



Installation, Konfiguration und Verwaltung.

# Benutzerhandbuch

Version 1.0 Beta

Juli 2020.

MiniBlocklyChain ist ein eingetragenes Warenzeichen von OpenQbit Inc. unter einer freien und kommerziellen Lizenz. Nutzungsbedingungen unter: [www.OpenQbit.com](http://www.OpenQbit.com)

## Inhalt

1.	Einleitung.....	3
2.	Was ist ein öffentliches oder privates Netzwerk in der Blockkettenregelung? .....	4
3.	Was ist blockweise Programmierung? .....	4
4.	Was ist Termux? .....	5
5.	Was ist eine Mini-BlocklyChain? .....	5
6.	Prozessarchitektur in Mini-BlocklyChain.....	10
7.	BlocklyChain-Funktionalitätsdiagramm (Mini BlocklyChain). .....	13
8.	Was ist der Mini-BlocklyCode?.....	14
9.	Installation eines Mini-BlocklyChain-Kommunikationsnetzes .....	15
10.	Synchronisierung in Systemknoten (Mobiltelefon) Minuten und Sekunden.....	37
11.	Speicherkonfiguration innerhalb von Termux. ....	45
12.	Tor"-Netzwerkinstallation und "Syncthing"-Installation.....	46
13.	Installation der Datenbank "Redis" und des SSH (Secure Shell)-Servers.....	48
14.	Konfiguration des SSH-Servers auf dem Mobiltelefon (Smartphone). .....	49
15.	Tor"-Netzwerkkonfiguration mit SSH (Secure Shell)-Dienst. ....	56
16.	Peer-to-Peer-Systemkonfiguration mit manueller Synchronisierung.....	59
17.	Ambientes Blockly (App Inventor, AppyBuilder und Thunkable).....	70
18.	Was ist der Quantennachweis (PQu)? .....	71
19.	Definition und Verwendung von Blöcken in Mini BlocklyChain.....	74
20.	Verwendung von Blöcken für SQLite-Datenbank (MiniSQLite-Version) .....	100
21.	Definition und Verwendung von Sicherheitsblöcken.....	104
22.	Einstellen von Sicherheitsparametern in Mini BlocklyChain.....	114
23.	Anhang "Erstellung von KeyStore & PublicKeys-Datenbanken". .....	117
24.	Anhang "RESTful SQLite GET/POST-Befehle".....	131
25.	Anhang "Java Code SQLite-Redis Connector". .....	137
26.	Anhang "Mini-BlocklyChain für Entwickler".....	140
27.	Anhang "BlocklyCode Smart Contracts".....	152
28.	Anhang "OpenQbit Quantum Computing". .....	158
29.	Anhang "Erweiterte Blöcke für SQLite-Datenbank" .....	162
30.	Anhang "Beispiel für die Erstellung eines Mini-BlocklyChain-Systems". .....	162
31.	Anhang "Integration mit Ethereum & Bitcoin-Umgebungen".....	187
32.	Lizenzierung und Nutzung von Software. .....	206

## 1. Einleitung.

Die Blockkette wird im Allgemeinen mit Bitcoin und anderen Krypto-Währungen in Verbindung gebracht, aber diese sind nur die Spitze des Eisbergs, da sie nicht nur für digitales Geld verwendet wird, sondern für alle Informationen, die für Benutzer und/oder Unternehmen einen Wert haben können. Diese Technologie, deren Ursprünge auf 1991 zurückgehen, als Stuart Haber und W. Scott Stornetta die erste Arbeit an einer Kette von kryptografisch gesicherten Blöcken beschrieben, wurde erst 2008 bemerkt, als sie mit der Einführung der Bitcoin populär wurde. Gegenwärtig wird seine Verwendung jedoch in anderen kommerziellen Anwendungen nachgefragt, und es wird prognostiziert, dass sie in mittlerer Zukunft auf mehreren Märkten, wie z.B. bei Finanzinstitutionen oder im Internet der Dinge (Internet of Things IoT), neben anderen Sektoren, wachsen wird.

Die Blockkette, besser bekannt unter dem Begriff Blockkette, ist ein einzelner, vereinbarter Datensatz, der über mehrere Knoten (elektronische Geräte wie PCs, Smartphones, Tablets usw.) in einem Netzwerk verteilt ist. Im Falle der Krypto-Währungen können wir uns das als das Buchhaltungsbuch vorstellen, in dem jede der Transaktionen aufgezeichnet wird.

Seine Funktionsweise kann kompliziert zu verstehen sein, wenn wir auf die internen Details seiner Umsetzung eingehen, aber die Grundidee ist einfach zu verfolgen.

Sie wird in jedem Block gespeichert:

1.- eine Anzahl gültiger Datensätze oder Transaktionen,

2.- Informationen über diesen Block,

3.- seine Verknüpfung mit dem vorherigen Block und dem nächsten Block durch den Hash jedes Blocks –ein eindeutiger Code, der wie der Fingerabdruck des Blocks aussehen würde.

Daher hat **jeder Block** einen **bestimmten und unverrückbaren Platz innerhalb der Kette**, da jeder Block Informationen aus dem Hash des vorherigen Blocks enthält. Die gesamte Kette wird auf jedem Netzwerkknoten gespeichert, der die Blockkette bildet, so dass **eine exakte Kopie der Kette auf allen Netzwerkteilnehmern gespeichert wird**.

Wenn neue Datensätze erstellt werden, werden sie zunächst von den Netzwerkknoten geprüft und validiert und dann zu einem neuen Block hinzugefügt, der mit der Kette verknüpft ist.

Nun stellen Sie sich vor, dass dieses Netzwerk von Geräten, die untereinander kommunizieren, die Fähigkeit hat, ohne das Eingreifen einer Person zu interagieren, d.h. der große Vorteil der Blockkette ist, dass sie autonome Entscheidungen treffen kann, was Vorteile in den Reaktionszeiten des Dienstes für die Nutzer hat, 24 Stunden am Tag verfügbar

ist, die Kosten in der Wirtschaft minimiert und vor allem ein Maß an Sicherheit bereits getestet, aus diesem und anderen Gründen hat sich so beliebt in der Verwendung in verschiedenen öffentlichen und privaten Sektoren.

## 2. Was ist ein öffentliches oder privates Netzwerk in der Blockkettenregelung?

Öffentliches Netzwerk. - Es ist ein Netzwerk von Computern oder mobilen Geräten, die miteinander kommunizieren und die Anonymität wahren, es ist nicht bekannt, wer in diesem Netzwerk auf formale Weise in ihren Transaktionen interagiert, in dieser Art von Netzwerk kann jede Person oder Firma jederzeit interagieren und sich verbinden, weil man keine Berechtigungen braucht, um sich zu verbinden, ein Beispiel ist die Blockkette von Bitcoin, in die jeder eintreten kann, um zu kaufen oder zu verkaufen. Normalerweise sind diese Art von Netzwerken auf den Kauf und Verkauf von digitalen Geldwerten oder deren Synonym "Kryptomonien" ausgerichtet oder gerichtet, Beispiele: DogCoin, Ethereum, LiteCoin, BitCoin, Wellen, etc.

Privates Netzwerk. - Es handelt sich dabei um ein Netzwerk von Computern oder mobilen Geräten, die miteinander kommunizieren. Im Gegensatz zu öffentlichen Netzwerken benötigen private Netzwerke jedoch eine vorherige Genehmigung von irgendeiner Instanz (Firma oder Person), um sich zu verbinden und Teil dieser Art von Netzwerk zu sein. Normalerweise werden private Blockkettennetzwerke in Unternehmen oder Konzernen verwendet, um Transaktionen oder Operationen mit einer Vielzahl verschiedener Arten von Informationen durchzuführen, die einen greifbaren Wert in Form von Dokumenten, Prozessen, Genehmigungen und/oder Geschäftsentscheidungen haben können, die von Blockketten angewendet und überwacht werden, Beispiele: Finanzsektor, Versicherungssektor, Regierung, u.a.

## 3. Was ist blockweise Programmierung?

Blockly ist eine **visuelle Programmiersprache**, die sich aus einem einfachen Satz von Befehlen zusammensetzt, die wir kombinieren können, als wären sie die Teile eines Puzzles. Es ist ein sehr nützliches Werkzeug für diejenigen, die lernen wollen, wie man auf intuitive und einfache Weise programmiert, oder für diejenigen, die bereits programmieren können und das Potenzial dieser Art von Programmierung sehen wollen.

Blockly ist eine Form des Programmierens, bei der man keinen Hintergrund in irgendeiner Computersprache benötigt, weil es nur das Zusammenfügen von Grafikblöcken ist, als ob wir Lego oder ein Puzzle spielen würden, man braucht nur etwas Logik und das war's!

Jeder kann Programme für Mobiltelefone (Smartphones) erstellen, ohne sich mit diesen schwer verständlichen Programmiersprachen herumschlagen zu müssen, indem er einfach Blöcke in grafischer Form zusammensetzt.

#### 4. Was ist Termux?

Termux ist ein Android-Terminalemulator und eine Anwendung für Linux-Umgebungen, die direkt ohne Routing oder Konfiguration arbeitet. Ein minimales Basissystem wird automatisch installiert.

Wir werden Termux wegen seiner Stabilität und der einfachen Installation und Verwaltung verwenden, Sie können jedoch eine installierte Umgebung von Ubuntu Linux für Android verwenden.

In dieser Linux-Umgebung haben Sie den "Kern" der Kommunikationsprozesse der MiniBlocklyChain.

#### 5. Was ist eine Mini-Blockchain?

Mini BlocklyChain ist eine voll funktionsfähige Blockkette ist eine Technologie, die für Mobiltelefone mit dem Betriebssystem **Android** entwickelt wurde, die als Knotenpunkt für das Senden und Empfangen von Transaktionen dienen wird. Wir haben die erste Blockkettentechnologie geschaffen, die durch Blockly-Programmierung "modular" strukturiert ist, wobei jeder mit minimalen Kenntnissen und ohne Programmierkenntnisse Programme für Mobiltelefone erstellen und entwickeln und seine eigene Blockkette entweder im öffentlichen oder privaten Netzwerkmodus erstellen kann. Wenn Sie Ihre eigene digitale Währung erstellen möchten, können Sie dies tun oder eine Lösung für den Einsatz in einem Unternehmen finden. Die Möglichkeiten basieren auf den Bedürfnissen jedes realen Falls und der Geschäftslogik, die der Benutzer entsprechend seinen Anforderungen übernimmt oder erstellt.

Mini BlocklyChain ist die erste modulare Blockkette für Mobiltelefone, mit der ein Transaktionssystem in kurzer Zeit implementiert werden kann.

Bevor wir mit der Definition und Verwendung von "Modul"-Blöcken beginnen, müssen wir die grundlegenden Konzepte der Mini-Blockchain-Komponenten kennen, um zu wissen, wann, wie und wo wir sie entsprechend dem realen Fall, den wir implementieren wollen, anwenden können.

Die folgenden Konzepte betonen, auf welche Art von Benutzer sie abzielen, daher ist es für Personen, die keine Programmierkenntnisse haben, nicht wichtig, dass die Konzepte für Entwicklungsbenuutzer vollständig verstanden werden.

Grundlegende Konzepte:

**Knoten.** - Jedes mobile Gerät (Telefon, Tablet, etc.) mit einem Android-Betriebssystem, das Teil des öffentlichen oder privaten Netzwerks von Mini BlocklyChain ist, wird als Knoten dieses Netzwerks benannt.

**Haschisch.** - Es handelt sich um eine digitale Signatur, die mit einer Reihe von Daten, Zeichenketten, Dokumenten oder irgendeiner Art von digitalen Informationen verbunden ist. Diese digitale Signatur ist einzigartig und unwiederholbar und hilft beim Senden oder Empfangen von Informationen, so dass der ursprüngliche Inhalt der gesendeten Informationen nicht verändert werden kann.

**Aktiv.** - Jede Art von digitalen Daten oder Informationen, die mit einem materiellen oder immateriellen Wert für Personen oder Unternehmen gewichtet werden können (Dokumente, digitale Münzen, Musik, Video, Bilder, digitale Genehmigungen usw.).

**Transaktion.** - Informationsaustausch zwischen Knotenpunkten, die eine Art von Vermögenswert belegen.

**UXTO-Transaktion** - Es handelt sich um einen Transaktionstyp, der eine oder mehrere Transaktionen aus den Knoten verpackt und alle nicht verbrauchten Transaktionen aus jedem Knoten (UXTO: Nicht verbrauchte *Transaktionsausgaben*) als Eingabe in einer neuen Transaktion verwendet werden können. Diese Transaktionen werden in einer Warteschlange gruppiert, die zu jedem bestimmten Zeitpunkt abgearbeitet wird und die entsprechend den Anforderungen des jeweiligen Informationsflusses reguliert werden kann.

**Transaktion (Eingabe).** - Es handelt sich um eine Art von Transaktion, die auf UXTO-Transaktionen basiert. Wenn eine UXTO-Transaktionswarteschlange eingeht, wird sie von einer **Eingabetransaktion** verarbeitet, die die Transaktion als Einzahlung oder Ausgabe klassifiziert und somit einen Saldo des Endergebnisses für jeden Benutzer haben kann.

**Adresse der Quelle.** - Dies ist die Adresse der Person, die die zu verarbeitende Transaktion in der SQLite Master-Warteschlange erzeugt oder sendet. Es handelt sich um eine alphanumerische Zahl von 64 Zeichen, die vom System erstellt und überprüft wird.

**Zieladresse.** - Dies ist die Adresse der Person, die die Transaktion empfängt und sie ihrem Guthaben hinzufügt oder das gesendete Gut aufbewahrt. Es handelt sich um eine 64-stellige alphanumerische Nummer, die vom System erstellt und überprüft wird.

**SQLite Master.** - API REST-Datenbank, die die Transaktionen empfängt und eine Warteschlange erstellt, die je nach Geschäftsbedarf zu jeder bestimmten variablen oder festen Zeit verarbeitet wird.

**Datenbank-Transaktion.** - Sind die Transaktionen, die in der SQLite-Datenbank reflektiert werden, die Mini BlocklyChain-Transaktionsinformationen reserviert oder speichert.

**AES (Advanced Encryption Standard)** - Es handelt sich um einen Sicherheitsprozess, der die Daten verschlüsselt, die in der SQLite-Datenbank von Mini BlocklyChain gespeichert sind.

**Ausgewogenheit.** - Nach der Durchführung eines beliebigen Transaktionsprozesses in der Mini-BlocklyChain wird eine Bilanz der Operation entweder als materieller Wert (Kryptomoney) oder als immaterieller Wert (Geschäftsprozesse zwischen Personen oder Unternehmen) erhalten.

**Geschäftsprozess.** - Es ist jeder Prozess, der eine Art von Informationsfluss beinhaltet, um ein Ergebnis an einen Endbenutzer zu erhalten. Der Endbenutzer kann eine Person oder ein Unternehmen aus einer Vielzahl von Sektoren des privaten oder öffentlichen Sektors sein.

**Öffentlicher und privater Schlüssel.** - Es handelt sich um eine Art der Informationsverschlüsselung, bei der zwei einander zugeordnete alphanumerische Schlüssel erzeugt und verwendet werden, um sensible Informationen über öffentliche oder private Netzwerke zu versenden. Der private Schlüssel wird verwendet, um Informationen zu verschlüsseln, und wenn sie durch das Netzwerk gesendet werden, können sie nicht verändert werden oder auf den ersten Blick lesbar sein. Der öffentliche Schlüssel ist derjenige, der geteilt wird und von jeder Person oder Firma gesehen wird, und dieser wird helfen, die gesendeten Informationen zu verifizieren und nur für den gültigen Adressaten lesbar zu sein.

**Digitale Unterschrift.** - Es handelt sich um eine Signatur, die mit dem privaten Schlüssel erstellt werden kann. Mit dieser Signatur stellt der Empfänger sicher, dass die Informationen nicht verändert wurden und bestätigt, dass die Informationen für den Empfänger gültig sind.

**Digitale Adresse (Mini-BlocklyChain-Adresse).** - Es handelt sich um eine Adresse, die sich aus alphanumerischen Zeichen zusammensetzt, die für jeden Benutzer von Mini BlocklyChain eindeutig sind. Diese Adresse wird verwendet, um Transaktionen zwischen Benutzern und unabhängig davon, ob es sich um ein öffentliches oder privates Netzwerk handelt, durchzuführen.

**Magische Zahl.** - Hierbei handelt es sich um eine durch Geschäftsregeln definierte Zufallszahl für jede laufende UXTO-Transaktion, die dazu dient, den Knoten als Kandidaten für die Ausführung der UXTO-Transaktion zu autorisieren und einen neuen Block, die Mini-BlocklyChain, zu erstellen.

**Erstellung eines neuen Blocks.** - Ein neuer Block in Mini BlocklyChain ist ein Hash, der von dem Knoten erstellt und angehängt wird, der als Gewinnerkandidat für die Verarbeitung der UXTO-Transaktion hervorgegangen ist.

**PoW (Proof of Work)-Konsensusprotokoll.** - Es handelt sich um einen Test, bei dem alle Knoten ausgeführt werden, und der erste Knoten, der den Test erfolgreich abschließt, ist derjenige, der für die Ausführung der UXTO-Transaktion ausgewählt wurde. Es handelt sich um einen Algorithmus, der sich aus mathematischen Berechnungen zusammensetzt, um ein

Ergebnis (Hash) gemäß den Regeln zu erhalten, die in jeder von Mini BlocklyChain ausgeführten Transaktion angegeben sind. Dieser Algorithmus basiert auf der Abnutzung oder dem Bedarf der informationsverarbeitenden Ressourcen von Computern; im Falle von Mini BlocklyChain wurde diese strukturiert und modifiziert, um eine "magische Zahl" zu erhalten, d.h. eine Zahl, die die Autorisierung oder den Konsens der Mehrheit der Knoten hat, um diejenige zu sein, die die UXTO-Transaktion ausführen kann. Der PoW hat einen maximalen Schwierigkeitsgrad von 5, da er nur zur Ermittlung der "magischen Zahl" verwendet wird.

**PoQu-Konsensus-Protokoll** (Quantentest) - Es handelt sich um einen Test, der alle Knoten ausführt und der anfänglich vom PoW zusammengesetzt wird, um die "magische Zahl" zu erhalten, und später einen Wahrscheinlichkeitsalgorithmus ausführt, der auf einem QRNG (Quantum Random Number Generator), einem Quantenzufallszahlengenerator, basiert. Dieser Prozess stellt die Gleichheit der Wahrscheinlichkeit für alle Knoten sicher und wählt so den Knoten aus, der die UXTO-Transaktion und die Erstellung des neuen Blocks ausführt.

**Merkle-Baum.** - Hierbei handelt es sich um eine Sicherheitsmethode, die Mini BlocklyChain versichert, dass Transaktionen gültig sind oder nicht von einer externen Instanz geändert wurden. Ein Hash-Merkle-Baum ist eine binäre oder nicht-binäre Datenbaumstruktur, in der jeder Knoten, der kein Blatt ist, mit dem Hash der Verkettung der Labels oder Werte seiner Kindknoten versehen ist. Sie sind eine Verallgemeinerung von Hash-Listen und Hash-Strings.

**Redis DB.** - Echtzeit-Transaktionsdatenbank, die zur Verarbeitung und zum Senden der angeforderten neuen Transaktionen an die Knoten verwendet wird.

**SQLite DB.** - Datenbank, in der die Waagen und neue Blöcke für Mini BlocklyChain gespeichert werden, um die Integrität des Netzwerks zu gewährleisten.

**Wache.** - Sicherheits- und Datenintegritätskonnektor zwischen Redis und SQLite. Dieser Verbinder oder dieses Programm ist für die Überprüfung, Verarbeitung, Validierung, Verteilung und Übersetzung der Transaktionen an die Knotenpunkte zuständig.

**OpenSSH.** - Sicherheitskonnektor zur Ausführung von Aufgaben innerhalb des Android-Betriebssystems.

**Termux Shell-Terminal.** - Programm, in dem Sie Abhängigkeiten von Dritten finden können, um die Transaktionen von Mini BlocklyChain zu verarbeiten. In dieser Version 1.0 wird es zur einfachen Installation verwendet, in zukünftigen Versionen wird es jedoch durch das BlocklyShell-System ersetzt, das derzeit entwickelt wird.

**Brieftasche.** - Es handelt sich um den digitalen Speicher, in dem zwei grundlegende Daten für jede Transaktion aufbewahrt werden: der öffentliche und der private Schlüssel, die als Quelladresse bzw. digitale Signatur für jede durchgeführte Transaktion verwendet werden, sowie der Saldo der gesendeten oder empfangenen Transaktionen. Es handelt sich um ein

Repository, in dem die Mini-BlocklyChain-Adressen sowie die Ergebnisse der UXTO-Transaktionen (Balance) aufbewahrt werden.

Anwendungsentwickler oder Programmierer:

**Objektorientierte Programmierung (Java)** - Mini BlocklyChain ist in Java erstellt.

**BlocklyCode.** - Es ist die Bezeichnung der intelligenten Verträge, die in der Mini BlocklyChain-Umgebung ausgeführt werden können, der BlocklyCode wird in Java-Programmierung erstellt.

**OpenJDK für Android.** - Es ist die Suite von JDK und JRE für Android, in der die BlocklyCode erstellt und ausgeführt werden.

**QRNG (Quantum Random Number Generator).** - Es handelt sich um einen Quanten-Zufallszahlengenerator, Mini BlocklyChain verfügt derzeit über zwei APIs für die Generierung dieser.

**rsync.** - Es handelt sich um eine kostenlose Anwendung für Systeme vom Typ Unix und Microsoft Windows, die eine effiziente Übertragung von inkrementellen Daten bietet, die auch mit komprimierten und verschlüsselten Daten arbeitet.

**ffsend.** - Es handelt sich um eine Anwendung zur einfachen und sicheren gemeinsamen Nutzung von Dateien über die Befehlszeile.

**NTP.** - Network Time Protocol, ist ein Internet-Protokoll zur Synchronisierung der Uhren von Computersystemen durch Paket-Routing in Netzwerken mit variabler Latenzzeit.

**Termux (Entwicklung).** - Shell-Terminal mit einer breiten Palette von Open-Source-Anwendungen und Bibliotheken.

**Blockweise Module (Daten).** - Entwickler können Module integrieren, um die Funktionen von Mini BlocklyChain zu erweitern.

**Peer-to-Peer.** - Dieser Begriff bezieht sich auf die Kommunikation zwischen den Knoten auf direktem Wege, d.h. die Aktualisierung der Informationen hängt nicht von einem zentralen Server ab, sondern jeder Knoten arbeitet als zentraler Server, der zwischen allen Knoten mit den gleichen Informationen kommuniziert, was dazu beiträgt, Fehlerpunkte zu vermeiden.

**Syncthing.** - Werkzeug zum Synchronisieren von Daten oder Dateien zwischen zwei Geräten unter Verwendung des Kommunikationstyps "Peer to Peer".

**Rotes Tor.** - Dabei handelt es sich um ein verteiltes, im Internet überlagertes Kommunikationsnetz mit geringer Latenzzeit, in dem das Routing der zwischen den Nutzern ausgetauschten Nachrichten ihre Identität, d.h. ihre IP-Adresse, nicht preisgibt (netzweite Anonymität).

**Mobile Weiterleitung.** - Installationsprozess von externer Software auf Ihrem Telefon, um als Systemadministrator in Linux-Betriebssystemen (Android) einzutreten. Der Administrator-Benutzer wird "root" genannt, um Ihr Telefon drehen zu können, wird Zugriff auf jeden Prozess haben. Es ist wichtig zu beachten, dass einige Hersteller von Mobiltelefonen (Smartphones) sagen, dass die Garantie aufgrund eines Fehlers des Mobiltelefons verloren geht.

## 6. Prozessarchitektur in Mini-BlocklyChain

Mini BlocklyChain besteht aus drei Prozessen, die seine Architektur bilden, wobei der erste die Geschäftsprozesse, der zweite die Kommunikationsprozesse und der dritte die Entwicklungsumgebung für komplementäre Module und/oder die Erstellung von BlocklyCode-"intelligenten Verträgen" ist.

Die Geschäftsprozesse sind die Blöcke, die eine Reihe von Routinen bilden, um eine Transaktion entweder im öffentlichen oder privaten Netzwerk zu erstellen. Diese Art von Prozess ist die Planung des Geschäfts, das Wie, Wann, Was, Wer, Wo und weitere Attribute werden geordnet, geplant und verteilt, so dass die Hauptfunktionalität jeder gesendeten, empfangenen, gespeicherten und/oder abgelehnten Transaktion erreicht wird. Der erste Prozess besteht aus den folgenden Arten von Blöcken, die in vier Kategorien unterteilt sind:

1. Dateneingabe-Blöcke
2. Datenverarbeitungsblöcke
3. Sicherheitsblöcke.
4. Modulare Datenblöcke (Entwickler)
5. Kommunikations-Bots.

Die Dateneingabeblocks sind diejenigen, die die Transaktion mit mindestens vier Eingabeparametern erhalten (**Quelladresse**, **Zieladresse**, **aktiv**, **Daten**) und können je nach der Variablen "data" mehr haben, dies kann ein von Dritten entwickelter Block oder ein Block mit einem Nullwert (NULL) sein.

Die Datenprozessblöcke entwickeln die Logistik, Berechnung, Datennormalisierung, logische Entscheidungen und Ablaufprüfungen für jede Transaktion. Diese Arten von Blöcken werden verwendet, um dem Geschäftsprozess Intelligenz zu verleihen und basieren hauptsächlich, wie der Name schon sagt, auf der Verarbeitung aller Arten von Informationen sowie deren Konvertierung aus verschiedenen Datentypen.

Die Sicherheitsblöcke werden verwendet, um die Informationen zu validieren und sicherzustellen, dass die Transaktionen von ihrem Ursprung bis zu ihrem Ziel nicht verändert wurden. Sie werden immer mit verschiedenen Sicherheitsalgorithmen verarbeitet, die in Blockchain-Technologien verwendet werden, zu den am häufigsten verwendeten Tools

gehören (Hash-Signatur, digitale Signatur, AES-Datenverschlüsselung, Erstellung und Validierung digitaler Adressen, u.a.).

Die modularen Datenblöcke, das sind die Blöcke, die von Dritten entwickelt werden, erinnern daran, dass Mini BlocklyChain auf modulare Weise geschaffen wurde, um durch neue Blöcke entsprechend den Bedürfnissen jedes Sektors, ob öffentlich oder privat, angereichert zu werden. Weitere Informationen über die Erstellung von Modulen finden Sie im Anhang für Entwickler.

Wie wir bereits zuvor kommentiert die Architektur von Mini BlocklyChian in seiner zweiten Komponente sind die Prozesse der Kommunikation, diese Prozesse sind diejenigen, die für die Kommunikationskanäle über TCP und Sockets der Kommunikation, um die Flexibilität des Sendens und Empfangens von Transaktionen entweder durch die verschiedenen Mittel, die im Moment die am häufigsten verwendeten sind: Mobiles Internet, Wifi arbeitet derzeit an der Einbeziehung des Kommunikationsmediums Bluetooth.

Die Kommunikationsblöcke basieren im Wesentlichen auf dem Austausch von Daten mit der im Übertragungskanal implementierten Sicherheit und dieser basiert auf einer Kommunikation über das SSH (Secure Shell)-Protokoll mit den verschiedenen Stufen, die eine gesendete oder empfangene Transaktion durchläuft.

Der Kommunikationsteil ist von grundlegender Bedeutung, da diese Prozesse die Funktion haben, die Informationen in allen Knoten über TCP und die "**Peer to Peer**" genannte Verbindung **zu** aktualisieren. Die Blöcke, die in die Kommunikation eingreifen, basieren auf dem Informationsaustausch zwischen den Knoten ohne das Eingreifen zwischengeschalteter Server, so dass jeder Knoten unabhängig wird und in der Lage ist, ein Netz von Knoten zu schaffen, in dem die Fehlerpunkte minimal oder fast null sind. Ebenso hilft ihnen diese Unabhängigkeit der einzelnen Knotenpunkte, Entscheidungen einzeln oder als Ganzes entsprechend den Bedürfnissen des Unternehmens zu treffen.

Die "Peer-to-Peer"-Architektur besteht aus drei Teilen, die das öffentliche oder private Netzwerk bilden, das Sie erstellen möchten. In beiden Fällen wird der Kommunikationskanal von Knoten zu Knoten verschlüsselt.

Die erste Komponente für die Kommunikation zwischen mobilen Geräten (Smartphones) oder Wifi ist es, den Knoten oder Geräten ein Netzwerk zur Verfügung zu stellen, in dem sie überall auf der Welt zu finden sind, das Kommunikationsnetzwerk, in dem MiniBlocklyChain montiert ist, ist das "**Tor**"-Netzwerk.

Was ist das Tor-Netzwerk? - (<https://www.torproject.org>)

"**Tor** (Akronym für **Te Onion Router** - auf Spanisch) ist ein Projekt, dessen Hauptziel die Entwicklung eines verteilten Kommunikationsnetzes mit geringer Latenz ist, das dem Internet überlagert ist, in dem das Routing der zwischen den Benutzern ausgetauschten Nachrichten nicht ihre Identität, d.h.

ihrer IP-Adresse, offenbart (Anonymität auf Netzebene) und das außerdem die Integrität und die Geheimhaltung der Informationen, die es durchlaufen, wahrt".

Die zweite Komponente und nicht weniger wichtige Aufgabe ist es, alle Knoten in MiniBlocklyChain dazu zu bringen, jederzeit die gleichen Daten zu haben oder ihre Datenbanken und Dateien zu synchronisieren, um diese Aufgabe zwischen den Knoten durchzuführen, wird "**syncthing**" implementiert.

Was ist das Syncing-Netzwerk? - (<https://syncthing.net>)

Syncthing ist eine kostenlose Open-Source-Anwendung zur Peer-to-Peer-Dateisynchronisierung, die für Windows, Mac, Linux, Android, Solaris, Darwin und BSD verfügbar ist. Sie können Dateien zwischen Geräten in einem lokalen Netzwerk oder zwischen entfernten Geräten über das Internet synchronisieren.

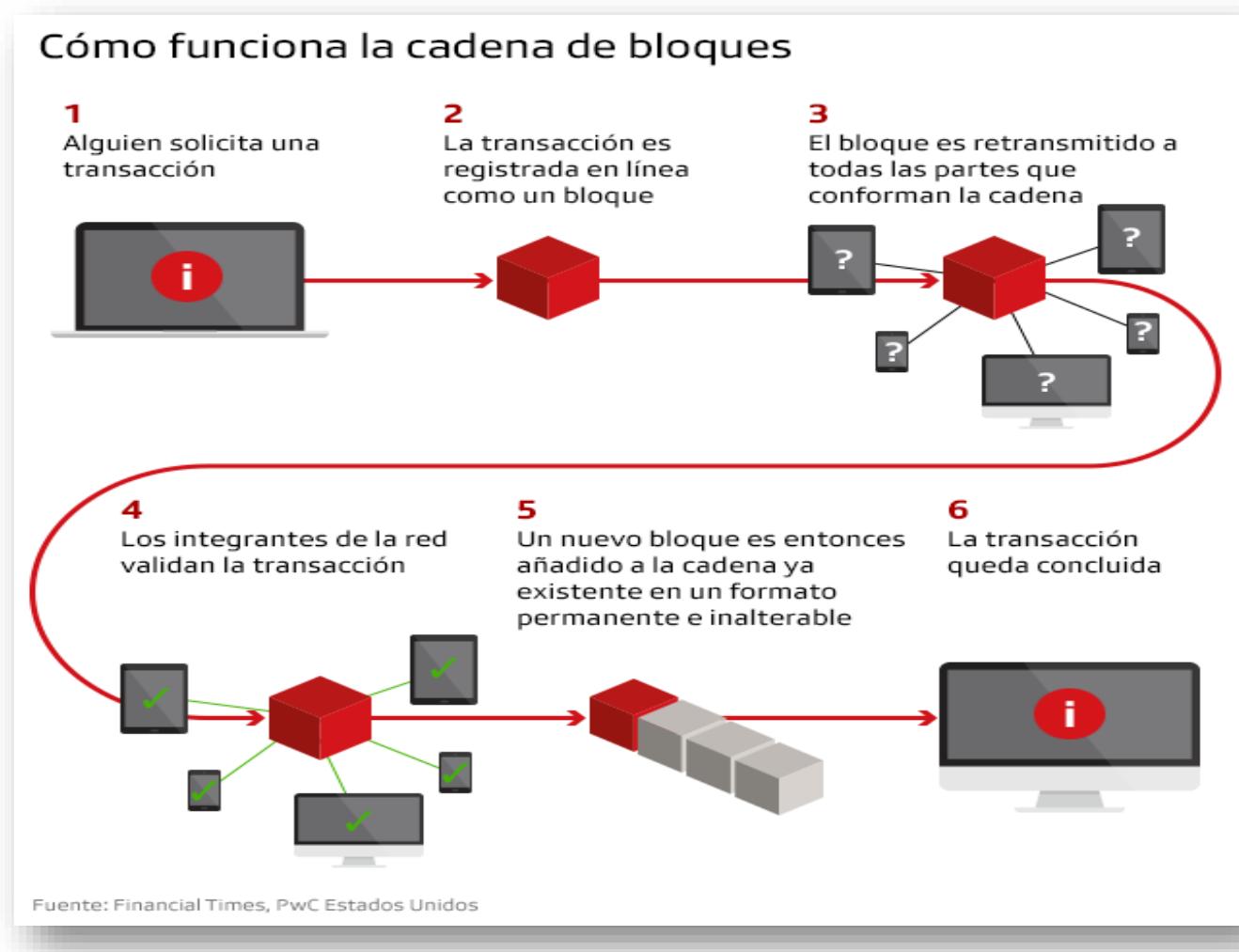
Auf der Grundlage der beiden vorhergehenden Kommunikationskomponenten können wir damit beginnen, ein Vertrauensnetzwerk zwischen den Knoten und mit dem Niveau der Datensynchronisation zu entwickeln, das das Rückgrat oder den "Kern" der Geschäftsprozesse bilden wird, um allen Anforderungen des Benutzers oder des Unternehmens gerecht zu werden.

Die dritte Komponente des Kommunikationsnetzes ist die Datenbank Redis, die alle Knoten in Echtzeit über neu eingegangene und gesendete Transaktionen informiert.

Was ist das Redis DB-Netz? - (<https://redis.io>)

**Redis** ist eine In-Memory-Datenbank-Engine, die auf der Speicherung in Hash-Tabellen (Schlüssel/Wert) basiert, die aber optional als dauerhafte oder persistente Datenbank verwendet werden kann.

## 7. BlocklyChain-Funktionalitätsdiagramm (Mini BlocklyChain).



## 8. Was ist der Mini-BlocklyCode?

Mini BlocklyCode sind Programme, die in der Java-Sprache erstellt und in einem von den Knoten unabhängigen Repository gespeichert werden, sie werden normalerweise als intelligente Verträge bezeichnet, diese Programme werden mit logischen Bedingungen entworfen, so dass sie, wenn diese Bedingungen erfüllt sind, automatisch ausgeführt werden, ohne von irgendeiner Autorisierung oder externen menschlichen Eingriffen abhängig zu sein.

"Ein intelligenter Vertrag ist ein Computerprogramm, das registrierte Vereinbarungen zwischen zwei oder mehr Parteien (z.B. Einzelpersonen oder Organisationen) erleichtert, sichert, durchsetzt und ausführt. Als solche würden sie sie bei der Aushandlung und Festlegung solcher Abkommen unterstützen, die bei Erfüllung einer Reihe spezifischer Bedingungen zu bestimmten Aktionen führen.

Ein intelligenter Vertrag ist ein Programm, das in einem System lebt, das weder von einer der beiden Parteien noch von ihren Agenten kontrolliert wird, und das einen automatischen Vertrag ausführt, der wie ein Wenn-Dann-Satz eines beliebigen anderen Computerprogramms funktioniert. Mit dem Unterschied, dass dies in einer Weise geschieht, die mit realen Vermögenswerten interagiert. Wenn ein vorprogrammierter Zustand, der keiner menschlichen Beurteilung unterliegt, ausgelöst wird, führt der intelligente Vertrag die entsprechende Vertragsklausel aus.

Sie zielen darauf ab, eine größere Sicherheit als das traditionelle Vertragsrecht zu bieten und die mit der Auftragsvergabe verbundenen Transaktionskosten zu verringern. Die Übertragung digitaler Werte durch ein System, das kein Vertrauen erfordert (z.B. Bitcoins), öffnet die Tür zu neuen Anwendungen, die intelligente Verträge nutzen können.“

Mini BlocklyCode richtet sich an Entwickler mit Erfahrung in der Java-Programmierung. In Mini BlocklyChain sind einige Bibliotheksarten aus Sicherheitsgründen für dasselbe System eingeschränkt, jedoch können Bibliotheken zur Erstellung von Transaktionen zum Senden oder Empfangen eines von zwei oder mehr Parteien geschaffenen oder vereinbarten vertraglichen Werts geschaffen werden.

Jede Mini-BlocklyChain erfüllt die folgenden Voraussetzungen:

- ✓ Jede erstellte Mini-BlocklyChain basiert darauf, dass nur Nachrichten und keine Ausführungen auf dem System ausgeführt werden. Diese Nachrichten können jedoch Berechtigungen, physische Vertragsgenehmigungen oder physische Aktionen enthalten.
- ✓ Jede erstellte Mini-BlocklyChain muss den Kommunikationsblock "Auditor" durchlaufen, der für die Überwachung von bösartigem Code oder verbotenem Code zur Überprüfung von Sätzen oder nicht autorisierten Befehlen im System zuständig ist.
- ✓ All Mini BlocklyChain wird nur auf der JVM des Quellknotens ausgeführt, Programme werden zum Schutz des Systems nicht global ausgeführt.

Weitere Einzelheiten finden Sie im Anhang "Mini BlocklyChain für Entwickler".

## 9. Installation eines Mini-BlocklyChain-Kommunikationsnetzes

### 1. Installation und Konfiguration des Mini-SQLSync-Netzwerks

Das Mini-SQLSync-Netzwerk ist dafür zuständig, die Transaktionen von den Knoten zu empfangen, damit sie diese Transaktionen verarbeiten können. Dieses Netzwerk besteht aus einem Hauptnetzwerk, das durch "Peer to Peer" zwischen den Knoten und einem Backup-Netzwerk namens Mini Sentinel RESTful gebildet wird.

Wir beginnen mit der Installation des RESTful Mini Sentinel Backup-Netzwerks, dieser Dienst ist derjenige, der die Transaktionen von den Knoten empfangen wird. Dieser Dienst basiert auf der Speicherung der Transaktionen für eine bestimmte Zeit, um später die Informationen über einen Konnektor, der eine Warteschlange aller verschlüsselten Transaktionen in einen Redis-Dienst injiziert, umzuwandeln. Später wird der Redis-Datenbankdienst in Echtzeit die Freigabe der verschlüsselten Transaktionswarteschlange an alle Knoten, die das Mini BlocklyChain-Netzwerk integrieren, ausführen.

Eine Dienstleistung oder ein Design vom Typ RESTful ist eine einfache und leichte Möglichkeit, über eine URL oder Internetadresse Informationen in einer Datenbank abzufragen, zu ändern, zu löschen und/oder in eine Datenbank einzufügen. In unserem Fall verwenden wir die SQLite-Datenbank, da sie die Eigenschaft hat, alle Informationen in einer Datei zu enthalten. Für eine Verwendung von Anforderungen mit Bedürfnissen der großen Skalierung werden wir in der Lage sein, eine andere Art von Datenbank wie MySQL, Oracle, DB2 oder eine andere kommerzielle Datenbank zu verwenden, wir müssen einfach nur den Konnektor von Mini BlocklyChain von SQLite (freie Version) für den Konnektor von Mini BlocklyChain DB andere (kommerzielle Version) ändern.

**WICHTIGER HINWEIS:** Die RESTful Mini Sentinel-Konfiguration verarbeitet zu keiner Zeit irgendeine Art von Transaktionen, sondern nur der Dienst, der die Informationen "formt", so dass die Knoten die Transaktionsverarbeitung ordnungsgemäß und in einer geplanten und synchronisierten Zeit für alle Knoten durchführen können.

Die RESTful-Service-Installation basiert auf einer SQLite-Datenbank. Aus praktischen Gründen wird diese Installation in einer Windows-Umgebung durchgeführt, jedoch kann dieses Modell leicht in einem Betriebssystem vom Typ Linux repliziert werden.

Für diejenigen, die die Leistung und Unterstützung von SQLite-Abfragen und -Einfügungen überprüfen möchten, verweisen wir auf diese Leistungstests:

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

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

RESTful Mini Sentinel Backup-Netzwerkinstallation. Wir werden diesen Dienst auf einem Windows 10-Computer installieren und konfigurieren, der Prozess wurde jedoch auch in einer Ubuntu 18.09 Linux-Umgebung problemlos installiert.

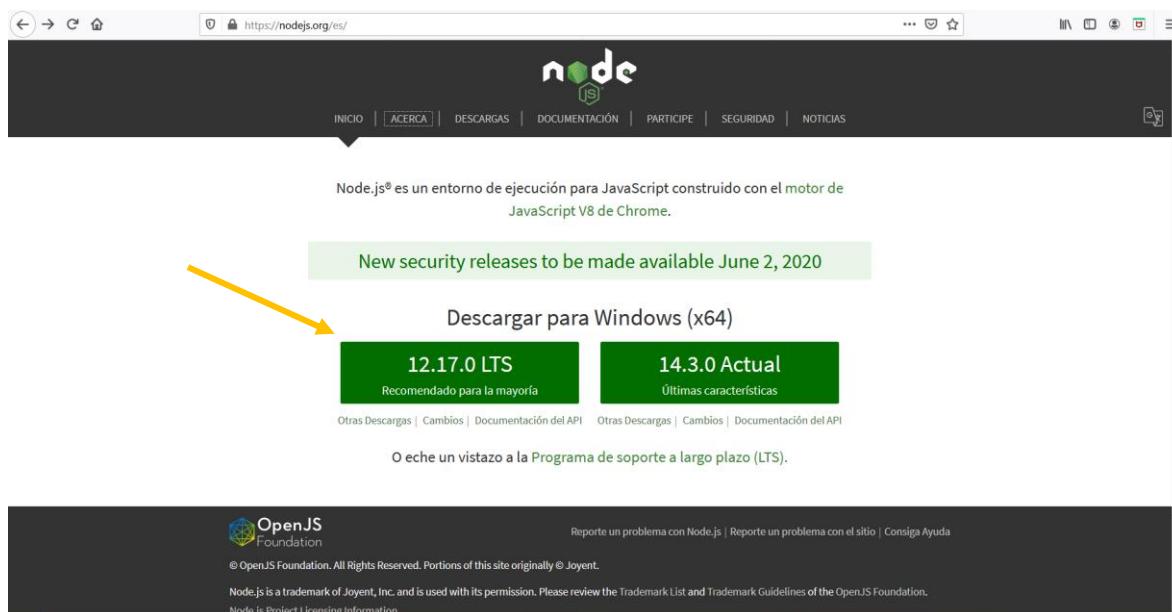
Anforderungen:

- ✓ SQLite-Datenbankserver im RESTful-Modus (<https://github.com/olsonpm/sqlite-to-rest>).
- ✓ SQLite-Redis Sentinel-Verbinder
- ✓ Redis-Datenbankserver im Master-Slave-Modus (<https://redis.io/topics/replication>)

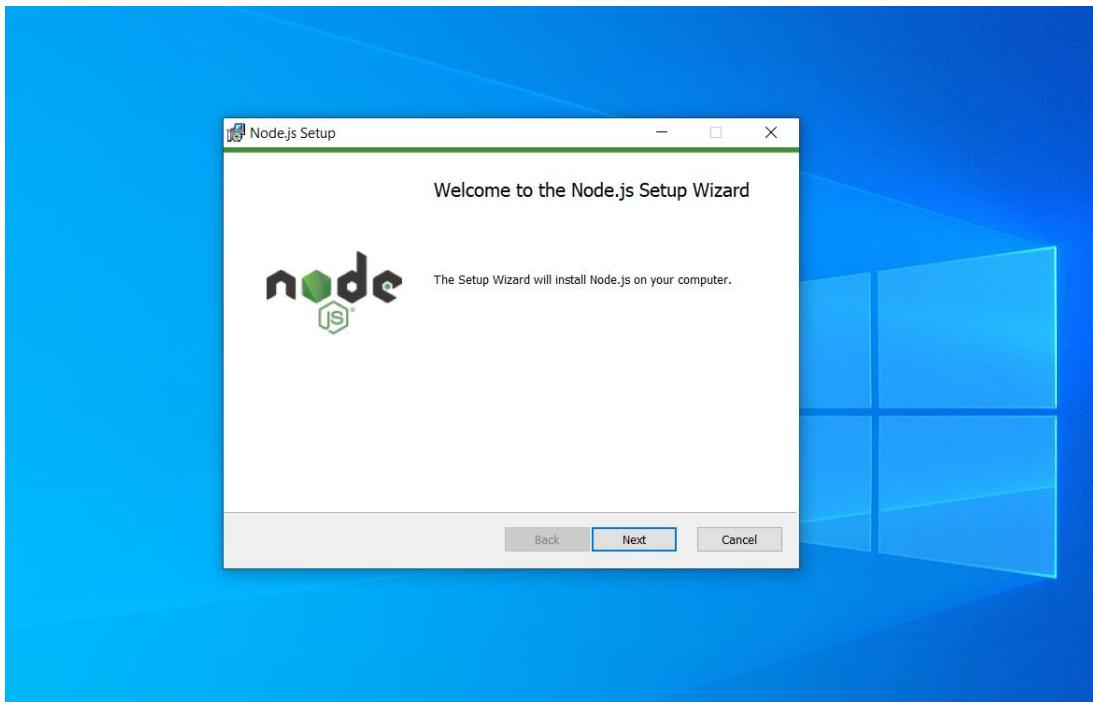
Installation des SQLite-Datenbankservers im RESTful-Modus

Für die Installation müssen wir mit den folgenden Softwarepaketen beginnen: Nodejs, sqlite3 und Git Bash für Windows.

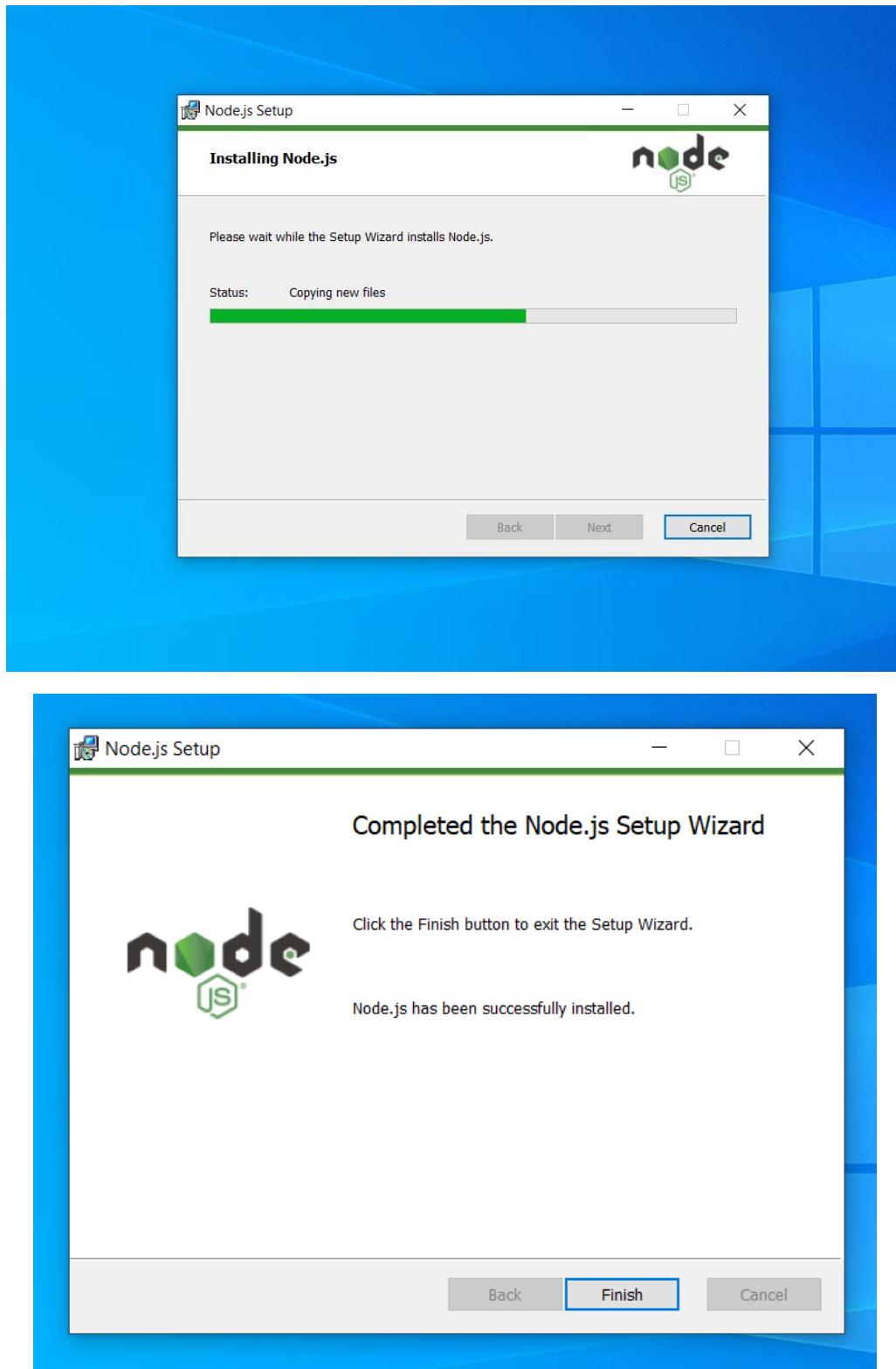
Um Nodejs zu erhalten, konsultierten wir ihre offizielle Website <https://nodejs.org/es/> und wählten die Version "Empfohlen für die meisten".



Nach dem Herunterladen der Datei mit der Erweiterung .msi doppelklicken wir auf die Datei, um sie zu installieren.



Wir führen die Installation standardmäßig durch, klicken Sie einfach auf "Weiter", ohne weitere von Ihnen gewünschte Optionen auszuwählen.



Da wir die Installation von Nodejs abgeschlossen haben, fahren wir mit der Installation der SQLite-Datenbank fort.

Wir gehen auf ihre offizielle Website <https://www.sqlite.org/index.html> und klicken auf den Teil, den Sie "herunterladen".



### What Is SQLite?

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured database engine in the world. SQLite is built into all mobile phones and most computers and is used by millions of people every day. [More Information...](#)

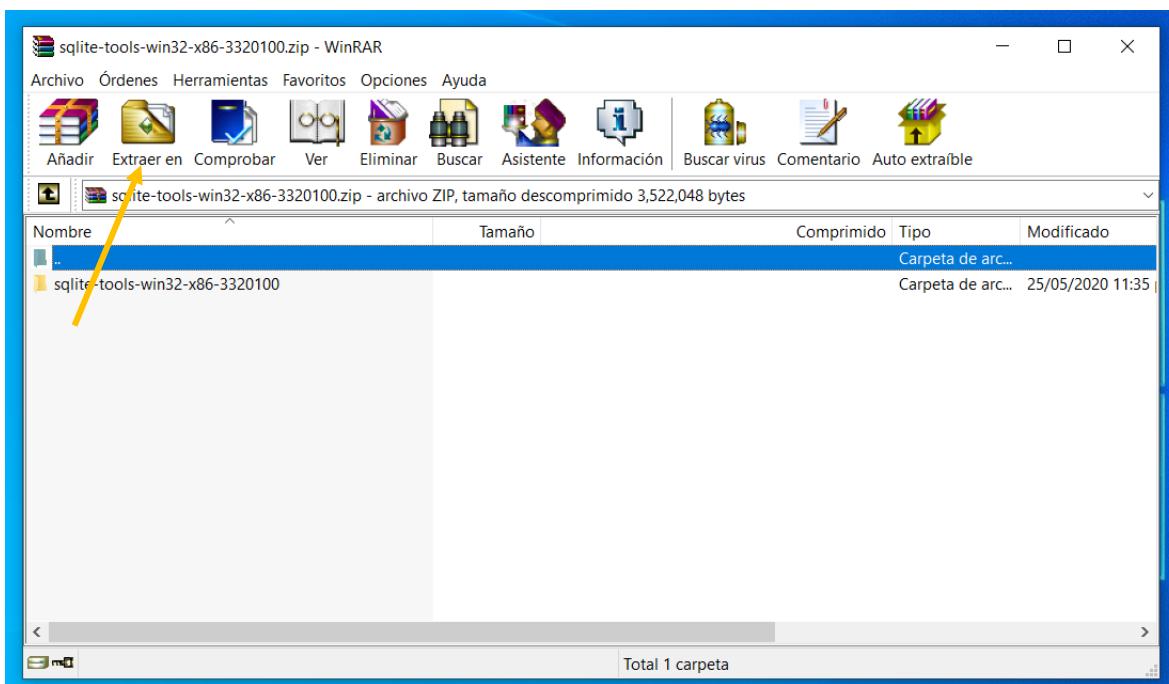
The SQLite [file format](#) is stable, cross-platform, and backwards compatible and the developers pledged to support it until 2050. SQLite database files are commonly used as containers to transfer rich content between systems. The file format is designed for data [[4](#)]. There are over 1 trillion (1e12) SQLite databases in active use [[5](#)].

SQLite [source code](#) is in the [public-domain](#) and is free to everyone to use for any purpose.

