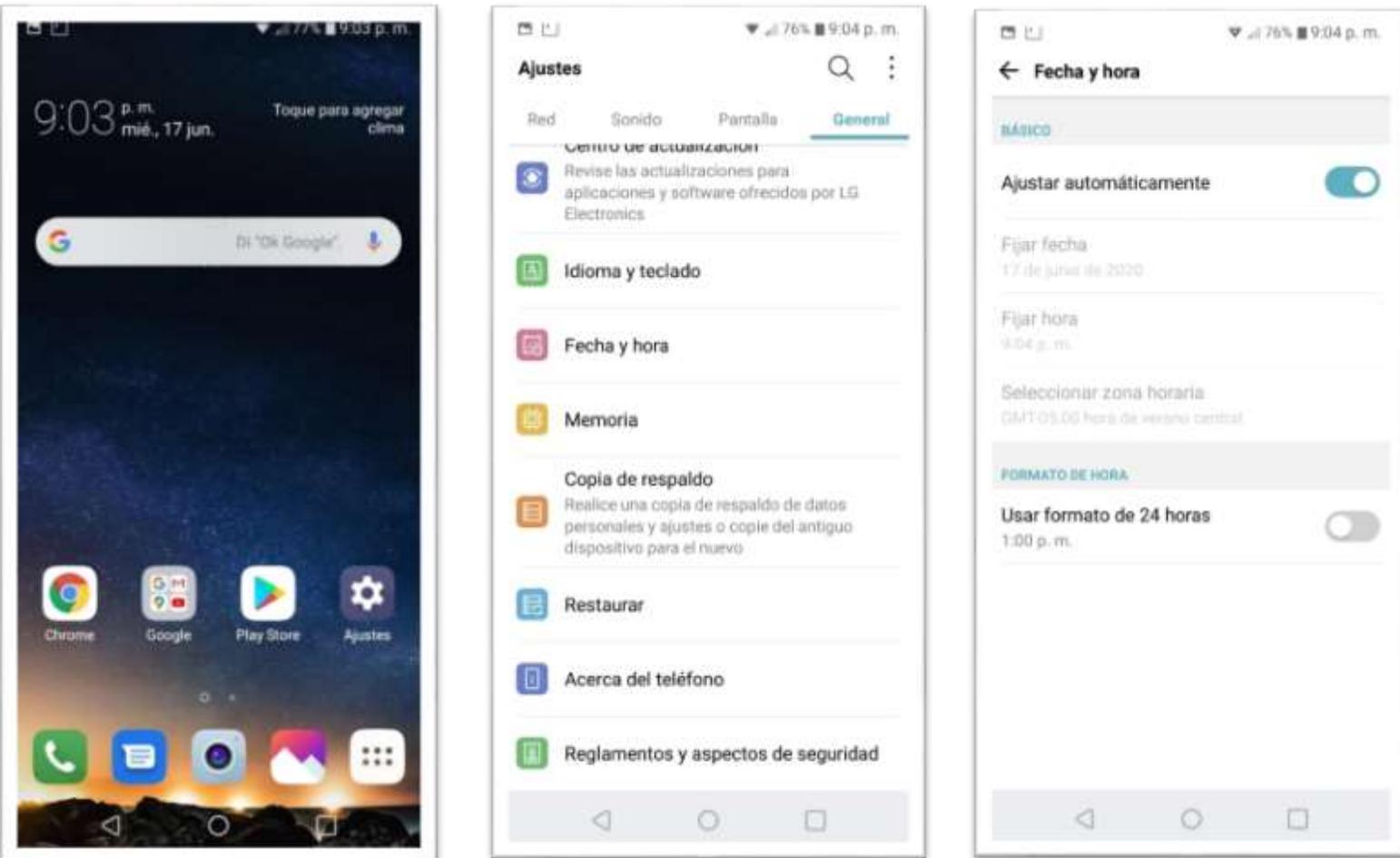
C:\jdk_directory java connector-sqlite-redis-v1

Om te eindigen moeten we het tabblad "Triggers" kiezen, hierin geven we de parameters van wanneer (dag, uur, minuten) we de taak willen laten uitvoeren, deze parameters zijn gebaseerd op de bedrijfsregels die het Mini BlocklyChain systeem zal moeten worden aangemaakt.

10. Synchronisatie in systeemknooppunten (mobiele telefoon) minuten en seconden.

Het is zeer belangrijk dat alle knooppunten gesynchroniseerd zijn, vooral in het gedeelte van de minuten en seconden. Aangezien wanneer het verzenden of publiceren in een bepaalde tijd van de transactiewachtrij wordt gedaan, moeten alle knooppunten worden gesynchroniseerd, aangezien dit zal afhangen van de taakmanager die in het lokale systeem zal worden gevestigd met de CRON-tool om op hetzelfde moment in alle knooppunten te worden uitgevoerd, zal dit ervoor zorgen dat alle knooppunten dezelfde waarschijnlijkheid hebben om het recht te krijgen om de gekozen te zijn om de transactiewachtrij te kunnen verwerken en om het nieuwe blok te kunnen genereren om het toe te voegen aan de blokketten van het systeem. Om de knooppunten te synchroniseren hebben we twee opties.

De eerste optie van de synchronisatie van het knooppunt, zullen we in staat zijn om het te doen op een eenvoudige manier. In ons toestel is door de interne optie die het Android



systeem omvat, zullen we moeten gaan naar het gedeelte van **Instellingen > Datum en Tijd > Automatisch aanpassen**

Met de vorige configuratie zullen we hebben gesynchroniseerd in minuten en seconden alle knooppunten van het systeem, ongeacht welk land in de wereld zijn al gesynchroniseerd is gebaseerd op elk geografisch gebied mogelijk wat veranderingen is de tijd, maar afhankelijk van de minuten en seconden moet worden gesynchroniseerd dit is genoeg voor ons om het proces van taken uit te voeren op een geplande basis met de cron tool in alle knooppunten te lopen elke bepaalde tijd in minuten, dat wil zeggen dat we een taak in crontab om elke 10 minuten of 30 minuten, afhankelijk van elk systeemontwerp uit te voeren.

Dit zal nuttig zijn bij het gebruik van het "Peer to Peer" communicatie netwerk, in het geval van het gebruik van het back-up netwerk zal dit proces niet van toepassing zijn omdat de distributie van de transactie wachtrij wordt gedaan in een client-server model en de server is degene die de cron tool bestuurt.

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

De tweede optie is het gebruik van een externe API waarbij we het Curl commando via de extensie (**ConnectorSSHClient**) uitvoeren.

De plaats waar we gebruik zullen maken van de externe diensten van NTP (Network Time Protocol) is:

<http://worldtimeapi.org/>

Nu zullen we kijken naar een manier om de tijd te krijgen van de NTP-servers die zich wereldwijd bevinden en dat zal ons helpen om alle nodes te krijgen om een query te hebben op een bepaalde tijd op dezelfde datum en tijd.

Voorbeeld van een query met extensie (**ConnectorSSHClient**).

\$ krul "http://worldtimeapi.org/api/timezone/America/Mexico_City"

We hebben de verbinding met de Termux-terminal gemaakt:



We voeren het Krulcommando uit:



We moeten er rekening mee houden dat het resultaat van het Curl-commando in JSON-formaat zal zijn, iets wat vergelijkbaar is met het volgende resultaat:

```

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

```

Ook kan het resultaat in JSON-formaat zijn zonder de gegevens geformatteerd in of JSON in lineaire vorm zoals hieronder weergegeven:

```
{"afkorting": "CDT", "client_ip": "200.77.16.151", "datetime": "2020-06-18T14:19:07".216800-05:00", "dag_van_week": 4, "dag_van_jaar": 170, "dst": waar, "dst_van": "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_nummer": 25}.
```

Een van de twee voorgaande manieren is het filteren van de informatie door een bestaande JSON-extensie zoals "JSONTOOLS" of het gebruik van filters in App Inventor in de tekstverwerking om alleen het uur, de dag of de seonden te verkrijgen, afhankelijk van de behoeftte van elk systeem. Na verwerking van het resultaat kan een logische vergelijking worden gemaakt en op basis van die vergelijking kunnen we een reeds geprogrammeerde taak uitvoeren met de "cron"-service die we later in elk knooppunt zullen zien.

JSONTOOLS uitbreidingsreferentie:

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

We hebben nu twee opties bekeken om de tijd van de knooppunten te synchroniseren. We gaan door met het configureren van de "cron" dienst op elk knooppunt.

Configuratie van geautomatiseerde taakplanning voor uitvoering met de **CRON-service** op Android-systemen (systeemknooppunten).

Eerst moeten we begrijpen hoe de automatische planner werkt.

De cron service heeft normaal gesproken alle systemen dit voorgeïnstalleerd en waar alle automatisch uit te voeren taken zijn ingepland staan in een bestand genaamd crontab.

We bewerken crontab-bestand met **\$ crontab -e**, we gebruiken **vi** editor, binnenin gebruiken we het formaat:

```
# m h dom mon dow gebruikerscommando
```

waar:

- **m** komt overeen met de minuut waarin het script wordt uitgevoerd, de waarde gaat van 0 tot 59
- **h de** exacte tijd, het formaat is 24 uur, de waarden gaan van 0 tot 23, zijnde 0 12:00 uur middernacht.
- **dom** verwijst naar de dag van de maand, bijv. u kunt er 15 opgeven als u elke 15e wilt lopen.
- **Downton** betekent de dag van de week, het kan numeriek zijn (0 tot 7, waarbij 0 en 7 zondag zijn) of de eerste 3 letters van de dag in het Engels: mon, tue, wed, thu, fri, sat, sun.
- **gebruiker** definieert de gebruiker die het commando zal uitvoeren, het kan root zijn, of een andere gebruiker zolang hij rechten heeft om het script uit te voeren.
- **commando** verwijst naar het commando of het absolute pad van het uit te voeren script, bijvoorbeeld: /home/user/scripts/update.sh, als u een script aanroeft moet het uitvoerbaar zijn

Om het duidelijk te maken een paar voorbeelden van cron taken uitgelegd:

```
15 10 * * * gebruiker /home/gebruiker/scripts/update.sh
```

Je draait het update.sh script elke dag om 10:15 uur.

```
15 22 * * * gebruiker /home/gebruiker/scripts/update.sh
```

Je draait het update.sh script elke dag om 22.15 uur.

```
00 10 * * 0 root apt-get - en update User root
```

Je zal elke zondag om 10:00 uur een update uitvoeren.

```
45 10 * * zonwortel apt-get - en update
```

Wortelgebruiker zal elke zondag (zon) om 10:45 uur een update uitvoeren.

We hebben de wijzigingen in de editor opgeslagen en daarmee de configuratie van de cron service afgerond. In het geval dat u de cron niet in het systeem hebt geïnstalleerd, kunt u dit doen met het volgende commando:

```
$ apt installeer cron
```

2. Installatie en configuratie van netwerkknooppunten - Mobiele telefoons.

Laten we beginnen met het communicatienetwerk voor knooppunten die gebruik zullen maken van Mini BlocklyChain.

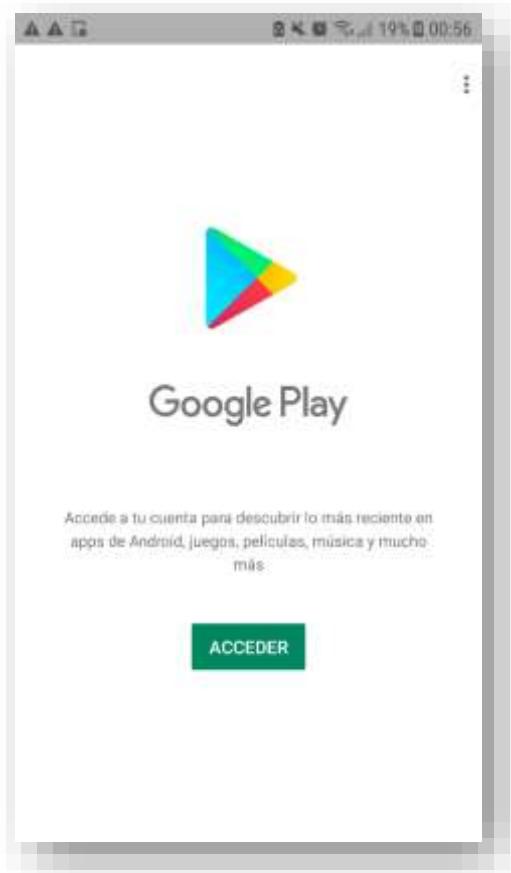
Eerst hebben we een Linux-omgeving nodig omdat elk Android-systeem gebaseerd is op Linux voor de veiligheid en flexibiliteit van de tools, we zullen de "Termux"-terminal gebruiken die de omgeving bevat waar we het communicatienetwerk zullen installeren.

Termux is een Linux emulator waar we de nodige pakketten zullen installeren om ons communicatienetwerk tussen knooppunten te creëren.

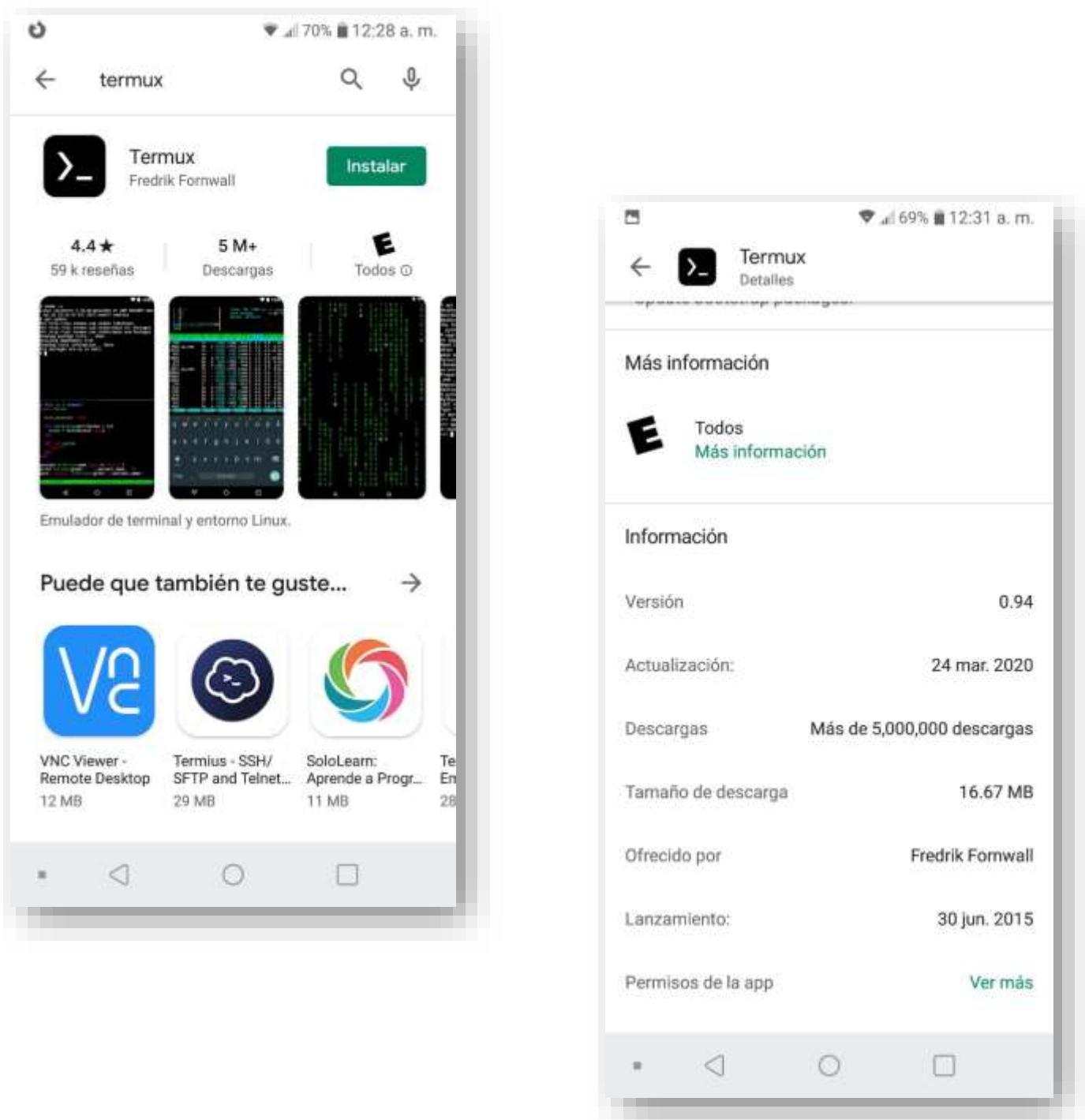
Een van de belangrijkste voordelen van het gebruik van Termux is dat u programma's kunt installeren zonder dat u de mobiele telefoon (Smartphone) hoeft te "draaien". Dit zorgt ervoor dat er geen fabrieksgarantie verloren gaat door deze installatie.

Termux-installatie.

Ga vanaf uw mobiele telefoon naar de Google Play-icoontoepassing (play.google.com).



Zoek op toepassing "Termux", selecteer deze en start het installatieproces.



Start van de Termux-toepassing.

Na het starten moeten we de volgende twee commando's uitvoeren om updates van de Linux-besturingssysteem-emulator uit te voeren:

\$ apt update

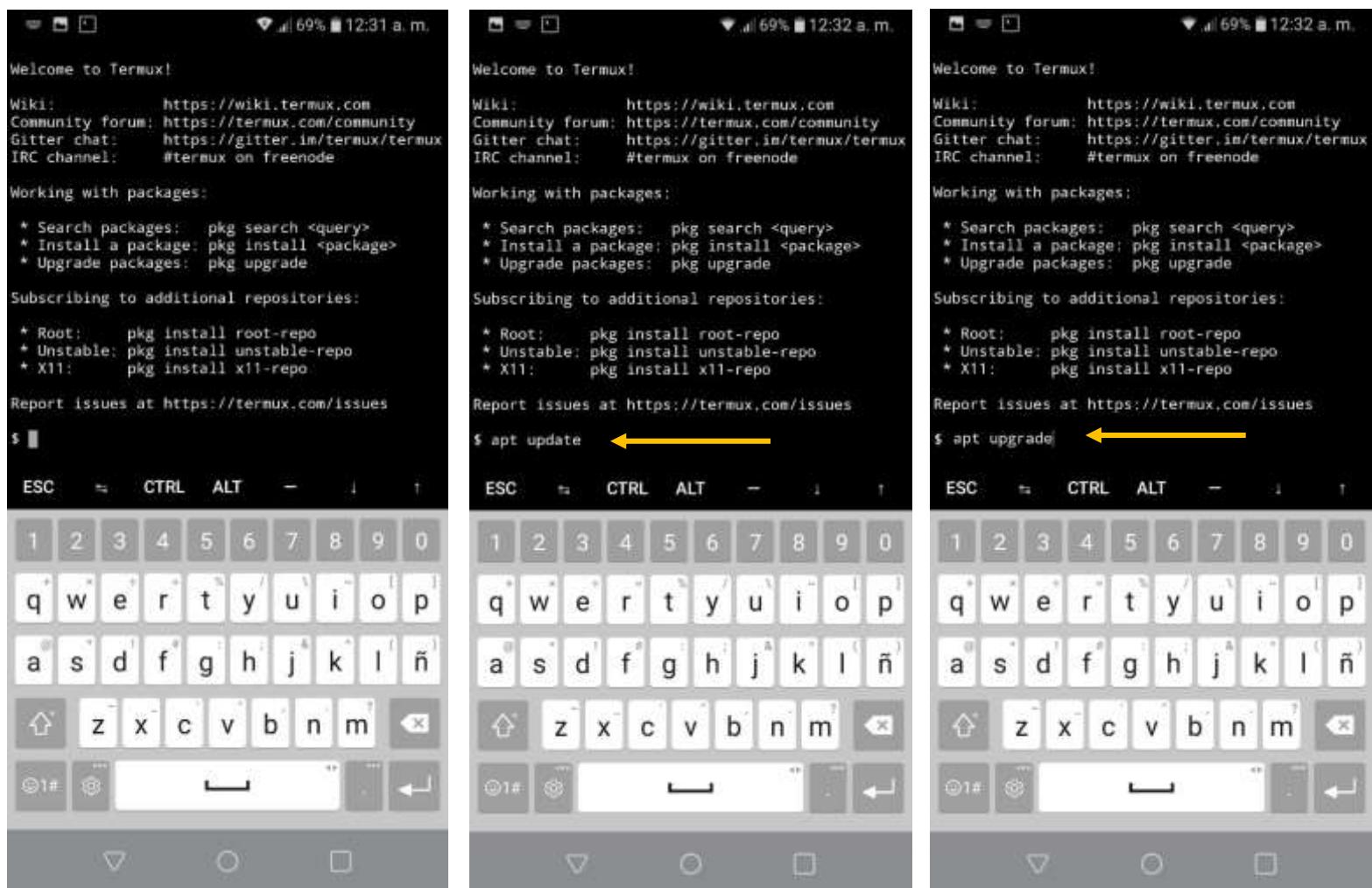
\$ apt upgrade

Bevestig alle opties Y(Ja)...

Termux

Home \$ apt update

\$ apt upgrade



11. Opslagconfiguratie binnen Termux.

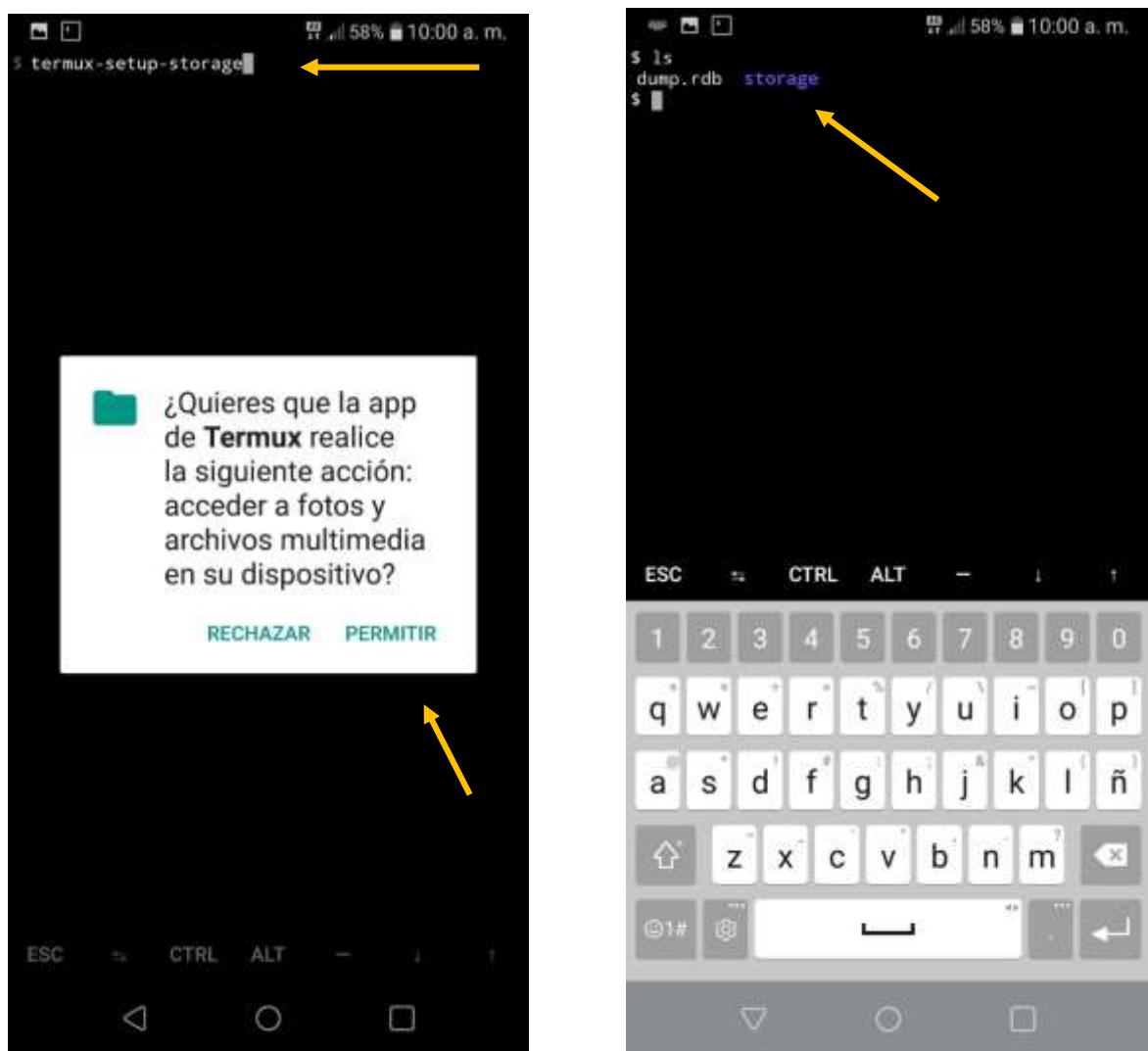
Nadat u het Termux systeem heeft bijgewerkt en geüpgraded, zullen we beginnen met het configureren van de interne opslag van de telefoon in het Termux systeem. Dit zal u helpen om informatie uit te wisselen tussen Termux en onze informatie in de telefoon.

Dit kan eenvoudig en snel worden gedaan door het volgende commando uit te voeren op een Termux-terminal.

`$ termux-setup-storage`

Wanneer u het vorige commando uitvoert, verschijnt er een venster dat u vraagt om de creatie van een virtuele **opslag** (directory) in Termux te bevestigen. We controleren het door het commando te geven:

`$ ls`



12. "Tor" netwerkinstallatie en "Syncthing" installatie.

\$ apt installeer tor

\$ apt installen syncthing

Accepteer de installatie door het invoeren van een hoofdletter Y in beide gevallen indien gewenst...

\$ apt installeer tor

```
68% 12:35 a.m.  
$ 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 installeer syncthing

```
68% 12:36 a.m.  
$ 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-pac  
ges-24/stable/main arm syncthing arm 1.5.0 [6407  
kB]  
27% [1 syncthing 2193 kB/6407 kB 34%]
```



13. Installatie van "Redis" database en SSH (Secure Shell) server.

\$ apt installeer redis

\$ apt installeer openssh

\$ apt installeer sshpass

\$ apt installeer 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 installeer 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-arm 1.18.1 [839 kB]
24% [2 krb5-131 KB/839 KB 16%]
```

\$ apt installeer 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]
```

We zijn klaar met de installatie van het communicatienetwerk, we gaan verder met de configuratie van de pakketten: Tor, Syncthing en Redis DB.

14. SSH-serverconfiguratie op mobiele telefoon (smartphone).

We zullen de SSH-server in de mobiele telefoon in staat stellen om verbinding te maken van onze PC naar de mobiele telefoon en om op een snellere en comfortabelere manier te werken, ook zal het dienen om te controleren of de dienst van de SSH-server in de mobiele telefoon correct werkt, omdat we het zullen gebruiken in het communicatiennetwerk in de Mini BlocklyChain.

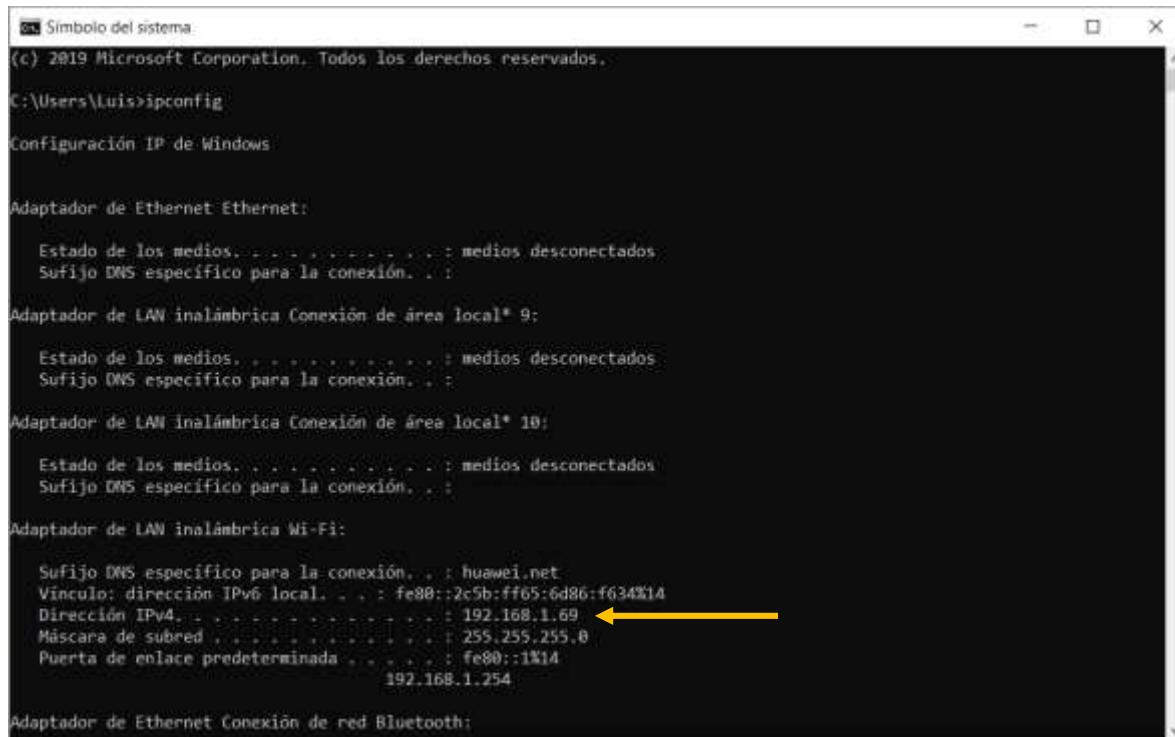
Het eerste wat we moeten doen is de mobiele telefoon en de PC op **hetzelfde WiFi-netwerk** aansluiten, zodat ze elkaar kunnen zien. De IP's of adressen moeten vergelijkbaar zijn met 192.168.XXX.XXX de XXX-waarden zijn variabele getallen die in elke computer willekeurig worden toegewezen.

Dit voorbeeld is getest op een LG Q6 mobiele telefoon en een PC met Windows 10 Home.

Controleer het IP of het adres dat de PC met de WiFi is verbonden we moeten een terminal openen in Windows.

In het onderste paneel waar de zoekloep staat, schrijft u cmd en drukt u op de Enter-toets. Er gaat een terminal open en daarin schrijven we het commando:

C:\User_Name> 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
  Vinculo: 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:

```

Het zal ons laten zien dat het IP dat aan de PC is toegewezen 192.168.1.69 is, maar dit zal waarschijnlijk in elk geval anders zijn.

OPMERKING: Het adres waar het "IPv4-adres" staat moet worden genomen, niet te verwarren met de Gateway.

Nu in het geval van de mobiele telefoon in de Termux terminal moeten we het volgende commando typen om de naam van onze gebruiker te kennen die we zullen gebruiken om verbinding te maken met de SSH-server die onze telefoon heeft, voeren we het volgende commando uit:

\$ whoami

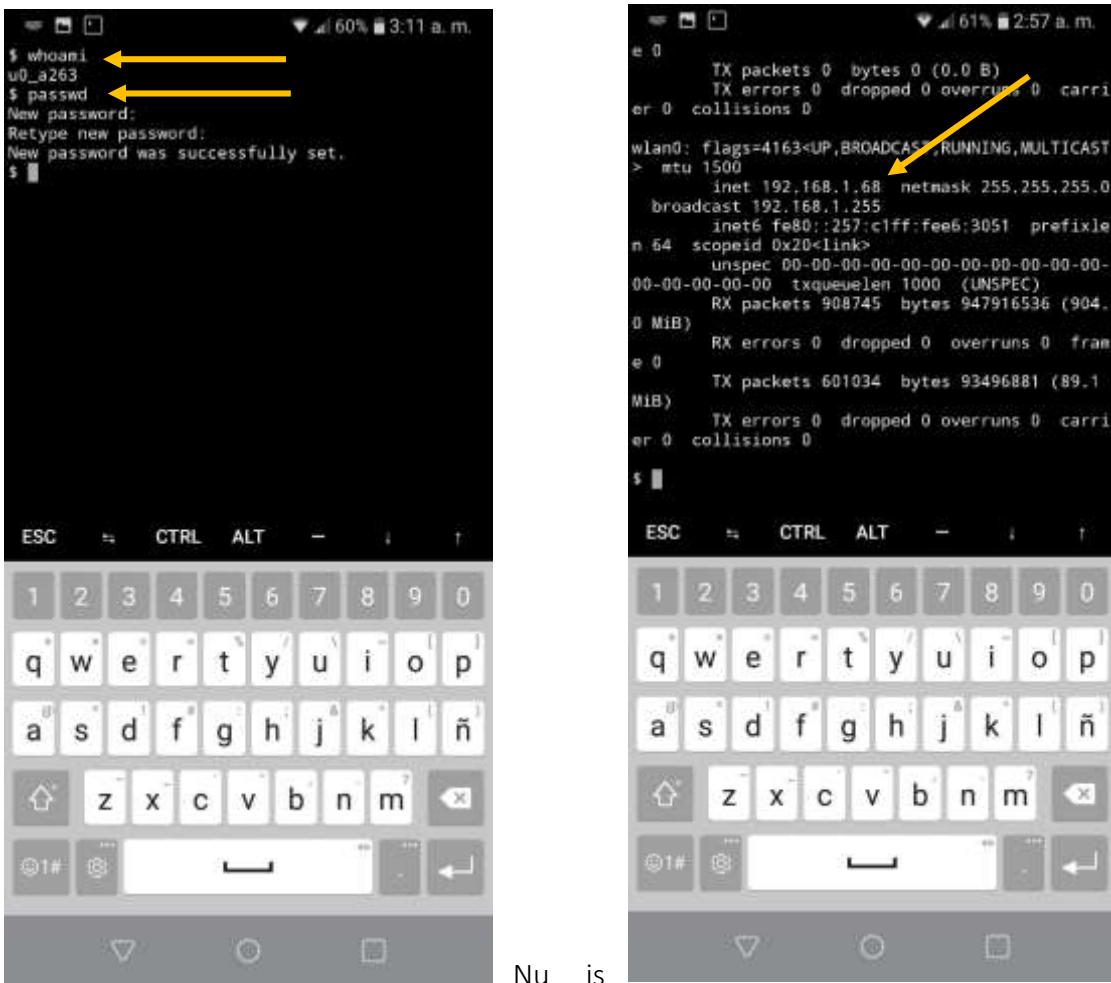
Later moeten we deze gebruiker een wachtwoord geven zodat we de volgende opdracht moeten uitvoeren:

\$ passwd

Het zal ons vragen om een wachtwoord te typen en op Enter te drukken, opnieuw vraagt het ons om het wachtwoord dat we bevestigen en op Enter drukken, als het **met succes "Nieuw wachtwoord is met succes ingesteld"** in **het** geval van het markeren van een fout is het mogelijk dat het wachtwoord niet correct is getypt. Voer de procedure opnieuw uit.

En om te weten welke IP we in Termux hebben typen we het volgende commando, het IP staat achter het woord "**inet**":

\$ ifconfig -a



het tijd om de SSH-serverdienst op uw telefoon te starten, zodat u de sessies vanaf uw PC kunt ontvangen. We voeren het volgende commando uit in de Termux terminal, dit commando geeft geen resultaat.

`$ sshd`



Nu zullen we een programma moeten installeren op de PC dat zal communiceren met de SSH-server van de telefoon vanaf de PC.

We moeten naar <https://www.putty.org> gaan.

Selecteer waar de link "U kunt PuTTY hier downloaden" is



Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. It is released under the MIT License, 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 recommendations.



Bitvise SSH Client

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

- graphical SFTP file transfer;
- single-click Remote Desktop tunnelling;
- 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](#).

Kies de 32-bits versie, het maakt niet uit of uw systeem 64-bits is zal prima werken.

Download PuTTY: latest release

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirrors](#)
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. See 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')

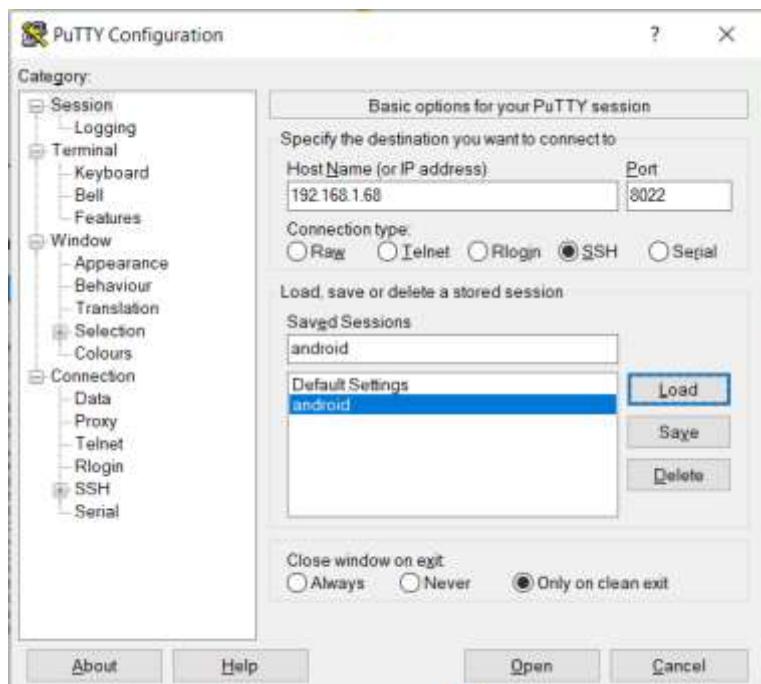
32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

Unix source archive

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

Zodra het is gedownload naar uw PC, voert u het uit en installeert u het met de standaardopties. Start dan de PuTTY toepassing.

In deze sessie voeren we de gegevens in van onze Openssh-server die we in de mobiele telefoon hebben geïnstalleerd.



Opslaan.

Voer het IP van de mobiele telefoon in.

HostName of IP-adres:

192.168.1.68 (IP voorbeeld)

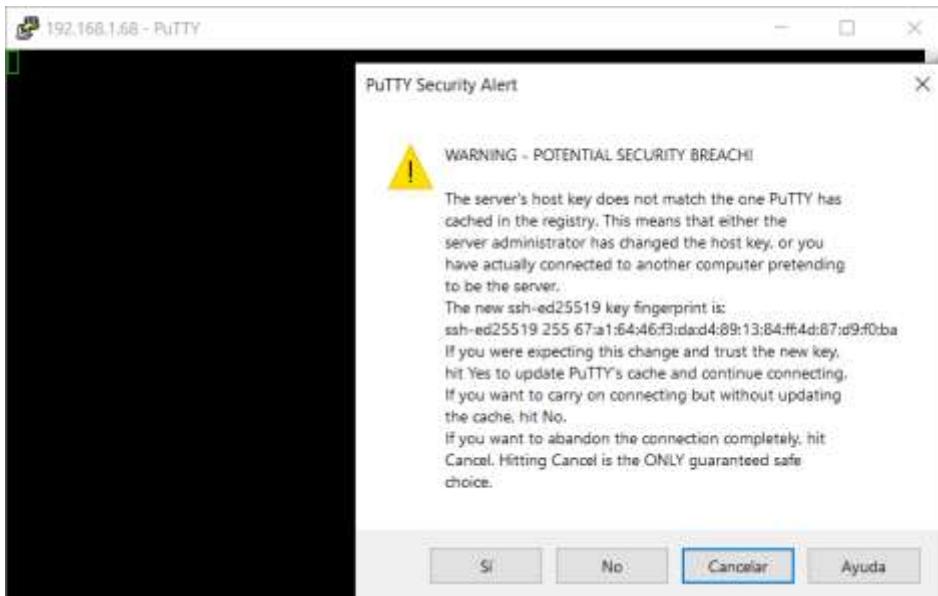
Haven:

8022 (Standaardpoort van de mobiele SSH-server).

We kunnen een naam geven aan de sessie in "Opgeslagen Sessies" en klikken op de knop Opslaan.

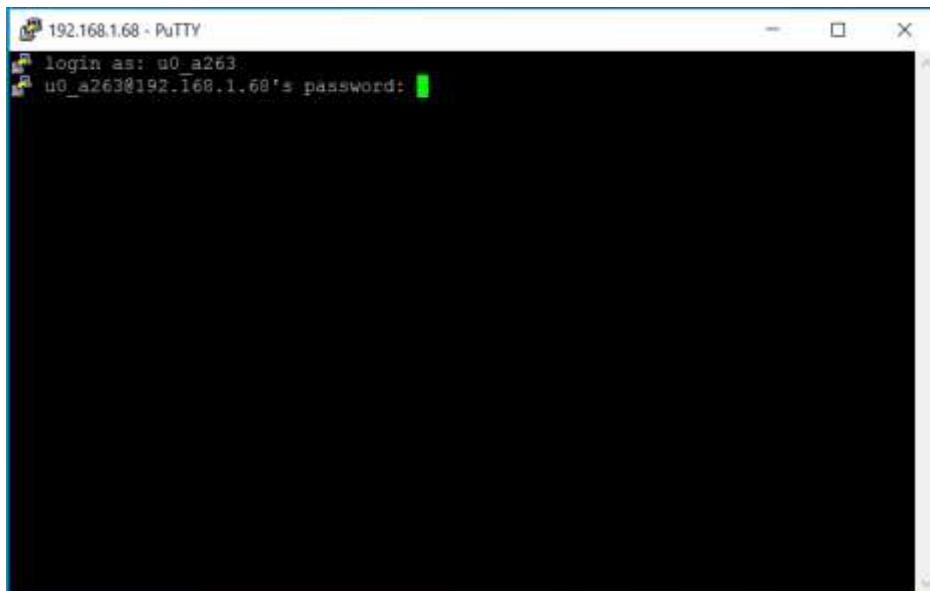
Later in het onderste gedeelte drukken we op om een verbinding met de server te openen met de knop "Openen".

Op de PC wordt u bij de eerste verbinding gevraagd om bevestiging van de informatiecoderingsleutel door op de "Ja"-knop te klikken.

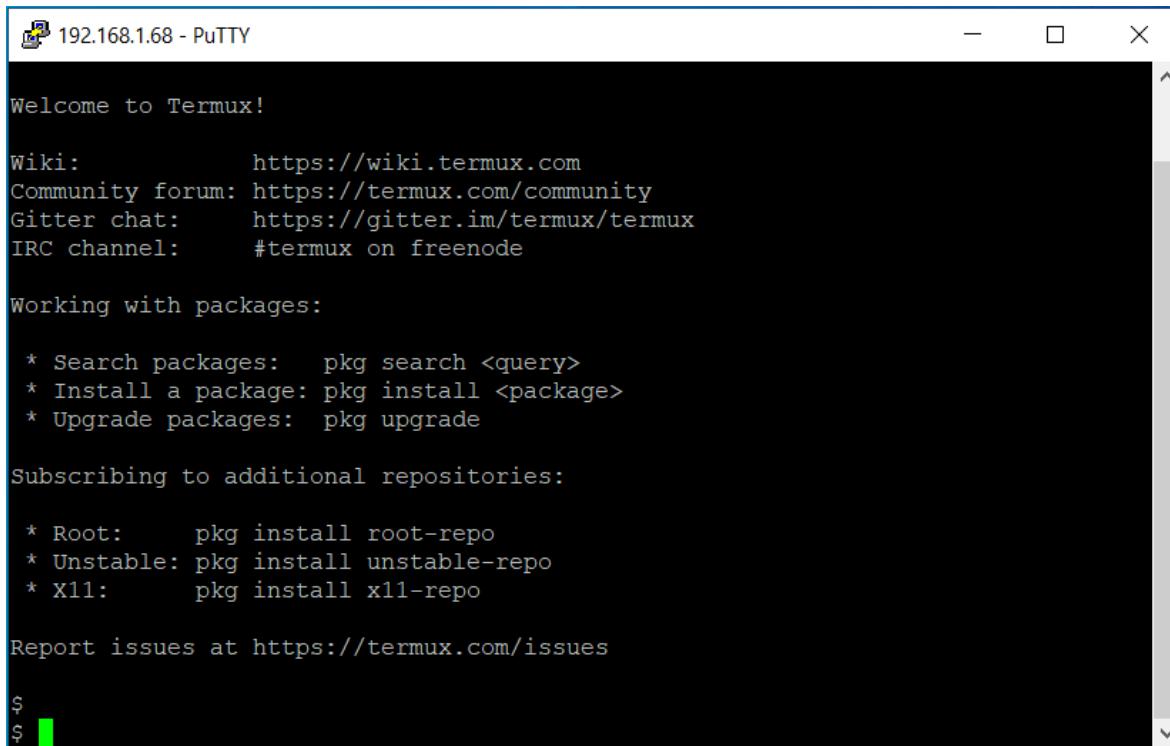


Later zullen we gevraagd worden naar de gebruiker waarmee we verbinding gaan maken. We gebruiken de informatie die we eerder hebben verkregen (gebruiker en wachtwoord).

In de **Login als:** we moeten onze gebruiker in te voeren en geef Enter, dan zullen we vragen om het wachtwoord opnieuw geef de Enter-knop.