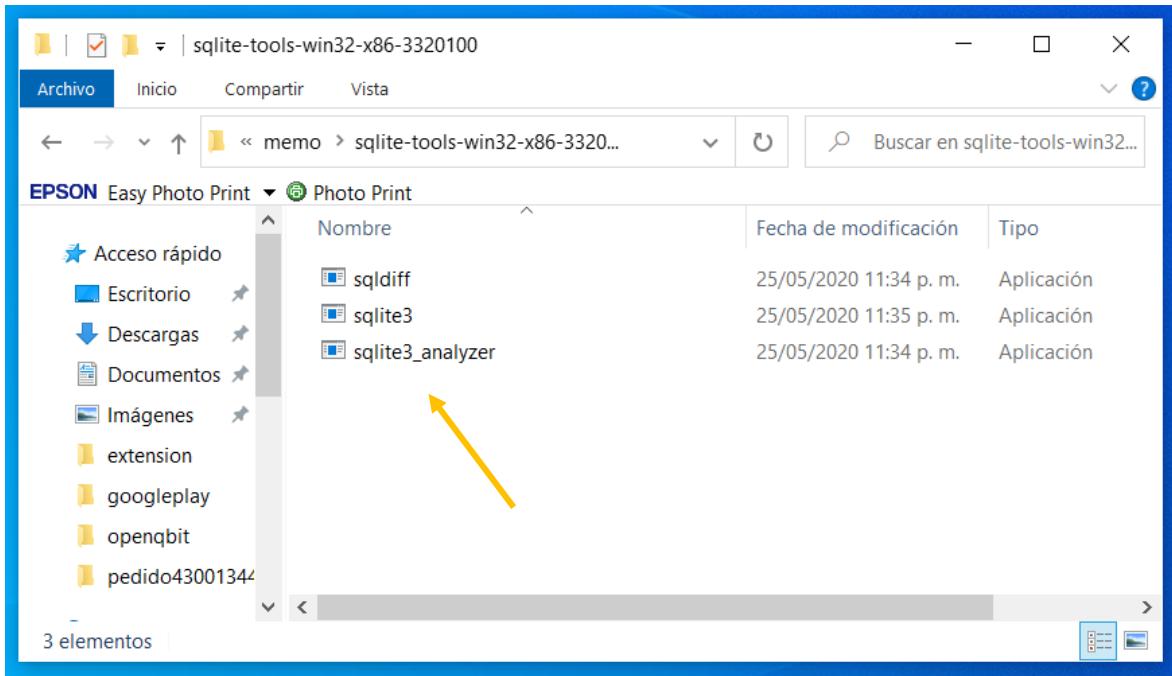
### Latest Release

[Version 3.32.1](#) (2020-05-25). [Download](#) [Prior Releases](#)

Als nächstes wählen wir die Option "Vorkompliierte Binärdateien für Windows" und klicken auf die Software-Version "sqlite-tools-win32-x86-3320100.zip". Wenn der Download abgeschlossen ist, dekomprimieren wir die Datei auf einfache Weise, öffnen den Ordner, in den die Datei heruntergeladen wurde, und doppelklicken auf die Datei, um sie zu dekomprimieren.

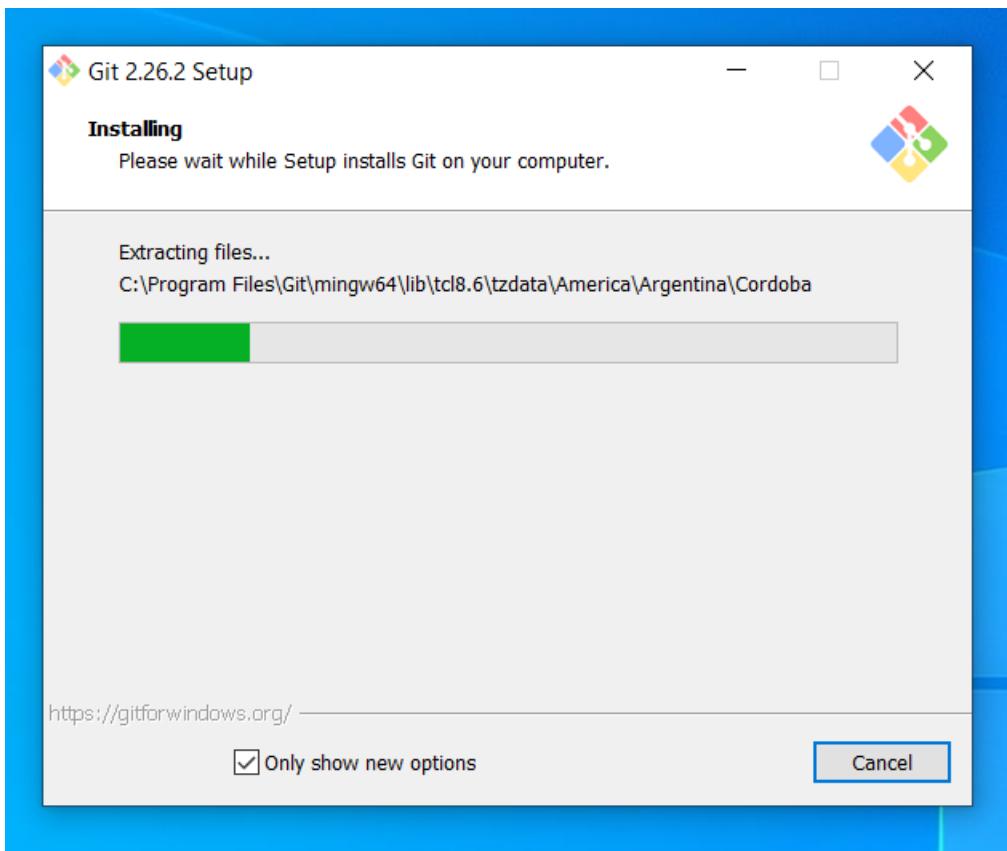


Nach der Dekomprimierung werden wir drei Dateien haben, das ist unsere Umgebung zur Erstellung unserer SQLite-Datenbank, die wir später verwenden werden.



Wir beginnen mit der Installation von Git Bash, einem Linux-ähnlichen Terminal-Emulator, der uns dabei helfen wird, den RESTful-Backup-Dienst ordnungsgemäß auszuführen.

Wir werden sie von ihrer offiziellen Website <https://gitforwindows.org/> herunterladen und sie mit den angegebenen Standardoptionen installieren.



Nun, da wir nodejs, sqlite und Git Bash auf unserem Windows 10-Rechner installiert haben, fahren wir mit der Installation der Umgebung fort, die unsere SQLite-Datenbank im RESTful-Modus unterstützt.

Zum Zeitpunkt des Herunterladens der Software, die die Funktionalität von RESTful an SQLite abgibt. Dazu müssen wir auf die folgende Website gehen, <https://github.com/olsonpm/sqlite-to-rest>. Dort klicken wir auf die grüne rechte Schaltfläche "Klonen oder herunterladen" und wählen die Option "Download ZIP", wodurch die Software auf unseren Computer heruntergeladen wird.

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-r>

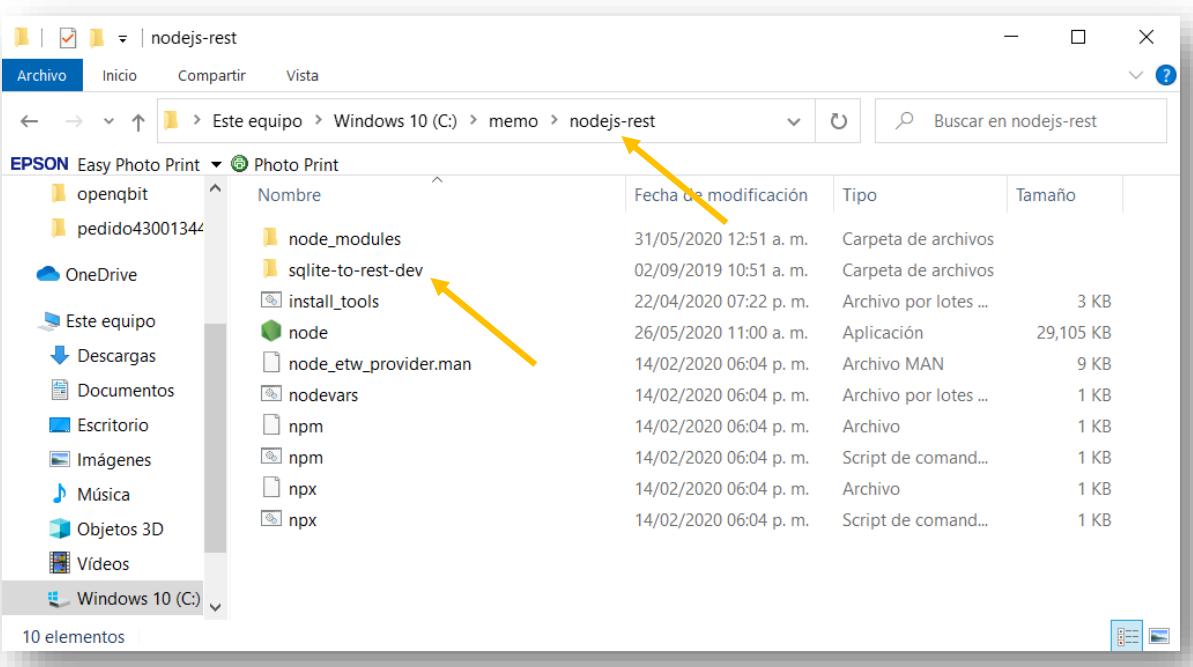
Open in Desktop Download ZIP

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

Nachdem wir es auf unseren Computer heruntergeladen haben, dekomprimieren wir es in dem Verzeichnis oder Ordner, in dem wir das **nodejs-Programm** installiert haben, und wir werden ein Verzeichnis mit dem Namen "**sqlite-to-rest-dev**" haben.

**HINWEIS:** Es ist wichtig, dass sich das Verzeichnis **sqlite-to-rest-dev** innerhalb des Verzeichnisses befindet, in dem **nodejs** installiert wurde.

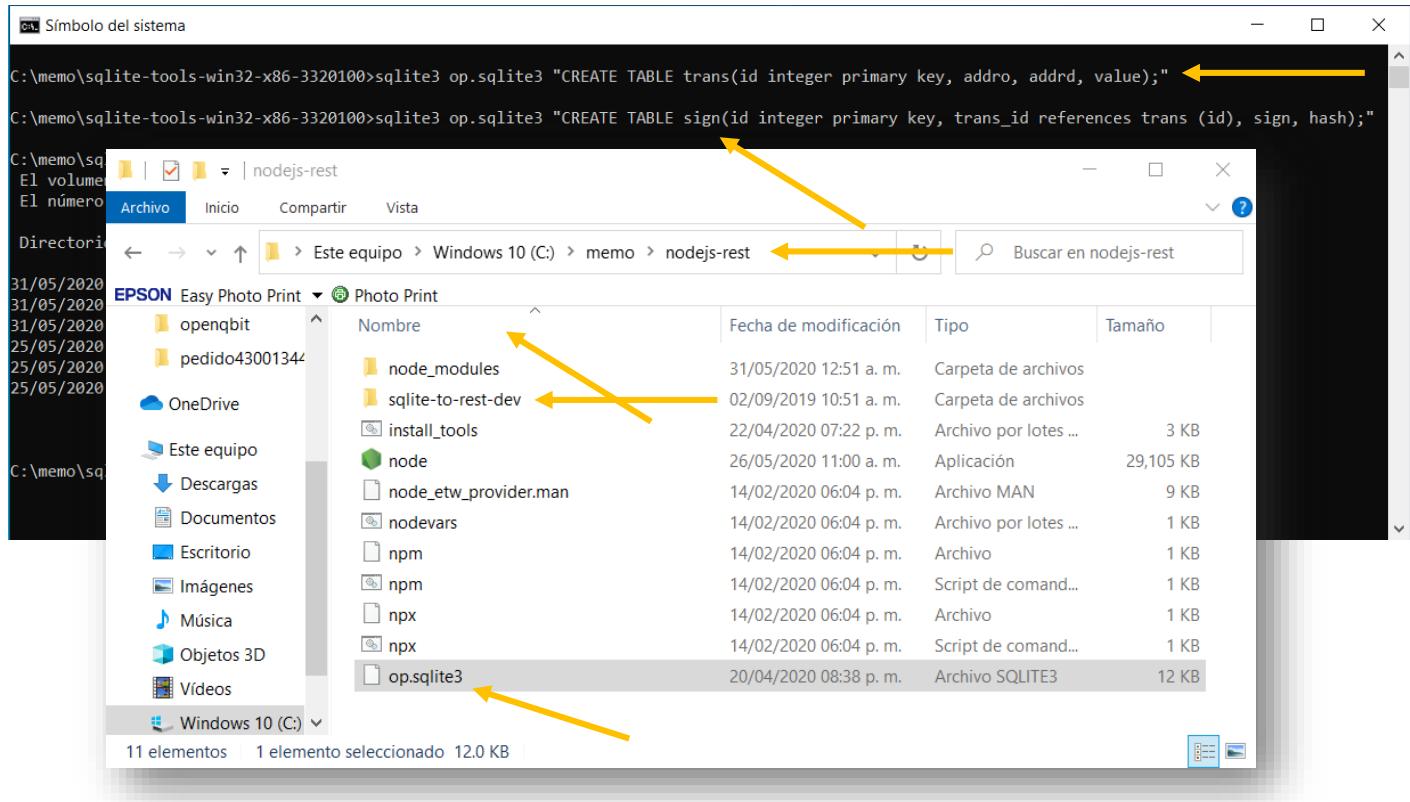


Nachdem alles installiert ist, fahren wir mit der Konfiguration der SQLite-Datenbank fort, die wir für die Speicherung der Transaktionen der Knoten in der RESTful-Backup-Umgebung verwenden werden.

Tabellendesign und Datenstruktur. Befehle, die online in der CMD-Befehlsliste ausgeführt werden

```
sqlite3 op.sqlite3 "CREATE TABLE trans (id integer primary key, addro, addrd, value);";;"
```

```
sqlite3 op.sqlite3 "TABELLE-Zeichen ERZEUGEN (id ganzzahliger Primärschlüssel, trans_id verweist auf trans (id), Zeichen, Hash);";
```

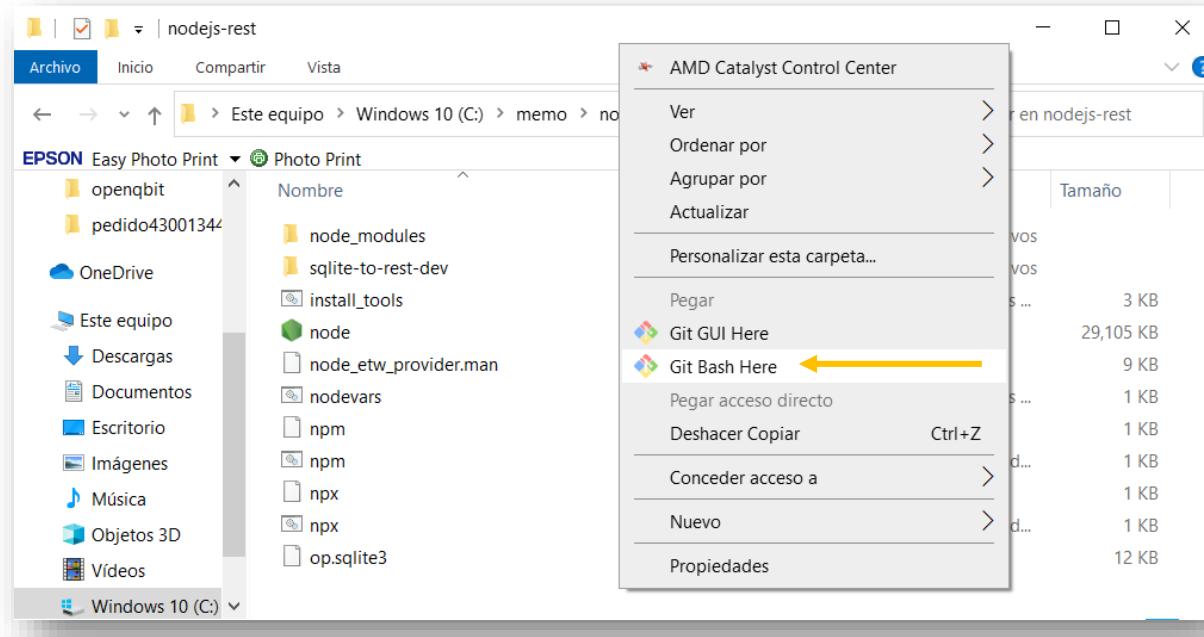


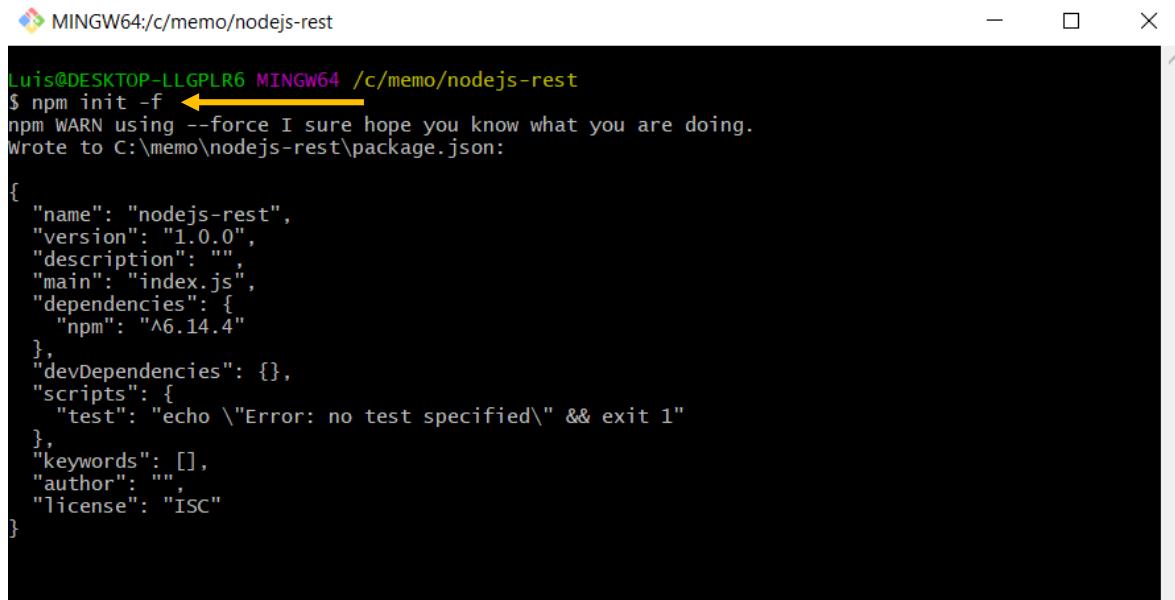
Nach der Erstellung der Datenbank op.sqlite3 müssen wir eine Kopie in dem Verzeichnis erstellen, in dem nodejs installiert wurde. Im **nodejs-Verzeichnis** muss sich auch die Kopie von "**sqlite-to-rest-dev**" befinden.

Wir platzieren uns in dem Ordner, in dem sich die **nodejs-Installation** befindet und in dem sich auch die "**sqlite-to-rest-dev**"-Software befinden sollte. Zeigen Sie mit dem Zeiger auf den Ordner und klicken Sie mit der rechten Maustaste, um das Menü anzuzeigen, und wählen Sie dort, wo "**Git Bash**" steht, um ein Terminal zu öffnen.

Im neuen offenen Git Bash-Terminal führen wir die folgenden Befehle aus:

\$ npm init -f

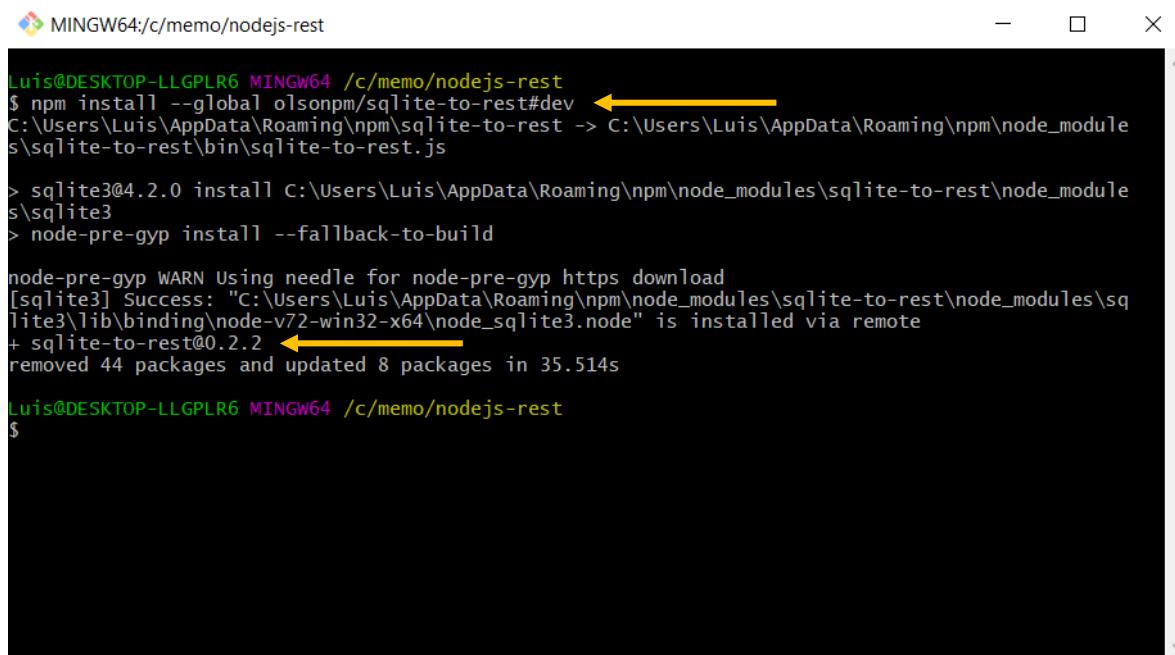




```
Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$ npm init -f
npm WARN using --force I sure hope you know what you are doing.
Wrote to C:\memo\nodejs-rest\package.json:

{
  "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
$
```

Nach der Ausführung des Befehls erscheint die Installation des "**sqlite-to-rest**"-Pakets.

Wir haben eine RESTful-Umgebung für SQLite mit der zuvor erstellten **op.sqlite3-Basis** generiert.

Wir führen den Befehl aus: `$ sqlite-to-rest generate-skeleton --db-path ./op.sqlite3`



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

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

Wir haben den RESTful SQLite-Dienst auf dem Standardport 8085 gestartet. Wir führen den folgenden Befehl aus: `$ node skeleton.js`



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

Wir haben den RESTful-Dienst mit dem Hinzufügen von Daten und der Abfrage von Daten



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s -H "Content-Type: application/json" -d '{"id":6,"addr": "QWERTY1234","addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/transactions ←
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s http://localhost:8085/transactions ←

"id":1,"addr": "CO","addrd": "Boulder", "value": "Avery"}
"id":2,"addr": "WI", "addrd": "New Glarus", "value": "New Glarus"}
"id":3,"addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id":4,"addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id":5,"addr": "WI", "addrd": "Madison", "value": "One Barrel"}
"id":6,"addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}
```

getestet.

Um den SQLite RESTful-Dienst zu testen, verwenden wir die folgenden Befehle:

Zum Einfügen von Daten in die Tabelle trans, die sich innerhalb der Datenbank op.sqlite3 befindet:

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

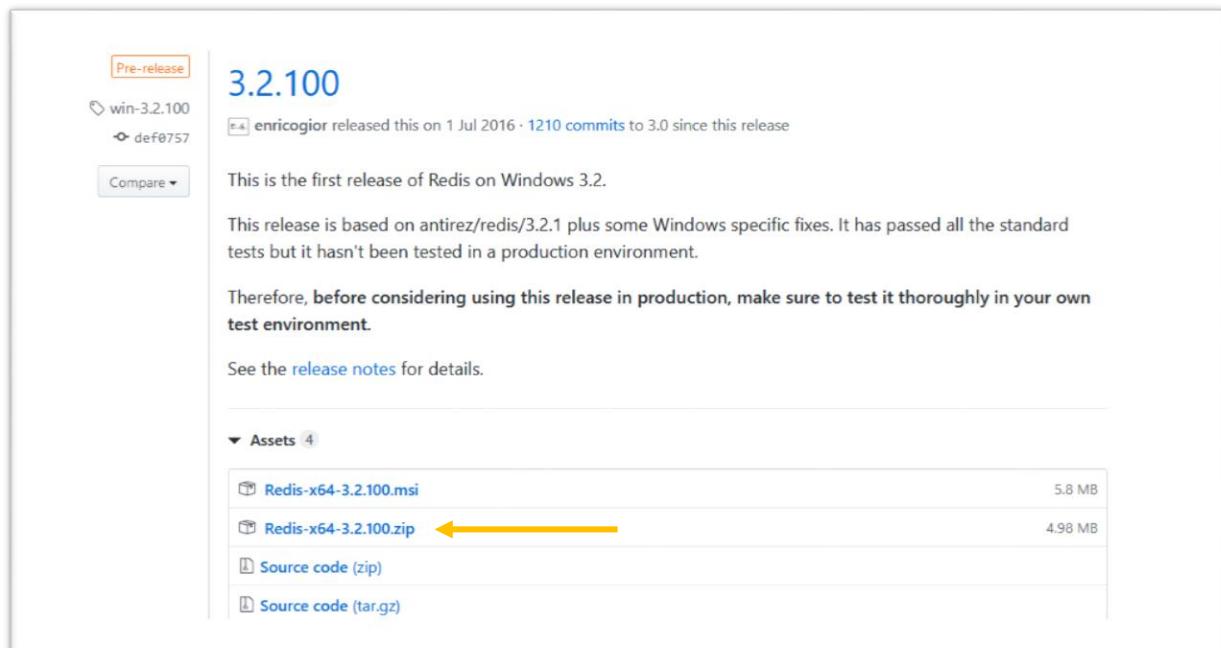
Um eine Abfrage aller Daten in der Trans-Tabelle durchzuführen:

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

Um im Detail zu erfahren, wie Sie alle RESTful-aktivierten Dienste (Abfragen, Einfügen, Aktualisieren und/oder Löschen von Daten) nutzen können, lesen Sie den **Anhang "Restful SQLite GET/POST-Befehle"**.

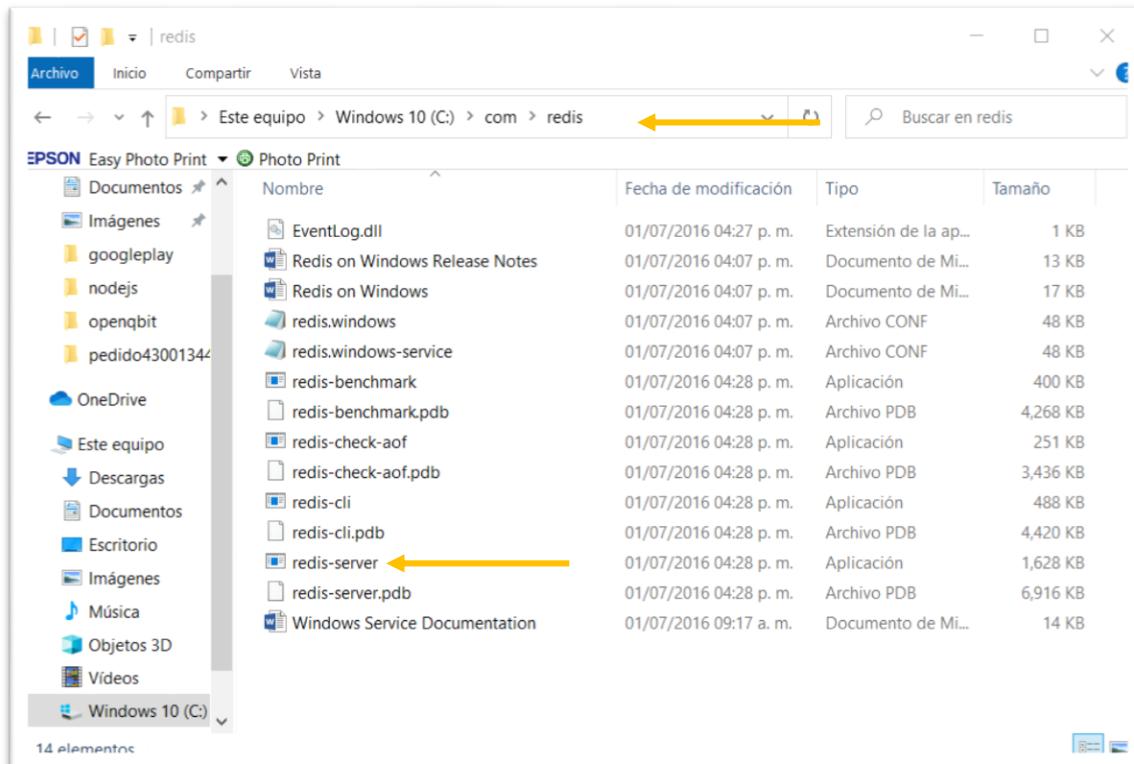
**ANMERKUNG:** Bei der vorherigen Installation werden alle Abfragen in der URL mit der Adresse "localhost" gemacht, aber wenn der Server mit einer öffentlichen IP (Internet) oder privaten IP (Wifi) exponiert ist, wird er ohne Probleme funktionieren. Wir werden dies testen, wenn wir die Kommunikationstests zwischen dem SQLite RESTful-Dienst und den Kommunikationsnetzknoten, die Mini BlocklyChain bilden, durchführen.

Redis-Datenbankinstallation für den Backup-Netzwerkdienst, um die Redis-Software für Windows herunterzuladen, müssen wir auf die Website <https://github.com/microsoftarchive/redis/releases/tag/win-3.2.100> gehen und das ZIP-Paket wählen.



Um die Installation zu finden, werden wir ein Verzeichnis namens "redis" innerhalb von Windows erstellen und die Datei mit einer ZIP-Erweiterung herunterladen, wir dekomprimieren sie in dem zuvor erstellten Verzeichnis, wir haben die Installation bereits abgeschlossen redis.

Wir testen die Installation, indem wir den Server durch Doppelklick auf den Befehl "**redis-server**" starten.



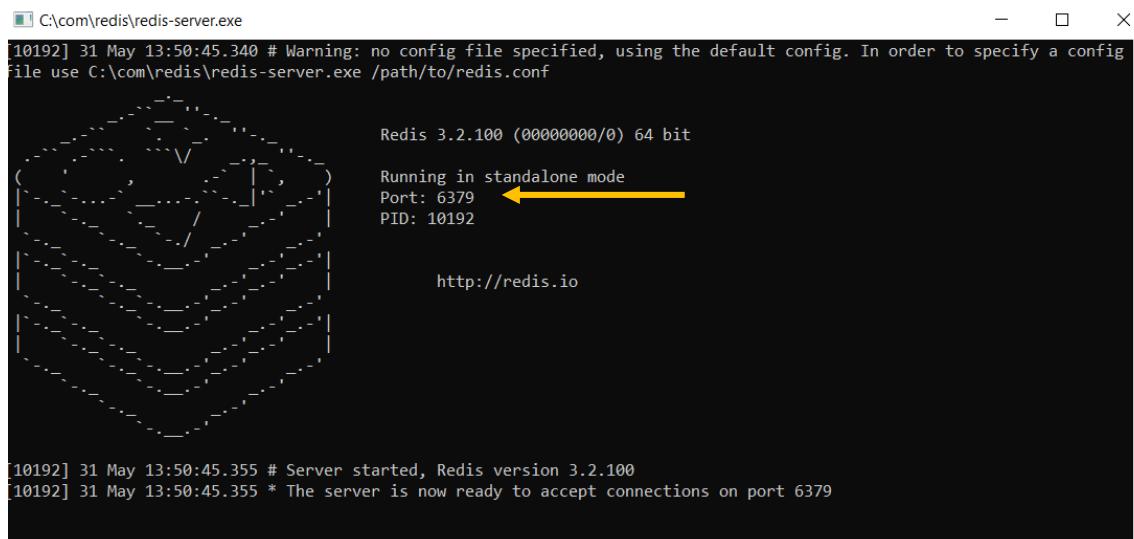
Nach der Ausführung des Befehls "redis-server" sehen wir den Server in einem Windows-CMD-Befehlsterminal auf dem Standard-Port 6379 laufen:

In diesem Test haben wir eine Version von Redis 3.2.100 für Windows 10, im Falle des Linux-Betriebssystems haben wir die Version 6.0.4 (veröffentlicht im Mai 2020), aber für unsere Mini BlocklyChain-Konfiguration hat die Windows-Version genügend Funktionen, die wir benötigen.

Um die Ausführung zu stoppen, machen wir die Tastenkombination Strg + C. Wir fahren fort, die Datenbank Redis 3.2.100 für Windows 10 mit einer Konfiguration zwischen Redis und den Mini-SQLSync-Kommunikationsnetzknoten zu konfigurieren, der Typ **Master(Master)-Slave(Slave)** ist wie folgt verteilt:

Der Master ist der von uns zu konfigurierende Redis-Server für Windows 10, der die Daten in Echtzeit und synchronisiert mit den gleichen Informationen an alle Knoten (Mobiltelefone) repliziert. Auf diesen Knoten wird ein Redis-Server im **Slave-Modus** installiert sein.

An diesem Punkt beginnen wir zu sehen, wie das Backup-Netzwerk, das wir installieren und konfigurieren, funktioniert. Diese Konfiguration wird uns dazu dienen, mit allen Knoten in Echtzeit zu kommunizieren, sie wird uns helfen, mit allen Knoten des Netzes zu kommunizieren, wenn es eine neue Transaktionswarteschlange gibt, die von den Knoten zu verarbeiten ist, in einer Gleichheit bei der Informationsübertragung an alle Knoten, so dass jeder Knoten die gleiche Wahrscheinlichkeit hat, die Informationen der



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

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

http://redis.io

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

"Transaktionswarteschlange" verarbeiten zu können.

Wir beginnen mit der Konfiguration des **SQLite-Redis Sentinel-Konnektors**.

Dieser Konnektor ist ein in der Sprache Java entwickeltes Programm und verbindet, wie der Name schon sagt, die Datenbanken SQLite und Redis (**Master**).

Die Funktion des Konnektors besteht darin, die Informationen (Transaktionen) von SQLite an Redis (**Master**) zu übertragen, und dies sendet die "Transaktionswarteschlange" an die Knoten (**Slaves**).

Weitere Einzelheiten über den Java-Code des **SQLite-Redis Sentinel** Connectors finden Sie im Anhang "SQLite-Redis Java Code Connector".

Konfiguration der Redis (**Master**)-Datenbank **redis.conf-Datei** für Windows 10.

Fügen Sie die folgenden Änderungen oder Direktiven in die Datei ein, speichern Sie die Änderungen und starten Sie den Redis-Server

Beginnen Sie mit der Suche nach der Einstellung **tcp-keepalive** und setzen Sie sie auf 60 Sekunden, wie in den Kommentaren vorgeschlagen. Dies wird Redis helfen, Netzwerk- oder Serviceprobleme zu erkennen:

**tcp-keepalive 60**

Finden Sie die **requirepass-Direktive** und konfigurieren Sie sie mit einer starken Passphrase. Während Ihr Redis-Verkehr vor Dritten geschützt werden muss, dient dies der Authentifizierung gegenüber Redis. Da Redis schnell ist und grenzwertige Passphrasenversuche nicht bewertet, wählen Sie eine starke, komplexe Passphrase, um sich gegen Versuche mit roher Gewalt zu schützen:

**requirepass type\_your\_network\_master\_password**

Beispiel:

**requirepass FPqwedsLMdf76ass7asddf2g45vBN8ty99**

Schließlich gibt es noch einige optionale Einstellungen, die Sie je nach Ihrem Anwendungsszenario anpassen können.

Wenn Sie nicht möchten, dass Redis automatisch die ältesten und am wenigsten benutzten Schlüssel einfügt, während es sich füllt, können Sie die automatische Schlüsselentfernung deaktivieren:

**maxmemory-policy noevasion**

Für verbesserte Haltbarkeitsgarantien können Sie die Dateipersistenz als Add-on-only aktivieren. Dies trägt dazu bei, den Datenverlust im Falle eines Systemausfalls auf Kosten größerer Dateien und etwas langsamerer Leistung zu minimieren:

**beifügen ja**

anhängen-dateiname "redis-staging-ao.aof"

Fahren Sie fort, um die Änderungen zu speichern und den Redis-Dienst für Windows 10 neu zu starten, stoppen Sie mit den Tasten Strg + C und führen Sie die Windows CMD-Befehlszeile erneut aus:

C:\redis\_redis\_Verzeichnis redis\_server

In unserem Beispiel können wir sehen, dass wir einen (**Slave-**)Knoten angeschlossen haben.

```

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

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

http://redis.io

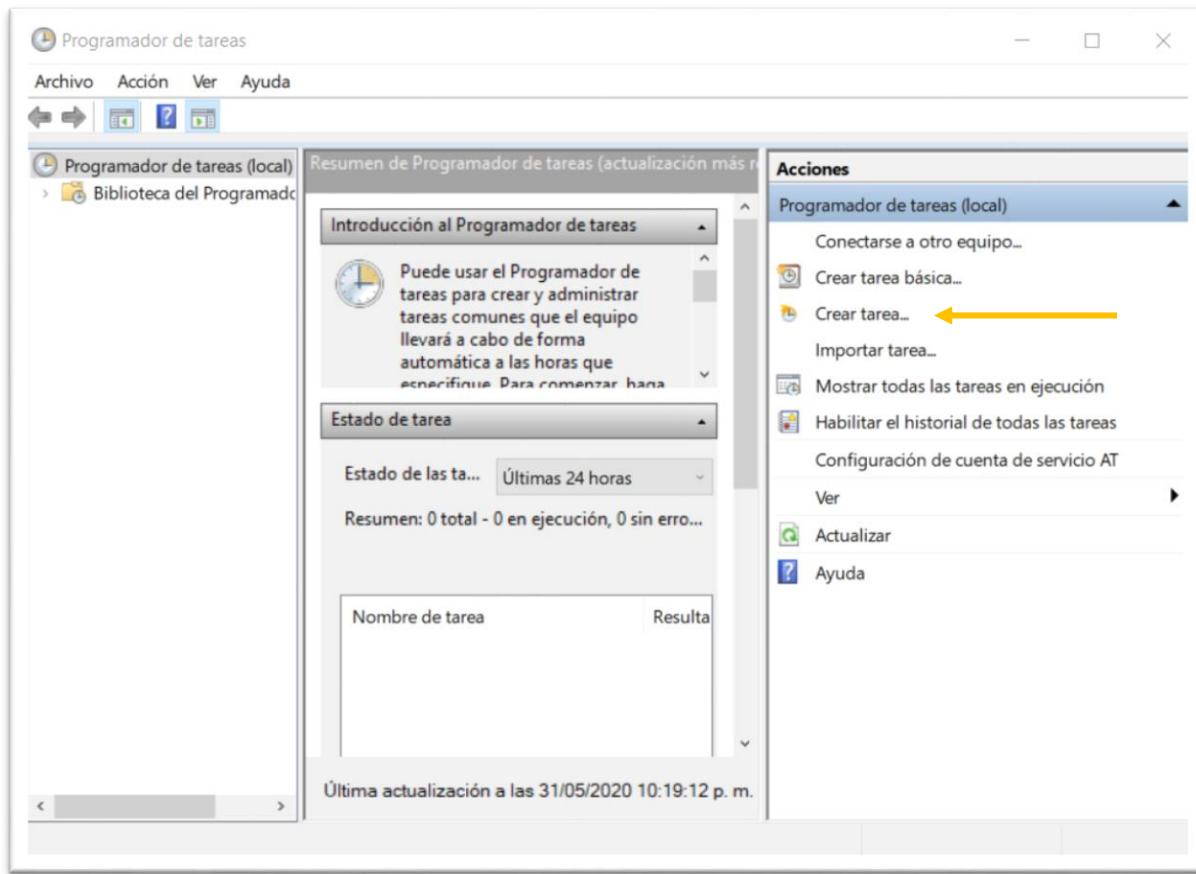
1:M 31 May 2020 23:44:56.648 # WARNING: The TCP backlog setting of 511 cannot be enforced
because /proc/sys/net/core/somaxconn is set to the lower value of 128.51:May 31 23:4
1:M 31 May 2020 23:44:56.648 # Server initialized
1:M 31 May 2020 23:44:56.649 # WARNING overcommit_memory is set to 0! Background save may
fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/s
ysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to t
ake effect.
1:M 31 May 2020 23:44:56.669 * DB loaded from append only file: 0.020 seconds
1:M 31 May 2020 23:44:56.669 * Ready to accept connections
1:M 31 May 2020 23:49:57.013 * 10 changes in 300 seconds. Saving...
1:M 31 May 2020 23:49:57.031 * Background saving started by pid 82
2:C 31 May 2020 23:49:57.052 * DB saved on disk
1:M 31 May 2020 23:49:57.133 * Background saving terminated with success
1:M 31 May 2020 23:50:24.600 * Replica 192.168.1.68:6379 asks for synchronization
1:M 31 May 2020 23:50:24.602 * Full resync requested by replica 192.168.1.68:6379
1:M 31 May 2020 23:50:24.602 * Starting BGSAVE for SYNC with target: disk
1:M 31 May 2020 23:50:24.619 * Background saving started by pid 83
3:C 31 May 2020 23:50:24.642 * DB saved on disk
1:M 31 May 2020 23:50:24.670 * Background saving terminated with success
1:M 31 May 2020 23:50:24.689 * Synchronization with replica 192.168.1.68:6379 succeeded

```

Wir können sehen, dass wir einen Knoten mit der IP 192.168.1.68 im Standard-Port 6379 synchronisiert haben.

Jetzt müssen wir im Betriebssystem Windows 10 die Aufgabe einplanen, den **SQLite-Redis Sentinel-Konnektor** automatisch auszuführen. Wir tun dies mit dem Windows 10-Tool, indem wir es von unten links durch Eingabe von "Task Scheduler" ausführen.

Wir werden eine neue Aufgabe "Aufgabe erstellen" erstellen, in der wir die Ausführung des



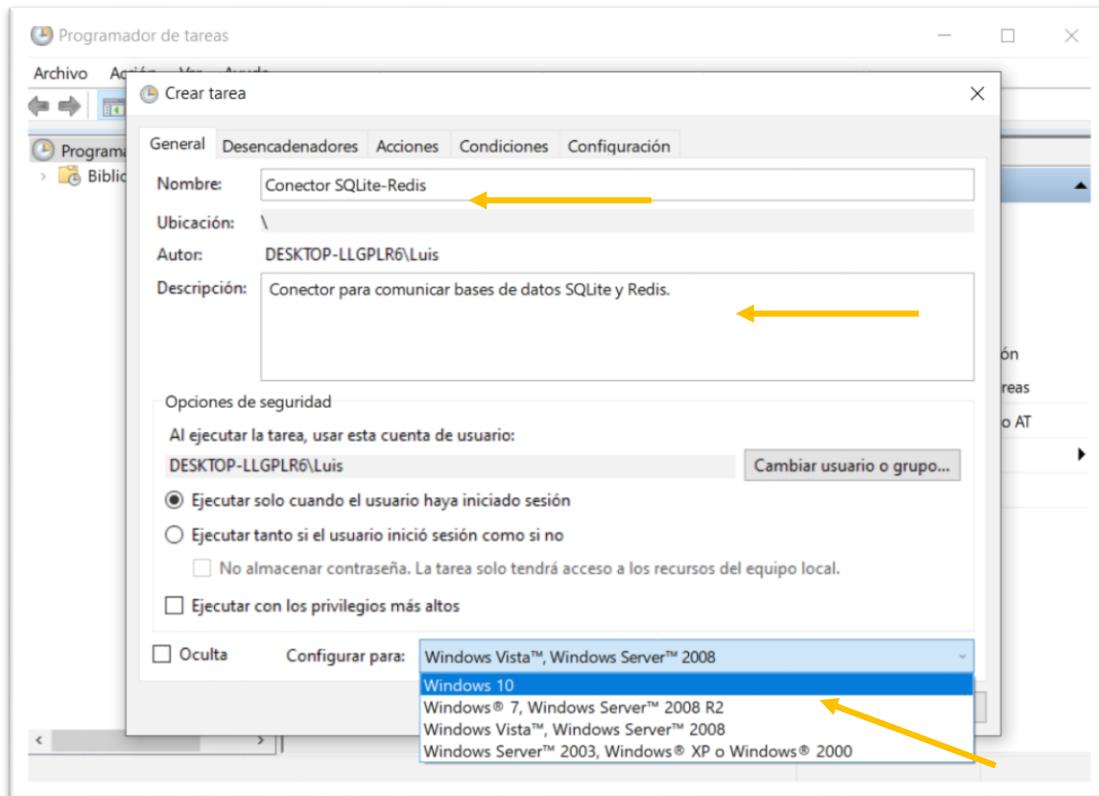
Konnektors aufnehmen werden.

Wenn Sie auf "Aufgabe erstellen" klicken, öffnet sich ein zusätzliches Fenster, in dem wir in der Registerkarte "Allgemein" die folgenden Parameter angeben müssen:

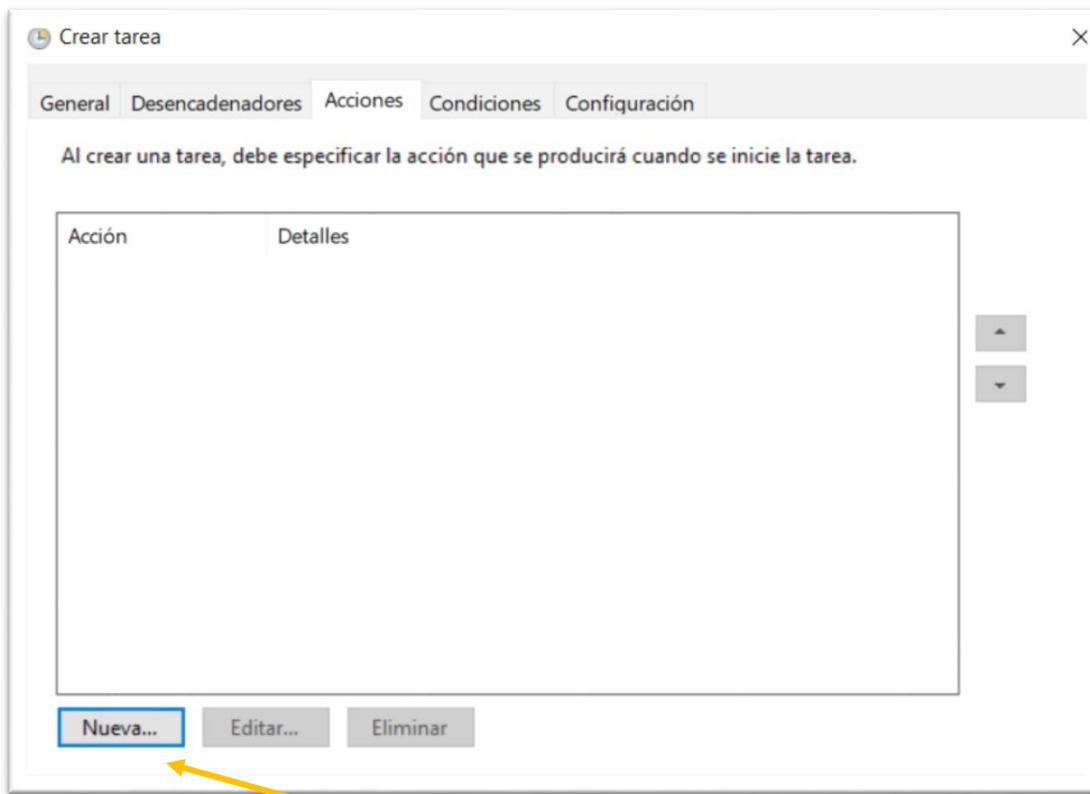
**Name:** Aufgabe\_Name

**Beschreibung:** optional

**Konfigurieren für:** select\_operating\_system\_windows\_version



Ändern Sie, indem Sie auf die Registerkarte "Aktionen" klicken und auf die Schaltfläche "Neu"

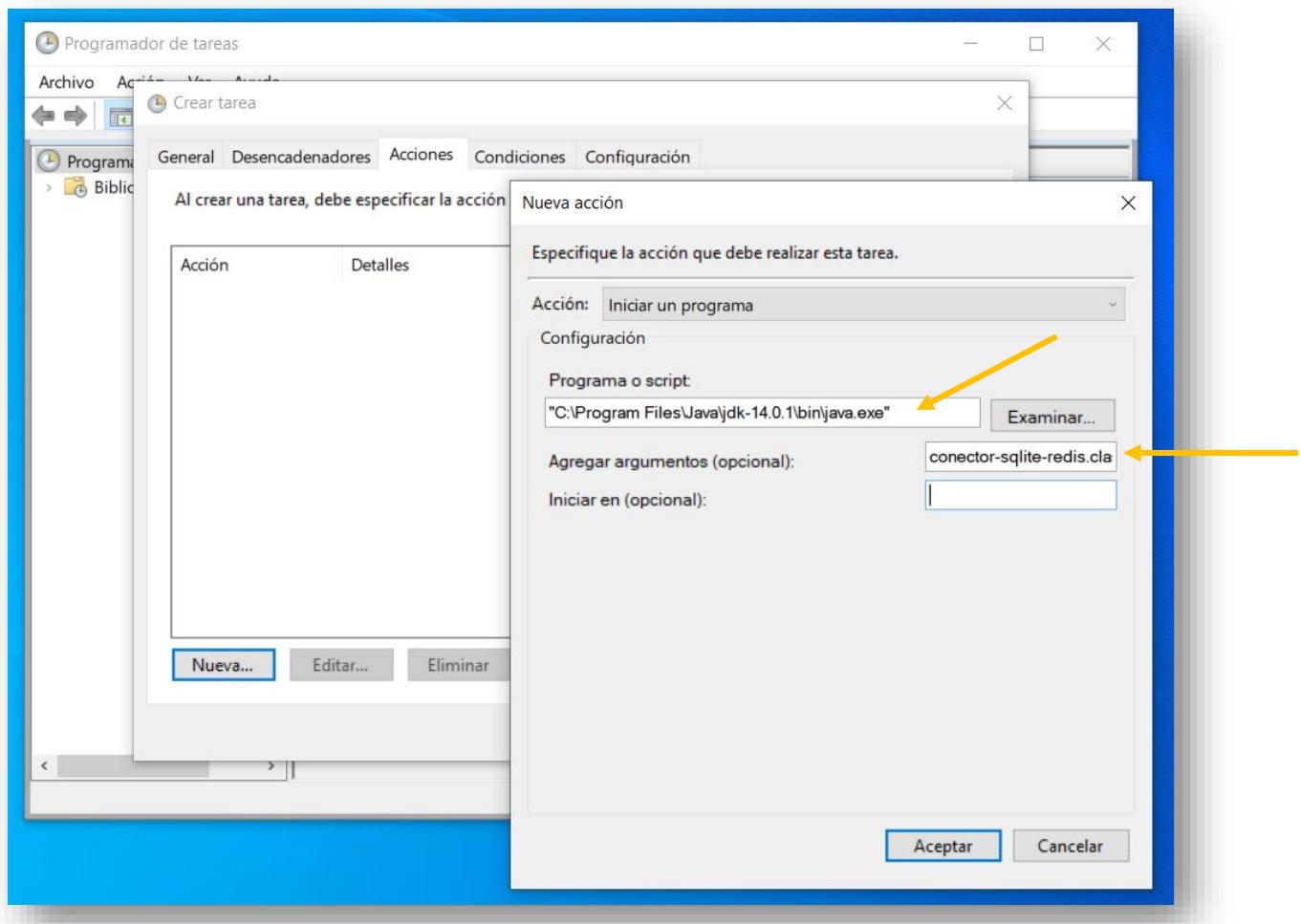


klicken:

Wir geben die folgenden Parameter an:

Programm oder Skript: connector\_path

Hinzufügen von Argumenten: Name des Konnektors



Die oben genannten Parameter können je nach Lage des Konnektors variieren. Die Hauptidee ist die Ausführung des Programms **connector-sqlite-redis-v1.class**, wie sie normalerweise auf der Kommandozeile erfolgt:

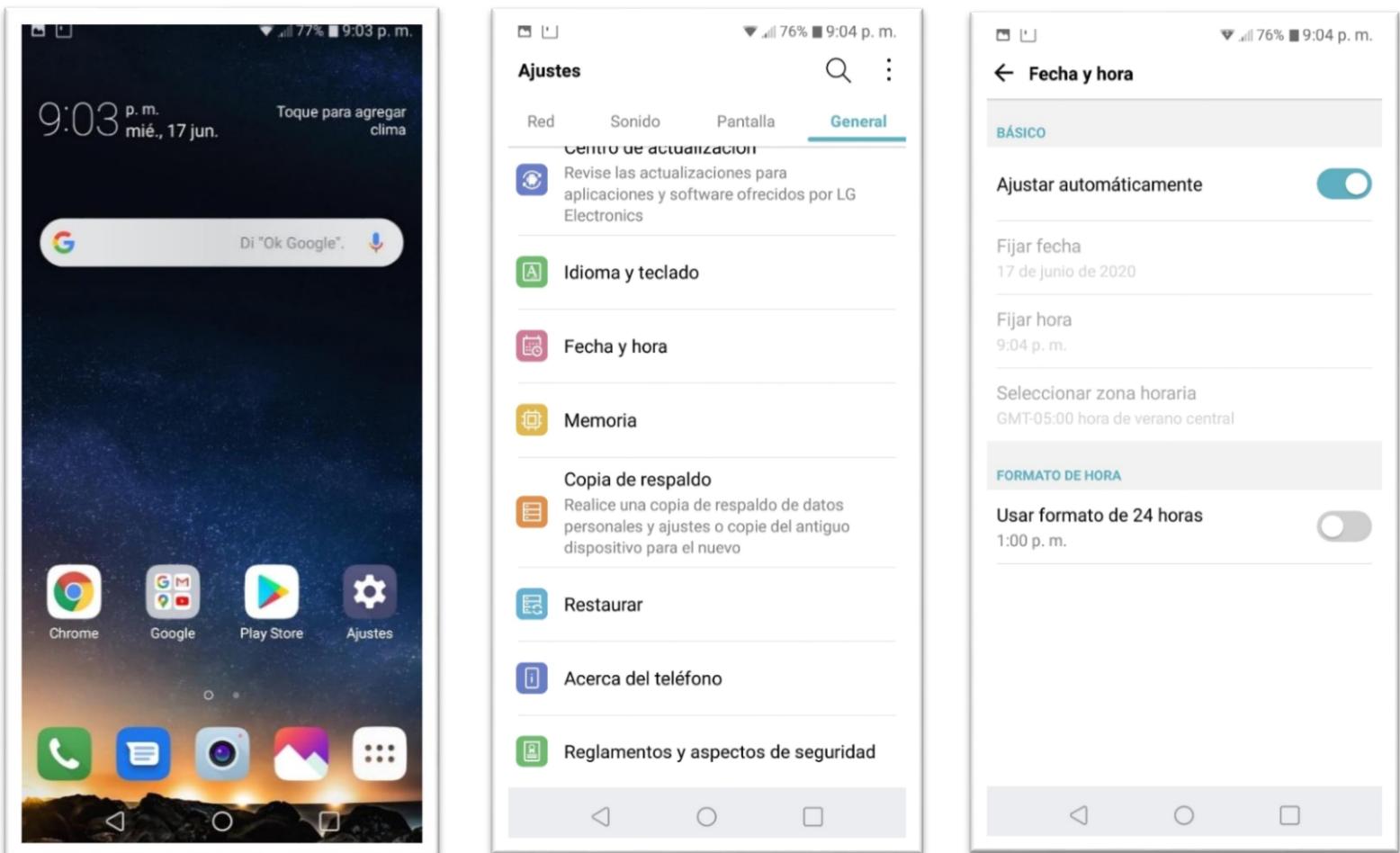
C:\jdk\_Verzeichnis java connector-sqlite-redis-v1

Zum Schluss müssen wir den Reiter "Auslöser" wählen, in dem wir die Parameter angeben, wann (Tag, Stunde, Minuten) die Aufgabe ausgeführt werden soll, diese Parameter basieren auf den Geschäftsregeln, die das Mini BlocklyChain-System erstellt werden müssen.

## 10. Synchronisierung in Systemknoten (Mobiltelefon) Minuten und Sekunden.

Es ist sehr wichtig, dass alle Knoten hauptsächlich im Minuten- und Sekundenteil synchronisiert sind. Da, wenn das Versenden oder die Veröffentlichung der Transaktionswarteschlange in einer bestimmten Zeit erfolgt, sollten alle Knoten synchronisiert werden, da dies vom Task-Manager abhängt, der im lokalen System mit dem CRON-Tool eingerichtet wird, das zur gleichen Zeit in allen Knoten ausgeführt wird, so dass alle Knoten die gleiche Wahrscheinlichkeit haben, das Recht zu erhalten, der Auserwählte zu sein, um die Transaktionswarteschlange bearbeiten und den neuen Block erzeugen zu können, um ihn der Blockkette des Systems hinzuzufügen. Zur Synchronisierung der Knoten haben wir zwei Möglichkeiten.

Die erste Option der Synchronisierung des Knotens, wir werden in der Lage sein, dies auf einfache Weise zu tun. In unserem Gerät ist durch die interne Option, die das Android-System beinhaltet, müssen wir zum Teil **Einstellungen > Datum und Zeit > Automatisch**



anpassen gehen

Mit der vorherigen Konfiguration werden wir in Minuten und Sekunden alle Knoten des Systems synchronisiert haben, egal welches Land in der Welt bereits synchronisiert sind, basiert auf jedem geographischen Gebiet möglicherweise, was ändert sich die Zeit, aber je nach Minuten und Sekunden synchronisiert werden sollte dies ist genug für uns, um den Prozess der Aufgaben auf einer geplanten Basis mit dem Cron-Tool in allen Knoten laufen zu lassen, um jede bestimmte Zeit in Minuten, dh wir können eine Aufgabe in crontab zu erstellen, um alle 10 Minuten oder 30 Minuten laufen, je nach jedem System-Design.

Dies ist bei Verwendung des "Peer to Peer"-Kommunikationsnetzwerks nützlich, bei Verwendung des Backup-Netzwerks entfällt dieser Vorgang, da die Verteilung der Transaktionswarteschlange in einem Client-Server-Modell erfolgt und der Server das Cron-Tool steuert.

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

Die zweite Möglichkeit ist die Verwendung einer externen API, bei der wir den Curl-Befehl über die Erweiterung (**ConnectorSSHClient**) ausführen werden.

Der Ort, an dem wir die externen Dienste des NTP (Network Time Protocol) nutzen werden, ist

<http://worldtimeapi.org/>

Jetzt werden wir nach einer Möglichkeit suchen, die Zeit von den NTP-Servern zu erhalten, die sich weltweit befinden, und die uns helfen wird, alle Knoten dazu zu bringen, eine Anfrage zu einer bestimmten Zeit am selben Datum und zur selben Zeit zu stellen.

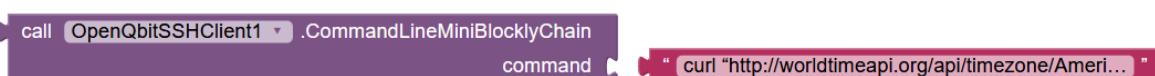
Beispiel für eine Abfrage mit Erweiterung (**ConnectorSSHClient**).

\$-Locke "http://worldtimeapi.org/api/timezone/America/Mexico\_City"

Wir haben die Verbindung zum Termux-Terminal hergestellt:



Wir führen den Curl-Befehl aus:



Wir müssen berücksichtigen, dass das Ergebnis des Curl-Befehls im JSON-Format vorliegen wird, ähnlich wie das folgende Ergebnis:

```
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
```

Das Ergebnis könnte auch im JSON-Format ohne die in oder JSON in linearer Form formatierten Daten vorliegen, wie unten dargestellt:

```
{"Abkürzung":"CDT","client_ip":"200.77.16.151","datetime":"2020-06-18T14:19:07.216800-05:00","Tag_der_Woche":4,"Tag_des_Jahres":170,"dst":true,"dst_von":"2020-04-05T08:00:00+00:00","dst_Offset":3600,"dst_Bis":"2020-10-25T07:00:00+00:00","raw_offset":-21600,"timezone":"Amerika/Mexiko_Stadt","unixtime":1592507947,"utc_datetime":"2020-06-18T19:19:07.216800+00:00","utc_offset":"-05:00","week_number":25}
```

Entweder müssen wir die Informationen durch eine bestehende JSON-Erweiterung wie "JSONTOOLS" filtern oder Filter im App Inventor in der Textverarbeitung verwenden, um nur die Stunde, den Tag oder die Sekunden je nach Bedarf des jeweiligen Systems zu erhalten. Nach der Verarbeitung des Ergebnisses kann ein logischer Vergleich vorgenommen werden, und auf der Grundlage dieses Vergleichs können wir eine bereits mit dem "cron"-Dienst programmierte Aufgabe ausführen, deren Konfiguration wir später in jedem Knoten sehen werden.

Verweis auf die JSONTOOLS-Erweiterung:

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

Wir haben nun zwei Optionen für die Zeitsynchronisierung der Knoten geprüft. Wir werden weiterhin den "cron"-Dienst auf jedem Knoten konfigurieren.

Konfiguration der automatischen Aufgabenplanung für die Ausführung mit dem **CRON-Dienst** auf Android-Systemen (Systemknoten).

Zuerst müssen wir verstehen, wie der automatische Scheduler funktioniert.

Der Cron-Dienst ist normalerweise auf allen Systemen vorinstalliert und befindet sich in einer Datei namens crontab, in der alle automatisch auszuführenden Aufgaben geplant sind.

Wir editieren die crontab-Datei mit **\$ crontab -e**, wir benutzen den **vi-Editor**, darin verwenden wir das Format:

```
# m h dom mon dow Benutzerbefehl
```

wo:

- **m** entspricht der Minute, in der das Skript ausgeführt wird, der Wert geht von 0 bis 59
- **h** die genaue Uhrzeit, das Format ist 24 Stunden, die Werte gehen von 0 bis 23, wobei 0 12:00 Mitternacht ist.
- **dom** bezieht sich auf den Tag des Monats, z.B. können Sie 15 angeben, wenn Sie jeden 15. laufen lassen wollen.
- **dow** steht für den Wochentag, er kann numerisch sein (0 bis 7, wobei 0 und 7 für Sonntag stehen) oder die ersten 3 Buchstaben des Tages auf Englisch: mon, tue, wed, thu, fri, sat, sun.
- **user** definiert den Benutzer, der den Befehl ausführt, es kann root sein, oder ein anderer Benutzer, solange er die Rechte zur Ausführung des Skripts hat.
- **Befehl** bezieht sich auf den Befehl oder den absoluten Pfad des auszuführenden Skripts, Beispiel: /home/user/scripts/update.sh, wenn Sie ein Skript aufrufen, muss es ausführbar sein

Um dies zu verdeutlichen, werden einige Beispiele für Cron-Aufgaben erläutert:

```
15 10 * * * Benutzer /home/benutzer/skripte/aktualisieren.sh  
Sie werden das Skript update.sh täglich um 10:15 Uhr ausführen.
```

```
15 22 * * * Benutzer /home/benutzer/skripte/aktualisieren.sh  
Sie werden das Skript update.sh täglich um 22:15 Uhr ausführen.
```

```
00 10 * * * 0 root apt-get - und Benutzer root aktualisieren  
Sie werden jeden Sonntag um 10:00 Uhr ein Update durchführen.
```

45 10 \* \* \* sun root apt-get - und aktualisieren

Der Root-Benutzer wird jeden Sonntag (Sonne) um 10:45 Uhr ein Update ausführen.

Wir haben Änderungen im Editor gespeichert und damit die Konfiguration des Cron-Dienstes abgeschlossen. Falls Sie den Cron nicht im System installiert haben, können Sie dies mit dem folgenden Befehl tun:

**\$ apt install cron**

## 2. Installation und Konfiguration von Netzwerkknoten - Mobiltelefone.

Beginnen wir mit dem Kommunikationsnetzwerk für Knoten, die Mini BlocklyChain verwenden werden.

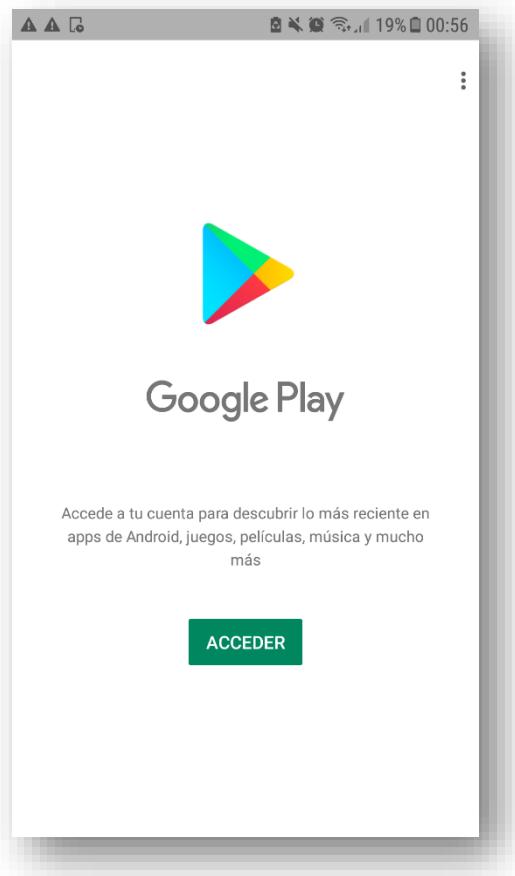
Zunächst brauchen wir eine Linux-Umgebung, da jedes Android-System auf Linux basiert, um Sicherheit und Flexibilität bei den Tools zu gewährleisten. Wir werden das "Termux"-Terminal verwenden, das diese Umgebung enthält, in der wir das Kommunikationsnetzwerk installieren werden.

Termux ist ein Linux-Emulator, in dem wir die notwendigen Pakete installieren werden, um unser Kommunikationsnetz zwischen den Knoten zu erstellen.

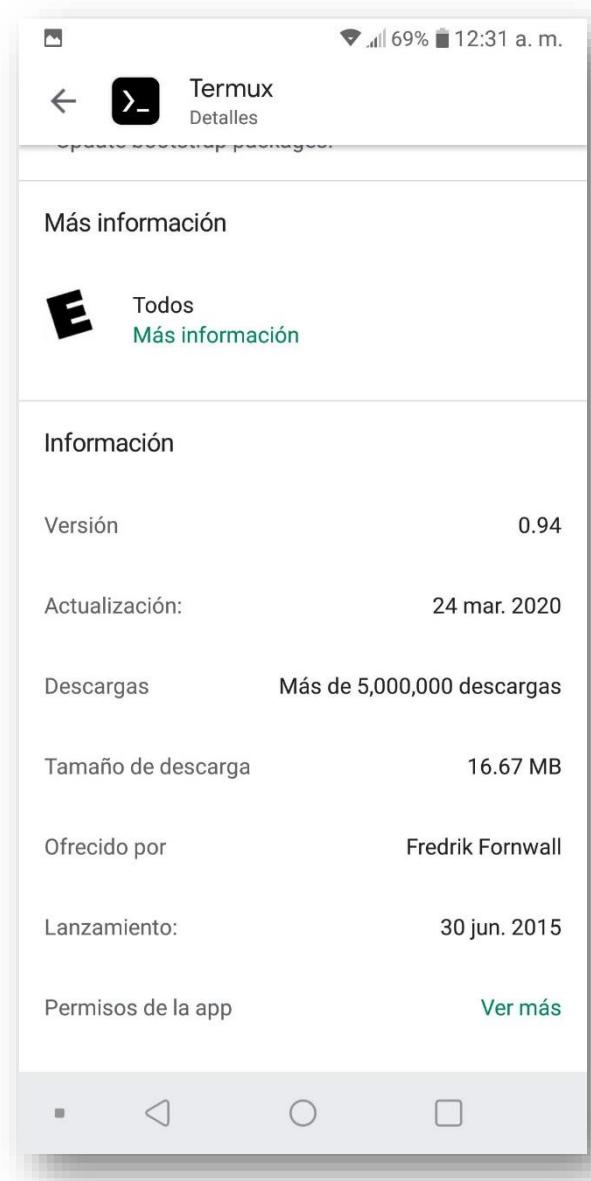
Einer der Hauptvorteile der Verwendung von Termux besteht darin, dass Sie Programme installieren können, ohne das Mobiltelefon (Smartphone) "drehen" zu müssen. Dadurch wird sichergestellt, dass durch diese Installation keine Herstellergarantie verloren geht.

Termux-Installation.

Rufen Sie von Ihrem Handy aus die Anwendung mit dem Google Pay-Symbol auf ([play.google.com](https://play.google.com)).



Suchen Sie nach der Anwendung "Termux", wählen Sie diese aus und starten Sie den Installationsprozess.



## Start der Termux-Anwendung.

Nach dem Start müssen wir die folgenden beiden Befehle ausführen, um Updates des Linux-Betriebssystememulators durchzuführen:

**\$ apt Aktualisierung**

OpenQbit.com

Seite 44 | 209

## \$ apt Upgrade

Bestätigen Sie alle Optionen Y(Ja)...

Termux

Home \$ apt Aktualisierung

\$ apt Upgrade

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

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

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

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

A standard Android-style keyboard is visible at the bottom.

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

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

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

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

A standard Android-style keyboard is visible at the bottom.

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

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

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

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

A standard Android-style keyboard is visible at the bottom.

## 11. Speicherkonfiguration innerhalb von Termux.

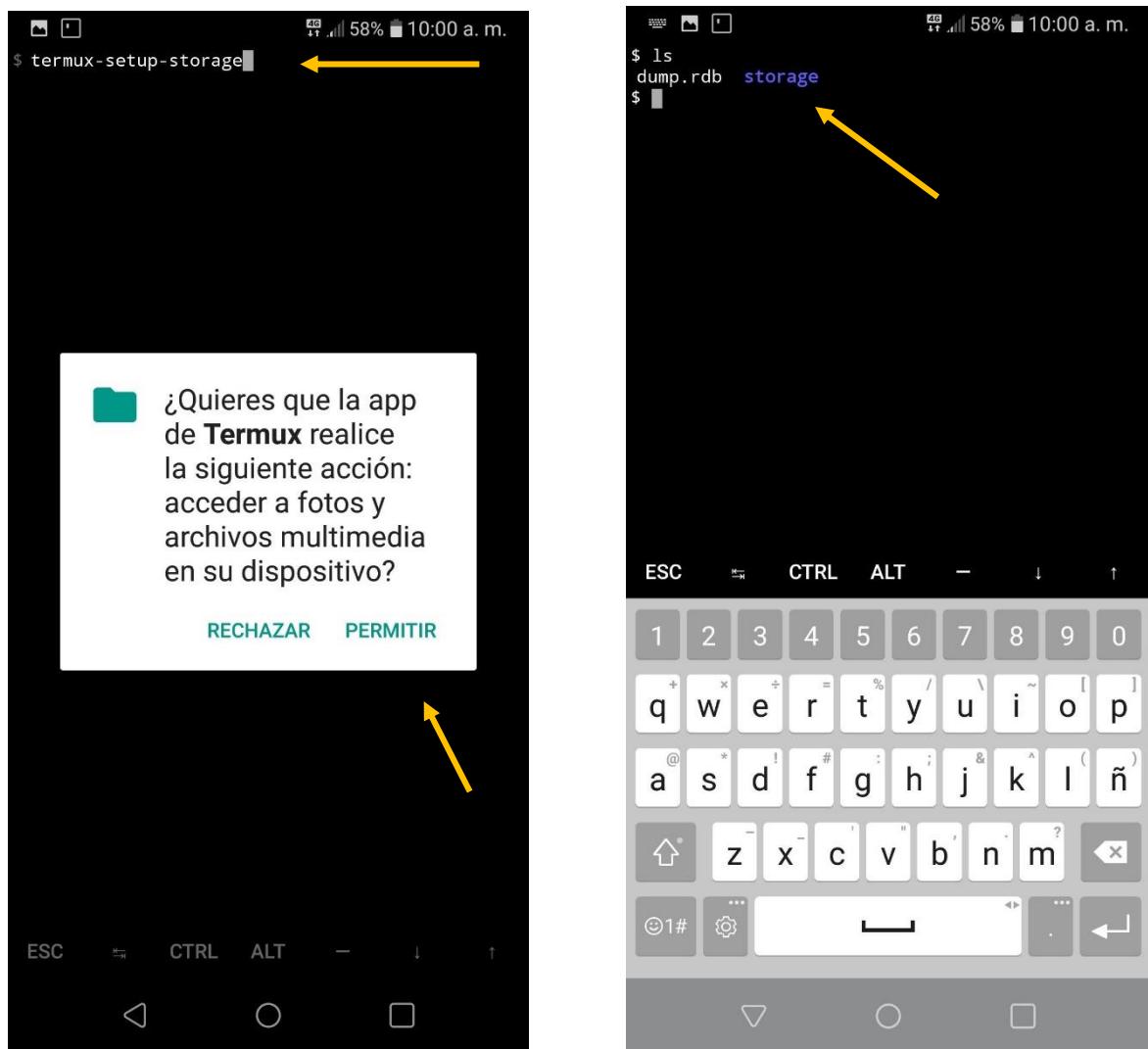
Nachdem Sie das Termux-System aktualisiert und aufgerüstet haben, beginnen wir mit der Konfiguration, wie der interne Speicher des Telefons im Termux-System angezeigt werden soll, damit Sie Informationen zwischen Termux und unseren Informationen im Telefon austauschen können.

Dies kann einfach und schnell durch Ausführen des folgenden Befehls auf einem Termux-Terminal erfolgen.

### \$ termux-setup-Lagerung

Wenn Sie den vorherigen Befehl ausführen, erscheint ein Fenster, in dem Sie aufgefordert werden, die Erstellung eines virtuellen **Speichers** (Verzeichnisses) in Termux zu bestätigen. Wir verifizieren, indem wir den Befehl geben:

\$ ls



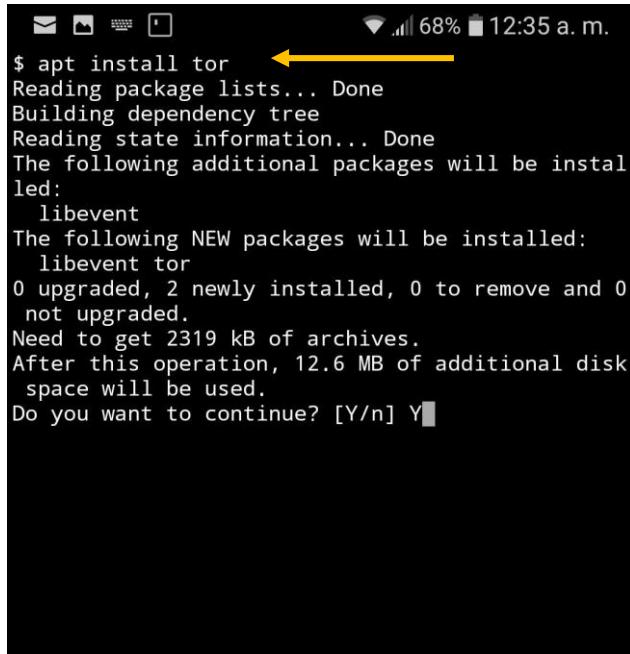
## 12. "Tor"-Netzwerkinstallation und "Syncthing"-Installation.

\$ apt install tor

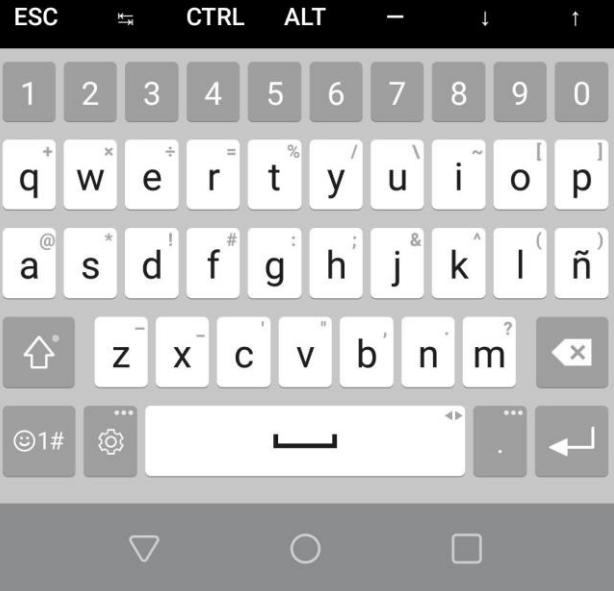
\$ apt install syncthing

Akzeptieren Sie die Installation, indem Sie auf Wunsch in beiden Fällen ein großes Y eingeben...

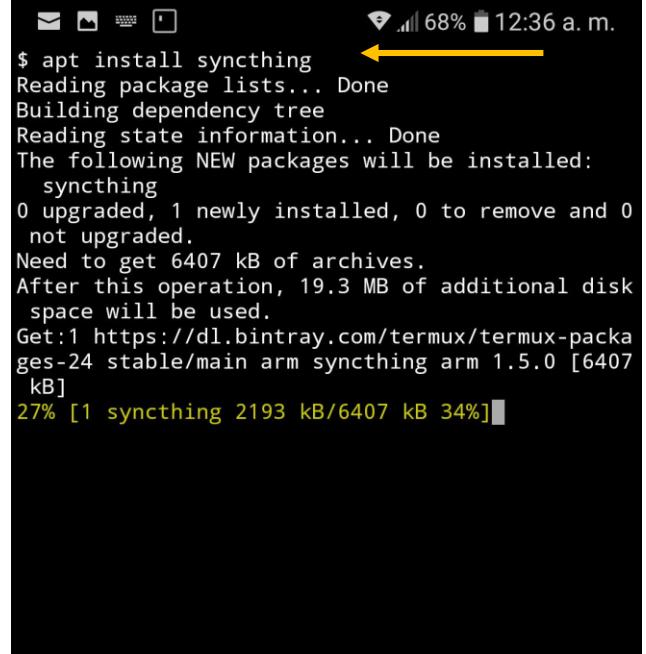
**\$ apt install tor**



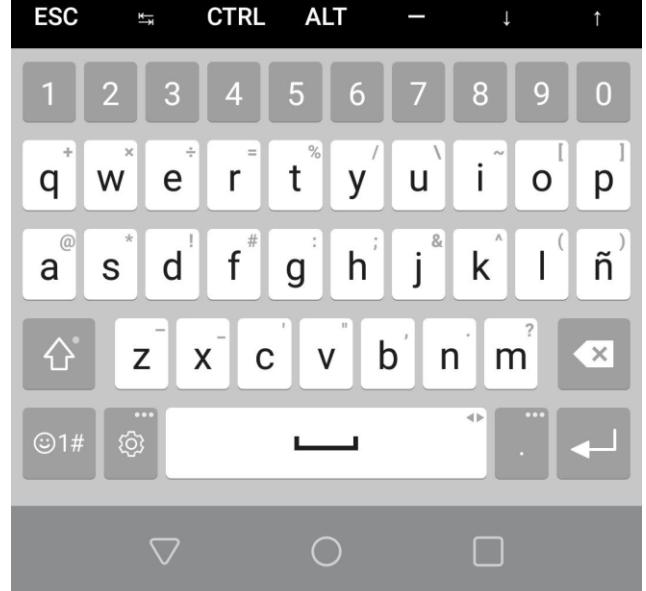
```
$ apt install tor
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libevent
The following NEW packages will be installed:
libevent tor
0 upgraded, 2 newly installed, 0 to remove and 0
not upgraded.
Need to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



**\$ apt install syncthing**



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



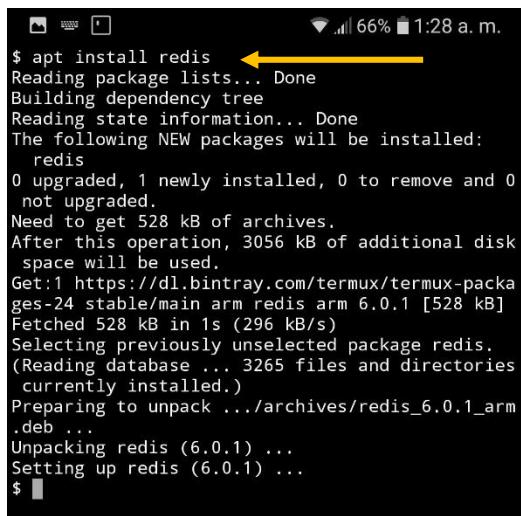
### 13. Installation der Datenbank "Redis" und des SSH (Secure Shell)-Servers.

```
$ apt install redis
```

```
$ apt install openssh
```

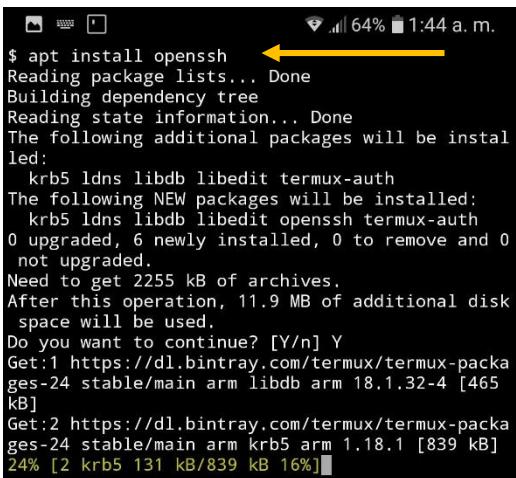
```
$ apt install sshpass
```

**\$ apt install redis**



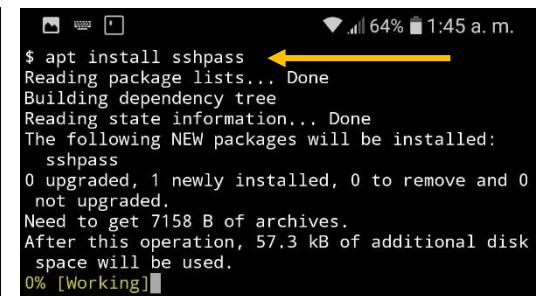
```
$ apt install redis
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  redis
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 528 kB of archives.
After this operation, 3056 kB of additional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

**\$ apt install openssh**



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

**\$ apt install sshpass**



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

Wir sind mit der Installation des Kommunikationsnetzes fertig, wir fahren mit der Konfiguration der Pakete fort: Tor, Syncthing und Redis DB.

## 14. Konfiguration des SSH-Servers auf dem Mobiltelefon (Smartphone).

Wir werden es dem SSH-Server im Handy ermöglichen, sich von unserem PC aus mit dem Handy zu verbinden und schneller und komfortabler arbeiten zu können, außerdem wird er dazu dienen, zu überprüfen, ob der Dienst des SSH-Servers im Handy korrekt funktioniert, da wir ihn im Kommunikationsnetz in der Mini-BlocklyChain verwenden werden.

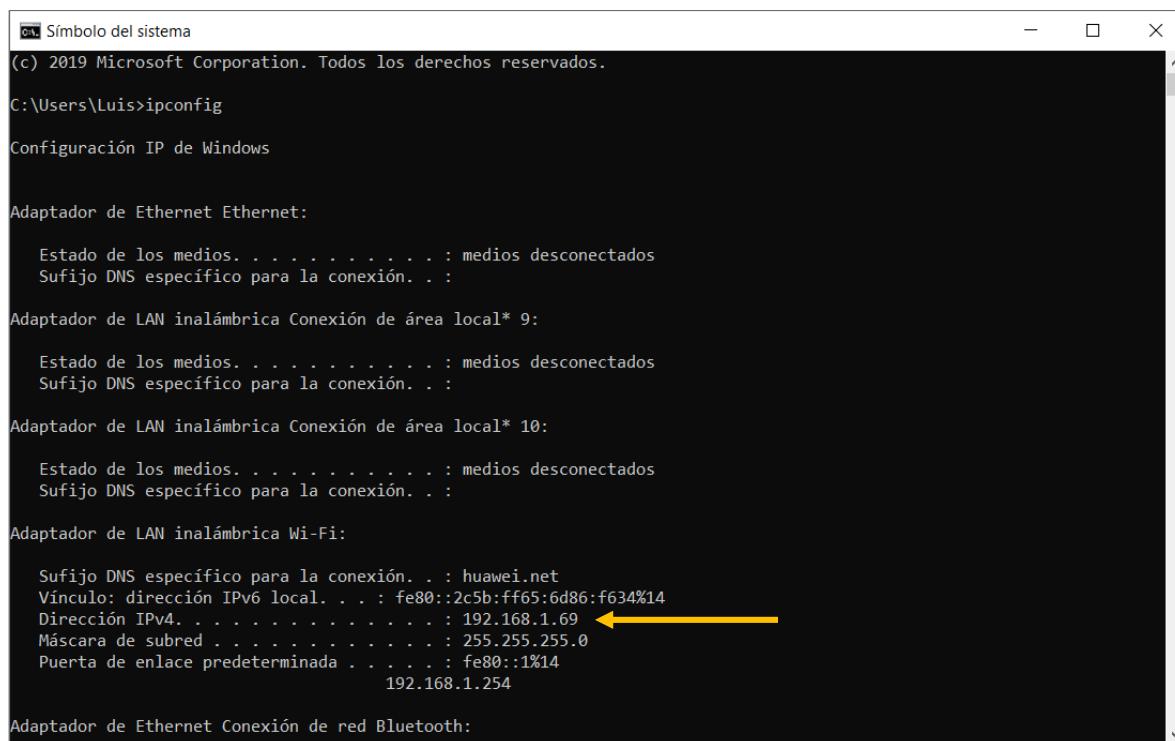
Als erstes müssen wir das Handy und den PC an dasselbe **WiFi-Netzwerk** anschließen, damit sie sich gegenseitig sehen können. Die IPs oder Adressen müssen ähnlich wie 192.168.XXX.XXX sein, die XXX-Werte sind variable Zahlen, die in jedem Computer zufällig zugewiesen werden.

Dieses Beispiel wurde auf einem LG Q6-Mobiltelefon und einem PC mit Windows 10 Home getestet.

Überprüfen Sie die IP oder die Adresse, die der PC mit dem WiFi verbunden hat, wir müssen ein Terminal in Windows öffnen.

Schreiben Sie in das untere Feld, in dem die Suchlupe steht, cmd und drücken Sie die Eingabetaste. Es öffnet sich ein Terminal, in das wir den Befehl schreiben:

C:Benutzername> ipconfig



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

C:\Users\Luis>ipconfig

Configuración IP de Windows

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

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

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

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

Adaptador de Ethernet Conexión de red Bluetooth:
```

Es wird uns zeigen, dass die dem PC darin zugewiesene IP 192.168.1.69 ist, aber dies ist höchstwahrscheinlich von Fall zu Fall unterschiedlich.

ANMERKUNG: Die Adresse, bei der "IPv4-Adresse" steht, ist zu nehmen, nicht zu verwechseln mit dem Gateway.

Nun müssen wir im Falle des Mobiltelefons im Termux-Terminal den folgenden Befehl eingeben, um den Namen unseres Benutzers zu erfahren, mit dem wir uns mit dem SSH-Server verbinden werden, der unser Telefon hat, führen wir folgenden Befehl aus:

Nun müssen wir im Falle des Mobiltelefons im Termux-Terminal den folgenden Befehl eingeben, um den Namen unseres Benutzers zu erfahren, mit dem wir uns mit dem SSH-Server verbinden werden, der unser Telefon hat, führen wir folgenden Befehl aus:

**\$ whoami**

Später müssen wir diesem Benutzer ein Passwort geben, so dass wir den folgenden Befehl ausführen müssen:

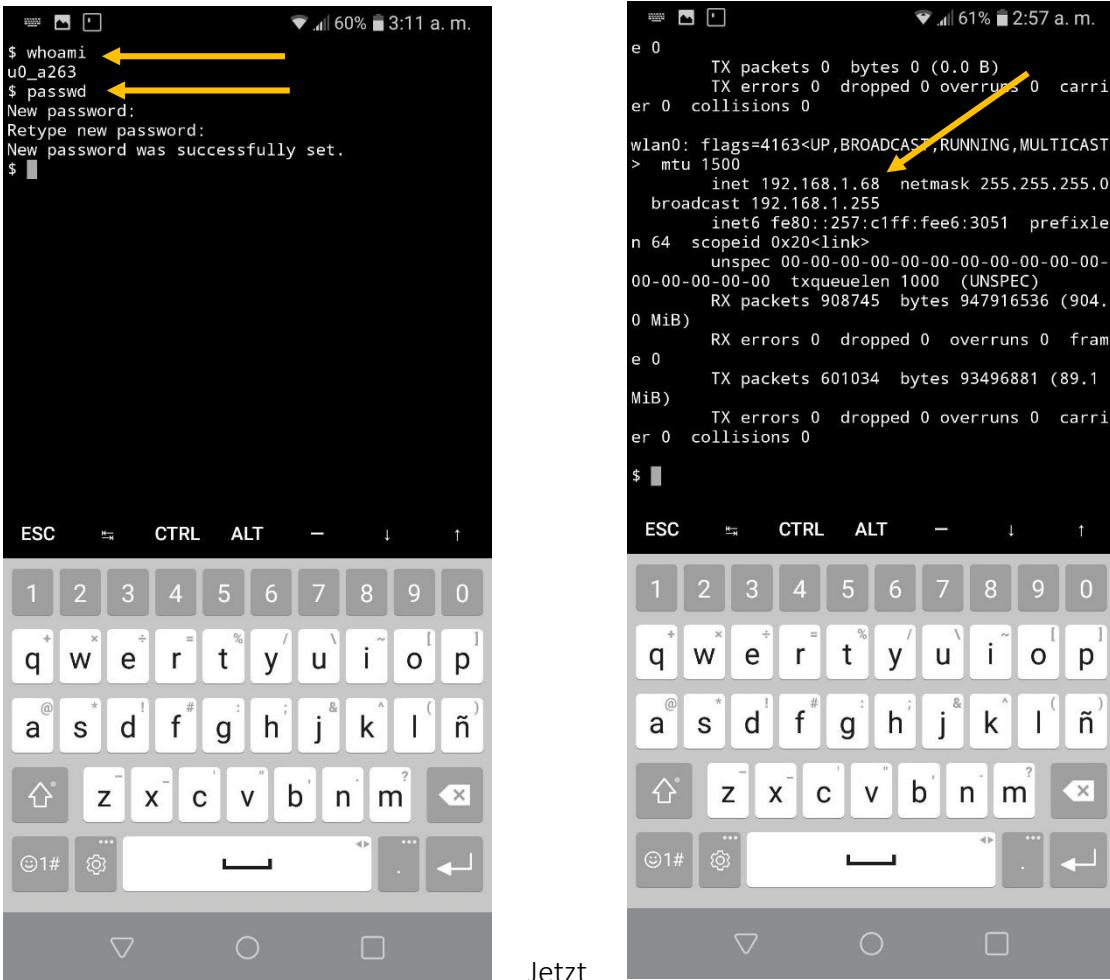
**\$ passwd**

Es fordert uns auf, ein Passwort einzugeben und die Eingabetaste zu drücken, wieder fragt es uns nach dem Passwort, wir bestätigen es und drücken die Eingabetaste, wenn es **erfolgreich war "Neues Passwort wurde erfolgreich gesetzt"** im Falle der Markierung ist es möglich, dass das Passwort nicht korrekt eingegeben wurde. Führen Sie das Verfahren erneut durch.

Und dann, um zu wissen, welche IP wir in Termux haben, geben wir den folgenden Befehl ein, die IP steht nach dem Wort "**inet**":

**\$ ifconfig -a**

Mini BlocklyChain - Heimwerken - 'Do It Yourself'



ist es an der Zeit, den SSH-Server-Dienst auf Ihrem Telefon zu starten, damit Sie Sitzungen von Ihrem PC empfangen können. Wir führen den folgenden Befehl im Termux-Terminal aus, dieser Befehl liefert kein Ergebnis.

\$ sshd



Jetzt müssen wir auf dem PC ein Programm installieren, das vom PC aus mit dem SSH-Server des Telefons kommuniziert.

Wir müssen zu <https://www.putty.org> gehen.

Wählen Sie, wo der Link "Sie können PuTTY hier herunterladen" ist



### Download PuTTY

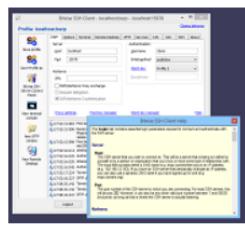
PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform, with source code and is developed and supported by a group of volunteers.

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 by Bitvise Software Ltd. It supports all features supported by PuTTY, as well as the following:

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

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

Wählen Sie die 32-Bit-Version, es spielt keine Rolle, ob Ihr System mit 64-Bit funktioniert.

## Download PuTTY: latest release

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

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

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

**Package files**

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

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

**MSI ('Windows Installer')**

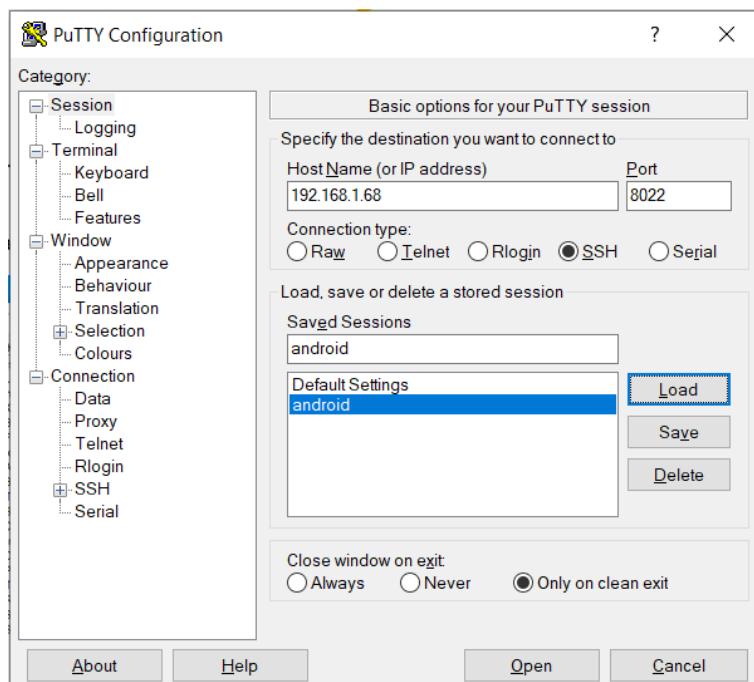
32-bit:	<a href="#">putty-0.73-installer.msi</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>
64-bit:	<a href="#">putty-64bit-0.73-installer.msi</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>

**Unix source archive**

.tar.gz:	<a href="#">putty-0.73.tar.gz</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>
----------	-----------------------------------	------------------------------	-----------------------------

Sobald es auf Ihren PC heruntergeladen wurde, führen Sie es aus und installieren es mit den Standardoptionen. Starten Sie dann die PuTTY-Anwendung.

In dieser Sitzung geben wir die Daten von unserem Openssh-Server ein, den wir auf dem Mobiltelefon installiert haben.



die Schaltfläche Speichern klicken.

Geben Sie die IP des Mobiltelefons ein.

HostName oder IP-Adresse:

**192.168.1.68** (IP-Beispiel)

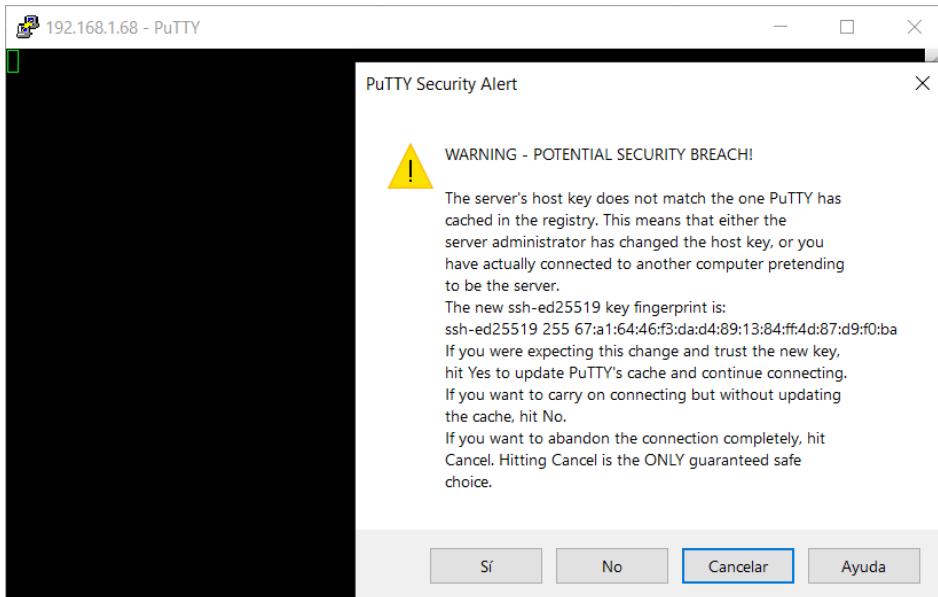
Hafen:

**8022** (Standard-Port des mobilen SSH-Servers).

Wir können der Sitzung unter "Gespeicherte Sitzungen" einen Namen geben und auf

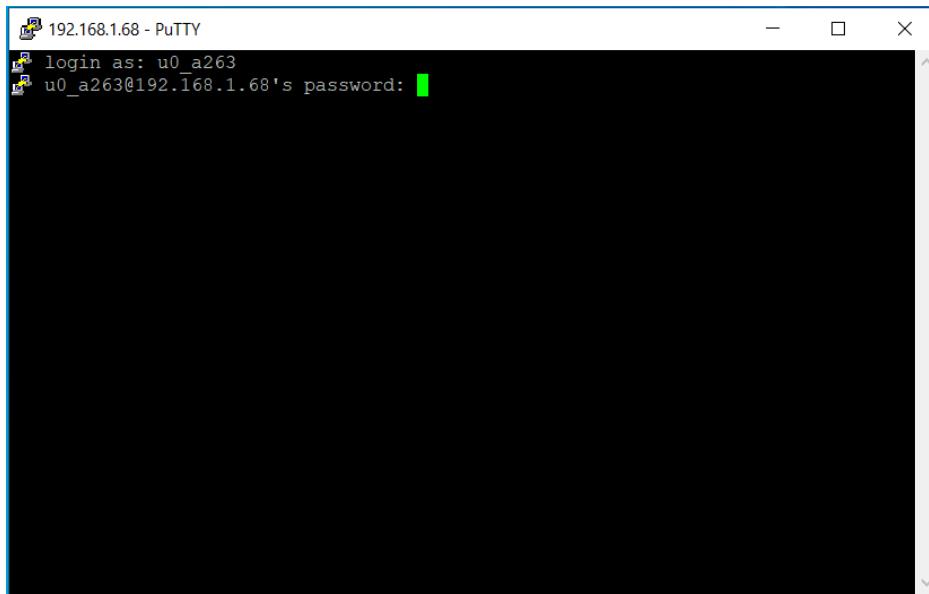
Später im unteren Teil drücken wir auf , um eine Verbindung zum Server zu öffnen, indem wir den Knopf "Öffnen" drücken.

Auf dem PC, wenn Sie zum ersten Mal eine Verbindung herstellen, werden Sie zur Bestätigung des Informationsverschlüsselungsschlüssels **aufgefordert**, indem Sie auf die Schaltfläche "Ja" klicken.



Später werden wir nach dem Benutzer gefragt, mit dem wir uns verbinden wollen. Wir werden die Informationen verwenden, die wir zuvor erhalten haben (Benutzer und Passwort).

Im **Login als:** müssen wir unseren Benutzer eingeben und Enter geben, dann werden wir erneut nach dem Passwort fragen und die Enter-Taste geben.



Wenn die Daten korrekt waren, befinden wir uns in einer SSH-Sitzung (Secure Shell), die vom PC (Client) aus am Telefon (SSH-Server) durchgeführt wird.

A screenshot of a Termux terminal window titled "192.168.1.68 - PuTTY". The window displays a series of welcome messages and instructions for using the package manager. It starts with "Welcome to Termux!", followed by links to the wiki, community forum, Gitter chat, and IRC channel. It then provides instructions for working with packages, subscribing to additional repositories, and reporting issues. The text is white on a black background with a vertical scroll bar on the right.

**WICHTIGER HINWEIS:** Denken Sie daran, dass sich die IP (Adresse) des PCs und die IP (Adresse) des Mobiltelefons, das an dasselbe WiFi angeschlossen ist, wahrscheinlich jedes

Mal ändern werden, wenn wir die Verbindung trennen und wieder herstellen, so dass wir doppelt prüfen müssen, welche Adressen jedes Gerät hat, dies wird den Erfolg der Verbindung zwischen den Geräten über den SSH-Server des Telefons und den PC (Client) sicherstellen.

Bisher konnten wir uns nur mit demselben WiFi-Netzwerk verbinden, aber wenn wir unser Telefon außerhalb desselben Netzwerks wie den PC bewegen, werden wir keine Verbindung herstellen können, weil es verschiedene Netzwerke gibt, die über andere, kompliziertere Kommunikationsgeräte laufen. Wir werden dies lösen, wenn wir das "Tor"-Netzwerk einrichten.

Denken Sie daran, dass diese Verbindung nur hergestellt wird, um den Dienst des Servers zu verifizieren, den wir auf dem Telefon installiert haben, und um eine komfortablere Arbeitsumgebung mit einer Fernsitzung von einem PC zum Telefon zu haben.

## 15.Tor"-Netzwerkkonfiguration mit SSH (Secure Shell)-Dienst.

Mit der Fernsitzung vom PC aus beginnen wir mit der Konfiguration des "Tor"-Netzwerks mit aktiviertem SSH-Dienst.

Die Bedeutung des "Tor"-Netzwerks liegt darin, den Geräten die Eigenschaft zu verleihen, überall auf der Welt über das Internet kommunizieren zu können, ohne sich im selben WiFi-Netzwerk zu befinden. Egal, wo wir uns befinden, wir werden in der Lage sein, uns zu verbinden und das "Peer-to-Peer"-Netzwerk zwischen den Knoten zu bilden, das eine wesentliche Funktionalität der Mini-BlocklyChain-Architektur darstellt.

Das "Tor"-Netzwerk hat eine große Flexibilität in seiner Konfiguration, da es mehrere Parameter in seiner Konfigurationsdatei "torrc" hat. Diese Datei befindet sich im Pfad (`$PREFIX/etc/tor/torrc`) in unserem Fall mit der Termux-Sitzung von unserem PC aus, was wir wissen, wenn wir den folgenden Befehl eingeben:

```
$ echo $PREFIX
```

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

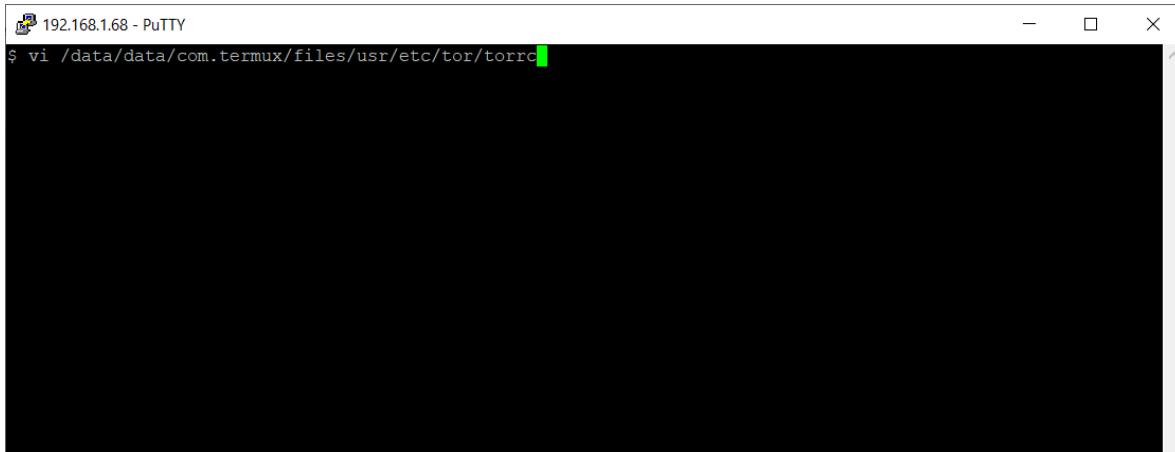
Daher befindet sich die Datei, die wir bearbeiten müssen, im Pfad:

`PREFIX/etc/tor/torrc`, die gleich `/data/data/com.termux/files/usr/etc/tor/torrc` ist

Wir fahren fort, die Konfigurationsdatei mit dem Kommandozeilen-Editor "vi" zu bearbeiten, indem wir den folgenden Befehl ausführen:

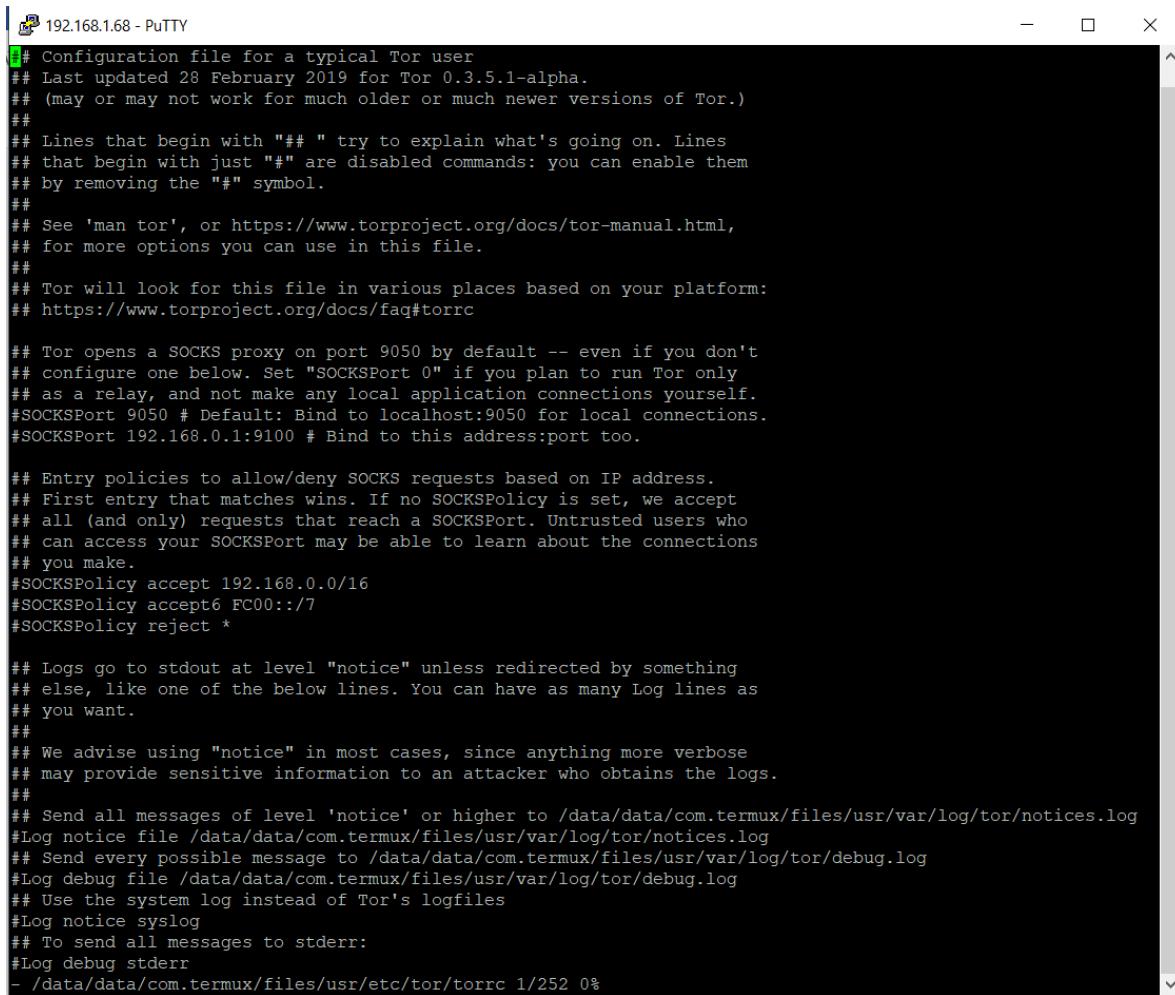
vi /data/data/com.termux/files/usr/etc/tor/torrc

Er wird diese Datei für uns bearbeiten, als Beispiel:



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

Bearbeitete "torrc"-Datei:



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

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

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

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

In dieser "torrc"-Datei müssen wir die Zeilen, die die Datei hat, hinzufügen oder verwenden, indem wir die folgenden Änderungen vornehmen, drei Zeilen, die die folgenden sind

Syntax: **SOCKSPort <Anwendungsportnummer>**

Beispiel: **SOCKSPort 9050**

Die Variable **SOCKSPort** sagt uns, dass dieser Kommunikations-Socket über das TCP-IP-Protokoll standardmäßig das mobile Gerät (Telefon) verwendet und Port 9050 geöffnet oder in Benutzung ist.