Als de gegevens correct waren, zullen we in een SSH (Secure Shell) sessie worden uitgevoerd vanaf de PC (Client) op de telefoon (SSH Server).



```
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

$
```

BELANGRIJKE OPMERKING: Vergeet niet dat het IP (adres) van de PC en het IP (adres) van de mobiele telefoon die verbonden is met dezelfde WiFi waarschijnlijk zal veranderen elke keer dat we de verbinding verbreken en opnieuw verbinden, dus we moeten dubbel controleren welke adressen elk apparaat heeft, dit zal het succes van de verbinding tussen de apparaten via de SSH-server van de telefoon en de PC (Client) garanderen.

Tot nu toe hebben we alleen verbinding kunnen maken met hetzelfde WiFi-netwerk, maar als we onze telefoon buiten hetzelfde netwerk als de PC plaatsen, kunnen we geen verbinding maken omdat er verschillende netwerken betrokken zijn bij het doorlopen van andere, meer gecompliceerde communicatieapparatuur. We zullen dit oplossen als we het "Tor"-netwerk opzetten.

Vergeet niet dat deze verbinding alleen wordt gemaakt om de service van de server die we hebben geïnstalleerd op de telefoon te controleren en om een comfortabelere werkomgeving te hebben met een sessie op afstand van een PC naar de telefoon.

15.Tor" netwerkconfiguratie met SSH (Secure Shell) service.

Met de remote sessie vanaf de PC zullen we beginnen met het configureren van het "Tor" netwerk met de SSH service ingeschakeld.

Het belang van het "Tor"-netwerk is om de apparaten de eigenschap te geven dat ze overal ter wereld via het internet kunnen communiceren zonder in hetzelfde WiFi-netwerk te zitten, ongeacht waar we ons bevinden zullen we in staat zijn om verbinding te maken en het "Peer to Peer"-netwerk tussen knooppunten te vormen dat een essentiële functionaliteit is van de Mini BlocklyChain-architectuur.

Het "Tor" netwerk heeft veel flexibiliteit in zijn configuratie omdat het verschillende parameters in zijn configuratiebestand "torrc" heeft dit bestand staat in het pad (`$_PREFIX/etc/tor/torrc`) in ons geval met de Termux-sessie van onze PC zullen we weten door het volgende commando te typen:

```
$ echo $PREFIX
```

```
/data/data/com.termux/bestanden/usr
```

Daarom zal het bestand dat we moeten bewerken in het pad staan:

```
PREFIX/etc/tor/torrc die gelijk is aan /data/data/com.termux/bestanden/usr/etc/tor/torrc
```

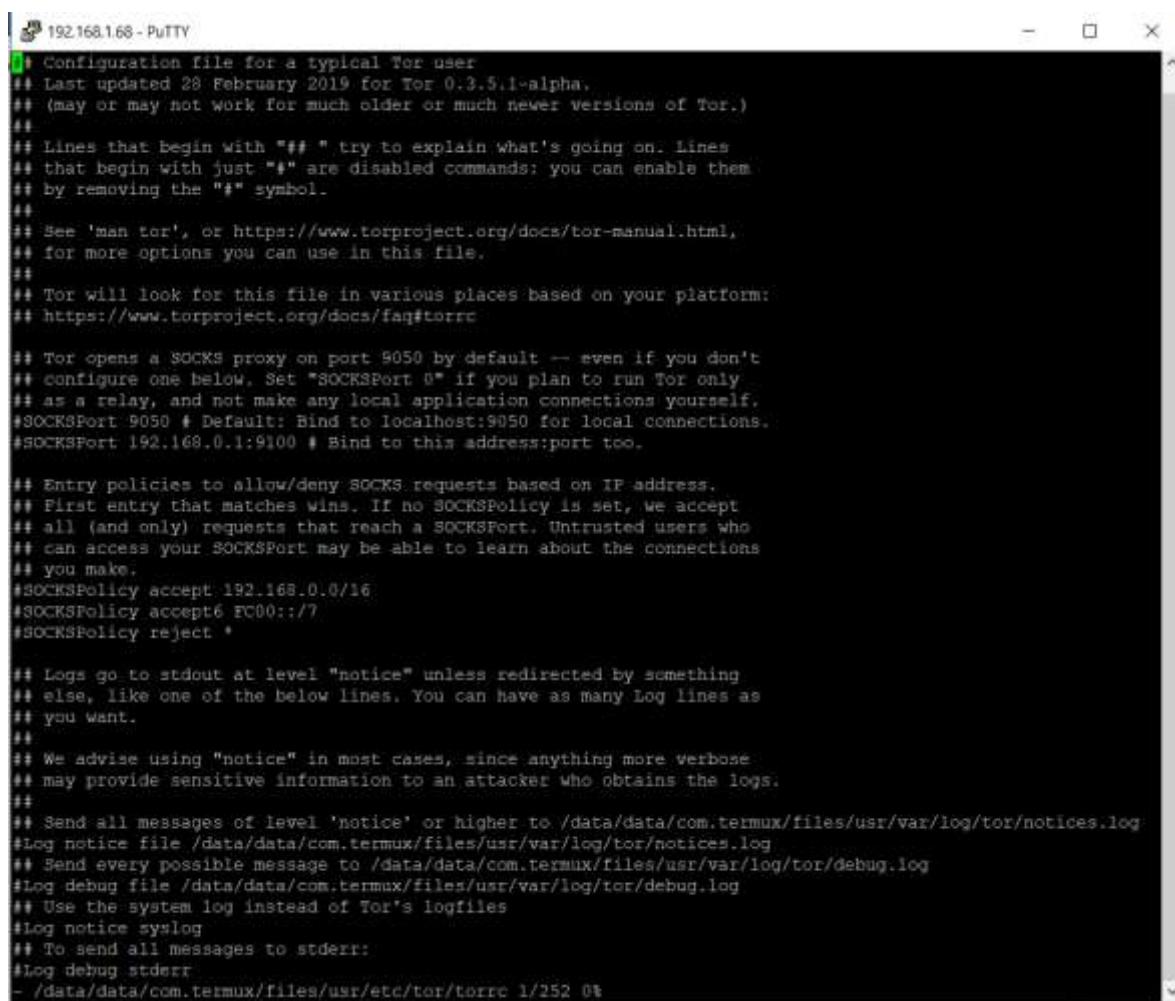
We gaan verder met het bewerken van het configuratiebestand met behulp van de commandoregel-editor "vi" door het volgende commando uit te voeren:

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

Hij zal dat bestand voor ons bewerken, als voorbeeld:



Bewerkt "torrc" bestand:



```

192.168.1.68 - PuTTY

# 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 0t

```

In dit "torrc" bestand zullen we de regels die het bestand heeft moeten toevoegen of gebruiken door de volgende wijzigingen aan te brengen, drie regels die de volgende zijn:

Syntax: **SOCKSPort <applicatie poortnummer>**

Voorbeeld: **SOCKSPort 9050**

De **SOCKSPort** variabele vertelt ons dat deze communicatie socket over het TCP-IP protocol standaard het mobiele apparaat (telefoon) zal gebruiken en poort 9050 zal worden geopend of in gebruik zijn.

Syntax: **HiddenServiceDir** <Directory waar de configuratie van de applicatie wordt opgeslagen>

Voorbeeld: **HiddenServiceDir /data/data/com.termux/bestanden/**

De **HiddenServiceDir** variabele vertelt ons dat dit de directory zal zijn waar de configuratie van de service die gebruikt zal worden via het Tor netwerk zal worden opgeslagen. In deze directory vind je het configuratie- en beveiligingsbestand voor de service, en in deze directory vind je een bestand met de **naam van de host**.

We kunnen het adres zien dat door het Tor netwerk is toegewezen voor de specifieke dienst die in ons geval wordt gecreëerd is het creëren van een SSH-dienst die zal worden geïmplementeerd op het Tor netwerk, de map die we als **hidden_ssh** noemen zal het **hostnaambestand** bevatten. Deze directory hoeft niet te worden aangemaakt, want als we de Tor netwerkdienst opstarten wordt deze automatisch aangemaakt.

Om het adres van het Tor netwerk te zien, kunnen we het commando "more" gebruiken. Voordat we dit commando gebruiken, moeten we het configureren van het "torrc" bestand afronden en de Tor netwerkdienst registreren, zodat de **hidden_ssh** directory en de bestanden daarin worden aangemaakt, zoals het geval is met het **hostname** bestand.

meer **/data/data/com.termux/bestanden/**

Het bovenstaande commando zal resulteren in een adres met een alfanumerieke tekenreeks met een extensie . ui, vergelijkbaar met :

uqwthf6ojdmoybyzvudoq3x2weq7k7rjitblhba3pjbawk2nvjmx3wer.onion

Als laatste moeten we de **HiddenServicePort** variabele toevoegen aan het "torrc" configuratiebestand. Deze parameter vertelt het Tor netwerk voor welke poort de applicatie waarvoor we ons aanmelden gebruikt zal worden over het Tor netwerk.

Syntax: **HiddenServicePort** <Haven van vertrek> <Lokaal of openbaar IP>: <Haven van vertrek>

O

VerborgenServicePort <Haven van vertrek>

Voorbeelden:

HiddenServicePort 22 127.0.0.1:8022

O

VerborgenServicePort 8022

Met het bovenstaande zijn we klaar met het configureren van het Tor netwerk voor generieke diensten en SSH (Secure Shell) service. Deze dienst zal worden gebruikt voor SQLite database update communicatie waar we de validatieprocessen van ons Mini BlocklyChain systeem zullen opslaan.

We gaan verder met de configuratie van het Mini BlocklyChain-communicatienetwerk.

16. Peer to Peer systeemconfiguratie met handmatige synchronisatie.

Een "Peer to Peer"-architectuur is fundamenteel in een blokketentechnologiesysteem omdat deze architectuur twee centrale punten in de systeemcommunicatieprocessen geeft, een ervan is om op elk moment dezelfde bijgewerkte (gesynchroniseerde) informatie te leveren in alle knooppunten, ongeacht of de knooppunten zich in een privénetwerk (Wifi) of een openbaar netwerk (internet) of een hybride van deze twee bevinden en een tweede punt van dit type architectuur is dat ze niet afhankelijk zijn van een tussenpersoon (server) om informatie tussen de knooppunten over te dragen, bij te werken of te raadplegen. Dit alles is te wijten aan het feit dat de communicatie rechtstreeks tussen de knooppunten gebeurt, in ons geval wordt de communicatie rechtstreeks tussen de telefoons die het netwerk vormen, zonder tussenkomst van een tussenpersoon, uitgevoerd.

Voor deze taak maken we gebruik van de open source Syncing tool die werkt op basis van een "Peer to Peer" architectuur.

De configuratie van Syncing voor apparaten (mobiele telefoons) wordt handmatig en automatisch bekeken. Het handmatige formulier wordt toegepast in synchronisatie met maximaal 5 knooppunten, in het geval dat het hebben van een groot aantal automatische configuratie is optimaal, het proces richt zich op de registratie van de knooppunten waarmee we de synchronisatie van de geselecteerde informatie uit te voeren.

De vereisten om te beginnen zijn:

1. Hebben de dienst van het Tor-netwerk geïnitieerd.
2. (optioneel - aanbevolen) Hebben geïnstalleerd een applicatie die kan lezen QR-codes, raden we de App inventor Android applicatie die gemakkelijk en eenvoudig te installeren van Google Play en later in deze handleiding zullen we het gebruiken in de sectie van "Definitie en het gebruik van blokken in Mini Blockchain".
3. Om de dienst van Syncing te hebben geïnitieerd.



We laten de App Inventor applicatie zien die we zullen gebruiken voor het lezen van QR-codes die ons in de toekomst zullen dienen voor de registratie van knooppunten in Syncthing.

Het volgende voorbeeld is gemaakt tussen twee mobiele apparaten (telefoons) met behulp van de volgende modellen:

Mobiel apparaat #1: Model LG Q6

Mobiel apparaat #2: Samsung-model

We beginnen met het starten van Tor-netwerkdiensten en synchrone diensten met behulp van de volgende commando's, openen twee terminals in Termux en voeren elk commando afzonderlijk uit:

\$ tor

Nadat je het opstartcommando van het Tor netwerkprogramma hebt getypt, moet je controleren of de uitvoering succesvol was, door te controleren of het 100% is uitgevoerd.

Als we het Tor-programma op een Termux-terminal laten draaien, controleren we of het op 100% draait.

\$ 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 htt
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 listen
er 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 23:00:38.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 23 23:00:39.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 23 23:00:39.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 23 23:00:39.000 [notice] Bootstrapped 20% (o
nehop_create): Establishing an encrypted directo
ry connection
May 23 23:00:39.000 [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 listen
er 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 ← → ↑ ↓

* ⌂ ○ □

Dan voeren we het synchrooncommando uit:

\$-synchronisatie

Na het uitvoeren van deze opdracht zal het een administratiepagina openen in de browser van onze telefoon als deze niet automatisch wordt geopend, we kunnen naar elke browser gaan die we hebben geïnstalleerd waar we normaal gesproken op het internet navigeren en we kunnen de volgende URL plaatsen:

<http://127.0.0.1:8384>

Het opent het administratiescherm van de tool die ons zal helpen om onze informatie te synchroniseren tussen alle knooppunten (telefoons) van het systeem.



```
$ 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-6RZI6S7-SK
V0U65-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...

```



Metric	Value
Velocidad de descarga	0 B/s (0 B)
Velocidad de subida	0 B/s (0 B)
Estado Local (Total)	0 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

Aangezien we het "syncthing" administratie scherm open hebben, gaan we over tot het registreren van het knooppunt of de knooppunten die we willen synchroniseren met de informatie. Op dit punt is waar we het programma dat QR-codes leest gaan bezetten.

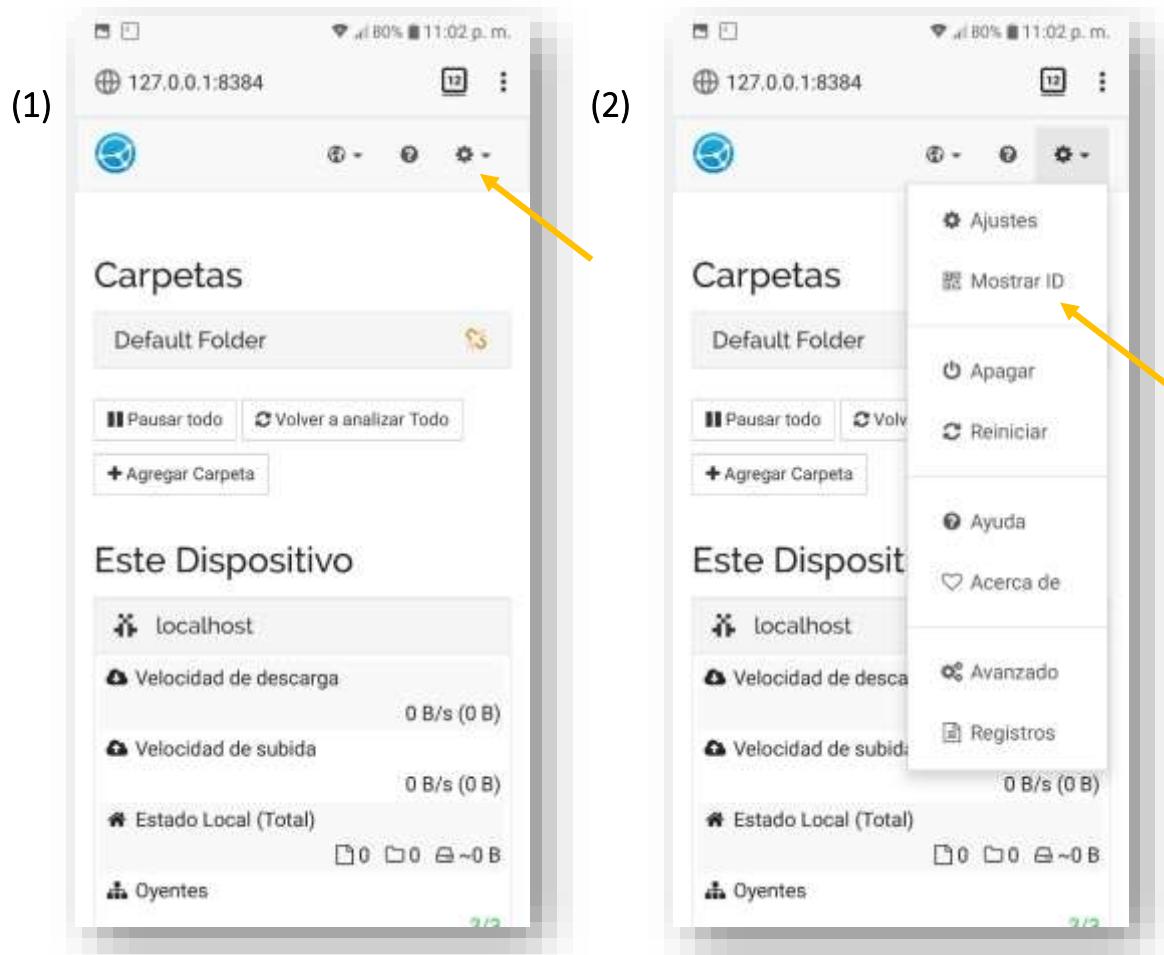
Het synchroonprogramma maakt bij de eerste start een unieke identifier van de telefoon die bestaat uit een groep van acht sets alfanumerieke tekens in hoofdletters, deze identifier (ID) is degene die we zullen registreren in het knooppunt of de knooppunten die we willen synchroniseren met de informatie.

In ons geval zal de LG Q6 telefoon-ID geregistreerd moeten worden op de Samsung telefoon en de Samsung telefoon-ID zal geregistreerd moeten worden op de LG Q6. Ze moeten in beide telefoons zitten om goed te kunnen werken.

We zullen de stappen van de Samsung mobiele telefoon registratie uitvoeren op de LG Q6 telefoon.

Eerst (1) bovenaan het administratiescherm (internetbrowser) van de Samsung-telefoon met synchrone weergave klikken we op het menutabblad rechtsboven.

Tweede (2) stap zien we een menu, hierin klikken we op "Toon ID" van de Samsung.

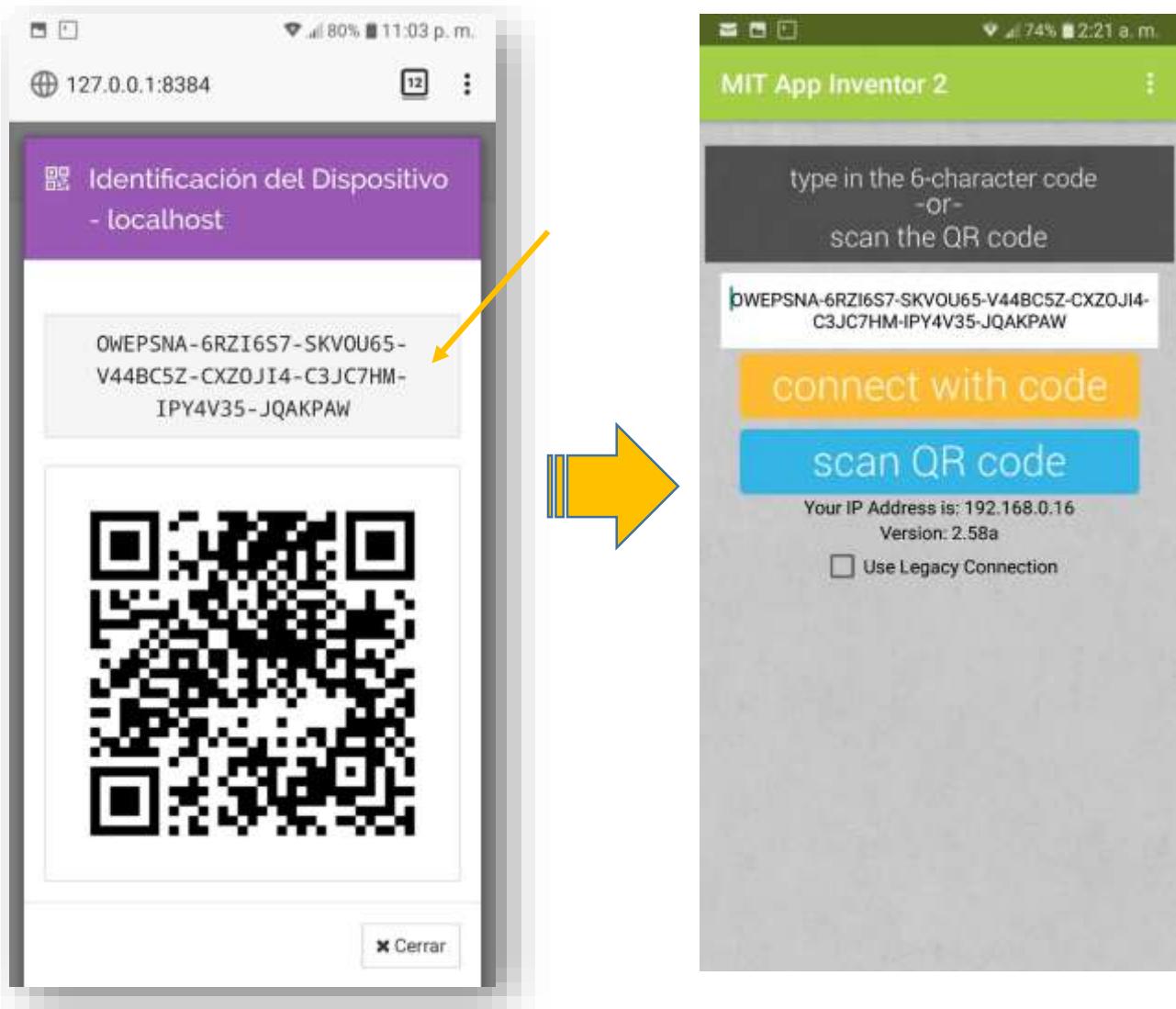


Wanneer u op "Toon ID" klikt, verschijnt het volgende scherm dat een QR-code is van de Samsung telefoon in ons geval de ID die de telefoon identificeert is

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

Dan in de LG Q6 telefoon, het programma dat ons zal helpen om deze Samsung's QR-code vast te leggen is degene die we voorstellen uit de App Inventor applicatie (optioneel), hoewel in het geval we het niet hebben kunnen we ook gewoon handmatig te introduceren wanneer we beginnen met de registratie in de LG Q6, echter, om te voorkomen dat elke fout die we voorstellen om het te gebruiken.

Met behulp van App programma uitvinder geïnstalleerd in de LG Q6 mobiele telefoon om QR-code vast te leggen van de externe Samsung telefoon die we willen synchroniseren.



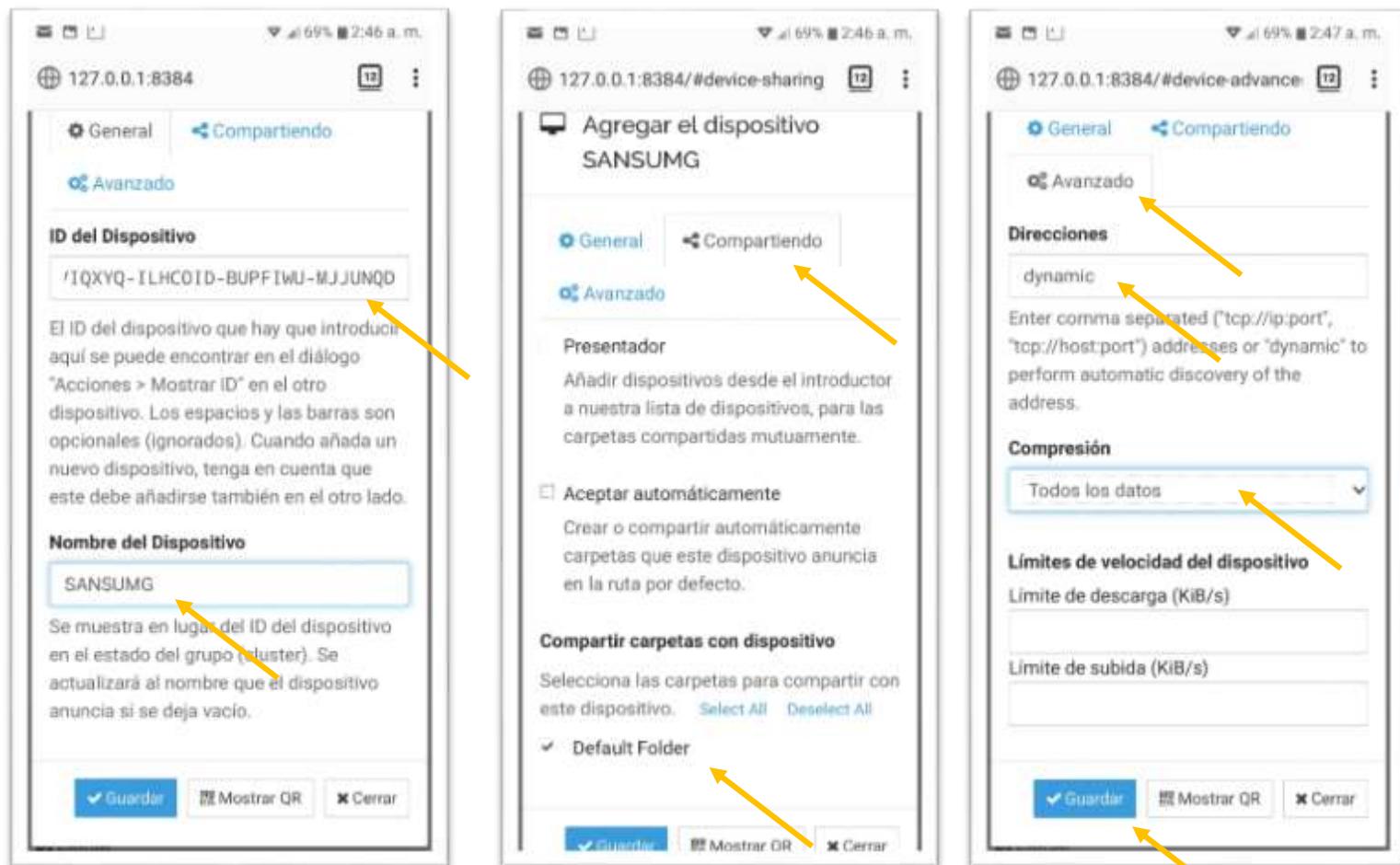
Vervolgens kopiëren we naar het klembord de QR code in het App inventor programma door op de vastgelegde code te klikken, we kiezen "SELECT ALL" → "COPY".

Met deze informatie gaan we over tot de registratie van de Samsung in de LG Q6. In het administratiescherm gaan we naar het onderste gedeelte waar "Andere apparaten" staat en klikken we op de knop "Voeg een apparaat toe", we introduceren de Samsung's ID en geven het een registratiernaam.

Vervolgens klikken we in het bovenste tabblad "Delen" en hiermee markeren we de onderste optie in de map "Default" en delen we een map die door Default is aangemaakt met de naam Sync in ons toestel.

Vervolgens klikken we bovenaan op het tabblad "Geavanceerd" en selecteren we onder "Alle gegevens" de optie gegevenscompressie.

Tot slot klikken we op de onderste bewaarknop, we eindigen de registratie in een knooppunt, ditzelfde proces moet worden gedaan in de externe telefoon of telefoons die we willen synchroniseren.



Bij het opslaan en registreren in beide telefoons zullen we een maximale periode van 30 seconden wachten waarin de apparaten zich bevinden en de verbinding tussen de apparaten zal als goed verschijnen met een bevestiging in het groen.

Apparaat #1 LG Q6



Apparaat #2 Sansumg



Otros dispositivos

Otros dispositivos

Alle informatie in de Sync-directory in het pad /data/data/com.termux/file/home/Sync wordt gesynchroniseerd, eventuele wijzigingen worden gekopieerd en gesynchroniseerd.

Peer to Peer systeemconfiguratie met Syncthing automatisch voor gebruik in Mini BlocklyChain.

Nu zullen we de configuratie op een automatische manier maken, eerder maakten we het handmatige proces dit is handig als we een minimum aantal nodes afhandelen, maar in het geval van een groot aantal nodes zou het inefficiënt zijn zodat we later de SyncthingManager tool hebben om het op een automatische manier te configureren met behulp van geautomatiseerde online commando's.

Redis databaseconfiguratie voor (Slave)-modusknooppunten.

Configuratie van het bestand `/data/data/com.termux/bestanden/usr/etc/redis.conf` van de Redis (**Slave**) database voor Android mobiele telefoons.

Voeg de volgende wijzigingen of richtlijnen toe aan het bestand, sla de wijzigingen op en start Redis server.

Zoek eerst het # teken en verwijder het van de **slavenlijn**. Deze richtlijn neemt het IP-adres en de poort die u gebruikt om veilig contact op te nemen met de Redis-masterserver, gescheiden door een spatie. Standaard luistert de Redis-server op 6379 op de lokale interface, maar elk van de netwerkbeveiligingsmethoden verandert de standaard op een bepaalde manier voor andere.

De waarden die u gebruikt zijn afhankelijk van de methode die u hebt gebruikt om uw netwerkverkeer te beschermen:

Geïsoleerd netwerk: gebruik het IP-adres van het geïsoleerde netwerk en de Redis-poort (6379) van de hoofdserver (bijv. slaaf van eenvoudig IP_adres 6379).

stunnel of spiped: gebruik de lokale interface (127.0.0.1) en de geconfigureerde poort om het verkeer te tunnelen

PeerVPN: Gebruik het IP VPN-adres van de hoofdserver en de normale Redis-poort.

De generaal zou het veranderen:

slaaf van ip_contact_server master_contact_port

Voorbeeld: **slaaf van** 192.168.1.69 6379

masterauth your_network_master_password

Voorbeeld: **masterauth** sdfssdfsdf12WqE34Rfgthtdfd

passeer uw_netwerk_slave_wachtwoord nodig hebben

Voorbeeld: **requirepass** asdsjdsh34sds67sdFGbbnh

Sla de server op en voer hem uit vanaf de Termux-terminal met het volgende commando.

\$ redis-server redis.conf

Na de uitvoering kunnen we zien hoe het synchroniseert met de Windows 10 server (Master).

Configuratiebestand 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 <-> REPLICAS 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: 8ea52fe3c02ae241292f0dcbb823b8febcb09b784:0
32672:S 31 May 2020 23:50:24.349 * MASTER <-> REPLICAS sync: receiving 578 bytes from master to disk
32672:S 31 May 2020 23:50:24.353 * MASTER <-> REPLICAS sync: Flushing old data
32672:S 31 May 2020 23:50:24.353 * MASTER <-> REPLICAS 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 <-> REPLICAS sync: Finished with success

```

Installatie van een synchrone beheertool voor knooppunten. We gaan over tot de installatie van **SyncthingManager**.

Eerst installeren we wat u nodig heeft om SyncthingManager goed te laten werken.

\$ apt installeer Python

\$ pip3 installeer -upgrade pip

\$ npm installen syncthingmanager

De SyncthingManager tool zal ons helpen om "peer to peer" te synchroniseren op een automatische manier en niet handmatig voor nieuwe en bestaande knooppunten.

```
$ apt install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (3.8.3),
0 upgraded, 0 newly installed, 0 to remove and 0
not upgraded.
$ 

$ pip3 install --upgrade pip
Requirement already up-to-date: pip in /data/dat
a/com.termux/files/usr/lib/python3.8/site-pac
kages (20.1.1)
$ 

$ pip3 install syncthingmanager
Requirement already satisfied: syncthingmanager
in /data/data/com.termux/files/usr/lib/python3.8/
site-packages (0.1.0)
Requirement already satisfied: syncthing in /dat
a/d/a/com.termux/files/usr/lib/python3.8/site-p
ackages (from syncthingmanager) (2.3.1)
Requirement already satisfied: python-dateutil==
2.6.1 in /data/d/a/com.termux/files/usr/lib/pyt
hon3.8/site-packages (from syncthing->syncthingm
anager) (2.6.1)
Requirement already satisfied: requests==2.18.4
in /data/d/a/com.termux/files/usr/lib/python3.8
/site-packages (from syncthing->syncthingmanag
er) (2.18.4)
Requirement already satisfied: six>=1.5 in /data
/d/a/com.termux/files/usr/lib/python3.8/site-pa
ckages (from python-dateutil==2.6.1->syncthing->
syncthingmanager) (1.15.0)
Requirement already satisfied: chardet<3.1.0,>=3
.0.2 in /data/d/a/com.termux/files/usr/lib/pyt
hon3.8/site-packages (from requests==2.18.4->sync
thing->syncthingmanager) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in
/d/a/d/a/com.termux/files/usr/lib/python3.8/si
te-packages (from requests==2.18.4->syncthing->
syncthingmanager) (2.6)
Requirement already satisfied: certifi>=2017.4.1
7 in /data/d/a/com.termux/files/usr/lib/python3
.8/site-packages (from requests==2.18.4->syncthi
ng->syncthingmanager) (2020.4.5.1)
Requirement already satisfied: urllib3<1.23,>=1
.21.1 in /data/d/a/com.termux/files/usr/lib/pyt
hon3.8/site-packages (from requests==2.18.4->sync
thing->syncthingmanager) (1.22)
$ 
```

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

App Inventor is een softwareontwikkelingsomgeving die door Google Labs is gecreëerd om applicaties voor het Android-besturingssysteem te bouwen. De gebruiker kan, visueel en vanuit een set van basistools, een reeks blokken aan elkaar koppelen om de applicatie te maken. Het systeem is gratis en kan eenvoudig worden gedownload van het web. Applicaties die met App Inventor worden gemaakt zijn zeer eenvoudig te maken, omdat er geen kennis van enige programmeertaal nodig is.

Alle huidige omgevingen die gebruik maken van Blockly's technologie zoals o.a. AppyBuilder en Thunkable hebben hun gratis versie, hun manier van gebruik kan via het internet op hun verschillende sites of het kan ook thuis worden geïnstalleerd.

De blokken waaruit de Mini BlocklyChain architectuur bestaat zijn getest in App Inventor en AppyBuilder, maar vanwege hun code-optimalisatie zouden ze op de andere platformen moeten werken.

Online versies:

App Inventor.

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

AppyBuilder.

<http://appybuilder.com/>

Denkbaar.

<https://thunkable.com/>

Versie die op uw computer (PC) moet worden geïnstalleerd:

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

Omgeving voor ontwikkelaars van Blockly-blokken.

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

18. Wat is het bewijs van Quantum (PQu)?

PoQu. - "Proof of Quantum" is een consensus algoritme ontwikkeld voor Mini BlocklyChain, deze test is een variant op de Test of Work (PoW) die als volgt werkt.

De Test of Quantum (PoQu) bij het opstarten wordt uitgevoerd met hetzelfde algoritme als de "Test of Work" (PoW) is gebaseerd op het zetten van de processor van het apparaat (PC, Server, Tablet or Mobile Phone) om te werken aan een reeks van karakters die een wiskundige puzzel is genaamd een "hash".

Vergeet niet dat een "hash" een algoritme of wiskundig proces is dat bij de introductie van een zin of een bepaald type digitale informatie zoals tekstbestanden, programma's, afbeeldingen, video, geluid of andere diverse soorten digitale informatie ons als resultaat een alfanumeriek karakter geeft dat de digitale handtekening vertegenwoordigt die het op een unieke en niet-reproduceerbare manier van de gegevens weergeeft, het hash-algoritme is unidirectioneel, dit betekent dat wanneer u een gegevens in te voeren om de handtekening "hash" zijn omgekeerde proces kan niet worden uitgevoerd, met een handtekening "hash" kunnen we niet weten welke informatie werd verkregen deze eigenschap geeft ons een veiligheid voordeel om de informatie die we sturen via het internet te verwerken. Hoe werkt het? Stel je voor dat het verzenden van elke vorm van informatie via niet-veilige kanalen en begeleiden met de respectieve "bron hash", de ontvanger bij de ontvangst van de informatie kan de "hash" van de ontvangen informatie zullen we het noemen "bestemming hash" en controleer het met de "bron hash" als beide "hashes" zijn hetzelfde kunnen we bevestigen dat de informatie niet is gewijzigd in het kanaal dat werd verzonden, is slechts een voorbeeld waar dit type van informatiebeveiliging proces wordt momenteel gebruikt.

Momenteel zijn er verschillende soorten algoritmen of hashprocessen die verschillen in het beveiligingsniveau. De meest gebruikte of bekende zijn: MD5, SHA256 en SHA512.

Voorbeeld van SHA256:

We hebben een ketting of zin als volgt: "Mini BlocklyChain is modular.

Als we een hash van het type SHA256 toepassen op de vorige string geeft dat ons de volgende hash.

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2afd827e8804db8

De bovenstaande alfanumerieke tekenreeks is de handtekening die de zin in het bovenstaande voorbeeld weergeeft

Zo kunnen we de site bijvoorbeeld ook op het internet gebruiken:

<https://emn178.github.io/online-tools/sha256.html>

In het geval van het "Testwerk" (PoW) algoritme werkt het door gebruik te maken van rekenkracht om een vooraf gedefinieerde hash te verkrijgen.

Laten we ons voorstellen dat we de vorige "hasj" hebben die we uit de "Mini BlocklyChain is modulair" keten hebben gehaald.

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2afd827e8804db8

Aan deze "hash" in het begin zetten we de parameter van de moeilijkheid die is gewoon om nullen "0" in het begin, dat wil zeggen als we zeggen dat de moeilijkheid is van 4 het zal hebben "**0000**" + "hash" om dit zullen we het noemen "zaad hash"

0000 f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8

Nu rekening houdend met het feit dat we de invoerinformatie kennen die de string is: "Mini BlocklyChain is modulair" voegen we aan het einde van de string een getal toe dat begint met nul "0" en we halen de hash eruit om het "hash nonce" te noemen:

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db80

We hebben hasj nonce:

7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8

Dan voeren we een vergelijking van de nieuwe "hash nonce" met de "hash seed" als ze gelijk zijn aan het knooppunt dat eerst de gelijkheid vindt zal de uitvoering van de verwerking van de huidige transactie te winnen. Zoals we kunnen zien is dit proces gebaseerd op de waarschijnlijkheid en de rekenkracht van het apparaat, wat de "Proof of Work" test een consensusgelijkheid geeft voor alle knooppunten.

Als de "hasj" niet samenvalt met de "hasj nonce", wordt de moeilijkheidsgraad met één verhoogd en wordt de "hasj nonce" weer verwijderd, het getal dat wordt verhoogd wordt het "nonce" getal genoemd, het wordt vergeleken met de "hasj" totdat ze samenvallen of gelijk zijn.

Zoals we kunnen zien is het getal "nonce" of verhoging het getal dat zal helpen om de "hasj" van gelijkheid te verkrijgen.

Gebaseerd op het "Test of Work" (PoW) algoritme, het Test of Quantum (PoQu) algoritme is gebaseerd op het verkrijgen van het getal "nonce" zoals PoW dat doet en met behulp van een minimum moeilijkheidsgraad variërend van 1 tot 5, dit dient alleen voor het mobiele apparaat om het recht te krijgen om een kandidaat te zijn om de consensus te winnen.

De Quantum Test (PoQu), wordt geactiveerd wanneer de mobiele telefoon de minimale PoW heeft voltooid en de pass wint om een waarschijnlijksnummer in het QRNG-systeem te verkrijgen.

De QRNG (Quantum Random Number Generator) is een Quantum Random Number Generator, dit systeem is gebaseerd op het genereren van echte willekeurige getallen op basis van kwantummechanica is vandaag de dag het veiligste systeem om dergelijke getallen te genereren. Voor meer details zie de bijlage "Kwantumberekening met OpenQbit".

Mini BlocklyChain kan zowel minimale PoW als PoQu concessie types implementeren.

De PoQu test is gebaseerd op het verkrijgen van het nummer "nonce" dit nummer in de PoQu test staat bekend als "Magic Number" waarbij het "Peer to Peer" systeem zal bevestigen of het nummer correct is en vervolgens zal een willekeurig nummer worden verkregen met de QRNG server pool. Dit willekeurige getal wordt in alle knooppunten geregistreerd, er wordt een lijst gemaakt met **((Node Sum /2)) +1** en uit deze lijst wordt degene gekozen met het hoogste waarschijnlijkspercentage om de winnaar van de consensus (PoQu) te zijn en deze zal de huidige transactiewachtrij uitvoeren.

Het PoQu algoritme maakt ook gebruik van **NIST** (National Institute of Standards and Technology) testen om ons te verzekeren dat de willekeurige getallen in de QRNG echt willekeurige getallen zijn.

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

In Mini BlocklyChain hebben we een blok voor PoW en een blok voor PoQu geïmplementeerd.

Deze blokken gebruiken een type hasj: SHA256 voor vrij gebruik, voor commercieel gebruik heb je een SHA512 en andere soorten hasj zoals vereist.

Voor meer details over het concept HASH zie:

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

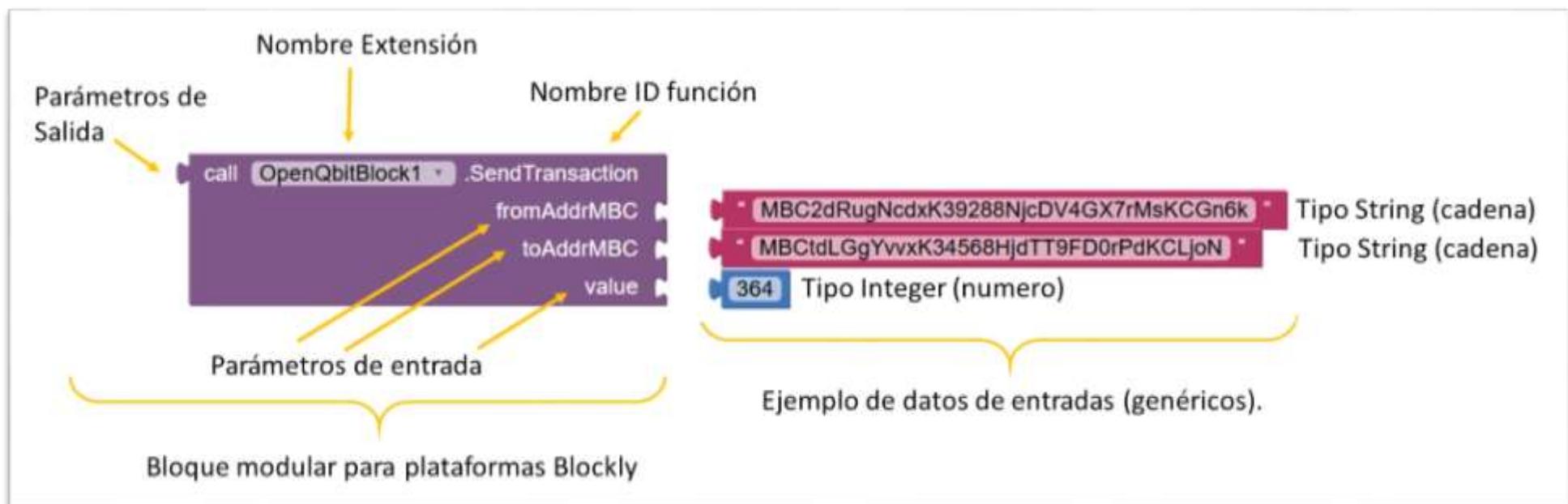
OPMERKING: De Test of Work (PoW) die in mobiele telefoons wordt gebruikt, kan slechts een maximale moeilijkheidsgraad van 5 gebruiken, aangezien de wiskundige verwerking van deze apparaten niet is toegewijd zoals servers of pc's. We gebruiken het PoW-algoritme alleen om uw pasje of toestemming te krijgen om het Quantum Random Number Generator (QRNG)-systeem in te voeren en daarmee het Quantum Random Number Generator (PoQu)-algoritme uit te voeren.

Gebruik op mobiele telefoons geen maximale moeilijkheidsgraad van 5, omdat het systeem kan vastlopen en niet goed reageert.

19. Definición y uso de bloques en Mini BlocklyChain

We beginnen met het uitleggen van de verdeling van de gegevens die alle blokken zullen hebben, hun syntax van gebruik en configuratie.

In het volgende voorbeeld zien we een modulair blok met zijn invoer- en uitvoerparameters en de soorten invoergegevens, deze gegevens kunnen van het type String (tekenreeks) of Integer (geheel getal of decimaal) zijn. We laten zien hoe het wordt gebruikt en configureren het voor de goede werking ervan.



Elk moduleblok zal zijn beschrijving hebben en zal worden benoemd in het geval dat het een verplichte of optionele afhankelijkheid heeft van andere blokken die worden gebruikt als invoerparameters, het integratieproces zal worden aangekondigd.

Blokkeer om een tijdelijke stringlijst te maken in een voorgedefinieerde array die intern "keten" wordt genoemd - (**AddHash**).



Invoerparameters: **blok** <string>

Uitgangsparameters: Niet van toepassing.

Beschrijving: Blok voor het opslaan van een tijdelijke reeks hashes of strings in een vooraf gedefinieerde reeks die intern "keten" wordt genoemd.

Blokkeren om key-value arrays (**Integer-Integer**) te creëren - (**AddKeyValueArrayItoi**)



Invoerparameters: **bcKey** <Integer>, **bcValue** <Integer>

Uitgangsparameters: Geeft de bij de ingang ingevoerde waarden terug.

Beschrijving: Het is een tijdelijke sleutelwaarde regeling, het is vooraf gedefinieerd met interne naam "**Itoi_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokkeren om key-value arrays te creëren (**Integer-String**) - (**AddKeyValueArrayItoS**)

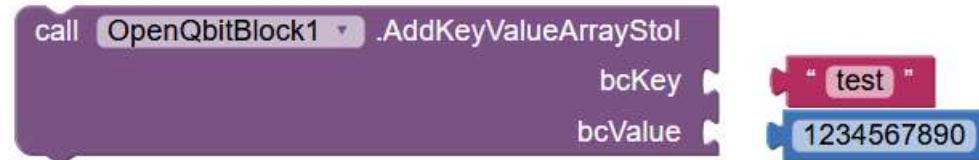


Invoerparameters: **bcKey** <Integer>, **bcValue** < String>

Uitgangsparameters: Geeft de bij de ingang ingevoerde waarden terug.

Beschrijving: Het is een tijdelijke sleutelwaarde regeling, het is vooraf gedefinieerd met interne naam "**ItoS_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokken om key-value arrays (**String-Integer**) te creëren - (**AddKeyValueArrayStol**)



Invoerparameters: **bcKey <String>**, **bcValue <Integer>**

Uitgangsparameters: Geeft de bij de ingang ingevoerde waarden terug.

Beschrijving: Het is een tijdelijke sleutelwaarde regeling, het is vooraf gedefinieerd met interne naam "**Stol_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokken om key-value arrays (**String-String**) te creëren - (**AddKeyValueArrayStoS**)

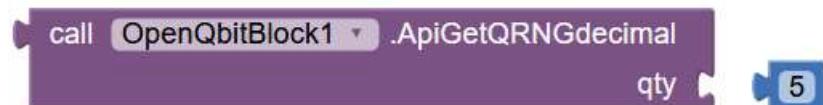


Invoerparameters: **bcKey <String>**, **bcValue <String>**

Uitgangsparameters: Geeft de bij de ingang ingevoerde waarden terug.

Beschrijving: Het is een tijdelijke sleutelwaarde regeling, het is vooraf gedefinieerd met interne naam "**StoS_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blok voor het genereren van decimale willekeurige kwantumcijfers - (**ApiGetQRNGdecimaal**)



Invoerparameters: **qty <Integer>**

Uitgangsparameters: Geeft de hoeveelheid "qty" van willekeurige kwantumdecimale getallen die zijn ingevoerd in de invoeraantallen liggen binnen het bereik van 0 en 1 in JSON-formaat.

Voorbeeld:

qty = 5; uitgang: {"resultaat": [0,5843012986202495, 0,7746497687824652, 0,05951126805960929, 0,1986079055812694, 0,03689783439899279]}.

Beschrijving: Quantum Random Number Generator (QRNG) API

Blok voor het genereren van decimale willekeurige kwantumcijfers - (**ApiGetQRNGdecimaal**)



Invoerparameters: **qty <Integer>**, min <Integer>, max <max>

Uitgangsparameters: Geeft de hoeveelheid "qty" aan willekeurige kwantumgehelen die in de invoer zijn ingevoerd de getallen liggen binnen het bereik van min en max in JSON-formaat.

Voorbeeld:

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

Beschrijving: Quantum Random Number Generator (QRNG) API

Hashblok "SHA256" voor tekenreeksen (**ApplySha256**).



Invoerparameters: **invoer <String>**

Uitgangsparameters: Berekent de "SHA256" hash van een tekenreeks.

Voorbeeld:

Input = "Mini BlocklyChain is modulair".

uitgang: f41af7e61c3b02fdd5e5c612302b62a2dd52fc38f9de97cb2af827e8804db8

Beschrijving: Functie voor het verwijderen van hasj "SHA256". Sha of SHA verwijzen naar: Secure Hash Algorithm, een set hash-functies, ontworpen door het United States National Security Agency. De SHA256 maakt gebruik van een 256-bits algoritme.

Blokkeren om de vooraf gedefinieerde interne array "ketting" (**ClearBlockList**) te reinigen.

call OpenQbitBlock1 .ClearBlockList

Ingangs- en uitgangsparameters: Niet van toepassing

Beschrijving: Verwijder alle elementen die de interne tijdsindeling "keten" hebben.

Blokkeren om de vooraf gedefinieerde interne array (**Integer-Integer**) - (**ClearItoI**) te reinigen.

call OpenQbitBlock1 .ClearItoI

Ingangs- en uitgangsparameters: Niet van toepassing

Beschrijving: Verwijder alle elementen die de interne tijdelijke regeling "ItoI_UTXO" hebben.

Blokkeren om de vooraf gedefinieerde interne array (**Integer-String**) - (**ClearItoS**) te reinigen.

call OpenQbitBlock1 .ClearItoS

Ingangs- en uitgangsparameters: Niet van toepassing

Beschrijving: Verwijder alle elementen die de interne tijdelijke regeling "ItoS_UTXO" hebben.

Blokkeren om de vooraf gedefinieerde interne array (**String-Integer**) schoon te maken - (**ClearStoI**).

call OpenQbitBlock1 .ClearStoI

Ingangs- en uitgangsparameters: Niet van toepassing

Beschrijving: Verwijder alle elementen die de interne tijdelijke regeling "StoI_UTXO" hebben.

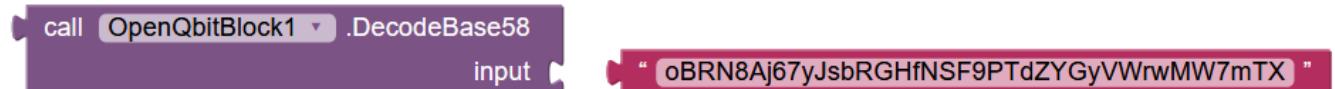
Blokkeren om de vooraf gedefinieerde interne array (**String-String**) schoon te maken - (**ClearStoS**).

call OpenQbitBlock1 .ClearStoS

Ingangs- en uitgangsparameters: Niet van toepassing

Beschrijving: Verwijder alle elementen die de interne tijdelijke regeling "StoS_UTXO" hebben.

Blokkeren om een string te decoderen naar Base10. (**DecodeBase58**)



Invoerparameters: **invoer<String>**

Uitgangsparameters: Het levert de originele string die in het blok werd gebruikt (**EncodeBase58**).

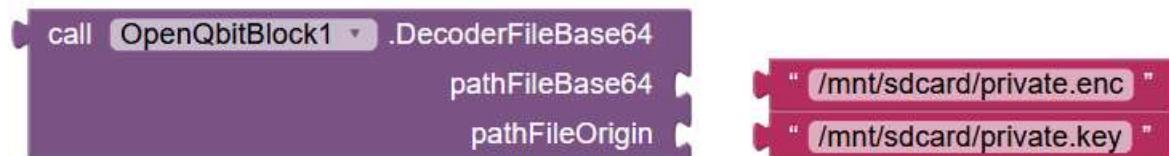
Voorbeeld:

Ingang = oBRN8Aj67enJsbRGHfNSF9PTdZYGandVWrwMW7mTX

Uitgang: "Mini BlocklyChain is modulair".

Omschrijving: Converteert een Base58 string naar de originele tekst in het blok (**EncodeBase58**)

Blokkeren om een bestand te decoderen met het Base64-algoritme (**DecoderFileBase64**).



Invoerparameters: **pathFileBase64 <String>**, **pathFileOrigin <String>**

Uitgangsparameters: Bronbestand dat in het blok is ingevoerd (**EncoderFileBase64**)

Beschrijving: Een Base64-bestand wordt geconverteerd naar het originele bestand dat in het blok is ingevoegd (**EncoderFileBase64**).

Blokkeren om te weten of de vooraf gedefinieerde interne array "ketting" leeg is. (**Lege Bloklijst**)



Invoerparameters: Niet van toepassing.

Uitgangsparameters: Geeft "Waar" als de gegevens leeg zijn of "Vals" als u gegevens heeft.

Beschrijving: Blok om te vragen of de voorgedefinieerde tijdelijke interne array "keten" elementen bevat.

Blokkeren om een tekenreeks te coderen naar Base58. (**EncodeBase58**).



Invoerparameters: **invoer<String>**

Uitgangsparameters: Het levert de originele string die in het blok werd gebruikt (**EncodeBase58**).

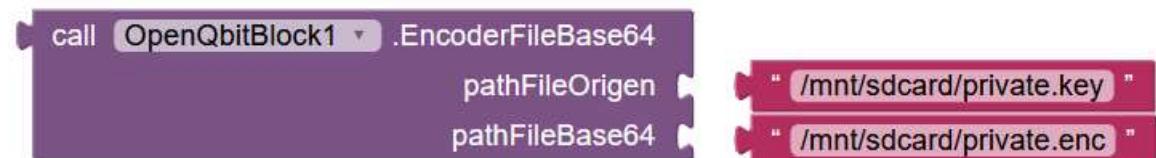
Voorbeeld:

Input = "Mini BlocklyChain is modular".

Output: oBRN8Aj67yJsbRGhfNSF9PTdZYGyVWrwMW7mTX

Omschrijving: Zet een borstketting om in een ketting in Base58. Het Base58 algoritme is een groep van binaire tot tekscoderingsschema's die worden gebruikt om grote gehele getallen als alfanumerieke tekst weer te geven, geïntroduceerd door Satoshi Nakamoto voor gebruik met Bitcoin.

Blokkeren voor het coderen van een bestand met Base64-algoritme (**EncoderFileBase64**).



Invoerparameters: **pathFileOrigin <String>**, **pathFileBase64 <String>**

Uitgangsparameters: Base64 gecodeerd bestand.

Omschrijving: Converteert een bronbestand van elk formaat naar een Base64-bestand. Bestandsnamen kunnen willekeurig zijn en door de gebruiker worden gekozen.

Blok Generator van gebruikersadressen (**GenerateAddrBitcoin**).



Verplichte eenheid: Blok (**ApiGetQRNGinteger**),

Afhankelijkheden (optioneel): **OpenQbitFileEncription-extensie**, **OpenQbitFileDecryption-extensie** en **OpenQbitSQLite-extensie**.

Invoerparameters: **qrng** < verplichte afhankelijkheid>

Uitgangsparameters: alfanumeriek transactieadres van 34 tekens en keyStore-gebruiksadres

Omschrijving: Blokkeren om een nieuw generiek Bitcoin-transactieadres voor gebruiker en privé-sleutelgenerator (Digitale handtekening voor het verzenden van transacties) en publieke sleutel (Publiek adres voor het maken van transacties) aan te maken. Deze sleutelgenerator is in principe de adresgenerator voor gebruik in een digitale portemonnee of de zogenaamde "Portemonnee".

Dit blok wordt gebruikt in combinatie met de Quantum Random Number Generator (QRNG) blokken en de twee uitgangsparameters moeten in de KeyStore worden ingevoegd.

KeyStore is een database die privé-sleutels in hexadecimaal formaat opslaat:

Voorbeeld van een hexadecimaal adres dat is opgeslagen in de KeyStore

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

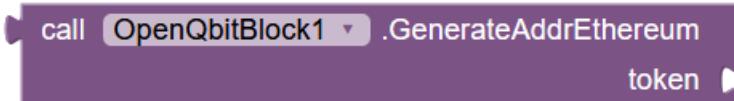
Deze basis wordt alleen gebruikt door de lokale gebruiker en de informatie wordt versleuteld, dit proces is door het gebruik van OpenQbitSQLite extensie en OpenQbitAESEncryptie en OpenQbitAESDecryptie informatie encryptie extensies.

Om een KeyStore aan te maken, zie de bijlage "Een KeyStore aanmaken". De gegenereerde adressen gebruiken hetzelfde Bitcoin-adresalgoritme, waarbij de initiële bitcoin-adresidentificatie "1" is.

De adressen die door het blok (**GenerateAddrBitcoin**) worden gegenereerd zijn 34 alfanumerieke karakters en bestaan uit 33 alfanumerieke karakters en 1 van de Bitcoin-identifier, en wel als volgt:

12dRugNcdxK39288NjcDV4GX7rMsKCGn6k

Gebruikersadres-Generator-Block (**GenereerAddrEthereum**).



Verplichte afhankelijkheid: Koop een penning op: <https://accounts.blockcypher.com/signup>

Afhankelijkheden (optioneel): **OpenQbitFileEncryption-extensie**, **OpenQbitFileDecryption-extensie** en **OpenQbitSQLite-extensie**.

Invoerparameters: **token** <verplichte afhankelijkheid - String>

Uitgangsparameters: 40 alfanumerieke tekens transactieadres bevat niet de initiële etherische indicator "0x" die ons de 42 karakters van een gemeenschappelijk adres zou geven. Het geeft ook de openbare sleutel en de privésleutel terug.

Voorbeeld:

```
{  
  "privé": "227ac59f480131272003c2d723a7795ebd3580acaab62b5c537989e2ce4e08ef",  
  "publiek":  
    "04e2d55ebccd32a7384e096df559cc36b856c64a16e5b402e10585dc3ea055672aafa84df8  
    a859531570a650a8ab1e7a22949100efa1aa5f072c035551cac1ce",  
  "adres": "14e150399b0399f787b4d6fe30d8b251375f0d66"}  
}
```

Omschrijving: Blokken om een nieuw transactieadres te creëren voor de gebruiker en privé-sleutelgenerator (Digitale handtekening om transacties te versturen) en publieke sleutel (Publiek adres om transacties te maken). Deze sleutelgenerator is een externe API en u moet een Wifi of mobiele verbinding hebben, het is in principe de adresgenerator om het te gebruiken in een digitale portemonnee of algemeen "Portemonnee" genoemd.

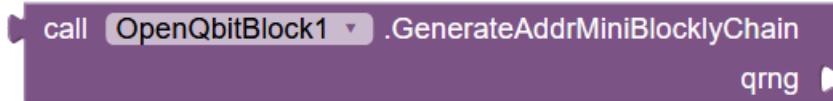
U kunt aKeyStore aanmaken is een database die privé sleutels in hexadecimaal formaat opslaat, zie bijlage "Aanmaken van KeyStore".

Voorbeeld van een hexadecimaal adres dat is opgeslagen in de KeyStore

0x14e150399b0399f787b4d6fe30d8b251375f0d66

De privé-sleutel wordt alleen gebruikt door de lokale gebruiker en de informatie wordt versleuteld, dit proces is door het gebruik van OpenQbitSQLite extensie en OpenQbitAESEncryptie en OpenQbitAESDecryptie informatie encryptie extensies.

Gebruikersadres Generator Block (**GenereerAddrMiniBlocklyChain**).



Verplichte

eenheid: Blok (**ApiGetQRNGinteger**),

Afhankelijkheden (optioneel): **OpenQbitFileEncription-extensie**, **OpenQbitFileDecryption-extensie** en **OpenQbitSQLite-extensie**.

Invoerparameters: **qrng** < verplichte afhankelijkheid>

Uitgangsparameters: 36 alfanumerieke tekens transactieadres en keyStore-gebruiksadres. Review Block Method (**PairKeysMBC**).

Omschrijving: Blokkeren om een nieuw transactieadres te creëren voor de gebruiker en privé-sleutelgenerator (Digitale handtekening om transacties te versturen) en publieke sleutel (Publiek adres om transacties te maken). Deze sleutelgenerator is in principe de adresgenerator om te gebruiken in een digitale portemonnee of in de volksmond "Portemonnee" genoemd.

Dit blok wordt gebruikt in combinatie met de Quantum Random Number Generator (QRNG) blokken en de twee uitgangsparameters moeten in de KeyStore worden ingevoegd.

KeyStore is een database die privé-sleutels in hexadecimaal formaat opslaat:

Voorbeeld van een hexadecimaal adres dat is opgeslagen in de KeyStore

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

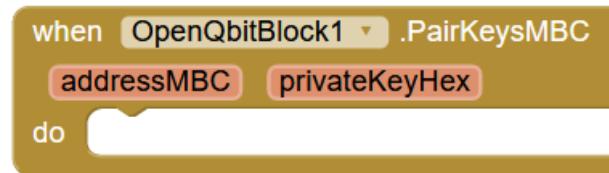
Deze basis wordt alleen gebruikt door de lokale gebruiker en de informatie wordt versleuteld, dit proces is door het gebruik van OpenQbitSQLite extensie en OpenQbitAESEncryptie en OpenQbitAESDecryptie informatie encryptie extensies.

Om een KeyStore aan te maken, zie de bijlage "Een KeyStore aanmaken". De gegenereerde adressen gebruiken hetzelfde bitcoin-adresalgoritme, het enige verschil is dat de bitcoin-adresidentificatie die "1" of "3" is, wordt gewijzigd in de Mini BlocklyChain-identificatie "MBC".

De door het blok gegenereerde adressen (**GenerateAddrMiniBlocklyChain**) zijn 36 alfanumerieke tekens en bestaan uit 33 alfanumerieke tekens en 3 van de Mini BlocklyChain-identificatiehoofdletters "MBC", en wel als volgt:

Mbc2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

Blokmethode (**PairKeysMBC**) is blokuitgang (**GenerateAddrMiniBlocklyChain**).



Invoerparameters: Niet van toepassing.

Uitgangsparameters: **adresMBC <String>**, **privateKeyHex <String>**.

Beschrijving: Geleverd openbaar gebruikersadres in Mini BlocklyChain formaat en privé sleutel in hexadecimaal formaat.

Voorbeeld:

adresMBC: MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

privateKeyHex:

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

Blok Generator van gebruikersadressen (**GenerateKeyValuePair**).



Verplichte eenheid: Blok (**ApiGetQRNGinteger**),

Afhankelijkheden (optioneel): **OpenQbitFileEncryption-extensie**, **OpenQbitFileDecryption-extensie** en **OpenQbitSQLite-extensie**.

Invoerparameters: **qrng < verplichte afhankelijkheid>**

Uitgangsparameters: Biedt de openbare sleutel en de primaire sleutel voor de lokale gebruiker

Beschrijving: Genereert publieke en primaire sleutels met behulp van ECC-algoritme (1) en met hexadecimaal formaat.

(1) Elliptische curvecryptografie (ECC) is een variant van asymmetrische of publieke-sleutel-cryptografie op basis van de wiskunde van elliptische krommen.

Blok om transacties te ondertekenen (**GenerateSignature**).

call OpenQbitBlock1 .GenerateSignature

Vereiste afhankelijkheid: Blok (**GenerateKeyPairs**), Blok (**SenderLoadKeyPair**), Blok (**RecipientLoadKeyPair**). Voorafgaand aan het gebruik van deze blokken.

Invoerparameters: <Verplichte controlepost afhankelijkheden>

Uitgangsparameters: Geen uitgang.

Beschrijving: Het genereert een digitale handtekening van de afzender toegepast op het actief dat in de transactie naar de ontvanger wordt verzonden, deze handtekening zal worden bevestigd wanneer de transactie wordt verwerkt door een of ander knooppunt van het netwerk, met deze handtekening het Mini BlocklyChain systeem zorgt ervoor dat het actief niet is gewijzigd in het verzonden en dat het voldoet aan de afzender-ontvanger relatie en niet wordt omgeleid of toegepast op een ander digitaal adres.

Blokkeren om het totale saldo van de activa van een gebruiker te verkrijgen (**GetBalance**).

call OpenQbitBlock1 .GetBalance

fromAddrMBC

" MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k "

Verplichte afhankelijkheid: Blok (**ConnectorTransactionTail**).

Invoerparameters: **vanTransTail** <Array String>, <Verplichte Boards afhankelijkheden>

Uitgangsparameters: Controleert de inkomende en uitgaande activakosten voor elke transactie.

Omschrijving: Controleert het saldo om goed te keuren als de gebruiker activa heeft om te verzenden of als de activa die hij ontvangt worden toegevoegd aan zijn saldo.

Blokkeer om q22 van de mobiele telefoon te krijgen. (**GetDeviceID**)

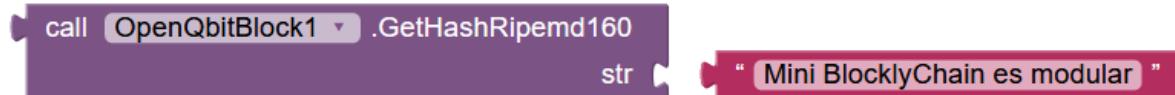
call OpenQbitBlock1 .GetDeviceID

Invoerparameters: **Niet van toepassing**

Uitgaande parameters: Het levert de interne identificatie van de mobiele telefoon.

Omschrijving: Geeft de IMEI mobiele telefoon identifier die uniek is voor elk apparaat.

Blokkeren om de RIPEMD-160 hash van een string te verwijderen. (**Ripemd-160**)



Invoerparameters: **str <String>**

Uitgangsparameters: Berekent de "RIPEMD-160" hash van een tekenreeks.

Voorbeeld:

Input = "Mini BlocklyChain is modular".

uitgang: ae29436e4b8ea8ed6143f3f92380dfa2f4f47336

Beschrijving: Verkrijg de "RIPEMD-160" hasj. Deze hash wordt gebruikt bij het genereren van een geldig adres voor Bitcoin en Mini BlocklyChain.

RIPEMD-160 (acroniem voor RACE Integrity Primitives Evaluation Message Digest) is een 160-bits algoritme voor het verteren van berichten (en een cryptografische hashfunctie).

Blok om Merkler's boom te berekenen. (**GetMerkleRoot**)



Invoerparameters: **transacties <Array String>**

Uitgangsparameters: Het levert een hasj type "SHA256".

Voorbeeld:

Input = transactiewachtrij, een regeling van alle te verwerken transacties.

uitgang: b4a44c42b6070825f763cd118d6ab49a8e80bbb7cdc0225064f8e042b94196bd

Beschrijving: Verkrijg de "SHA256" hasj met behulp van Merkle's boomalgoritme

Een Merkle Hash Tree is een binaire of niet-binaire gegevensboomstructuur waarin elke knoop die geen blad is, wordt gelabeld met de hash van de aaneenschakeling van de labels

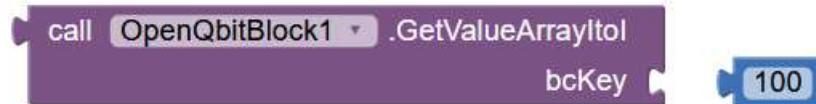
of waarden (voor bladknopen) van zijn kindknobbels. Dit zijn een veralgemening van haslijsten en hasjsnaren.

Het maakt het mogelijk om een groot aantal afzonderlijke gegevens te koppelen aan één enkele hasjwaarde, de hasj van de boomwortelknoop. Dit biedt een veilige en efficiënte methode om de inhoud van grote datastructuren te controleren. In de praktische toepassingen wordt de hasj van de wortelknooppunt meestal ondertekend om de integriteit ervan te garanderen en om ervoor te zorgen dat de verificatie volledig betrouwbaar is. Om aan te tonen dat een bladknoop deel uitmaakt van een bepaalde hasjboom is een hoeveelheid gegevens nodig die in verhouding staat tot het logaritme van het aantal knooppunten in de boom.

Op dit moment is het belangrijkste gebruik van Merkle trees om de datablokken die van andere peers in de peer-to-peer netwerken worden ontvangen, veilig te maken, zodat ze zonder schade en zonder te worden gewijzigd worden ontvangen.

Het wordt gebruikt om te controleren of een wachtrij die de te verwerken transacties heeft niet is gewijzigd en om de integriteit ervan te garanderen voor verspreiding in alle knooppunten van het Mini BlocklyChain-netwerk. Alle knooppunten moeten dit algoritme uitvoeren om de integriteit van elke transactie die zal worden toegepast te waarborgen.

Blokkeren om een waarde van de voorgedefinieerde array "Itol_UTXO" (**GetValueArrayItol**) te verkrijgen.

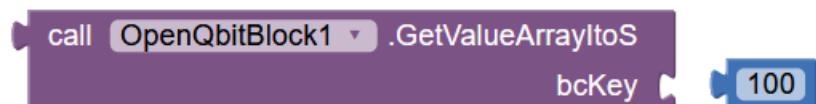


Invoerparameters: **bcKey** <Integer>

Uitgangsparameters: Geeft als resultaat de waarde <Integer> die is opgeslagen in het label met het opgegeven getal als invoer.

Beschrijving: Het is een verzoek om de tijdelijke sleutelwaarde regeling, die is vooraf gedefinieerd met interne naam "Itol_UTXO" dit is nuttig om UTXO transacties te verwerken.

Blokkeer om een waarde te verkrijgen uit de vooraf gedefinieerde array "ItoS_UTXO" (**GetValueArrayItoS**).

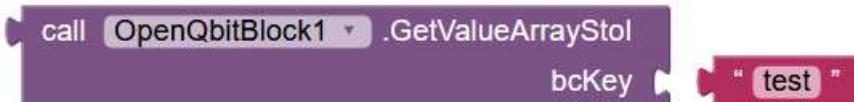


Invoerparameters: **bcKey** <Integer>

Uitgangsparameters: Geeft als resultaat de waarde <String> die is opgeslagen in het label met het opgegeven getal als invoer.

Beschrijving: Het is een verzoek om de tijdelijke sleutelwaarde regeling, die is vooraf gedefinieerd met interne naam "**ItoS_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokkeren om een waarde te verkrijgen uit de vooraf gedefinieerde array "StoI_UTXO" (**GetValueArrayStoI**).

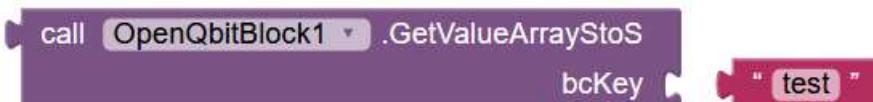


Invoerparameters: **bcKey** < String>

Uitgangsparameters: Geeft als resultaat de waarde <Integer> die is opgeslagen in het label met de voornaam als invoer.

Beschrijving: Het is een verzoek om de tijdelijke sleutelwaarde regeling, die is vooraf gedefinieerd met interne naam "**StoI_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokkeren om een waarde te verkrijgen uit de vooraf gedefinieerde array "StoS_UTXO" (**GetValueArrayStoS**).



Invoerparameters: **bcKey** < String>

Uitgangsparameters: Geeft als resultaat de waarde <String> die is opgeslagen in het label met de voornaam als invoer.

Beschrijving: Het is een verzoek om de tijdelijke sleutelwaarde regeling, die is vooraf gedefinieerd met interne naam "**StoS_UTXO**" dit is nuttig om UTXO transacties te verwerken.

Blokvalidator van de blokketen. (**IsChainValid**)

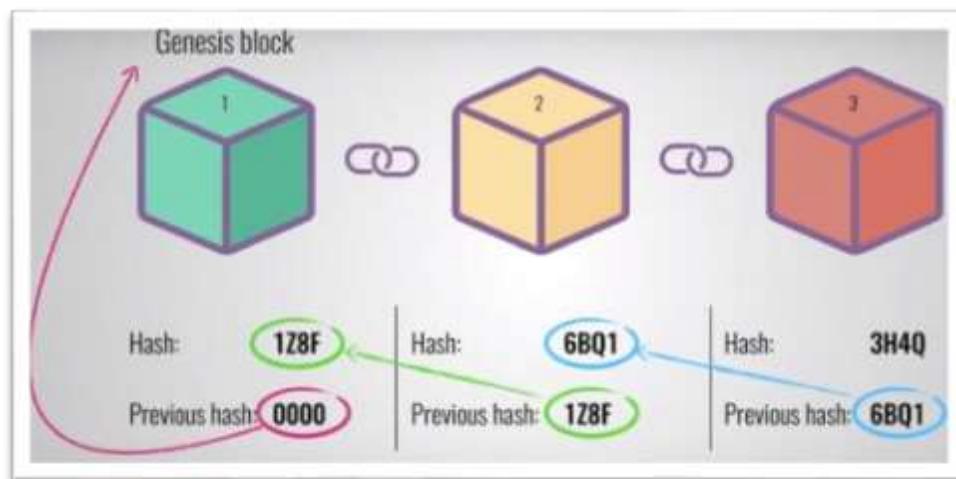


Vereiste afhankelijkheid: Blok (**GenerateKeyPairs**), Blok (**SenderLoadKeyPair**), Blok (**RecipientLoadKeyPair**), Blok (**GenerateSignature**). Voorafgaand aan het gebruik van deze blokken.

Invoerparameters: Niet van toepassing.

Uitgangsparameters: Levering "True" als de validatie van de blokreeks correct is of "False" als de validatie mislukt.

Omschrijving: Het geeft ons de validatie van de componenten die eerder in het blokketensysteem zijn ingebracht, deze validatie is de belangrijkste van het hele systeem. Hoe werkt de blokketen validatie of hoe is het algemeen bekend op een generieke manier (BlockChain). Het is de kern van elk systeem.



Zoals we in de vorige afbeelding zien is elk blok dat in het opslagsysteem is toegevoegd gerelateerd aan het vorige blok door middel van de hash-algoritmes.

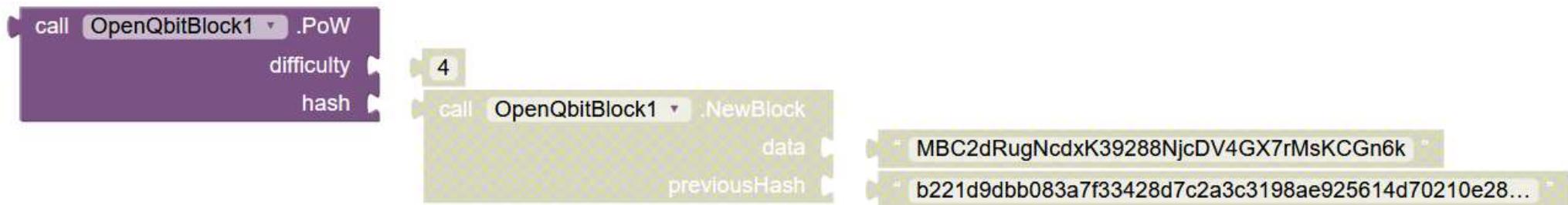
Bijvoorbeeld, bij het aanmaken van een nieuw blok om toe te voegen, wordt de hash die de nieuwe informatie vertegenwoordigt verkregen door de hash van het nieuwe informatieblok + de vorige hash te nemen. Met dit type schakel wordt elke minimale wijziging in de opslag van blokkettingen gedetecteerd, wat een zeer hoge en effectieve databaseveiligheid mogelijk maakt.

Hieronder staat een voorbeeld van hoe een reeks van blokken wordt opgeslagen in een SQLite database, we zullen zien hoe de vorige hash is gekoppeld aan de nieuwe hash van het laatste blok (laatste wachtrij van verwerkte transacties) dat werd ingevoegd. Elke hash in zowel de "prevhash" als de "newhash" kolom vertegenwoordigt de informatie van de transactiewachtrij die op dat moment werd verwerkt.

rowid	id	prevhash	newhash	opera	trans	balan
Click here to define a filter						
1	1	0767c864cef0334f27473902eb9868e7	bdc9065d20a4cd037bb1a7538486403e	2	0	0
2	2	bdc9065d20a4cd037bb1a7538486403e	6619f4809d73a267a4b9ac554bb4523a	1	5	5
3	3	6619f4809d73a267a4b9ac554bb4523a	4d186321c1a7f0f354b297e8914ab240	1	5	5

De vorige hash is gerelateerd aan de nieuwe hash.

Blok om PoW Work Test uit te voeren en om nieuwe blokken te ontginnen. (**PoW**)



Verplichte afhankelijkheid: **NewBlock**. Geef dit blok een voorwoord alvorens het te gebruiken.

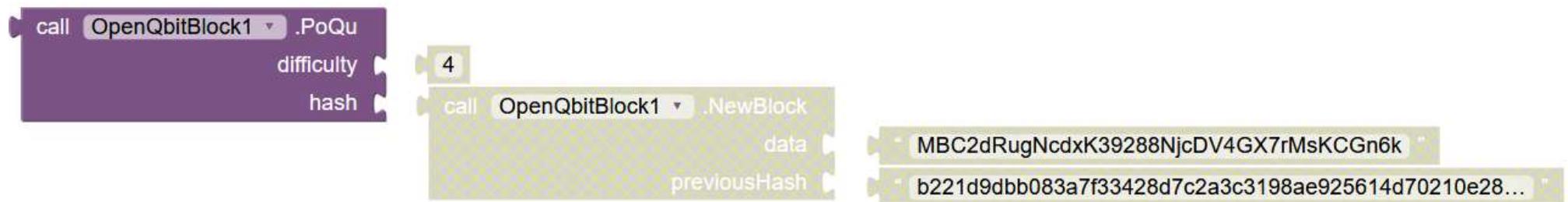
Invoerparameters: **moeilijkheidsgraad** <Integer>, hasj <Vereiste afhankelijkheid>

Uitgangsparameters: Geeft als uitgang een hasj "SHA256" samengesteld met het #Nummer van "nullen" gegeven in de ingangsparameter moeilijkheidsgraad.

Beschrijving: Dit blok voert een activiteit genaamd "mijnbouw" een nieuw blok is wat we eerder hebben beschreven als het proces van PoW (Proof of Work), zie paragraaf Wat is Proof of Quantum (PQu)?

Het ontginnen of uitvoeren van het PoW is een concept waarbij de knooppunten een taak krijgen om een wiskundige puzzel op te lossen. Wie in het geval van Mini BlocklyChain het eerst oplost, krijgt de kans om door te gaan met het proces om zo de winnaar te worden gekozen van het kunnen verwerken van de transactiewachtrij die wacht om verwerkt te worden.

Blok om PoW Work Test uit te voeren en om nieuwe blokken te ontginnen. (**PoQu**)



Verplichte afhankelijkheid: **NewBlock**. Geef dit blok een voorwoord alvorens het te gebruiken.

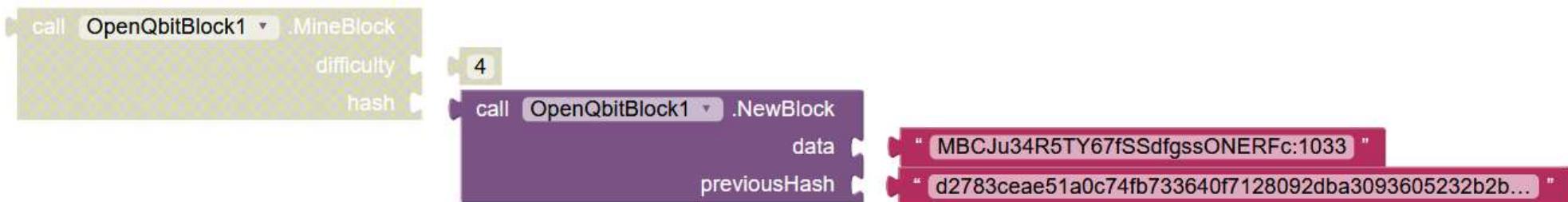
Invoerparameters: **moeilijkheidsgraad** <Integer>, hasj <Vereiste afhankelijkheid>

Uitgangsparameters: Resulteert in het getal "Magisch getal" en een willekeurig kwantumnummer dat binnen het bereik ligt van 0 en 1 decimaal getal van 16 cijfers, voorbeeld (0. 5843012986202495) en een hasj "SHA256" samengesteld met het #Nummer van "nullen" dat in de invoerparameter moeilijkheidsgraad wordt gegeven.

Beschrijving: Dit blok voert het proces van "mijnbouw" een nieuw blok is wat we eerder hebben beschreven als het proces van PoW (Proof of Work), maar dit proces noemt de functie van willekeurige kwantumnummer gegenereerd na de berekening van het aantal "nonce" geschikt om de moeilijkheidsgraad die kan worden in het bereik van minimaal 1, maximaal 5. Zie paragraaf Wat is Proof of Quantum (PQu - Proof Quantum) te voldoen?

Het ontginnen of uitvoeren van de PoW of PoQu is een concept waarbij de knooppunten een taak krijgen om een wiskundige puzzel op te lossen. Het knooppunt dat het eerst oplost in het geval van een blokketensysteem krijgt de kans om door te gaan met het proces om zo de winnaar te worden gekozen van het kunnen verwerken van de transactiewachtrij die wacht om te worden verwerkt.

Blok voor de creatie van **nieuwe** blokken (**NewBlock**).

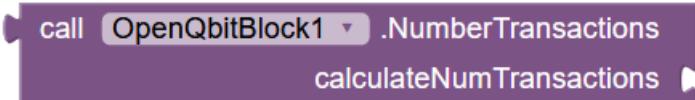


Invoerparameters: **gegevens <String>**, **vorigeHash <String>**

Uitgangsparameters: Het levert een hasj berekend als: **SHA256 (data (invoerparameters) + previousHash (invoerparameter) + IMEI (interne parameter) + MerkleRoot (interne parameter))**.

Beschrijving: Dit blok voert de creatie van een nieuw te verwerken blok uit. Het geeft als uitgang een "SHA256" hash die bestaat uit de string van invoerparameters in het geval van gegevens varieert afhankelijk van elk systeemontwerp en de vorige hash is gebaseerd op de hash van de vorige transactiestring die al werd verwerkt en is opgeslagen in de Mini BlocklyChain blokstring systeem, deze twee zijn variabele parameters, er zijn twee interne systeemparameters die vast zijn en de integriteit van de informatie en het systeem garanderen, deze twee interne parameters zijn de unieke ID van de mobiele telefoon en de hash van de merkletboom van de transactiewachtrij die wordt verwerkt.

Blok van de totale lokale transacties van het knooppunt (**NumberTransactions**)



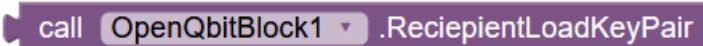
Verplichte afhankelijkheid: Blok (**AddKeyValueToArrayStoS**). Geef dit blok een voorwoord alvorens het te gebruiken.

Invoerparameters: < Verplichte afhankelijkheid>.

Uitgangsparameters: Geeft het totaal van de lokale transacties terug naar het knooppunt.

Beschrijving: Geeft het totaal van de transacties die alleen op de lokale rekeningen van het knooppunt zullen worden toegepast.

Blokkeren om het adres van de ontvanger te lezen in een binair bestand. (**RecipientLoadKeyPair**)

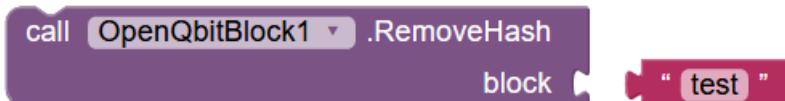


Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat een privésleutel en een openbare sleutel in een Base64-formaat.

Omschrijving: Krijgt het privé en publieke adres van de ontvanger uit een binair bestand als invoerparameter.

Blokkeren om element te verwijderen in de interne tijdelijke regeling "keten" (**RemoveHash**).

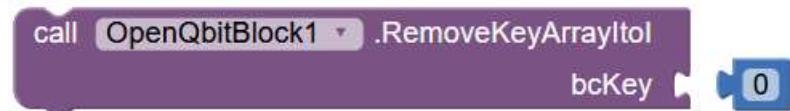


Invoerparameters: **blok <String>**.

Uitgangsparameters: Niet van toepassing.

Omschrijving: Krijgt het privé en publieke adres van de ontvanger uit een binair bestand als invoerparameter.

Blokken om element te verwijderen in de interne tijdelijke regeling "ItoL_UTXO" (**RemoveKeyArrayItoL**)

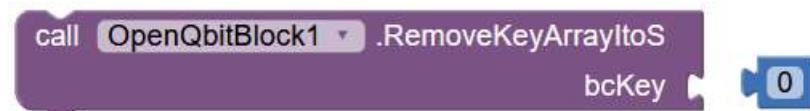


Invoerparameters: **bcKey < Integer >**.

Uitgangsparameters: Niet van toepassing, type element verwijderen <Integer>

Omschrijving: Het element dat verbonden is met het numerieke label van de interne tijdelijke regeling "ItoL_UTXO" wordt verwijderd.

Blokken om element te verwijderen in de interne tijdelijke regeling "ItoS_UTXO" (**RemoveKeyArrayItoS**).

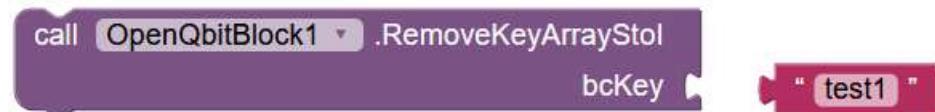


Invoerparameters: **bcKey < Integer >**.

Uitgangsparameters: Niet van toepassing, type element dat moet worden verwijderd <String>

Omschrijving: Het element dat verbonden is met het numerieke label van de interne tijdelijke regeling "ItoS_UTXO" wordt verwijderd.

Blok voor het verwijderen van element in de interne tijdelijke indeling "StoL_UTXO" (**RemoveKeyArrayStoL**)

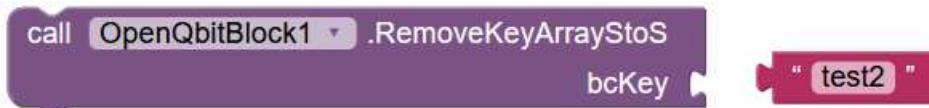


Invoerparameters: **bcKey < String >**.

Uitgangsparameters: Niet van toepassing, type element verwijderen <Integer>

Omschrijving: Het element dat verbonden is met het numerieke label van de interne tijdelijke regeling "StoL_UTXO" wordt verwijderd.

Blok voor het verwijderen van element in de interne tijdelijke opstelling "StoS_UTXO" (RemoveKeyArrayStoS)



Invoerparameters: **bcKey < String>**.

Uitgangsparameters: Niet van toepassing, type element dat moet worden verwijderd <String>

Omschrijving: Verwijder element van de interne tijdelijke regeling label "StoS_UTXO".

Blokkeren om een waarde van de interne tijdelijke regeling "keten" te vervangen (ReplaceHash).



Invoerparameters: **blok < String>, index < Integer>**

Uitgangsparameters: Niet van toepassing.

Omschrijving: Het element dat is gekoppeld aan de index "index" wordt vervangen door de waarde van het label "blok" in de tijdelijke interne regeling "keten".

Blokkeren om een nieuwe transactie te starten (**verzenden**) (SendTrasaction).



Invoerparameters: **vanAddrMBC < String>, naarAddrMBC < String>, waarde < Integer>**

Uitgangsparameters: Geeft de waarde "Waar" als de transactie succesvol werd verzonden of "Valse" als het niet succesvol was hij verzonden.

Omschrijving: Blok dat het afzender- en ontvangstadres omzet in een binair formaat en dat wordt toegevoegd aan de te verwerken transactiewachtrij.

Blokkeren om het adres van de afzender te lezen in een binair bestand. (**SenderLoadKeyPair**)

 call **OpenQbitBlock1** .**SenderLoadKeyPair**

Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat een privésleutel en een openbare sleutel in een Base64-formaat.

Omschrijving: Krijgt het privé en publieke adres van de afzender uit een binair bestand als invoerparameter.

Blok om het element nummer van de tijdelijke regeling "keten" (**SizeBlockList**) te verkrijgen.

 call **OpenQbitBlock1** .**SizeBlockList**

Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat het elementnummer van de interne array "keten".

Omschrijving: Krijgt het element nummer van de interne array "ketting".

Blok om het elementnummer van de tijdelijke regeling "Itol_UTXO" te verkrijgen (**Sizeltol**)

 call **OpenQbitBlock1** .**Sizeltol**

Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat het elementnummer van de interne array "Itol_UTXO".

Beschrijving: Krijgt het element nummer van de interne array "Itol_UTXO".

Blok om het elementnummer van de tijdelijke regeling "ItoS_UTXO" te verkrijgen (**SizeltoS**)

 call **OpenQbitBlock1** .**SizeltoS**

Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat het elementnummer van de interne array "ItoS_UTXO".

Omschrijving: Krijgt het element nummer van de interne array "ItoS_UTXO".

Blok om het elementnummer van de tijdelijke regeling "Stol_UTXO" te verkrijgen (**SizeStol**)



Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat het elementnummer van de interne array "Stol_UTXO".

Omschrijving: Krijgt het element nummer van de interne array "Stol_UTXO".

Blok om het elementnummer van de tijdelijke regeling "StoS_UTXO" te verkrijgen (**SizeStoS**)

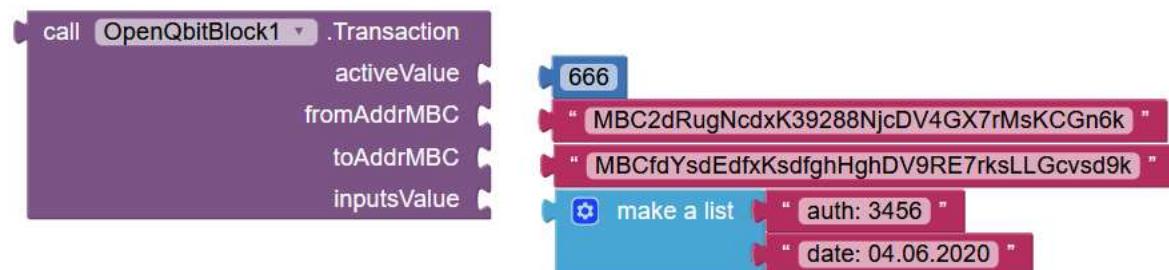


Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft als resultaat het elementnummer van de interne array "StoS_UTXO".

Omschrijving: Krijgt het element nummer van de interne array "StoS_UTXO".

Blokkering van transactie met extra waarden (**Transactie**).



Invoerparameters: **activeValue < Integer>, vanAddrMBC < String>, totAddrMBC < String>, inputsValue < Array String>**.

Uitgangsparameters: Het geeft ons, de adressen in binair formaat en de waarde inputValue omgezet in een string 'String', en geeft ons de hash "SHA256" van de ActiveValue waarde.

Beschrijving: Bereidt een nieuwe transactie voor die door de knooppunten moet worden verwerkt.

Mini BlocklyChain-adres validatieblokje (**ValidateAddMiniBlocklyChain**)



Invoerparameters: Gebruikersadressen in Mini BlocklyChain-formaat.

Uitgangsparameters: Geeft "Waar" als het adres in het juiste formaat is of "Vals" als het adres ongeldig is.

Omschrijving: Valideert of het ingevoerde adres van de Mini BlocklyChain correct is, dit blok past een algoritme toe om te controleren of het adres is aangemaakt met behulp van het adresaanmaakmechanisme dat in het Mini BlocklyChain-systeem moet worden gebruikt.

Bitcoin-adres validatieblok. (**ValidateAddBitcoin**)

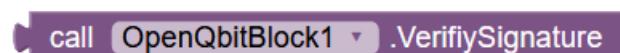


Invoerparameters: **addr < String >**, Bitcoin geformateerde gebruikersadressen (accepteert Bitcoin-adressen met initiële identifier "1", adressen met identifier "3" zijn niet van toepassing).

Uitgangsparameters: Geeft "Waar" als het adres in het juiste formaat is of "Vals" als het adres ongeldig is.

Omschrijving: Valideert of het ingevoerde Bitcoin-adres correct is, dit blok past een algoritme toe om te controleren of het adres is aangemaakt met behulp van het adresaanmaakmechanisme dat in het Bitcoin-systeem moet worden gebruikt.

Verificatieblok voor digitale handtekeningen voor de huidige transactie. (**VerifySignature**).



Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Geeft "Waar" als de controle geldig is of "Vals" als de controle ongeldig is.

Beschrijving: Krijgt de verificatie van de digitale handtekening die de verzender had moeten zien tijdens het proces van het verzenden van de transactie die u wilt maken. In dit verificatieproces wordt gecontroleerd of de verzonden bronwaarde niet is gewijzigd in het kanaal waar de transactie is verzonden, en wordt de ontvanger geverifieerd waarop de transactie moet worden toegepast.

20. Gebruik van blokken voor SQLite database (MiniSQLite versie)

In deze sectie zullen we zien hoe we de blokken kunnen gebruiken om twee belangrijke bewerkingen uit te voeren waarin we geïnteresseerd zijn voor de functionaliteit van het Mini BlocklyChain systeem, namelijk het "INSERT" van gegevens in de blokketten en het opvragen van deze gegevens.

OPMERKING: De transacties in de SQLite database zijn verschillend van de transacties die door de knooppunten worden verzonden om te worden toegepast in het Mini BlocklyChain systeem.

De transacties in de database zijn gebaseerd op een CRUD (Create, Read, Update en Delete) model en zijn de processen die kunnen worden uitgevoerd met externe of interne gegevens alleen in de SQLite database. In de blokketten systemen worden vooral de processen van Create en Read gebruikt, voor de beveiliging worden de processen van het updaten of verwijderen weggegooid en meer waar het is opgeslagen de keten van blokken die de integrale veiligheid van het systeem geeft.

Aan de andere kant, de transacties verzonden door de knooppunten verwijzen naar alle processen die het maken van de actie van het verzenden van een bepaald type van activa tussen de leden (knooppunten) van de Mini BlocklyChain systeem, dit type van transacties omvatten diverse processen voor de toepassing ervan, kunnen deze van de oprichting van een digitaal adres, een digitale handtekening, een handtekening validatie, een proces binnen de SQLite database, onder andere processen.

We beginnen met de definitie en het gebruik van de blokken van de SQLite database MiniSQLite versie deze versie is slechts geïntegreerd door 8 blokken. U heeft een volledige versie om de gegevens in de SQLite database te manipuleren, maar voor praktische doeleinden is het Mini BlocklyChain systeem met de MiniSQLite versie voldoende.

Voor het geval u alle componenten wilt bekijken, zie de bijlage "Uitgebreide blokken voor SQLite database".

MiniSQLite versie blokken:

Blokkeren om een soort transactie te starten in de SQLite database (**BeginTransaction**)

call **OpenQbitQSQLite1** .BeginTransaction

Verplichte eenheid (eenheden): Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **Gebruik vóór < Verplichte afhankelijkheid(en) >**

Uitgangsparameters: Niet van toepassing, het start een transactie in een SQLite database.

Beschrijving: Blok dat een proces in de SQLite database start, moet u eerst het database-importblok (**ImportDatabase**) en het database-openblok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om Commit te maken in SQLite. (CommitTrasactie)

call **OpenQbitQSQLite1** .CommitTransaction

Vereiste afhankelijkheid(en): Blok (**BeginTransactie**), Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **Gebruik vóór < Verplichte afhankelijkheid(en) >**

Uitvoerparameters: Niet van toepassing, het voert een **vastlegging** uit **van een** transactie in een SQLite database.

Beschrijving: Blok dat een **vastleggingsproces** op de SQLite database start, moet u eerst het database-importblok (**ImportDatabase**) en het database-openblok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om de SQLite database die werd geïmporteerd of geëxporteerde te sluiten (**CloseDatabase**)

call **OpenQbitQSQLite1** .CloseDatabase

Verplichte eenheid (eenheden): Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **Gebruik vóór < Verplichte afhankelijkheid(en) >**

Uitgangsparameters: Niet van toepassing, het sluit een SQLite database.

Omschrijving: Blok dat de SQLite database afsluit, u moet eerst het database import blok (**ImportDatabase**) en het database open blok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om een SQLite database (**ExportDatabase**) te exporteren.

call **OpenQbitQSQLite1** .ExportDatabase
fileName " mbcExport.sqlite "

Verplichte eenheid (eenheden): Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **fileName < String>** voer een pad in waar u een reeds aangemaakte database met SQLite formaat kunt vinden.

Gebruik vóór < Verplichte afhankelijkheid(en) >

Uitvoerparameters: Exporteer naar een SQLite database.

Beschrijving: Blok dat een SQLite database exporteert, u moet eerst het database-importblok (**ImportDatabase**) en het database-open blok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om SQLite database te importeren. (**ImportDatabase**)

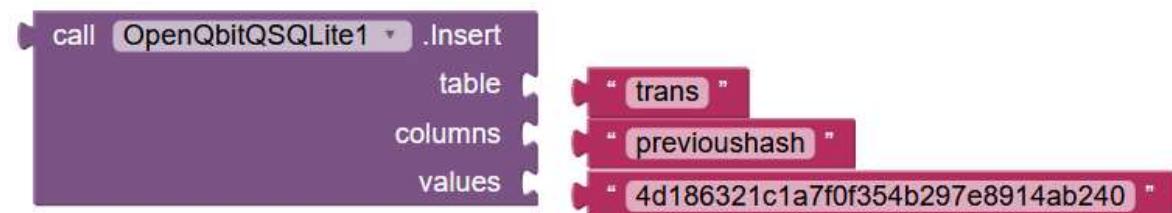


Invoerparameters: **fileName < String>** voer een pad in waar u een reeds aangemaakte database met SQLite formaat kunt vinden.

Uitgangsparameters: Start een SQLite database om transacties uit te voeren.

Beschrijving: Blok dat een proces in de SQLite database start, moet u eerst het database-importblok (**ImportDatabase**) en het database-openblok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om gegevens in de SQLite database in te voegen. (**Insert**)



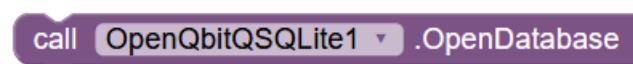
Verplichte eenheid (eenheden): Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **tabel < String>** , **kolommen < String>** , **waarden < String>** , **Gebruik voor < Verplichte afhankelijkheid(en)>**

Uitgangsparameters: Voegt een transactie in een SQLite database in.

Beschrijving: Blok dat gegevens in de SQLite database invoert, moet u eerst het database-importblok (**ImportDatabase**) en het database-openblok (**OpenDatabase**) hebben uitgevoerd.

Blokkeren om SQLite database te openen (**OpenDatabase**)



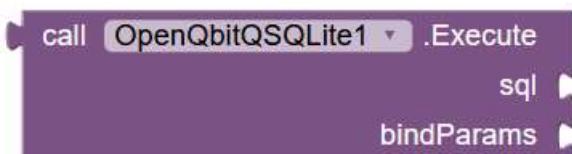
Verplichte eenheid (eenheden): Blok (**ImportDatabase**).

Invoerparameters: **Gebruik vóór < Verplichte afhankelijkheid(en) >**

Uitgangsparameters: Niet van toepassing, start of open een SQLite database om transacties uit te voeren.

Beschrijving: Blok dat een SQLite database start, moet er eerder zijn.

Blokkeren voor data query in SQLite (**Execute**).



Verplichte eenheid (eenheden): Blok (**ImportDatabase**), Blok (**OpenDatabase**).

Invoerparameters: **sql** < String> , **bindParams** < String> , **gebruik voor** < Verplichte afhankelijkheid(en)>

Uitvoerparameters: De SQL-instructie wordt uitgevoerd om een transactie in een SQLite database aan te maken.

Omschrijving: Blok dat SQL statements in de SQLite database uitvoert, moet eerst het blok van de importdatabase (**ImportDatabase**) en het blok van het openen van de database (**OpenDatabase**) hebben uitgevoerd.

21. Definitie en gebruik van veiligheidsblokken.

In deze sessie zullen we het gebruik van de blokken die ons voorzien van beveiligingsniveaus voor het opslaan, valideren en overbrengen van transacties van de Mini BlocklyChain netwerkknooppunten beoordelen.

De veiligheidsblokken zijn gebaseerd op de volgende uitbreiding:

- I. OpenQbitAESEncryptie-uitbreiding.

- II. OpenQbitAESDecryptie-uitbreiding.
- III. OpenQbitAESToString uitbreiding.
- IV. OpenQbitEncDecData uitbreiding.
- V. OpenQbitFileHash uitbreiding.
- VI. OpenQbitRSA uitbreiding.
- VII. OpenQbitSSHClient-extensie (zou vereisen)
- VIII. OpenQbitStringHash uitbreiding.

Met uitzondering van de OpenQbitSSHClient-extensie, die verplicht is, zijn de bovenstaande extensies optioneel te gebruiken bij het opzetten van een openbaar of privé Mini BlocklyChain-netwerk.

Om echter een veilig systeem voor uw transacties te hebben, afhankelijk van elke business case, moet het gebruik van de extensies die optioneel zijn, naar eigen goeddunken worden toegepast.

In het geval van het gebruik van hasj kunnen we bijvoorbeeld gebruik maken van een aantal algoritme-opties zoals MD5, SHA1, SHA128, SHA256, SHA512 voor een reeks karakters en worden toegepast op elk type bestand, afhankelijk van de informatiestroom van elk systeem dat met Mini BlocklyChain is gemaakt.

OpenQbitAESEncryptie-uitbreiding.

Blokkeren om het bestand te versleutelen met AES-beveiliging. (**AESEncryptie**).



Invoerparameters: **pathFileIn < String>** , **pathFileOut < String>** , **pathFile < String>**, **pathFileVI < String>** en **passwd < String>**. Bestandsnamen zijn willekeurig en naar eigen goeddunken van elk systeemontwerp.

Uitgangsparameters: AES-gecodeerd bestand ingevoerd in het **invoerparameterpathFileIn**.

Beschrijving: Blok dat ons drie uitvoerbestanden geeft een van deze is het bronbestand al gecodeerd door het AES-algoritme met een 256-bits sleutel, de andere twee bestanden worden gebruikt om de extensie te ontcijferen en het oorspronkelijke bestand te herstellen.

OpenQbitAESDecryptie-uitbreiding.

Blokkeren om het AES-bestand te ontcijferen. (**AESDecryptie**).



Invoerparameters: **pathFileIn** < String> , **pathFileOut** < String> , **pathFile** < String>, **pathFileVI** < String> en **passwd** < String>. De namen van de bestanden zijn dezelfde als die in het blok (**AESEncryptie**).

Uitgangsparameters: Origineel bestand gedecodeerd met AES ingevoerd in invoerparameter **pathFileIn**, in dit geval om het bestand te decoderen wordt het ingevoerd in **pathFileIn** het versleutelde bestand en in **pathFileOut** zal het ons het originele bestand (gedecodeerd) geven.

Beschrijving: Blok dat ons een bestand geeft in de **pathFileOut** parameter dat de uitvoer zal worden gedecodeerd door het AES-algoritme met een 256-bits sleutel.

OpenQbitAEToString uitbreiding.

Deze uitbreiding is eenmalig per apparaatsessie, d.w.z. het werkt alleen als het apparaat niet opnieuw wordt opgestart (mobiele telefoon), omdat de VI-coderingswaarde tijdelijk wordt gegenereerd.

OPMERKING: Als het apparaat opnieuw wordt opgestart, kan de gecodeerde string niet worden hersteld.

Blok voor tijdelijke tekenreeksencodering (**DecrypSecretText**)

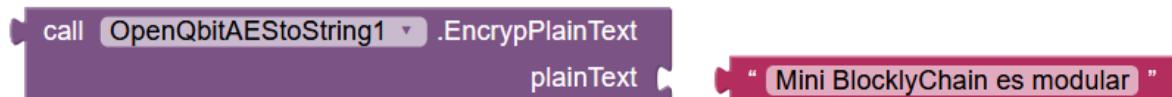


Invoerparameters: **secretText** < String>

Uitgangsparameters: Originele tekenreeks gedecodeerd met AES met 256-bits sleutel

Beschrijving: Blok dat ons een tekenreeks geeft, is de invoerparameter die in het blok werd geïntroduceerd (**EncrypPlainText**).

Blokkeren om een string te decoderen (**EncrypPlainText**)



Invoerparameters: **plainText** < String >

Uitgangsparameters: AES-gecodeerde tekenreeks met behulp van een 256-bits sleutel.

Beschrijving: Blok dat ons een alfanumerieke tekenreeks geeft die met AES is gecodeerd met behulp van een 256-bits digitale sleutel.

OpenQbitEncDecData uitbreiding.

Gespecialiseerd coderingsblok voor generieke databases (**Encryptiegegevens**)



Invoerparameters: **plainText** < String >

Uitgangsparameters: AES-gecodeerde tekenreeks met behulp van een 256-bits sleutel.

Beschrijving: Blok dat ons een alfanumerieke tekenreeks geeft die met AES is gecodeerd met behulp van een 256-bits digitale sleutel.

Gespecialiseerd coderingsblok voor generieke databases (**Decryptiegegevens**)



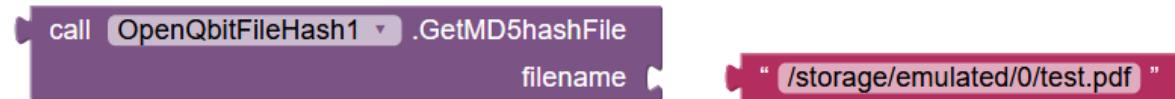
Invoerparameters: **plainText** < String >

Uitgangsparameters: AES-gecodeerde tekenreeks met behulp van een 256-bits sleutel.

Beschrijving: Blok dat ons een alfanumerieke tekenreeks geeft die met AES is gecodeerd met behulp van een 256-bits digitale sleutel.

OpenQbitFileHash uitbreiding.

Blokkeren om MD5 uit een bestand te genereren (**GetMD5hashFile**)

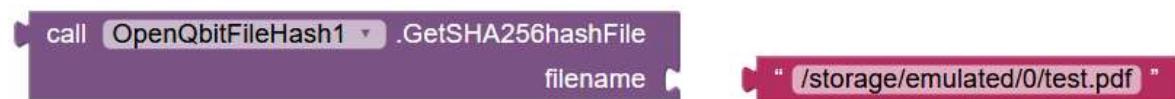


Invoerparameters: **bestandsnaam** <string>

Uitgangsparameters: Levert het MD5-hash-bestand.

Omschrijving: Blokkeren om de MD5 hash van het bestand te maken dat in de invoerparameter is opgegeven.

Blokkeren om SHA256 uit een bestand te genereren (**GetSHA256hashFile**)

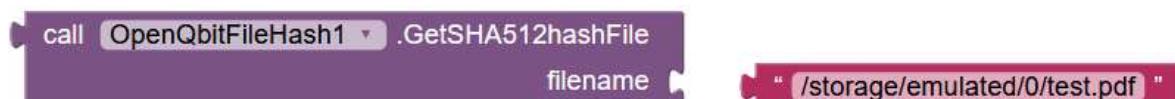


Invoerparameters: **bestandsnaam** <string>

Uitgangsparameters: Levert de SHA256 archiefhasj.

Beschrijving: Blokkeren om de SHA256 hash van het bestand te maken dat in de invoerparameter wordt gegeven.

Blokkeren om SHA512 uit een bestand te genereren (**GetSHA512hashFile**)

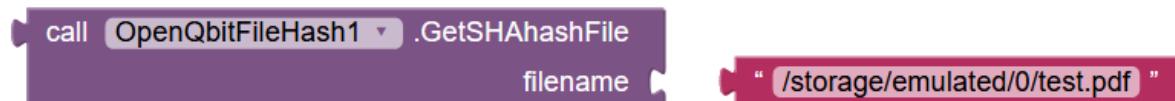


Invoerparameters: **bestandsnaam** <string>

Uitgangsparameters: Levert de SHA256 archiefhasj.

Beschrijving: Blokkeren om de SHA256 hash van het bestand te maken dat in de invoerparameter wordt gegeven.

Blokken om SHA1 uit een bestand te genereren (**GetSHA1hashFile**)



Invoerparameters: **bestandsnaam** <string>

Uitgangsparameters: Levert de SHA1-archieffhasj.

Beschrijving: Blokkeren om de SHA1-hash van de matrijs in de ingangsparameter te maken.

OpenQbitRSA uitbreiding.

Blok voor het **ontcijferen** van de string met RSA (**Decrypt**)



Vereiste afhankelijkheid(en): Blok (**Encrypt**), Blok (**OpenFromDiskPrivateKey**), Blok (**OpenFromDiskPublicKey**).

Invoerparameters: **resultaat** <String>

Uitgangsparameters: Tekenreeks gedecodeerd met RSA.

Beschrijving: Blok dat ons een reeks alfanumerieke tekens geeft die ontcijferd zijn met behulp van een sleutel van het formaat dat in het blok werd gebruikt (**GenKeyPair**).

Blok voor het coderen van de string met RSA (**Encrypt**)



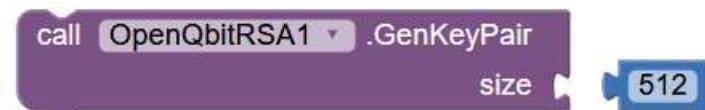
Vereiste eenheid (eenheden): Blok (**GenKeyPair**), Blok (**SaveFromDiskPrivateKey**), Blok (**SaveFromDiskPublicKey**)

Invoerparameters: **gewoon** < String >

Uitgangsparameters: RSA-gecodeerde tekenreeks

Beschrijving: Blok dat ons een reeks alfanumerieke tekens geeft die ontcijferd zijn met behulp van een sleutel van het formaat dat in het blok werd gebruikt (**GenKeyPair**).

Blok voor het **ontcijferen van de** string met RSA (**Decrypt**)

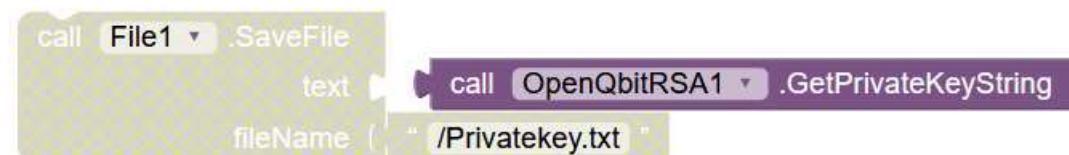


Invoerparameters: **grootte** < Integer >

Uitgangsparameters: Niet van toepassing.

Beschrijving: Blok voor het genereren van een privésleutel en een publieke sleutel op basis van de gekozen grootte.

Blokkeren om de privé-sleutel te verkrijgen (**GetPrivateKeyString**)



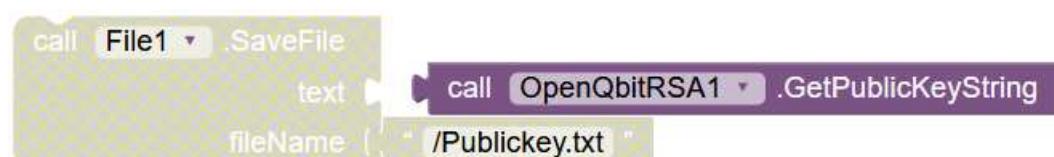
Verplichte eenheid (eenheden): Blok (**GenKeyPair**), Blok (**GenKeyPair**), Blok (**Bestand**)

Invoerparameters: **Niet van toepassing**.

Uitvoerparameters: Bestand met RSA gecodeerde tekenreeks (privésleutel)

Beschrijving: Blok dat ons een alfanumerieke tekenreeks geeft die de privé-sleutel voorstelt die gecodeerd is met de grootte van het blok (**GenKeyPair**).

Blok om de privé-sleutel te verkrijgen (**GetPublicKeyString**)



Verplichte eenheid (eenheden): Blok (**GenKeyPair**), Blok (**GenKeyPair**), Blok (**Bestand**)

Invoerparameters: **Niet van toepassing**.

Uitgangsparameters: Bestand met RSA gecodeerde string (publieke sleutel)

Beschrijving: Blok dat ons een alfanumerieke tekenreeks geeft die de openbare sleutel voorstelt, gecodeerd met behulp van de sleutelgrootte die in het blok wordt gebruikt (**GenKeyPair**).

OPMERKING: In de vorige blokken (**GetPrivateKeyString**) en (**GetPublicKeyString**) in de afhankelijkheden zullen we gebruik maken van het generieke blok (**File**) van de App Inventor applicatie van de "Storage" pallet sessie.

Deze manier van opslaan van de private en publieke sleutel door middel van het blok (**bestand**) kan ons helpen om de informatie beter te manipuleren.

Blokkeer om een private sleutel uit een bestand te lezen (**OpenFromDiskPrivateKey**).



Vereiste afhankelijkheid(en): Blok (**SaveFromDiskPrivateKey**) of Blok (**GetPrivateKeyString**).

Invoerparameters: **pad < String >**

Uitgangsparameters: RSA gecodeerde tekenreeks voor systeembelasting met privé-sleutel.

Beschrijving: Blok dat ons een gecodeerde alfanumerieke tekenreeks geeft van de privé-sleutel die is opgeslagen in het meegeleverde bestandspad.

Blokkeren om de publieke sleutel van een bestand te lezen (**OpenFromDiskPublicKey**)



Verplichte Eenheid(en): Blok (**SaveFromDiskPublicKey**) of Blok (**GetPublicKeyString**).

Invoerparameters: **pad < String >**

Uitgangsparameters: Laad in het systeem RSA openbare sleutel gecodeerde tekenreeks.

Beschrijving: Blok dat ons een gecodeerde alfanumerieke string geeft van de publieke sleutel die is opgeslagen in het meegeleverde bestandspad.

Blokkeren voor het opslaan van de private key in een bestand (**SaveToDiskPrivateKey**).



Verplichte eenheid (eenheden): Blok (**GenKeyValuePair**).

Invoerparameters: **gewoon** < String >

Uitvoerparameters: Bestand met RSA gecodeerde string. (privésleutel)

Beschrijving: Blok dat ons een bestand geeft met een alfanumerieke tekenreeks die is versleuteld met behulp van een privé-sleutel van de grootte die in het blok wordt gebruikt (**GenKeyValuePair**).

Blokkeren voor het opslaan van de private key in een bestand (**SaveToDiskPublicKey**).



Verplichte eenheid (eenheden): Blok (**GenKeyValuePair**).

Invoerparameters: **gewoon** < String >

Uitvoerparameters: Bestand met RSA gecodeerde string. (openbare sleutel)

Beschrijving: Blok dat ons een bestand geeft met een alfanumerieke tekenreeks die is versleuteld met behulp van een openbare sleutel van de grootte die in het blok werd gebruikt (**GenKeyValuePair**).

OpenQbitSSHClient-extensie.

Connector Block SSH Client (**ConnectorMiniBlocklyChain**)



Invoerparameters: **gebruikersnaam** <string>, **wachtwoord** <string>, **host** <string>, **poort**<integer>

Uitgangsparameters: Als de verbinding met de ssh-server van de Termux-terminal succesvol is, geeft het ons een bericht; "**Connect SSH**", als het niet succesvol is, geeft het ons een **NUL берicht**.

Beschrijving: Communicatieblok om Mini BlocklyChain te verbinden met Termux-terminal, via SSH (Secure Shell) communicatieprotocol.

Blok voor het uitvoeren van commando's in Termux Linux Terminal



(**CommandLineMiniBlocklyChain**).

Invoerparameters: **commando** <string>

Uitgangsparameters: Variabele gegevens, afhankelijk van het uitgevoerde commando of programma.

Omschrijving: Commando-uitvoeringsblok in Termux-terminal is nodig om een verbinding met het blok te maken (**ConnectorMiniBlocklyChain**) kan allerlei commando's online uitvoeren en/of specifieke uitvoeringsgegevens krijgen van scripts of programma's die een CLI (Command-Line Interface) online hebben.

Ontkoppel**MiniBlocklyChainSSH**.

```
call OpenQbitSSHClient1 .DisconnectMiniBlockchainSSH
```

Ingangs- en uitgangsparameters: Niet van toepassing (geen)

Beschrijving: Blok om de SSH-sessie af te sluiten.

OpenQbitStringHash uitbreiding.

Blok om MD5 string te genereren (**MD5ThisString**)

```
call OpenQbitStringHash1 .MD5ThisString
      string
```

“ Mini BlocklyChain es modular ”

Invoerparameters: string

Uitgangsparameters: Levert de MD5 hash.

Beschrijving: Blokkeren om de MD5-hash van de string te maken die in de invoerparameter wordt gegeven.

Blok om SHA256 tekenreeks te genereren (**SHA256ThisString**)

```
call OpenQbitStringHash1 .SHA256ThisString
      string
```

“ Mini BlocklyChain es modular ”

Invoerparameters: string

Uitgangsparameters: Levert de SHA256 hash.

Beschrijving: Blokkeren om de SHA256 hash van de in de invoerparameter gegeven string te maken.

Blok om SHA512 string te genereren (**SHA512ThisString**)

```
call OpenQbitStringHash1 .SHA512ThisString
      string
```

“ Mini BlocklyChain es modular ”

Invoerparameters: string

Uitgangsparameters: Levert de SHA512 hash.

Beschrijving: Blokkeren om de SHA512 hash van de in de invoerparameter gegeven string te maken.

Blok om SHA1-karakterreeks te genereren (**SHAThisString**)



Invoerparameters: string

Uitgangsparameters: Levert de SHA1 hash.

Beschrijving: Blokkeren om de SHA1 hash van de in de invoerparameter gegeven string te maken.

22. Instellen van veiligheidsparameters in Mini BlocklyChain.

De veiligheidsparameters zijn verdeeld in drie componenten van elk systeem dat is ontworpen en toegepast in de volgende componenten:

- a. Redis-database (back-up communicatienetwerk)
- b. Peer to Peer-synchronisatiesysteem
- c. Integreer beveiliging om de SQLite database te beschermen
- d. Ontwikkelaarsomgeving voor module-integratie

- a. Redis-database (back-up communicatienetwerk)

Het hernoemen van gevaarlijke commando's, de extra ingebouwde veiligheidsfunctie van Redis houdt in dat sommige commando's die als gevaarlijk worden beschouwd, worden hernoemd of uitgeschakeld.

Bij uitvoering door onbevoegde gebruikers kunnen deze commando's worden gebruikt om hun gegevens te herconfigureren, te vernietigen of te verwijderen. Net als het authenticatiwachtwoord wordt het hernoemen of uitschakelen van commando's geconfigureerd in dezelfde sectie van het bestand /etc/redis/redis.conf.

Sommige van de commando's werden als gevaarlijk beschouwd: **FLUSHDB**, **FLUSHALL**, **KEYS**, **PEXPIRE**, **DEL**, **CONFIG**, **SHUTDOWN**, **BGREWRITEAOF**, **BGSAVE**, **SAVE**, **SPOP**, **SREM**, **RENAME**

en **DEBUG**. Dit is geen volledige lijst, maar het hernoemen of uitschakelen van alle commando's op die lijst is een goed begin om de veiligheid van uw Redis-server te verbeteren.

Afhankelijk van uw specifieke behoeften of die van uw site, moet u een opdracht hernoemen of uitschakelen. Als je weet dat je nooit een commando zult gebruiken dat gemanipuleerd kan worden, kun je het uitschakelen. Aan de andere kant wil je het misschien een andere naam geven.

Om Redis-opdrachten in of uit te schakelen, opent u het configuratiebestand opnieuw:

```
$ vi /etc/redis/redis.conf
```

Waarschuwing: De volgende stappen om opdrachten uit te schakelen en te hernoemen zijn voorbeelden. U moet alleen kiezen voor het uitschakelen of hernoemen van de commando's die op u van toepassing zijn. U kunt de volledige lijst van commando's bekijken en bepalen hoe ze kunnen worden misbruikt in redis.io/commands.

Om een commando uit te schakelen, hernoemt u het eenvoudigweg zodat het een lege tekenreeks wordt (gesymboliseerd door een paar aanhalingsstekens zonder tekens ertussen), zoals hieronder weergegeven:

```
/etc/redis/redis.conf
```

```
.
.
.
# Het is ook mogelijk om een commando volledig te doden door het te
hernoemen naar
# een lege snaar:
#
hernoemd FLUSHDB ""
rename-command FLUSHALL ""
hernoeming DEBUG ""
```

Om een commando te hernoemen, geef het een andere naam zoals in de voorbeelden hieronder. Hernoemde commando's zouden voor anderen moeilijk te raden moeten zijn, maar voor u gemakkelijk te onthouden.

```
/etc/redis/redis.conf
```

```
.
.
.
# rename-command CONFIG "
hernoeming SHUTDOWN SHUTDOWN_MENOT
hernoemen CONFIG ASC12_CONFIG
```

Sla de wijzigingen op en sluit het bestand.

Na het hernoemen van een commando, past u de wijziging toe door Redis opnieuw op te starten:

- sudo systemctl restart redis.service

Om het nieuwe commando te testen, voert u de opdrachtregel van Redis in:

- redis-cli

Voer dan de authenticatie uit:

- auth your_redis_password

Uitgang

OK

Stel dat u, net als in het vorige voorbeeld, het CONFIG commando hernoemt naar ASC12_CONFIG. Probeer eerst het originele CONFIG commando te gebruiken. Omdat je het een andere naam hebt gegeven, zou het niet moeten werken:

- config requirepass krijgen

Uitgang

(fout) ERR onbekend commando "config".

Het hernoemde commando kan echter met succes worden opgeroepen. Het is niet hoofdlettergevoelig:

- asc12_config get requirepass

Uitgang

1) "requirepass"
2) "your_redis_password".

Tenslotte zult u in staat zijn om de herindeling te sluiten:

- afslag

Merk op dat als u de opdrachtregel van Redis al gebruikt en Redis opnieuw opstart, u zich opnieuw moet authenticeren. Anders, als u een opdracht typt, wordt deze fout weergegeven:

Uitgang

NOAUTH Authenticatie vereist.

b. Peer to Peer-synchronisatiesysteem

Bij het gebruik van het "Peer to Peer"-systeem tussen knooppunten worden de volgende drie elementen in het communicatienetwerk toegepast

-Private: er is nergens anders dan op uw computers informatie opgeslagen. Er is geen centrale server die kan worden gecompromitteerd (legaal of illegaal).

-Gecodeerd: Alle communicatie wordt beveiligd via het TLS (Transport Layer Security) protocol; een cryptografisch protocol dat een perfecte volgorde bevat om te voorkomen dat iemand buiten uw vertrouwen toegang krijgt tot uw informatie.

-Authentiek: Elk knooppunt wordt geïdentificeerd met een sterk cryptografisch certificaat. Alleen de knooppunten die u expliciet hebt toegestaan, kunnen verbinding maken met uw informatie.

<https://docs.syncthing.net/users/security.html>

Met behulp van het online commando SyncthingManager:

<https://github.com/classicsc/syncthingmanager>

c. Integreer beveiliging om de SQLite database te beschermen

Bij gebruik van de extensie (**OpenQbitSQLite**) of in het geval van de extensie (**ConnectorSSHClient**) om de SQLite CLI te gebruiken kunnen beide extensies gecombineerd worden met de veiligheidsextensie AES (**OpenQbitEncDecData**).

Zie bijlage "Voorbeeld van het creëren van een Mini BlocklyChain systeem".

d. Ontwikkelaarsomgeving voor module-integratie

Het implementeren van veiligheid in de ontwikkelingsomgeving kan met twee processen gebeuren:

- Beperk het gebruik van OpenJDK-bibliotheken.
- Gebruik beperkt tot geautoriseerde systeemknooppunten.

23. Bijlage "Oprichting van KeyStore & PublicKeys-databases".

Een KeyStore is een opslagplaats van veiligheidscertificaten, ofwel autorisatiecertificaten ofwel publieke sleutelcertificaten, wachtwoorden of generieke veiligheidssleutels zoals de corresponderende privé-sleutels (adressen) van een gebruiker, die gebruikt wordt om transacties aan te maken of te verwerken.

Het is een gecodeerde database, zodat alleen de eigenaar er gebruik van kan maken. Normaal gesproken is het een lokaal type, maar in het Mini BloclyChain systeem is het een beveiligde database, maar het wordt gedeeld en gedistribueerd in alle knooppunten, dit is in principe te wijten aan een eenvoudige reden, alle knooppunten moeten de adressen van alle gebruikers kennen (publieke adressen), de privé-adressen zijn altijd lokaal en zijn van uniek en exclusief gebruik van elke gebruiker, echter bij het maken van een ingangs-(stort)- of uitgangs-(kosten)transactie heeft elke transactie altijd ten minste drie componenten wanneer deze wordt aangemaakt: Oorspronkelijk adres, bestemmingsadres en activa of waarde die wordt verzonden.

Om onze KeyStore te creëren zullen we de volgende stappen en vereisten moeten volgen.

Een eerste vereiste is om een opslagtype te hebben waar ze zullen worden opgeslagen, in ons geval zullen we gebruik maken van de SQLite database en we zullen twee KeyStore een met de gebruiker prive-sleutels die lokaal zal zijn en het zal worden beveiligd "versleuteld" de informatie en een andere database die de openbare sleutels zal opslaan deze basis zal worden gedeeld in alle knooppunten van het netwerk, de basis die zal worden gedeeld in het netwerk zal ook worden versleuteld, maar deze zal een wachtwoord dat alleen de leden (knooppunten) van het netwerk in staat zal zijn om te delen hebben.

Tweede fundamentele vereiste is het proces van informatievercijfering, dit zal worden gedaan met de gespecialiseerde extensie voor gebruik in een generieke database genaamd (**OpenQbitEncDecData**) die gebruik maakt van een AES-algoritme.

De extensie (**OpenQbitEncDecData**) is functioneel voor de verificatie van unitaire elementen van de blokketten, maar in het geval dat de totale integriteit van de blokketten moet worden gecontroleerd zal het niet efficiënt zijn, dus we zullen ook een encryptie van een kopie uitvoeren, maar in dit geval zal het via de extensies gaan: (**OpenQbitAESEncryptie**) en (**OpenQbitAESDecryptie**) die werken op een bestand, niet op basis van gerefereerde gegevens.

OPMERKING: De SSH-server moet op de Termux-terminal draaien om de **KeyStore-database** goed te laten functioneren. Voer het commando uit in de terminal:

\$ sshd

Uitbreidingen op basis van het AES-algoritme kunnen worden gebruikt voor elk type gegevens of bestand en dit algoritme is gekozen omdat het als enige is bewezen dat het bestand bestand bestand is tegen aanvallen op basis van quantumcomputers.

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

In bovenstaande tabel wordt verwezen naar de Nationale Academies van Wetenschap, Techniek en Geneeskunde.

<https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects>

Beschrijving

Quantummechanica, het subveld van de fysica dat het gedrag van zeer kleine deeltjes (quantums) beschrijft, vormt de basis voor een nieuw computerparadigma. Voor het eerst voorgesteld in de jaren tachtig als een manier om de computationele modellering van kwantumsystemen te verbeteren, heeft het veld van de kwantumberekening de laatste tijd veel aandacht getrokken door de vooruitgang in de bouw van kleinschalige apparaten. Er zal echter aanzienlijke technische vooruitgang nodig zijn voordat een praktische kwantumcomputer op grote schaal kan worden gerealiseerd.

Quantum Computing: Progress and Prospects biedt een inleiding op het gebied, met inbegrip van de unieke kenmerken en beperkingen van de technologie, en beoordeelt de haalbaarheid en implicaties van het creëren van een functionele quantumcomputer die in staat is om problemen in de echte wereld aan te pakken. In dit verslag worden de hardware- en softwarevereisten, de kwantumalgoritmen, de drijvende krachten achter de vooruitgang

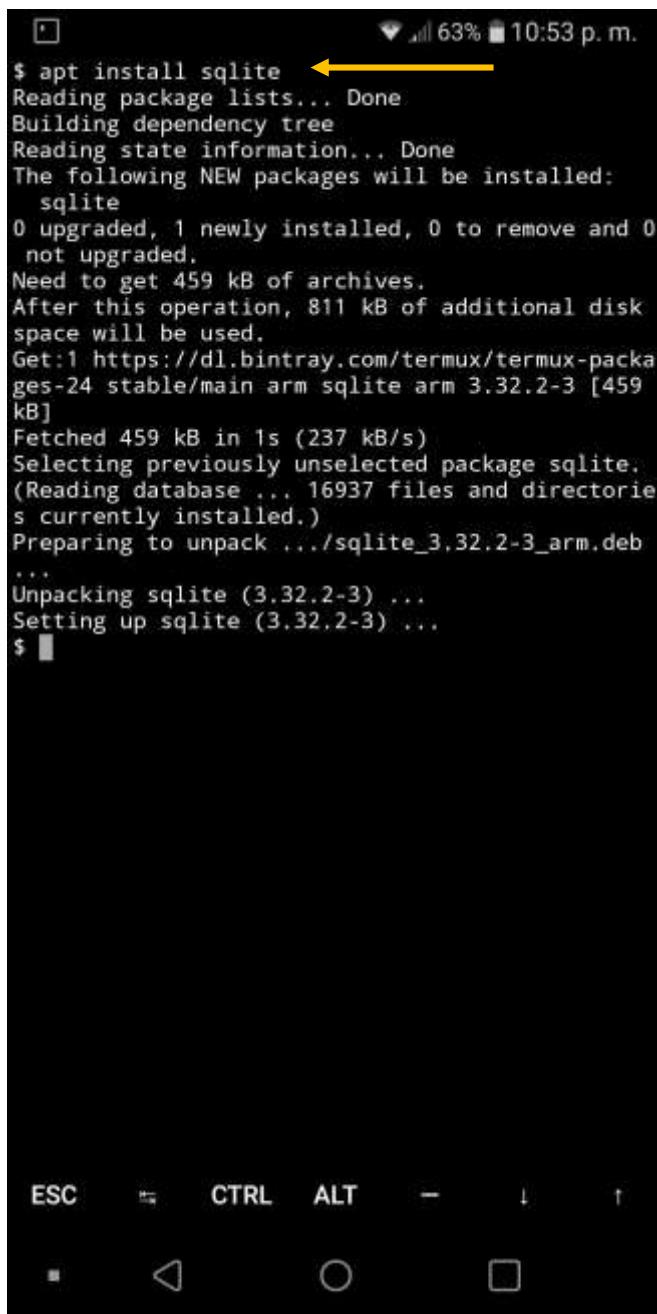
op het gebied van kwantumcomputing en kwantumapparaten, de benchmarks in verband met relevante gebruikssituaties, de benodigde tijd en middelen en de manier waarop de kans op succes kan worden beoordeeld, in overweging genomen.

Installatie van SQLite database handler in TERMUX terminal en het testen van de CLI (Command-Line) van het **sqlite3** commando door het creëren van een database genaamd **keystore.db**.

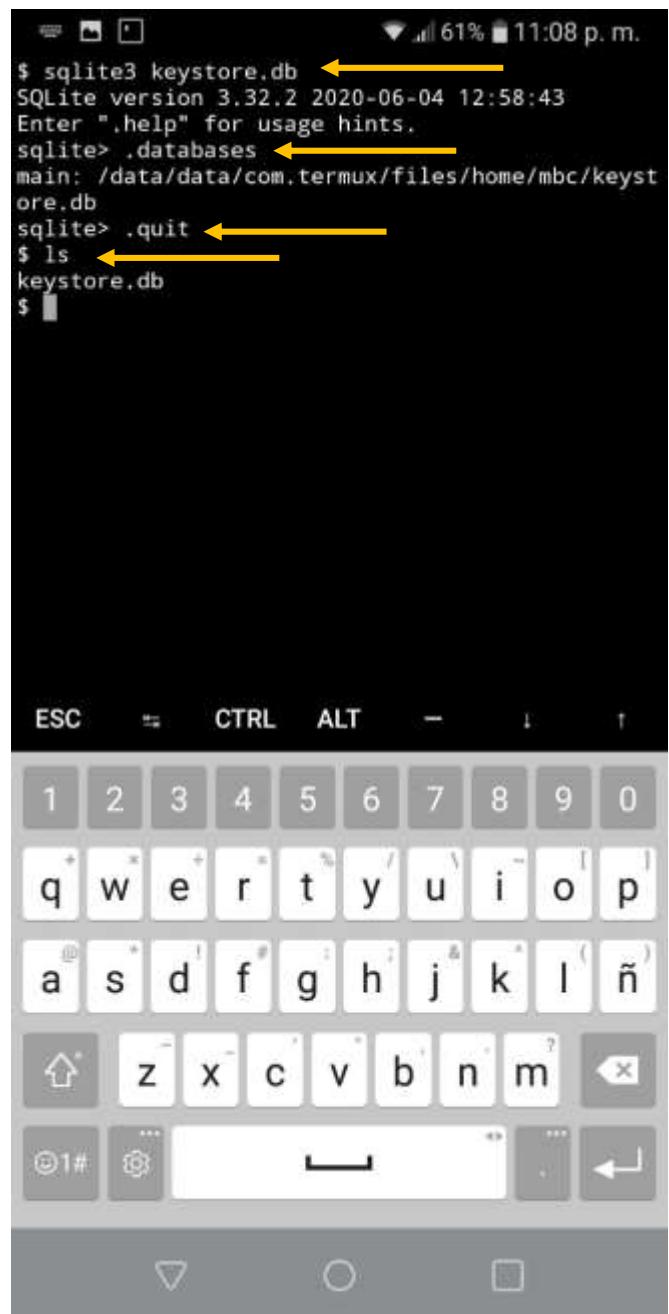
We gebruiken de commando's:

```
$ apt installen sqlite
```

```
$ sqlite3 keystore.d.
```



```
$ 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-packages-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
$
```

Vervolgens voeren we in de **sqlite> handler** de **.databases** zin uit om te controleren in welk pad de database die we aan het maken zijn, vervolgens om de database af te sluiten en op te slaan geven we de zin van **.quit**.

OPMERKING: In beide statements of commando's binnen de **sqlite> handler** moet u eerst een punt **".** in de syntaxis zetten.

Tot slot controleren we of de (lege) database al is aangemaakt door het commando te geven:

\$ ls

We gaan verder met het maken van een tabel waarin de primaire sleutels worden opgeslagen in drie verschillende formaten: hexadecimaal, binair en het Mini BlocklyChain gebruikersreferentieadres dat de publieke sleutel van de respectievelijke primaire sleutel is.

AES-dataversleuteling wordt alleen toegepast in hexadecimale en binaire formaten. In het geval van het adres van de gebruiker addrMBC en de alias niet omdat het de publieke sleutel is die kan en moet worden gedeeld in het netwerk om transacties te ontvangen, evenals de alias om de zoekopdracht uit te voeren door dit veld.

Een tabel gemaakt met de naam "privatekey" in de database in SQLite (keystore.db).

```
CREATE TABLE privatekey (
    id gehele primaire sleutel
    a.k.a.      VARCHAR(50) NIET ONGELDIG
    addrHexVARCHAR (65) NIET NULLEN
    addrMBCVARCHAR (65) NIET NULLES
    addrBinBLOB NOT NULL
);
```

Laten we de zinnen in de sqlite CLI uitvoeren om de keystore.db database aan te maken.

Laten we de sqlite3 commandoregel opnieuw gebruiken met het volgende commando:

\$ sqlite3

Dit stuurt ons in de **sqlite> database** manager. In deze eerste openen we de database die al is aangemaakt om er aan te kunnen werken met de volgende zin in de manager:

Sqlite> .open keystore.db

Vervolgens zullen we de SQL-instructie "**CREATE TABLE**" invoeren om de **privatekey-tabel aan te maken**.

Vervolgens worden twee opties getoond om dezelfde tabel te maken, één is via een enkele regel waar alle SQL-informatie van de elementen die deze integrerden zijn opgenomen. Het tweede voorbeeld is door de introductie van elk element dat de structuur op een gesegmenteerde manier integreert.

\$ sqlite3

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.

sqlite> .open keystore.d

sqlite> CREATE TABLE privatekey (id integer primary key AUTOINCREMENT NOT NULL, aka VARCHAR(50) NOT NULL, addrHex VARCHAR(65) NOT NULL, addrMBC VARCHAR(65) NOT NULL, addrBin BLOB NOT NULL);

sqlite> .quit

De creatie van dezelfde **privatekey-tabel** wordt nu getoond, maar het is door de introductie van de SQL-statement op een gesegmenteerde manier:

```
sqlite> CREATE TABLE privatekey (
...>   id  integer primaire sleutel AUTOINCREMENT
...>   aka  VARCHAR(50) NOT NULL,
...>   addrHex VARCHAR(65) NOT NULL,
...>   addrMBC VARCHAR(65) NOT NULL,
...>   addrBin BLOB NOT NULL
...> );
sqlite> .tables
privésleutel
sqlite> .quit
```

De twee bovenstaande voorbeelden geven hetzelfde resultaat. Later voeren we de **.tables** zin uit om te controleren of de privatekey-tabel is aangemaakt, aan het eind geven we de **.quit** zin met dit proces hebben we de **privatekey-tabel** al aangemaakt in de SQLite **keystore.db** database.

Tot dit moment hebben we al de structuur van de keystore.db database en de privatekey tabel waar de private sleutels versleuteld worden opgeslagen.

Dit moet iets dergelijks laten zien in het knooppunt (mobiele telefoon) met de TERMUX terminal.