Syntax: **HiddenServiceDir <Verzeichnis, in dem die Konfiguration der Anwendung gespeichert wird>**

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

Die Variable **HiddenServiceDir** sagt uns, dass dies das Verzeichnis sein wird, in dem die Konfiguration des Dienstes, der über das Tor-Netzwerk genutzt wird, gespeichert wird. In diesem Verzeichnis finden Sie die Konfigurations- und Sicherheitsdatei für den Dienst, und in diesem Verzeichnis finden Sie eine Datei namens **hostname**.

Wir können sehen, dass die Adresse, die vom Tor-Netzwerk für den spezifischen Dienst vergeben wird, der in unserem Fall einen SSH-Dienst erstellt, der im Tor-Netzwerk implementiert wird. Das Verzeichnis, das wir als **hidden\_ssh** benennen, wird die **Hostnamen-Datei** enthalten. Dieses Verzeichnis muss nicht erstellt werden, denn wenn wir den Tor-Netzwerkdienst starten, wird es automatisch erstellt.

Um die vom Tor-Netzwerk bereitgestellte Adresse zu sehen, können wir den Befehl "more" verwenden. Bevor wir diesen Befehl verwenden, müssen wir die Konfiguration der Datei "torrc" beenden und den Tor-Netzwerkdienst registrieren, so dass das Verzeichnis "**hidden\_ssh**" und die darin enthaltenen Dateien erstellt werden, wie es bei der Datei "hostname" der Fall ist.

mehr /data/data/com.termux/files/

Der obige Befehl führt zu einer Adresse mit einer alphanumerischen Zeichenfolge mit der Erweiterung . onion ähnlich wie :

uqwthf6ojdmoybyzvudoq3x2weq7k7rjitblhba3pjawk2nvjmx3wer.onion

Schließlich müssen wir noch die Variable **HiddenServicePort** in die Konfigurationsdatei "torrc" einfügen. Dieser Parameter teilt dem Tor-Netzwerk mit, welchen Port die Anwendung, für die wir uns anmelden, über das Tor-Netzwerk benutzen wird.

Syntax: **HiddenServicePort <Abfahrthafen> <Lokale oder öffentliche IP>: <Abfahrthafen>**

O

HiddenServicePort <Abfahrthafen>

Beispiele:

VersteckterServicePort 22 127.0.0.1:8022

O

VersteckterServicePort 8022

Damit haben wir die Konfiguration des Tor-Netzwerks für generische Dienste und den SSH (Secure Shell)-Dienst abgeschlossen. Dieser Dienst wird für die SQLite-Datenbank-Update-Kommunikation verwendet werden, wo wir die Validierungsprozesse unseres Mini-BlocklyChain-Systems speichern werden.

Wir fahren mit der Konfiguration des Mini-BlocklyChain-Kommunikationsnetzes fort.

## 16. Peer-to-Peer-Systemkonfiguration mit manueller Synchronisierung.

Eine "Peer-to-Peer"-Architektur ist in einem System mit Blockkettentechnologie von grundlegender Bedeutung, da diese Architektur zwei zentrale Punkte in den Kommunikationsprozessen des Systems bietet: Einerseits sollen in allen Knoten jederzeit die gleichen aktualisierten (synchronisierten) Informationen bereitgestellt werden, unabhängig davon, ob sich die Knoten in einem privaten Netzwerk (Wifi) oder einem öffentlichen Netzwerk (Internet) oder einer Mischform dieser beiden befinden, und andererseits sind sie bei dieser Art von Architektur nicht von einem Vermittler (Server) abhängig, um Informationen zwischen den Knoten zu übertragen, zu aktualisieren oder abzufragen. All dies ist auf die Tatsache zurückzuführen, dass die Kommunikation direkt zwischen den Knoten erfolgt, in unserem Fall Mini BlocklyChain erfolgt die Kommunikation direkt zwischen den Telefonen, die das Netzwerk bilden, ohne einen Vermittler.

Für diese Aufgabe werden wir das Open-Source-Syncthing-Tool verwenden, das auf der Grundlage einer "Peer-to-Peer"-Architektur arbeitet.

Die Konfiguration von Syncthing für Geräte (Mobiltelefone) wird manuell und automatisch angezeigt. Die manuelle Form ist in der Synchronisation mit bis zu 5 Knoten angewendet, im Falle einer großen Anzahl von automatischen Konfiguration ist optimal, der Prozess



konzentriert sich auf die Registrierung der Knoten, mit denen wir die Synchronisation der ausgewählten Informationen durchführen wollen.

Die Voraussetzungen für den Start sind:

1. Haben den Dienst des Tor-Netzwerkes initiiert.
2. (optional - empfohlen) Haben Sie eine Anwendung installiert, die QR-Codes lesen kann, empfehlen wir die Anwendung App inventor Android, die einfach und leicht über Google Play zu installieren ist und die wir später in diesem Handbuch im Abschnitt "Definition und Verwendung von Blöcken in Mini-Blocklychain" verwenden werden.
3. Den Dienst von Synthing initiiert zu haben.

Wir zeigen die Anwendung App Inventor, die wir zum Lesen von QR-Codes verwenden werden, die uns in Zukunft für die Registrierung von Knoten in Syncthing dienen werden.

Das folgende Beispiel wurde zwischen zwei mobilen Geräten (Telefonen) anhand der folgenden Modelle erstellt:

Mobilgerät #1: Modell LG Q6

Mobiles Gerät Nr. 2: Samsung-Modell

Wir beginnen damit, die Tor-Netzwerkdiene und Synchronisierungsdienste mit den folgenden Befehlen zu starten, öffnen zwei Terminals innerhalb von Termux und führen jeden Befehl separat aus:

**\$ tor**

Nachdem Sie den Befehl zum Starten des Tor-Netzwerkprogramms eingegeben haben, sollten Sie überprüfen, ob die Ausführung erfolgreich war, indem Sie überprüfen, ob sie zu 100% durchgeführt wurde.

Wenn wir das Tor-Programm auf einem Termux-Terminal laufen lassen, verifizieren wir, dass es zu 100% läuft.

\$ 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 liste
ner on 127.0.0.1:9050
May 23 23:00:30.000 [notice] Parsing GEOIP IPv4
file /data/data/com.termux/files/usr/share/tor/g
eoip.
May 23 23:00:32.000 [notice] Parsing GEOIP IPv6
file /data/data/com.termux/files/usr/share/tor/g
eoip6.
May 23 23:00:34.000 [notice] Bootstrapped 0% (st
arting): Starting
May 23 23:00:34.000 [notice] Starting with guard
context "default"
May 23 23:00:35.000 [notice] Bootstrapped 5% (co
nn): Connecting to a relay
May 23 23:00:36.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 23 23:00:36.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 23 23:00:36.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 23 23:00:36.000 [notice] Bootstrapped 20% (o
nehop_create): Establishing an encrypted directo
ry connection
May 23 23:00:36.000 [notice] Bootstrapped 25% (r

```



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

```

Dann führen wir den Syncthing-Befehl aus:

### \$-Synchronisierung

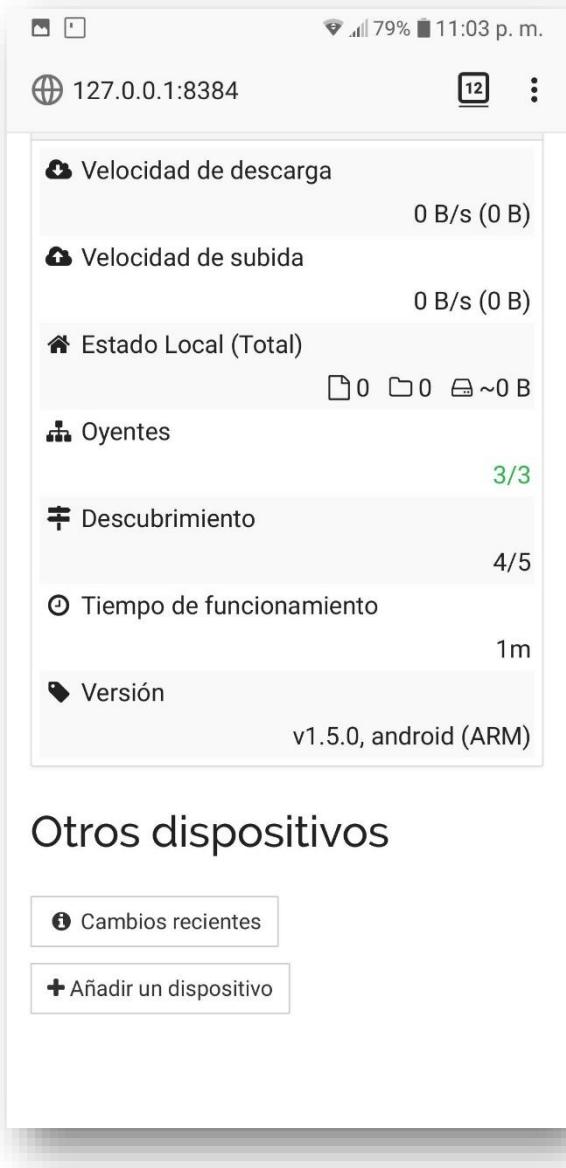
Nach der Ausführung dieses Befehls wird eine Verwaltungsseite im Browser unseres Telefons geöffnet, falls sie sich nicht automatisch öffnet, können wir zu jedem Browser gehen, den wir installiert haben, wo wir normalerweise im Internet surfen, und wir können die folgenden Schritte ausführen:

<http://127.0.0.1:8384>

Es öffnet den Verwaltungsbildschirm des Tools, das uns helfen wird, unsere Informationen zwischen allen Knoten (Telefonen) des Systems zu synchronisieren.



```
$ 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@6bdf8622238a 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 and/or linked to new config
[start] 04:02:08 INFO: Default config saved. Edit /data/data/com.termux/files/home/.config/syncthing/config.xml to taste (with Syncthing stopped) or use the GUI
[OWEPS] 04:02:08 INFO: My ID: OWEPSNA-6RZI6S7-SKVOU65-V44BC5Z-CXZOJI4-C3JC7HM-IPY4V35-JQAKPAW
[OWEPS] 04:02:09 INFO: Single thread SHA256 performance 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 schema version 1...
```



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

### Otros dispositivos

- [Cambios recientes](#)
- [Añadir un dispositivo](#)

Da wir den "Syncthing"-Verwaltungsbildschirm geöffnet haben, fahren wir damit fort, den oder die Knoten zu registrieren, deren Informationen wir synchronisieren möchten. An diesem Punkt werden wir das Programm besetzen, das QR-Codes liest.

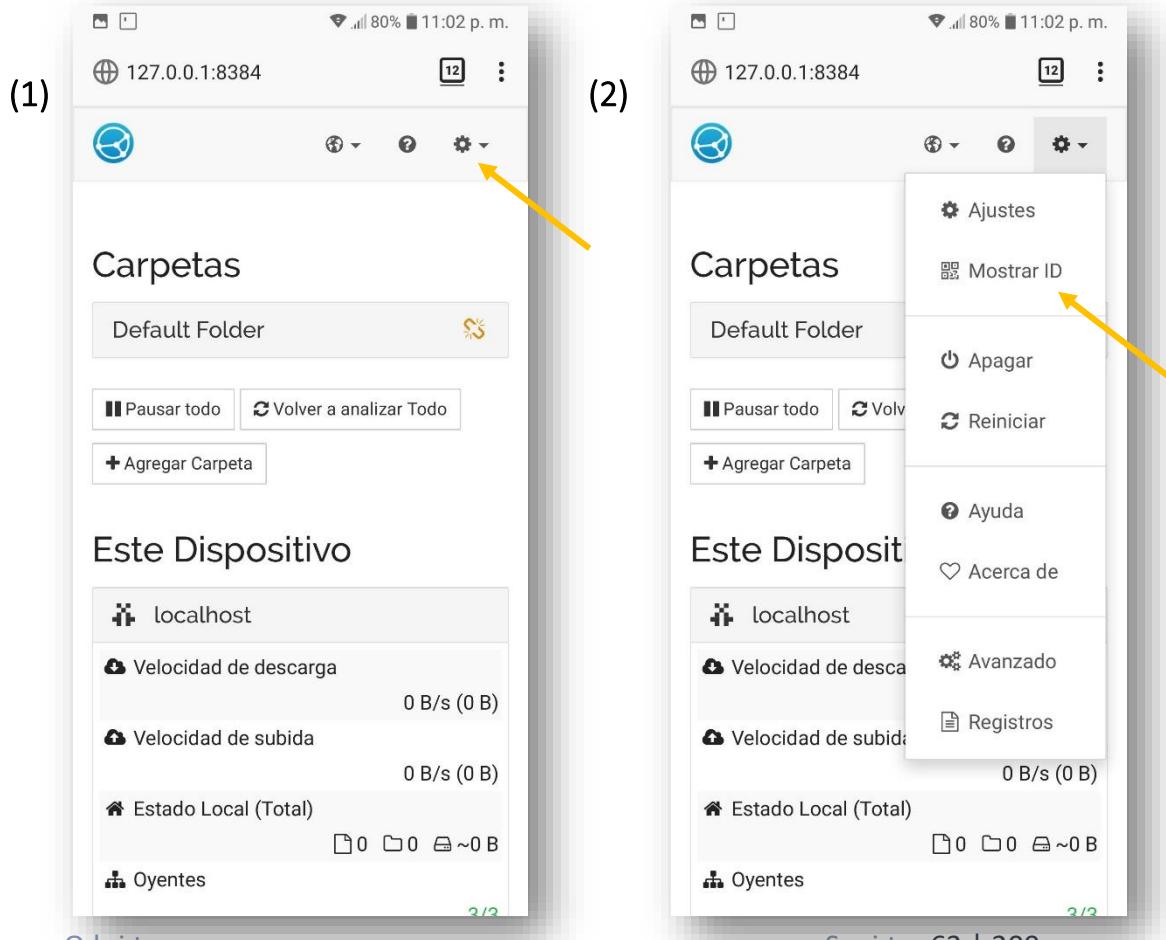
Wenn das Synchronisierungsprogramm zum ersten Mal gestartet wird, erstellt es einen eindeutigen Identifikator des Telefons, der aus einer Gruppe von acht alphanumerischen Zeichensätzen in Großbuchstaben besteht. Dieser Identifikator (ID) ist derjenige, den wir in dem oder den Knoten registrieren werden, die wir mit den Informationen synchronisieren wollen.

In unserem Fall muss die LG Q6-Telefon-ID auf dem Samsung-Telefon und die Samsung-Telefon-ID auf dem LG Q6 registriert werden. Sie müssen sich in beiden Telefonen befinden, damit sie richtig funktionieren.

Wir werden die Schritte der Samsung-Handy-Registrierung auf dem LG Q6-Telefon durchführen.

Zuerst (1) klicken wir oben auf dem Verwaltungsbildschirm (Internet-Browser) des Samsung-Telefons mit Syncthing auf den Menüreiter oben rechts.

Im zweiten (2) Schritt sehen wir ein Menü, in dem wir auf "ID anzeigen" klicken.

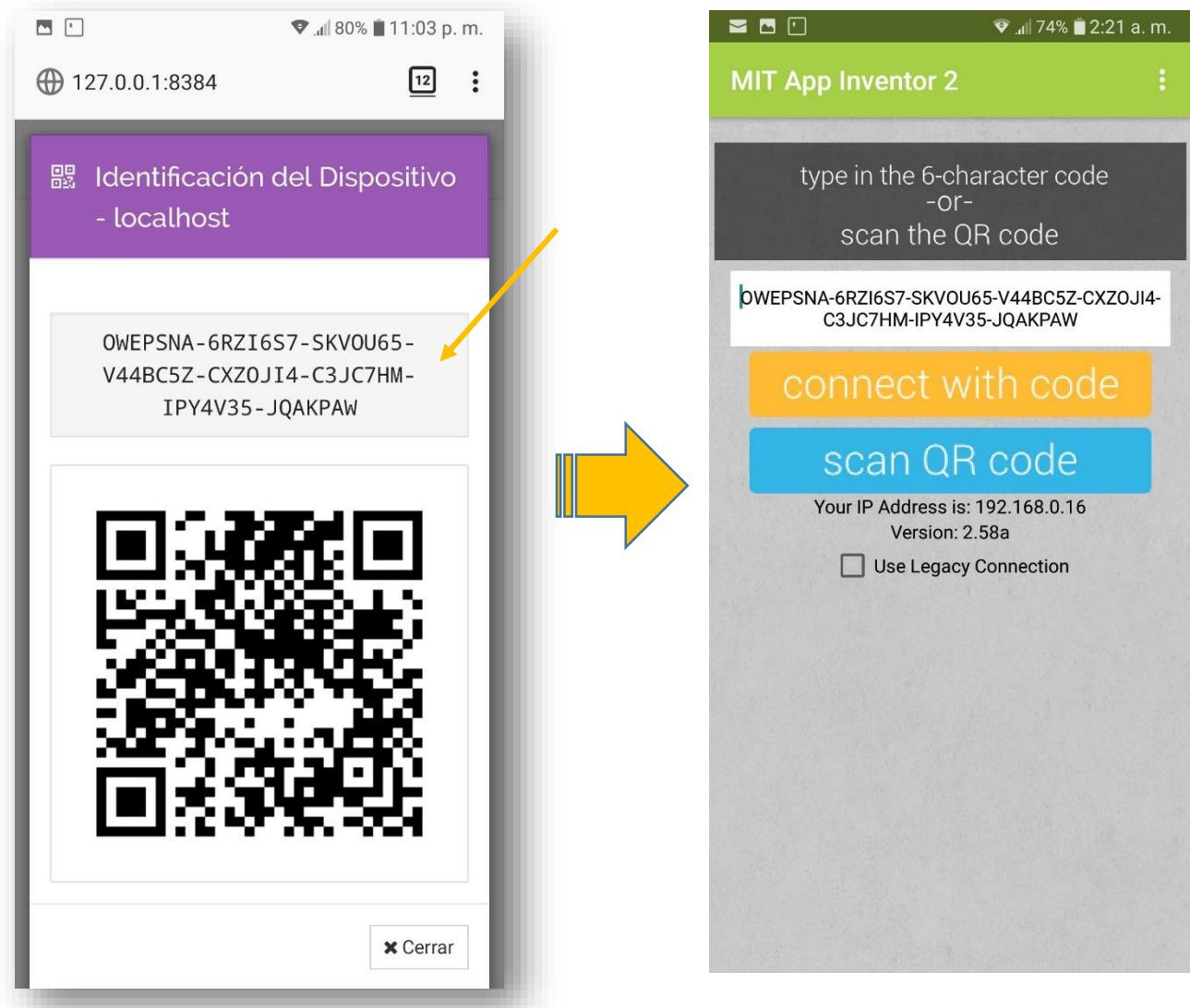


Wenn Sie auf "ID anzeigen" klicken, erscheint der folgende Bildschirm, der einen QR-Code des Samsung-Telefons darstellt. In unserem Fall ist die ID, die das Telefon identifiziert

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

Im LG Q6-Handy ist das Programm, das uns bei der Erfassung des QR-Codes dieser Samsung helfen wird, dasjenige, das wir aus der Anwendung App Inventor (optional) vorschlagen. Falls wir es jedoch nicht haben, können wir es auch einfach manuell einführen, wenn wir die Registrierung im LG Q6 starten, um Fehler zu vermeiden, schlagen wir vor, es zu verwenden.

Mit Hilfe von App program inventor, das auf dem LG Q6 Mobiltelefon installiert ist, erfassen wir QR-Code von dem entfernten Samsung-Telefon, das wir synchronisieren wollen.



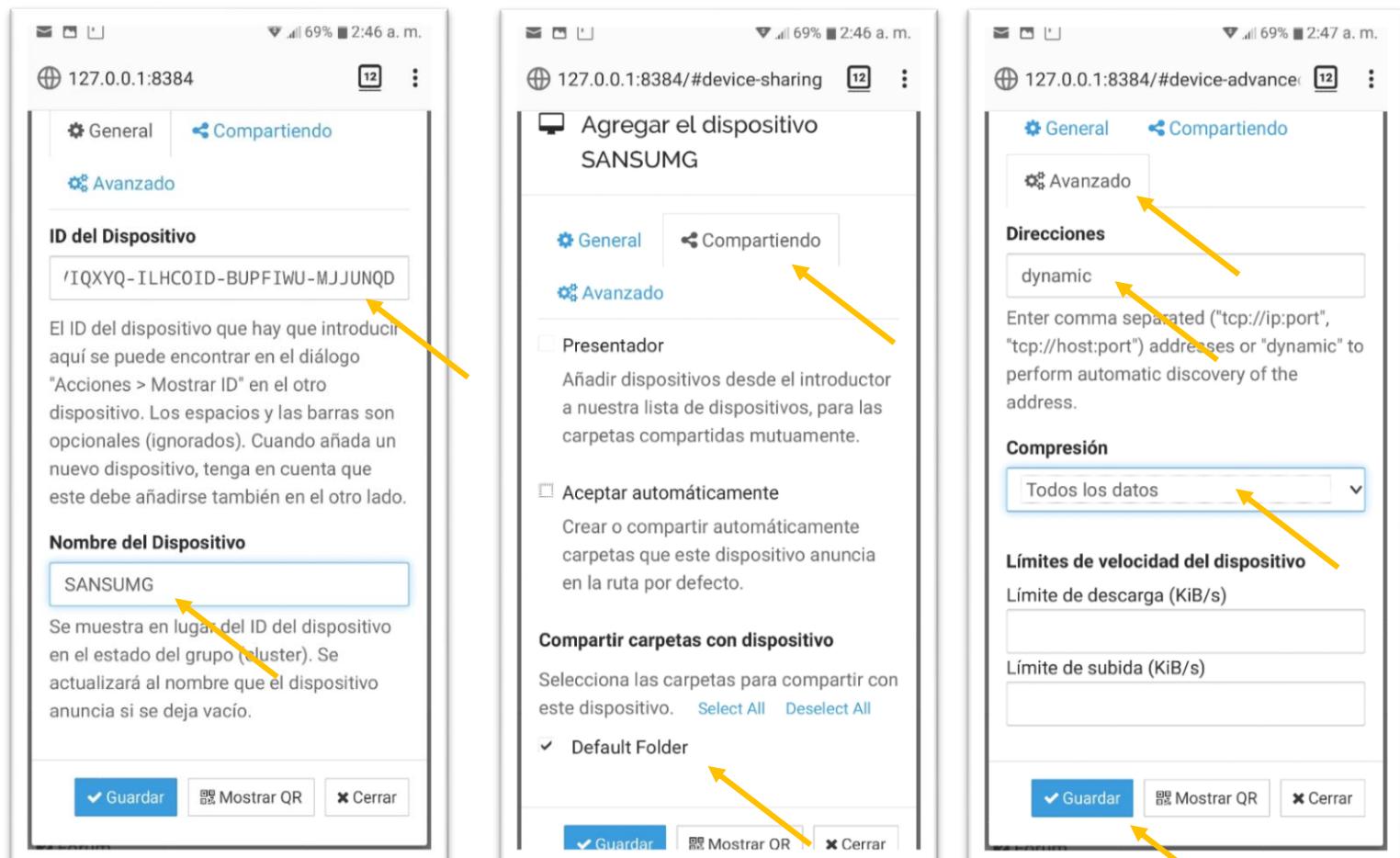
Dann kopieren wir den QR-Code im App inventor Programm in die Zwischenablage, indem wir auf den erfassten Code klicken, wir wählen "SELECT ALL" → "COPY".

Mit diesen Informationen fahren wir fort, die Sansumg in der LG Q6 zu registrieren. Im Administrationsbildschirm gehen wir in den unteren Teil, wo "Andere Geräte" steht, und klicken auf die Schaltfläche "Ein Gerät hinzufügen", wir geben die ID der Sansumg ein und geben ihr einen Registrierungsnamen.

Dann klicken wir im oberen Reiter "Freigabe" und markieren darin die untere Option im Ordner "Standard". Damit geben wir ein Verzeichnis, das von Standard erstellt wurde, in unserem Gerät unter dem Namen Sync frei.

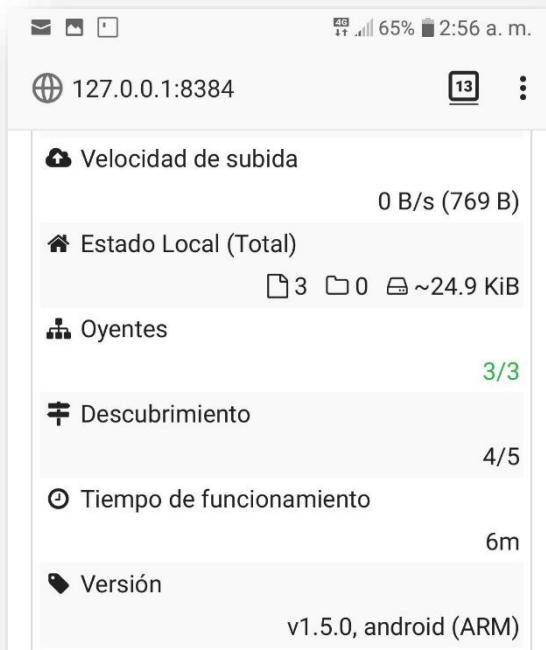
Dann klicken wir oben auf die Registerkarte "Erweitert" und wählen unter "Alle Daten" die Option "Datenkomprimierung".

Zum Schluss klicken wir auf den unteren Speichern-Button, wir beenden die Registrierung in einem Knoten, der gleiche Prozess muss im entfernten Telefon oder in den entfernten Telefonen, die wir synchronisieren wollen, durchgeführt werden.



Beim Speichern und Registrieren in beiden Telefonen werden wir maximal 30 Sekunden warten, in denen sich die Geräte befinden, und die Verbindung zwischen den Geräten wird als gut erscheinen und eine Bestätigung in grün geben.

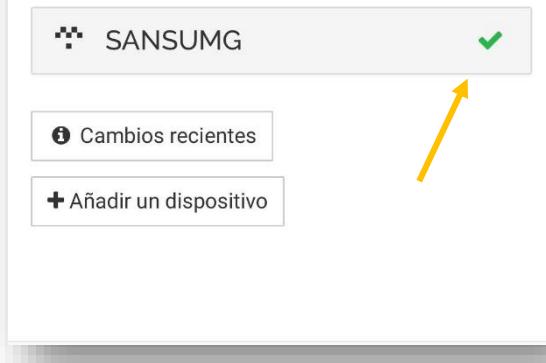
Gerät #1 LG Q6



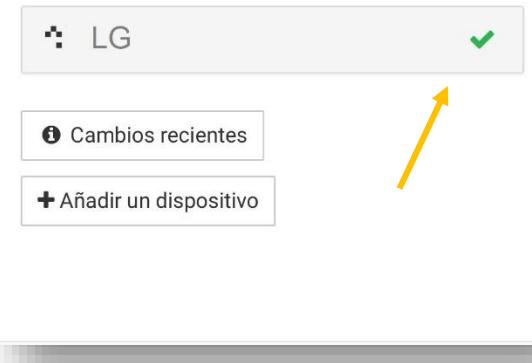
Gerät #2 Samsung



## Otros dispositivos



## Otros dispositivos



Alle Informationen im Sync-Verzeichnis, das sich im Pfad /data/data/com.termux/file/home/Sync befindet, werden synchronisiert, alle Änderungen werden kopiert und synchronisiert.

## Peer-to-Peer-Systemkonfiguration mit automatischer Synchronisierung zur Verwendung in Mini BlocklyChain.

Jetzt werden wir die Konfiguration auf automatische Weise vornehmen, vorher haben wir den manuellen Prozess gemacht, dies ist nützlich, wenn wir eine minimale Anzahl von Knoten handhaben, aber im Falle einer großen Anzahl von Knoten wäre es ineffizient, so dass wir später das SyncThingManager-Werkzeug haben werden, um es auf automatische Weise mit automatisierten Online-Befehlen zu konfigurieren.

### Redis-Datenbankkonfiguration für Knoten im (Slave-)Modus.

Konfiguration der Datei **/data/data/com.termux/files/usr/etc/redis.conf** der Datenbank Redis (**Slave**) für Android-Mobiltelefone.

Fügen Sie die folgenden Änderungen oder Direktiven in die Datei ein, speichern Sie die Änderungen und starten Sie den Redis-Server.

Suchen Sie zunächst das Zeichen # und entfernen Sie es aus der **Slaveof-Zeile**. Diese Direktive nimmt die IP-Adresse und den Port, die Sie zur sicheren Kontaktaufnahme mit dem Redis-Masterserver verwenden, durch ein Leerzeichen getrennt. Standardmäßig lauscht der Redis-Server unter 6379 auf der lokalen Schnittstelle, aber jede der Netzwerksicherheitsmethoden ändert die Voreinstellung in irgendeiner Weise für andere.

Welche Werte Sie verwenden, hängt von der Methode ab, die Sie zum Schutz Ihres Netzwerkverkehrs verwendet haben:

Isoliertes Netzwerk: Verwenden Sie die IP-Adresse des isolierten Netzwerks und den Redis-Port (6379) des Master-Servers (z.B. Slave mit einfacher IP-Adresse 6379).

stunnel oder spiped: Verwenden Sie die lokale Schnittstelle (127.0.0.1) und den konfigurierten Port, um den Verkehr zu tunneln

PeerVPN: Verwenden Sie die IP VPN-Adresse des Master-Servers und den normalen Redis-Port.

Der General würde es ändern:

**slaveof ip\_contact\_server master\_contact\_port**

Beispiel: **Sklave von** 192.168.1.69 6379

**masterauth your\_network\_master\_password**

Beispiel: **masterauth** sdfssdfsdf12WqE34Rfgthtdfd

**requirepass your\_network\_slave\_password**

Beispiel: **requirepass** asdsjdsh34sds67sdFGbbnh

Speichern und starten Sie den Server vom Termux-Terminal aus mit dem folgenden Befehl.

**\$ redis-server redis.conf**

Nach der Ausführung können wir sehen, wie er sich mit dem Windows 10 Server (**Master**) synchronisiert.

Konfigurationsdatei redis.conf **\$ redis-server redis.conf**

The image consists of two side-by-side screenshots of the Termux terminal on an Android device. Both screenshots show the same command being run: `$ redis-server redis.conf`.  
 In the left screenshot, the user has navigated to the directory `/data/data/com.termux/files/usr/etc` using the command `pwd`, and then listed the files in that directory with `ls`. The file `redis.conf` is visible among other configuration files like `inputrc`, `krb5.conf`, `ssh`, `motd`, `tls`, `tmux.conf`, `profile`, `profile.d`, and `wgetrc`. A yellow arrow points from the terminal window to the file `redis.conf` in the list.  
 In the right screenshot, the output of the `redis-server` command is shown. It starts with the server initializing and loading RDB produced by version 6.0.1. It then connects to a master at 192.168.1.69:5379 and begins a replication sync. The log shows various events such as non-blocking connects, master replies to PING, and partial resynchronization attempts. Finally, it completes the sync successfully. A yellow arrow points from the terminal window to the end of the log message "Finished with success".

```
$ 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 192.168.1.69:5379
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: 8ea52fe3c02ae241292f0dcbb823b8feb09b784:0
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
```

Installation eines Syncthing-Verwaltungstools für Knoten. Wir fahren mit der Installation von **SyncthingManager** fort.

Zuerst installieren wir, was Sie benötigen, damit SyncthingManager richtig funktioniert.

**\$ apt Python installieren**

**\$ pip3 installieren -upgrade pip**

**\$ npm Syncthingmanager installieren**

Das Tool SyncthingManager wird uns helfen, "Peer-to-Peer" automatisch und nicht manuell für neue und bestehende Knoten zu synchronisieren.

```
$ 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
kes (20.1.0)
$ 
```

```
$ 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/data/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/data/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/data/com.termux/files/usr/lib/python3.8/
site-packages (from syncthing->syncthingmanager
) (2.18.4)
Requirement already satisfied: six>=1.5 in /data
/data/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/data/com.termux/files/usr/lib/pyth
on3.8/site-packages (from requests==2.18.4->sync
thing->syncthingmanager) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in
/data/data/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/data/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/data/com.termux/files/usr/lib/pyth
on3.8/site-packages (from requests==2.18.4->sync
thing->syncthingmanager) (1.22)
$ 
```

## 17. Ambientes Blockly (App Inventor, AppyBuilder und Thunkable).

App Inventor ist eine Software-Entwicklungsumgebung, die von Google Labs erstellt wurde, um Anwendungen für das Android-Betriebssystem zu erstellen. Der Benutzer kann visuell und aus einer Reihe von Basiswerkzeugen eine Reihe von Blöcken miteinander verknüpfen, um die Anwendung zu erstellen. Das System ist kostenlos und kann einfach aus dem Internet heruntergeladen werden. Mit App Inventor erstellte Anwendungen sind sehr einfach zu erstellen, da keine Kenntnisse einer Programmiersprache erforderlich sind.

Alle aktuellen Umgebungen, die die Technologie von Blockly nutzen, wie z.B. AppyBuilder und Thunkable, haben ihre kostenlose Version, ihre Nutzung kann über das Internet in ihren verschiedenen Sites erfolgen oder auch zu Hause installiert werden.

Die Blöcke, aus denen sich die Mini BloclyChain-Architektur zusammensetzt, wurden in App inventor und AppyBuilder getestet, aber aufgrund ihrer Code-Optimierung sollten sie auf den anderen Plattformen funktionieren.

Online-Versionen:

App-Erfinder.

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

AppyBuilder.

<http://appybuilder.com/>

Denkbar.

<https://thunkable.com/>

Version, die auf Ihrem Computer (PC) installiert werden soll:

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

Umgebung für Entwickler von Blockly-Blöcken.

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

## 18. Was ist der Quantennachweis (PQu)?

PoQu. - "Proof of Quantum" ist ein Konsens-Algorithmus, der für Mini BlocklyChain entwickelt wurde. Dieser Test ist eine Variante des Test of Work (PoW), der wie folgt funktioniert.

Der Test of Quantum (PoQu) wird beim Start mit demselben Algorithmus ausgeführt wie der "Test of Work" (PoW), der darauf basiert, den Prozessor des Geräts (PC, Server, Tablet oder Mobiltelefon) in Gang zu setzen, um eine Zeichenfolge zu erhalten, die ein mathematisches Rätsel namens "Hash" darstellt.

Denken Sie daran, dass ein "Hash" ein Algorithmus oder ein mathematischer Prozess ist, der uns beim Einfügen einer Phrase oder einer Art digitaler Information wie Textdateien, Programme, Bilder, Videos, Töne oder anderer unterschiedlicher digitaler Informationen als Ergebnis ein alphanumerisches Zeichen gibt, das die digitale Signatur repräsentiert, die sie in einer einzigartigen und nicht wiederholbaren Weise der Daten darstellt, der Hash-Algorithmus ist unidirektional, d.h. wenn Sie Daten eingeben, um die Signatur "Hash" zu erhalten, kann der umgekehrte Prozess nicht durchgeführt werden, da wir bei einer Signatur "Hash" nicht wissen können, welche Informationen erhalten wurden. Diese Eigenschaft gibt uns einen Sicherheitsvorteil bei der Verarbeitung der Informationen, die wir über das Internet senden. Wie funktioniert das? Stellen Sie sich vor, Sie senden jede Art von Information über unsichere Kanäle und begleiten sie mit ihrem jeweiligen "Quell-Hash", der Empfänger kann beim Empfang der Information den "Hash" der empfangenen Information erhalten, wir nennen ihn "Ziel-Hash" und überprüfen ihn mit dem "Quell-Hash", wenn beide "Hashes" gleich sind, können wir bestätigen, dass die Information in dem Kanal, der gesendet wurde, nicht verändert wurde, ist nur ein Beispiel dafür, wo diese Art von Informationssicherheitsverfahren derzeit verwendet wird.

Gegenwärtig gibt es verschiedene Arten von Algorithmen oder Hash-Verfahren, die sich in der Sicherheitsstufe unterscheiden. Die am häufigsten verwendeten oder bekannten sind: MD5, SHA256 und SHA512.

Beispiel für SHA256:

Wir haben eine Kette oder einen Satz wie folgt: "Mini BlocklyChain ist modular aufgebaut.

Wenn wir einen SHA256-Hash auf die vorherige Zeichenfolge anwenden, erhalten wir den nächsten Hash.

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2afd827e8804db8**

Die obige alphanumerische Zeichenfolge ist die Signatur, die den Satz im obigen Beispiel darstellt

Für weitere Beispiele können wir die Website im Internet nutzen:

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

Im Falle des "Test Work" (PoW)-Algorithmus arbeitet er mit Rechenleistung, um einen vordefinierten Hash zu erhalten.

Stellen wir uns vor, wir hätten den vorherigen "Hash", den wir von der Kette "Mini BlocklyChain ist modular" übernommen haben.

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2af827e8804db8**

Zu diesem "Hash" am Anfang setzen wir den Parameter der Schwierigkeit, der einfach darin besteht, Nullen "0" an den Anfang zu setzen, d.h. wenn wir sagen, dass die Schwierigkeit bei 4 liegt, wird sie "**0000**" + "Hash" haben, dann nennen wir sie "Samen-Hash".

**0000 f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db8**

Wenn wir nun berücksichtigen, dass wir die Eingabeinformation kennen, die die Zeichenkette ist: "Mini BlocklyChain is modular", fügen wir am Ende der Zeichenkette eine Zahl beginnend bei Null "0" hinzu und nehmen den Hash heraus, den wir "hash nonce" nennen:

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db80**

Wir haben Haschisch nonce:

**7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8**

Dann führen wir einen Vergleich der neuen "hash nonce" mit den "hash seed" durch, wenn sie gleich sind, gewinnt der Knoten, der zuerst die Gleichheit findet, die Ausführung der Verarbeitung der aktuellen Transaktion. Wie wir sehen können, basiert dieser Prozess auf der Wahrscheinlichkeit und der Rechenstärke des Geräts, das dem "Proof of Work"-Test eine Konsensgerechtigkeit für alle Knotenpunkte verleiht.

Wenn der "Samen-Hash" nicht mit dem "Hash-Nonce" übereinstimmt, wird die Schwierigkeit um eins erhöht und der "Hash-Nonce" wieder entfernt, die Zahl, die erhöht wird, wird als "Nonce"-Zahl bezeichnet, sie wird mit dem "Samen-Hash" verglichen, bis sie übereinstimmen oder gleich sind.

Wie wir sehen können, ist die Zahl "nonce" oder Erhöhung diejenige, die dazu beiträgt, den "Hash" der Gleichheit zu erreichen.

Basierend auf dem "Test of Work" (PoW)-Algorithmus basiert der "Test of Quantum" (PoQu)-Algorithmus darauf, wie beim PoW die Zahl "nonce" zu erhalten und unter Verwendung eines

minimalen Schwierigkeitsgrades von 1 bis 5 dient dies nur dem mobilen Gerät, um das Recht zu erlangen, ein Kandidat für den Konsens zu sein.

Der Quantentest (PoQu), wird aktiviert, wenn das Mobiltelefon den minimalen PoW-Wert erreicht hat und den Pass gewinnt, um eine Wahrscheinlichkeitszahl im QRNG-System zu erhalten.

Der QRNG (Quantum Random Number Generator) ist ein Quantenzufallszahlengenerator. Dieses System basiert auf der Erzeugung echter Zufallszahlen auf der Grundlage der Quantenmechanik und ist heute das sicherste System zur Erzeugung solcher Zahlen. Für weitere Details siehe Anhang "Quantenberechnung mit OpenQbit".

Mini BlocklyChain kann sowohl minimale PoW- als auch PoQu-Konzessionstypen implementieren.

Der PoQu-Test basiert auf dem Erhalten der Zahl "nonce" diese Zahl im PoQu-Test ist bekannt als "Magic Number" damit wird das "Peer to Peer"-System bestätigen, ob die Zahl korrekt ist und dann wird eine Zufallszahl mit dem QRNG-Serverpool erhalten. Diese Zufallszahl wird in allen Knoten registriert, es wird eine Liste erstellt, die **((Knotensumme /2)) +1** enthält, und aus dieser Liste wird derjenige mit der höchsten prozentualen Wahrscheinlichkeit, der Gewinnerkandidat des Konsens (PoQu) zu sein, ausgewählt und dieser führt die aktuelle Transaktionswarteschlange aus.

Der PoQu-Algorithmus verwendet auch Tests **des NIST** (National Institute of Standards and Technology), um uns zu versichern, dass die Zufallszahlen im QRNG wirklich Zufallszahlen sind.

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

In Mini BlocklyChain haben wir einen Block für PoW und einen Block für PoQu implementiert.

Diese Blöcke verwenden einen Hash-Typ: SHA256 für die freie Nutzung, für die kommerzielle Nutzung haben Sie einen SHA512 und andere Hash-Typen nach Bedarf.

Weitere Einzelheiten zum Konzept von HASH siehe:

[https://es.wikipedia.org/wiki/Funcion\\_hash](https://es.wikipedia.org/wiki/Funcion_hash)

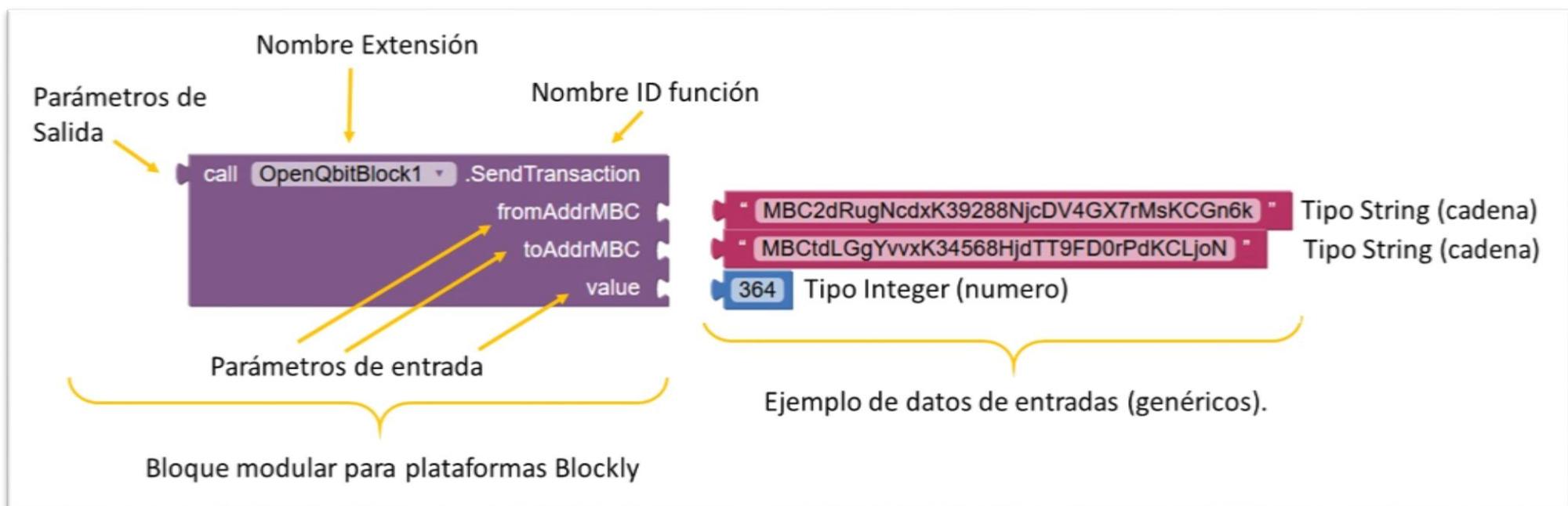
HINWEIS: Der in Mobiltelefonen verwendete Test of Work (PoW) kann nur einen Schwierigkeitsgrad von maximal 5 verwenden, da die mathematische Verarbeitung dieser Geräte nicht dediziert wie bei Servern oder PCs erfolgt. Wir verwenden den PoW-Algorithmus nur, um die Gelegenheit zu erhalten, Ihren Pass oder Ihre Erlaubnis zum Eintritt in das System des Quantenzufallszahlengenerators (QRNG) und damit zur Ausführung des Quantenzufallszahlengenerators (PoQu) zu erhalten.

Verwenden Sie bei Mobiltelefonen keinen Schwierigkeitsgrad von maximal 5, da sich das System blockieren und nicht richtig reagieren könnte.

## 19. Definition und Verwendung von Blöcken in Mini BlocklyChain

Wir beginnen damit, die Verteilung der Daten, die alle Blöcke haben werden, ihre Syntax der Verwendung und Konfiguration zu erklären.

Im folgenden Beispiel sehen wir einen modularen Block und seine Ein- und Ausgabeparameter sowie die Arten von Eingabedaten, diese Daten können vom Typ String (Zeichenfolge) oder Integer (ganzzahlig oder dezimal) sein. Wir zeigen, wie sie benutzt wird, und konfigurieren sie für ihr einwandfreies Funktionieren.



Jeder Modulblock hat seine eigene Beschreibung und wird benannt, falls er eine obligatorische oder optionale Abhängigkeit(en) von anderen Blöcken als Eingabeparameter hat, wird der Integrationsprozess angekündigt.

Block zur Erstellung einer temporären Zeichenkettenliste in einem vordefinierten Array, das intern "Kette" genannt wird - (**AddHash**).

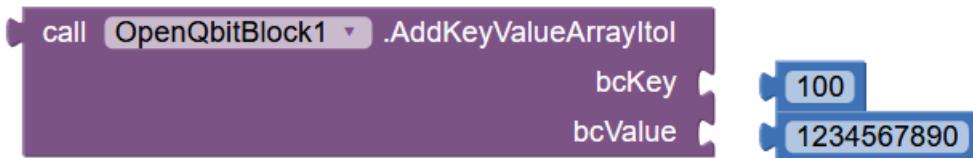


Eingabeparameter: **Block** <String>

Ausgabeparameter: Nicht zutreffend.

Beschreibung: Block zum Speichern eines temporären Arrays von Hashes oder Zeichenketten in einem vordefinierten Array, das intern "Kette" genannt wird.

Block zum Erstellen von Schlüssel-Wert-Arrays (**Ganzzahl-Ganzzahl**) - (**AddKeyValueArrayItol**)

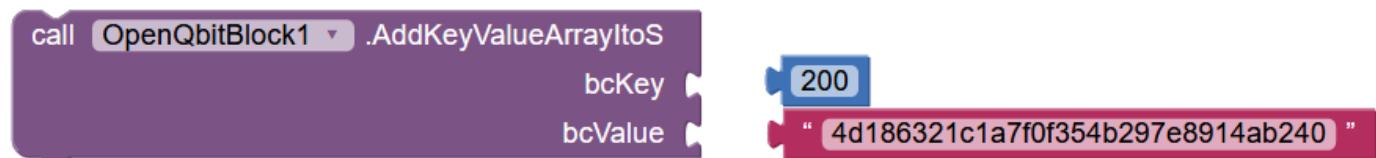


Eingabeparameter: **bcKey** <Ganzzahl>, **bcValue** <Ganzzahl>

Ausgabeparameter: Gibt die bei der Eingabe eingegebenen Werte zurück.

Beschreibung: Es handelt sich um eine temporäre Schlüssel-Wert-Anordnung, sie ist mit dem internen Namen "**Itol\_UTXO**" vordefiniert, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block zum Erstellen von Schlüssel-Wert-Arrays (**Ganzzahl-String**) - (**AddKeyValueArrayItos**)

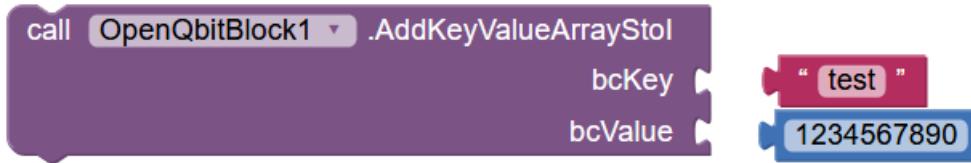


Eingabeparameter: **bcKey** <Integer>, **bcValue** < String>

Ausgabeparameter: Gibt die bei der Eingabe eingegebenen Werte zurück.

Beschreibung: Es handelt sich um eine temporäre Schlüssel-Wert-Anordnung, sie ist mit dem internen Namen "**Itos\_UTXO**" vordefiniert, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block zum Erstellen von Schlüssel-Wert-Arrays (**String-Integer**) - (**AddKeyValueArrayStoI**)



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

Ausgabeparameter: Gibt die bei der Eingabe eingegebenen Werte zurück.

Beschreibung: Es handelt sich um eine temporäre Schlüssel-Wert-Anordnung, sie ist mit dem internen Namen "**StoI\_UTXO**" vordefiniert, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block zum Erstellen von Schlüssel-Wert-Arrays (**String-String**) - (**AddKeyValueArrayStoS**)

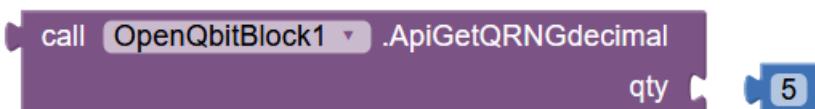


Eingabeparameter: **bcKey** <Zeichenkette>, **bcValue** <Zeichenkette>

Ausgabeparameter: Gibt die bei der Eingabe eingegebenen Werte zurück.

Beschreibung: Es handelt sich um eine temporäre Schlüssel-Wert-Anordnung, sie ist mit dem internen Namen "**StoS\_UTXO**" vordefiniert, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block zur Erzeugung von dezimalen Zufallsquantenzahlen - (**ApiGetQRNGdecimal**)



Eingabeparameter: **Menge** < Ganzzahl>

Ausgabeparameter: Gibt die Menge "qty" von zufälligen Quanten-Dezimalzahlen an, die in die Eingabezahlen eingegeben werden, die im Bereich von 0 und 1 im JSON-Format liegen.

Beispiel:

Anzahl = 5; Ausgabe: {"Ergebnis": [0,5843012986202495, 0,7746497687824652, 0,05951126805960929, 0,1986079079055812694, 0,0368978343989999279]}

Beschreibung: Quantum Random Number Generator (QRNG) API

Block zur Erzeugung von dezimalen Zufallsquantenzahlen - (**ApiGetQRNGdecimal**)



Eingabeparameter: **Menge <Ganzzahl>**, min <Ganzzahl>, max <Max>

Ausgabeparameter: Gibt die Menge "qty" zufälliger Quantenganzzahlen an, die in die Eingabe eingegeben wurden, wobei die Zahlen im Bereich von min und max im JSON-Format liegen.

Beispiel:

Anzahl = 8, min = 1, max = 100; Ausgabe: {"Ergebnis": [3, 53, 11, 2, 66, 44, 9, 78]}

Beschreibung: Quantum Random Number Generator (QRNG) API

Hash-Block "SHA256" für Zeichenketten (**Anwenden vonSha256**).



Eingabeparameter: **Eingabe <Zeichenkette>**

Ausgabeparameter: Berechnet den "SHA256"-Hash einer Zeichenkette.

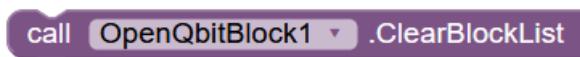
Beispiel:

Eingabe = "Mini BlocklyChain ist modular"

Ausgabe: f41af7e61c3b02fdd5e5c612302b62a2dd52fc38f9de97cb2af827e8804db8

Beschreibung: Funktion zum Entfernen des Rautezeichens "SHA256". Sha oder SHA beziehen sich auf: Sicherer Hash-Algorithmus, eine Reihe von Hash-Funktionen, die von der Nationalen Sicherheitsbehörde der Vereinigten Staaten entwickelt wurden. Der SHA256 verwendet einen 256-Bit-Algorithmus.

Block zum Bereinigen der vordefinierten internen Array-"Kette" (**ClearBlockList**).



Eingabe- und Ausgabeparameter: Nicht zutreffend

Beschreibung: Löschen Sie alle Elemente, die die interne zeitliche Anordnung "Kette" haben.

Block zum Bereinigen des vordefinierten internen Arrays (**Integer-Integer**) - (**ClearItol**).

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

Eingabe- und Ausgabeparameter: Nicht zutreffend

Beschreibung: Löschen Sie alle Elemente, die die interne temporäre Anordnung "Itol\_UTXO" haben.

Block zur Bereinigung des vordefinierten internen Arrays (**Integer-String**) - (**ClearItoS**).

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

Eingabe- und Ausgabeparameter: Nicht zutreffend

Beschreibung: Löschen Sie alle Elemente, die die interne temporäre Anordnung "ItoS\_UTXO" haben.

Block zur Bereinigung des vordefinierten internen Arrays (**String-Integer**) - (**ClearStol**).

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

Eingabe- und Ausgabeparameter: Nicht zutreffend

Beschreibung: Löschen Sie alle Elemente, die die interne temporäre Anordnung "Stol\_UTXO" haben.

Block zur Bereinigung des vordefinierten internen Arrays (**String-String**) - (**ClearStoS**).

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

Eingabe- und Ausgabeparameter: Nicht zutreffend

Beschreibung: Löschen Sie alle Elemente, die die interne temporäre Anordnung "StoS\_UTXO" haben.

Block zum Dekodieren einer Zeichenfolge nach Base10. (**DecodeBase58**)

call **OpenQbitBlock1** .**DecodeBase58**

input

" oBRN8Aj67yJsbRGHfNSF9PTdZYGyVWrwMW7mTX "

Eingabeparameter: **Eingabe<Zeichenkette>**

Ausgabeparameter: Es wird die ursprüngliche Zeichenfolge geliefert, die im Block verwendet wurde (**EncodeBase58**).

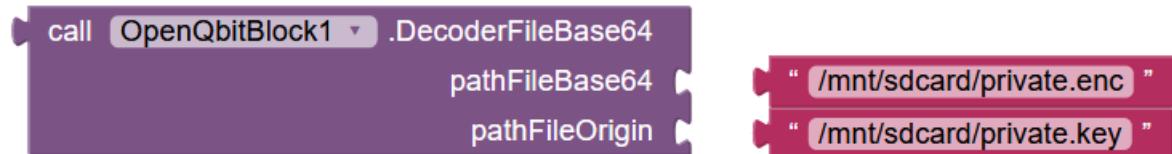
Beispiel:

Eingabe = oBRN8Aj67undJsbRGHfNSF9PTdZYGundVWrwMW7mTX

Ausgabe: "Mini BlocklyChain ist modular".