```
$ 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
$
```

We zullen nu enkele beveiligingsuitbreidingen gebruiken om "private key" gegevens in te voegen en vervolgens deze gegevens te raadplegen.

We zullen het blok gebruiken om een nieuw gebruikersadres te genereren ([GenerateAddrMiniBlocklyChain](#)), dit blok zal ons het privé-adres in twee formaten (hexadecimaal en binair) geven, evenals het publieke adres in MBC generiek adresformaat. We zullen ook de extensie ([OpenQbitSSHClient](#)) en de extensie ([OpenQbitEncDecData](#)) gebruiken om meer details te zien van deze check de sectie "Definitie en gebruik van veiligheidsblokken". In de Termux-terminal zou u de SSH-dienst moeten draaien.

INSERT gecodeerde gegevens in **keystore.db** database

```
when SveKeyStore .Click
do evaluate but ignore result
call OpenQbitBlock1 .GenerateAddrMiniBlocklyChain
    qrng " 11, 2, 45, 89, 23, 5, 8, 66, 3, 1, 99 "
    pathFilePrivateKey "/mnt/sdcard/dcim/MiniPrivate.key"
```



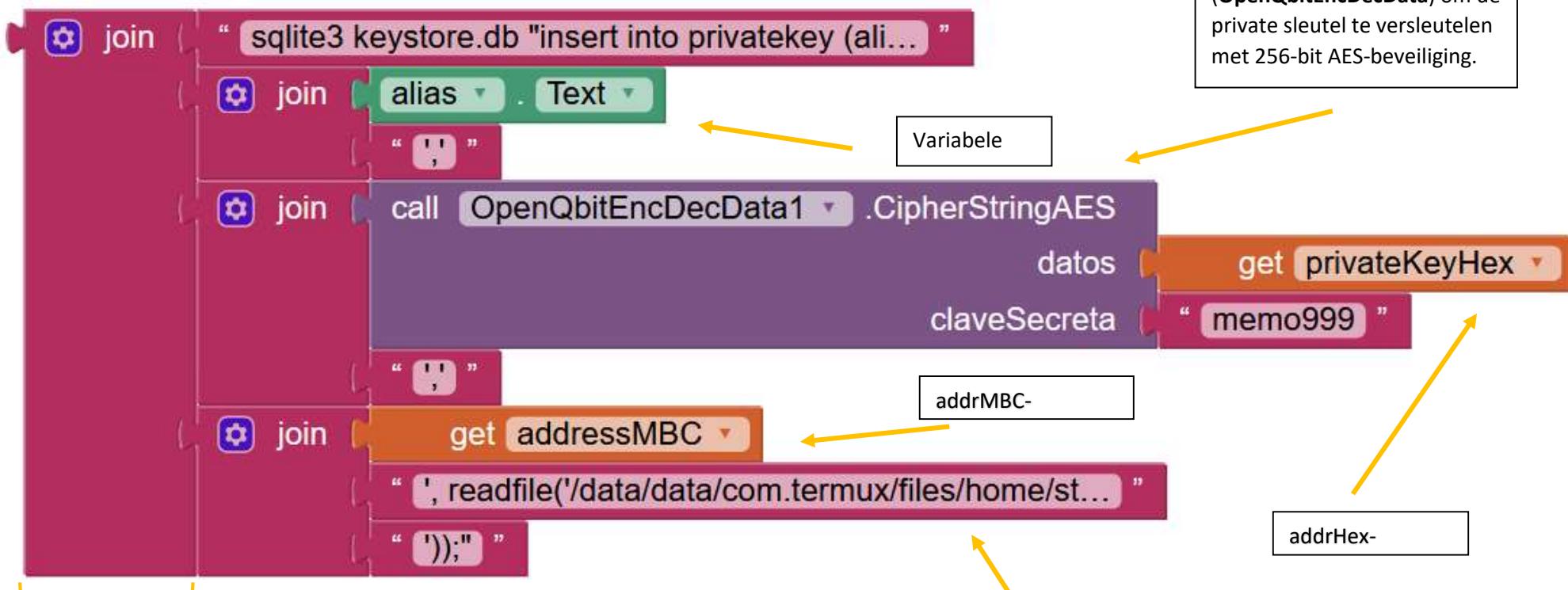
Quantum random number series gegenereerd door het blok (ApiGetQRNGInteger) controleer hoe het JSON resultaat te formatteren, aangezien de blokinvoer (GenerateAddrMiniBlocklyChain) een reeks getallen nodig heeft die alleen door komma's worden gescheiden ",".

```
when OpenQbitBlock1 .PairKeysMBC
addressMBC privateKeyHex pathFilePrivateKey
do evaluate but ignore result
call OpenQbitSSHClient1 .ConnectorMiniBlocklyChain
    username " u0_a251 "
    password " memo1234 "
    host " localhost "
    port 8022
set global SelectSQLenc to
call OpenQbitSSHClient1 .CommandLineMiniBlocklyChain
    command join " sqlite3 keystore.db \"insert into privatekey (ali...\" "
        join alias Text
        join call OpenQbitEncDecData1 .CipherStringAES
            datos get privateKeyHex
            claveSecreta " memo999 "
    join
        get addressMBC
        " , readfile('/data/data/com.termux/files/home/st...\" "
        " );\" "
call OpenQbitSSHClient1 .DisconnectMiniBlocklyChainSSH
```

De syntax van het commando is zeer belangrijk, we moeten het volgende commando in het juiste formaat invoeren in de Blockly-omgeving.

De waarden van de waarden zullen altijd de variabelen zijn, bijvoorbeeld:

`sqlite3 keystore.db "in te voegen in privatekey (alias, addrHex, addrMBC, addrBin) waarden ("memo", "QWERTY", "MBC12345", readfile ("~/data/data/com.termux/files/home/storage/dcim/MiniPrivate.key"))";`



Generieke join-blok van
omgevingen in de "Tekst" tools.

Variabele

Na het uitvoeren van de blokken hebben we een resultaat in de privatekey-tabel van de keystore.db database, vergelijkbaar met het volgende:

We gaan de sqlite handler binnen in de Termux terminal, openen de keystore.db basis en voeren de straf uit:

`$ sqlite3`

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.

`sqlite> .open keystore.d`

`sqlite> selecteer * van 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|07JBBizTcC0Ce6aB8ve5aTV410l1DKiUQZyPSfJuRbVUZnvIwQJcry4LiiBM5jHavQeMo1iN8rC087V5Xka2YaKR4fswopeTQmI/Q+ipzI=|2eENpFt2HSuGtXNxoeiwfrp5d6e87Z7y5|000
sqlite>

```

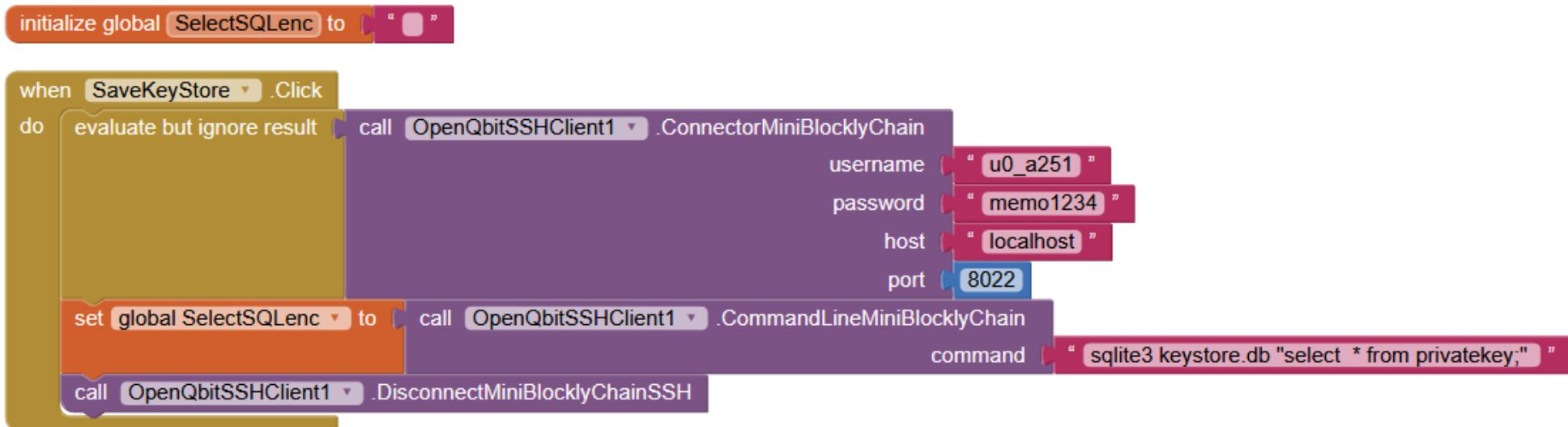
Alias: mexico

addrBin (privésleutel in binair)

addrHex

addrMBC-adres

CONSULTEREN alle gegevens in de **privatekey**-tabel van de SQLite **keystore.db**-database



CONSULTEREN de aliassen van de gegevens in de **privatekey**-tabel van de SQLite **keystore.db**-database

sqlite3 keystore.db "selecteer alias van privatekey;"

Query alle gegevens in de addrHex kolom gecodeerd in de **privatekey** tabel van de SQLite **keystore.db** database

sqlite3 keystore.db "select addrHex van privatekey;"

CONSULTEREN om de addrBin private key op te halen in de **privatekey**-tabel van de SQLite **keystore.db**-database

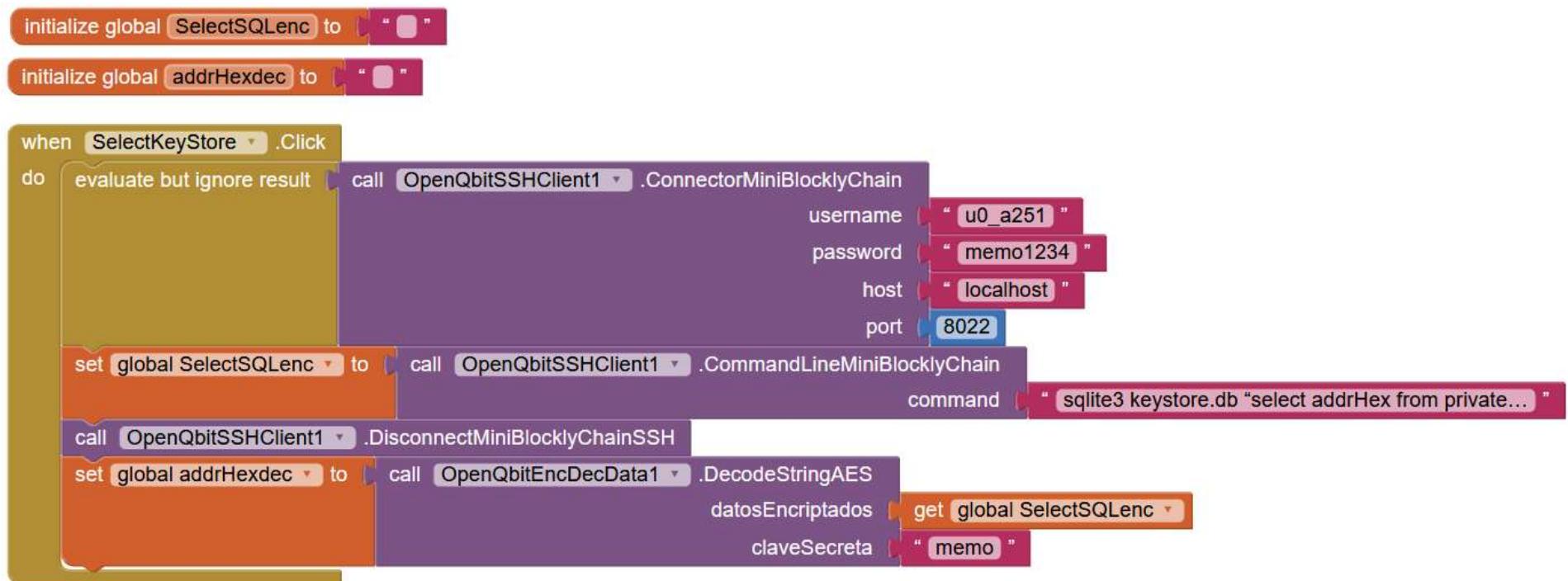
sqlite3 **writefile** ('PrivateKey.key', *addrBin*) **FROM** **privatekey** **WHERE** alias='mexico'; (2)

- (1) Dit type vraag zal de belangrijkste zijn om de privé-sleutel te raadplegen om het proces van het ondertekenen van de transacties uit te voeren.

Query for Decryption of Data by Alias in de addrHex kolom van de **privatekey** tabel van de SQLite **keystore.db** database

In dit geval gebruiken we het blok (**DecodeStringAES**) om de gegevens die zijn opgeslagen in de kolom "addrHex" te decoderen.

sqlite3 keystore.db "selecteer addrHex van privatekey where='mexico';".



Deze raadpleging geeft ons de privé-sleutel in hexadecimaal formaat, dit is het fundamentele onderdeel van elke ontvangst of verzending van transacties. Het wordt herhaaldelijk aanbevolen om een back-up van deze keystore.db database te bewaren.

Database ontwerp publickeys.db met tafel publicaddr:

```
$ sqlite3
```

```
SQLite versie 3.32.2 2020-06-20 15:25:24
```

```
Voer ".help" in voor gebruikstips.
```

```
Verbonden met een tijdelijke in-memory database.
```

```
Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.
```

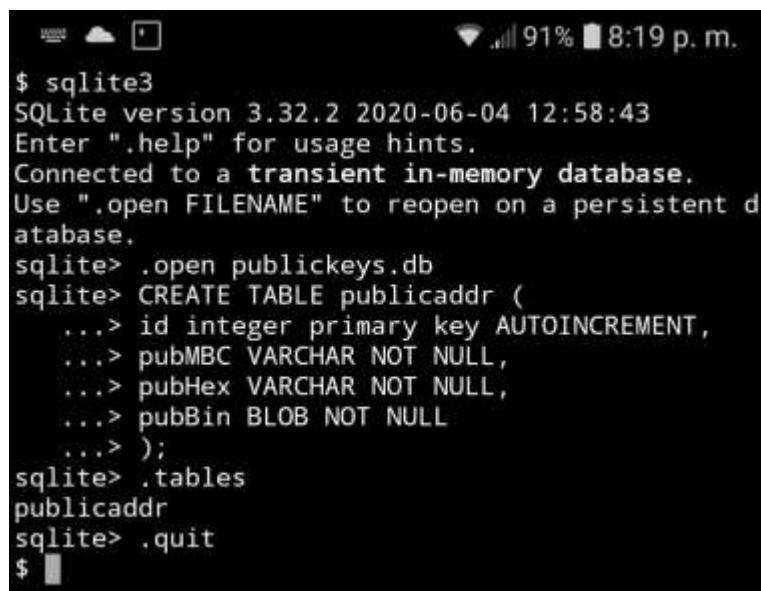
```
sqlite> .open publickeys.d iem.
```

```
sqlite> CREATE TABLE publicaddr (id integer primary key AUTOINCREMENT NOT NULL,  
pubMBC VARCHAR NOT NULL, pubHex VARCHAR NOT NULL, pubBin BLOB NOT NULL);
```

```
sqlite> .quit
```

Vervolgens wordt de aanmaak van de **gepubliceerde** tabel getoond met de volgende SQL-zin in gesegmenteerde vorm:

```
sqlite> CREATE TABLE publishedaddr (  
...> id integer primaire sleutel AUTOINCREMENT  
...> pubMBC VARCHAR NOT NULL,  
...> pubHex VARCHAR NOT NULL,  
...> pubBin BLOB NOT NULL  
...> );  
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 window displays the following text:

```
$ 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. Bijlage "RESTful SQLite GET/POST-commando's".

Vervolgens worden de verschillende commando's getoond die we kunnen gebruiken in de Restful SQLite van het backup netwerk. Deze zijn geraadpleegd uit de oorspronkelijke ontwikkeling van GITHUB (<https://github.com/olsonpm/sqlite-to-rest>)

CRUD-bewerkingen (Aanmaken, lezen, bijwerken en verwijderen) RESTful.

Hieronder volgt een lijst van beschikbare bewerkingen die door de RESTful API ter beschikking worden gesteld in de vorm van pseudo-voorbeelden. We gaan uit van een basis "op.sqlite" en twee bijbehorende "trans"-tabellen voor transacties en een ander "teken" voor bronadres, bestemmingsadres en vermogenswaarde met twee kolommen id INTEGER PRIMARY KEY.

Zoals vermeld in de beperkingen, moet u er rekening mee houden dat de methoden (DELETE en POST) slechts één rij per keer kunnen beïnvloeden.

OBTAIN Dit maakt de grootste variatie mogelijk. Later zullen we alle beschikbare query operators zien.

Koppen kunnen net onder de URL's worden opgegeven.

/transAanvragen

voor alle rijen

/trans? id=1Waar

id=1

/trans

bereik: rijen=0-2Eerste
drie rijen

/trans

bereik: rijen=-5Laaste
vijf rijen

/trans

bereik: rijen=0-

Zoveel rijen als de server kan leveren, wat in de praktijk het minste zal zijn van maxRange en het totale aantal rijen.

/trans

bereik: rijen=1-

Zoveel rijen als de server kan leveren, beginnend bij rij 1.

/trans

volgorde: naamGeordend op
naam in oplopende volgorde

/trans

volgorde: naam aflopendGeschikt
op naam aflopend

/trans

Volgorde: naam aflopend, id

Bijgedragen, maar eerst gesorteerd op aflopende naam, en in het geval van een stropdas door oplopende id.

/trans? id>1

Waar ga je heen > 1

/trans? id>=2&id<5

Waar id >= 2 en id < 5

/trans? naam_NOTNULL

Waar de naam niet nietig is

/trans? name_ISNULLWhere

name is null

/trans? id!=5&name_LIKE'Spotted%'.

Waar id != 5 en de naam is LIKE "Spotted%" (aanhalingsstekens negeren)

/trans? id>=1&id<10&name_LIKE'Avery%'.

Volgorde: naam aflopend, id-bereik: rijen=2-4

Bij wijze van voorbeeld.

Verkrijgen van transactie met identifiers tussen 1 en 9 inclusief, met een naam als "Avery%", eerst geordend door aflopende naam en vervolgens door oplopende identifier, waardoor de derde tot en met vijfde rij van het resultaat wordt verkregen. Of in SQL:

DELETE Vereist een query string met alle primaire toetsen gelijk aan een waarde. Dit vereist een maximale verwijdering van één rij.

/trans?

id=1Deletes

trans met id=1

Als de transactie in plaats daarvan een hoofdsleutel had die bestond uit id en naam.

/trans? id=1&naam='Avery IPA'.

POST creëren

Je moet niet door een querystring heengaan. Als een querystring wordt gepasseerd, wordt een POST-update verondersteld. Alle POST-verzoeken moeten het header-inhoudstype passeren: applicatie / json.

Houd er rekening mee dat het lichaam alle niet-annulerbare PRIMARY KEY-kolommen moet bevatten en niet INTEGRATE. In het andere geval wordt er een antwoord gestuurd van 400 personen, waarin wordt aangegeven welke velden verloren zijn gegaan. De nullable-kolommen zullen nul zijn en de INTEGER PRIMARY KEY-kolommen zullen automatisch worden verhoogd volgens de sqlite3 specificaties.

De JSON-gegevens worden net onder de URL's gespecificeerd.

/trans

```
{"id":1,"naam":"Serendipity"}Creëert  
een trans met id = 1 en naam = "Serendipity".
```

/trans

```
{"id":1}Creëert  
een trans met id = 1 en naam = NULL
```

/trans

```
{"naam": "Serendipity"}
```

Creëert een transactie met id ingesteld op de volgende waarde verhoogd met sqlite3 Specificaties INTEGER PRIMARY KEY.

/trans

```
{}
```

Creëert een transactie met verhoogde id en de naam ingesteld op NULL.

POST-update

Het moet een querystring bevatten. Zonder een querystring wordt uitgegaan van POST-creatie. Net als bij POST-creatie is het kopje inhoudstype: applicatie / json vereist.

De querystring moet alle primaire sleutels bevatten om ervoor te zorgen dat slechts één rij wordt bijgewerkt. Als er onjuiste waarden worden doorgegeven, wordt er een 400 teruggegeven met de beleidende toetsen.

Het lichaam van de aanvraag moet een niet-leeg voorwerp bevatten en moet geldige sleutels bevatten die overeenkomen met de namen van de kolommen.

De JSON-gegevens worden net onder de URL's gespecificeerd.

/trans?

```
{"id":2}
```

OpenQbit.com

id=1

Update de transactie met een ID van 1 door deze op 2 te zetten.

/trans? **id=1**

```
{ "naam" : " MCBza45Rt56cvbfdR2Swd788kj" } .
```

Update de transactie met de ID van 1 door de naam of waarde ervan in te stellen op "MCBza45Rt56cvbfdR2Swd788kj".

Als de handelsdesk in plaats daarvan een hoofdsleutel had die bestond uit id en naam.

/trans?id=1&name=MCBza45Rt56cvbfdR2Swd788kj

```
{ "naam" : " MCB3ofFG5Hj678MNb09vLdfaasx " }
```

Werk de transactie bij waarbij id één is en het adres MCBza45Rt56cvbfdR2Swd788kj is, waarbij de MCB3ofFG5Hj678MNb09vLdfaasx wordt ingesteld.

Referentie

isSqliteFile

Controleer de eerste 16 bytes van het bestand om te zien of het gelijk is aan 'sqlite formaat 3' gevolgd door een null byte.

isDirectory

Geeft als resultaat fs.statsSync gevolgd door .isDirectory

isFile

Geeft als resultaat fs.statsSync gevolgd door .isFile

GET-adviseurs

De aanvraagvoorwaarden moeten worden gemarkerd met verbindingssymbolen, bijv. id> 5 & naam = MCBza45Rt56cvbfdR2Swd788kj

Binaire operators (vereist een waarde na) Man.

=

!=

>=

<=

>

<

LIKE

LIKE is speciaal omdat het eenvoudige openings- en sluitingscitaten moet hebben. Anders zal er een fout van 400 worden gegenereerd die aangeeft waar de analyse niet kon worden voltooid en wat er werd verwacht. Zie CRUD RESTful Operations voor voorbeelden.

Individuele marktdeelnemers (moet de naam van een kolom volgen)

IS NIET KLAAR

NIET NULLES

Routerconfiguratieobject

isLadenPlainObject

Het doel van dit object is om een generieke configuratie voor de sqlite router aan te bieden. De volgende eigenschappen zijn toegestaan:

voorvoegsel: isLadenString De string wordt doorgegeven aan de optie koa-router prefix builder. De skeletserver geeft bijvoorbeeld geen prefix aan, waardoor de bier-API direct vanuit de root van het http-domein kan worden geraakt: // localhost: 8085 / trans. Als u het voorvoegsel op '/ api' zet, dan moet u verzoeken sturen naar http: // localhost: 8085 / api / trans.

allTablesAndViews: een configuratieobject in tabelvorm

De instellingen die in dit object zijn gespecificeerd, worden toegepast op alle tabellen en weergaven, optioneel overschreven door de eigenschap tabellenAndViews.

tabellenAndViews: isLadenPlainObject Het afgelopen object moet sleutels hebben die overeenkomen met de databasekolom of de namen bekijken. Anders wordt er een vriendelijke foutmelding gegeven. De waarden voor elke tabel en weergave moeten een tabellarisch configuratieobject zijn.

Tabellarisch configuratieobject

isLadenPlainObject Dit object vertegenwoordigt instellingen die kunnen worden ingesteld voor weergaven of tabellen. Het laat de volgende eigenschappen toe:

maxRange: isPositiveNumber

Standaard toepassing: 1000

Dit is het maximale bereik dat uw server toelaat om aanvragen in te dienen. Als een GET-aanvraag arriveert zonder een range header, gaat de specificatie ervan uit dat u de gehele bron wilt hebben. Als het aantal rijen dat resulteert in GET groter is dan maxRange, wordt een 416-status gereturneerd met de aangepaste max-range header. De standaardwaarde van de applicatie is bewust conservatief in de hoop dat de auteurs maxRange in overeenstemming met hun behoeften zullen instellen.

Let op: 'Infinity' is een geldig positief getal.

vlaggen: isLadenArray

Momenteel is de enige geaccepteerde indicator de string 'sendContentRangeInHEAD'. Wanneer ingesteld, zal HEAD verzoeken het bereik van de beschikbare inhoud in de vorm content-bereik: * / <max-bereik> retourneren. De reden dat het configurerbaar is, is dat het berekenen van het maximale bereik meer werk kan zijn dan het waard is, afhankelijk van de belasting van de server en de grootte van uw tafels.

Aangepaste headers

Toepassing

volgorde: deze header is alleen gedefinieerd voor GET, en kan worden beschouwd als het equivalent van sql ORDER BY. Het moet een door komma's afgebakende kolomnaam bevatten, elk naar keuze gevolgd door een spatie en de strijkers 'asc' of 'desc'. Als er onjuiste orderwaarden worden verzonden, geeft een antwoord van 400 aan welke dat zijn.

Antwoord

Niet alles is noodzakelijkerwijs maatwerk, maar elk gebruik valt buiten de specificatie en heeft daarom verduidelijking nodig.

GET

max-range: deze header wordt gereturneerd wanneer het aantal gevraagde rijen het geconfigureerde maxRange overschrijdt. Merk op dat het verzoek misschien niet de range header specificeert, maar het aantal rijen dat resulteert in die bron zal nog steeds worden geverifieerd.

inhoudelijk bereik: rfc7233 staten

Alleen de statuscodes 206 (Gedeeltelijke inhoud) en 416 (Onbevredigend bereik) beschrijven een betekenis voor Inhoudsbereik.

Wanneer sqlite-to-rest reageert met een statuscode 200, wordt de koptekst van het inhoudsbereik verzonden met het formaat 206 van <row start> - <row end> / <row count>.

Wanneer een verzoek wordt verzonden zonder een range header en het resulterende aantal rijen groter is dan maxRange, wordt een 400 gereturneerd met het bereik van de inhoud ingesteld op het 416-formaat van * / <rijtelling>.

Merk op dat deze header kan worden teruggestuurd in een HEAD reactie.

accept-order: wordt gereturneerd als de volgorde van de aanvraagkop een onjuiste syntax is of onjuiste kolomnamen heeft opgegeven. Voor meer details, zie HEAD -> aanvaardingsopdracht hieronder.

25. Bijlage "Java Code SQLite-Redis Connector".

Een code van het verband tussen de communicatie van beide SQLite-Redis-databases wordt hieronder weergegeven. In principe, zoals we kunnen zien, verandert de connector het SQLite formaat in een generieke string van de data (transacties) voor Redis om te ontvangen.

Het bovenstaande proces fungeert als een transactiewachtrigger voor alle knooppunten die verbonden zijn en die mogelijke kandidaten zijn voor het verwerken van de huidige transactiewachtrij.

Wanneer de Redis-database de transactiewachtrij van de Master-Slave-configuratie ontvangt, zal deze de informatie in realtime en gelijkelijk over alle knooppunten verdelen.

Een ander fundamenteel punt is dat alle nodes gesynchroniseerd moeten worden in hun klok, dit zal gebeuren zoals we eerder zagen met de functionaliteit van de query naar een server pool NTP (Network Time Protocol).

Deze connector voert ook het eerste niveau van databeveiligingscontrole uit door de Merkle Root tree van alle transacties te berekenen.

Basiscode van de SQLite-Redis Connector.

```
importeer java.sql.*;
importeer java.util.*;
importeer java.util.Arrays;
importeer java.util.List;
importeer java.util.ArrayList;
import java.security.*;
importeer java.security.MessageDigest;
import redis.clients.jedis.Jedis;
klasse RediSqlite{
    openbare String vorigeHash;
    openbare String merkleRoot;
    openbare statische ArrayList<String> transacties = nieuwe ArrayList<String>();
    openbare statische ArrayList<String> treeLayer = nieuwe ArrayList<String>();
    openbare statische String getMerkleRoot() {
        int count = transacties.size();
        int item = 1;
        Odd int = 0;
        ArrayList<String> previousTreeLayer = transacties;
        terwijl (telling > 1) {
```

```

treeLayer = nieuwe ArrayList<String>();
voor(int i=1; i <= tellen ; i=i+2) {

    als (!esPar(count) && i == count) odd = 1;
        treeLayer.add(applySha256(previousTreeLayer.get(i-item)
previousTreeLayer.get(i-impar)));
    }

    item = 1;
    Oneven = 0;
    telling = treeLayer.size();
Voorafgaand aan treeLayer = treeLayer;
}

String merkleRoot = (treeLayer.size() == 1) ? treeLayer.get(0) : "";
terug te keren merkleRoot;
}

//Defineer of Merkle Tree even of oneven is
statische booleaanse enPar(int nummer){
    als (nummer%2==0) waar is; anders is het foutief;
}

openbare statische String applySha256(String input){
Probeer {
    MessageDigest digest = MessageDigest.getInstance ("SHA-256");
    //Aangepast sha256 aan onze input,
    byte[] hash = digest.digest(input.getBytes("UTF-8"));
    StringBuffer hexString = nieuwe StringBuffer(); // Deze bevat hash als hexidecimaal
    voor (int i = 0; i < hash.lengte; i++) {
        String hex = Integer.toHexString(0xff & hash[i]);
        if(hex.length() == 1) hexString.append('0');
        hexString.append(hex);
    }
    hexString.toString() terug te geven;
}
vangst (uitzondering e) {
    gooie nieuwe RuntimeException(e);
}
}

openbare statische nietige hoofdleiding (String args[]){
    Probeer het maar eens.

```

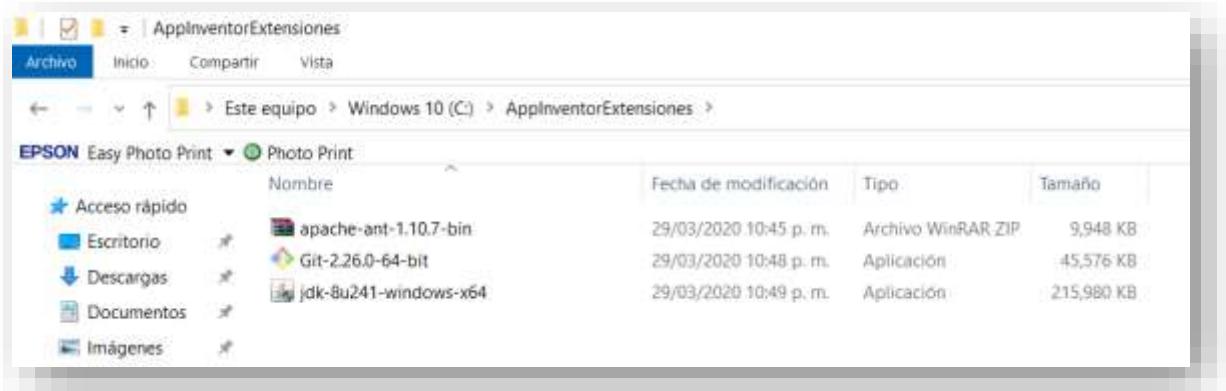
```
Aansluiting      con=DriverManager.getConnection("jdbc:sqlite:C://memo/sqlite-tools-
win32-x86-3310100/trans.sqlite3");
//Connecting to Redis server on localhost
Jedis = nieuwe Jedis ("localhost");
jedis.auth("memo1234");
Verklaring stmt=con.createStatement();
String sql = "SELECT * VAN BROUWERIJ";
ResultaatSet rs=stmt.executeQuery(sql);
terwijl(rs.next()) {
    transactions.add(rs.getString(2));
jedis.set
("LATAM:"+String.valueOf(rs.getInt(1))","["+""+"+"+"+"+", "+""+"+"+"+"+"+".getString(3)+"+","+"
"+""+"+"+"+".getString(4)+"+");
}
jedis.set("LATAM:merkleroot", getMerkleRoot());
System.out.println("Aantal ArrayList elementen: "+treeLayer.size());
met .close();
}catch(Exception e){ System.out.println(e);}

}
```

26. Bijlage "Mini BlocklyChain voor ontwikkelaars".

In deze bijlage zullen we de configuratie, de installatie en het basisgebruik van het genereren van externe modules op basis van de Java-programmeertaal voor de Blockly-omgeving bekijken en zullen we in staat zijn om gespecialiseerde modules te creëren voor elke business case en functionaliteiten toe te voegen aan het Mini BlocklyChain-systeem of andere functionaliteiten toe te voegen aan onze mobiele applicatie.

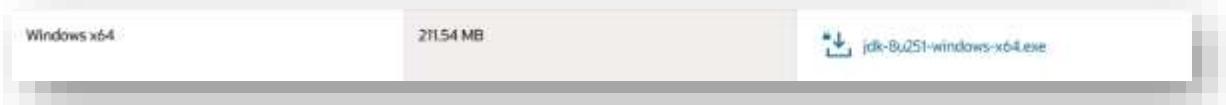
We hebben in ons Windows systeem een directory aangemaakt met de naam "AppInventorExtensions" en deze zal de volgende publieke softwarepakketten downloaden.



1.- We gaan de laatste versie van de **JDK (Java Development Kit)** downloaden.

voorbeeld: jdk-8u251-ramen-x64.exe (211 MB)

<https://www.oracle.com/java/technologies/javase/javase8-downloads.html>



2.- **Apache** Mierenbibliotheek die JAVA gebruikt om applicaties te bouwen, <http://ant.apache.org/bindownload.cgi>, in mijn geval heb ik mier 1.10.8 (Binary Distributions) gedownload (apache-ant-1.10.8-bin.zip). Er kunnen meer geavanceerde versies zijn.

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.

3.- We hebben de Git Bash van uw site <https://git-scm.com/download/win> geïnstalleerd.

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.- Unzip "Apache Ant." Als je klaar bent met het uitpakken, kun je het voorbeeld in een dubbele map doen:

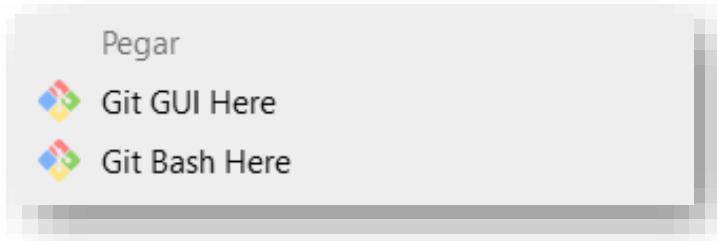
C:\AppInventorExtensions\apache-ant-1.10.8-bin\apache-ant-1.10.8-bin

- Verander het in C:\Apache-ant-1.10.8-bin

	Nombre	Fecha de modificación	Tipo	Tamaño
o	bin	01/09/2019 11:43 a. m.	Carpeta de archivos	
o	etc	01/09/2019 11:43 a. m.	Carpeta de archivos	
o	lib	01/09/2019 11:43 a. m.	Carpeta de archivos	
o	manual	01/09/2019 11:43 a. m.	Carpeta de archivos	
o	CONTRIBUTORS	01/09/2019 11:43 a. m.	Archivo	7 KB
o	contributors	01/09/2019 11:43 a. m.	Documento XML	33 KB
o	fetch	01/09/2019 11:43 a. m.	Documento XML	14 KB
o	get-m2	01/09/2019 11:43 a. m.	Documento XML	5 KB
o	INSTALL	01/09/2019 11:43 a. m.	Archivo	1 KB
o	KEYS	01/09/2019 11:43 a. m.	Archivo	94 KB
o	LICENSE	01/09/2019 11:43 a. m.	Archivo	15 KB
o	NOTICE	01/09/2019 11:43 a. m.	Archivo	1 KB
o	patch	01/09/2019 11:43 a. m.	Documento XML	2 KB
o	README	01/09/2019 11:43 a. m.	Archivo	5 KB
o	WHATSNEW	01/09/2019 11:43 a. m.	Archivo	250 KB

5.- We hebben Git Bash geïnstalleerd. We hebben alles standaard in de installatie laten staan.

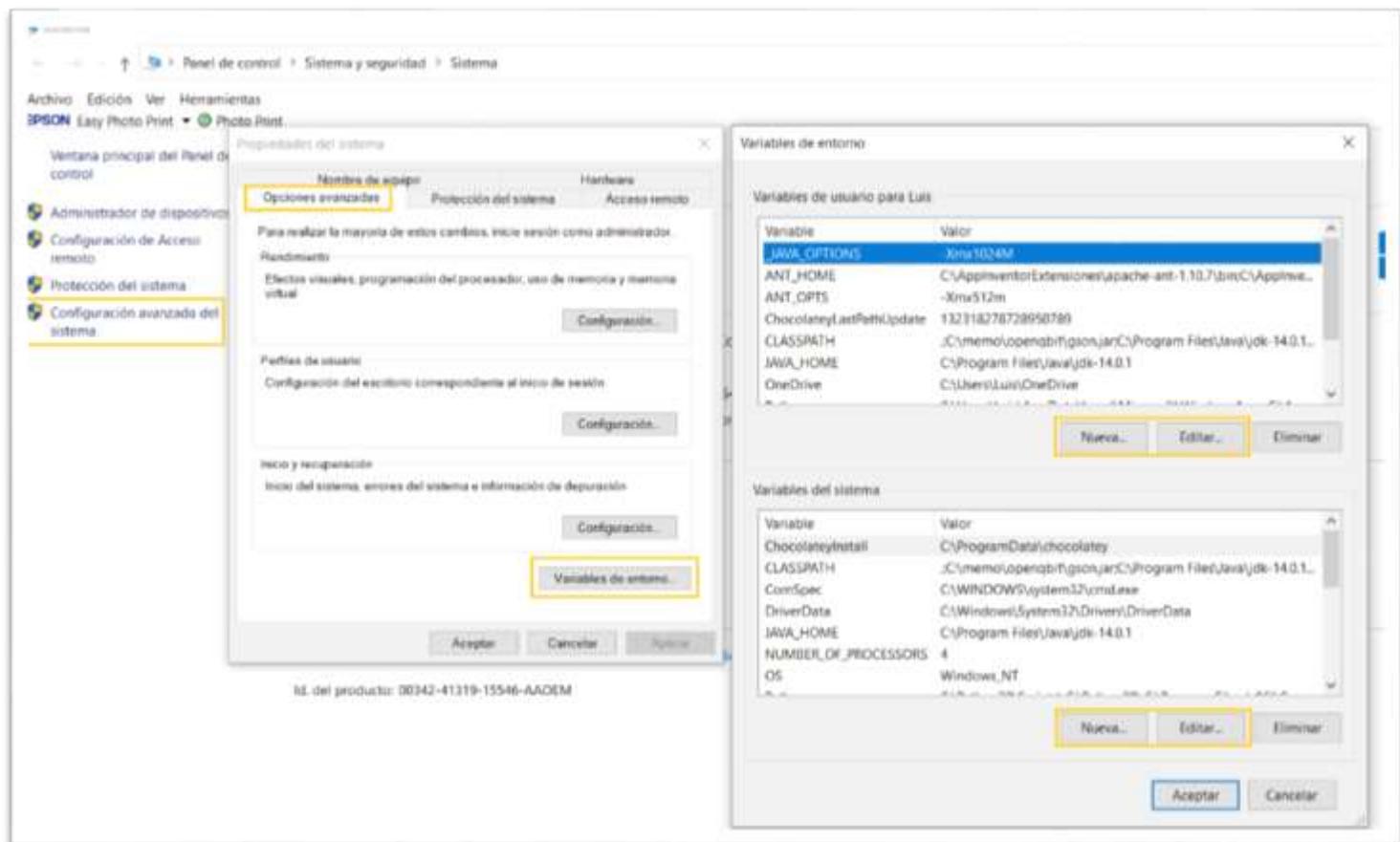
- Later kunnen we zien dat het een link heeft in de Windows Start knop, door te klikken op de linkse knop in de bestandsbrowser.



6.- We hebben de Java SE Development Kit geïnstalleerd. Afhankelijk van de versie van Windows moet u waarschijnlijk uw computer opnieuw opstarten.

Windows-systeem omgevingsvariabelen. We zullen de omgevingsvariabelen creëren. Dat zullen we doen:

Bedieningspaneel -> Systeem -> Geavanceerde systeeminstellingen -> Geavanceerde opties -> Omgevingsvariabelen.



Afhankelijk van of we een nieuwe variabele willen plaatsen of een bestaande willen bewerken, drukken we op de knop "New..." of "Edit..." .

Om adressen toe te voegen aan de reeds vastgestelde adressen, scheiden we ze per puntkomma;

In de sectie Gebruikersvariabelen voor John, zetten we deze nieuwe...

JAVA_OPTIONS we zetten u van waarde -Xmx1024m

ANT_HOME we zetten waarde C:\\\\nventorExtensions\\apache-ant-1.10.8-bin [dat wil zeggen, de map waar we apache-ant hebben gedecomprimeerd]

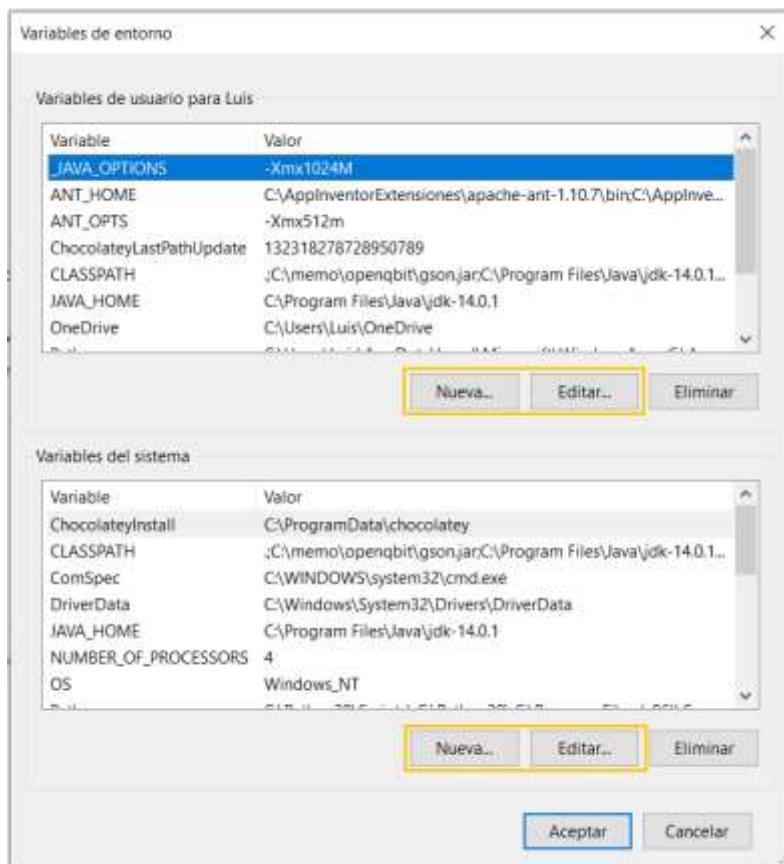
ANT_OPTS we zetten van waarde -Xmx256M

JAVA_HOME is ingesteld op C:\\\\programmabestanden\\\\jdk1.8.0_131 [Als het een andere waarde heeft, wijzig deze dan. Merk op dat het jdk NOT jdr is].

CLASSPATH we zetten van de waarde %ANT_HOME%\\lib;%JAVA_HOME%\\lib

In PATH hebben we ;%ANT_HOME%\\bin;%JAVA_HOME%\\bin [Merk op dat ; begint met een puntkomma; om toe te voegen aan de puntkomma's die er al zijn].

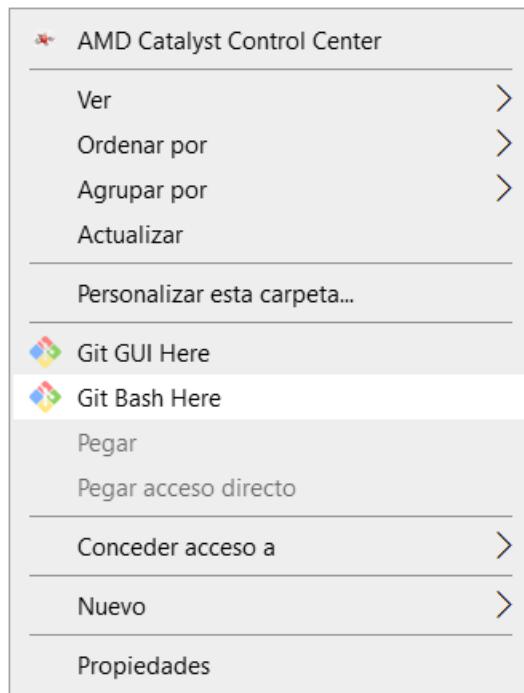
OPMERKING: Variabelen worden van elkaar gescheiden door een puntkomma: variabele-één; variabele-n; variabele-n+1



7.- Aanmaken van de App Inventor kloon in onze computer

Wij maken een "kloon" kopie van App Inventor op onze server (PC), downloaden deze direct van het internet en maken die kopie.

Om dit te doen, gebruiken we de **Git Bash** applicatie en klikken we erop om een terminal te openen.



We voeren het commando op de Git Bash terminal:

```
$ git kloon https://github.com/mit-cml/appinventor-sources.git
```

A screenshot of a terminal window titled 'MINGW64 /home/llgpl'. The command 'git clone https://github.com/mit-cml/appinventor-sources.git' is entered and its output is displayed. The output shows the progress of cloning the repository, including the number of objects being transferred, the size of the transferred files, and the transfer rate. The process is completed successfully with a message 'Checking out files: 100% (308K/308K), done.'

De plaats waar de opslagplaats zich bevindt:

<https://github.com/mit-cml/appinventor-sources/>

Het zal een map aanmaken genaamd appinventor-sources met de App Inventor bron.

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

We hebben de volgende directory aangemaakt in het pad C: Gebruikers - Appinventor-sourcesv-babkup - Appinventor-componenten -rc-openqbit

Binnenin hebben we ons volgende testprogramma, OpenQbitQRNGch.java, gekopieerd

Nombre	Fecha de modificación	Tipo	Tamaño
OpenQbitQRNGch	11/06/2020 01:39 a. m.	Archivo JAVA	5 KB

Creatie van de uitbreiding.

RESPECT THE CAPS AND MINUSCULES, het is niet hetzelfde Hallo en Hallo.

Leg geen accenten. We zijn klaar om onze eerste uitbreiding te maken. Het zal de quantum random number generator zijn.

We gebruiken een Text Editor, Notepad++, we maken een bestand genaamd OpenQbitQRNGch.java dat willekeurige kwantum getallen genereert met de volgende code:

```
// Deze uitbreiding wordt gebruikt om Quantum Random Number Generator
QRNG Zwitserland API uit te voeren.

pakket com.openqbit. OpenQbitQRNGch;
com.google.appinventor
components.importeren.annotations.DesignerComponent;
importeren.com.google.appinventor.components.annotations.DesignerProperty
;
importeren.com.google.appinventor.components.annotations.PropertyCategory
;
importeren.com.google.appinventor.components.annotations.SimpleEvent;
importeren.com.google.appinventor.components.annotations.SimpleFunction;
importeren.com.google.appinventor.components.annotations.SimpleObject;
importeer com.google.appinventor.components.annotations.SimpleProperty;
importeren.com.google.appinventor.components.common.ComponentCategory;
importeer com.google.appinventor.components.common.PropertyTypeConstants;
importeer.google.appinventor.components.runtime.use.MediaUtil;
import com.google.appinventor.components.runtime.*;
import java.io.*;

@DesignerComponent(versie = OpenQbitQRNGch.VERSIE,
    beschrijving = "API Quantum Random Number Generator. Quantum random
numbers as a service, QRNGaaS, web API voor de quantum random number
generator van Quantis ontwikkeld door het Zwitserse bedrijf - " +
"Guillermo Vidal - OpenQbit.com",
    categorie = ComponentCategorie.EXTENSIE
    Onzichtbaar = waar,
    icoonNaam = "http://www.pinntar.com/logoQbit.png")
@SimpleObject (extern = waar)

publieke klasse OpenQbitQRNGch breidt AndroidNonvisibleComponent
implementeert Component {

    publiek statisch eindresultaat in VERSIE = 1;
    openbare statische finale String DEFAULT_TEXT_1 = "";
    private ComponentContainer container;
    private String text_1 = "";

    openbare OpenQbitQRNGch (ComponentContainer container) {
        super(container.$form());
        this.container = container;
    }

    // Oprichting van de Eigenschappen.
    //@DesignerProperty(editorType =
    PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue =
    OpenQbitQRNGch.DEFAULT_TEXTO_1 + "")
    //@SimpleProperty (beschrijving = "API Get Quantum Random Number
Generator, random number tussen 0 en 1. Voer variabele *kwantiteit* van
te genereren getallen in")
```

```
@SimpleFunction (beschrijving = "API Get Quantum Random Number Generator, random number tussen 0 en 1. Voer het variabele gehele getal *kwantiteit* van de te genereren getallen in")
openbare String APIGetQRNGdecimale (String qty) gooit Uitzondering {
    String url = "http://random.openqu.org/api/rand?size=" + qty;
    String[] commando = {"curl", url};

    ProcessBuilder proces = nieuwe ProcessBuilder(opdracht);
    Proces p;
    p = proces.start();
    BufferedReader reader = nieuwe BufferedReader (nieuwe
InputStreamReader(p.getInputStream()));
    StringBuilder bouwer = nieuwe StringBuilder();
    Koordlijn = nihil;
    terwijl ((regel = reader.readLine()) != null) {
        builder.append(line);

        builder.append(System.getProperty("line.separator"));
    }
    String resultaat = builder.toString();
    //System.out.print(resultaat);
    retourresultaat;

}

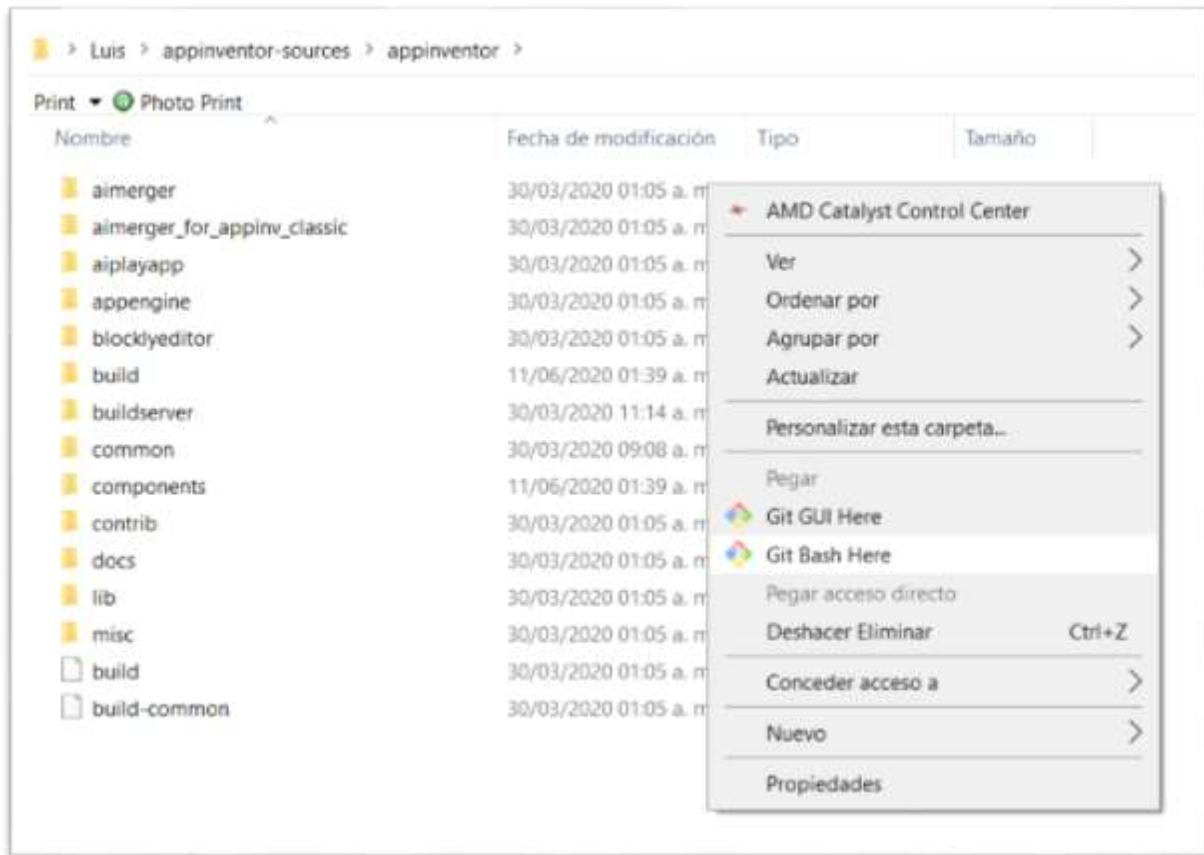
@SimpleFunction(description = "API Get Quantum Random Number Generator, random number between a range min to max numbers. Voer het variabele gehele getal *kwantiteit* van getallen in om een bereik van minimaal *min* aantal en maximaal *max* aantal te genereren")
publiek String APIGetQRNGinteger (String qty, String min, String max)
gooit Uitzondering {
    String url = "http://random.openqu.org/api/randint?size=" + qty +
"&min=" + min + "&max=" + max.
    String[] commando = {"curl", url};

    ProcessBuilder proces = nieuwe ProcessBuilder(opdracht);
    Proces p;
    p = proces.start();
    BufferedReader reader = nieuwe BufferedReader (nieuwe
InputStreamReader(p.getInputStream()));
    StringBuilder bouwer = nieuwe StringBuilder();
    Koordlijn = nihil;
    terwijl ((regel = reader.readLine()) != null) {
        builder.append(line);

        builder.append(System.getProperty("line.separator"));
    }
    String resultaat = builder.toString();
    //System.out.print(resultaat);
    retourresultaat;

}
}
```

We voeren de **Git Bash** uit door ons te positioneren op het volgende pad: **C: Luis-appinventor-sources-appinventor**. In deze directory klikken we op de linker muisknop en selecteren we de **Git Bash** terminal.



De Git bash terminal zal in de volgende directory worden gepositioneerd:

C: Gebruikers-Luisuitvinders-bronnenuitvinder (meester)

```
MINGW64:/c/Users/Luis/appinventor-sources/appinventor
Luis@DESKTOP-LLGPLR6 MINGW64 ~ /appinventor-sources/appinventor (master)
$ |
```

In de Git Bash terminal en voer het volgende commando uit:

\$ mierenextensies

```
MINGW64/C:/Users/Luis/appinventor-sources/appinventor
luis@DESKTOP-LLGP8R: 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_ConnectVersion:
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
```

Als alles goed gaat, krijgen we het: MET SUCCES TE BOUWEN.

Onze uitbreiding is gemaakt in...

C: Gebruikers-uitvinders-brons-uitvinders-onderdelen bouwen...

OPMERKING: de inhoud van de extensiemap wordt telkens als we een mierenextensie maken, verwijderd en opnieuw gemaakt.

```
MINGW64/C:/Users/Luis/appinventor-sources/appinventor
luis@DESKTOP-LLGP8R: MINGW64 ~/appinventor-sources/appinventor (master)
$ ant extensions
JarException:
[jar] Building jar: C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents-class/com.openqbbit.OpenQbitQRNGch.jar
PreGuard:
[copy] Copying 1 File to C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents/com.openqbbit.OpenQbitQRNGch/files
[echo] Generated build file AndroidRuntime.jar (com.openqbbit.OpenQbitQRNGch)
dexAllExtensions:
dexExtension:
[java] Picked up _JAVA_OPTIONS: -Xmx1024m
[echo] Dexing extension: com.openqbbit.OpenQbitQRNGch
extensions:
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build/extensions
packExtensions:
[zip] Building zip: C:/Users/Luis/appinventor-sources/appinventor/components/build/extensions/com.openqbbit.OpenQbitQRNGch.aix
BUILD SUCCESSFUL
Total time: 37 seconds
luis@DESKTOP-LLGP8R: MINGW64 ~/appinventor-sources/appinventor (master)
$
```

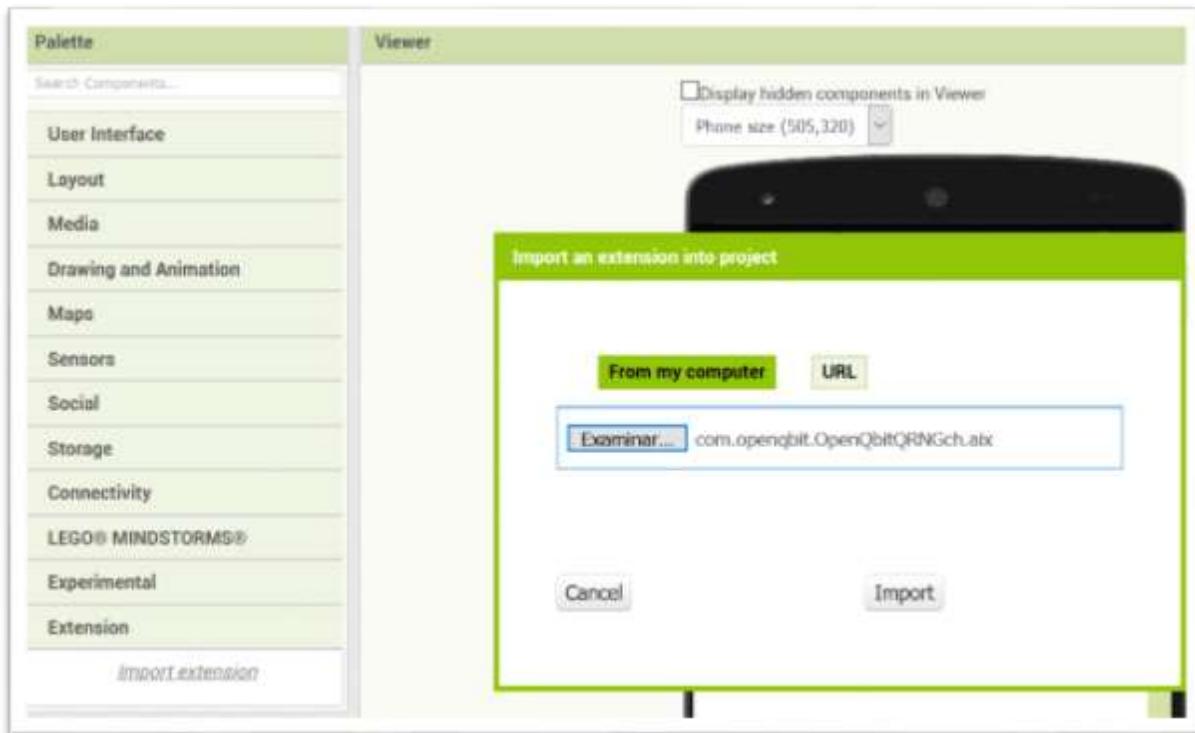
Laten we naar die map gaan:

C: Gebruikers-uitvinders-brons-uitvinders-onderdelen bouwen...

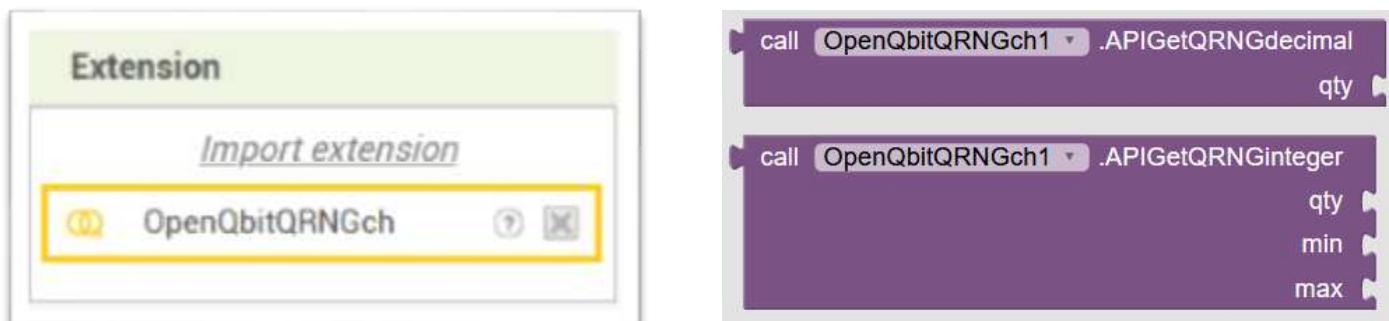
en kopieer het bestand com.openqbit.OpenQbitQRNGch.aix, dit bestand is onze extensie.

Omdat we al een account hebben in App Inventor of een ander Blockly systeem hebben we de nieuwe extensie toegevoegd en getest.

Ga naar de onderkant waar de Extensie optie is en klik op "Import extensie":



We drukken op de "Import" knop. Nu hebben we de blokken (**APIGetQRNGdecimaal**) en (**APIGetQRNGinteger**) om te gebruiken:



We hebben al onze eerste gecreëerde en functionele uitbreiding.

27. Bijlage "BlocklyCode Slimme Contracten".

BlocklyCodes zijn programma's die in java-taal zijn gemaakt. De uitbreiding om dit soort programma's te creëren, die gewoonlijk "Smart Contracts" worden genoemd, is een manier om overeenkomsten tussen gebruikers (bedrijven of mensen) te verwerken.

Dit blok (**BlocklyCode**), wordt geïmplementeerd in een programma dat al parameters heeft vastgelegd en dat afhankelijk van het type overeenkomsten die door het Mini BlocklyChain systeem op een automatische manier kunnen worden uitgevoerd wanneer de gebouwen die het "Slimme Contract" regelen, worden voldaan.

De parameters die in aanmerking komen voor de voorgestelde of de invoerparameters "input" zijn

Vervaldatum

Gerefereerde datum

Afloopijd

Gerefereerde tijd

Gerefereerd activum

Totaal vaste activa

Variabele activa

Contractleden

Bevestigde gegevens (String)

Bevestigde gegevens (bestandsnaam)

Variabele gebeurtenis

Vaste gebeurtenis

Validatie van document(en)

String Validatie

Handtekening geldig

Gedefinieerd interval (Integer)

Decimaal interval

Vastgesteld minimum

Vastgesteld maximum

Voordat we het blok (**BlocklyCode**) gaan gebruiken moeten we eerst het systeem installeren dat de uitvoering van de "Smart Contracts" gaat uitvoeren, dit gebeurt door installatie in de terminal van Termux OpenJDK en OpenJRE.

OpenJDK (Open Java Development Kit). - Het is het gereedschap om programma's in java te ontwikkelen, het bevat de bibliotheken, de compiler en de JVM (Java Virtual Machine).

OpenJRE (Open Java Runtime Environment). - Dit is de tool voor het uitvoeren van alleen java programma's. Het bevat de JVM (Java Virtual Machine).

We gaan verder met de installatie van OpenJRE en OpenJDK in de Termux terminal van de knooppunten die het Mini BlocklyChain systeem vormen.

We hebben de kamers geïnstalleerd

\$ apt installeren - en wget



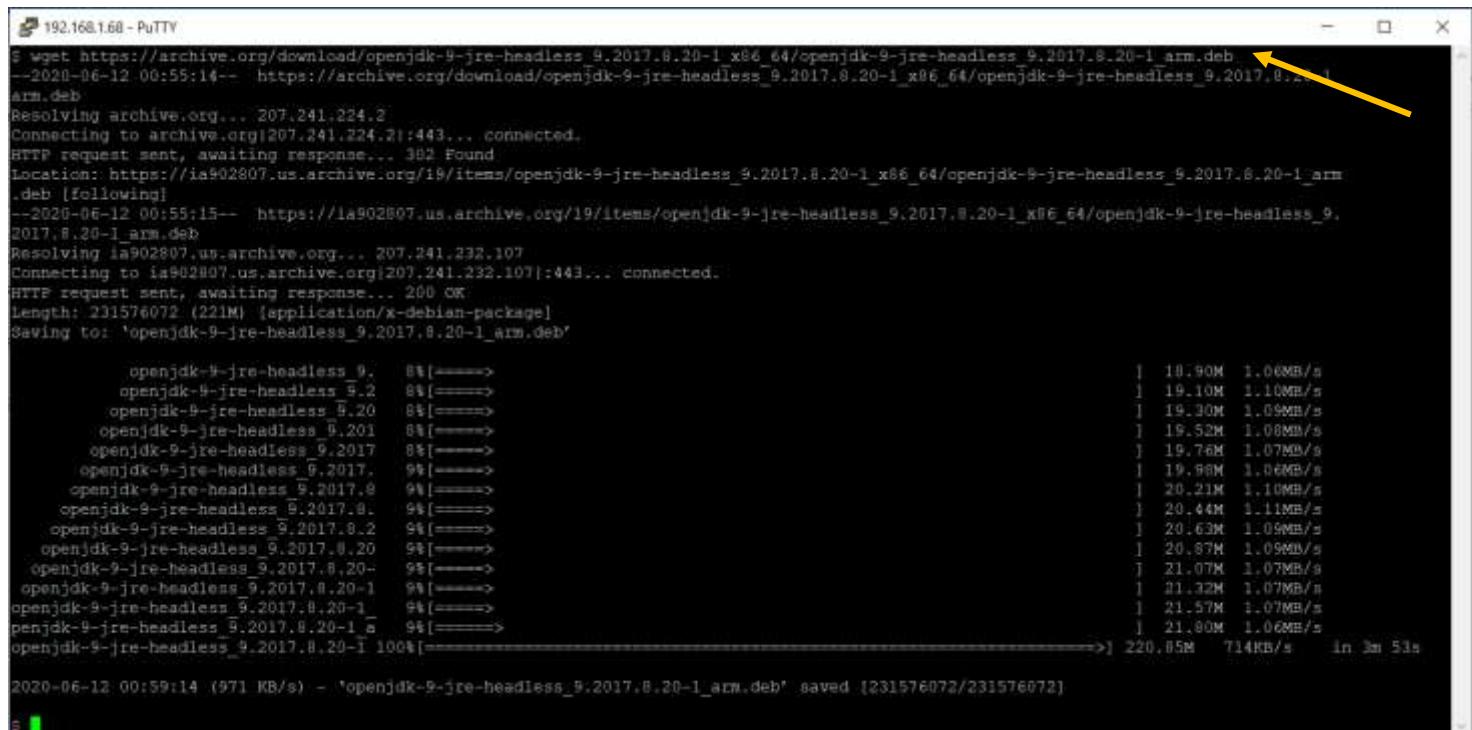
A screenshot of a Termux terminal window. The terminal shows the output of the command `sudo apt install wget`. The output includes package lists, dependency building, and the download and unpacking of the wget package from a URL. A yellow arrow points to the top right corner of the terminal window, which displays the time as 1:07 a.m. Below the terminal is a virtual keyboard.

```
[root@localhost ~]# 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-packages-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 directories
currently installed.)
Preparing to unpack .../archives/wget_1.20.3-2_arm.deb ...
Unpacking wget (1.20.3-2) ...
Setting up wget (1.20.3-2) ...
[  ]
```

We hebben de OpenJRE

gedownload

\$ 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



```

$ 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://ia802807.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://ia802807.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 ia802807.us.archive.org... 207.241.232.107
Connecting to ia802807.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'

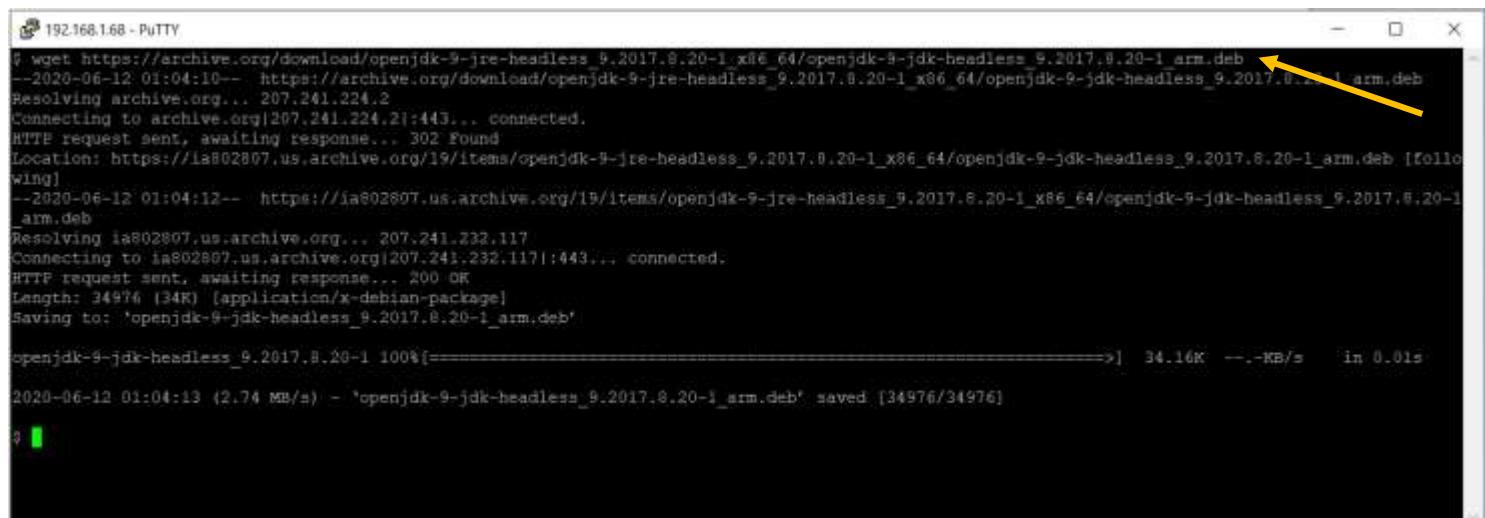
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  18.90M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  19.10M  1.10MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  19.30M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  19.52M  1.08MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  19.76M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  19.98M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  20.21M  1.10MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  20.44M  1.11MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  20.63M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  20.87M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  21.07M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  21.32M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  21.57M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  21.80M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  220.05M  714KB/s  in 3m 51s

2020-06-12 00:59:14 (971 KB/s) - 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb' saved [231576072/231576072]

$
```

We hebben de OpenJDK gedownload

\$ 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



```

$ 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'

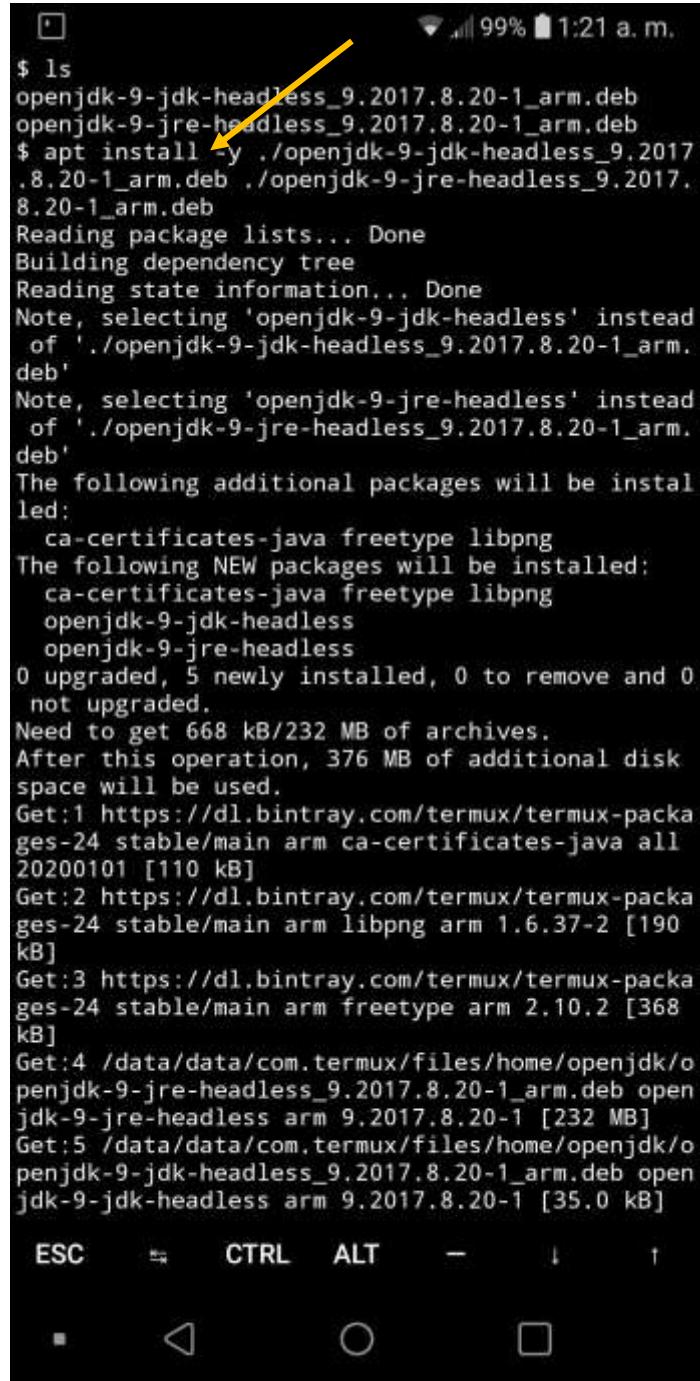
openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb          100%[=====]  34.16K  --.-KB/s  in 0.01s

2020-06-12 01:04:13 (2.74 MB/s) - 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb' saved [34976/34976]

$
```

Wij voeren de installatie uit vanaf de Termux-terminal van OpenJDK en OpenJRE:

```
$ apt installen - en ./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-packa
ges-24 stable/main arm ca-certificates-java all
20200101 [110 kB]
Get:2 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm libpng arm 1.6.37-2 [190
kB]
Get:3 https://dl.bintray.com/termux/termux-packa
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]
```



```
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-certi
ficates-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)
...
$ █
```

Sinds we klaar zijn met de installatie van de OpenJDK en OpenJRE omgeving. Deze installaties bevatten de JVM (Java Virtual Machine) die de omgeving zal zijn die de BlocklyCode zal uitvoeren.

Nu zullen we een editor installeren om een bestand online aan te maken, we zullen de editor genaamd **vi** gebruiken om het volgende commando uit te voeren:

\$ apt installeer vim

Laten we nu proberen een testbestand samen te stellen en uit te voeren. We maken in de Termux terminal met de **vi** editor en schrijven de java code van een "Hello World", en slaan deze op in het HelloWorld.java bestand.

```
openbare klasse HelloWorld {  
    openbare statische nietige hoofdleiding (String[] args) {  
        // Drukt "Hallo, Wereld" af op het terminalvenster.  
        System.out.println("Hallo, Wereld");  
    }  
}
```

Vervolgens draaien we het volgende commando in de Termux-terminal voor de compilatie en vervolgens de uitvoering van het programma:

\$ javac HelloWorld.java (Na afloop wordt een bytecode HelloWorld.class bestand aangemaakt)

\$ java HelloWorld (Na het uitvoeren en afdrukken van de advertentie, Hello World)

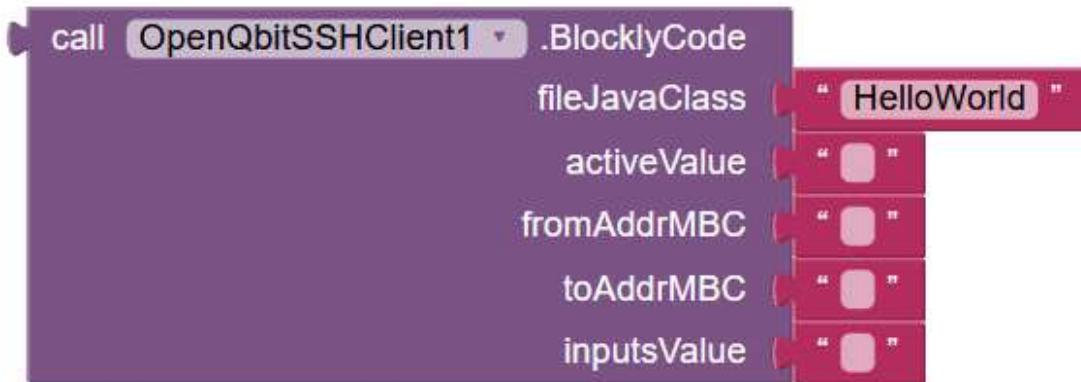


A screenshot of an Android device's terminal application. The screen shows the following command-line session:

```
$ javac HelloWorld.java
$ ls
HelloWorld.class  HelloWorld.java
$ java HelloWorld
Hello, World
$
```

The terminal window has a black background with white text. At the top, there are standard Android status icons (signal strength, battery level at 98%, time 1:57 a.m.).

Met behulp van het blok (**BlocklyCode**) bevindt dit blok zich in de uitbreiding



(ConnectorSSHClient).

In het vorige blok kunt u zien hoe het HelloWorld.class bestand wordt uitgevoerd, dat al is gecompileerd en gepositioneerd in de gebruikersbasisdirectory.

Het is een blok waarin het mogelijk is om een programma uit te voeren dat al in java is gecompileerd, wat in de ontwikkelomgeving algemeen bekend staat als het bytecodebestand in zijn binaire vorm.

Dit type bestand is klaar om te worden uitgevoerd door de Java Virtual Machine (JVM).

Er zijn twee manieren om een bytecodebestand te maken of met de extensie .CLASS, de gebruiker kan het in zijn PC maken door comfort of zijn soortgenoten kan ook worden gedaan in de mobiele telefoon, vergeet niet dat we al eerder de omgeving hebben geïnstalleerd om JDK (Java Development Kit) te compileren, hoewel het minder efficiënt zal zijn omdat de beperking van het toetsenbord zal tellen, ook in het geval van het kiezen van de Android-omgeving zal rekening moeten houden met het feit dat de bibliotheken zijn beperkt tot OpenJDK die is geïnstalleerd.

De invoerparameters zijn open in <inputsValue> en de andere vaste parameters zijn die van activeValue, fromaddrMBC en toAddrMBC.

In het geval van uitvoeringstesten kunnen ze leeg gelaten worden zonder inhoud en wordt alleen het bytecodebestand zonder parameters uitgevoerd.

Het vorige blok is als de volgende opdrachtregel die wordt uitgevoerd:

\$ java HelloWorld

De programma's die voor BlocklyCode zijn ontwikkeld, zijn onderworpen aan elke specifieke ontwerpmgeving en kunnen worden gecontroleerd en uitgevoerd door middel van routine met de "cron"-agent en worden gedeeld met de syncthingmanager.

28. Bijlage "OpenQbit Quantum Computing".

Hoe werkt quantumcomputing? ⁽²⁾

De digitale transformatie brengt sneller dan ooit tevoren veranderingen in de wereld teweeg. Zou u geloven dat het digitale tijdperk ten einde loopt? **Digitale geletterdheid** is al geïdentificeerd als een gebied waar open kennis en toegankelijke mogelijkheden om over technologie te leren dringend nodig zijn om de lacunes in de sociale en economische ontwikkeling aan te pakken. Het leren van de sleutelbegrippen van het digitale tijdperk zal nog kritischer worden met de aanstaande komst van een nieuwe technologische golf die in staat is om bestaande modellen met verbazingwekkende snelheid en kracht te transformeren: de **kwantumtechnologieën**.

In dit artikel vergelijken we de basisconcepten van traditional computing en quantum computing; en we beginnen ook de toepassing ervan in andere gerelateerde gebieden te onderzoeken.

Wat zijn kwantumtechnologieën?

In de loop van de geschiedenis heeft de mens de technologie ontwikkeld zoals hij door de wetenschap heeft begrepen hoe de natuur werkt. Tussen 1900 en 1930 leidde de studie van

enkele nog niet goed begrepen natuurkundige verschijnselen tot een nieuwe natuurkundige theorie, **Quantum Mechanica**. Deze theorie beschrijft en verklaart de werking van de microscopische wereld, de natuurlijke habitat van moleculen, atomen of elektronen. Dankzij deze theorie is het niet alleen mogelijk geweest om deze fenomenen te verklaren, maar is het ook mogelijk geweest om te begrijpen dat de subatomaire werkelijkheid op een volledig contra-intuïtieve, bijna magische manier werkt, en dat er in de microscopische wereld gebeurtenissen plaatsvinden die niet in de macroscopische wereld voorkomen.

Deze **kwantumeigenschappen** omvatten kwantumsuperpositie, kwantumverstrekeling en kwantumteleportatie.

- **Quantum superpositie** beschrijft hoe een deeltje in verschillende toestanden tegelijk kan zijn.
- **Quantumverstrekeling** beschrijft hoe twee deeltjes die zo ver uit elkaar liggen als gewenst, zodanig met elkaar in verband kunnen worden gebracht dat de ander zich er bij de interactie met de ene bewust van is.
- **Kwantumteleportatie** gebruikt kwantumverstrekeling om informatie van de ene plaats naar de andere in de ruimte te sturen zonder er doorheen te hoeven reizen.

Kwantumtechnologieën zijn gebaseerd op deze kwantumeigenschappen van het subatomaire karakter.

In dit geval stelt het begrip van de microscopische wereld door middel van Quantum Mechanics ons vandaag de dag in staat om technologieën uit te vinden en te ontwerpen die het leven van mensen kunnen verbeteren. Er zijn veel en zeer verschillende technologieën die gebruik maken van kwantumfenomenen en sommige daarvan, zoals lasers of magnetische resonantie beeldvorming (MRI), zijn al meer dan een halve eeuw bij ons. Op dit moment zijn we echter getuige van een technologische revolutie op gebieden als quantumcomputing, quantuminformatie, quantumsimulatie, quantumoptiek, quantummetrologie, quantumklokken of quantumsensoren.

Wat is quantum computing? Eerst moet je de klassieke informatica begrijpen.

FIGURA 1.
Ejemplos de caracteres en lenguaje binario.

Carácter	Bits
7	111
A	01000001
\$	00100100
)	0011101000101001

Om te begrijpen hoe kwantumcomputers werken, is het handig om eerst uit te leggen hoe de computers die we dagelijks gebruiken, en die we in dit document zullen aanduiden als digitale of klassieke computers, werken. Deze gebruiken, net als de rest van de elektronische apparaten zoals tablets of mobiele telefoons, bits als de fundamentele eenheden van het geheugen. Dit betekent dat programma's en applicaties gecodeerd zijn in bits, dat wil zeggen in binaire taal van nullen en enen. Elke keer dat we met een van deze apparaten communiceren, bijvoorbeeld door een toets op het toetsenbord in te drukken, worden er reeksen nullen en enen aangemaakt, vernietigd en/of gewijzigd in de computer.

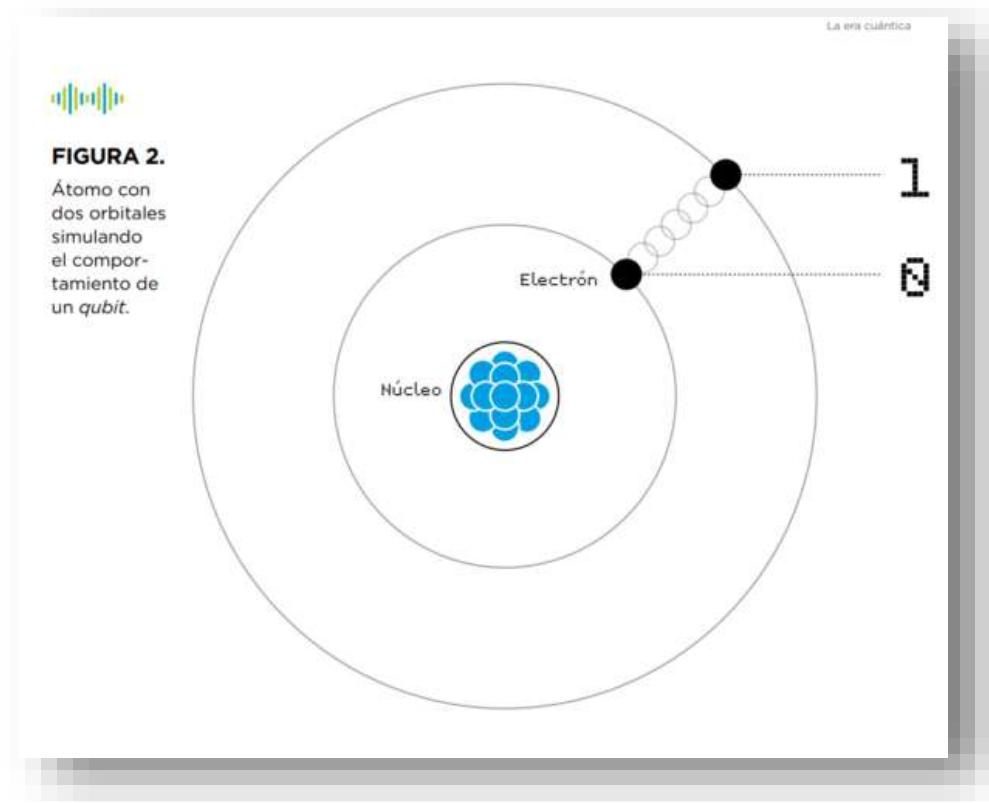
De interessante vraag is, wat zijn deze nullen en enen fysiek in de computer? De nul- en één-toestanden komen overeen met de elektrische stroom die al dan niet circuleert door microscopische onderdelen die transistors worden genoemd en die als schakelaars fungeren. Als er geen stroom loopt, is de transistor "uit" en komt hij overeen met bit 0, en als hij loopt is hij "aan" en komt hij overeen met bit 1.

Simpeler, het is alsof de bits 0 en 1 overeenkomen met gaten, zodat een leeg gat een beetje 0 is en een gat bezet door een elektron een beetje 1. Dit is de reden waarom deze apparaten elektronica worden genoemd. Als voorbeeld toont figuur 1 het binaire schrift van sommige karakters. Nu we een idee hebben van hoe de huidige computers werken, laten we proberen te begrijpen hoe kwantums werken.

Van bits tot qubits

De fundamentele eenheid van informatie in kwantumberekeningen is de kwantumbit of qubit. Qubits zijn per definitie kwantumsystemen op twee niveaus - we zien hier voorbeelden - die net als bits op het lage niveau kunnen staan, dat overeenkomt met een toestand van lage excitatie of energie gedefinieerd als 0, of op het hoge niveau, dat overeenkomt met een toestand van hogere excitatie of gedefinieerd als 1. Echter, en hier ligt het fundamentele

verschil met de klassieke rekenkunst, qubits kunnen ook in een van de oneindige tussenliggende toestanden tussen 0 en 1 liggen, zoals een toestand die half 0 en half 1 is, of driekwart van 0 en een kwart van 1.



Kwantumalgoritmen, exponentieel krachtiger en efficiënter computergebruik

Het doel van kwantumcomputers is om gebruik te maken van deze kwantumeigenschappen van *qubits*, zoals kwantumsystemen die ze zijn, om kwantumalgoritmen te draaien die gebruik maken van overlappende en interleaving om veel meer rekenkracht te bieden dan de klassiekers. Het is belangrijk om erop te wijzen dat de echte verandering van het paradigma niet bestaat uit het doen van hetzelfde wat digitale of klassieke computers doen - de huidige - maar dat kwantumalgoritmes het mogelijk maken om bepaalde bewerkingen op een totaal andere manier uit te voeren die in veel gevallen efficiënter blijken te zijn - dat wil zeggen, in veel minder tijd of met veel minder rekenkundige middelen -.

Laten we eens kijken naar een concreet voorbeeld van wat dit inhoudt. Laten we ons voorstellen dat we in Bogotá zijn en dat we de beste route willen weten om naar Lima te komen uit een miljoen opties om er te komen ($N=1.000.000$). Om met behulp van computers de optimale route te vinden moeten we 1.000.000 opties digitaliseren, wat betekent dat ze moeten worden vertaald in bittaal voor de klassieke computer en in *qubits* voor de kwantumcomputer. Terwijl een klassieke computer een voor een alle paden zou moeten analyseren tot het vinden van de gewenste, maakt een kwantumcomputer gebruik van het

proces dat bekend staat als kwantumparallelisme en dat het mogelijk maakt om alle paden in één keer te overwegen. Dit houdt in dat, terwijl de klassieke computer de volgorde van $N/2$ stappen of iteraties nodig heeft, dat wil zeggen 500.000 pogingen, de kwantumcomputer het optimale pad zal vinden na slechts VN bewerkingen op het register, dat wil zeggen 1.000 pogingen.

In het vorige geval is het voordeel kwadratisch, maar in andere gevallen is het zelfs exponentieel, wat betekent dat we met n *qubits* een rekencapaciteit kunnen verkrijgen die gelijk is aan 2^n bits. Om dit te illustreren is het gebruikelijk om te tellen dat we met ongeveer 270 qubits meer basistoestanden zouden kunnen hebben in een kwantumcomputer - meer verschillende en gelijktijdige tekenreeksen - dan het aantal atomen in het universum, dat wordt geschat op ongeveer 280. Een ander voorbeeld is dat we met een kwantumcomputer van tussen de 2000 en 2500 *qubits* naar schatting vrijwel alle huidige cryptografie zouden kunnen breken (de zogenaamde publieke sleutelcryptografie).

Waarom is het belangrijk om de kwantumtechnologie te kennen?