Beschreibung: Konvertiert eine Base58-Zeichenfolge in den im Block angegebenen Originaltext (**EncodeBase58**)

Block zum Dekodieren einer Datei mit Base64-Algorithmus (**DecoderFileBase64**).



Eingabeparameter: **pathFileBase64 <Zeichenkette>**, **pathFileOrigin <Zeichenkette>**

Ausgabeparameter: Quelldatei, die in den Block eingegeben wurde (**EncoderFileBase64**)

Beschreibung: Eine Base64-Datei wird in die Originaldatei konvertiert, die in den Block eingefügt wurde (**EncoderFileBase64**).

Block, um zu wissen, ob das vordefinierte interne Array "chain" leer ist. (**LeereBlockListe**)



Eingabeparameter: Nicht zutreffend.

Ausgabeparameter: Gibt "Wahr" zurück, wenn leer, oder "Falsch", wenn Sie Daten haben.

Beschreibung: Block zur Abfrage, ob die vordefinierte temporäre interne Array "Kette" Elemente hat.

Block zur Kodierung einer Zeichenkette zur Base58. (**EncodeBase58**).



Eingabeparameter: **Eingabe<Zeichenkette>**

Ausgabeparameter: Es wird die ursprüngliche Zeichenfolge geliefert, die im Block verwendet wurde (**EncodeBase58**).

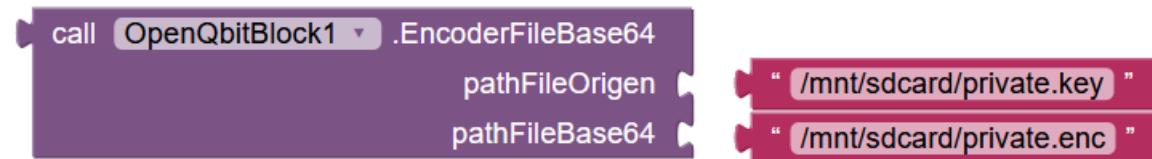
Beispiel:

Eingabe = "Mini BlocklyChain ist modular".

Output: oBRN8Aj67yJsbRGHfNSF9PTdZYGyVWrwMW7mTX

Beschreibung: Konvertiert eine Brustkette in eine Kette in Bae58. Der Base58-Algorithmus ist eine Gruppe von Binär-zu-Text-Kodierungsschemata zur Darstellung großer Ganzzahlen als alphanumerischer Text, die von Satoshi Nakamoto für die Verwendung mit Bitcoin eingeführt wurde.

Block zur Kodierung einer Datei mit dem Base64-Algorithmus (**EncoderFileBase64**).

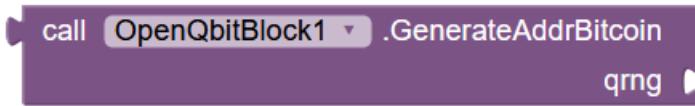


Eingabeparameter: **pathFileOrigin <String>, pathFileBase64 <String>**

Ausgabeparameter: Base64-kodierte Datei.

Beschreibung: Konvertiert eine Quelldatei in einem beliebigen Format in eine Base64-Datei. Die Dateinamen können beliebig sein und vom Benutzer gewählt werden.

Block Generator von Benutzeradressen (**GenerateAddrBitcoin**).



Obligatorische Einheit: Block (**ApiGetQRNGinteger**),

Abhängigkeiten (optional): **OpenQbitFileEncription-Erweiterung**, **OpenQbitFileDecryption-Erweiterung** und **OpenQbitSQLite-Erweiterung**.

Eingabeparameter: **qrng < obligatorische Abhängigkeit>**

Ausgabeparameter: 34-stellige alphanumerische Transaktionsadresse und keyStore-Verwendungsadresse

Beschreibung: Block zum Erstellen einer neuen generischen Bitcoin-Transaktionsadresse für den Benutzer und den Generator für den privaten Schlüssel (Digitale Signatur für das Senden von Transaktionen) und den öffentlichen Schlüssel (Öffentliche Adresse für die Durchführung von Transaktionen). Dieser Schlüsselgenerator ist im Grunde der Adressgenerator zur Verwendung in einer digitalen Brieftasche oder allgemein als "Wallet" bezeichnet.

Dieser Block wird in Verbindung mit den Blöcken des Quantum Random Number Generator (QRNG) verwendet, und die beiden Ausgabeparameter müssen in den KeyStore eingefügt werden.

KeyStore ist eine Datenbank, die private Schlüssel im hexadezimalen Format speichert:

Beispiel für eine hexadezimale Adresse, die im KeyStore gespeichert ist

**024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD**

Diese Basis wird nur vom lokalen Benutzer verwendet und die Informationen werden verschlüsselt. Dieser Prozess erfolgt durch die Verwendung der OpenQbitSQLite-Erweiterung und der OpenQbitAESEncryption- und OpenQbitAESDecryption-Erweiterungen zur Verschlüsselung von Informationen.

Zum Erstellen eines KeyStore siehe Anhang "Erstellen eines KeyStore". Die generierten Adressen verwenden den gleichen Bitcoin-Adressalgorithmus, wobei die anfängliche Bitcoin-Adresskennung "1" ist.

Die vom Block generierten Adressen (**GenerateAddrBitcoin**) sind 34 alphanumerische Zeichen und setzen sich aus 33 alphanumerischen Zeichen und 1 der Bitcoin-Kennung wie folgt zusammen:

**12dRugNcdxK39288NjcDV4GX7rMsKCGn6k**

Benutzeradressen-Generatorblock (**GenerateAddrEthereum**).



Obligatorische Abhängigkeit: Besorgen Sie sich eine Wertmarke unter:  
<https://accounts.blockcypher.com/signup>

Abhängigkeiten (optional): **OpenQbitFileEncription-Erweiterung**, **OpenQbitFileDecryption-Erweiterung** und **OpenQbitSQLite-Erweiterung**.

Eingabeparameter: **Token** < obligatorische Abhängigkeit - String>

Ausgabeparameter: Die Transaktionsadresse mit 40 alphanumerischen Zeichen enthält nicht den anfänglichen Ethereum-Indikator "0x", der uns die 42 Zeichen einer gemeinsamen Adresse geben würde. Es gibt auch den öffentlichen Schlüssel und den privaten Schlüssel zurück.

Beispiel:

```
{"privat": "227ac59f480131272003c2d723a7795ebd3580acaab62b5c537989e2ce4e08ef",
"öffentlich":
"04e2d55ebcccd32a7384e096df559cc36b856c64a16e5b402e10585dc3ea055672aafa84df8
a859531570a650a8ab1e7a22949100efa1aa5f072c035551cac1ce",
"Adresse": "14e150399b0399f787b4d6fe30d8b251375f0d66"}
```

Beschreibung: Block zum Erstellen einer neuen Transaktionsadresse für Benutzer und Generator für private Schlüssel (Digitale Signatur zum Senden von Transaktionen) und öffentliche Schlüssel (Öffentliche Adresse zum Durchführen von Transaktionen). Bei diesem Schlüsselgenerator handelt es sich um eine externe API, und Sie müssen über eine Wifi- oder Mobilverbindung verfügen. Es handelt sich im Grunde genommen um den Adressgenerator zur Verwendung in einer digitalen Brieftasche oder allgemein als "Wallet" bezeichnet.

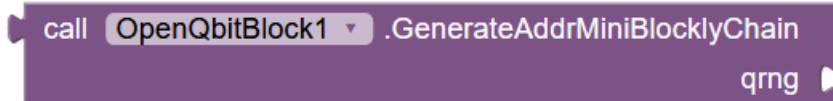
Sie können einen KeyStore erstellen. AKeyStore ist eine Datenbank, die private Schlüssel im Hexadezimalformat speichert, siehe Anhang "KeyStore-Erstellung".

Beispiel für eine hexadezimale Adresse, die im KeyStore gespeichert ist

**0x14e150399b0399f787b4d6fe30d8b251375f0d66**

Der private Schlüssel wird nur vom lokalen Benutzer verwendet und die Informationen werden verschlüsselt. Dieser Prozess erfolgt durch die Verwendung der OpenQbitSQLite-Erweiterung und der OpenQbitAESEncryption- und OpenQbitAESDecryption-Erweiterungen zur Verschlüsselung von Informationen.

Benutzeradressgenerator-Block (**GenerateAddrMiniBlocklyChain**).



Obligatorische Einheit: Block (**ApiGetQRNGinteger**),

Abhängigkeiten (optional): **OpenQbitFileEncription-Erweiterung**, **OpenQbitFileDecryption-Erweiterung** und **OpenQbitSQLite-Erweiterung**.

Eingabeparameter: **qrng < obligatorische Abhängigkeit>**

Ausgabeparameter: Transaktionsadresse mit 36 alphanumerischen Zeichen und keyStore-Verwendungsadresse. Überprüfung der Blockmethode (**PairKeysMBC**).

Beschreibung: Block zum Erstellen einer neuen Transaktionsadresse für Benutzer und Generator für private Schlüssel (Digitale Signatur zum Senden von Transaktionen) und öffentliche Schlüssel (Öffentliche Adresse zum Durchführen von Transaktionen). Dieser Schlüsselgenerator ist im Grunde der Adressgenerator zur Verwendung in einer digitalen Brieftasche oder allgemein als "Wallet" bezeichnet.

Dieser Block wird in Verbindung mit den Blöcken des Quantum Random Number Generator (QRNG) verwendet, und die beiden Ausgabeparameter müssen in den KeyStore eingefügt werden.

KeyStore ist eine Datenbank, die private Schlüssel im hexadezimalen Format speichert:

Beispiel für eine hexadezimale Adresse, die im KeyStore gespeichert ist

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

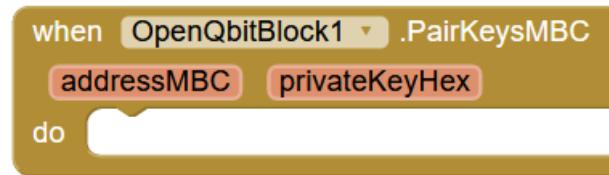
Diese Basis wird nur vom lokalen Benutzer verwendet und die Informationen werden verschlüsselt. Dieser Prozess erfolgt durch die Verwendung der OpenQbitSQLite-Erweiterung und der OpenQbitAESEncryption- und OpenQbitAESDecryption-Erweiterungen zur Verschlüsselung von Informationen.

Zum Erstellen eines KeyStore siehe Anhang "Erstellen eines KeyStore". Die generierten Adressen verwenden den gleichen Bitcoin-Adressen-Algorithmus, der einzige Unterschied besteht darin, dass die Bitcoin-Adresskennung, die "1" oder "3" ist, in die Mini BlocklyChain-Kennung "MBC" geändert wird.

Die vom Block generierten Adressen (**GenerateAddrMiniBlocklyChain**) bestehen aus 36 alphanumerischen Zeichen und setzen sich aus 33 alphanumerischen Zeichen und 3 Großbuchstaben der Mini BlocklyChain-Kennung "MBC" wie folgt zusammen:

**Mbc2dRugNcdxK39288NjcDV4GX7rMsKCGn6k**

Blockmethode (**PairKeysMBC**) ist Blockausgabe (**GenerateAddrMiniBlocklyChain**).



Eingabeparameter: Nicht zutreffend.

Ausgabeparameter: **AdresseMBC** <Zeichenkette>, **privateKeyHex** <Zeichenkette>.

Beschreibung: Die öffentliche Benutzeradresse wurde im Mini BlocklyChain-Format und der private Schlüssel im Hexadezimalformat geliefert.

Beispiel:

**AdresseMBC**: MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

**privatKeyHex**:

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

Block Generator von Benutzeradressen (**GenerateKeyValuePair**).



Obligatorische Einheit: Block (**ApiGetQRNGinteger**),

Abhängigkeiten (optional): **OpenQbitFileEncription-Erweiterung**, **OpenQbitFileDecryption-Erweiterung** und **OpenQbitSQLite-Erweiterung**.

Eingabeparameter: **qrng** < obligatorische Abhängigkeit>

Ausgabeparameter: Liefert den öffentlichen Schlüssel und den Primärschlüssel für den lokalen Benutzer

Beschreibung: Generiert öffentliche und Primärschlüssel unter Verwendung des ECC-Algorithmus (1) und mit hexadezimalem Format.

- (1) Die Kryptographie mit elliptischen Kurven (ECC) ist eine Variante der asymmetrischen oder Public-Key-Kryptographie, die auf der Mathematik der elliptischen Kurven basiert.

Block zum Signieren von Transaktionen (**GenerateSignature**).

call OpenQbitBlock1 .GenerateSignature

Erforderliche Abhängigkeit: Block (**GenerateKeyPairs**), Block (**SenderLoadKeyPair**), Block (**RecipientLoadKeyPair**). Stellen Sie diese Blöcke vor der Verwendung vor.

Eingabeparameter: < Obligatorische Prüfungsabhängigkeiten>

Ausgabeparameter: Keine Ausgabe.

Beschreibung: Es erzeugt eine digitale Signatur des Absenders, die auf das Asset angewendet wird, das in der Transaktion an den Empfänger gesendet wird. Diese Signatur wird bestätigt, wenn die Transaktion von irgendeinem Knoten des Netzwerks verarbeitet wird. Mit dieser Signatur stellt das Mini BlocklyChain-System sicher, dass das Asset in der gesendeten Datei nicht verändert wurde und dass es der Sender-Empfänger-Beziehung entspricht und nicht umgeleitet oder auf eine andere digitale Adresse angewendet wird.

Block, um den Gesamtsaldo der Vermögenswerte eines Benutzers zu erhalten (**GetBalance**).

call OpenQbitBlock1 .GetBalance

fromAddrMBC

" MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k "

Obligatorische Abhängigkeit: Block (**ConnectorTransactionTail**).

Eingabeparameter: **fromTransTail** <Array String>, < Obligatorische Boards-Abhängigkeiten>

Ausgabeparameter: Überprüft die ein- und ausgehenden Vermögensaufwendungen für jede Transaktion.

Beschreibung: Prüft den Saldo, um zu genehmigen, ob der Benutzer Assets zu senden hat oder ob die Assets, die er erhält, zu seinem Saldo hinzugefügt werden.

Blockieren, um q22 vom Mobiltelefon zu erhalten. (**GetDeviceID**)

call OpenQbitBlock1 .GetDeviceID

Eingabeparameter: **Nicht zutreffend**

Ausgangsparameter: Es liefert die interne Kennung des Mobiltelefons.

Beschreibung: Stellt die für jedes Gerät eindeutige IMEI-Mobiltelefonkennung bereit.

Block zum Entfernen des RIPEMD-160-Hashes aus einer Zeichenfolge. (**Ripemd160**)



Eingabeparameter: **str <String>**

Ausgabeparameter: Berechnet den "RIPEMD-160"-Hash einer Zeichenkette.

Beispiel:

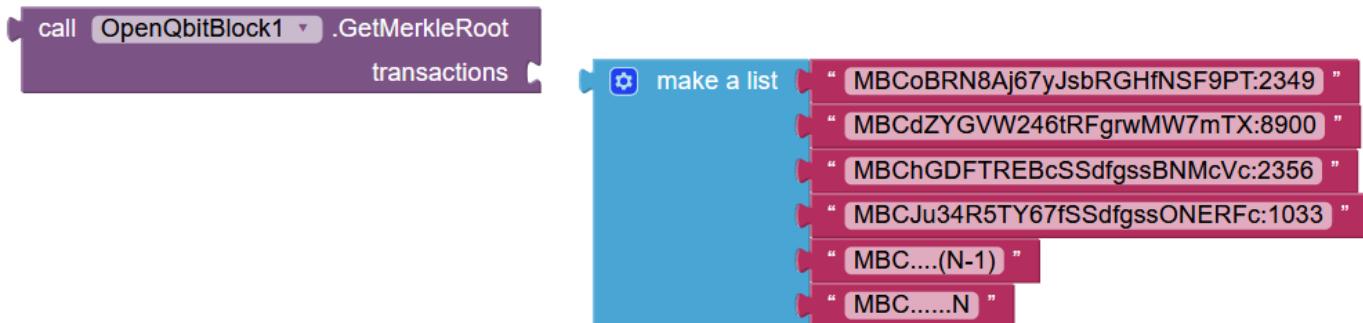
Eingabe = "Mini BlocklyChain ist modular"

Ausgabe: ae29436e4b8ea8ed6143f3f92380dfa2f4f47336

Beschreibung: Erhalten Sie den "RIPEMD-160"-Hash. Dieser Hash wird bei der Generierung einer gültigen Adresse für Bitcoin und Mini BlocklyChain verwendet.

RIPEMD-160 (Akronym für RACE Integrity Primitives Evaluation Message Digest) ist ein 160-Bit Message Digest-Algorithmus (und kryptographische Hash-Funktion).

Block zur Berechnung von Merklers Baum. (**GetMerkleRoot**)



Eingabeparameter: **Transaktionen <Array String>**

Ausgabeparameter: Es wird ein Hashtyp "SHA256" geliefert.

Beispiel:

Eingabe = Transaktionswarteschlange, eine Anordnung aller zu verarbeitenden Transaktionen.

Ausgabe: b4a44c42b6070825f763cd118d6ab49a8e80bbb7cdc0225064f8e042b94196bd

Beschreibung: Erhalten Sie den "SHA256"-Hash mit Merkle's Baum-Algorithmus

Ein Merkle-Hash-Baum ist eine binäre oder nicht-binäre Datenbaumstruktur, in der jeder Knoten, der kein Blatt ist, mit dem Hash der Verkettung der Bezeichnungen oder Werte (für

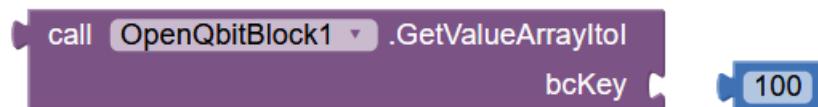
Blattknoten) seiner untergeordneten Knoten versehen ist. Dabei handelt es sich um eine Verallgemeinerung von Hash-Listen und Hash-Strings.

Es ermöglicht die Verknüpfung einer großen Anzahl separater Daten mit einem einzigen Hash-Wert, dem Hash-Wert des Baumwurzelknotens. Dies stellt eine sichere und effiziente Methode zur Verifizierung des Inhalts großer Datenstrukturen dar. In der praktischen Anwendung wird der Hash des Wurzelknotens in der Regel signiert, um seine Integrität zu gewährleisten und sicherzustellen, dass die Verifizierung absolut zuverlässig ist. Der Nachweis, dass ein Blattknoten Teil eines bestimmten Hash-Baums ist, erfordert eine Datenmenge, die proportional zum Logarithmus der Anzahl der Knoten im Baum ist.

Gegenwärtig besteht die Hauptverwendung von Merkle-Bäumen darin, die von anderen Peers in den Peer-to-Peer-Netzwerken empfangenen Datenblöcke sicher zu machen, um sicherzustellen, dass sie ohne Beschädigung und ohne Veränderung empfangen werden.

Sie wird verwendet, um zu überprüfen, ob eine Warteschlange mit den zu verarbeitenden Transaktionen nicht verändert wurde, und um ihre Integrität für die Verteilung in allen Knoten des Mini BlocklyChain-Netzwerks sicherzustellen. Alle Knoten müssen diesen Algorithmus ausführen, um die Integrität jeder Transaktion, die angewendet wird, zu gewährleisten.

Block, um einen Wert des vordefinierten Arrays "Itol\_UTXO" (**GetValueArrayItol**) zu erhalten.

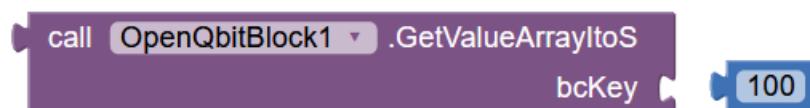


Eingabeparameter: **bcKey** <Integer>

Ausgabeparameter: Gibt den Wert <Integer> zurück, der im Etikett mit der angegebenen Nummer als Eingabe gespeichert ist.

Beschreibung: Es handelt sich um eine Anfrage an die temporäre Schlüssel-Wert-Anordnung, die mit dem internen Namen "Itol\_UTXO" vordefiniert ist, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block, um einen Wert aus dem vordefinierten Array "ItoS\_UTXO" (**GetValueArrayItoS**) zu erhalten.

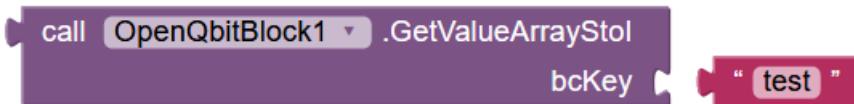


Eingabeparameter: **bcKey** <Integer>

Ausgabeparameter: Gibt den Wert <String> zurück, der im Etikett mit der angegebenen Nummer als Eingabe gespeichert ist.

Beschreibung: Es handelt sich um eine Anfrage an die temporäre Schlüssel-Wert-Anordnung, die mit dem internen Namen "**ItoS\_UTXO**" vordefiniert ist, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block, um einen Wert aus dem vordefinierten Array "StoI\_UTXO" (**GetValueArrayStoI**) zu erhalten.



Eingabeparameter: **bcKey** < String>

Ausgabeparameter: Gibt den Wert <Integer> zurück, der in dem Etikett mit dem gegebenen Namen als Eingabe gespeichert ist.

Beschreibung: Es handelt sich um eine Anfrage an die temporäre Schlüssel-Wert-Anordnung, die mit dem internen Namen "**StoI\_UTXO**" vordefiniert ist, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Block, um einen Wert aus dem vordefinierten Array "StoS\_UTXO" (**GetValueArrayStoS**) zu erhalten.



Eingabeparameter: **bcKey** < String>

Ausgabeparameter: Gibt den Wert <String> zurück, der in dem Etikett mit dem gegebenen Namen als Eingabe gespeichert ist.

Beschreibung: Es handelt sich um eine Anfrage an die temporäre Schlüssel-Wert-Anordnung, die mit dem internen Namen "**StoS\_UTXO**" vordefiniert ist, was für die Verarbeitung von UTXO-Transaktionen nützlich ist.

Blockprüfer der Blockkette. (**IsKetteGültig**)

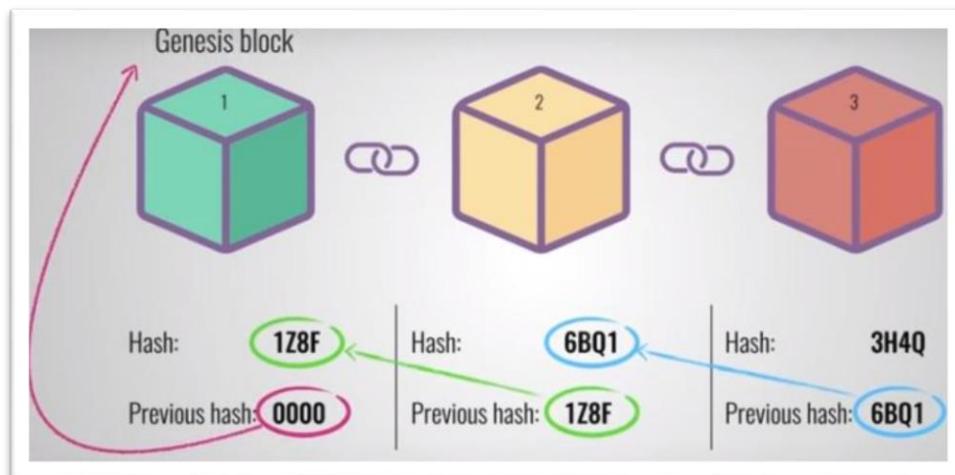
call [OpenQbitBlock1 .IsChainValid]

Erforderliche Abhängigkeit: Block (**GenerateKeyPairs**), Block (**SenderLoadKeyPair**), Block (**RecipientLoadKeyPair**), Block (**GenerateSignature**). Stellen Sie diese Blöcke vor der Verwendung vor.

Eingabeparameter: Nicht zutreffend.

Ausgabeparameter: Lieferung "True", wenn die Validierung der Blockzeichenfolge korrekt ist, oder "False", wenn die Validierung fehlschlägt.

Beschreibung: Es bietet uns die Validierung der Komponenten, die zuvor in das Blockkettensystem eingefügt wurden, diese Validierung ist die wichtigste des gesamten Systems. Wie funktioniert die Blockkettenvalidierung oder wie ist sie allgemein auf generische Weise bekannt (BlockChain). Es ist das Kernstück eines jeden Systems.



Wie wir im vorherigen Bild sehen, ist jeder Block, der im Speichersystem hinzugefügt wird, durch die Hash-Algorithmen mit dem vorherigen Block verbunden.

Wenn Sie beispielsweise einen neuen hinzuzufügenden Block erstellen, wird der Hash, der die neue Information darstellt, aus dem Hash des neuen Informationsblocks + dem vorherigen Hash ermittelt. Mit dieser Art der Verknüpfung wird jede minimale Veränderung in der Speicherung von Blockketten erkannt, was eine sehr hohe und effektive Datensicherheit ermöglicht.

Unten sehen Sie ein Beispiel dafür, wie eine Zeichenfolge von Blöcken in einer SQLite-Datenbank gespeichert wird. Wir werden sehen, wie der vorherige Hash mit dem neuen Hash des letzten Blocks (letzte Warteschlange von verarbeiteten Transaktionen), der eingefügt wurde, verknüpft wird. Jeder Hash in den Spalten "prevhash" und "newhash" stellt die Informationen der Transaktionswarteschlange dar, die zu diesem Zeitpunkt verarbeitet wurde.

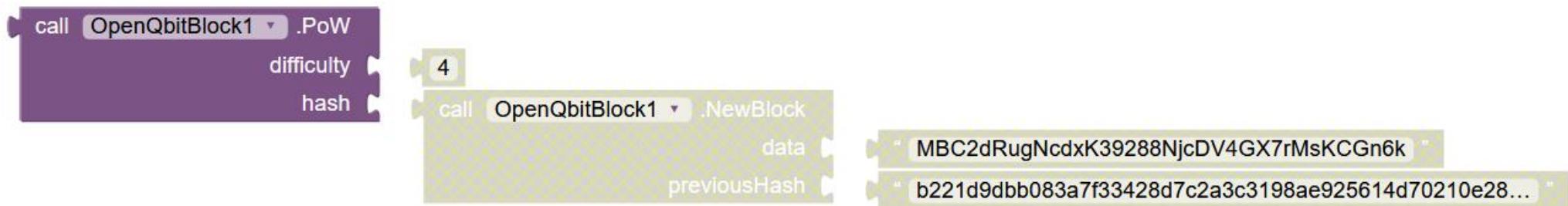
The screenshot shows the SQLite Expert Personal 5.3 interface. The title bar reads "SQLite Expert Personal 5.3 (x64)". The menu bar includes File, View, Database, Object, SQL, Transaction, Tools, and Help. The toolbar contains various icons for database management. The database path is "Database: miniblock Table: nblock File: C:\memo\thunkable\miniblock.db". The left sidebar shows tables: countries, lex, mbmc, MX, nblock (selected), tracking, and US. The main area displays a table with columns: rowid, id, prevhash, newhash, opera, trans, and balan. The table data is:

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

Two yellow arrows point from the "prevhash" column of row 2 to the "newhash" column of row 3, illustrating the connection between blocks.

Bisheriger Hash ist mit dem neuen Hash verwandt.

Block zur Durchführung von PoW Work Test und zum Abbau neuer Blöcke. (**PoW**)



Obligatorische Abhängigkeit: **NewBlock**. Geben Sie diesen Block vor der Verwendung ein.

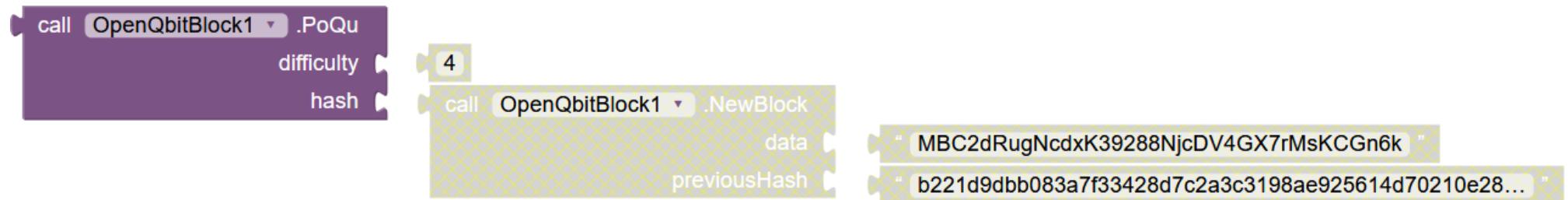
Eingabeparameter: **Schwierigkeit** <Ganzzahl>, Raute <Pflichtabhängigkeit>

Ausgabeparameter: Gibt als Ausgabe einen Hash "SHA256" aus, der mit der im Eingabeparameter Schwierigkeitsgrad angegebenen #Anzahl von "Nullen" zusammengesetzt ist.

Beschreibung: Dieser Block führt eine Aktivität namens "Mining" aus. Ein neuer Block ist das, was wir zuvor als den Prozess des PoW (Proof of Work) beschrieben haben, siehe Abschnitt Was ist ein Quantennachweis (PQu)?

Mining oder das Ausführen des PoW ist ein Konzept, bei dem die Knoten eine Aufgabe zur Lösung eines mathematischen Rätsels erhalten. Wer es im Fall von Mini BlocklyChain zuerst löst, erhält die Möglichkeit, mit dem Prozess fortzufahren, um als Gewinner ausgewählt zu werden, um die Transaktionswarteschlange, die auf die Verarbeitung wartet, bearbeiten zu können.

Block zur Durchführung von PoW Work Test und zum Abbau neuer Blöcke. (**PoQu**)



Obligatorische Abhängigkeit: **NewBlock**. Geben Sie diesen Block vor der Verwendung ein.

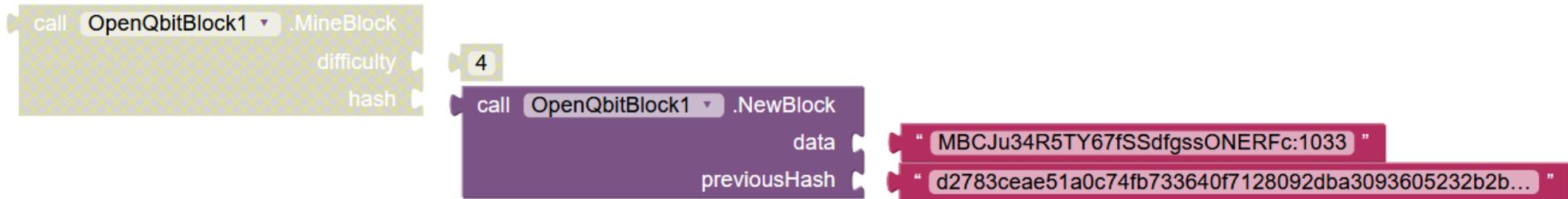
Eingabeparameter: **Schwierigkeit** <Ganzzahl>, Raute <Pflichtabhängigkeit>

Ausgabeparameter: Ergibt die Zahl "Magic Number" und eine zufällige Quantenzahl, die im Bereich von 0 und 1 liegt, 16-stelliger Dezimaltyp, Beispiel (0. 5843012986202495) und einen Hash "SHA256", der mit der im Eingabeparameter Schwierigkeitsgrad angegebenen #Anzahl von "Nullen" zusammengesetzt ist.

Beschreibung: Dieser Block führt den Prozess der "Bergbau" einen neuen Block ist das, was wir zuvor beschrieben haben, wie der Prozess der PoW (Proof of Work), aber dieser Prozess nennt die Funktion der zufälligen Quantenzahl nach der Berechnung der Zahl "nonce" geeignet, um den Schwierigkeitsgrad, die im Bereich von mindestens 1, maximal 5 erfüllen können generiert. siehe Abschnitt Was ist Proof of Quantum (PQu - Proof Quantum)?

Mining oder Ausführen des PoW oder PoQu ist ein Konzept, bei dem die Knoten eine Aufgabe zur Lösung eines mathematischen Rätsels erhalten. Der Knoten, der es im Falle eines beliebigen Blockkettensystems zuerst löst, erhält die Möglichkeit, mit dem Prozess fortzufahren, um als Gewinner ausgewählt zu werden und die Transaktionswarteschlange, die auf die Verarbeitung wartet, bearbeiten zu können.

Block für die Erstellung **neuer** Blöcke (**NewBlock**).

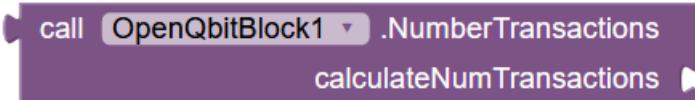


Eingabeparameter: **Daten** <Zeichenkette>, **vorherigerHash** <Zeichenkette>

Ausgabeparameter: Es liefert einen Hash berechnet als: **SHA256 (Daten (Eingabeparameter) + previousHash (Eingabeparameter) + IMEI (interner Parameter) + MerkleRoot (interner Parameter))**.

Beschreibung: Dieser Block führt die Erstellung eines neuen, zu verarbeitenden Blocks durch. Sie gibt als Ausgabe einen "SHA256"-Hash aus, der sich aus der Zeichenfolge der Eingabeparameter zusammensetzt, wenn die Daten je nach Systemdesign unterschiedlich sind und der vorherige Hash auf dem Hash der vorherigen Transaktionszeichenfolge basiert, die bereits verarbeitet wurde und im Mini-BlocklyChain-Blockstringsystem gespeichert ist, diese beiden sind variable Parameter, es gibt zwei interne Systemparameter, die fest sind und die Integrität der Informationen und des Systems gewährleisten, diese beiden internen Parameter sind die eindeutige ID des Mobiltelefons und der Hash des Merklet-Baums der Transaktionswarteschlange, die gerade verarbeitet wird.

Block der gesamten lokalen Transaktionen des Knotens (**AnzahlTransaktionen**)



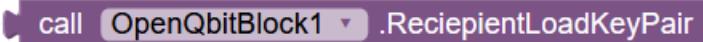
Obligatorische Abhangigkeit: Block (**AddKeyValueArrayStoS**). Geben Sie diesen Block vor der Verwendung ein.

Eingabeparameter: < Obligatorische Abhangigkeit>.

Ausgabeparameter: Gibt die Summe der lokalen Transaktionen an den Knoten zuruck.

Beschreibung: Liefert die Summe der Transaktionen, die nur auf die lokalen Konten des Knotens angewendet werden.

Block zum Lesen der Adresse des Empfanglers in einer Binardatei. (**EmpfangerLastschlusselpaar**)

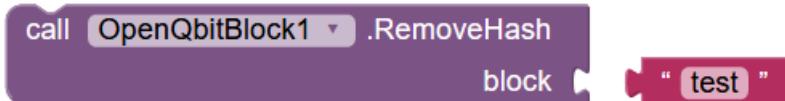


Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt privaten und offentlichen Schlssel in einem Base64-Format zuruck.

Beschreibung: Ruft die private und offentliche Adresse des Empfanglers aus einer Binardatei als Eingabeparameter ab.

Block zum Loschen von Elementen in der internen temporaren Anordnung "Kette" (**RemoveHash**).

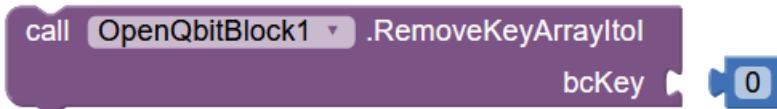


Eingabeparameter: **Block < String >**.

Ausgabeparameter: Nicht zutreffend.

Beschreibung: Ruft die private und offentliche Adresse des Empfanglers aus einer Binardatei als Eingabeparameter ab.

Block zum Löschen eines Elements in der internen temporären Anordnung "Itol\_UTXO" (**RemoveKeyArrayItol**)

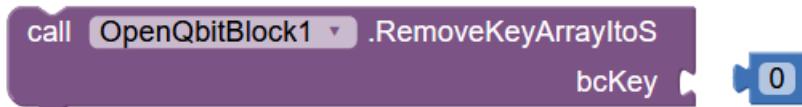


Eingabeparameter: **bcKey < Ganzzahl >**.

Ausgabeparameter: Nicht anwendbar, Art der Elementlöschung <Integer>

Beschreibung: Das Element, das mit dem numerischen Label der internen temporären Anordnung "Itol\_UTXO" verbunden ist, wird gelöscht.

Block zum Löschen eines Elements in der internen temporären Anordnung "ItoS\_UTXO" (**RemoveKeyArrayItoS**).

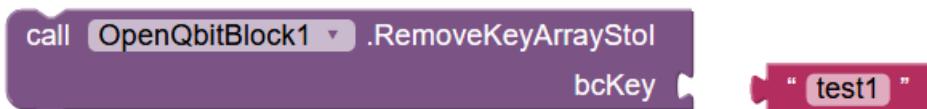


Eingabeparameter: **bcKey < Ganzzahl >**.

Ausgabeparameter: Nicht anwendbar, Typ des zu löschenen Elements <String>

Beschreibung: Das Element, das mit dem numerischen Label der internen temporären Anordnung "ItoS\_UTXO" verbunden ist, wird gelöscht.

Block zum Löschen von Elementen in der internen temporären Anordnung "Stol\_UTXO" (**RemoveKeyArrayStol**)

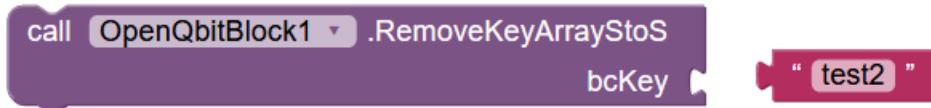


Eingabeparameter: **bcKey < String >**.

Ausgabeparameter: Nicht anwendbar, Art der Elementlöschung <Integer>

Beschreibung: Das Element, das mit der numerischen Bezeichnung der internen vorübergehenden Anordnung "Stol\_UTXO" verbunden ist, wird gelöscht.

Block zum Löschen von Elementen in der internen temporären Anordnung "StoS\_UTXO" (RemoveKeyArrayStoS)



Eingabeparameter: **bcKey < String >**.

Ausgabeparameter: Nicht anwendbar, Typ des zu löschen Elements <String>

Beschreibung: Löschen Sie das Element der internen temporären Anordnung "StoS\_UTXO".

Block zum Ersetzen eines Wertes der internen temporären Anordnung "Kette" (ReplaceHash).

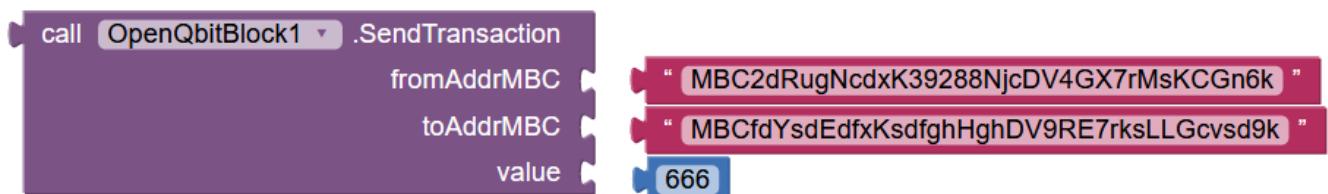


Eingabeparameter: **Block < String >, Index < Ganzzahl >**

Ausgabeparameter: Nicht zutreffend.

Beschreibung: Das mit dem Index "index" verbundene Element wird in der temporären internen Anordnung "chain" durch den Wert des Labels "block" ersetzt.

Block zum Starten (**Senden**) einer neuen Transaktion (**SendTrasaction**).



Eingabeparameter: **vonAddrMBC < String >, bisAddrMBC < String >, Wert < Ganzzahl >**

Ausgabeparameter: Gibt den Wert "True" zurück, wenn die Transaktion erfolgreich gesendet wurde, oder "False", wenn sie nicht erfolgreich gesendet wurde.

Beschreibung: Block, der die Absender- und Empfängeradresse in ein Binärformat umwandelt und an die zu verarbeitende Transaktionswarteschlange angehängt wird.

Block zum Lesen der Absenderadresse in einer Binärdatei. (**SenderLoadKeyPair**)

 call OpenQbitBlock1 .SenderLoadKeyPair

Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt privaten und öffentlichen Schlüssel in einem Base64-Format zurück.

Beschreibung: Ruft die private und öffentliche Adresse des Absenders aus einer Binärdatei als Eingabeparameter ab.

Block, um die Elementnummer der temporären Anordnung "Kette" (**SizeBlockList**) zu erhalten.

 call OpenQbitBlock1 .SizeBlockList

Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt die Elementnummer des internen Arrays "chain" zurück.

Beschreibung: Ruft die Elementnummer der internen Array "Kette" ab.

Block zum Erhalt der Elementnummer der vorübergehenden Anordnung "ItoI\_UTXO" (**SizeItoI**)

 call OpenQbitBlock1 .SizeItoI

Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt die Elementnummer des internen Arrays "ItoI\_UTXO" zurück.

Beschreibung: Liefert die Elementnummer des internen Arrays "ItoI\_UTXO".

Block zum Erhalt der Elementnummer der vorübergehenden Anordnung "ItoS\_UTXO" (**SizeItoS**)

 call OpenQbitBlock1 .SizeItoS

Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt die Elementnummer des internen Arrays "ItoS\_UTXO" zurück.

Beschreibung: Ruft die Elementnummer des internen Arrays "ItoS\_UTXO" ab.

Block zum Erhalt der Elementnummer der vorübergehenden Anordnung "StoI\_UTXO" (SizeStoI)



Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt die Elementnummer des internen Arrays "StoI\_UTXO" zurück.

Beschreibung: Ruft die Elementnummer des internen Arrays "StoI\_UTXO" ab.

Block zum Erhalt der Elementnummer der vorübergehenden Anordnung "StoS\_UTXO" (SizeStoS)

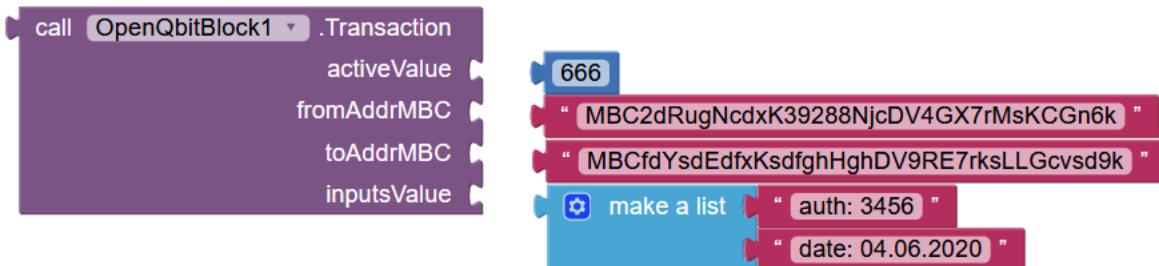


Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt die Elementnummer des internen Arrays "StoS\_UTXO" zurück.

Beschreibung: Ruft die Elementnummer des internen Arrays "StoS\_UTXO" ab.

Blockerstellung einer Transaktion mit zusätzlichen Werten (**Transaktion**).



Eingabeparameter: `activeValue < Integer >`, `vonAddrMBC < String >`, `bisAddrMBC < String >`, `inputsValue < Array String >`.

Ausgabeparameter: Es gibt uns die Adressen im Binärformat und den Wert `inputValue`, der in einen String 'String' konvertiert wurde, und gibt uns den Hash "SHA256" des ActiveValue-Wertes.

Beschreibung: Bereitet eine neue Transaktion vor, die von den Knoten verarbeitet werden soll.

### Mini-BlocklyChain-Adressvalidierungsblock (**ValidierenHinzufügenMiniBlocklyChain**)



Eingabeparameter: Benutzeradressen im Mini-BlocklyChain-Format.

Ausgabeparameter: Gibt "Wahr" zurück, wenn die Adresse im richtigen Format ist, oder "Falsch", wenn die Adresse ungültig ist.

Beschreibung: Validiert, ob die eingegebene Mini-BlocklyChain-Adresse korrekt ist. Dieser Block wendet einen Algorithmus an, um zu überprüfen, ob die Adresse mit dem Adresserstellungsmechanismus erstellt wurde, der im Mini-BlocklyChain-System verwendet werden soll.

### Bitcoin-Adressvalidierungsblock. (**ValidateAddBitcoin**)



Eingabeparameter: **addr < String >**, Benutzeradressen im Bitcoin-Format (akzeptiert Bitcoin-Adressen mit der Anfangskennung "1", Adressen mit der Kennung "3" sind nicht anwendbar).

Ausgabeparameter: Gibt "Wahr" zurück, wenn die Adresse im richtigen Format ist, oder "Falsch", wenn die Adresse ungültig ist.

Beschreibung: Überprüft, ob die eingegebene Bitcoin-Adresse korrekt ist. Dieser Block wendet einen Algorithmus an, um zu überprüfen, ob die Adresse mit dem Adresserstellungsmechanismus erstellt wurde, der im Bitcoin-System verwendet werden soll.

### Verifizierungsblock der digitalen Signatur für die aktuelle Transaktion. (**VerifizierenSignatur**).



Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Gibt "True" zurück, wenn die Prüfung gültig ist, oder "False", wenn die Prüfung ungültig ist.

Beschreibung: Ruft die Überprüfung der digitalen Signatur ab, die der Absender beim Senden der gewünschten Transaktion hätte sehen müssen. Bei diesem Verifizierungsprozess wird überprüft, ob der gesendete Quellwert in dem Kanal, über den die Transaktion gesendet wurde, nicht geändert wurde, und es wird auch der Empfänger überprüft, auf den die Transaktion angewendet werden soll.

## 20. Verwendung von Blöcken für SQLite-Datenbank (MiniSQLite-Version)

In diesem Abschnitt werden wir sehen, wie die Blöcke verwendet werden, um zwei Hauptoperationen durchzuführen, die für die Funktionalität des Mini BlocklyChain-Systems von Interesse sind, nämlich Daten in die Blockkette "EINFÜGEN" und diese abfragen.

HINWEIS: Die Transaktionen in der SQLite-Datenbank unterscheiden sich von den Transaktionen, die von den Knoten gesendet werden, die im Mini-Blocklychain-System angewendet werden sollen.

Die Transaktionen in der Datenbank basieren auf einem CRUD-Modell (Create, Read, Update and Delete) und sind die Prozesse, die mit externen oder internen Daten nur in der SQLite-Datenbank ausgeführt werden können. In der Blockkette Systeme sind vor allem die Prozesse des Erstellen und Lesen verwendet werden, für die Sicherheit sind die Prozesse der Aktualisierung oder Löschen und mehr, wo es gespeichert ist die Kette von Blöcken, die die integrale Sicherheit des Systems gibt verworfen.

Auf der anderen Seite, die Transaktionen, die von den Knoten gesendet werden, beziehen sich auf alle Prozesse, die die Aktion des Sendens einer Art von Vermögenswert zwischen den Mitgliedern (Knoten) des Mini BlocklyChain-Systems, diese Art von Transaktionen umfassen verschiedene Prozesse für seine Anwendung, diese können von der Erstellung einer digitalen Adresse, eine digitale Signatur, eine Signatur-Validierung, ein Prozess innerhalb der SQLite-Datenbank, unter anderem Prozesse.

Wir beginnen mit der Definition und Verwendung der Blöcke der SQLite-Datenbank MiniSQLite Version diese Version ist nur durch 8 Blöcke integriert. Sie verfügen über eine Vollversion, um die Daten in der SQLite-Datenbank zu manipulieren. Für praktische Zwecke ist jedoch das Mini BlocklyChain-System mit der MiniSQLite-Version ausreichend.

Für den Fall, dass Sie alle Komponenten ihre Verwendung und Beschreibung überprüfen möchten, siehe Anhang "Erweiterte Blöcke für die SQLite-Datenbank".

### MiniSQLite-Versionsblöcke:

Block zum Starten irgendeiner Art von Transaktion in der SQLite-Datenbank (**BeginTransaction**)

call OpenQbitQSQLite1 .BeginTransaction

Obligatorische Einheit(en): Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **Verwendung vor < Obligatorische Abhängigkeit(en)>**

Ausgabeparameter: Nicht anwendbar, es startet eine Transaktion in einer SQLite-Datenbank.

Beschreibung: Block, der einen Prozess in der SQLite-Datenbank startet, Sie müssen zuerst den Datenbank-Importblock (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Ausführen von Commit in SQLite. (**CommitTrasaction**)

call **OpenQbitQSQLite1** .**CommitTransaction**

Erforderliche Abhängigkeit(en): Block (**BeginTransaction**), Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **Verwendung vor** < Obligatorische Abhängigkeit(en)>

Ausgabeparameter: Nicht anwendbar, es führt einen **Commit einer** Transaktion in einer SQLite-Datenbank durch.

Beschreibung: Block, der einen **Commit-Prozess auf** der SQLite-Datenbank startet, Sie müssen zuerst den Datenbank-Import-Block (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Schließen der SQLite-Datenbank, die importiert oder exportiert wurde (**CloseDatabase**)

call **OpenQbitQSQLite1** .**CloseDatabase**

Obligatorische Einheit(en): Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **Verwendung vor** < Obligatorische Abhängigkeit(en)>

Ausgabeparameter: Nicht anwendbar, es schließt eine SQLite-Datenbank.

Beschreibung: Block, der die SQLite-Datenbank schließt, Sie müssen zuerst den Datenbank-Importblock (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Exportieren einer SQLite-Datenbank (**ExportDatabase**).

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

Obligatorische Einheit(en): Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **fileName < String >** geben Sie einen Pfad ein, in dem Sie eine bereits mit SQLite-Format erstellte Datenbank finden können.

**Verwendung vor < Obligatorische Abhängigkeit(en) >**

Ausgabeparameter: Export in eine SQLite-Datenbank.

Beschreibung: Block, der eine SQLite-Datenbank exportiert, müssen Sie zuerst den Datenbank-Importblock (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Importieren der SQLite-Datenbank. (**Datenbank importieren**)

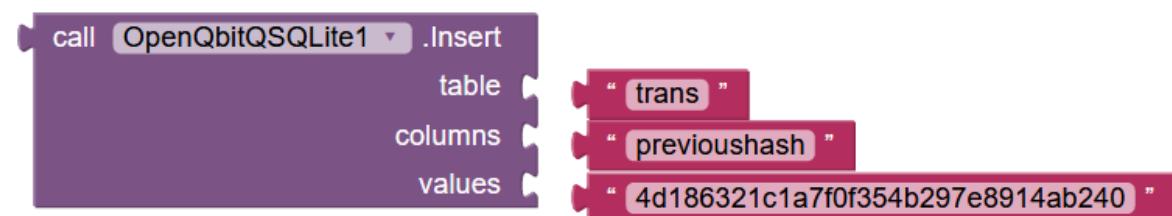


Eingabeparameter: **fileName < String >** geben Sie einen Pfad ein, in dem Sie eine bereits mit SQLite-Format erstellte Datenbank finden können.

Ausgabeparameter: Startet eine SQLite-Datenbank zur Ausführung von Transaktionen.

Beschreibung: Block, der einen Prozess in der SQLite-Datenbank startet, Sie müssen zuerst den Datenbank-Importblock (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Einfügen von Daten in die SQLite-Datenbank. (**Einfügen**)



Obligatorische Einheit(en): Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **Tabelle < String >**, **Spalten < String >**, **Werte < String >**, **Verwendung vor < Obligatorische Abhängigkeit(en) >**

Ausgabeparameter: Fügt eine Transaktion in eine SQLite-Datenbank ein.

Beschreibung: Block, der Daten in die SQLite-Datenbank einfügt, müssen Sie zuerst den Datenbank-Importblock (**ImportDatabase**) und den Datenbank-Offen-Block (**OpenDatabase**) ausgeführt haben.

Block zum Öffnen der SQLite-Datenbank (**OpenDatabase**)

```
call OpenQbitQSQLite1 .OpenDatabase
```

Obligatorische Einheit(en): Block (**ImportDatenbank**).

Eingabeparameter: **Verwendung vor** < Obligatorische Abhängigkeit(en)>

Ausgabeparameter: Nicht zutreffend, starten oder öffnen Sie eine SQLite-Datenbank, um Transaktionen durchzuführen.

Beschreibung: Block, der eine SQLite-Datenbank startet, muss es vorher geben.

Block für Datenabfrage in SQLite (**Ausführen**).

```
call OpenQbitQSQLite1 .Execute
      sql
      bindParams
```

Obligatorische Einheit(en): Block (**ImportDatabase**), Block (**OpenDatabase**).

Eingabeparameter: **sql** < String> , **bindParams** < String> , **Verwendung vor** < Obligatorische Abhängigkeit(en)>

Ausgabeparameter: Die SQL-Anweisung wird ausgeführt, um eine Transaktion in einer SQLite-Datenbank zu erstellen.

Beschreibung: Block, der SQL-Anweisungen in der SQLite-Datenbank ausführt, muss zuerst den Block der importierenden Datenbank (**ImportDatabase**) und den Block der Öffnung der Datenbank (**OpenDatabase**) ausgeführt haben.

## 21. Definition und Verwendung von Sicherheitsblöcken.

In dieser Sitzung werden wir die Verwendung der Blöcke überprüfen, die uns Sicherheitsebenen zum Speichern, Validieren und Übertragen von Transaktionen von den Mini-BlocklyChain-Netzwerknoten bieten.

Die Sicherheitsblöcke basieren auf der folgenden Erweiterung:

- I. OpenQbitAESEncryption-Erweiterung.
- II. OpenQbitAESDecryption-Erweiterung.
- III. OpenQbitAEToString-Erweiterung.
- IV. OpenQbitEncDecData-Erweiterung.
- V. OpenQbitFileHash-Erweiterung.
- VI. OpenQbitRSA-Erweiterung.
- VII. OpenQbitSSHClient-Erweiterung (würde erfordern)
- VIII. OpenQbitStringHash-Erweiterung.

Mit Ausnahme der OpenQbitSSHClient-Erweiterung, die obligatorisch ist, sind die oben genannten Erweiterungen optional zur Verwendung bei der Erstellung eines öffentlichen oder privaten Mini BlocklyChain-Netzwerks.

Um jedoch ein sicheres System für Ihre Transaktionen in Abhängigkeit vom jeweiligen Geschäftsfall zu haben, sollte die Verwendung der optionalen Erweiterungen nach Ihrem Ermessen erfolgen.