We bevinden ons in een moment van digitale transformatie waarbij verschillende opkomende technologieën zoals blockchain, kunstmatige intelligentie, drones, Internet of things, virtual reality, 5G, 3D-printers, robots of autonome voertuigen meer en meer aanwezig zijn in meerdere velden en sectoren. Deze technologieën, die geroepen zijn om de kwaliteit van het leven van de mens te verbeteren en de ontwikkeling te versnellen en sociale gevolgen te genereren, gaan vandaag de dag op een parallelle manier vooruit. Slechts zelden zien we bedrijven producten ontwikkelen die gebruik maken van combinaties van twee of meer van deze technologieën, zoals blockchain en IoT of drones en kunstmatige intelligentie. Hoewel ze voorbestemd zijn om te convergeren en zo een exponentiële grotere impact te genereren, is de eerste fase van de ontwikkeling waarin ze zich bevinden en de schaarste aan ontwikkelaars en mensen met een technisch profiel een nog hangende taak.

Vanwege hun ontwrichtend vermogen wordt verwacht dat kwantumtechnologieën niet alleen convergeren met al deze nieuwe technologieën, maar dat ze ook een transversale invloed hebben op vrijwel al deze technologieën. Quantum computing vormt een bedreiging voor de authenticatie, uitwisseling en veilige opslag van gegevens en heeft een grote impact op technologieën waar cryptografie een meer relevante rol speelt, zoals cyberveiligheid of blokketten, en een kleine negatieve impact, maar moet ook in aanmerking worden genomen bij technologieën zoals 5G, IoT of drones.

Wil je oefenen in quantum computing?

Er zijn al tientallen kwantumcomputers beschikbaar op het net met verschillende programmeertalen die al in gebruik zijn, zoals C, C++, Java, Matlab, Maxima, Python of Octave. Ook nieuwe talen zoals Q#, gelanceerd door Microsoft. U kunt met een virtuele kwantummachine verkennen en spelen via platforms zoals IBM en Rigetti.

Mini BlocklyChain is gemaakt door OpenQbit.com bedrijf dat zich richt op het ontwikkelen van quantum computing technologie voor verschillende soorten private en publieke sectoren.

Waarom Mini BlocklyChain anders is dan andere blockchains, simpelweg omdat het systeem is gemaakt om modulair te zijn en quantum-computing te bevatten.

- (2) <https://blogs.iadb.org/conocimiento-abierto/es/como-funciona-la-computacion-cuantica/>

29. Bijlage "Uitgebreide blokken voor SQLite database".

Om meer details te zien over het juiste en tijdige gebruik van elk blok dat de extensie OpenQbitSQLite vormt, controleert u de oorspronkelijke auteur van de extensie in de volgende link.

OpenQbitSQLite is een kloon van het originele ontwerp met kleine aanpassingen voor gebruik met een AES-beveiligingsniveau.

<https://github.com/frdfsnlght/aix-SQLite>

30. Bijlage "Voorbeeld van het creëren van een Mini BlocklyChain systeem".

We zullen een testsysteem maken waarbij we de functionaliteiten, voordelen en het gemak van het creëren van een Mini BlocklyChain systeem kunnen verifiëren.

We beginnen met het ontwerp en de ontwikkeling van de opslag waar de informatie zal verblijven wat we al hebben gedefinieerd als een keten van blokken. Het ontwerp zal gebaseerd zijn op de SQLite database en afhankelijk van elke organisatie of ontwerp kan deze eenvoudig worden gewijzigd door een andere database zoals Microsoft SQL Server, MySQL, Maria DB, Oracle, DB2, PostgreSQL onder andere. Hoewel ook hier weer door het proces van transacties en de gelijktijdigheid met de SQLite database kan worden gebruikt op elk niveau en voor zakelijke of persoonlijke toepassing.

Database creatie genaamd **minibc** in deze zal een tabel hebben die de blokreeks zal opslaan. Deze database zal worden ingebed in de definitieve programmacode die zal worden geïnstalleerd in de knooppunten, deze opslagplaats wordt "assets packaged" genoemd is een interne soort in Blockly omgevingen zoals App Inventor.

\$ sqlite3 minibc.d.

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.
sqlite>.quit

Het ontwerpen van velden van de **nblock** tafel.

```
CREATE TABLE nblock (
    id gehele primaire sleutel AUTOINCREMENT
    , prevhashVARCHAR NOT NULL
    , newhashVARCHAR NOT NULL
    ntransINTEGER NIET NULLES
        NIET NULLES
    qrng GEHEEL GETAL NIET ONGELDIG
);

```

We hebben de **nblock** tafel gemaakt.

\$ sqlite3 minibc.d.

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.

sqlite>.open minibc.d.

```
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 );
```

We zullen iets dergelijks zien:

```

$ 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
$ 

```

Vervolgens maken we de eerste hash van het systeem genaamd "**Genesis**", die gebaseerd is op het creëren van een nieuwe hash van het eerste blok van de blokketen die we zullen gebruiken (**NewBlock**). Dan maken we een tweede hasj die we "één" noemen op basis van de "Genesis"-hash omdat deze consistent moet zijn met de eerste hasj, omdat de hasjvalidatietest op de eerste blokketen altijd moet voldoen aan: prevhash = newhash.

NewBlock. Om de "Genesis" hash te creëren zullen we de invoerparameters gebruiken:



Invoerparameter: **gegevens** <String>, **vorigeHash** <String>

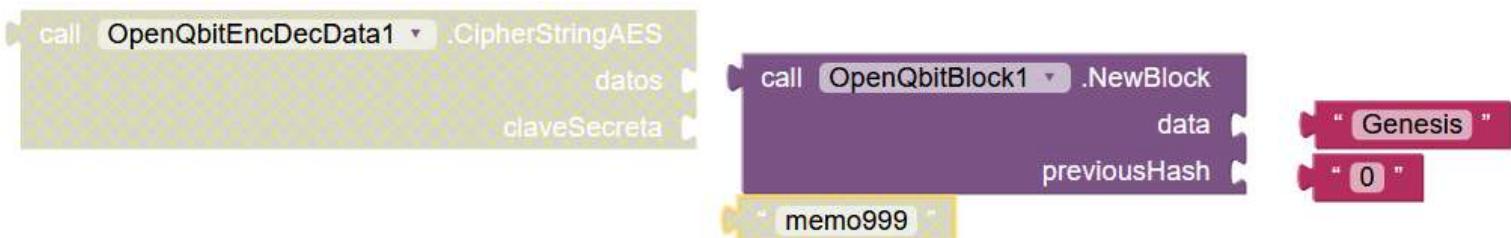
Uitgangsparameter: Een SHA256 Hash.

In dit geval gebruiken we als "previousHash" initiaal gelijk aan "0" en in het geval van invoergegevens is "data" het woord "Genensis".

Dit geeft ons een berekende hash van de combinatie van beide gegevens:

SHA256 (Genesis0) = eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642

De bovenstaande string zal worden ingevoegd in de nblock tabel, deze string moet worden versleuteld voor deze gebruiken we de extensie (**OpenQbitEncDecData**).



We zullen de OpenQbitSSHClient extensie gebruiken om de gegevens in de minibc.db database in te voegen, het zou de INSERT verklaring aan de minibc.db database van de nblock tabel zijn.

```
sqlite3 keystore.db "insert into nblock (prevhash, newhash, ntrans, nonce, qrng) values ('0',
'eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642','1','0',
'0');".
```

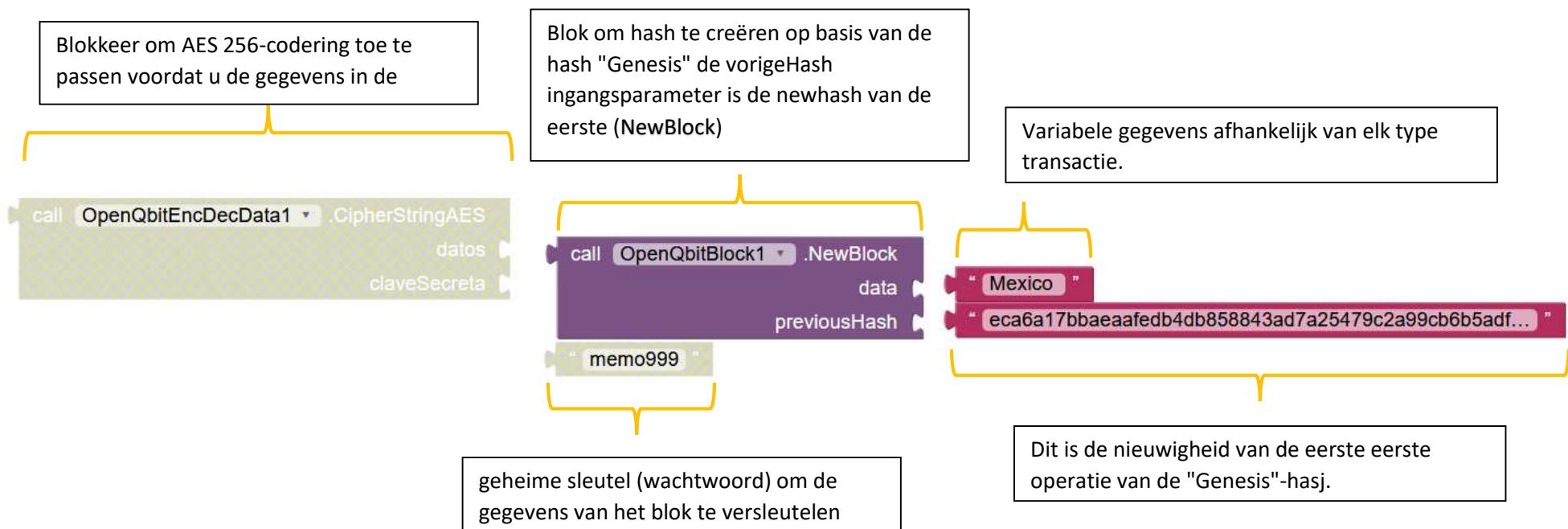
We hebben de "één" hasj gemaakt met de SQL-instructie op basis van de "Genensis" hasj:

```
sqlite3 keystore.db "insert into nblock (prevhash, newhash, ntrans, nonce, qrng) values ("eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642",f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68","1","0","0");"
```

De hash "een" newhash wordt berekend als:

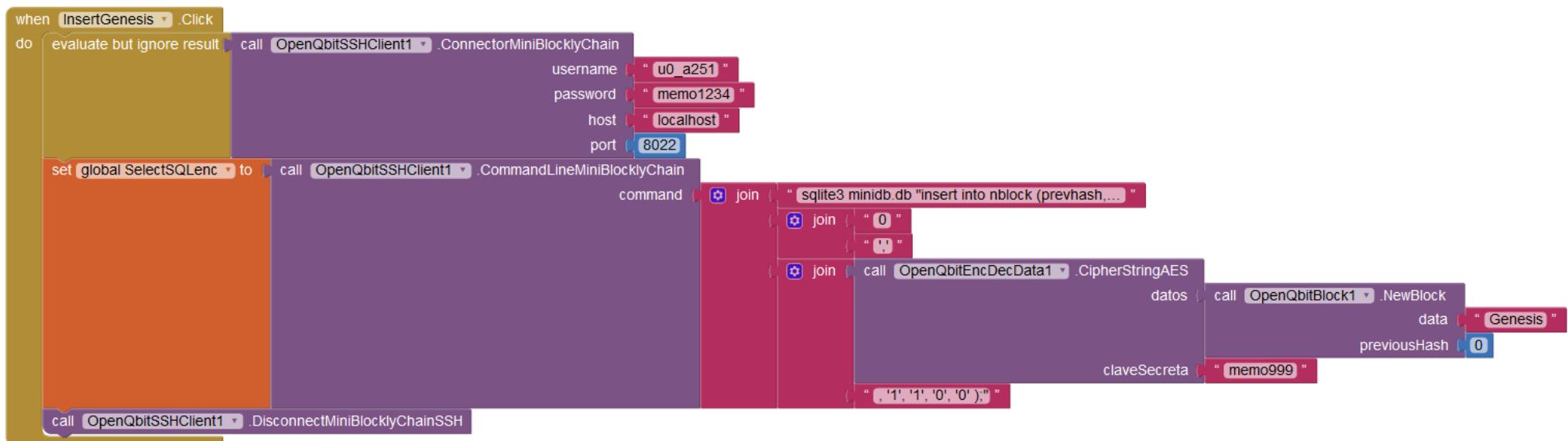
SHA256 (eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642Mexico) =
f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78ab59a68

Tweede hash op basis van de eerste initialisatie hash "Genesis", moeten we twee INSERTS doen aan het begin, omdat elke eerste validatie van de blokketen moet voldoen aan de gelijkheid van de velden: prevhash (voorlaatste gegevens) = newhash (laatste gegevens).



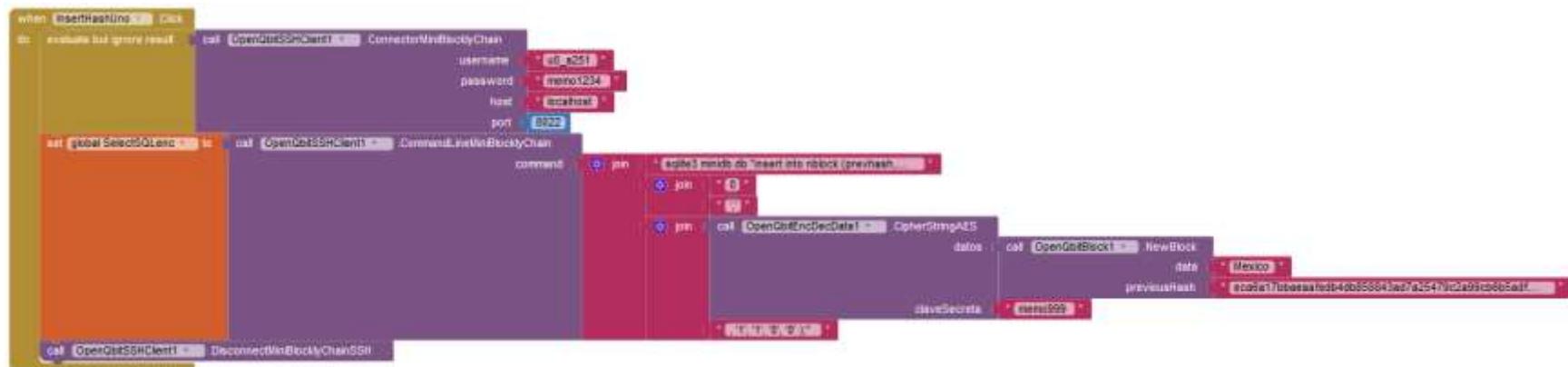
We gaan verder met het creëren van de uitvoering binnen App Inventor van de hasj hierboven; "Genesis" hasj en "start" hasj.

Nu laten we zien hoe de INSERT van de "**Genesis**" hasj zou worden geïntegreerd in de initiële structuur van de blokken binnen de nblocktafel in het App Inventor systeem:



Aangezien we de eerste gegevens in de nblock tabel hebben ingevoegd op basis van de "Genesis" hash, gaan we over tot het maken van de tweede INSERT die verwijst naar de "een" hash.

Het zou de INSERT van de "One" hash integreren in de initiële structuur van de blokketen binnen de nblock tafel in het App Inventor systeem:



De twee vorige programmeerstructuren (hash "Genesis" en hash "one") moeten worden geïntegreerd door het initiële knooppunt van het Mini BlocklyChain systeem, een belangrijk punt is dat de minibc.db database met al zijn interne structuur (tabellen) zal worden gerepliceerd via het Mini BlocklyChain netwerk op basis van een "Peer to Peer" architectuur.

Tot nu toe hebben we de basis- en fundamentele structuur van de blokketenopslag voltooid en is het klaar om nieuwe blokken te ontvangen van toekomstige transacties die in het systeem ontstaan of worden aangevraagd.

We hebben al de eerste hash data "Genesis" en de hash "een" aan onze **minibc.db** database toegevoegd, nu zullen we de volgende SQL statements gebruiken om de **prevhash** en **newhash** velden in de **nblock** tabel te bevragen.

Dit punt zal van fundamenteel belang zijn, aangezien we op basis van de vergelijking van deze twee velden zullen weten of de blokketten consistent is en of deze de integriteit en veiligheid van de transacties handhaaft. Dit is slechts een startmechanisme om de blokketten te herzien, aangezien we in de toekomst het gebruik van het blok zullen herzien om de merkboom te berekenen, wat ook een essentieel instrument zal zijn om de integriteit en de veiligheid van de blokketten in ons systeem te waarborgen.

Voorlopig zullen we alleen de structuur of SQL-zinnen herzien die we moeten gebruiken zoals we eerder hebben gedaan met de extensie ([OpenQbitSSHClient](#)), waarmee we de prevhash en newhash velden kunnen raadplegen. Later kunnen we de eenvoudige vergelijking van gelijke gegevens doen voor die controle elke keer als we een nieuw blok gaan toevoegen.

Voor de prevhash:

```
sqlite3 minibc.d iem. "SELECT prevhash FROM nblock ORDER BY id DESC LIMIT 1;".
```

Voor newhash:

```
sqlite3 minibc.db "SELECT newhash FROM nblock ORDER BY id DESC LIMIT 1;"
```

We zullen ons nu richten op de manier waarop we een verzoek zullen indienen om een nieuwe transactie en/of transactie in te voegen in de transactiewachtrij.

Er zijn twee stappen als vereisten om een transactie naar de transactiewachtrij te sturen.

De eerste vereiste is dat ons rekeningsaldo over voldoende "activa" beschikt om het te verzenden bedrag te dekken. Vergeet niet dat de "activa" niet alleen een aantal of een bedrag kunnen zijn, maar dat we "activa" van welke aard dan ook kunnen creëren, afhankelijk van elk aan te maken systeem.

Voorbeelden van activa:

- ✓ Intrinsieke hoeveelheid van een "virtuele of crypto-valuta" hoeveelheid van nieuwe oorsprong.
- ✓ Gegevens met betrekking tot digitale gegevens (alle soorten documenten)
- ✓ Hash authenticatie handtekeningen voor strings of alle soorten bestanden.
- ✓ Elke transactie of overdracht van digitale informatie of weergave daarvan.

Het saldo van de "activa" van elk knooppunt wordt verkregen met behulp van het blok (**GetBalance**).



De tweede vereiste is het verstrekken van de minimumparameters bij het verzenden van een transactie, die zijn

- ✓ Bronadres. - is het adres van waaruit de goederen worden verzonden.
- ✓ Bestemmingsadres. - is het adres van de gebruiker die de verzonden goederen zal ontvangen.
- ✓ Actief. - Het zijn de gegevens met een intrinsieke waarde die in elk systeem worden gegeven en die in elke transactie moeten worden verzonden.

De bovenstaande adressen (herkomst-bestemming) komen overeen met de publieke sleutels van elke gebruiker bij het aanmaken van de accounts van elke gebruiker. Vergeet niet dat bij het aanmaken van een gebruikersaccount twee soorten publieke en private sleutels worden aangemaakt.

De publieke sleutel is het adres dat in het hele systeem wordt gedeeld en dat elke gebruiker van het systeem kan zien en gebruiken om transacties te verzenden of te ontvangen.

De privésleutel is het adres dat lokaal door elk knooppunt wordt gebruikt en aangezien de naam ervan aangeeft dat het voor privégebruik is dat helpt om digitale handtekeningen te creëren om de verzonden transacties te beveiligen, mag deze sleutel nooit worden gedeeld en is deze sleutel voor lokaal gebruik en uniek voor het bronknooppunt.

Om de vorige parameters te gebruiken zullen we vertrouwen op twee repositories waar de "private keys" adressen die in de keystore.db database zijn opgeslagen en zullen van lokaal gebruik zijn van elk knooppunt zonder te delen in het systeem en de andere repository waar alle "public keys" adressen die in de publickeys.db database zijn opgeslagen zullen worden gedeeld door middel van het "Peer to Peer" protocol in alle knooppunten van het systeem.

De databases "keystore.db" en "publickeys.db" zijn al aangemaakt in de bijlage "KeyStore & PublicKeys database creatie".

De database en het systeem voor de transactiewachtrij is al aangemaakt in de sectie "Installatie en configuratie van het RESTful Mini Sentinel netwerk". Waar we al een database hebben aangemaakt voor de transacties die op.sqlite3 worden genoemd, hebben we twee tabellen, een is "trans" die zorgt voor de transacties (**transactiewachtrij**) en een andere heet "sign" waar de digitale handtekeningen van elke transactie worden opgeslagen.

De SQLite RESTful systeem is de back-up netwerk in deze gelegenheid zullen we het gebruiken voor het gemak en om de back-up systeem te testen, echter, om te veranderen

en dezelfde database op.sqlite3 te gebruiken in een "Peer to Peer" model zullen we alleen maar om de database te gebruiken in een gedeelde model weer in de synchrone systeem, dit zullen we later zien.

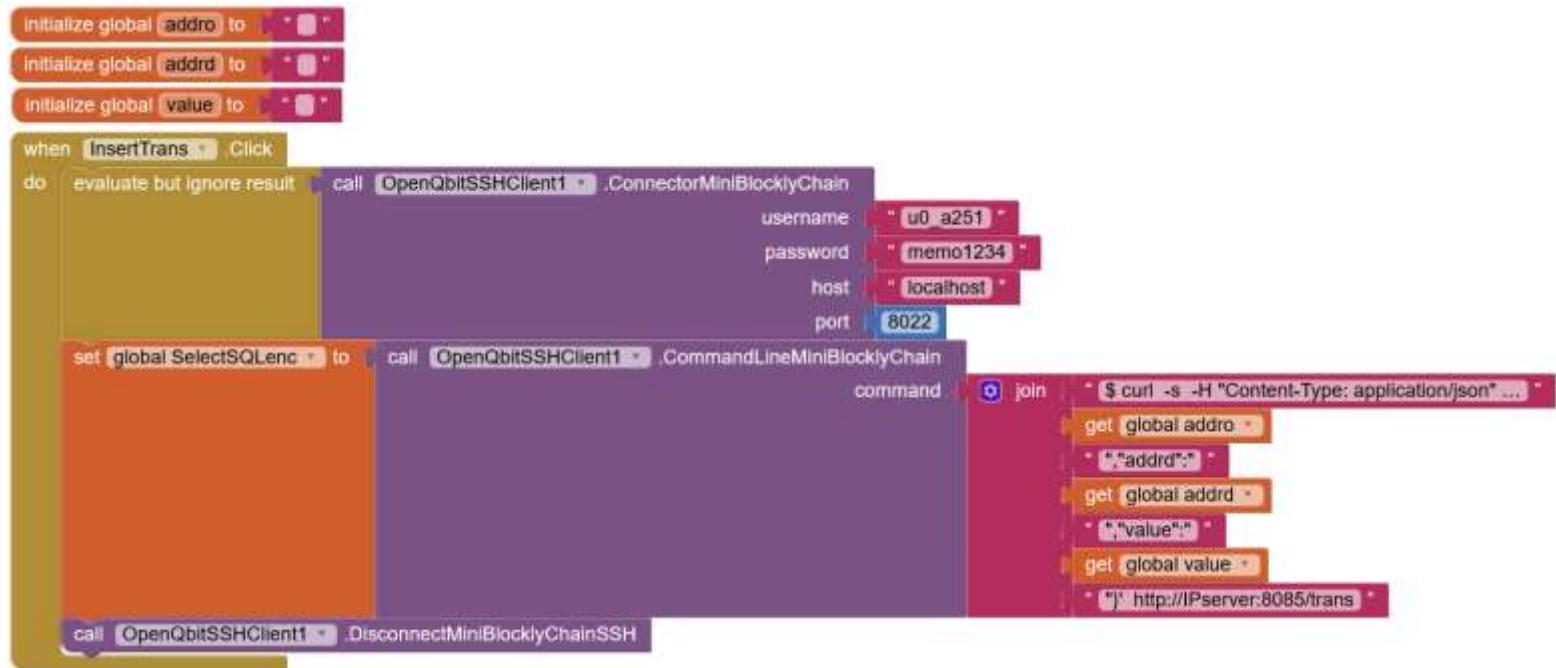
We zullen beginnen met het verzenden van operaties naar de transactie wachtrij met behulp van RESTful toegepast op de op.sqlite database.

We hebben een nieuwe transactie in de "**trans**"-tabel gemaakt...

```
$ curl -s -H "Content-Type: applicatie/json" -d '{"addro":  
"MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9", "addrd":  
"MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh", "waarde": "999"}' http://IPserver:8085/trans
```

```
# uitgangen  
{  
  "id": 1,  
  "addro": "MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",  
  "addrd": "MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh",  
  "waarde": "999"  
}
```

Implementatie in App Inventor:



De invoerparameters: addro, addrd, waarde zijn variabelen die in het algoritme kunnen worden ingevoerd en uit de keystore.db database worden gehaald.

Zie bijlage "Aanmaken van KeyStore database".

We voegen nu de digitale handtekeningen van de vorige transactie toe. In de tabel "**bord**"

```
$ curl -s -H "Content-Type: applicatie/json" -d '{
  "trans_id":1
  "teken": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",
  "hasj": "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68"
  http://IPserver:8085/sign
}
```

```
# uitgangen
{
  "id": 1,
  "trans_id": 1,
  "teken": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",
  "hasj": "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68".
}
```

OPMERKING: De digitale handtekening (sign) moet worden verkregen voor het verzenden van het curl command statement, het wordt gegenereerd met het blok (**GenerateSignature**).

De invoerparameters: Trans_id, sign, hash zijn variabelen die in het algoritme kunnen worden ingevoerd en de digitale handtekening van de transactie zal worden gegenereerd met het blok (**GenerateSignature**).

We zullen nu zien hoe de procedure voor het genereren van een digitale handtekening wordt toegepast op elke transactie die wordt uitgevoerd.

Generatie van digitale handtekening voor transactie. Wanneer we het blok (**GenerateSignature**) gebruiken hebben we als resultaat een **bestandstype .sig dit bestand zullen we gebruiken** om te worden opgeslagen in het teken veld in de tabel "sign", deze handtekening zal worden gebruikt wanneer de transactie wachtrij wordt verwerkt en het is een andere parameter om de veiligheid te geven tussen de oorsprong en de bestemming, evenals het bedrag of het type van de transactie tussen hen.

Om binaire gegevens zoals het bestand.sig te kunnen verwerken, moeten we het converteren naar base64 en vervolgens opslaan in de tekenvariabele in de "sign"-tabel. Hiervoor gebruiken we het blok (**EncoderFileBase64**).

Hoe de hash-veldwaarde wordt berekend voor de "teken"-tabel van elke transactie:

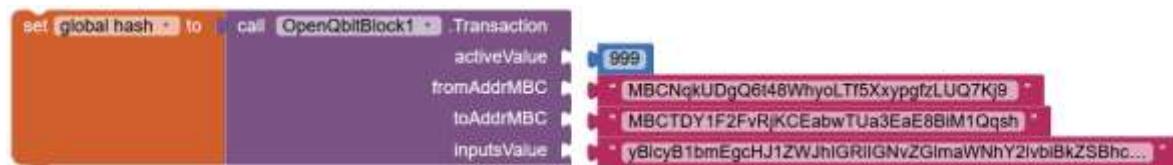
Transactie Hash = SHA256(Bron Adres + Bestemmingsadres + Asset + fileBase64.sig)

Voorbeeld van hasj type SHA256:

SHA256(MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9 +
MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh + 999 +).

Uitgang: 9d45198faaef624f2e7d1897dd9b3cede6ecca7fbac516ed1756b350fe1d56b4

Voor de berekening van de hasj zullen we moeten leunen op het blok (**Transactie**).



De uitgangsparameter is de hash die wordt opgeslagen voor elke transactie die vanuit een willekeurig knooppunt in het systeem wordt verzonden. Deze wordt opgeslagen in het tekenveld van de "sign"-tabel in de basis op.sqlite

Met de vorige operatie hebben we ervoor gezorgd dat alleen een unieke en niet-herhaalbare hash elke transactie die naar de wachtrij wordt gestuurd zal vertegenwoordigen en deze representatieve hash is de totaliteit van de verzonden informatie.

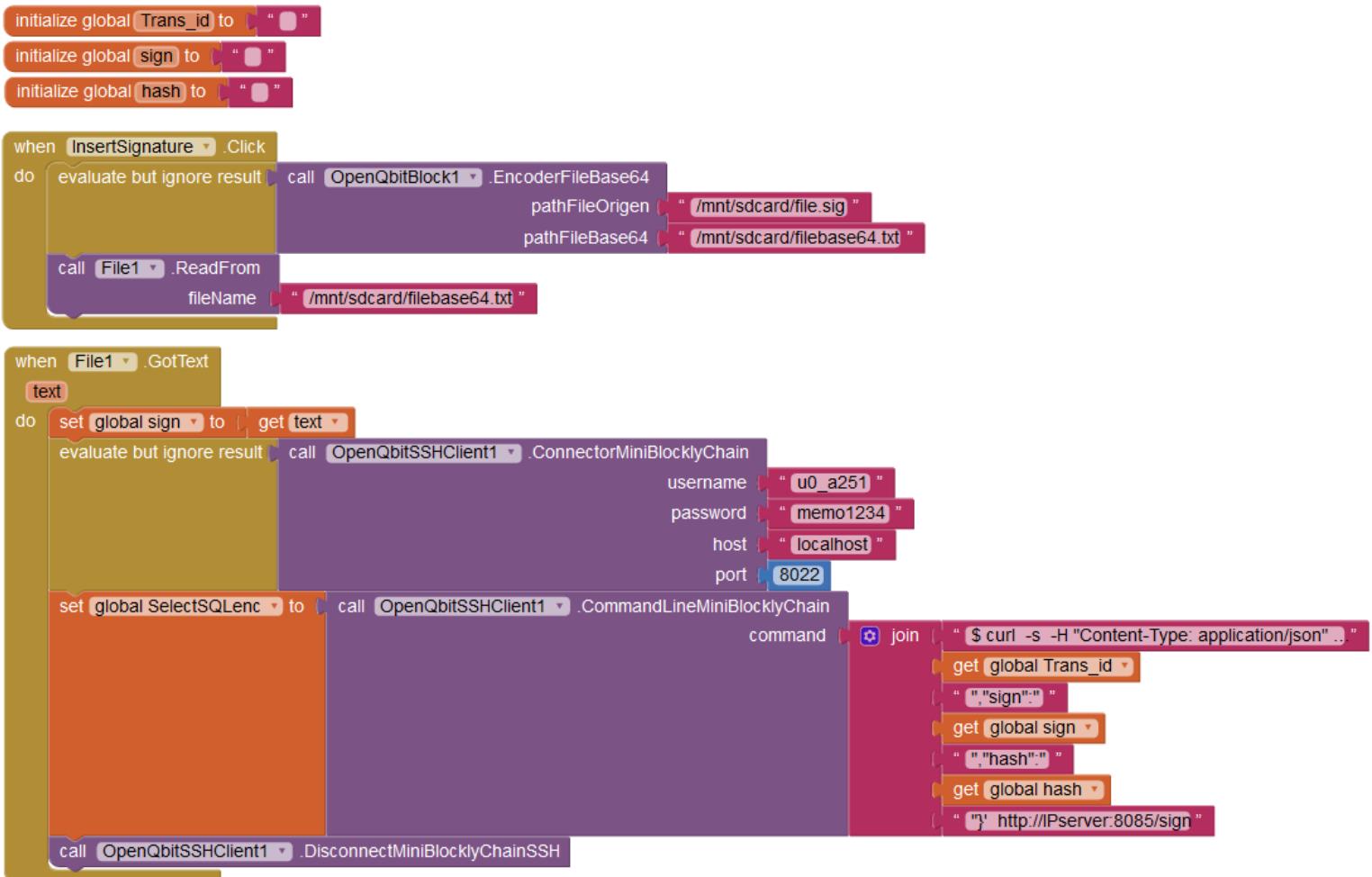
Het enige wat ontbreekt is het opnemen van de Trans_id variabele, die een identifier kan zijn om te onderscheiden welk knooppunt wordt verstuurd voor elke transactie. In ons voorbeeld zullen we de Trans_id variabele met een vaste waarde van "1" plaatsen. Omdat het het eerste knooppunt is dat in ons netwerk wordt geconfigureerd. Deze waarde kan de syntax variëren volgens een bepaald ontwerp, dus voor de eenvoud hebben we gekozen voor een eenvoudige identifier.

Aangezien we alle variabelen hebben gedefinieerd zullen we de structuur en volgorde van de blokuitvoering binnen App Inventor creëren, om de INSERT uit te voeren in de tabel "teken".

OPMERKING: Het is belangrijk om er rekening mee te houden dat elke verzending van een transactie is samengesteld uit twee INSERTS in de op.sqlite3 database, een moet worden gedaan in de "trans"-tabel en hun respectievelijke waarden in de "sign"-tabel.

In het ontwerp van de op.sqlite3 database zijn twee aparte tabellen gemaakt omdat het in de toekomst alleen mogelijk is om de "sign"-tabel te versleutelen omdat deze informatie bevat die niet op een vlakke manier in het netwerk moet worden verzonden, deze optie wordt overgelaten aan de overweging van elk toekomstig ontwerp.

We zullen de uitvoeringsstructuur van blokken en methoden binnen de App Inventor van het INSERT in de tabel "teken" creëren.



Tot op dit moment zijn we al klaar met de INSERT naar de "sign" tabel en naar de "trans" tabel. Deze bevinden zich in de **op.sqlite3** database en de gegevens die in deze twee tabellen zijn ingevoegd zullen worden gebruikt om de transactiewachtrij aan te maken die naar alle knooppunten zal worden gestuurd voor de verwerking ervan.

Op dit moment zullen we gebruik maken van de Java SQLite-Redis Connector. Deze connector zal de conversie van de data van de op.sqlite3 database naar de Redis database uitvoeren. Zie bijlage "Java SQLite-Redis Code Connector".

Na de implementatie van de SQLite-Redis Connector wordt de transactiewachtrij via het Redis-systeem aan alle knooppunten van het systeem geleverd.

We zullen het proces starten van hoe we de transactiewachtrij op een goede manier kunnen ontvangen om deze te kunnen verwerken.

De transactiewachtrij wordt geleverd via de Redis service. Dit is een real-time database die zijn respectievelijke controleblokken heeft in de App Inventor omgeving.

Configuratie van de Redis-omgeving binnen het Blockly-systeem van App Inventor.

Een belangrijk punt om op te merken is dat de blokken voor het aansturen van een Redis-server tot het moment van schrijven van deze handleiding alleen beschikbaar zijn in het App Inventor systeem. In het geval van het gebruik van een Blockly systeem anders dan de App Inventor zult u gebruik moeten maken van de Redis CLI (Command-Line) uitvoering hiervan door het sturen van commando's naar de Termux terminal via de extensie (OpenQbitSSHClient).

Voorbeeld van gebruik in Redis CLI (Command-Line):

Om alle **labels** of sleutels te verkrijgen bij Redis.

```
$ redis-cli KEYS '*'.
```

Om waarde te halen uit een sleutel.

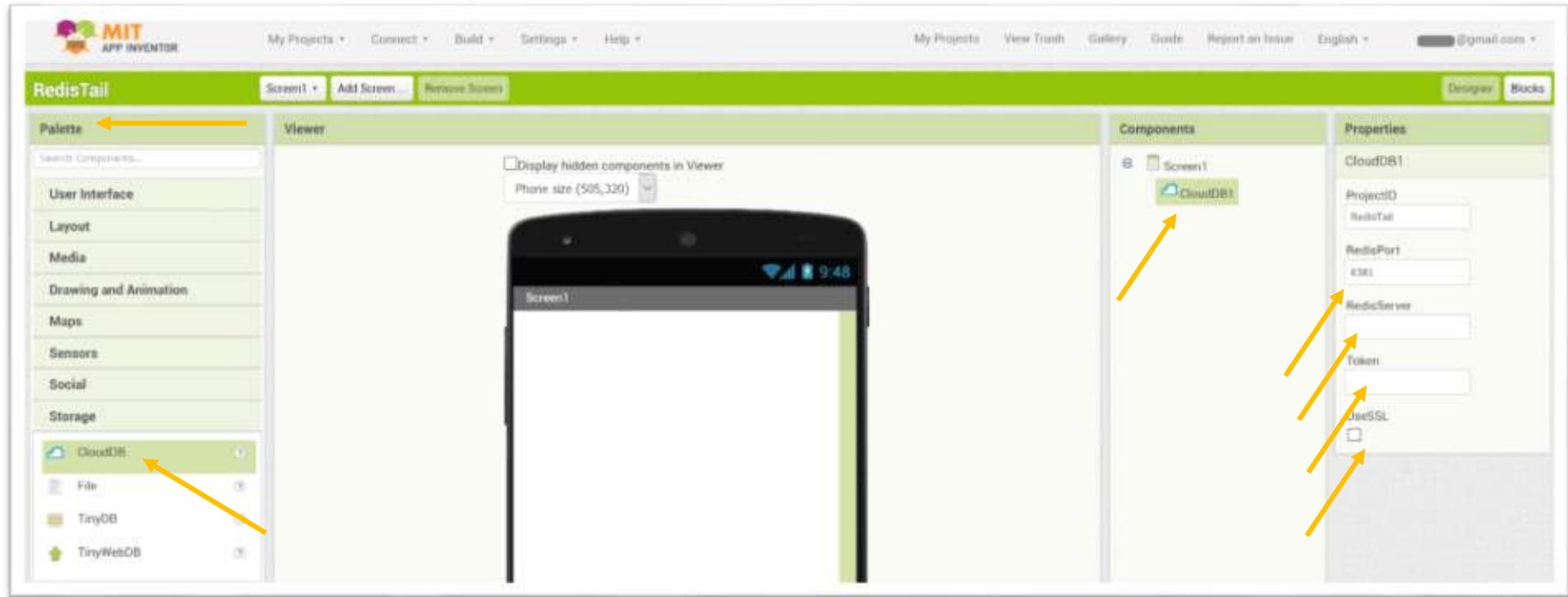
```
$ redis-cli GET <sleutel>
```

In ons geval, omdat we App Inventor gebruiken, zullen we bekijken hoe we de blokken kunnen gebruiken om met Redis te werken.

Binnen App Inventor hebben we verschillende blokken om te gebruiken met Redis, we zijn begonnen met het maken van een nieuw project dat we gaan maken: Mijn projecten > Start nieuw project



Na het aanmaken van het project gaan we naar linksboven en in de sectie "Palette" klikken we op "Storage" en slepen het object "CloudDB" dit zal alle functies integreren om een verbinding met een Redis-server te configureren.



De configuratie is heel eenvoudig, we hoeven alleen de volgende parameters op te geven om verbinding te kunnen maken met onze Redis-server (Lokaal) die in onze node (Mobiele telefoon) draait.

ProjectID. - Dit is het ID dat het label dat de transactiewachtrij in Redis identificeert zal starten.

RedisPort. - Dit is de poort van de Redis-server waar we standaard verbinding maken is 6379

RedisServer. - Dit is het domein of Lokale IP van het knooppunt in ons geval en dat van alle knooppunten zal 127.0.0.1 zijn.

Token. - Dit is het wachtwoord van de Redis-server waarmee u verbinding kunt maken.

GebruikSSL. - Deze optie geeft ons de mogelijkheid om SSL-certificaten te gebruiken in ons geval laten we het niet toe.

In ons model en systeemontwerp hebben we de volgende parameters in alle netwerkknooppunten.



Het is tijd om de blokken te herzien die ons voorzien van controle en gegevensverwerking over een Redis-databaseomgeving.

We beginnen met het methodeblok (**DataChanged**)



Deze methode geeft ons twee waarden elke keer als er een verandering is in de Redis-server die we hebben geconfigureerd:

tikkertje. - Deze waarde is de tag die de transactiewachtrij identificeert.
waarde. - Deze waarde bevat de lijst van alle transacties die voor verwerking zijn verzonden.
Deze waarde is een stringlist van het type <String>.

In het geval van de "waarde" is waar we onze informatieverwerking als volgt gaan uitvoeren.

Aangezien u ons een keten van alle hashwaarde-transacties ter beschikking stelt, zullen we eerst controleren of deze keten geldig is en nagaan of ze niet gewijzigd is sinds haar ontstaan. Dit doen we door gebruik te maken van een zeer nuttig algoritme voor de verificatie van grote hoeveelheden informatie.

We gebruiken het boomalgoritme van Merkle. Het toepassen van dit algoritme geeft ons een veiligheid in de integriteit van de informatie die we ontvangen (transactiewachtrij).

Laten we het volgende voorbeeld zien van een transactiewachtrij in Redis, we voeren de Redis CLI (Command-Line) uit vanaf een Termux terminal om de "LATAM:HASH" toets te controleren, we moeten de SQLite-Redis Connector al hebben uitgevoerd om deze toets te kunnen raadplegen.

```
C:memor redis-cli
127.0.0.1:6379> haal LATAM: HASH
"9d45198faaef624f2e7d1897dd9b3cde6ecca7fbac516ed1756b350fe1d56b4,"
f71c801a5fd25fc303ebc8c616204b4877ffb93006ec6a88bc30acf43ec250f5,"\60f8a3bcac1
ea7d38e86efbc3e3e00480807d23f980391000766e804ed14ecb2 ,
8a6dfe1d38c22e0f9212052efa6136da3edf1fb1b2a3e25224ac3d689124b754].
127.0.0.1:6379>
```

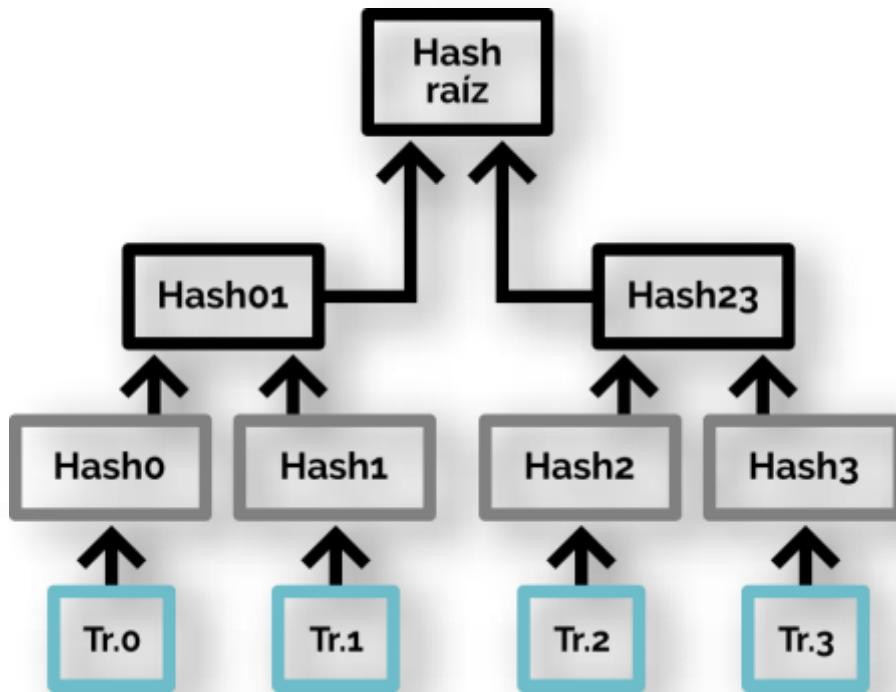
De vorige transactiewachtrij heeft slechts drie elementen aangezien we vier hashes zien die een individuele transactie vertegenwoordigen en ze zijn de hashes van elke transactie die in eerste instantie werd geïnjecteerd in de op.sqlite3 database binnen de "trans"- en "teken"-tabellen.

Nu is het tijd om te begrijpen hoe de merkboom werkt.

Een hasj merkboom is een binaire of niet-binaire gegevensboomstructuur waarin elk knooppunt dat geen blad is, wordt gelabeld met de hasj van de aaneenschakeling van de labels of waarden van de kinderknooppunten. Ze zijn een veralgemening van haslijsten en hasjsnaren.

In ons geval zullen we de volgende berekening maken om de merkboom te berekenen voor vier elementen dit wordt gedaan door het resultaat te berekenen van het nemen van paren van gegevens aaneengeschakeld en krijgen hun respectievelijke hash, de resultaten van het eerste niveau zal worden toegepast op de resultaten van het tweede niveau totdat er slechts een laatste element, dit zal worden genoemd de hash merkleroot (wortel hash).

Laten we eens kijken naar het volgende schema dat dit proces beschrijft.



De op hasj gebaseerde transactiewachtrij wordt via het Block (**GetMerkleRoot**) uit de wachtrij gehaald.



Hasjwortel: 51431822de7c94b90dc06d47b8f6275f315a4976c8479d30c32747fa90325432

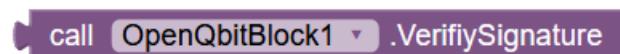
Het resultaat wordt dan vergeleken met de LATAM:merkleroot toets in het lokale Redis-systeem en beide toetsen moeten overeenkomen om de data-integriteit te verifiëren.

Een ander veiligheidspunt van de merkboom is de bevestiging dat een specifieke transactie vanuit de herkomst ervan in de wachtrij is opgenomen en dat deze niet op frauduleuze wijze door een extern of intern communicatiemiddel is ingevoerd.

In het geval dat de keten bestaat uit vreemde elementen in zijn sommatie, dupliceert het laatste element zich om een regeling te hebben voor en te beginnen met de uitvoering van het algoritme.

Een ander, niet minder belangrijk punt is de tijd om te valideren dat elke transactie overeenkomt met de herkomst-bestemming en dat het actief het actief is dat door het herkomstadres wordt verzonden.

Hier staat het blok (VerifySignature).