Im Falle der Verwendung von Hash können wir zum Beispiel eine Reihe von Algorithmusoptionen verwenden, wie MD5, SHA1, SHA128, SHA256, SHA512 für eine Zeichenfolge sowie auf jeden Dateityp angewendet werden, abhängig vom Informationsfluss jedes mit Mini BlocklyChain erstellten Systems.

OpenQbitAESEncryption-Erweiterung.

Block zur Verschlüsselung von Dateien mit AES-Sicherheit. (**AESE-Verschlüsselung**).



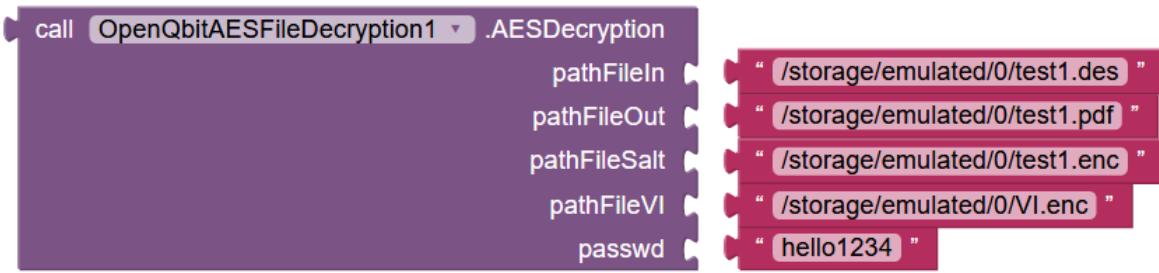
Eingabeparameter: **pathFileIn < String >**, **pathFileOut < String >**, **pathFile < String >**, **pathFileVI < String >** und **passwd < String >**. Die Dateinamen sind willkürlich und liegen im Ermessen des jeweiligen Systemdesigns.

Ausgabeparameter: AES-verschlüsselte Datei, eingegeben im Eingabeparameter **pathFileIn**.

Beschreibung: Block, der uns drei Ausgabedateien liefert. Eine davon ist die Quelldatei, die bereits durch den AES-Algorithmus mit einem 256-Bit-Schlüssel verschlüsselt wurde, die beiden anderen Dateien werden zur Steuerung der Erweiterung verwendet, um die Originaldatei zu entschlüsseln und wiederherzustellen.

OpenQbitAESDecryption-Erweiterung.

Block zum Entschlüsseln der AES-Datei. (**AESEntschlüsselung**).



Eingabeparameter: **pathFileIn < String >**, **pathFileOut < String >**, **pathFile < String >**, **pathFileVI < String >** und **passwd < String >**. Die Namen der Dateien sind die gleichen wie die, die als Ergebnis im Block erhalten wurden (**AESEncryption**).

Ausgabeparameter: Originaldatei entschlüsselt mit AES, eingegeben im Eingabeparameter **pathFileIn**, in diesem Fall zum Entschlüsseln der Datei wird sie in **pathFileIn** in die verschlüsselte Datei und in **pathFileOut** in die Originaldatei (entschlüsselt) eingegeben.

Beschreibung: Block, der uns im **pathFileOut**-Parameter eine Datei liefert, die vom AES-Algorithmus mit einem 256-Bit-Schlüssel entschlüsselt ausgegeben wird.

OpenQbitAESToString-Erweiterung.

Diese Erweiterung ist eine Einmalverwendung pro Gerätesitzung, d.h. sie funktioniert nur, wenn das Gerät nicht neu gestartet wird (Mobiltelefon), da der VI-Verschlüsselungswert temporär generiert wird.

HINWEIS: Wenn das Gerät neu gestartet wird, kann die verschlüsselte Zeichenfolge nicht wiederhergestellt werden.

Block für temporäre Zeichenkettenverschlüsselung (**DecrypSecretText**)

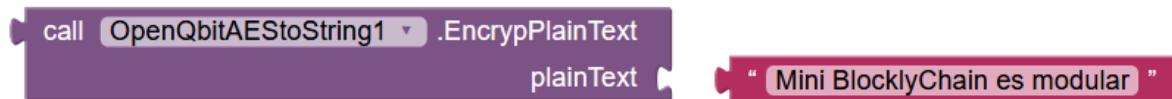


Eingabeparameter: **secretText** < String >

Ausgabeparameter: Original-Zeichenkette, entschlüsselt mit AES mit 256-Bit-Schlüssel

Beschreibung: Block, der uns eine Zeichenfolge liefert, ist der Eingabeparameter, der in den Block eingeführt wurde (**EncrypPlainText**).

Block zum Dekodieren einer Zeichenfolge (**EncrypPlainText**)



Eingabeparameter: **plainText** < String >

Ausgabeparameter: AES-verschlüsselte Zeichenfolge mit 256-Bit-Schlüssel.

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die mit AES unter Verwendung eines digitalen 256-Bit-Schlüssels verschlüsselt wurde.

OpenQbitEncDecData-Erweiterung.

Spezialisierter Verschlüsselungsblock für generische Datenbanken (**Verschlüsselungsdaten**)



Eingabeparameter: **plainText** < String >

Ausgabeparameter: AES-verschlüsselte Zeichenfolge mit 256-Bit-Schlüssel.

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die mit AES unter Verwendung eines digitalen 256-Bit-Schlüssels verschlüsselt wurde.

Spezialisierter Verschlüsselungsblock für generische Datenbanken (**DescriptonData**)



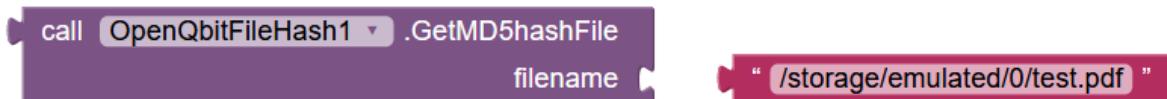
Eingabeparameter: **plainText** < String >

Ausgabeparameter: AES-verschlüsselte Zeichenfolge mit 256-Bit-Schlüssel.

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die mit AES unter Verwendung eines digitalen 256-Bit-Schlüssels verschlüsselt wurde.

OpenQbitFileHash-Erweiterung.

Block zum Erzeugen von MD5 aus einer Datei (**GetMD5hashFile**)

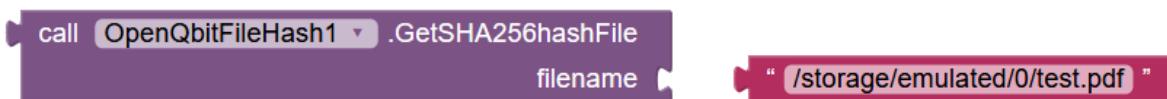


Eingabeparameter: **Dateiname** <String>

Ausgabeparameter: Liefert die MD5-Hash-Datei.

Beschreibung: Block zum Erstellen des MD5-Hashes der im Eingabeparameter angegebenen Datei.

Block zur Erzeugung von SHA256 aus einer Datei (**GetSHA256hashFile**)

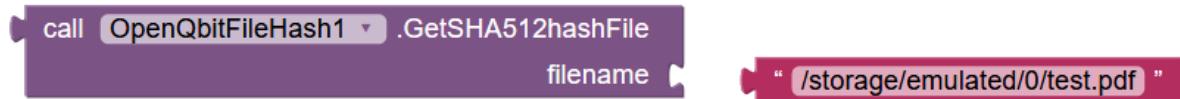


Eingabeparameter: **Dateiname** <String>

Ausgabeparameter: Liefert den SHA256-Archiv-Hash.

Beschreibung: Block zum Erstellen des SHA256-Hashes der im Eingabeparameter angegebenen Datei.

Block zum Erzeugen von SHA512 aus einer Datei (**GetSHA512hashFile**)

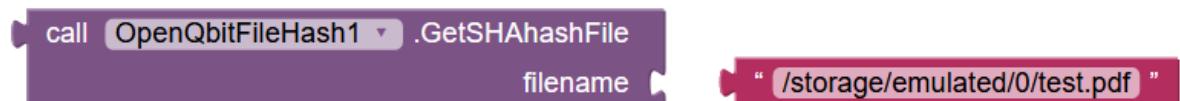


Eingabeparameter: **Dateiname** <String>

Ausgabeparameter: Liefert den SHA256-Archiv-Hash.

Beschreibung: Block zum Erstellen des SHA256-Hashes der im Eingabeparameter angegebenen Datei.

Block zur Erzeugung von SHA1 aus einer Datei (**GetSHA1hashFile**)



Eingabeparameter: **Dateiname** <String>

Ausgabeparameter: Liefert den SHA1-Archiv-Hash.

Beschreibung: Block zum Erstellen des SHA1-Hashes des Würfels im Eingabeparameter.

OpenQbitRSA-Erweiterung.

Block zum **Entschlüsseln von** Zeichenketten mit RSA (**Decrypt**)



Erforderliche Abhängigkeit(en): Block (**Verschlüsseln**), Block (**OpenFromDiskPrivateKey**), Block (**OpenFromDiskPublicKey**).

Eingabeparameter: **Ergebnis** < String>

Ausgabeparameter: Mit RSA dekodierte Zeichenkette.

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die mit Hilfe eines Schlüssels der im Block verwendeten Größe entziffert wurde (**GenKeyPair**).

Block zum Verschlüsseln von Zeichenfolgen mit RSA (**Encrypt**)



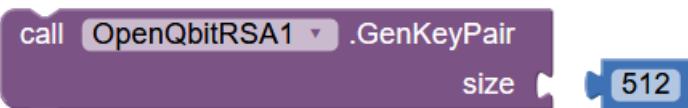
Erforderliche Einheit(en): Block (**GenKeyPair**), Block (**SaveFromDiskPrivateKey**), Block (**SaveFromDiskPublicKey**)

Eingabeparameter: **einfach** < String >

Ausgabeparameter: RSA-verschlüsselte Zeichenfolge

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die mit Hilfe eines Schlüssels der im Block verwendeten Größe entziffert wurde (**GenKeyPair**).

Block zum **Entschlüsseln von** Zeichenketten mit RSA (**Decrypt**)

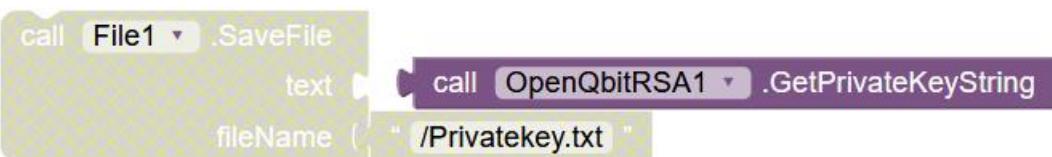


Eingabeparameter: **Größe** < Ganzzahl >

Ausgabeparameter: Nicht zutreffend.

Beschreibung: Block zur Generierung von privatem Schlüssel und öffentlichem Schlüssel auf der Grundlage der gewählten Größe.

Block zum Abrufen des privaten Schlüssels (**GetPrivateKeyString**)



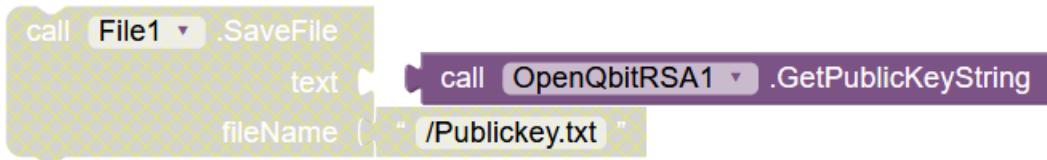
Obligatorische Einheit(en): Block (**GenKeyPair**), Block (**GenKeyPair**), Block (**Datei**)

Eingabeparameter: **Nicht zutreffend**.

Ausgabeparameter: Datei mit RSA-verschlüsselter Zeichenfolge (privater Schlüssel)

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die den privaten Schlüssel repräsentiert, der mit dem Größenschlüssel verschlüsselt wurde, der im Block verwendet wurde (**GenKeyPair**).

Block zum Abrufen des privaten Schlüssels (**GetPublicKeyString**)



Obligatorische Einheit(en): Block (**GenKeyPair**), Block (**GenKeyPair**), Block (**Datei**)

Eingabeparameter: **Nicht zutreffend**.

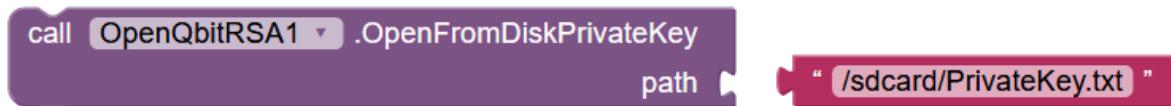
Ausgabeparameter: Datei mit RSA-verschlüsselter Zeichenfolge (öffentlicher Schlüssel)

Beschreibung: Block, der uns eine alphanumerische Zeichenfolge liefert, die den öffentlichen Schlüssel darstellt, der mit der im Block verwendeten Schlüsselgröße verschlüsselt wurde (**GenKeyPair**).

HINWEIS: In den vorherigen Blöcken (**GetPrivateKeyString**) und (**GetPublicKeyString**) verwenden wir in den Abhängigkeiten den generischen Block (**File**) der App Inventor-Anwendung der Palettensitzung "Storage".

Diese Art der Speicherung des privaten und öffentlichen Schlüssels mit Hilfe des Blocks (**Datei**) kann uns helfen, die Informationen besser zu manipulieren.

Block zum Lesen des privaten Schlüssels aus einer Datei (**OpenFromDiskPrivateKey**).



Erforderliche Abhängigkeit(en): Block (**SaveFromDiskPrivateKey**) oder Block (**GetPrivateKeyString**).

Eingabeparameter: **Pfad < String >**

Ausgabeparameter: System lädt eine mit dem privaten RSA-Schlüssel verschlüsselte Zeichenkette.

Beschreibung: Block, der uns eine verschlüsselte alphanumerische Zeichenfolge des privaten Schlüssels liefert, die im angegebenen Dateipfad gespeichert ist.

Block zum Lesen des öffentlichen Schlüssels aus einer Datei ([OpenFromDiskPublicKey](#))



Obligatorische Einheit(en): Block ([SaveFromDiskPublicKey](#)) oder Block ([GetPublicKeyString](#)).

Eingabeparameter: **Pfad** < String >

Ausgabeparameter: Verschlüsselte RSA-Zeichenkette mit öffentlichem Schlüssel in das System laden.

Beschreibung: Block, der uns eine verschlüsselte alphanumerische Zeichenfolge des öffentlichen Schlüssels liefert, die im angegebenen Dateipfad gespeichert ist.

Block zum Speichern des privaten Schlüssels in einer Datei ([SaveToDiskPrivateKey](#)).



Obligatorische Einheit(en): Block ([GenKeyPair](#)).

Eingabeparameter: **einfach** < String >

Ausgabeparameter: Datei mit RSA-verschlüsselter Zeichenfolge. (privater Schlüssel)

Beschreibung: Block, der uns eine Datei mit einer alphanumerischen Zeichenfolge liefert, die mit einem privaten Schlüssel der im Block verwendeten Größe verschlüsselt ist ([GenKeyPair](#)).

Block zum Speichern des privaten Schlüssels in einer Datei ([SaveToDiskPublicKey](#)).



Obligatorische Einheit(en): Block ([GenKeyPair](#)).

Eingabeparameter: **einfach** < String >

Ausgabeparameter: Datei mit RSA-verschlüsselter Zeichenfolge. (öffentlicher Schlüssel)

Beschreibung: Block, der uns eine Datei mit einer alphanumerischen Zeichenfolge liefert, die mit einem öffentlichen Schlüssel der Größe, die im Block verwendet wurde ([GenKeyPair](#)), verschlüsselt wurde.

OpenQbitSSHClient-Erweiterung.

**Verbinderblock** SSH-Client (**VerbinderMiniBlocklyChain**)

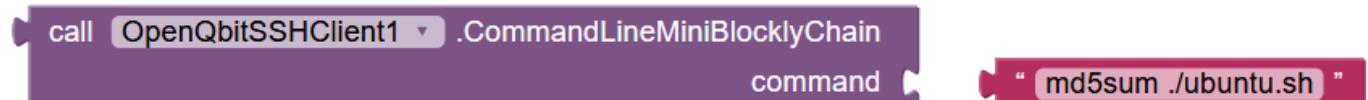


Eingabeparameter: Benutzername <string>, Passwort <string>, Host <string>, Port<integer>

Ausgabeparameter: Wenn die Verbindung mit dem ssh-Server des Termux-Terminals erfolgreich ist, gibt er uns eine Meldung; "**Connect SSH**", wenn sie nicht erfolgreich ist, gibt er uns eine **NUL-Meldung**.

Beschreibung: Kommunikationsblock zur Verbindung von Mini BlocklyChain mit dem Termux-Terminal über das SSH-Kommunikationsprotokoll (Secure Shell).

Block für die Ausführung von Befehlen im Termux Linux Terminal



(**CommandLineMiniBlocklyChain**).

Eingabeparameter: Befehl <string>

Ausgabeparameter: Variable Daten, abhängig vom ausgeführten Befehl oder Programm.

Beschreibung: Befehlsausführungsblock im Termux-Terminal Voraussetzung für die Herstellung einer Verbindung mit dem Block (**ConnectorMiniBlocklyChain**) kann alle Arten von Befehlen online ausführen und/oder spezifische Ausführungsdaten von Skripten oder Programmen erhalten, die eine CLI (Command-Line Interface) online haben.

TrennenMiniBlocklyChainSSH.

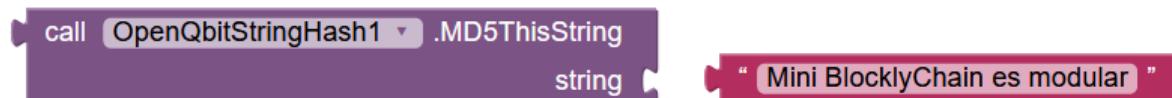


Eingabe- und Ausgabeparameter: Nicht anwendbar (keine)

Beschreibung: Blockieren zum Schließen der SSH-Sitzung.

OpenQbitStringHash-Erweiterung.

Block zur Erzeugung einer MD5-Zeichenfolge (**MD5ThisString**)

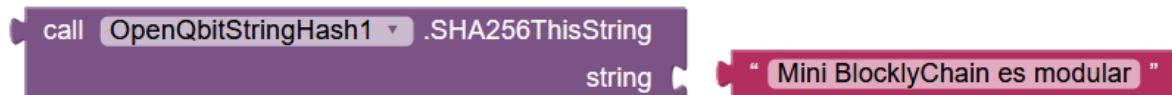


Eingabeparameter: Zeichenkette

Ausgabeparameter: Liefert den MD5-Hash.

Beschreibung: Block zum Erstellen des MD5-Hashes der im Eingabeparameter angegebenen Zeichenfolge.

Block zur Erzeugung einer SHA256-Zeichenkette (**SHA256ThisString**)

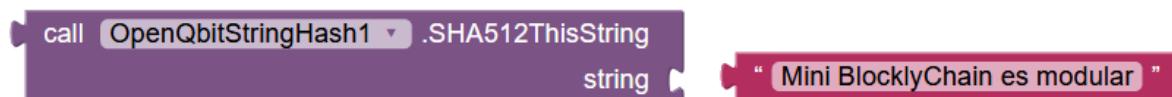


Eingabeparameter: Zeichenkette

Ausgabeparameter: Liefert den SHA256-Hash.

Beschreibung: Block zum Erstellen des SHA256-Hashes der im Eingabeparameter angegebenen Zeichenfolge.

Block zur Erzeugung einer SHA512-Zeichenkette (**SHA512ThisString**)

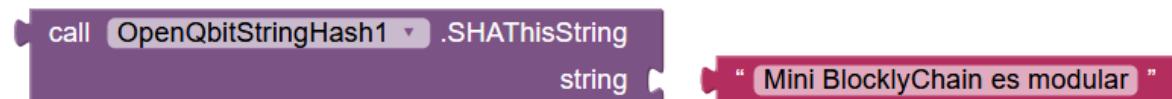


Eingabeparameter: Zeichenkette

Ausgabeparameter: Liefert den SHA512-Hash.

Beschreibung: Block zum Erstellen des SHA512-Hashes der im Eingabeparameter angegebenen Zeichenfolge.

Block zur Erzeugung einer SHA1-Zeichenkette (**SHAThisString**)



Eingabeparameter: Zeichenkette

Ausgabeparameter: Liefert den SHA1-Hash.

Beschreibung: Block zum Erstellen des SHA1-Hashes der im Eingabeparameter angegebenen Zeichenfolge.

## 22. Einstellen von Sicherheitsparametern in Mini BlocklyChain.

Die Sicherheitsparameter werden in drei Komponenten eines jeden Systems unterteilt, das in den folgenden Komponenten entworfen und angewendet wird:

- a. Redis-Datenbank (Backup-Kommunikationsnetz)
- b. Peer-to-Peer-Synchronisationssystem
- c. Integrierte Sicherheit zum Schutz der SQLite-Datenbank
- d. Entwicklerumgebung für die Modulintegration

- a. Redis-Datenbank (Backup-Kommunikationsnetz)

Um gefährliche Befehle umzubenennen, beinhaltet die zusätzliche eingebaute Sicherheitsfunktion von Redis die Umbenennung oder Deaktivierung einiger Befehle, die als gefährlich gelten.

Wenn diese Befehle von nicht autorisierten Benutzern ausgeführt werden, können sie dazu verwendet werden, ihre Daten zu rekonfigurieren, zu zerstören oder zu löschen. Wie das Authentifizierungs-Passwort wird das Umbenennen oder Deaktivieren von Befehlen im gleichen SECURITY-Abschnitt der Datei `/etc/redis/redis.conf` konfiguriert.

Einige der Befehle, die als gefährlich gelten: **FLUSHDB**, **FLUSHALL**, **KEYS**, **PEXPIRE**, **DEL**, **CONFIG**, **SHUTDOWN**, **BGREWRITEAOF**, **BGSAVE**, **SAVE**, **SPOP**, **SREM**, **RENAME** und **DEBUG**. Dies ist keine vollständige Liste, aber die Umbenennung oder Deaktivierung aller Befehle auf dieser Liste ist ein guter Anfang, um die Sicherheit Ihres Redis-Servers zu verbessern.

Je nach Ihren spezifischen Bedürfnissen oder denen Ihrer Website müssen Sie einen Befehl umbenennen oder deaktivieren. Wenn Sie wissen, dass Sie niemals einen Befehl verwenden werden, der manipuliert werden kann, können Sie ihn deaktivieren. Auf der anderen Seite möchten Sie es vielleicht umbenennen.

Um Redis-Befehle zu aktivieren oder zu deaktivieren, öffnen Sie die Konfigurationsdatei erneut:

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

**Warnung:** Die folgenden Schritte zum Deaktivieren und Umbenennen von Befehlen sind Beispiele. Sie sollten nur die Befehle deaktivieren oder umbenennen, die auf Sie zutreffen. Sie können die vollständige Liste der Befehle einsehen und feststellen, wie sie in [redis.io/commands](https://redis.io/commands) missbraucht werden können.

Um einen Befehl zu deaktivieren, benennen Sie ihn einfach so um, dass er zu einer leeren Zeichenfolge wird (symbolisiert durch ein Anführungszeichenpaar ohne Zeichen dazwischen), wie unten gezeigt:

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

```
.
.
.
# Es ist auch möglich, ein Kommando komplett abzuschaffen, indem man es
umbenennt in
# eine leere Zeichenfolge:
#
Umbenennen-Befehl FLUSHDB ""
Umbenennen-Befehl FLUSHALL ""
umbenennen-Befehl DEBUG ""
```

Um einen Befehl umzubenennen, geben Sie ihm einen anderen Namen, wie in den folgenden Beispielen gezeigt. Umbenannte Befehle sollten für andere schwer zu erraten, für Sie aber leicht zu merken sein.

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

```
.
.
.
# rename-befehl CONFIG ""
rename-Befehl SHUTDOWN SHUTDOWN_MENOT
rename-Befehl CONFIG ASC12_CONFIG
```

Speichern Sie die Änderungen und schließen Sie die Datei.

Nachdem Sie einen Befehl umbenannt haben, wenden Sie die Änderung an, indem Sie Redis neu starten:

- sudo systemctl Neustart redis.service

Um den neuen Befehl zu testen, geben Sie die Redis-Befehlszeile ein:

- redis-cli

Führen Sie dann die Authentifizierung durch:

- auth your\_redis\_password

Ausgabe

OK

Angenommen, Sie benennen den CONFIG-Befehl wie im vorherigen Beispiel in ASC12\_CONFIG um. Versuchen Sie zunächst, den ursprünglichen CONFIG-Befehl zu verwenden. Weil Sie es umbenannt haben, sollte es nicht funktionieren:

- Konfiguration erhalten requirepass

Ausgabe

(Fehler) ERR unbekannter Befehl 'config'

Der umbenannte Befehl kann jedoch erfolgreich aufgerufen werden. Es wird nicht zwischen Groß- und Kleinschreibung unterschieden:

- asc12\_config liefert requirepass

Ausgabe

1) "Erfordernispass".  
2) "Ihr\_Redis\_Passwort".

Schließlich können Sie die Neueinteilung der Bezirke schließen:

- Ausfahrt

Beachten Sie, dass Sie sich erneut authentifizieren müssen, wenn Sie bereits die Redis-Befehlszeile verwenden und Redis neu starten. Andernfalls wird dieser Fehler angezeigt, wenn Sie einen Befehl eingeben:

Ausgabe

NOAUTH Authentifizierung erforderlich.

## b. Peer-to-Peer-Synchronisationssystem

Bei der Verwendung des "Peer-to-Peer"-Systems zwischen Knoten hat das System die folgenden drei im Kommunikationsnetz angewandten Elemente

-Privat: Es gibt keine Informationen, die irgendwo anders als auf Ihren Computern gespeichert sind. Es gibt keinen zentralen Server, der (legal oder illegal) kompromittiert werden kann.

-Verschlüsselt: Die gesamte Kommunikation wird durch das TLS-Protokoll (Transport Layer Security) gesichert; ein kryptographisches Protokoll, das eine perfekte Sequenz enthält, um zu verhindern, dass jemand außerhalb Ihres Vertrauens auf Ihre Informationen zugreifen kann.

-Authentifiziert: Jeder Knoten wird mit einem starken kryptografischen Zertifikat identifiziert. Nur die Knoten, die Sie explizit erlaubt haben, können eine Verbindung zu Ihren Informationen herstellen.

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

Verwendung des Online-Befehls SyncthingManager:

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

c. Integrierte Sicherheit zum Schutz der SQLite-Datenbank

Bei Verwendung der Erweiterung (**OpenQbitSQLite**) oder in ihrem Fall der Erweiterung (**ConnectorSSHClient**) zur Verwendung der SQLite CLI können beide Erweiterungen mit der Sicherheitserweiterung AES (**OpenQbitEncDecData**) kombiniert werden.

Siehe Anhang "Beispiel für die Erstellung eines Mini BlocklyChain-Systems".

d. Entwicklerumgebung für die Modulintegration

Die Implementierung von Sicherheit in der Entwicklungsumgebung kann mit zwei Verfahren erfolgen:

- Beschränken Sie die Verwendung von OpenJDK-Bibliotheken.
- Verwendung beschränkt auf autorisierte Systemknoten.

## 23. Anhang "Erstellung von KeyStore & PublicKeys-Datenbanken".

Ein KeyStore ist ein Repository von Sicherheitszertifikaten, entweder Autorisierungszertifikate oder Public-Key-Zertifikate, Passwörter oder generische Sicherheitsschlüssel wie die entsprechenden privaten Schlüssel (Adressen) eines Benutzers, das zur Erstellung oder Bearbeitung von Transaktionen verwendet wird.

Es handelt sich um eine verschlüsselte Datenbank, so dass nur der Eigentümer davon Gebrauch machen kann. Normalerweise ist es ein lokaler Typ, aber im Mini BlocklyChain System ist es eine sichere Datenbank, aber es wird in allen Knoten geteilt und verteilt, dies ist im Grunde aus einem einfachen Grund, alle Knoten müssen die Adressen aller Benutzer kennen (öffentliche Adressen), die privaten Adressen sind immer lokal und sind von einzigartigem und exklusivem Nutzen für jeden Benutzer, aber wenn eine Eingangs- oder Ausgangstransaktion durchgeführt wird, hat jede Transaktion immer mindestens drei Komponenten, wenn sie erstellt wird: Herkunftsadresse, Zieladresse und Vermögenswert oder Wert, der gesendet wird.

Um unseren KeyStore zu erstellen, müssen wir die folgenden Schritte und Anforderungen befolgen.

In unserem Fall verwenden wir die SQLite-Datenbank und erstellen zwei KeyStore, einen mit den privaten Schlüsseln des Benutzers, der lokal ist und die Informationen sicher "verschlüsselt" speichert, und eine weitere Datenbank, die die öffentlichen Schlüssel speichert. Diese Datenbank wird in allen Knoten des Netzwerks gemeinsam genutzt, die Datenbank, die im Netzwerk gemeinsam genutzt wird, wird ebenfalls verschlüsselt, jedoch mit einem Passwort, das nur die Mitglieder (Knoten) des Netzwerks gemeinsam nutzen können.

Zweite grundlegende Anforderung ist der Prozess der Informationsverschlüsselung, dies wird mit der spezialisierten Erweiterung für die Verwendung in einer generischen Datenbank namens (**OpenQbitEncDecData**), die einen AES-Algorithmus verwendet, durchgeführt.

Die Erweiterung (**OpenQbitEncDecData**) ist funktional für die Verifizierung der einheitlichen Elemente der Blockkette, aber im Falle, dass die gesamte Integrität der Blockkette überprüft werden muss, wird sie nicht effizient sein, so dass wir auch eine Verschlüsselung einer Kopie durchführen werden, aber in diesem Fall wird es durch die Erweiterungen geschehen: (**OpenQbitAESEncryption**) und (**OpenQbitAESDecryption**), die an einer Datei arbeiten, nicht an referenzierten Daten.

**HINWEIS:** Der SSH-Server muss auf dem Termux-Terminal laufen, damit die **KeyStore-Datenbank** ordnungsgemäß funktioniert. Führen Sie den Befehl im Terminal aus:

`$ sshd`

Erweiterungen, die auf dem AES-Algorithmus basieren, können für jeden Daten- oder Dateityp verwendet werden, und dieser Algorithmus wurde gewählt, da er der einzige ist, der nachweislich gegen auf Quantencomputing basierende Angriffe beständig ist.

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		6.681	$3,36 \times 10^7$	$2,29 \times 10^{32}$ años	
RSA	Cifrado asimétrico	1024 2048 4096	80 112 128	Algoritmo de Shor	2.290 4.338 8.434	$2,56 \times 10^6$ $6,2 \times 10^6$ $1,47 \times 10^7$	3,58 horas 28,63 horas 229 horas	Migrar a un algoritmo PQC seleccionado por el NIST
ECC Problema del logaritmo discreto	Cifrado asimétrico	256 386 512	128 192 256	Algoritmo de Shor	2.330 3.484 4.719	$3,21 \times 10^6$ $5,01 \times 10^6$ $7,81 \times 10^6$	10,5 horas 37,67 horas 95 horas	Migrar a un algoritmo PQC seleccionado por el NIST
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

Die obige Tabelle bezieht sich auf die Nationalen Akademien der Wissenschaften, Ingenieurwissenschaften und Medizin.

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

#### Beschreibung

Die Quantenmechanik, das Teilgebiet der Physik, das das Verhalten sehr kleiner Teilchen (Quanten) beschreibt, liefert die Grundlage für ein neues Paradigma der Informatik. Erstmals in den 1980er Jahren als ein Weg zur Verbesserung der rechnergestützten Modellierung von Quantensystemen vorgeschlagen, hat das Gebiet der Quanteninformatik in letzter Zeit aufgrund der Fortschritte bei der Konstruktion von Geräten im Kleinmaßstab große Aufmerksamkeit erregt. Bis ein praktischer Quantencomputer in großem Maßstab realisiert werden kann, sind jedoch noch erhebliche technische Fortschritte erforderlich.

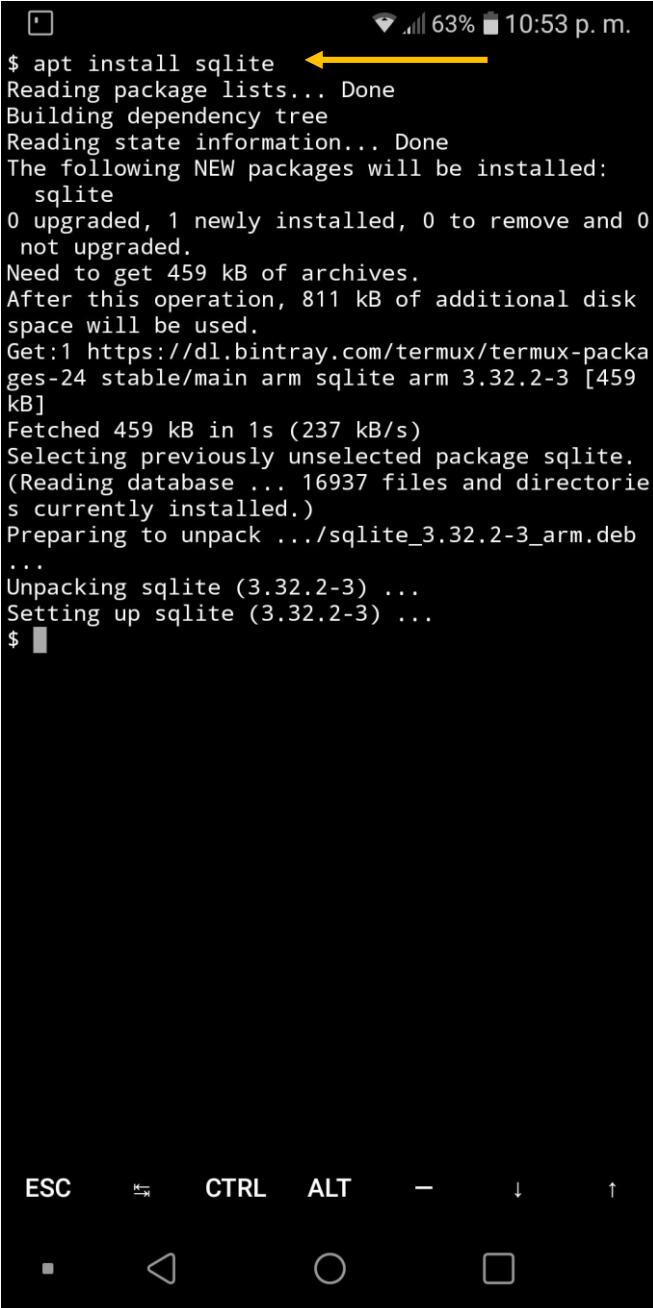
Quantum Computing: Progress and Prospects (Quantencomputer: Fortschritt und Perspektiven) bietet eine Einführung in das Gebiet, einschließlich der einzigartigen Eigenschaften und Grenzen der Technologie, und bewertet die Durchführbarkeit und die Auswirkungen der Schaffung eines funktionalen Quantencomputers, der in der Lage ist, Probleme der realen Welt zu lösen. Dieser Bericht befasst sich mit Hardware- und Software-Anforderungen, Quantenalgorithmen, Treibern für Fortschritte im Quantencomputing und bei Quantengeräten, Benchmarks in Verbindung mit relevanten Anwendungsfällen, dem Zeit- und Ressourcenaufwand und der Einschätzung der Erfolgswahrscheinlichkeit.

Installation des SQLite-Datenbank-Handlers im TERMUX-Terminal und Testen der CLI (Command-Line) des **sqlite3-Befehls** durch Erstellen einer Datenbank namens **keystore.db**

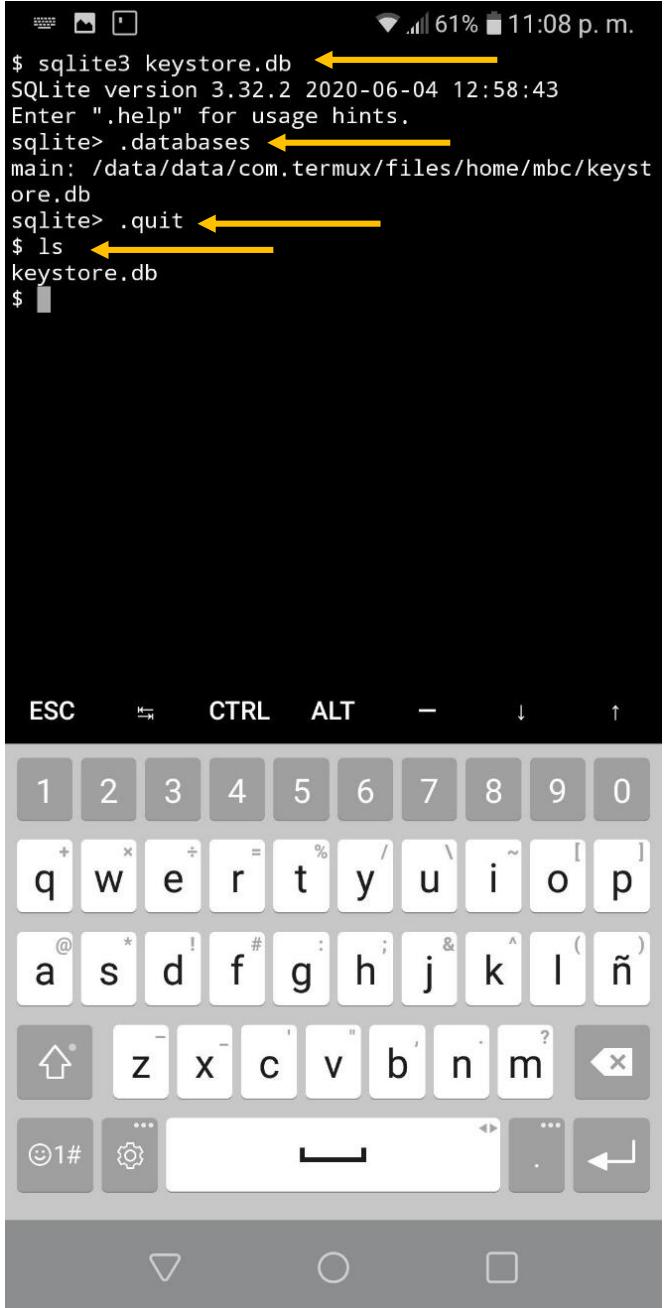
Wir benutzen die Befehle:

```
$ apt install sqlite
```

```
$ sqlite3 keystore.db
```



```
$ apt install sqlite
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sqlite
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 459 kB of archives.
After this operation, 811 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm sqlite arm 3.32.2-3 [459
kB]
Fetched 459 kB in 1s (237 kB/s)
Selecting previously unselected package sqlite.
(Reading database ... 16937 files and directo
ries 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/mbe/keyst
ore.db
sqlite> .quit
$ ls
keystore.db
$
```

Dann führen wir innerhalb des **sqlite>-Handlers** den Satz **.databases** aus, um zu prüfen, in welchem

Pfad sich die zu erstellende Datenbank befindet, und um dann die Datenbank zu verlassen und zu speichern, geben wir den Satz von . quit ein.

**HINWEIS:** In beiden Anweisungen oder Befehlen innerhalb des **sqlite>-Handlers** müssen Sie zuerst einen Punkt "." in die Syntax setzen.

Zum Schluss überprüfen wir, ob die (leere) Datenbank bereits erstellt wurde, indem wir den Befehl geben:

**\$ ls**

Wir fahren mit der Erstellung einer Tabelle fort, in der die Primärschlüssel in drei verschiedenen Formaten gespeichert werden: hexadezimal, binär und die Mini BlocklyChain-Benutzerreferenzadresse, die der öffentliche Schlüssel des jeweiligen Primärschlüssels ist.

Die AES-Datenverschlüsselung wird nur in hexadezimalen und binären Formaten angewendet. Im Falle der Adresse des Benutzers addrMBC und des Alias nicht, weil es sich um den öffentlichen Schlüssel handelt, der im gesamten Netzwerk gemeinsam genutzt werden kann und sollte, um Transaktionen zu empfangen, sowie um den Alias, um die Suche nach diesem Feld durchzuführen.

Erstellte eine Tabelle namens "privatekey" in der Datenbank in SQLite (keystore.db).

```
CREATE TABLE privatekey erzeugen (
    id ganzzahliger Primärschlüssel
        a.k.a.      VARCHAR(50) NICHT NULL
    addrHexVARCHAR    (65) NICHT NULL
    addrMBCVARCHAR   (65) NICHT NULL
    addrBinBLOB      NICHT NULL
);
```

Lassen Sie uns die Sätze in der sqlite CLI ausführen, um die Datenbank keystore.db zu erstellen.

Benutzen wir wieder die Kommandozeile sqlite3 mit folgendem Befehl:

**\$ sqlite3**

Dies wird uns in den **sqlite> Datenbankmanager** schicken. In diesem ersten öffnen wir die bereits erstellte Datenbank, um sie mit dem folgenden Satz im Manager bearbeiten zu können:

**Sqlite> .open keystore.db**

Dann geben wir die SQL-Anweisung "**CREATE TABLE**" ein, um die **private Schlüsseltabelle zu erstellen**.

Als nächstes werden zwei Möglichkeiten gezeigt, um dieselbe Tabelle zu erstellen. Die eine geht über eine einzige Zeile, in der alle SQL-Anweisungen der Elemente, die sie integrieren, enthalten sind. Das zweite Beispiel besteht in der Einführung jedes Elements, das die Struktur in segmentierter Weise integriert.

**\$ sqlite3**

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

Geben Sie ".help" für Benutzungshinweise ein.

Verbunden mit einer transienten In-Memory-Datenbank.

Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.

**sqlite> .open keystore.db**

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

**sqlite> .quit**

Die Erstellung derselben **Private-Key-Tabelle** wird als Nächstes gezeigt, allerdings durch Einführung der SQL-Anweisung in segmentierter Form:

```
sqlite> TABELLE ERSTELLEN privatschlüssel (
...> id  ganzzahliger Primärschlüssel AUTOINKREMENT
...> alias  VARCHAR(50) NICHT NULL,
...> addrHex VARCHAR (65) NICHT NULL,
...> addrMBC VARCHAR (65) NICHT NULL,
...> adrBin BLOB  NICHT NULL
...> );
sqlite> .tabellen
privatschlüssel
sqlite> .quit
```

Die beiden obigen Beispiele führen zum gleichen Ergebnis. Später führen wir den Satz **.tables** aus, um zu überprüfen, ob die **privatekey-Tabelle** erstellt wurde, am Ende geben wir den Satz **.quit** mit diesem Prozess haben wir bereits die **privatekey-Tabelle** innerhalb der SQLite **keystore.db-Datenbank** erstellt.

Bis zu diesem Zeitpunkt haben wir bereits die Struktur der Datenbank keystore.db und ihre private Schlüsseltabelle, in der die privaten Schlüssel verschlüsselt gespeichert werden.

Dies sollte etwas Ähnliches in dem Knoten (Handy) mit dem TERMUX-Terminal zeigen.



```
$ 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
$
```

SQL-Anweisungen zum Erstellen einer neuen

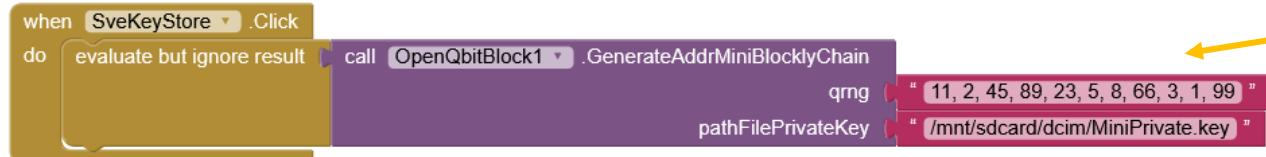
The screenshot shows a Termux terminal window on an Android device. The terminal displays a session in the SQLite command-line interface. The user has entered several commands: '.open keystore.db' to open a database, 'CREATE TABLE privatekey (...)' to define a new table with columns for id, alias, addrHex, addrMBC, and addrBin, '.tables' to list the tables, and '.quit' to exit the interface. The terminal window includes a standard Android keyboard at the bottom. A callout box on the right side of the terminal window contains the text 'SQL-Anweisungen zum Erstellen einer neuen' (SQL statements for creating a new), which corresponds to the highlighted portion of the terminal session where the table creation command is shown.

Wir werden nun einige Sicherheitserweiterungen verwenden, um "private Schlüssel"-Daten einzufügen und diese Daten dann zu konsultieren.

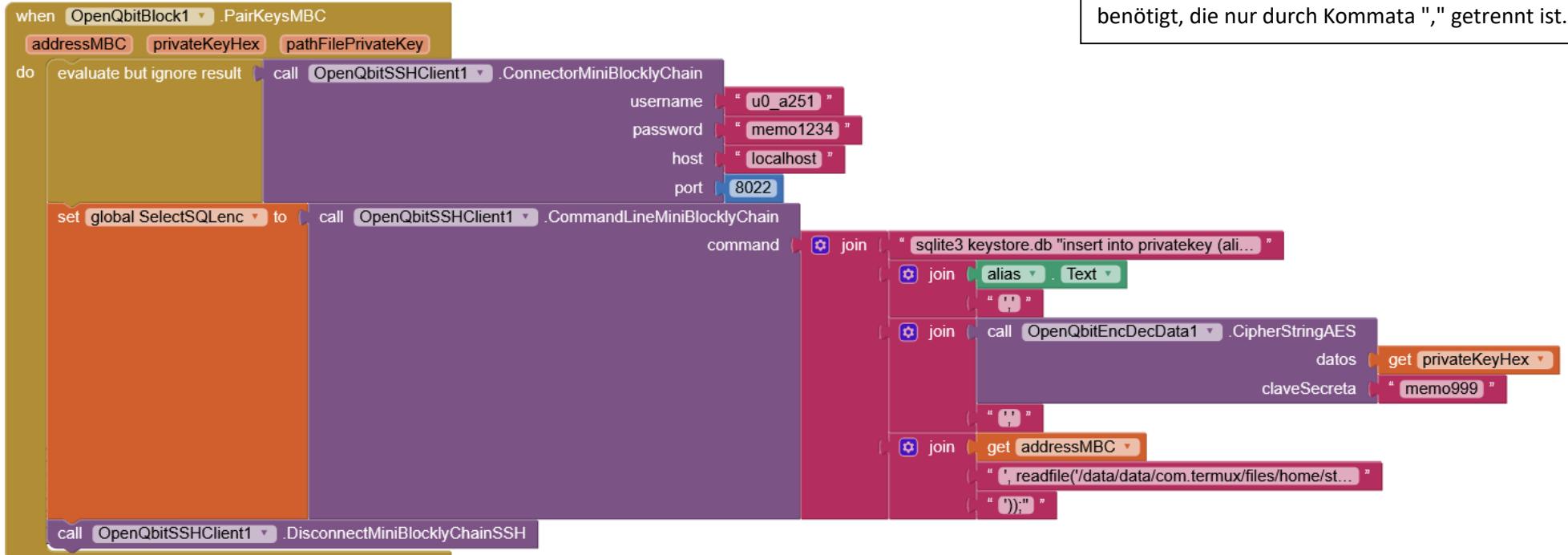
Wir werden den Block verwenden, um eine neue Benutzeradresse zu generieren ([GenerateAddrMiniBlocklyChain](#)). Dieser Block wird uns die Privatadresse in zwei Formaten

(hexadezimal und binär) sowie die öffentliche Adresse im generischen MBC-Adressformat liefern. Wir werden auch die Erweiterung ([OpenQbitSSHClient](#)) und die Erweiterung ([OpenQbitEncDecData](#)) verwenden, um mehr Details zu diesen im Abschnitt "Definition und Verwendung von Sicherheitsblöcken" zu sehen. Auf dem Termux-Terminal sollten Sie den SSH-Dienst ausführen.

INSERT verschlüsselte Daten in die **keystore.db**-Datenbank



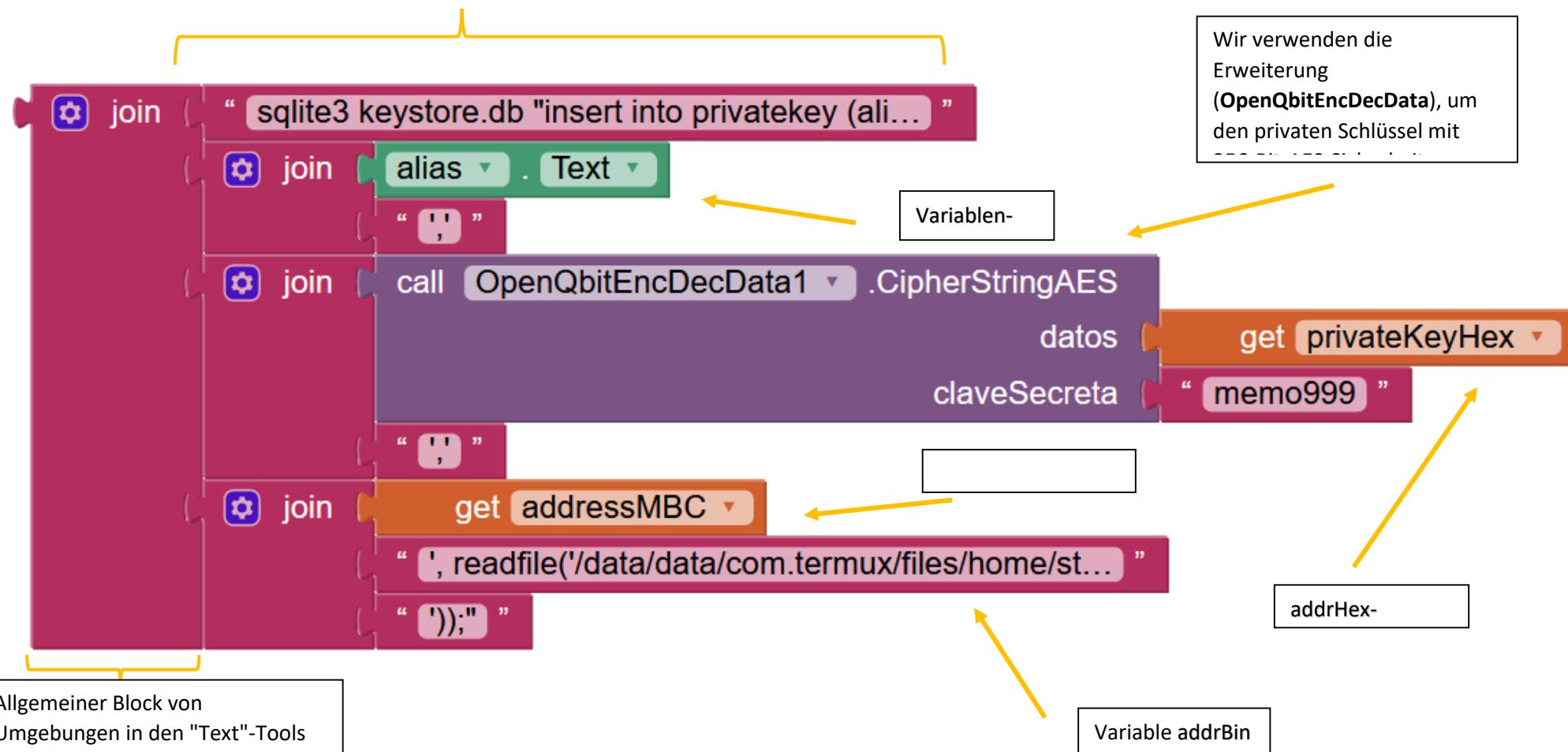
Durch den Block generierte Quantenzufallszahlenreihen (ApiGetQRNGInteger) prüfen, wie das JSON-Ergebnis zu formatieren ist, da der Blockeintrag (GenerateAddrMiniBlocklyChain) eine Zahlenreihe benötigt, die nur durch Kommata "," getrennt ist.



Die Syntax des Befehls ist sehr wichtig, wir müssen den folgenden Befehl im richtigen Format in der Blockly-Umgebung einführen.

Die Werte der Werte werden immer die Variablen sein, Beispiel:

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



Nach der Ausführung der Blöcke erhalten wir in der privatekey-Tabelle der keystore.db-Datenbank ein Ergebnis ähnlich dem folgenden:

Wir geben den Sqlite-Handler in das Termux-Terminal ein, öffnen die keystore.db-Basis und führen den Satz aus:

\$ sqlite3

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

Geben Sie ".help" für Benutzungshinweise ein.

Verbunden mit einer transienten In-Memory-Datenbank.

Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.

sqlite> .open keystore.db

sqlite> wählen Sie \* von 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|mexiko|07JBBizTcC0Ce6a8Pwe5aTV41Ql1DKiUQZyPSfJuRbVUZnvIwQJcry4LilBMs5jHavQeMo1iN8rCO87V5Xka2YaKR4fswopeTQmI/Q+ipzI=|2eENpFt2HSuGtXNXoeiwfrp5d6e87Z7y5|0♦♦
sqlite>

ESC   CTRL   ALT   -   +   ↑
1  2  3  4  5  6  7  8  9  0
+  ×  ÷  =  %  /  \  $  €  £
@  *  !  #  :  ;  &  _  (  )
<  >  -  '  "  ,  .  ?  ←
abc  ☺  ↴  ↳  ⏪  ⏩  ⏴  ⏵

```

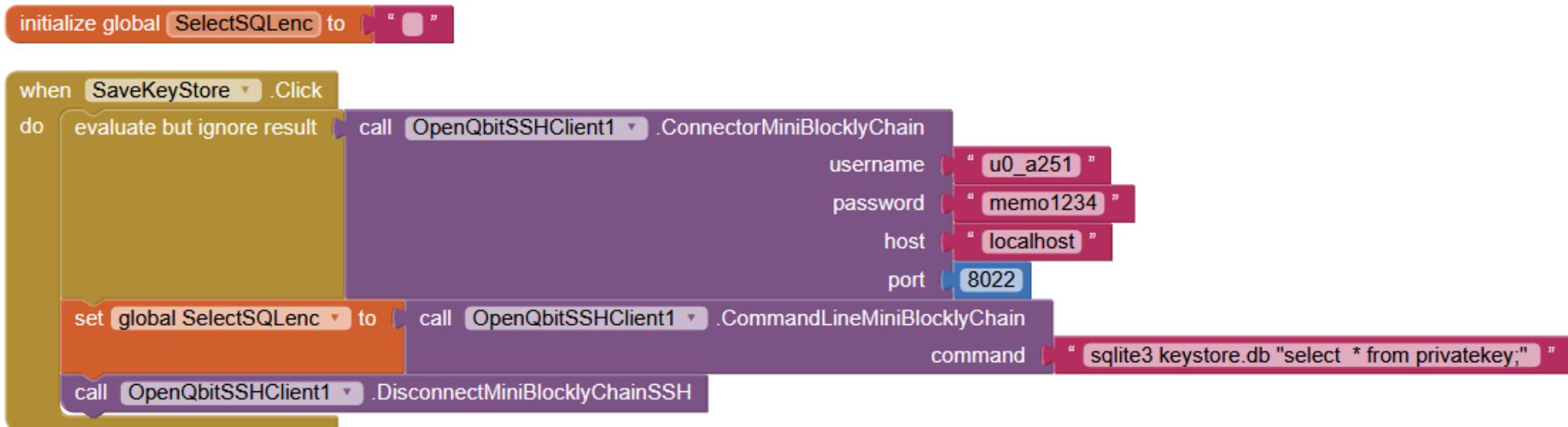
Alias: mexiko

addrBin (privater Schlüssel in)

addrHex-

addrMBC-Adresse

BERATEN Sie alle Daten in der **Privatekey-Tabelle** der Datenbank SQLite **keystore.db**



BERATEN Sie die Aliase der Daten in der **Privatekey-Tabelle** der SQLite **keystore.db**-Datenbank

sqlite3 keystore.db "Alias aus privatem Schlüssel auswählen;"

**Abfrage** aller Daten in der Spalte addrHex, die in der **Privatekey-Tabelle** der Datenbank SQLite **keystore.db** verschlüsselt sind

sqlite3 keystore.db "select addrHex from privatekey;"

**CONSULT** zum Abrufen des privaten addrBin-Schlüssels in der **Private-Key-Tabelle** der Datenbank SQLite **keystore.db**

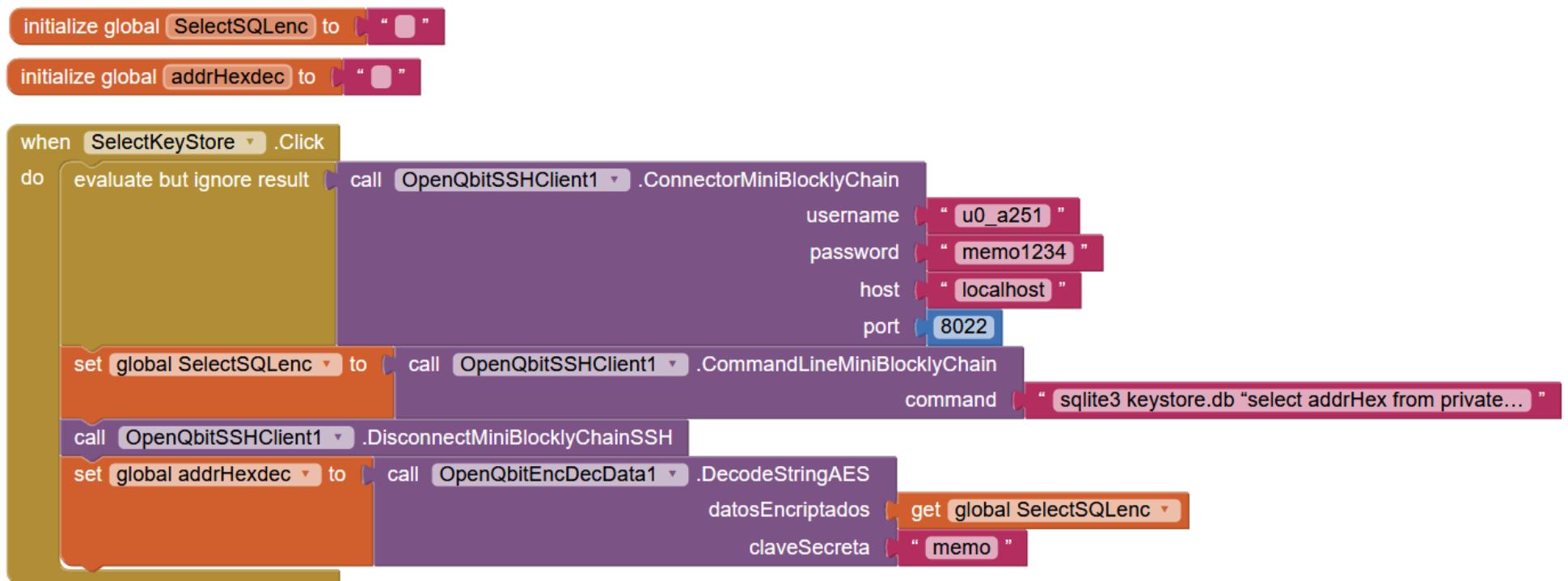
sqlite3 **writefile('PrivateKey.key', addrBin)** FROM privatekey WHERE alias='mexico'; (2)

- (1) Bei dieser Art von Abfrage wird vor allem der private Schlüssel abgefragt, um den Prozess der Unterzeichnung der Transaktionen durchzuführen.

Abfrage zur Entschlüsselung von Daten nach Alias in der Spalte addrHex der **privatekey**-Tabelle der Datenbank SQLite **keystore.db**

In diesem Fall werden wir den Block (**DecodeStringAES**) verwenden, um die in der Spalte "addrHex" gespeicherten Daten zu dekodieren.

sqlite3 keystore.db "select addrHex from privatekey where='mexico';"



Diese Konsultation gibt uns den privaten Schlüssel im hexadezimalen Format, dies ist der grundlegende Teil jedes Empfangens oder Sendens von Transaktionen. Es wird wiederholt empfohlen, ein Backup dieser keystore.db-Datenbank aufzubewahren.

Datenbankdesign **publickeys.db** mit Tabelle **publicaddr**:

```
$ sqlite3
```

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

```
Geben Sie ".help" für Benutzungshinweise ein.
```

```
Verbunden mit einer transienten In-Memory-Datenbank.
```

```
Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.
```

```
sqlite> .open publickeys.db
```

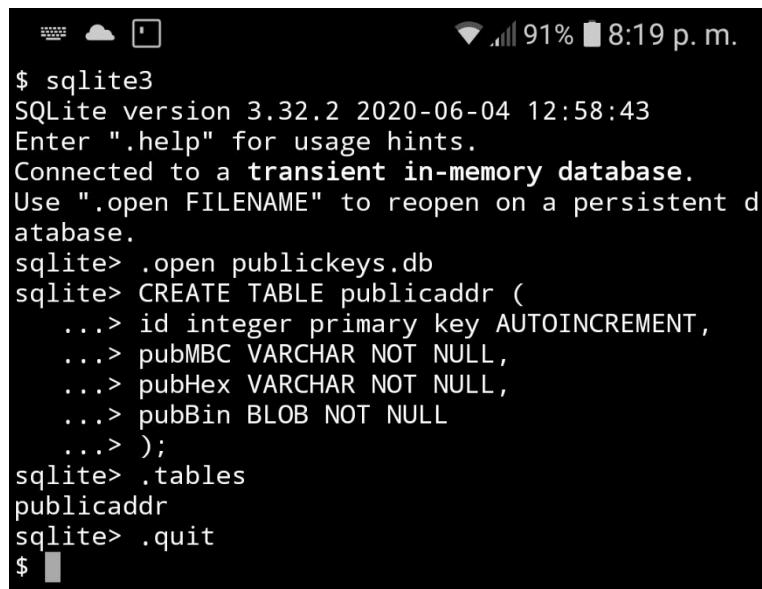
```
sqlite> TABLE TABLE ERSTELLEN publicaddr (id integer primary key AUTOINCREMENT NOT
```

```
NULL, pubMBC VARCHAR NOT NULL, pubHex VARCHAR NOT NULL, pubBin BLOB NOT NULL);
```

```
sqlite> .quit
```

Als nächstes wird die Erstellung der **veröffentlichten** Tabelle mit dem folgenden SQL-Satz in segmentierter Form gezeigt:

```
sqlite> TABELLE ERSTELLEN veröffentlichtaddr (
...> id ganzzahliger Primärschlüssel AUTOINKREMENT
...> pubMBC VARCHAR NICHT NULL,
...> pubHex VARCHAR NICHT NULL,
...> pubBin BLOB NICHT NULL
...> );
sqlite> .tabellen
publishedaddr
sqlite> .quit
```



The screenshot shows a terminal window on a mobile device. The status bar at the top indicates signal strength, battery level (91%), and the time (8:19 p.m.). The terminal prompt is '\$'. The user enters the command '\$ sqlite3' followed by the SQLite version information. Then, they enter the command 'CREATE TABLE publicaddr (' followed by the column definitions: 'id integer primary key AUTOINCREMENT, pubMBC VARCHAR NOT NULL, pubHex VARCHAR NOT NULL, pubBin BLOB NOT NULL'. Finally, they enter ')'; to complete the table definition. After this, they run '.tables' to list the tables, which shows 'publicaddr'. They then exit the SQLite shell with '.quit'.

```
$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open publickeys.db
sqlite> CREATE TABLE publicaddr (
...> id integer primary key AUTOINCREMENT,
...> pubMBC VARCHAR NOT NULL,
...> pubHex VARCHAR NOT NULL,
...> pubBin BLOB NOT NULL
...> );
sqlite> .tables
publicaddr
sqlite> .quit
$
```

## 24. Anhang "RESTful SQLite GET/POST-Befehle".

Als nächstes werden die verschiedenen Befehle gezeigt, die wir in der Restful SQLite des Sicherungsnetzwerks verwenden können. Diese wurden aus der ursprünglichen Entwicklung von GITHUB (<https://github.com/olsonpm/sqlite-to-rest>) konsultiert.

CRUD-Operationen (Erstellen, Lesen, Aktualisieren und Löschen) RESTful.

Im Folgenden finden Sie eine Liste der verfügbaren Operationen, die von der RESTful-API in Form von Pseudo-Beispielen zur Verfügung gestellt werden. Wir nehmen eine Basis- "op.sqlite" und zwei zugehörige "trans"-Tabellen für Transaktionen und ein weiteres "Vorzeichen" für Quelladresse, Zieladresse und Vermögenswert mit zwei Spalten id INTEGER PRIMARY KEY an.

Wie bei den Einschränkungen erwähnt, beachten Sie bitte, dass die Methoden (DELETE und POST) jeweils nur eine Zeile zur gleichen Zeit betreffen können.

OBTAIIN Dies ermöglicht die größte Variation. Später werden wir alle verfügbaren Abfrageoperatoren sehen.

Überschriften können direkt unterhalb der URLs angegeben werden.

### **/transAnforderungen**

für alle Zeilen

### **/trans?id=1Wobei**

id = 1

### **/trans**

Bereich: Zeilen=0-2Erste  
drei Zeilen

### **/trans**

Bereich: Zeilen=-5Letzte  
fünf Zeilen

### **/trans**

Bereich: Zeilen=0-

So viele Zeilen, wie der Server zur Verfügung stellen kann, was in der Praxis der kleinere Wert von maxRange und Gesamtzahl der Zeilen sein wird.

### **/trans**

Bereich: Zeilen=1-

So viele Zeilen wie der Server bereitstellen kann, beginnend mit Zeile 1.

**/trans**

Reihenfolge:

Name Aufsteigend nach Name

**/trans**

Reihenfolge: Name absteigendOrdnung

nach Name absteigend

**/trans**

Reihenfolge: Name absteigend, id

Beigesteuert, aber zunächst nach absteigendem Namen und im Falle eines Gleichstandes nach aufsteigender ID sortiert.

**/trans?id>1**

Wo id > 1

**/trans?id>=2&id<5**

wobei id >= 2 und id < 5

**/trans?name\_NOTNULL**

Wo Name nicht null ist

**/trans?name\_ISNULLWo**

Name null

ist

**/trans?id!=5&name\_LIKE'Gefleckt%'**

wobei id != 5 und der Name LIKE "Spotted%" ist (Anführungszeichen ignorieren)

**/trans?id>=1&id<10&name\_LIKE'Avery%'**

Reihenfolge: Name desc, id Bereich: Zeilen=2-4

Er hat um des Beispiels willen einen Beitrag geleistet.

Erhalten Sie Transaktionen mit Identifikatoren zwischen 1 und 9 einschließlich, mit einem Namen wie "Avery%", zuerst nach absteigendem Namen und dann nach aufsteigendem Identifikator geordnet, wodurch Sie die dritte bis fünfte Zeile des Ergebnisses erhalten. Oder in SQL:

DELETE Erfordert eine Abfragezeichenfolge, bei der alle Primärschlüssel gleich einem Wert gesetzt sind. Dies erfordert eine maximal einzeilige Löschung.

**/trans?id=1Löscht**

trans mit id=1

Wenn die Transaktion stattdessen einen Hauptschlüssel hatte, der sich aus id und Name zusammensetzt.

**/trans?id=1&name='Avery IPA'.**

POST erstellen

Sie dürfen keinen Query-String übergeben. Wenn eine Abfragezeichenfolge übergeben wird, wird eine POST-Aktualisierung angenommen. Alle POST-Anforderungen müssen den Header-Inhaltstyp: application / json übergeben.

Bitte beachten Sie, dass der Hauptteil alle nicht löschenbaren PRIMARY KEY-Spalten enthalten muss und nicht INTEGRIEREN. Andernfalls wird eine Antwort von 400 gesendet, die angibt, welche Felder verloren gegangen sind. Die löschenbaren Spalten werden null sein und die INTEGER PRIMARY KEY-Spalten werden automatisch gemäß den sqlite3-Spezifikationen erhöht.

Die JSON-Daten werden direkt unterhalb der URLs angegeben.

**/trans**  
{"id":1,"name":"Serendipity"} Erstellt  
einen Trans mit id = 1 und name = "Serendipity".

**/trans**  
{"id":1}Erzeugt  
einen Trans mit id = 1 und Name = NULL

**/trans**  
{"Name": "Serendipity" }

Erstellt eine Transaktion, bei der id auf den folgenden Wert, erhöht um sqlite3 Specifications INTEGER PRIMARY KEY, gesetzt ist.

**/trans**  
{ }

Erstellt eine Transaktion mit erhöhter id und dem auf NULL gesetzten Namen.

POST-Aktualisierung

Sie muss eine Abfragezeichenfolge enthalten. Ohne eine Abfragezeichenfolge wird die POST-Erstellung angenommen. Wie bei der POST-Erstellung ist der Header-Inhaltstyp: application / json erforderlich.

Die Abfragezeichenfolge muss alle Primärschlüsse enthalten, um sicherzustellen, dass nur eine Zeile aktualisiert wird. Wenn falsche Werte weitergegeben werden, wird eine 400 mit den beanstandeten Schlüsseln zurückgegeben.

Der Hauptteil der Anforderung muss ein nicht leeres Objekt enthalten und muss gültige Schlüssel enthalten, die den Spaltennamen entsprechen.

Die JSON-Daten werden direkt unterhalb der URLs angegeben.

**/trans?id=1**

{ "id":2 }

Aktualisieren Sie die Transaktion mit einer ID von 1, indem Sie sie auf zwei setzen.

**/trans?id=1**

{ "Name ":" MCBza45Rt56cvbfdR2Swd788kj" }

Aktualisieren Sie die Transaktion mit der ID von 1, indem Sie ihren Namen oder Wert auf "MCBza45Rt56cvbfdR2Swd788kj" setzen.

Wenn der Handelsschalter stattdessen einen Hauptschlüssel hätte, der sich aus ID und Name zusammensetzt.

**/trans?id=1&name=MCBza45Rt56cvbfdR2Swd788kj**

{ "Name ":" MCB3ofFG5Hj678MNb09vLdfaasx " }

Aktualisieren Sie die Transaktion, bei der id eins ist und die Adresse MCBza45Rt56cvbfdR2Swd788kj lautet, und setzen Sie die MCB3vonFG5Hj678MNb09vLdfaasx.

Referenz

isSqliteFile

Überprüfen Sie einfach die ersten 16 Bytes der Datei, um zu sehen, ob sie gleich 'Sqlite-Format 3' ist, gefolgt von einem Null-Byte.

isVerzeichnis

Gibt das Ergebnis von fs.statsSync gefolgt von .isDirectory zurück

isDatei

Gibt das Ergebnis von fs.statsSync gefolgt von ..isFile zurück

GET-Beratungsoperatoren

Die Anfragebedingungen müssen mit Verbindungssymbolen gekennzeichnet werden, z.B. id>  
5 & name = MCBza45Rt56cvbfdR2Swd788kj

Binäre Operatoren (erfordert einen Wert nach) Man.

=

!=

>=

<=  
>  
<  
**Z.B.**

LIKE ist etwas Besonderes, weil es einfache Anfangs- und Schlussquotierungen haben muss. Andernfalls wird ein 400-Fehler generiert, der anzeigt, wo die Analyse nicht abgeschlossen werden konnte und was erwartet wurde. Siehe CRUD RESTful Operations für Beispiele.

Einzelne Operatoren (müssen dem Namen einer Spalte folgen)

IST NULL

NICHT NULL

Router-Konfigurationsobjekt

isLadenPlainObjekt

Der Zweck dieses Objekts ist es, eine generische Konfiguration für den Sqlite-Router bereitzustellen. Die folgenden Eigenschaften sind zulässig:

prefix: isLadenString Die Zeichenfolge, die an die Option koa-router prefix builder übergeben wird. Beispielsweise gibt der Skeleton-Server kein Präfix an, wodurch die Bier-API direkt von der Wurzel der http-Domain getroffen werden kann: // localhost: 8085 / trans. Wenn Sie das Präfix auf '/ api' setzen, dann sollten Sie Anfragen an http: // localhost: 8085 / api / trans senden.

allTablesAndViews: ein tabellarisches Konfigurationsobjekt

Die in diesem Objekt angegebenen Einstellungen werden auf alle Tabellen und Views angewendet, optional überschrieben durch die Eigenschaft tablesAndViews.

tablesAndViews: isLadenPlainObject Das vergangene Objekt muss Schlüssel haben, die mit der Datenbankspalte übereinstimmen oder die Namen anzeigen. Andernfalls wird eine freundliche Fehlermeldung ausgegeben. Die Werte für jede Tabelle und jeden View müssen ein tabellarisches Konfigurationsobjekt sein.

Tabellarisches Konfigurationsobjekt

isLadenPlainObject Dieses Objekt repräsentiert Einstellungen, die für Ansichten oder Tabellen festgelegt werden können. Es ermöglicht die folgenden Eigenschaften:

maxRange: isPositivNummer

Standardanwendung: 1000

Dies ist die maximale Reichweite, die Ihr Server Anfragen zulässt. Wenn eine GET-Anfrage ohne Bereichsüberschrift eintrifft, geht die Spezifikation davon aus, dass Sie die gesamte Ressource wünschen. Wenn die Anzahl der Zeilen, die zu GET führen, größer als maxRange ist, wird ein Status 416 mit dem benutzerdefinierten Max-Range-Header zurückgegeben. Der Standardwert der Anwendung ist absichtlich konservativ, in der Hoffnung, dass die Autoren maxRange entsprechend ihren Bedürfnissen einstellen werden.

Bitte beachten Sie, dass 'Unendlich' eine gültige positive Zahl ist.

Flaggen: isLadenArray

Derzeit ist das einzige akzeptierte Kennzeichen die Zeichenfolge 'sendContentRangesInHEAD'. Wenn diese Option gesetzt ist, wird bei HEAD-Anfragen der Bereich der verfügbaren Inhalte in der Form Content-range: \* / <max-range> zurückgegeben. Der Grund dafür, dass sie konfigurierbar ist, liegt darin, dass die Berechnung der maximalen Reichweite je nach der Serverlast und der Größe Ihrer Tabellen mehr Arbeit sein kann, als sie wert ist.

Benutzerdefinierte Kopfzeilen

Bewerbung

ORDER: Dieser Header ist nur für GET definiert und kann als Äquivalent von sql ORDER BY betrachtet werden. Sie muss einen durch Komma getrennten Spaltennamen enthalten, optional jeweils gefolgt von einem Leerzeichen und den Zeichenfolgen 'asc' oder 'desc'. Wenn falsche Bestellwerte gesendet werden, wird in einer 400er-Antwort angegeben, welche.

Antwort

Nicht alle sind notwendigerweise kundenspezifisch, aber alle Anwendungen liegen außerhalb der Spezifikation und bedürfen daher der Klärung.

GET

max-range: Dieser Header wird zurückgegeben, wenn die Anzahl der angeforderten Zeilen den konfigurierten maxRange überschreitet. Beachten Sie, dass die Anfrage möglicherweise nicht den Bereichskopf angibt, aber die Anzahl der Zeilen, die zu dieser Ressource führen, wird trotzdem überprüft.

inhaltsbereich: rfc7233-staaten

Nur die Statuscodes 206 (Partieller Inhalt) und 416 (Unbefriedigender Bereich) beschreiben eine Bedeutung für Content-Range.

Wenn Sqlite-to-rest mit einem Statuscode 200 antwortet, wird der Inhaltbereichskopf im Format 206 im Format <Zeilenanfang> - <Zeilenende> / <Zeilenanzahl> gesendet.

Wenn eine Anfrage ohne Bereichskopf gesendet wird und die resultierende Anzahl von Zeilen maxRange überschreitet, wird eine 400 zurückgegeben, wobei der Inhaltsbereich auf das 416-Format von \* / <Zeilenanzahl> gesetzt ist.

Beachten Sie, dass dieser Header in einer HEAD-Antwort zurückgegeben werden kann.

accept-order: wird zurückgegeben, wenn die Reihenfolge des Anfragekopfes eine falsche Syntax oder falsche Spaltennamen angegeben hat. Weitere Einzelheiten finden Sie unten unter HEAD -> Bestellung annehmen.

## 25. Anhang "Java Code SQLite-Redis Connector".

Ein Code der Verbindung zwischen der Kommunikation der beiden SQLite-Redis-Datenbanken ist unten dargestellt. Im Grunde, wie wir sehen können, ändert der Konnektor das SQLite-Format in eine generische Zeichenfolge der Daten (Transaktionen), die Redis empfangen soll.

Der obige Prozess fungiert als Transaktionswarteschlangen-Trigger für alle Knoten, die verbunden sind und mögliche Kandidaten für die Verarbeitung der aktuellen Transaktionswarteschlange sind.

Wenn die Redis-Datenbank die Transaktionswarteschlange durch ihre Master-Slave-Konfiguration erhält, verteilt sie die Informationen in Echtzeit und gleichmäßig an alle Knoten.

Ein weiterer grundlegender Punkt ist, dass alle Knoten in ihrer Uhr synchronisiert werden müssen, dies wird, wie wir zuvor gesehen haben, mit der Funktionalität der Anfrage an einen Serverpool NTP (Network Time Protocol) geschehen.

Dieser Konnektor führt auch die erste Ebene der Datensicherheitsprüfung durch, indem er den Merkle-Root-Baum aller Transaktionen berechnet.

Basiscode des SQLite-Redis Connectors.

```
java.sql.* importieren;  
java.util.* importieren;  
java.util.arrays importieren;  
java.util.list importieren;  
java.util.array importieren;  
java.security.* importieren;  
java.security.messageDigest importieren;  
redis.kunden.jedis.jedis.Jedis importieren;  
Klasse RediSqlite{
```

```

öffentlicher String previousHash;
öffentlicher String merkleRoot;
öffentliche statische ArrayList<String> Transaktionen = neue ArrayList<String>();
öffentliche statische ArrayList<String> treeLayer = neue ArrayList<String>();
öffentlicher statischer String getMerkleRoot() {
    int count = transaktionen.größe();
    int-Punkt = 1;
    Ungerade int = 0;
    ArrayList<String> previousTreeLayer = Transaktionen;
    while(Anzahl > 1) {
        treeLayer = neue ArrayList<Zeichenfolge>();
        for(int i=1; i <= zählen ; i=i+2) {

            if (!esPar(count) && i == count) ungerade = 1;
            treeLayer.add(applySha256(vorherigerTreeLayer.get(i-item)
+ vorherigerTreeLayer.get(i-impar)));

        }
        Punkt = 1;
        Ungerade = 0;
        count = treeLayer.size();
        previousTreeLayer = treeLayer;
    }
    Zeichenfolge merkleRoot = (treeLayer.size() == 1) ? treeLayer.get(0) : "";
    Rückkehr merkleRoot;
}
//Definieren Sie, ob Merkle-Baum gerade oder ungerade ist
statische boolesche enPar(int Zahl){
    if (numero%2==0) gibt true zurück; andernfalls false;
}

öffentlicher statischer String applySha256(String-Eingabe){
    versuchen {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        //Anwendung von sha256 auf unseren Input,
        byte[] hash = digest.digest(input.getBytes("UTF-8"));
        StringBuffer hexString = new StringBuffer(); // Dies wird Hash als hexadezimale Zahl
        enthalten
        für (int i = 0; i < Hash.länge; i++) {
            String hex = Ganzzahl.toHexString(0xff & hash[i]);
            if(hex.Länge() == 1) hexString.anhängen('0');
        }
    }
}

```

```
    hexString.append(hex);
}
gibt hexString.toString() zurück;
}
catch(Ausnahme e) {
    neue RuntimeException(e) werfen;
}
}

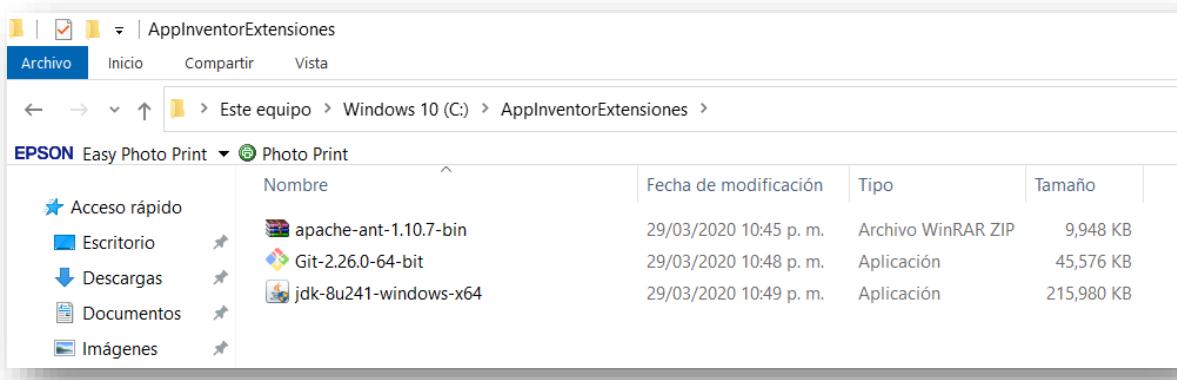
öffentliche statische Leere main(String args[]){
    versuchen{
        Verbindung      con=DriverManager.getConnection("jdbc:sqlite:C://memo/sqlite-tools-
win32-x86-3310100/trans.sqlite3");
        //Verbindung zum Redis-Server auf dem Localhost
        jedis = neue jedis ("lokaler Wirt");
        jedis.auth ("memo1234");
        Anweisung stmt=con.createStatement();
        String sql = "SELECT * FROM brewery";
        ErgebnisSet rs=stmt.executeQuery(sql);
        while(rs.next()) {
            transaktionen.hinzufügen(rs.getString(2));
        jedis.set
        ("LATAM:"+String.valueOf(rs.getInt(1)), "["+"\""+rs.getString(2)+"\""+","+"\""+rs.getString(3)
        +"\""+","+"\""+rs.getString(4)+"\""+"));
        }
        jedis.set("LATAM:merkleroot", getMerkleRoot());
        System.out.println("Anzahl der ArrayList-Elemente:: "+treeLayer.size());
        mit .close();
    }catch(Ausnahme e){ System.out.println(e);}
}

}
```

## 26. Anhang "Mini-BlocklyChain für Entwickler".

In diesem Anhang werden wir die Konfiguration, die Installation und die grundlegende Verwendung der Generierung externer Module auf der Grundlage der Java-Programmiersprache für die Blockly-Umgebung überprüfen. Wir werden in der Lage sein, spezialisierte Module für jeden Geschäftsfall zu erstellen und Funktionalitäten in das Mini-BlocklyChain-System einzufügen oder andere Funktionalitäten zu unserer mobilen Anwendung hinzuzufügen.

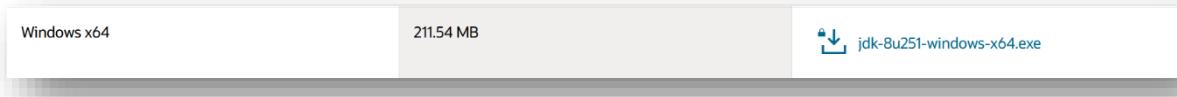
Wir haben in unserem Windows-System ein Verzeichnis namens "ApplInventorExtensions" erstellt, das die folgenden öffentlichen Softwarepakete herunterladen wird.



1.- Wir werden die neueste Version des **JDK (Java Development Kit)** herunterladen.

Beispiel: jdk-8u251-fenster-x64.exe (211 MB)

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>



2.- Apache-Ant-Bibliothek, die JAVA zum Erstellen von Anwendungen verwendet, <http://ant.apache.org/bindownload.cgi>, in meinem Fall habe ich Ant 1.10.8 (Binärdistributionen) heruntergeladen (apache-ant-1.10.8-bin.zip). Es kann fortgeschrittenere Versionen geben.

### 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.- Wir haben den Git Bash von Ihrer Website <https://git-scm.com/download/win> installiert.

## 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.- Entpacken Sie "Apache Ant." Wenn Sie mit dem Entpacken fertig sind, können Sie es z.B. in einem doppelten Ordner tun:

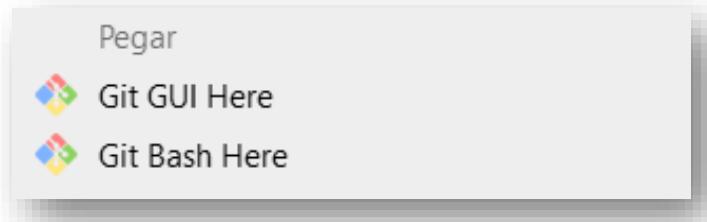
C:\AppInventorExtensions\apache-ant-1.10.8-bin\apache-ant-1.10.8-bin

- Ändern Sie ihn in C:\Apache-ant-1.10.8-bin

Nombre	Fecha de modificación	Tipo	Tamaño
bin	01/09/2019 11:43 a. m.	Carpeta de archivos	
etc	01/09/2019 11:43 a. m.	Carpeta de archivos	
lib	01/09/2019 11:43 a. m.	Carpeta de archivos	
manual	01/09/2019 11:43 a. m.	Carpeta de archivos	
CONTRIBUTORS	01/09/2019 11:43 a. m.	Archivo	7 KB
contributors	01/09/2019 11:43 a. m.	Documento XML	33 KB
fetch	01/09/2019 11:43 a. m.	Documento XML	14 KB
get-m2	01/09/2019 11:43 a. m.	Documento XML	5 KB
INSTALL	01/09/2019 11:43 a. m.	Archivo	1 KB
KEYS	01/09/2019 11:43 a. m.	Archivo	94 KB
LICENSE	01/09/2019 11:43 a. m.	Archivo	15 KB
NOTICE	01/09/2019 11:43 a. m.	Archivo	1 KB
patch	01/09/2019 11:43 a. m.	Documento XML	2 KB
README	01/09/2019 11:43 a. m.	Archivo	5 KB
WHATSNEW	01/09/2019 11:43 a. m.	Archivo	250 KB

5.- Wir haben Git Bash installiert. Wir haben alles standardmäßig in der Installation belassen.

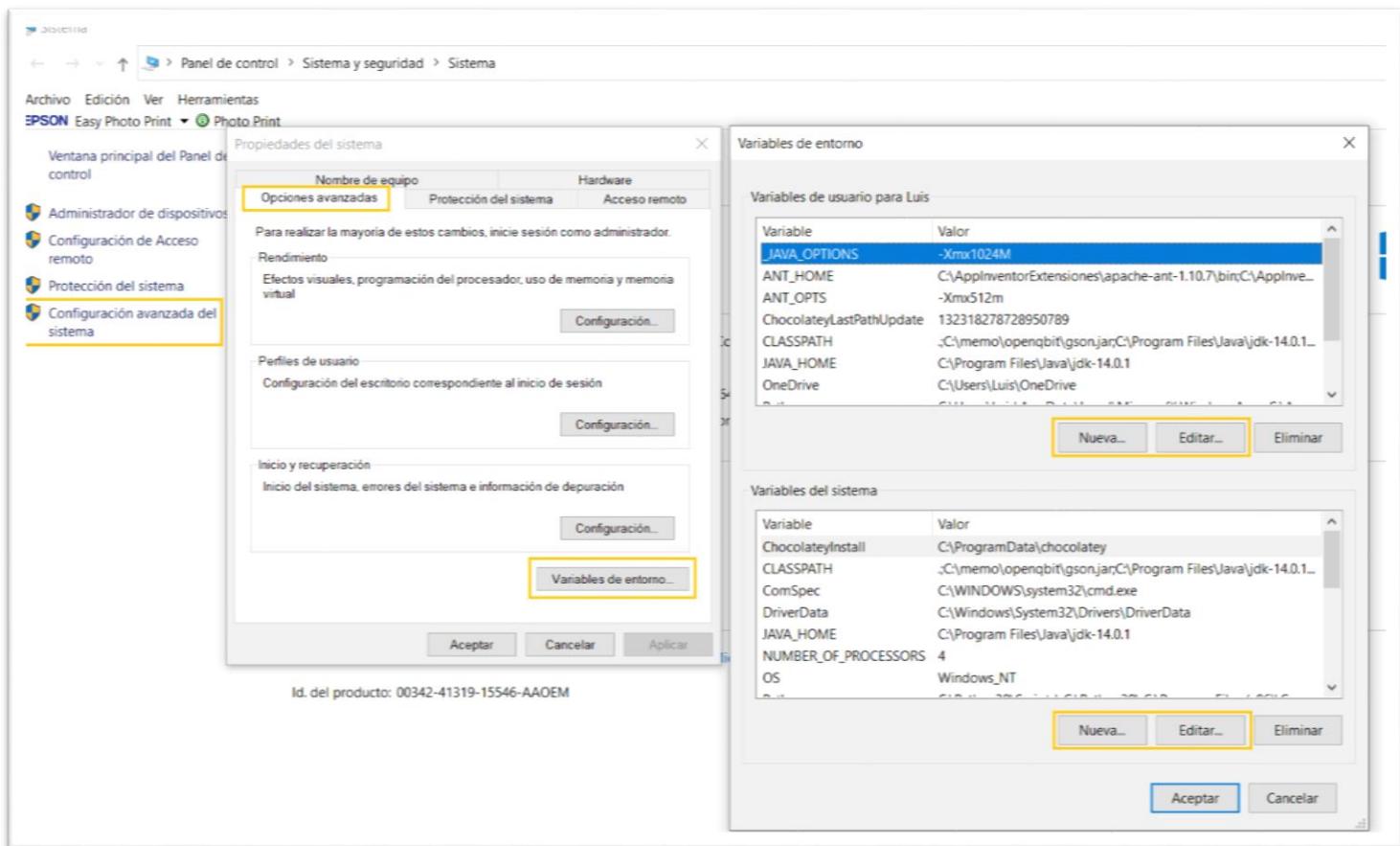
- Später können wir sehen, dass es einen Link in der Windows-Startschaltfläche hat, indem wir auf die linke Schaltfläche im Dateibrowser klicken.



6.- Wir haben das Java SE Development Kit installiert. Je nach Windows-Version müssen Sie Ihren Computer wahrscheinlich neu starten.

Windows-System-Umgebungsvariablen. Wir werden die Umgebungsvariablen erstellen. Dazu werden wir das tun:

Systemsteuerung -> System -> Erweiterte Systemeinstellungen -> Erweiterte Optionen -> Umgebungsvariablen.



Je nachdem, ob wir eine neue Variable einfügen oder eine bestehende Variable bearbeiten wollen, drücken wir die Schaltfläche "Neu..." oder "Bearbeiten...".

Um Adressen zu den bereits festgelegten hinzuzufügen, trennen wir sie durch Semikolon;

In der Sektion Benutzervariablen für John haben wir diese neuen...

JAVA\_OPTIONS wir setzen Sie von Wert -Xmx1024m

ANT\_HOME setzen wir von Wert C:\AppInventorExtensions\apache-ant-1.10.8-bin [d.h. der Ordner, in dem wir apache-ant dekomprimiert haben].

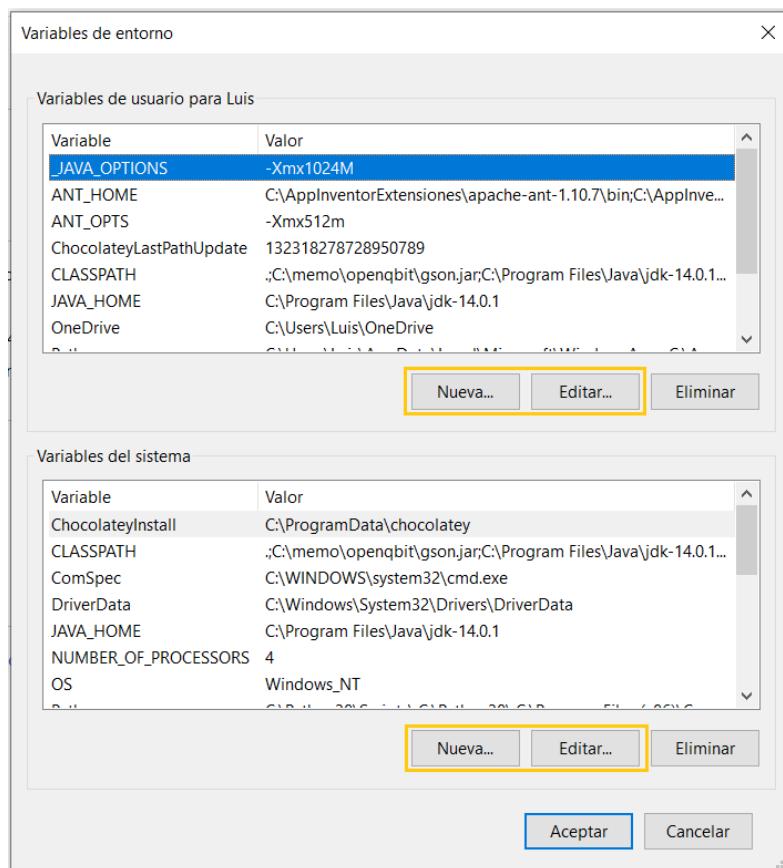
ANT\_OPTS setzen wir von Wert -Xmx256M

JAVA\_HOME ist auf C:\Program Files\jdk1.8.0\_131 [Wenn er einen anderen Wert hat, ändern Sie ihn. Beachten Sie, dass er jdk NOT jdr lautet] eingestellt.

CLASSPATH legen wir von Wert %ANT\_HOME%\lib;%JAVA\_HOME%\lib

In PATH haben wir ;%ANT\_HOME%\bin;%JAVA\_HOME%\bin [Beachten Sie, dass ; mit einem Semikolon beginnt; um die bereits vorhandenen zu ergänzen] hinzugefügt.

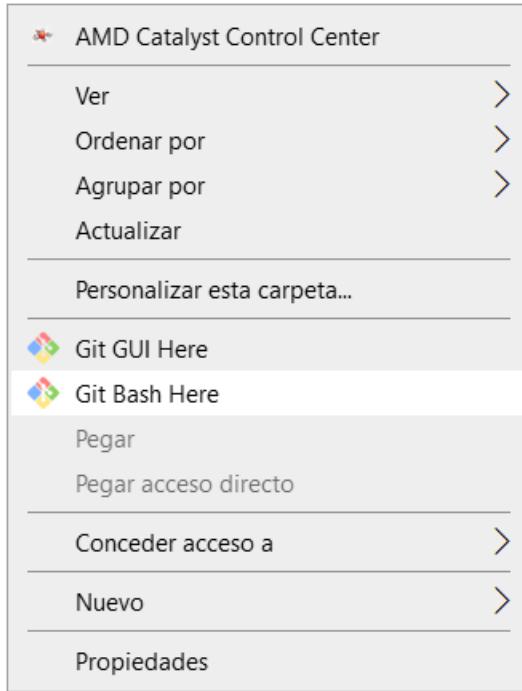
**ANMERKUNG:** Variablen werden durch ein Semikolon voneinander getrennt: Variable-Eins; Variable-N; Variable-N+1



Erstellung des App Inventor-Klons in unserem Computer

Wir werden eine "Klon"-Kopie von App Inventor auf unserem Server (PC) erstellen, sie direkt aus dem Internet herunterladen und diese Kopie erstellen.

Dazu verwenden wir die Anwendung **Git Bash** und klicken darauf, um ein Terminal zu öffnen.



Wir führen den Befehl auf dem Git Bash Terminal aus:

```
$ git-Klon https://github.com/mit-cml/appinventor-sources.git
```

```
uis@DESKTOP-LLGPLR6 MINGW64 ~
git clone https://github.com/mit-cml/appinventor-sources.git
Cloning into 'appinventor-sources'...
remote: Counting objects: 41191, done.
remote: Total 41191 (delta 0), reused 0 (delta 0), pack-reused 41190
receiving objects: 100% (41191/41191), 553.30 MiB | 494.00 KiB/s, done.
resolving deltas: 100% (21999/21999), done.
Checking out files: 100% (1758/1758), done.
uis@DESKTOP-LLGPLR6 MINGW64 ~
```

Der Standort, an dem sich das Repository befindet:

<https://github.com/mit-cml/appinventor-sources/>

Es wird ein Ordner namens appinventor-sources mit der App Inventor-Quelle erstellt.

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

Wir haben das folgende Verzeichnis im Pfad C erstellt: Users - Appinventor-sourcesv-babkup - Appinventor-components -rc-openqbit

Darin haben wir unser folgendes Testprogramm namens OpenQbitQRNGch.java kopiert

Nombre	Fecha de modificación	Tipo	Tamaño
OpenQbitQRNGch	11/06/2020 01:39 a. m.	Archivo JAVA	5 KB

Erstellung der Erweiterung.

**RESPEKTIEREN SIE DIE KAPS UND MINUSKULÄRE, es ist nicht dasselbe Hallo und Hallo.**

Setzen Sie keine Akzente. Wir sind bereit für unsere erste Erweiterung, den Quantenzufallszahlengenerator.

Wir verwenden einen Texteditor, Notepad++, wir erstellen eine Datei namens OpenQbitQRNGch.java, die zufällige Quantenzahlen mit dem folgenden Code generiert:

```
// Diese Erweiterung wird zur Ausgabe des Quantum Random Number Generator
QRNG Switzerland API verwendet.

Paket com.openqbit. OpenQbitQRNGch;
com.google.appinventor.components.annotations.DesignerComponent
importieren;
com.google.appinventor.components.annotations.DesignerProperty
importieren;
com.google.appinventor.components.annotations.PropertyCategory
importieren;
com.google.appinventor.components.annotations.SimpleEvent importieren;
com.google.appinventor.components.annotations.SimpleFunction importieren;
com.google.appinventor.components.annotations.SimpleObject importieren;
com.google.appinventor.components.annotations.SimpleProperty importieren;
com.google.appinventor.components.common.ComponentCategory importieren;
com.google.appinventor.components.common.PropertyTypeConstants
importieren;
com.google.appinventor.components.runtime.util.MediaUtil importieren;
com.google.appinventor.komponenten.laufzeit.* importieren;
java.io.* importieren;

@DesignerComponent(Version = OpenQbitQRNGch.VERSION,
    Beschreibung = "API-Quantenzufallszahlengenerator".
Quantenzufallszahlen als Dienst, QRNGaaS, Web API für den
Quantenzufallszahlengenerator von Quantis, entwickelt von der Schweizer
Firma - " + "Guillermo Vidal - OpenQbit.com",
    Kategorie = KomponenteKategorie.ERWEITERUNG
    nicht sichtbar = wahr,
    iconName = "http://www.pinntar.com/logoQbit.png")
@EinfachObjekt(extern = wahr)

öffentliche Klasse OpenQbitQRNGch erweitert AndroidNonvisibleComponent
implementiert Komponente {

    öffentliche statische Endfassung int VERSION = 1;
    öffentlicher statischer Endstring DEFAULT_TEXT_1 = "";
    privater ComponentContainer-Behälter;
    private Zeichenfolge text_1 = "";

    öffentlich OpenQbitQRNGch(Komponenten-Container-Behälter) {
        super(Behälter.$form());
        this.container = Behälter;
    }

    // Festlegung der Eigenschaften.
    //@DesignerProperty(editorType =
    PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue =
    OpenQbitQRNGch.DEFAULT_TEXTO_1 + "")
    //@SimpleProperty(Beschreibung = "API Get Quantum Random Number
    Generator, Zufallszahl zwischen 0 und 1. Variable *Menge* der zu
    generierenden Zahlen eingeben")
```

```

    @SimpleFunction(Beschreibung = "API Get Quantum Random Number Generator,
Zufallszahl zwischen 0 und 1. Variable Ganzzahl *Menge* der zu
generierenden Zahlen eingeben")
    public String APIGetQRNGdecimal(String qty) wirft Exception {
        Zeichenfolge url = "http://random.openqu.org/api/rand?size=" +
Anzahl;
        Befehl String[] = { "curl", url };

        ProcessBuilder-Prozess = neuer ProcessBuilder(-Befehl);
        Verfahren p;
        p = process.start();
        BufferedReader-Lesegerät = neu BufferedReader(neu
InputStreamReader(p.getInputStream()));
        StringBuilder-BUILDER = neu StringBuilder();
        String-Linie = null;
        while ((Zeile = reader.readLine()) != null) {
            builder.append(Zeile);

            builder.append(System.getProperty("Zeile.Trennzeichen"));
        }
        String-Ergebnis = builder.toString();
        //System.out.print(Ergebnis);
        Rückgabeergebnis;

    }

    @SimpleFunction(Beschreibung = "API Get Quantum Random Number Generator,
Zufallszahl zwischen einem Bereich von min. bis max. Geben Sie eine
variable ganzzahlige Zahl *Menge* von Zahlen ein, um einen Bereich von
minimaler *Min* Zahl und maximaler *Max* Zahl zu erzeugen")
    public String APIGetQRNGinteger(String qty, String min, String max)
wirft Exception {
    Zeichenfolge url = "http://random.openqu.org/api/randint?size=" +
qty + "&min=" + min + "&max=" + max
    Befehl String[] = { "curl", url };

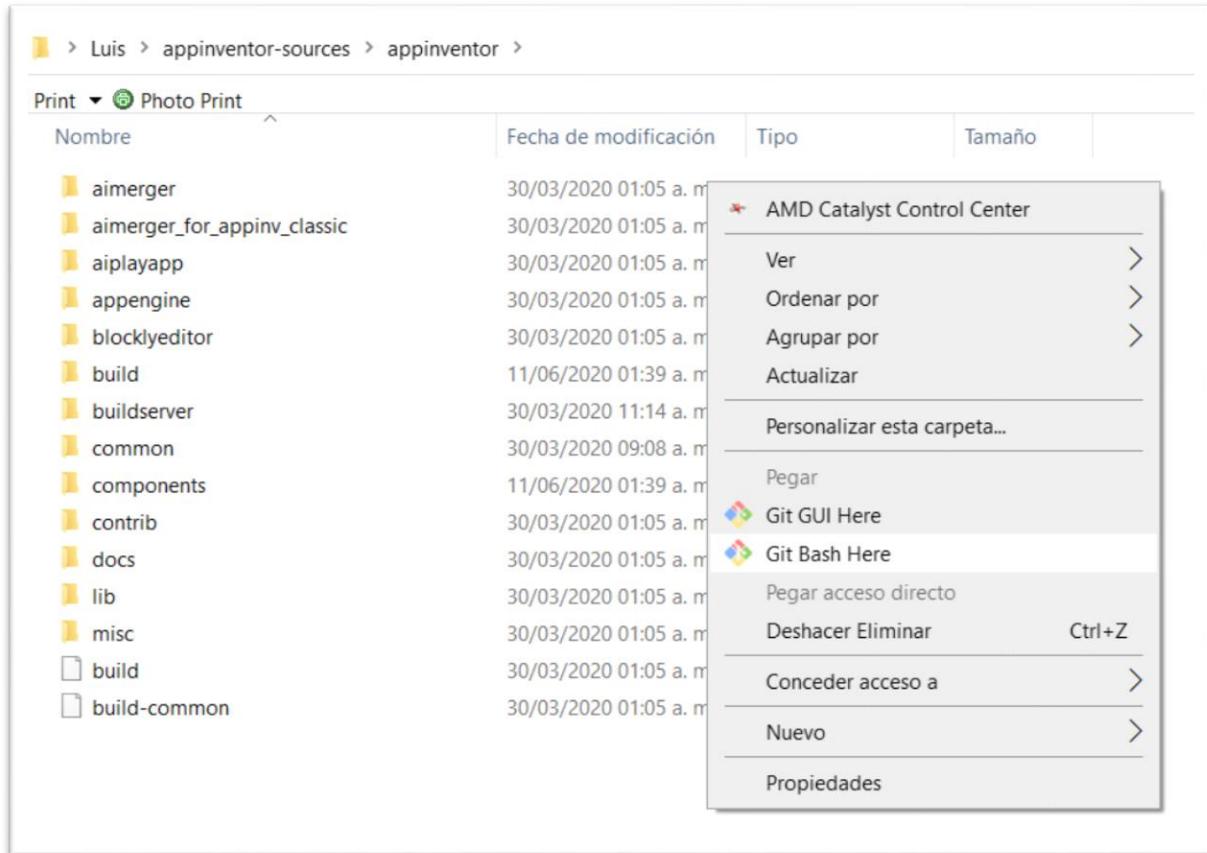
    ProcessBuilder-Prozess = neuer ProcessBuilder(-Befehl);
    Verfahren p;
    p = process.start();
    BufferedReader-Lesegerät = neu BufferedReader(neu
InputStreamReader(p.getInputStream()));
    StringBuilder-BUILDER = neu StringBuilder();
    String-Linie = null;
    while ((Zeile = reader.readLine()) != null) {
        builder.append(Zeile);

        builder.append(System.getProperty("Zeile.Trennzeichen"));
    }
    String-Ergebnis = builder.toString();
    //System.out.print(Ergebnis);
    Rückgabeergebnis;

}
}

```

Wir führen den **Git Bash** durch, indem wir uns auf folgenden Weg positionieren: C: Luis-appinventorsources-appinventor. In diesem Verzeichnis klicken wir mit der linken Maustaste und wählen das Git **Bash** Terminal aus.



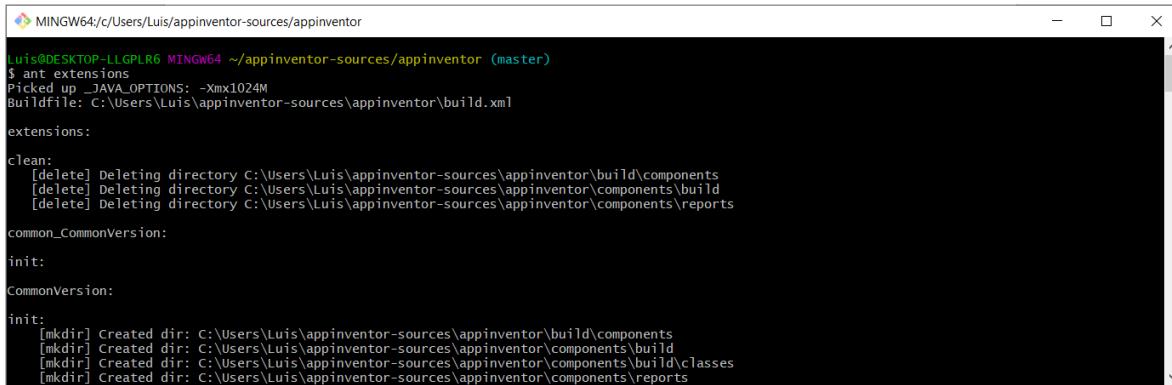
Das Git bash-Terminal wird im folgenden Verzeichnis positioniert:

C:\Benutzer\Luis\Erfinder-Quellen\Erfinder (Meister)

The terminal window title is 'MINGW64/c/Users/Luis/appinventor-sources/appinventor'. The prompt shows 'Luis@DESKTOP-LLGPLR6 MINGW64 ~ /appinventor-sources/appinventor (master)'. The terminal is currently empty, with a single dollar sign (\$) at the beginning of the line.

Im Git Bash-Terminal und führen Sie den folgenden Befehl aus:

### § Ameisenverlängerungen



```
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
$ ant extensions
Picked up _JAVA_OPTIONS: -Xmx1024m
Buildfile: C:/Users/Luis/appinventor-sources/appinventor/build.xml

extensions:
clean:
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/build/components
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/components/build
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/components/reports

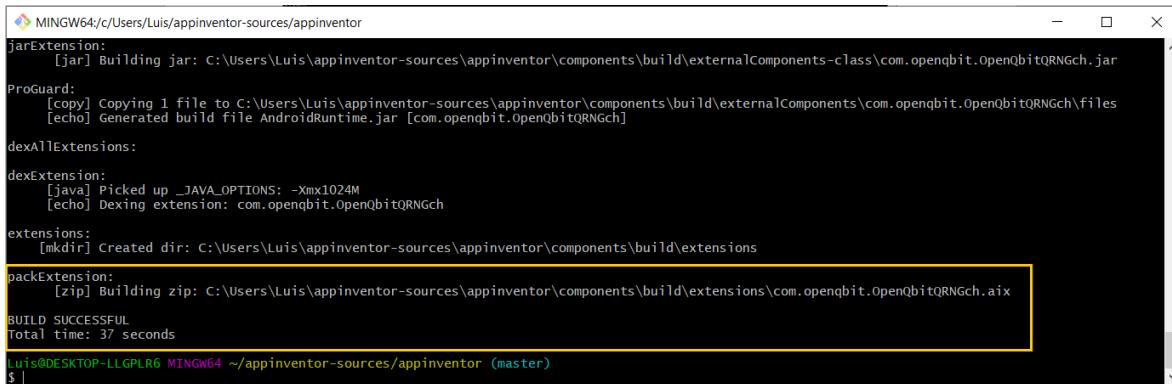
common_CommonVersion:
init:
CommonVersion:
init:
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/build/components
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build/classes
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/reports
```

Wenn alles gut gegangen ist, werden wir es bekommen: ERFOLGREICH AUFBAUEN.

Unsere Erweiterung wurde in...

C:\Benutzer\Luis\Erfinder-Quellen\Erfinder-Bausteine\Erweiterungen

**HINWEIS:** Der Inhalt des Ordners "Erweiterungen" wird jedes Mal, wenn wir eine Ameisenerweiterung machen, gelöscht und neu erstellt.