Voordat u het vorige blok uitvoert, moet u eerst het bestand (file.sig) in binair formaat downloaden uit de "sign"-tabel:

```
sqlite3 op.sqlite3 "selecteer teken van teken where=id_addr;"
```

id_addr: Dit is de id van het bronadres van de "trans"-tabel.

De vorige zoekopdracht geeft ons een Base64 formaat gegevens van de digitale handtekening van de huidige transactie, dit om deze om te zetten in het originele formaat (binair) zullen we de Block (**DecoderFileBase64**) nodig hebben.

Aangezien we ons binaire bestand (file.sig) hebben moeten we de publieke sleutel van de oorsprong en de publieke sleutel van de ontvanger ook in binair formaat in het systeem laden, waarbij de vier gegevens in hun respectievelijke formaten de tijd zullen zijn om de verificatie van de digitale handtekening in het systeem uit te voeren.

- ✓ Openbare sleutel thuisadres
- ✓ Publiek sleuteladres van de ontvanger
- ✓ Actief gestuurd.
- ✓ Digitale handtekening (file.sig)

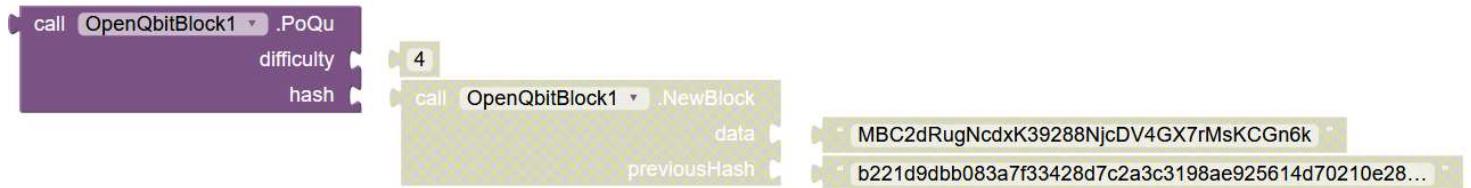
Openbare sleutels in binair formaat kunnen worden gedownload in het juiste formaat (binair) van de gedeelde database publickeys.db

Dit proces moet worden uitgevoerd voor elke afzonderlijke verrichting in de transactiewachtrij.

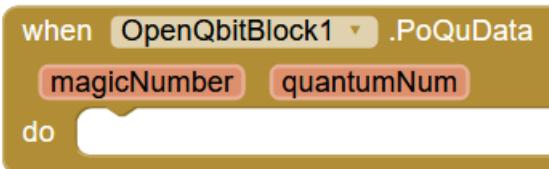
Tot slot zullen we bekijken hoe het systeem op een consensuele manier een knooppunt kiest om het volgende blok aan de blokketen te kunnen toevoegen en degene te zijn die de transactiewachtrij verwerkt.

Deze manier van het kiezen van het winnende knooppunt om de transactiewachtrij te verwerken is gebaseerd op ons algoritme dat is ontwikkeld voor een Mini BlocklyChain systeem.

Hoe we de PoQu "Proof of Quantum" consensus hebben geïmplementeerd. Dit consensusproces is gebaseerd op het genereren van kwantum willekeurige getallen en wordt toegepast met behulp van het (**PoQu**) blok.



Eerst krijgen we twee parameters die het blok (**PoQu**) ons levert in de methode (**PoQuData**) de parameters zijn het magicNumber en quantumNum.



Het **magicNumber** is het getal "nonce", is een geheel getal dat wordt verkregen door het maken van een intern PoW met moeite niet groter dan 5, dit getal is belast met het geven van een eerste vereiste aan het knooppunt met het magicNumber het knooppunt in staat zal zijn om een vereiste te maken om een nummer van het systeem van generatie van kwantum willekeurige getallen te verkrijgen dat in het bereik van de 0 tot 1 ligt, dit getal zal een waarschijnlijkheid geven die willekeurig is vastgesteld voor het knooppunt in uitvoering dat zal worden opgeslagen in de tabel genaamd "vote".

De "vote" tabel zal het door elk knooppunt berekende quantumNum opslaan, deze opslag zal gebeuren met een aantal knooppunten tot een bepaalde tijd vastgesteld in elk systeemontwerp dat wordt aanbevolen om te hebben van entries $((N/2) + 1)$ waarbij "N" het aantal beschikbare knooppunten in het systeem is en het kan worden vastgesteld of gecontroleerd door een actie in de "cron" taakbeheertool van elk knooppunt.

Een belangrijk punt is dat u op dat moment al een lokale tijdsynchronisatie van elk knooppunt via het automatische systeem van de mobiele telefoon had moeten instellen in geval van gebruik van het "**Peer to Peer**"-netwerk in de overdracht van de transactiewachtrij.

De configuratie van de cron agent in de knooppunten. Zie paragraaf "Tijdsynchronisatie in de systeemknooppunten (mobiele telefoon) in minuten en seconden".

In dit voorbeeld gebruiken we het back-up communicatienetwerk en hebben we geen synchronisatie van minuten en seconden nodig voor de nodes, omdat ons voorbeeld een "**Client-Server**" schema gebruikt, dit type communicatie zit alleen in het transmissieproces van de transactiewachtrij. Alle andere processen tussen de knooppunten verlopen via een "**Peer to Peer**"-communicatie.

Nu zullen we kijken naar de structuur, het ontwerp en de creatie van de "vote"-tabel die zich zal bevinden binnen de database genaamd "quorum.db" die we in dit voorbeeld ook zullen creëren.

\$ sqlite3

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.

sqlite> .open quorum.d iem.

sqlite> CREATE TABLE vote (id integer primary key AUTOINCREMENT NOT NULL, node_imei VARCHAR NOT NULL, quantumNum INTEGER NOT NULL, nonce INTEGER NOT NULL);

sqlite> .quit

De creatie van dezelfde **stemtabel** wordt hieronder getoond, maar het is door de invoering van de SQL-statement in een gesegmenteerde vorm:

\$ sqlite3

SQLite versie 3.32.2 2020-06-20 15:25:24

Voer ".help" in voor gebruikstips.

Verbonden met een tijdelijke in-memory database.

Gebruik ".open FILENAME" om opnieuw te openen op een permanente database.

sqlite> .open quorum.d iem.

sqlite> CREATE TABLE stemming (

...> id integer primaire sleutel AUTOINCREMENT
...> node_imei VARCHAR NOT NULL,
...> quantumNum INTEGER NOT NULL,
...> nonce INTEGER NOT NULL
...>);

sqlite> .tables

stemmen

sqlite> .quit

We zullen iets dergelijks zien in de oprichting van de basis "quorum.db" en tafelstemming.



A screenshot of an Android mobile device. At the top, there's a black status bar with icons for signal strength, battery level (26%), and time (9:02 p.m.). Below it is a white terminal window showing SQLite command-line interface output:

```
$ 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
$
```

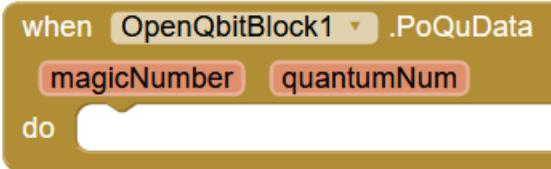
Below the terminal is a standard QWERTY keyboard. The bottom of the screen features a navigation bar with three icons: a triangle pointing down, a circle, and a square.

Laten we nu eens kijken hoe we de waarden uit de "stem"-tabel krijgen.

node_imei. - We krijgen deze waarde met behulp van het blok (**GetDeviceID**).

call [OpenQbitBlock1] .GetDeviceID

quantumNum - Deze waarde is een van de resultaten van de (**PoQuData**) methode die wordt verkregen met behulp van het (**PoQu**) blok.



nonce. - Deze waarde wordt verkregen door het blok (**PoQu**) al integraal uit te voeren en hetzelfde als het magische nummer van de methode (**PoQuData**).

Na het invullen van de INSERTS in de "vote" tabel, is het noodzakelijk om een kopie van de database te maken.

Na een bepaalde tijd en op basis van het ontwerp van elk systeem zal de "cron"-dienst worden uitgevoerd op elk netwerkknooppunt en zal de volgende SELECT worden verwerkt in de "vote"-tabel:

```
SELECT node_imei FROM vote WHERE magicNumber= (SELECT max(magicNumber) FROM vote);
```

De vorige SELECT geeft het IMEI resultaat met een hogere waarschijnlijkheid, nu zal elke node die SELECT uitvoert een vergelijking maken van zijn IMEI met het resultaat IMEI en alleen de node die overeenkomt zal een bestand maken met het hoogste waarschijnlijkhedsgetal dat zal worden gerepliceerd in het netwerk "Peer to Peer" door een bestand met het formaat IMEI.mbc dat de IMEI van de winnende node zal bevatten.

Het winnende knooppunt zal in staat zijn om te beginnen met de verwerking van de transactiewachtrij. Gebaseerd op alle bovenstaande blokken.

Twee belangrijke punten zijn dat afhankelijk van elke systeemcreatie drie processen moeten worden herzien en aangepast door elke ontwerper.

1.- Wanneer het winnende knooppunt begint met de verwerking van de transactiewachtrij, moet een methode of proces worden geïmplementeerd waarbij moet worden gecontroleerd of het winnende knooppunt online is en de communicatie met het netwerk niet verloren is gegaan.

2.- Wanneer de verwerking van de transactiewachtrij begint, moet het winnende knooppunt twee controlevlaggen starten die het begin van de verwerking aangeven en een andere om te bevestigen dat de verwerking van de transactiewachtrij is voltooid. Deze twee vlaggen moeten worden gedeeld in het netwerk naar alle knooppunten, dit zal helpen bij het lokaliseren van verbindingfouten of verwerkingsfouten of te veel verwerkingstijd.

3.- In geval van een communicatiefout of een andere gebeurtenis waarbij het winnende knooppunt niet in staat is geweest om de transactiewachtrij te verwerken, moet het volgende knooppunt in het onmiddellijke waarschijnlijkheidsbereik worden gekozen.

Het vorige punt kan worden gecontroleerd met een service die controleert dat de winnende node online is en kan gebruik maken van de "cron" service, moet het script worden ontwikkeld voor elk ontworpen geval, maar een generiek voorbeeld van een shell script wordt hieronder getoond, zodat het kan worden aangepast aan de behoeften van elk Mini Blocklychain systeem.

```
#!/bin/bash
dir="/data/data/com.termux/bestanden/home/Sync/imei";
als [ !"$!(ls $directory)" ]
dan
sqlite3 quorum.db "UPDATE stemming SET magicNumber=0 WHERE magicNumber=
(SELECT max(magicNumber) FROM vote);"
anders

MAX_NUM=$(sqlite3 quorum.db "SELECT max(magicNumber) FROM vote;")
IMEI_quorum=$(sqlite3 quorum.db "SELECT node_id FROM vote WHERE=MAX_NUM")
IMEI_local=$(cat device_imei) // Gebruik het blok (GetDevice)
als [ IMEI_quorum -eq IMEI_local ]
dan
druk op $MAX_NUM > IMEI.mbc
fi
fi
afslag
```

31. Bijlage "Integratie met Ethereum & Bitcoin-omgevingen".

Nu zullen we zien hoe we de twee blokketensystemen kunnen integreren die wereldwijd het meest bekend zijn en die gespecialiseerd zijn in cryptomonieën zoals Ethereum en Bitcoin.

Laten we beginnen met de installatie van de software die ons zal helpen om alle mogelijke transacties in de Ethereum-omgeving uit te voeren.

Wat is Ethereum?

Ethereum is een open source platform, gedecentraliseerd in tegenstelling tot andere blokketens. Ethereum kan veel meer. Het is programmeerbaar, wat betekent dat ontwikkelaars het kunnen gebruiken om nieuwe soorten applicaties te maken.

Deze gedecentraliseerde toepassingen (of "dapps") krijgen de voordelen van de cryptomontage- en blokketentechnologie. Ze zijn betrouwbaar en voorspelbaar, wat betekent dat als ze eenmaal "geladen" zijn in het Ethereum, ze altijd op schema zullen lopen. Zij kunnen digitale middelen beheren om nieuwe soorten financiële toepassingen te creëren. Ze kunnen gedecentraliseerd worden, wat betekent dat geen enkele entiteit of persoon er zeggenschap over heeft.

Op dit moment zijn duizenden ontwikkelaars over de hele wereld bezig met het maken van toepassingen op Ethereum en het bedenken van nieuwe soorten toepassingen, waarvan u er veel vandaag de dag kunt gebruiken:

- Crypto-valutaportefeuilles die u in staat stellen om goedkope, directe betalingen te doen met ETH of andere activa
- Financiële toepassingen waarmee u uw digitale activa kunt lenen, uitlenen of investeren
- Gedecentraliseerde markten, waardoor u digitale middelen kunt uitwisselen, of zelfs "voorspellingen" over gebeurtenissen in de echte wereld kunt uitwisselen.
- Games waarbij je activa in het spel hebt en zelfs echt geld kan winnen.
- Het heeft intelligente contracten die programma's zijn met overeenkomsten die moeten worden uitgevoerd wanneer het pand waarmee het is uitgewerkt of gecreëerd, wordt uitgevoerd.

Intelligente contracten vertonen overeenkomsten met **DApps** (decentrale toepassingen), maar scheiden deze ook van enkele belangrijke verschillen.

Net als slimme contracten is een DApp een interface die een gebruiker via een decentraal peer netwerk verbindt met een dienst van een provider. Maar, terwijl slimme contracten een vast aantal deelnemers nodig hebben om te worden gecreëerd, hebben DApp's geen limiet

aan het aantal gebruikers. Bovendien zijn ze niet alleen beperkt tot financiële toepassingen zoals slimme contracten: een DApp kan elk denkbaar doel dienen.

In ons geval zullen we gebruik maken van de bibliotheek genaamd "Web3j" die is ontwikkeld in Java, en die het mogelijk maakt om op een eenvoudige en intuïtieve manier te communiceren met de Ethereum-blokketen.

Voer de volgende opdracht uit om "Web3j" te installeren:

```
$ curl -L get.web3j.io | sh
```

Dan moeten we de omgevingsvariabele **\$JAVA_HOME** aanmaken met het pad waar de uitvoerbare "java" zich bevindt, aangezien de bibliotheek deze variabele zal doorzoeken om succesvol te kunnen worden uitgevoerd.

```
$ JAVA_HOME= /data/data/com.termux/files/usr/bin
```

Zodra de variabele is aangemaakt moeten we naar de directory waar de bibliotheek "Web3j" is geïnstalleerd gaan door het volgende commando uit te voeren, een belangrijk punt is dat de directory wordt verborgen na dr het commando "cd" zetten we een punt "." En dan de directory zonder spaties, als volgt:

```
$ cd .web3j
```

Eenmaal binnen testen we of de bibliotheek goed werkt met het volgende commando:

```
./web3j versie
```

Het resulteert in iets wat erg lijkt op:

```

$ ls
source.sh web3j web3j-4.5.16
$ ./web3j version

Version: 4.5.16
Build timestamp: 2020-03-06 14:13:49.943 UTC
$ 
```

Later zullen we een portemonnee maken voor gebruik in de blokketenomgeving van Ethereum op de volgende manier:

```
./web3j portemonnee maken
```

Het vorige commando geeft ons het adres van 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
```

Voor de Bitcoin-omgeving hebben we twee opties: het gebruik van het blok () dat de Bitcoin-account en de respectievelijke publieke en private sleutels genereert. We kunnen deze sleutels gebruiken om te integreren in de Bitcoin-omgeving door de "Bitcoij"-java-bibliotheek te installeren.

\$ npm installen bitcoinj

Het resultaat is een bestand in JSON-formaat met het adres en de gecodeerde gegevens die later worden gebruikt om de privé-sleutel voor de zojuist aangemaakte account te genereren.

Dit JSON-bestand wordt aangemaakt in een standaardmap genaamd keystore.

Vanaf hier kunnen we al bewerkingen uitvoeren met behulp van en/of het uitvoeren van het Web3j commando.

We kunnen dit doen door gebruik te maken van de extensie (ConnectorSSHClient).

Om te weten hoe de verschillende parameters en bewerkingen van de "Web3j" bibliotheek te gebruiken kunnen we ons helpen door de documentatie op de officiële site te raadplegen.

<https://docs.web3j.io/>



```
$ npm install bitcoinj
+ bitcoinj@0.0.0
added 1 package from 1 contributor and audited 2
09 packages in 20.528s
$
```

Om gebruik te maken van deze bibliotheek zullen we vertrouwen op de officiële site.

<https://bitcoinj.github.io/>

Een tweede optie om te integreren in de Bitcoin-blokketenomgeving is het installeren van het Bitcoind-pakket voor Termux zoals hieronder weergegeven.

\$ apt installeer 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-packa
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 directorie
s currently installed.)
Preparing to unpack .../bitcoin_0.20.0_arm.deb .
.
Unpacking bitcoin (0.20.0) ...
Setting up bitcoin (0.20.0) ...
$
```

Als we nu de bitcoind agent op de Termux terminal draaien zal deze automatisch een adres in de directory aanmaken:

/data/data/com.termux/bestanden/hme/.bitcoin

In dit geval van Bitcoin vertrouwen we op de volgende documentatie van de "Bitcoind"-agent.

In dit geval kunnen we ook de extensie (**ConnectorSSHClient**) gebruiken om de "bitcoind" agent uit te voeren, afhankelijk van elke behoefte.

Beschrijving van de parameters voor gebruik met bitcoind.

NAAM

bitcoind - lanceer Bitcoin Core Daemon

SYNOPSIS

bitcoind [*options*] Start *Bitcoin Core Daemon*

BESCHRIJVING

Start Bitcoin Core Daemon

OPTIES

-?

Print dit hulpbericht en sluit af

-alertnotify=<cmd>

Voer het commando uit wanneer een relevante waarschuwing wordt ontvangen of we een zeer lange vork zien (%s in cmd wordt vervangen door het bericht)

-assumevalid=<hex>

Als dit blok in de string zit, neem dan aan dat hij en zijn voorouders geldig zijn en sla mogelijk je schrijfverificatie over (0 voor verify all, standaard:
0000000000000000f1c54590ee18d15ec70e68cd4cfbadb1b4f11697eee, testnet:
0000000000000037a8cd3e06cd5edbfe9dd1dbcc5dacab279376ef7fcf2b4c75).

-blockmeldung=<cmd>

Voer het commando uit wanneer het beste blok verandert (%s in cmd wordt vervangen door het blok hash)

-blockreconstructionextratxn=<n>

Extra transacties om in het geheugen te bewaren voor compacte blokreconstructies (standaard: 100)

-blocksdir=<dir>

Geef de blokmap op (standaard: <datadir>/blokken)

-alleen in blok

Als u transacties van netwerkpartners weigert. Portefeuille- of RPC-transacties worden niet beïnvloed. (standaard: 0)

-conf=<archief>

Geef het configuratiebestand op. De relatieve paden worden voorafgegaan door de locatie van de datadir. (standaard: bitcoin.conf)

-daemon

Het loopt op de achtergrond als een demon en accepteert de commando's

-datadir=<dir>

Geef de gegevensmap op

-dbcache=<n>

Maximale database cachegrootte <n> MiB (4 tot 16384, standaard: 450). Daarnaast wordt het ongebruikte mempoolgeheugen voor deze cache gedeeld (zie **-maxmempool**).

-debuglogfile=<file>

Geef de locatie van het debug-logbestand op. Relatieve paden worden voorafgegaan door een netwerkspecifieke datadirlocatie. (**-nodebuglogfile** uit te schakelen; standaard: debug.log)

-inclusiefconf=<bestand>

Geef een extra configuratiebestand op, ten opzichte van het **-datadir-pad** (kan alleen worden gebruikt vanuit het configuratiebestand, niet vanuit de opdrachtregel)

-loadblock=<bestand>

Importeer blokken uit het externe bestand blk000???.dat bij het begin

-maxmempool=<n>

Houd de transactie-geheugenpool onder <n> megabytes (standaard: 300)

-maxorphantx=<n>

Houd maximaal <n> ontkoppelbare transacties in het geheugen (standaard: 100)

-mempoolexpiry=<n>

Bewaar transacties niet langer dan <n> uur in de mempool (standaard: 336)

-par=<n>

Stelt het aantal verificatiedraden in het streepje in (-6 tot 16, 0 = auto, <0 = laat alle kernen vrij, standaard: 0)

-persistmempool

Als u de mempool opslaat wanneer u deze afsluit en laadt wanneer u opnieuw opstart (standaard: 1)

-pid=<bestand>

Geef het PID-bestand op. Relatieve paden worden voorafgegaan door een netwerkspecifieke datadirlocatie. (standaard: bitcoind.pid)

-punza=<n>

De behoefte aan opslag verminderen door het snoeien (verwijderen) van oude blokken mogelijk te maken. Hierdoor kan de RPC-snoeketten worden opgeroepen om specifieke blokken te verwijderen en kunnen oude blokken automatisch worden gesnoeid als er een doelmaat in MIB is voorzien. Deze modus is niet compatibel met **-txindex** en **-rescan**.

Waarschuwing: Om deze instelling om te keren is het noodzakelijk om de gehele blokketen opnieuw te downloaden (standaard: 0 = bloksnoeien uitschakelen, 1 = handmatig snoeien via RPC toestaan, >=550 = blokbestanden automatisch snoeien om onder de in MIB aangegeven doelgrootte te blijven).

-reindex

Herstelt de stringstatus en blokindex van blk*.dat-bestanden op schijf

-reindex-chainstate

Reconstrueer de toestand van de keten uit de momenteel geïndexeerde blokken. In de snoeimodus of als de blokken op de schijf beschadigd zijn, gebruik dan de volledige **herindexering**.

-zucht

Maak nieuwe bestanden aan met standaard systeemrechten, in plaats van umask 077
(alleen effectief met portfoliofunctionaliteit uitgeschakeld)

-txindex

Behoud van een volledige transactie-index, gebruikt door de getrawtransaction rpc-oproep (standaard: 0)

-versie

Print & Go versie

Aansluitmogelijkheden:

-addnode=<ip>

Voeg een node toe om verbinding mee te maken en probeer de verbinding open te houden (zie het RPC-commando help van het "addendum" voor meer informatie). Deze optie kan meerdere malen worden gespecificeerd om meerdere knooppunten toe te voegen.

-banscore=<n>

Drempel voor het loskoppelen van wangedrag van collega's (standaard: 100)

-Ondertussen...

Aantal seconden om te voorkomen dat collega's die zich misdragen, opnieuw verbinding maken (standaard: 86400)

-bind=<addr>

Bind jezelf vast aan het opgegeven adres en luister er altijd naar. Gebruik de [host]:poortnotatie voor IPv6

-aansluiting=<ip>

Maak alleen verbinding met het gespecificeerde knooppunt; **-niet verbinden schakelt** automatische verbindingen **uit** (de regels voor dit paar zijn hetzelfde als voor **-addnode**). Deze optie kan meerdere malen worden gespecificeerd om verbinding te maken met meerdere knooppunten.

-ontdek

Ontdek uw eigen IP adressen (standaard: 1 bij het luisteren en niet **-extern** of **-proxy**)

-dns

Sta DNS-zoekopdrachten toe voor **-addnode**, **-seednode** en **-connect** (standaard: 1)

-dnsseed

Adresvraag via DNS-zoekopdracht, indien laag op de adressen (standaard: 1 tenzij -**verbinding wordt** gebruikt)

-enablebip61

Stuur afwijzingsberichten door BIP61 (standaard: 1)

-externalip=<ip>

Geef uw eigen openbaar adres op

-geforceerd zaad

Altijd de adressen van collega's opvragen via DNS lookup (standaard: 0)

-luisteren naar

Accepteer verbindingen van buitenaf (standaard: 1 indien geen **-proxy** of **-verbinding**)

-belichting

Maak automatisch de Tor verborgen service aan (standaard: 1)

-maxconnecties=<n>

Onderhouden van maximaal <n> verbindingen met collega's (standaard: 125)

-maximale-ontvangstbuffer=<n>

Maximale ontvangstbuffer per verbinding, <n>*1000 bytes (standaard: 5000)

-maxsendbuffer=<n>

Maximale zendbuffer per verbinding, <n>*1000 bytes (standaard: 1000)

-maximale tijdstelling

Het maximum dat is toegestaan voor de aanpassing van de gemiddelde tijdcompensatie van de paren. Het lokale tijdspectief kan worden beïnvloed door de voor- of achterwaartse paren met dit bedrag. (standaard: 4200 seconden)

-maxuploadtarget=<n>

Probeer het uitgaande verkeer onder het opgegeven doel te houden (in MiB voor 24 uur),
0 = geen limiet (standaard: 0)

-onion=<>ip:poort>

Gebruik een aparte SOCKS5 proxy om peers te bereiken via de verborgen diensten van
Tor, stel **-middagion** in om uit te schakelen (standaard: **-proxy**)

-onlynet=<net>

Maak alleen uitgaande verbindingen via het <net> netwerk (ipv4, ipv6 of ui). Inkomende
verbindingen worden niet beïnvloed door deze optie. Deze optie kan meerdere malen
worden gespecificeerd om meerdere netwerken toe te staan.

-paarfilters

Ondersteunt het blokkeren van filtering en transactie met bloefilters (standaard: 1)

-permitbaremultisig

Relais niet P2SH multisig (standaard: 1)

-port=<port>

Luister naar de verbindingen in 'poort' (standaard: 8333, testnet: 18333, regtest: 18444)

-proxy=<ip: poort>

Maak verbinding via SOCKS5 proxy, stel **-noproxy** in op uit (standaard: uit)

-proxyrandomize

Willekeurig de referenties voor elke proxy-verbinding Dit maakt Tor-stroomisolatie
mogelijk (standaard: 1)

-zaadknop=<ip>

Maak verbinding met een knooppunt om de koppelingsadressen op te halen en verbreek
de verbinding. Deze optie kan meerdere malen worden gespecificeerd om verbinding te
maken met meerdere knooppunten.

-timeout=<n>

Geef de time-out van de verbinding op in milliseconden (minimum: 1, standaard: 5000)

-torcontrol=<ip>:<port>:<<<>.

Tor control poort te gebruiken als uien luisteren is ingeschakeld (standaard: 127.0.0.1:9051)

-password=<pass>

Tor control poortwachtwoord (standaard: blanco)

-upnp

Gebruik UPnP om de luisterpoort toe te wijzen (standaard: 0)

-whitebind=<addr>

Link naar een bepaald adres en de whitelisted partners die er verbinding mee maken.
Gebruik de [host]:poortnotatie voor IPv6

-whitelist=<IP-adres of netwerk>

De witte lijsten zijn verbonden met het opgegeven IP-adres (bijv. 1.2.3.4) of het geannoteerde netwerk van de CIDR (bijv. 1.2.3.0/24). Het kan meerdere malen worden gespecificeerd. Whitelisted paren kunnen niet worden verboden door de DoS

Portefeuilleopties:

-adrestype

Welk type adressen te gebruiken ("legacy", "p2sh-segwit", of "bech32", standaard: "p2sh-segwit")

-vermijd gedeeltelijke kosten

Groepeer de uitgangen op richting, waarbij alle of geen uitgangen worden geselecteerd, in plaats van op elke uitgang. De privacy wordt verbeterd omdat een adres slechts eenmaal wordt gebruikt (tenzij iemand het verzendt nadat het is uitgegeven), maar kan leiden tot iets hogere koersen omdat de keuze van de valuta's suboptimaal kan zijn als gevolg van de toegevoegde beperking (standaard: 0).

-typeverandering

Welke wisselkoers te gebruiken ("legacy", "p2sh-segwit", of "bech32"). De standaardinstelling is hetzelfde als **-addresstype**, behalve dat **-addresstype=p2sh-segwit** native segwit-uitvoer gebruikt bij het verzenden naar een native segwit-adres)

-uit de portemonnee

Laad de portemonnee niet en schakel RPC-oproepen uit de portemonnee uit.

-discardfee=<amt>

Het tarief (in BTC/kB) dat de tolerantie aangeeft om de wijziging weg te gooien door deze toe te voegen aan het tarief (standaard: 0,0001). Opmerking: Een output wordt weggegooid als het stof in dit tempo is, maar we zullen altijd weggooien tot de stoftransmissiesnelheid en een weggooisnelheid daarboven wordt beperkt door de schatting van de snelheid voor het langere doel.

-fallbackfee=<amt>

Een vergoedingstarief (in BTC/kB) dat moet worden gebruikt wanneer de raming van de vergoeding onvoldoende gegevens bevat (standaard: 0,0002).

-keypool=<n>

Stel de sleutelgrootte van de pool in op <n> (standaard: 1000)

-mintxfee=<amt>

Tarieven (in FTK/kB) lager dan dit worden beschouwd als een nultarief voor de creatie van de transactie (standaard: 0,00001).

-paytxfee=<amt>

Tarief (in BTC/kB) om toe te voegen aan de transacties die u stuurt (standaard: 0,00)

-rescan

Rescan de blokken om te zoeken naar ontbrekende portefeuilletransacties in het begin

-salvagewallet

Poging om de privé-sleutels van een corrupte portemonnee in de kofferbak op te halen

-verspilling van de uitwisseling van informatie

Geef de onbevestigde wijziging door bij het verzenden van de transacties (standaard: 1)

-txconfirmtarget=<n>

Als er geen betalingskosten zijn vastgesteld, moet u voldoende kosten in rekening brengen voor transacties om de bevestiging gemiddeld binnen n blokken te beginnen (standaard: 6).

-portefeuillebeheer

Update van de portefeuille naar het nieuwste formaat aan het begin

-portefeuille=<pad>

Specificeer het pad van de portfoliodatabase. U kunt het meerdere malen opgeven om meerdere portefeuilles te laden. Het pad wordt geïnterpreteerd in relatie tot <walletdir> als het niet absoluut is, en wordt aangemaakt als het niet bestaat (zoals een directory met een wallet.dat bestand en logbestanden). Voor achterwaartse compatibiliteit zal het ook de namen van bestaande gegevensbestanden in <walletdir> accepteren.

-palletuitzending

Maak de portefeuilleverspreidingstransacties (standaard: 1)

-walletdir=<dir>

Specificeer de map om de portefeuilles op te slaan (standaard: <datadir>/portfolio's indien aanwezig, anders <datadir>)

-walletnotify=<cmd>

Uitvoeringsopdracht bij wijziging van een portefeuilletransactie (%s in cmd wordt vervangen door TxID)

-walletrbf

Stuur de transacties met de optie om het hele RBF te activeren (alleen RPC, standaard: 0)

-zapwallettxes=<modus>

Verwijder alle transacties uit de portefeuille en haal alleen de delen van de blokkeringketen op via **-rescan** aan het begin (1 = tx-metadata bewaren, bijv. informatie over betalingsverzoeken, 2 = tx-metadata laten vallen)

ZeroMQ-meldingsopties:

-zmqpubhashblock=<adres>

Schakel het publicatieblok in 'adres' in

-zmqpubhashblockhwm=<n>

Stel het publicatieblok voor uitgaande berichten in met een hoog watermerk (standaard: 1000)

-zmqpubhashtx=<adres>

Maak het mogelijk om de hasj-transactie te publiceren in 'adres'.

-zmqpubhashtxhwm=<n>

Stel het uitvoerbericht van de hoogwatermerk hashtransactie publiceren in (standaard: 1000)

-zmqpubrawblock=<adres>

Schakel het ruwe publicatieblok in 'adres' in

-zmqpubrawblockhwm=<n>

Stel Raw Block Outgoing Message High Watermark in (standaard: 1000)

-zmqpubrawtx=<adres>

Maak de publicatie van de ruwe transactie in 'adres' mogelijk.

-zmqpubrawtxhwm=<n>

Stel bruto transactie-uitvoerbericht hoog watermerk publiceren in (standaard: 1000)

Debugging/Testing opties:

-debug=<categorie>

Informatie over het debuggen van de uitvoer (standaard: **-nodebug**, met vermelding van 'categorie' is optioneel). Als <categorie> niet wordt geleverd, of als <categorie> = 1, wordt alle debug-informatie uitgevoerd. <categorie> kan zijn: net, tor, mempool, http, bank, zmq, db, rpc, schatting, addrman, selectcoins, reindex, cmpctblock, rand, snoei, proxy, mempoolrej, libevent, coindb, qt, leveldb.

-debugexclude=<categorie>

Sluit het debuggen van informatie uit een categorie uit. Kan worden gebruikt in combinatie met **-debug=1** om debug-records te genereren voor alle categorieën behalve één of meer gespecificeerde categorieën.

-help-debug

Printhalpbericht met debugging-opties en uitgang

-logips

Neem de IP-adressen op in de debug-uitgang (standaard: 0)

-logtimestamps

Bereid de debug-uitvoer voor met de tijdstempel (standaard: 1)

-maxtxfee=<amt>

Maximale totale vergoedingen (in FTK) voor gebruik op één enkele portefeuilletransactie of op een brutotransactie; indien te laag ingesteld, kan het grote transacties afbreken (standaard: 0,10).

-bedrukking voor de console

Stuur traceer/debugging informatie naar de console (standaard: 1 als er geen **-daemon** is). Om het loggen naar een bestand uit te schakelen, stel **-nodebuglogfile** in)

-shrinkdebugfile

Debug.logbestand verminderen bij het opstarten van de klant (standaard: 1 wanneer er geen **-debug** is)

-uacomment=<cmt>

Voeg een opmerking toe aan de user agent string

Kettingselectiemogelijkheden:

-testnet

Gebruik de testketen...

Node heruitzendingsopties:

-bytespersigop

Byte-equivalenten per sigop bij uitzend- en mijnbouwtransacties (standaard: 20)

-dieetdrager

Relais- en mijndragertransacties (standaard: 1)

-datacarrierize

Maximale omvang van de gegevens in de transacties van de gegevensdragers die we doorsturen en extraheren (standaard: 83)

-mempoolvervanging

Vervanging van transacties in de geheugenpool mogelijk maken (standaard: 1)

-minrelaytxfee=<amt>

Vergoedingen (in FTK/kB) die lager zijn dan deze worden beschouwd als een nultarief voor de doorgifte, extractie en creatie van transacties (standaard: 0,00001).

-whitelistforcerelay

Het dwingen van de doorgifte van transacties van partners op de witte lijst, zelfs als de transacties al in een mempool stonden of in strijd zijn met het lokale doorgiftebeleid (standaard: 0).

-whitelistrelay

Accepteer verzonden transacties die zijn ontvangen van paren op de witte lijst, zelfs als er geen transacties worden verzonden (standaard: 1).

Blokkeer de creatiemogelijkheden:

-blockmaximumgewicht=<n>

Stel het maximale gewicht van het BIP141-blok in (standaard: 3996000)

-blockmintxfee=<amt>

Stel het laagste commissietarief (in FTK/kB) in voor transacties die zijn opgenomen in de blokaanmaak. (standaard: 0,00001)

RPC-serveropties:

-Restaurant

Accepteer publieke REST-verzoeken (standaard: 0)

-rpcallowip=<ip>

Laat JSON-RPC-verbindingen van de gespecificeerde bron toe. Ze zijn geldig voor <ip> een enkel IP (bijv. 1.2.3.4), een netwerk/netwerkmasker (bijv. 1.2.3.4/255.255.255.0), of een netwerk/CIDR (bijv. 1.2.3.4/24). Deze optie kan meerdere malen worden gespecificeerd

-rpcauth=<userpw>

Gebruikersnaam en wachtwoord HMAC-SHA-256 voor JSON-RPC verbindingen. Het veld 'userpw' komt in het formaat: 'username': 'SALT': \$ 'HASH'. Een canoniek pythonscript is opgenomen in share/rpcauth. De client maakt dan normaal gesproken verbinding met de rpcuser=<USERNAME>/rpcpassword=<PASSWORD> argumentpaar. Deze optie kan meerdere malen worden gespecificeerd

-rpcbind=<addr>[:poort]

Link naar een bepaald adres om naar JSON-RPC-verbindingen te luisteren. Stel de RPC-server niet bloot aan onbetrouwbare netwerken zoals het openbare internet! Deze optie wordt genegeerd, tenzij **-rpcallowip** ook wordt gepasseerd. De poort is facultatief en vervangt **-rpcport**. Gebruik de [host]:poortnotatie voor IPv6. Deze optie kan meerdere malen worden opgegeven (standaard: 127.0.0.1 en ::1 d.w.z. localhost)

-rpccookiefile=<loc>

Locatie van het autorisatiecookie. Relatieve paden worden voorafgegaan door een netwerkspecifieke datadirlocatie. (standaard: data dir)

-rpcpassword=<pw>

Wachtwoord voor JSON-RPC-verbindingen

-rpcport=<port>

Luister naar JSON-RPC-verbindingen in 'poort' (standaard: 8332, testnet: 18332, regtest: 18443)

-rpcserialversie

Stelt de ruwe transactiereeks of de blokhex in non-verbale modus in, niet segwit(0) of segwit(1) (standaard: 1)

-rpcthreads=<n>

Stel het aantal threads in om RPC-oproepen te behandelen (standaard: 4)

-rpcuser=<gebruiker>

Gebruikersnaam voor JSON-RPC verbindingen

-server

Accepteer commandoregelcommando's en JSON-RPC

32. Licentie en gebruik van software.

Android

<https://source.android.com/setup/start/licenses>

Termux

<https://github.com/termux/termux-app/blob/master/LICENSE.md>

Knooppunt

<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>

sqlite-to-rest

<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>

Tor Netwerk

<https://github.com/torproject/tor/blob/master/LICENSE>

Synchronisatie van het netwerk

<https://forum.syncthing.net/t/syncthing-is-now-mplv2-licensed/2133>

OpenSSH

<https://www.openssh.com/features.html>

Stopverf SSH

<https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

MIT App Inventor 2 Companion en App Inventor Blockly Uitvinden

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

SQLite Expert Personal -freeware

<http://www.sqliteexpert.com/download.html>

Apache mier

<https://ant.apache.org/license.html>

WGET

<https://www.gnu.org/software/wget/>

OpenJDK

<https://openjdk.java.net/legal/>

Externe uitbreidingen:

JSONTOOLS

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

Licensing opensource en commerciële versies van Mini BlocklyChain systeem raadpleeg de officiële website <http://www.openqbit.com>.

Mini BlocklyChain, MiniBlockly, BlocklyCode, MiniBlockMiniChain, QBlockly son marcas registradas por OpenQbit.

Mini BlocklyChain is in het publieke domein.

Alle code en documentatie in Mini BlocklyChain is door de auteurs aan het publieke domein gewijd. Alle code-auteurs en vertegenwoordigers van de bedrijven waarvoor zij werken hebben beëdigde verklaringen ondertekend waarin zij hun bijdragen aan het publieke

domein opdragen en de originelen van die beëdigde verklaringen zijn opgeslagen in een kluisje in het hoofdkantoor van OpenQbit Mexico. Het staat iedereen vrij om de originele Mini BlocklyChain (OpenQbit) extensies te publiceren, te gebruiken of te distribueren, hetzij als broncode of als gecompileerde binaries, voor elk doel, commercieel of niet-commercieel, en op welke manier dan ook.

De vorige paragraaf is van toepassing op de code en documentatie die in de Mini BlocklyChain wordt geleverd, die delen van de Mini BlocklyChain-bibliotheek die daadwerkelijk een grotere toepassing groeperen en verzenden. Sommige scripts die als onderdeel van het compilatieproces worden gebruikt (bijvoorbeeld 'configuratie'-scripts die door autoconf worden gegenereerd) kunnen in andere open-sourcelicenties worden opgenomen. Geen van deze compilatiescripts maken echter deel uit van de uiteindelijke Mini BlocklyChain-bibliotheek, dus de licenties die aan deze scripts verbonden zijn, mogen geen rol spelen bij de beoordeling van uw rechten om de Mini BlocklyChain-bibliotheek te kopiëren en te gebruiken.

Alle te leveren code in de Mini BlocklyChain is vanaf het begin geschreven. Er is geen code van andere projecten of van het open internet gehaald. Elke regel code kan worden getraceerd naar de oorspronkelijke auteur, en al die auteurs hebben publieke domein toewijzingen in het bestand. Daarom is de Mini BlocklyChain code basis is schoon en niet besmet met code gelicenseerd van andere open source projecten, niet open bijdrage

Mini BlocklyChain is open source, wat betekent dat je zoveel kopieën kunt maken als je wilt en kunt doen wat je wilt met die kopieën, zonder beperking. Maar Mini BlocklyChain is niet open source. Om Mini BlocklyChain in het publieke domein te houden en om ervoor te zorgen dat de code niet besmet is met eigen of gelicenseerde inhoud, accepteert het project geen patches van onbekende mensen. Alle code in de Mini BlocklyChain is origineel, omdat deze speciaal is geschreven voor gebruik door Mini BlocklyChain. Er is geen code gekopieerd uit onbekende bronnen op het internet.

Mini BlocklyChain is in het publieke domein en vereist geen licentie. Toch willen sommige organisaties een wettelijk bewijs van hun recht om Mini BlocklyChain te gebruiken. De omstandigheden waarin dit gebeurt zijn onder meer de volgende:

- Uw bedrijf wil een vergoeding voor claims van inbreuk op het auteursrecht.
- U gebruikt Mini BlocklyChain in een rechtsgebied dat het publieke domein niet erkent.
- U gebruikt Mini BlocklyChain in een rechtsgebied dat het recht van een auteur om zijn of haar werk in het publieke domein te plaatsen niet erkent.
- U wilt een tastbaar wettelijk document als bewijs dat u het wettelijke recht heeft om Mini BlocklyChain te gebruiken en te distribueren.
- Uw juridische afdeling vertelt u dat u een licentie moet kopen.

Als een van de bovenstaande omstandigheden op u van toepassing is, zal OpenQbit, het bedrijf dat alle Mini BlocklyChain ontwikkelaars in dienst heeft, u een Mini BlocklyChain Titel Garantie verkopen. Een Title Warranty is een wettelijk document dat stelt dat de geclaimde auteurs van Mini BlocklyChain de echte auteurs zijn, en dat de auteurs het wettelijke recht hebben om de Mini BlocklyChain aan het publieke domein te wijden, en dat OpenQbit zich krachtig zal verdedigen tegen de licentieclaims. Alle opbrengsten uit de verkoop van Mini BlocklyChain's titelgaranties worden gebruikt om de voortdurende verbetering en ondersteuning van Mini BlocklyChain te financieren.

Bijdragende Code

Om de Mini BlocklyChain volledig vrij en royaltyvrij te houden, accepteert het project geen patches. Als u een voorgestelde wijziging wilt aanbrengen en een patch wilt opnemen als bewijs van het concept, dan zou dat geweldig zijn. Wees echter niet beledigd als we uw patch vanaf nul herschrijven. Het type van niet-commerciële of open source licentie die het gebruikt in deze modaliteit en een aantal soortgelijke zonder aankoop van ondersteuning, hetzij individueel of zakelijk gebruik, ongeacht de grootte van het bedrijf zal worden geregeld door de volgende wettelijke bepalingen.

Garantieclausule. Tenzij vereist door de toepasselijke wetgeving of schriftelijk overeengekomen, levert de Licentiegever het Werk (en elke Bijdrager levert zijn Bijdragen) "zoals", **ZONDER GARANTIES OF VOORWAARDEN VAN EEN KIND**, explicet of impliciet, met inbegrip van, maar niet beperkt tot, alle garanties of voorwaarden van TITEL, NONINFRINGEMENT, MERCHANTABILITY OF VEILIGHEID VOOR EEN BIJZONDERLIJK DOEL. U bent als enige verantwoordelijk voor het bepalen van het juiste gebruik of de juiste herverdeling van het Werk en voor het nemen van de risico's die verbonden zijn aan de uitoefening van uw machtingen onder deze Licentie.

Eventuele financiële of andere verliezen die door het gebruik van deze software worden geleden, komen ten laste van de betrokken partij. Alle juridische geschillen zullen door de partijen uitsluitend worden voorgelegd aan de rechtbanken in het rechtsgebied van Mexico-Stad, land Mexico.

Voor commerciële ondersteuning, gebruik en licentieverlening moet een overeenkomst of contract worden gesloten tussen OpenQbit of haar bedrijf en de geïnteresseerde partij.

De voorwaarden voor distributiemarketing kunnen zonder voorafgaande kennisgeving worden gewijzigd, ga naar de officiële website www.openqbit.com om eventuele wijzigingen in de ondersteunings- en licentieclausules niet-commercieel en commercieel te bekijken.

Elke persoon, gebruiker, private of publieke entiteit van welke juridische aard dan ook of uit welk deel van de wereld dan ook die de software gewoonweg gebruikt, aanvaardt zonder voorwaarden de clausules die in dit document zijn vastgelegd en die op elk moment in het portaal van www.openqbit.com kunnen worden gewijzigd zonder voorafgaande

kennisgeving en naar eigen goeddunken van OpenQbit kunnen worden toegepast in niet-commercieel of commercieel gebruik.

Alle vragen en informatie over Mini BlocklyChain moeten worden gericht aan de App Inventor gemeenschap of aan de verschillende Blockly systeem gemeenschappen zoals ze zijn: AppBuilder, Trunkable, etc. en/of naar de mail opensource@openqbit.com voor de vraag van vragen kan het antwoord van 3 tot 5 werkdagen in beslag nemen.

Ondersteuning bij commercieel gebruik.

support@openqbit.com

Verkoop voor commercieel gebruik.

sales@openqbit.com

Juridische informatie en vragen of problemen met betrekking tot vergunningen

legal@openqbit.com