```
JarExtension:
[jar] Building jar: C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents-class/com.openqbit.OpenQbitQRNGch.jar
ProGuard:
[copy] Copying 1 file to C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents/com.openqbit.OpenQbitQRNGch/files
[echo] Generated build file AndroidRuntime.jar [com.openqbit.OpenQbitQRNGch]
dexAllExtensions:
dexExtension:
[java] Picked up _JAVA_OPTIONS: -Xmx1024M
[echo] Dexing extension: com.openqbit.OpenQbitQRNGch
extensions:
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build/extensions
packExtension:
[zip] Building zip: c:/Users/Luis/appinventor-sources/appinventor/components/build/extensions/com.openqbit.OpenQbitQRNGch.aix
BUILD SUCCESSFUL
Total time: 37 seconds
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
$ |
```

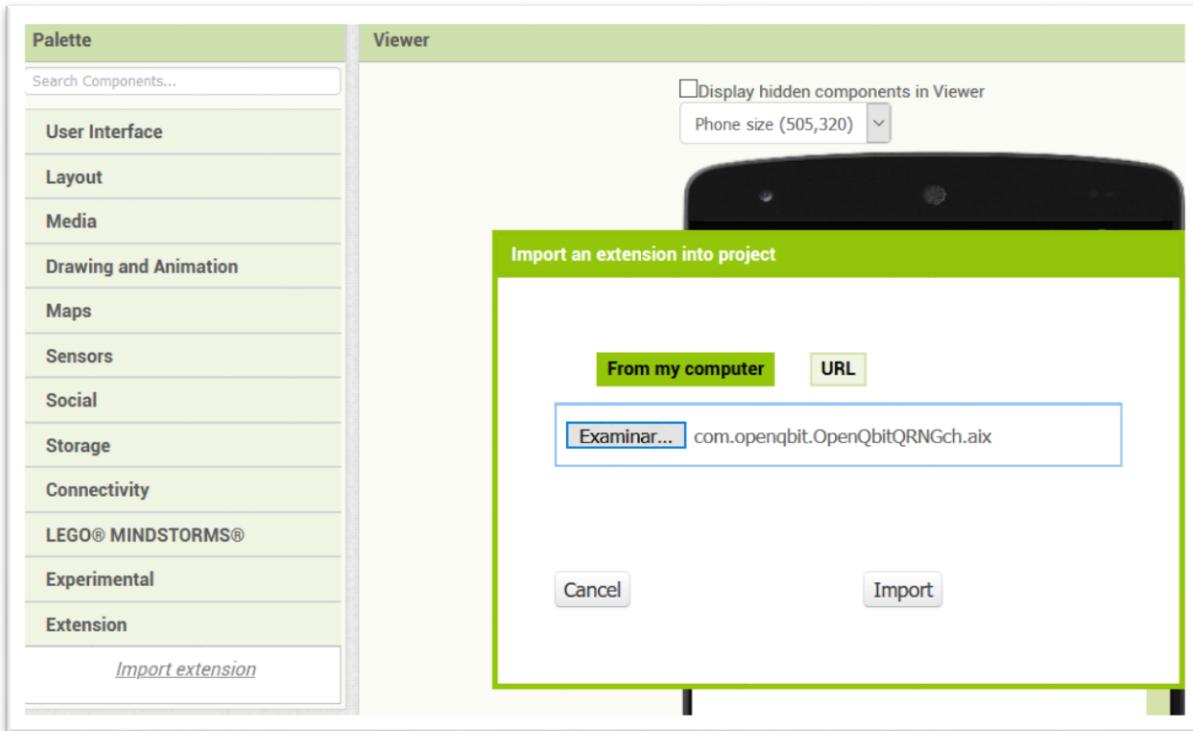
Gehen wir zu diesem Ordner:

C:\Benutzer\Luis\Erfinder-Quellen\Erfinder-Bausteine\Erweiterungen

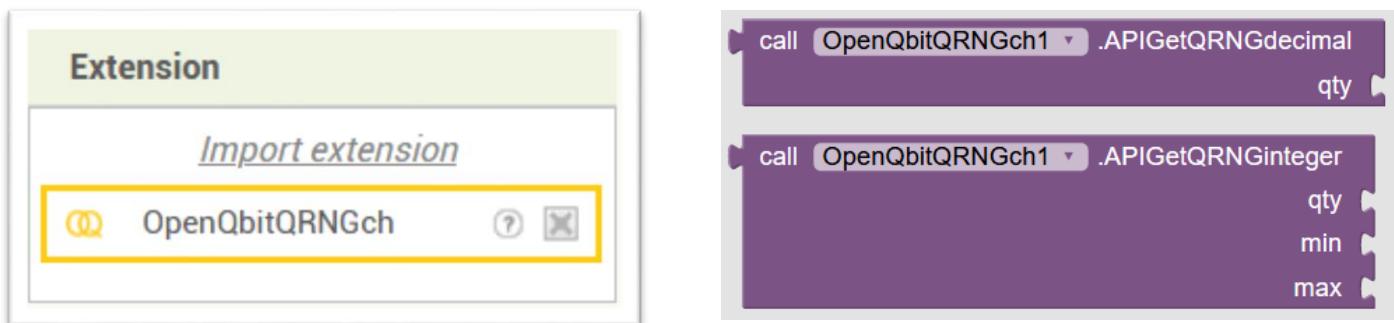
und kopieren Sie die Datei com.openqbit.OpenQbitQRNGch.aix, diese Datei ist unsere Erweiterung.

Da wir bereits ein Konto in App Inventor oder einem anderen blockweisen System haben, haben wir die neue Erweiterung hinzugefügt und getestet.

Gehen Sie nach unten, wo sich die Option "Erweiterung" befindet, und klicken Sie auf "Erweiterung importieren":



Wir klicken auf die Schaltfläche "Importieren". Jetzt haben wir die zu verwendenden Blöcke (APIGetQRNGdecimal) und (APIGetQRNGinteger):



Wir haben bereits unsere erste geschaffene und funktionale Erweiterung.

## 27. Anhang "BlocklyCode Smart Contracts".

BlocklyCodes sind Programme, die in der Sprache Java erstellt wurden. Die Erweiterung zur Schaffung dieser Art von Programmen, die allgemein als "Smart Contracts" bezeichnet werden, ist eine Möglichkeit, Vereinbarungen zwischen Benutzern (Unternehmen oder Personen) zu bearbeiten.

Dieser Block (**BlocklyCode**), ist in einem Programm implementiert, das bereits Parameter festgelegt hat und das je nach Art der Vereinbarungen, die durch das Mini BlocklyChain-System automatisch ausgeführt werden könnten, wenn die Voraussetzungen, die den "Smart Contract" regeln, erfüllt sind. Die zu berücksichtigenden Parameter für vorgeschlagene oder Eingabeparameter 'Eingaben' sind

Verfallsdatum

Referenziertes Datum

Verfallszeit

Referenzierte Zeit

Referenziertes Vermögenswert

Total Anlagevermögen

Variable Vermögenswerte

Vertragsmitglieder

Bestätigte Daten (String)

Bestätigte Daten (Dateiname)

Variables Ereignis

Feste Veranstaltung

Validierung von Dokument(en)

String-Validierung

Unterschrift gültig

Definiertes Intervall (Ganzzahl)

Dezimales Intervall

Etabliertes Minimum

Festgelegtes Maximum

Bevor wir mit der Verwendung des Blocks (**BlocklyCode**) beginnen, müssen wir zunächst das System installieren, das die Ausführung der "Smart Contracts" durchführt. Dies geschieht durch Installation im Terminal von Termux OpenJDK und OpenJRE.

OpenJDK (Offenes Java-Entwicklungskit). - Es ist das Werkzeug zur Entwicklung von Programmen in Java, es enthält die Bibliotheken, den Compiler und die JVM (Java Virtual Machine).

OpenJRE (Offene Java-Laufzeitumgebung). - Dies ist das Werkzeug nur für die Ausführung von Java-Programmen. Es enthält die JVM (Java Virtual Machine).

Wir fahren mit der Installation von OpenJRE und OpenJDK im Termux-Terminal der Knoten fort, die das Mini BlocklyChain-System bilden. Wir installierten die Räume

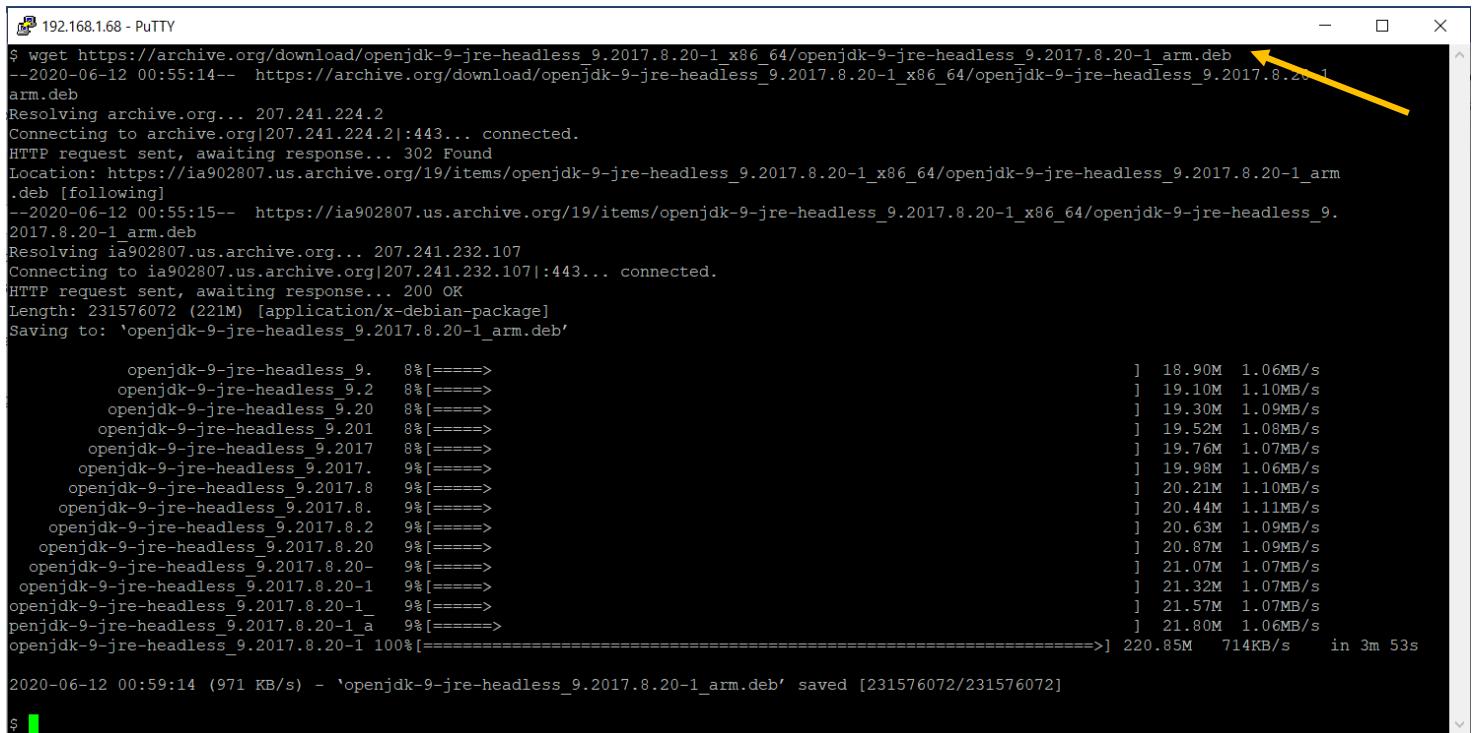
\$ apt install - und wget



```
$ 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-pac
ges-24 stable/main arm wget arm 1.20.3-2 [206 kB]
]
Fetched 206 kB in 1s (128 kB/s)
Selecting previously unselected package wget.
(Reading database ... 16936 files and directo
ries currently installed.)
Preparing to unpack .../archives/wget_1.20.3-2_a
rm.deb ...
Unpacking wget (1.20.3-2) ...
Setting up wget (1.20.3-2) ...
$
```

Wir haben die OpenJRE heruntergeladen

\$ 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](https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb)



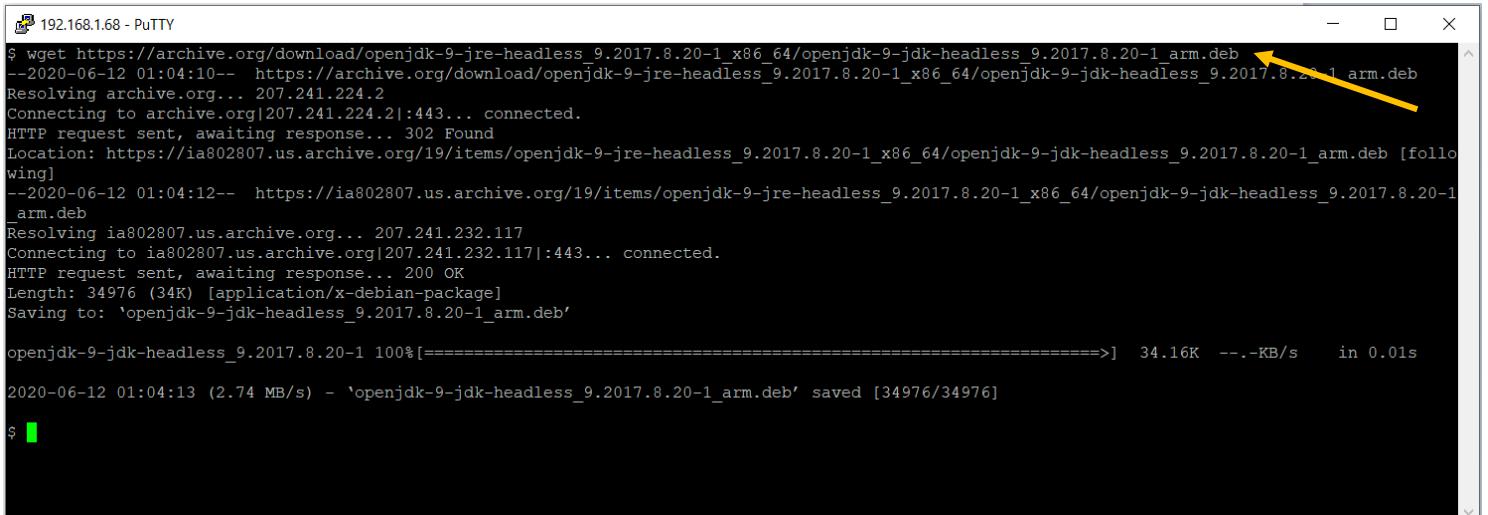
```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
--2020-06-12 00:55:14-- https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org[207.241.224.2]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 00:55:15-- https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving ia902807.us.archive.org... 207.241.232.107
Connecting to ia902807.us.archive.org[207.241.232.107]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231576072 (221M) [application/x-debian-package]
Saving to: 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb'

openjdk-9-jre-headless_9. 8%[=====] 18.90M 1.06MB/s
openjdk-9-jre-headless_9.2 8%[=====] 19.10M 1.10MB/s
openjdk-9-jre-headless_9.20 8%[=====] 19.30M 1.09MB/s
openjdk-9-jre-headless_9.201 8%[=====] 19.52M 1.08MB/s
openjdk-9-jre-headless_9.2017 8%[=====] 19.76M 1.07MB/s
openjdk-9-jre-headless_9.2017. 9%[=====] 19.98M 1.06MB/s
openjdk-9-jre-headless_9.2017.8 9%[=====] 20.21M 1.10MB/s
openjdk-9-jre-headless_9.2017.8. 9%[=====] 20.44M 1.11MB/s
openjdk-9-jre-headless_9.2017.8.2 9%[=====] 20.63M 1.09MB/s
openjdk-9-jre-headless_9.2017.8.20 9%[=====] 20.87M 1.09MB/s
openjdk-9-jre-headless_9.2017.8.20- 9%[=====] 21.07M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1 9%[=====] 21.32M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_a 9%[=====] 21.57M 1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_a 100%[=====] 220.85M 714KB/s in 3m 53s
2020-06-12 00:59:14 (971 KB/s) - 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb' saved [231576072/231576072]

$
```

Wir haben das OpenJDK heruntergeladen

\$ 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](https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb)



```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
--2020-06-12 01:04:10-- https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org[207.241.224.2]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 01:04:12-- https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving ia802807.us.archive.org... 207.241.232.117
Connecting to ia802807.us.archive.org[207.241.232.117]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34976 (34K) [application/x-debian-package]
Saving to: 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb'

openjdk-9-jdk-headless_9.2017.8.20-1 100%[=====] 34.16K --.-KB/s in 0.01s
2020-06-12 01:04:13 (2.74 MB/s) - 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb' saved [34976/34976]

$
```

Wir führen die Installation vom Termux-Terminal von OpenJDK und OpenJRE aus durch:

```
$ apt install - und ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
```



```
$ ls
openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
$ apt install -y ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'openjdk-9-jdk-headless' instead
of './openjdk-9-jdk-headless_9.2017.8.20-1_arm.
deb'
Note, selecting 'openjdk-9-jre-headless' instead
of './openjdk-9-jre-headless_9.2017.8.20-1_arm.
deb'
The following additional packages will be instal
led:
  ca-certificates-java freetype libpng
The following NEW packages will be installed:
  ca-certificates-java freetype libpng
  openjdk-9-jdk-headless
  openjdk-9-jre-headless
0 upgraded, 5 newly installed, 0 to remove and 0
not upgraded.
Need to get 668 kB/232 MB of archives.
After this operation, 376 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm ca-certificates-java all
20200101 [110 kB]
Get:2 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm libpng arm 1.6.37-2 [190
kB]
Get:3 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm freetype arm 2.10.2 [368
kB]
Get:4 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jre-headless_9.2017.8.20-1_arm.deb open
jdk-9-jre-headless arm 9.2017.8.20-1 [232 MB]
Get:5 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jdk-headless_9.2017.8.20-1_arm.deb open
jdk-9-jdk-headless arm 9.2017.8.20-1 [35.0 kB]
```



```
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 directo
ries 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
) ...
$
```

Da wir die Installation der OpenJDK- und OpenJRE-Umgebung abgeschlossen haben. Diese Installationen enthalten die JVM (Java Virtual Machine), die die Umgebung sein wird, in der der BlocklyCode ausgeführt wird.

Jetzt werden wir einen Editor installieren, um eine Datei online zu erstellen, wir werden den Editor namens **vi** verwenden und den folgenden Befehl ausführen:

```
$ apt install vim
```

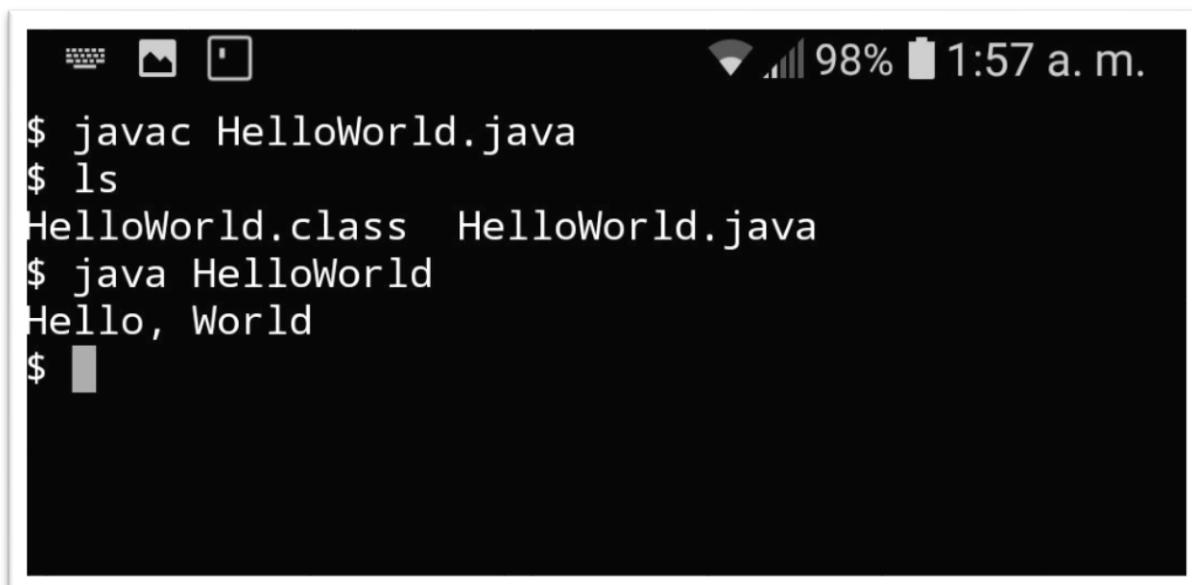
Lassen Sie uns nun versuchen, eine Testdatei zu kompilieren und auszuführen. Wir erstellen im Termux-Terminal mit dem **vi-Editor und** schreiben den Java-Code einer "Hello World" und speichern ihn in der Datei HelloWorld.java

```
public classe HelloWorld {  
  
    öffentliche statische Leere main(String[] args) {  
        // Druckt "Hello, World" in das Terminalfenster.  
        System.out.println ("Hallo, Welt");  
    }  
  
}
```

Dann führen wir den folgenden Befehl im Termux-Terminal für die Kompilierung und dann die Ausführung des Programms aus:

```
$ javac HelloWorld.java (Nach der Ausführung wird eine Bytecode-Datei HelloWorld.class erstellt)
```

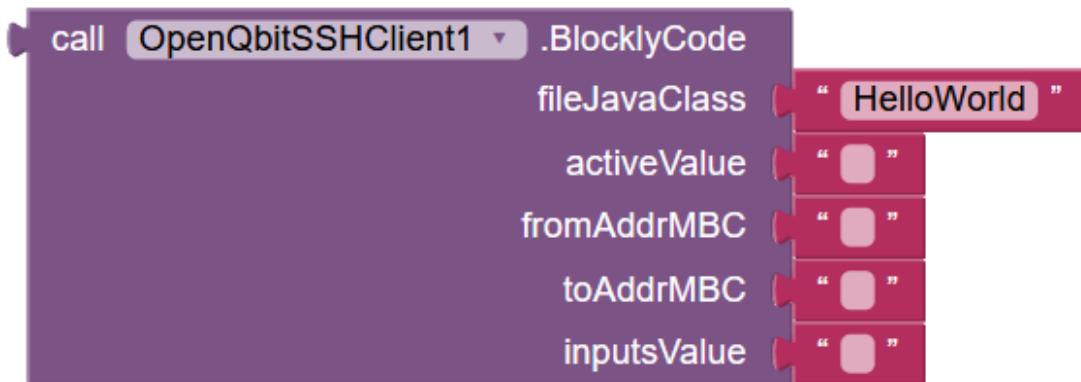
```
$ java HelloWorld (Nach dem Ausführen und Drucken der Anzeige, Hello World)
```



The screenshot shows a Termux terminal window on a mobile device. The status bar at the top indicates signal strength, battery level (98%), and time (1:57 a.m.). The terminal window displays the following command-line session:

```
$ javac HelloWorld.java  
$ ls  
HelloWorld.class  HelloWorld.java  
$ java HelloWorld  
Hello, World  
$
```

Bei Verwendung des Blocks (**BlocklyCode**) befindet sich dieser Block in der Erweiterung



(**ConnectorSSHClient**).

Im vorherigen Block können Sie sehen, wie die Datei `HelloWorld.class` ausgeführt wird, die bereits kompiliert und im Benutzer-Basisverzeichnis positioniert wurde.

Es ist ein Block, in dem es möglich ist, ein bereits in Java kompiliertes Programm auszuführen, das in der Entwicklungsumgebung allgemein als Bytecode-Datei in ihrer binären Form bekannt ist.

Dieser Dateityp ist bereit, von der Java Virtual Machine (JVM) ausgeführt zu werden.

Es gibt zwei Möglichkeiten, um eine Bytecode-Datei zu erstellen oder mit der Erweiterung `.CLASS`, kann der Benutzer es in seinem PC durch Komfort oder ähnliches kann auch in das Handy getan werden, denken Sie daran, dass wir bereits die Umgebung installiert haben, um JDK (Java Development Kit) zu kompilieren, bevor, obwohl es weniger effizient sein wird, weil die Beschränkung der Tastatur wird zählen, auch im Falle der Wahl der Android-Umgebung wird zu berücksichtigen haben, dass die Bibliotheken auf OpenJDK, die installiert wurde begrenzt sind.

Die Eingabeparameter sind offen in `<inputsValue>` und die anderen festen sind die von `activeValue`, `fromaddrMBC` und `toAddrMBC`.

Im Falle von Ausführungstests können sie ohne Inhalt leer gelassen werden und nur die Bytecode-Datei wird ohne Parameter ausgeführt.

Der vorherige Block ist wie die nächste Befehlszeile, die ausgeführt wird:

`$ java HalloWelt`

Die für BlocklyCode entwickelten Programme unterliegen der jeweiligen Designumgebung und können durch Routine mit dem "cron"-Agenten gesteuert und ausgeführt und mit dem Syncthingmanager gemeinsam genutzt werden.

## 28. Anhang "OpenQbit Quantum Computing".

### Wie funktioniert das Quantencomputing? <sup>(2)</sup>

Der digitale Wandel bringt schneller als je zuvor Veränderungen in der Welt mit sich. Würden Sie glauben, dass das digitale Zeitalter kurz vor dem Ende steht? **Die digitale Kompetenz** wurde bereits als ein Bereich identifiziert, in dem offenes Wissen und zugängliche Möglichkeiten zum Erlernen von Technologie dringend erforderlich sind, um Lücken in der sozialen und wirtschaftlichen Entwicklung zu schließen. Das Lernen von den Schlüsselkonzepten des digitalen Zeitalters wird mit der bevorstehenden Ankunft einer weiteren neuen technologischen Welle, die in der Lage ist, bestehende Modelle mit erstaunlicher Geschwindigkeit und Kraft zu transformieren, noch kritischer werden: die **Quantentechnologien**.

In diesem Artikel vergleichen wir die grundlegenden Konzepte des traditionellen Rechnens und des Quantencomputings; und wir beginnen auch mit der Untersuchung ihrer Anwendung in anderen verwandten Bereichen.

#### Was sind Quantentechnologien?

Im Laufe der Geschichte hat der Mensch die Technik entwickelt, da er durch die Wissenschaft verstanden hat, wie die Natur funktioniert. Zwischen 1900 und 1930 führte die Untersuchung einiger physikalischer Phänomene, die noch nicht gut verstanden waren, zu einer neuen physikalischen Theorie, der **Quantenmechanik**. Diese Theorie beschreibt und erklärt die Funktionsweise der mikroskopischen Welt, dem natürlichen Lebensraum von Molekülen, Atomen oder Elektronen. Dank dieser Theorie war es nicht nur möglich, diese Phänomene zu erklären, sondern auch zu verstehen, dass die subatomare Realität auf eine völlig kontraintuitive, fast magische Weise funktioniert und dass in der mikroskopischen Welt Ereignisse stattfinden, die in der makroskopischen Welt nicht vorkommen.

Zu diesen **Quanteneigenschaften** gehören Quantenüberlagerung, Quantenverschränkung und Quantenteleportation.

- **Die Quantenüberlagerung** beschreibt, wie sich ein Teilchen gleichzeitig in verschiedenen Zuständen befinden kann.
- **Die Quantenverschränkung** beschreibt, wie zwei beliebig weit voneinander entfernte Teilchen so korreliert werden können, dass bei der Wechselwirkung mit dem einen das andere sich dessen bewusst wird.
- **Die Quantenteleportation** nutzt die Quantenverschränkung, um Informationen von einem Ort im Raum zu einem anderen zu senden, ohne ihn durchqueren zu müssen.

Quantentechnologien basieren auf diesen Quanteneigenschaften der subatomaren Natur.

In diesem Fall erlaubt uns heute das Verständnis der mikroskopischen Welt durch die Quantenmechanik, Technologien zu erfinden und zu entwerfen, die das Leben der Menschen verbessern können. Es gibt viele und sehr unterschiedliche Technologien, die sich Quantenphänomene zunutze machen, und einige von ihnen, wie z.B. Laser oder Magnetresonanztomographie (MRI), begleiten uns seit mehr als einem halben Jahrhundert. Derzeit erleben wir jedoch eine technologische Revolution in Bereichen wie Quantencomputer, Quanteninformation, Quantensimulation, Quantenoptik, Quantenmetrologie, Quantenuhren oder Quantensensoren.

Was ist Quanteninformatik? Zuerst müssen Sie das klassische Rechnen verstehen.

**FIGURA 1.**  
Ejemplos de caracteres en lenguaje binario.

Caracter	Bits
7	111
A	01000001
\$	00100100
:)	0011101000101001

Um zu verstehen, wie Quantencomputer funktionieren, ist es sinnvoll, zunächst zu erklären, wie die Computer funktionieren, die wir täglich benutzen und die wir in diesem Dokument als digitale oder klassische Computer bezeichnen. Diese, wie auch die übrigen elektronischen Geräte wie Tablets oder Mobiltelefone, verwenden Bits als die grundlegenden Speichereinheiten. Dies bedeutet, dass Programme und Anwendungen in Bits kodiert sind, d.h. in binärer Sprache mit Nullen und Einsen. Jedes Mal, wenn wir mit einem dieser Geräte interagieren, zum Beispiel durch Drücken einer Taste auf der Tastatur, werden im Computer Zeichenketten aus Nullen und Einsen erzeugt, zerstört und/oder verändert.

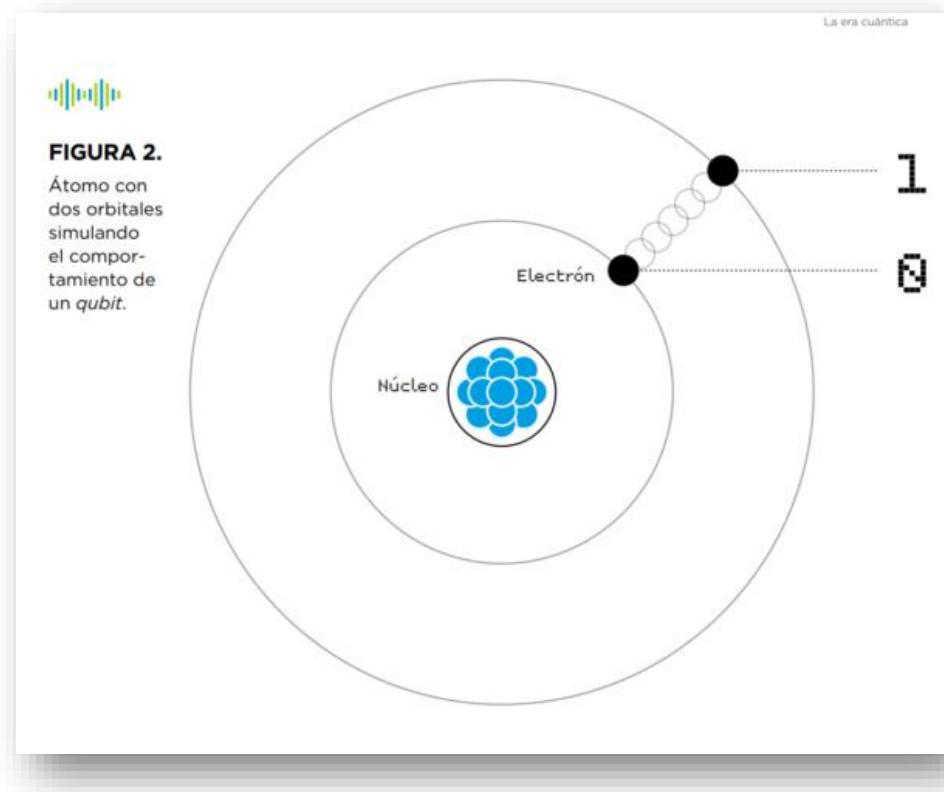
Die interessante Frage ist, was diese Nullen und Einsen physisch im Computer sind. Der Null- und der Eins-Zustand entsprechen elektrischem Strom, der durch mikroskopisch kleine Teile, die als Transistoren bezeichnet werden und als Schalter fungieren, zirkuliert oder nicht. Wenn kein Strom fließt, ist der Transistor "aus" und entspricht Bit 0, und wenn er fließt, ist er "an" und entspricht Bit 1.

Einfacher gesagt, es ist so, als ob die Bits 0 und 1 Löchern entsprechen, so dass ein leeres Loch ein Bit 0 und ein von einem Elektron besetztes Loch ein Bit 1 ist. Deshalb werden diese Geräte als Elektronik bezeichnet. Als Beispiel zeigt Abbildung 1 die binäre Schrift einiger

Zeichen. Da wir nun eine Vorstellung davon haben, wie die heutigen Computer funktionieren, wollen wir versuchen zu verstehen, wie die Quanten funktionieren.

### Von Bits zu Qubits

Die grundlegende Informationseinheit im Quantencomputing ist das Quantenbit oder Qubit. Qubits sind per Definition zweistufige Quantensysteme - wir werden hier Beispiele sehen -, die sich wie Bits auf dem niedrigen Niveau befinden können, was einem Zustand niedriger Anregung oder Energie entspricht, der als 0 definiert ist, oder auf dem hohen Niveau, was einem Zustand höherer Anregung entspricht oder als 1 definiert ist. Allerdings, und hier liegt der grundlegende Unterschied zum klassischen Rechnen, können Qubits auch in jedem der unendlichen Zwischenzustände zwischen 0 und 1 liegen, wie z.B. in einem Zustand, der halb 0 und halb 1 oder drei Viertel von 0 und ein Viertel von 1 ist.



Quantenalgorithmen, exponentiell leistungsfähigeres und effizienteres Rechnen

Der Zweck von Quantencomputern ist es, diese Quanteneigenschaften von *Qubits* als Quantensysteme, die sie sind, zu nutzen, um Quantenalgorithmen auszuführen, die Überlappung und Verschachtelung verwenden, um eine viel größere Rechenleistung als die Klassiker zu erreichen. Es ist wichtig, darauf hinzuweisen, dass der wirkliche Paradigmenwechsel nicht darin besteht, das Gleiche zu tun wie digitale oder klassische Computer - die aktuellen -, sondern schneller, wie in vielen Artikeln zu lesen ist, sondern dass Quantenalgorithmen es erlauben, bestimmte Operationen auf eine völlig andere Art und

Weise durchzuführen, die sich in vielen Fällen als effizienter erweist - d.h. in viel weniger Zeit oder mit viel weniger Rechenressourcen -.

Schauen wir uns ein konkretes Beispiel dafür an, was dies beinhaltet. Stellen wir uns vor, wir sind in Bogotá, und wir wollen die beste Route nach Lima aus einer Million Möglichkeiten kennen, um dorthin zu gelangen ( $N=1.000.000$ ). Um mit Hilfe von Computern den optimalen Weg zu finden, müssen wir 1.000.000 Optionen digitalisieren, was bedeutet, sie für den klassischen Computer in Bitsprache und für den Quantencomputer in *Qubits* zu übersetzen. Während ein klassischer Computer einen nach dem anderen alle Pfade analysieren müsste, bis er den gewünschten gefunden hat, nutzt ein Quantencomputer den als Quantenparallelität bekannten Prozess, der es ihm erlaubt, alle Pfade auf einmal zu berücksichtigen. Dies bedeutet, dass, während der klassische Computer die Reihenfolge von  $N/2$  Schritten oder Iterationen benötigt, d.h. 500.000 Versuche, der Quantencomputer den optimalen Pfad nach nur  $\sqrt{N}$  Operationen auf dem Register findet, d.h. nach 1.000 Versuchen.

Im vorherigen Fall ist der Vorteil quadratisch, in anderen Fällen ist er sogar exponentiell, was bedeutet, dass wir mit  $n$  *Qubits* eine Rechenkapazität von  $2^n$  Bits erreichen können. Um dies zu veranschaulichen, ist es üblich zu zählen, dass wir mit etwa 270 Qubits in einem Quantencomputer mehr Basiszustände - mehr unterschiedliche und gleichzeitige Zeichenketten - haben könnten als die Anzahl der Atome im Universum, die auf etwa 280 geschätzt wird. Ein weiteres Beispiel ist, dass man schätzt, dass wir mit einem Quantencomputer mit 2000 bis 2500 *Qubits* praktisch die gesamte heute verwendete Kryptographie (die sogenannte Public-Key-Kryptographie) brechen könnten.

Warum ist es wichtig, etwas über Quantentechnologie zu wissen?

Wir befinden uns in einem Moment der digitalen Transformation, in dem verschiedene aufkommende Technologien wie Blockchain, künstliche Intelligenz, Drohnen, Internet der Dinge, virtuelle Realität, 5G, 3D-Drucker, Roboter oder autonome Fahrzeuge immer mehr Präsenz in verschiedenen Bereichen und Sektoren haben. Diese Technologien, die dazu berufen sind, die Lebensqualität des Menschen zu verbessern, die Entwicklung zu beschleunigen und soziale Auswirkungen zu erzeugen, schreiten heute parallel voran. Nur selten sehen wir Unternehmen, die Produkte entwickeln, die Kombinationen aus zwei oder mehr dieser Technologien nutzen, wie z.B. Blockchain und IoT oder Drohnen und künstliche Intelligenz. Obwohl sie dazu bestimmt sind, zu konvergieren und damit eine exponentiell größere Wirkung zu erzielen, ist die Konvergenz aufgrund des Anfangsstadiums der Entwicklung, in dem sie sich befinden, und des Mangels an Entwicklern und Personen mit technischem Profil noch immer eine offene Aufgabe.

Aufgrund ihres disruptiven Potenzials wird erwartet, dass die Quantentechnologien nicht nur mit all diesen neuen Technologien konvergieren, sondern einen Querschnittseinfluss auf praktisch alle von ihnen haben werden. Die Quanteninformatik wird die Authentifizierung,

den Austausch und die sichere Speicherung von Daten bedrohen, was einen großen Einfluss auf jene Technologien hat, bei denen die Kryptographie eine wichtigere Rolle spielt, wie z.B. Cybersicherheit oder Blockchain, und einen geringen negativen Einfluss hat, aber auch in Technologien wie 5G, IoT oder Drohnen in Betracht gezogen werden muss.

Wollen Sie sich im Quantencomputing üben?

Dutzende von Quantencomputer-Simulatoren sind bereits im Netz verfügbar, wobei verschiedene Programmiersprachen wie C, C++, Java, Matlab, Maxima, Python oder Octave bereits im Einsatz sind. Auch neue Sprachen wie Q#, die von Microsoft eingeführt wurden. Sie können eine virtuelle Quantenmaschine über Plattformen wie IBM und Rigetti erforschen und mit ihr spielen.

Mini BlocklyChain wird von der Firma OpenQbit.com entwickelt, die sich auf die Entwicklung von Quantencomputertechnologie für verschiedene Arten von privaten und öffentlichen Sektoren konzentriert.

**Warum Mini BlocklyChain sich von anderen Blockketten unterscheidet, einfach weil das System modular aufgebaut ist und Quanten-Computing beinhaltet.**

- (2) <https://blogs.iadb.org/conocimiento-abierto/es/como-funciona-la-computacion-cuantica/>

## 29.Anhang "Erweiterte Blöcke für SQLite-Datenbank".

Um mehr Details über die ordnungsgemäße und rechtzeitige Verwendung der einzelnen Blöcke zu sehen, aus denen die Erweiterung OpenQbitSQLite besteht, überprüfen Sie den ursprünglichen Autor der Erweiterung im folgenden Link.

OpenQbitSQLite ist ein Klon des ursprünglichen Designs mit kleinen Modifikationen zur Verwendung mit einer AES-Sicherheitsstufe.

<https://github.com/frdfsnlght/aix-SQLite>

## 30.Anhang "Beispiel für die Erstellung eines Mini-BlocklyChain-Systems".

Wir werden ein Testsystem erstellen, mit dem wir die Funktionalitäten, Vorteile und die Einfachheit der Erstellung eines Mini-BlocklyChain-Systems verifizieren können.

Wir werden mit dem Entwurf und der Entwicklung des Speichers beginnen, in dem die Informationen gespeichert werden, was wir bereits als eine Kette von Blöcken definiert haben. Das Design wird auf der SQLite-Datenbank basieren, und je nach Organisation oder Design kann diese leicht von einer anderen Datenbank wie Microsoft SQL Server, MySQL, Maria DB, Oracle, DB2, PostgreSQL u.a. geändert werden. Obwohl wieder aufgrund des

Prozesses der Transaktionen und der Übereinstimmung mit der SQLite-Datenbank kann auf jeder Ebene und für geschäftliche oder persönliche Anwendung verwendet werden.

Die Datenbankerstellung, die hier **minibc** genannt wird, verfügt über eine Tabelle, in der die Blockzeichenfolge gespeichert wird. Diese Datenbank wird in den endgültigen Programmcode eingebettet, der in den Knoten installiert wird, dieser Speicherort wird "assets packaged" genannt ist eine interne Spezies in Blockly-Umgebungen wie App Inventor.

```
$ sqlite3 minibc.db
```

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

```
Geben Sie ".help" für Benutzungshinweise ein.
```

```
Verbunden mit einer transienten In-Memory-Datenbank.
```

```
Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.
```

```
sqlite> .quit
```

Gestalten von Feldern der **nBlock**-Tabelle.

```
TABELLE nBlock ERSTELLEN (
    id ganzzahliger Primärschlüssel AUTOINCREMENT
    , prevhashVARCHAR NOT NULL
    , newhashVARCHAR NOT NULL
    ntransINTEGER      NICHT NULL
    nonceINTEGER      NICHT NULL
    ,qrng            GANZZAHL NICHT NULL
);
```

Wir haben die **nblock**-Tabelle erstellt.

```
$ sqlite3 minibc.db
```

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

```
Geben Sie ".help" für Benutzungshinweise ein.
```

```
Verbunden mit einer transienten In-Memory-Datenbank.
```

```
Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.
```

```
sqlite> .open minibc.db
```

```
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 );
```

Wir werden etwas Ähnliches sehen:

```

$ 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
$ 

```

SQL-Anweisung für die Erstellung der nBlock-Tabelle.

Als nächstes erstellen wir den anfänglichen Hash des Systems namens "**Genesis**", der auf der Erstellung eines neuen Hashes des ersten Blocks der Blockkette basiert, die wir verwenden werden (**NewBlock**). Dann erstellen wir einen zweiten Hash, den wir auf der Grundlage des "Genesis"-Hash als "einen" bezeichnen, weil er mit dem ersten Hash konsistent sein muss,

denn der Hash-Validierungstest auf der Kette des Anfangsblocks muss immer der folgenden Bedingung entsprechen: prevhash = newhash.

**NeuerBlock**. Um den "Genesis"-Hash zu erstellen, werden wir die Eingabeparameter verwenden:



Eingabeparameter: Daten <Zeichenkette>, vorherigerHash <Zeichenkette>

Ausgabeparameter: Ein SHA256 Hash.

In diesem Fall verwenden wir als "previousHash" den Anfangsbuchstaben gleich "0" und im Falle von Eingabedaten ist "data" das Wort "Genensis".

Dadurch erhalten wir einen berechneten Hash aus der Kombination beider Daten:

SHA256 (Genesis0) =  
eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfb09ba54e802e642

Die obige Zeichenfolge wird in die nblock-Tabelle eingefügt, diese Zeichenfolge muss verschlüsselt werden, dazu verwenden wir die Erweiterung (**OpenQbitEncDecData**).



Wir werden die OpenQbitSSHClient-Erweiterung verwenden, um die Daten in die minibc.db-Datenbank einzufügen, es wäre die INSERT-Anweisung in die minibc.db-Datenbank der nblock-Tabelle.

```
sqlite3 keystore.db "in nblock (prevhash, newhash, ntrans, nonce, qrng) Werte einfügen  
('0', 'eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfb09ba54e802e642', '1', '0',  
'0');";"
```

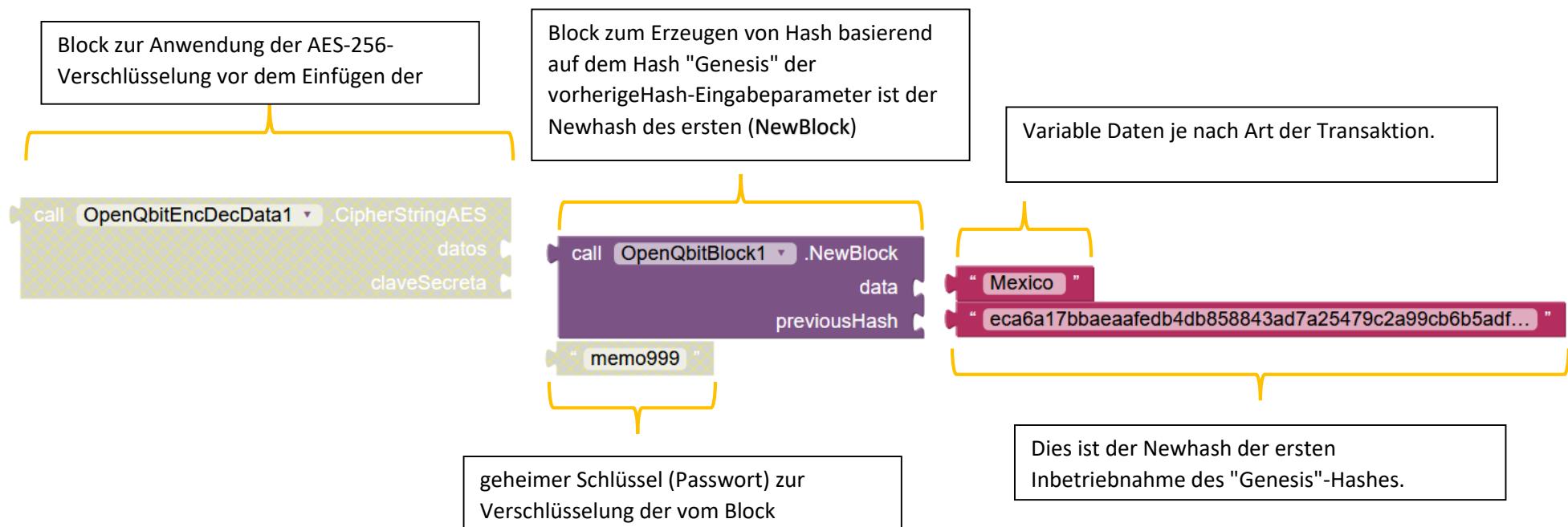
Wir haben den "einen" Hash mit der SQL-Anweisung erstellt, die auf dem "Genensis"-Hash basiert:

```
sqlite3 Schlüsselspeicher.db "in nblock (prevhash, newhash, ntrans, nonce, qrng) Werte  
einfügen ('  
eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642',f6a6b509376  
deebc8a5d70da9d0436943b76c3933f35c419ed48e78e78ababab59a68','1','0','0');"
```

Der Hash "ein" Newhash wird berechnet als:

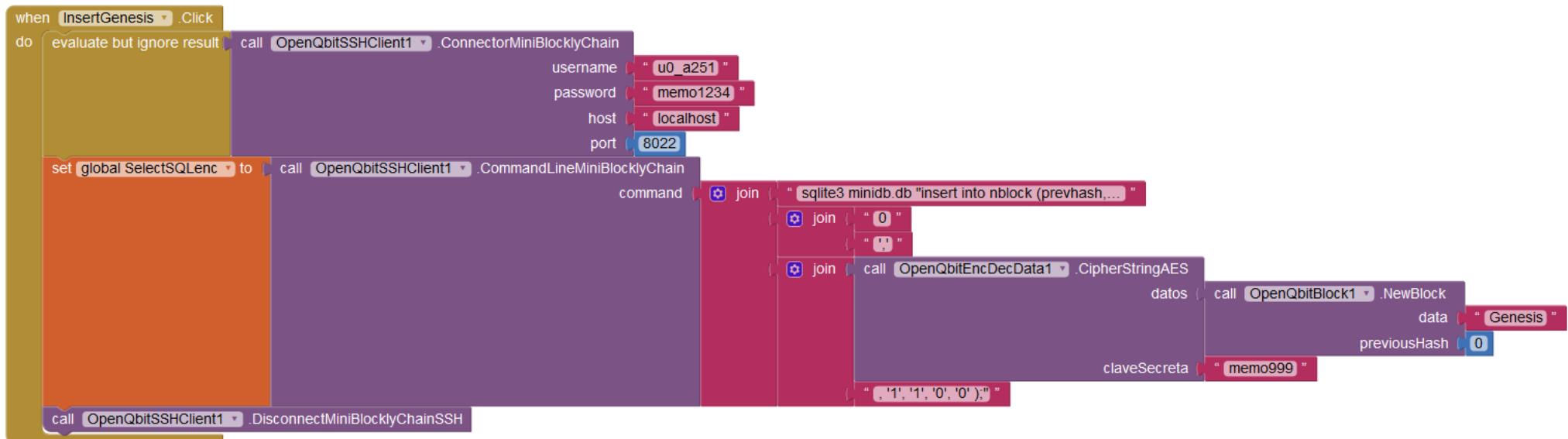
SHA256 (eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642Mexiko) =  
f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78ab59a68

Zweiter Hash, der auf dem ersten Initialisierungshash "Genesis" basiert, müssen wir zu Beginn zwei EINFÜGUNGEN vornehmen, da jede anfängliche Validierung der Blockkette der Gleichheit der Felder entsprechen muss: prevhash (vorletzte Daten) = newhash (letzte Daten).



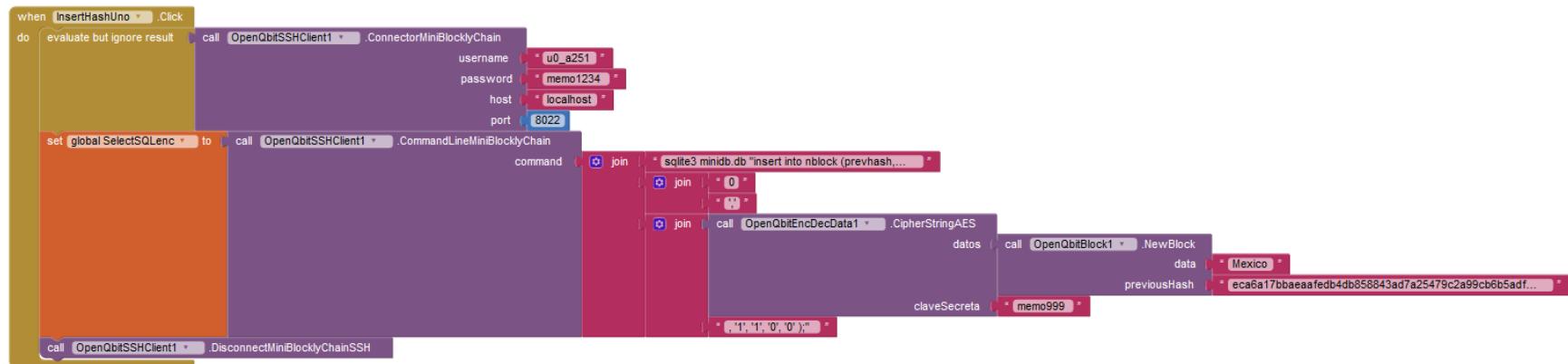
Wir fahren mit der Erstellung der Ausführung innerhalb von App Inventor der obigen Hashes fort; "Genesis"-Hash und "Start"-Hash.

Nun zeigen wir, wie das INSERT des "**Genesis**"-Hashes in die Anfangsstruktur der Blockkette innerhalb der nblock-Tabelle im App Inventor-System integriert würde:



Da wir die ersten Daten auf der Grundlage des "Genesis"-Hashes in die n-Block-Tabelle eingefügt haben, fahren wir mit dem zweiten INSERT fort, das sich auf den "einen" Hash bezieht.

Es würde das INSERT des "One"-Hashes in die anfängliche Struktur der Blockkette innerhalb der nblock-Tabelle im App Inventor-System integrieren:



Die beiden bisherigen Programmierstrukturen (Hash "Genesis" und Hash "one") sollten durch den Anfangsknoten des Mini BlocklyChain-Systems integriert werden. Ein wichtiger Punkt ist, dass die Datenbank minibc.db mit ihrer gesamten internen Struktur (Tabellen) durch das Mini BlocklyChain-Netzwerk auf der Grundlage einer "Peer-to-Peer"-Architektur repliziert wird.

Bis jetzt haben wir die grundlegende und grundlegende Struktur der Blockkettenspeicherung abgeschlossen, und sie ist bereit, neue Blöcke aus künftigen Transaktionen, die aus dem System stammen oder in diesem System angefordert werden, aufzunehmen.

Wir haben bereits die ersten Hash-Daten "Genesis" und den Hash "one" in unsere Datenbank **minibc.db** eingefügt, jetzt werden wir die folgenden SQL-Anweisungen verwenden, um die Felder **prevhash** und **newhash** in der Tabelle **nblock** abzufragen.

Dieser Punkt wird von grundlegender Bedeutung sein, da wir auf der Grundlage des Vergleichs dieser beiden Bereiche wissen werden, ob die Blockkette konsistent ist und ob sie die Integrität und Sicherheit der Transaktionen aufrechterhält. Dies ist nur ein Anfangsmechanismus zur Überprüfung der Blockkette, da wir in Zukunft die Verwendung des Blocks zur Berechnung des Merkle-Baums überprüfen werden, der ein weiteres wichtiges Instrument zur Gewährleistung der Integrität und Sicherheit der Blockkette auch in unserem System sein wird.

Im Moment werden wir nur die Struktur oder die SQL-Sätze überprüfen, die wir wie bisher mit der Erweiterung (**OpenQbitSSHClient**) verwenden müssen, mit diesen werden wir in der Lage sein, die Felder **prevhash** und **newhash** zu konsultieren. Später können wir jedes Mal, wenn wir einen neuen Block hinzufügen, einen einfachen Vergleich gleicher Daten für diese Prüfung durchführen.

Für Vorwäsche:

```
sqlite3 minibc.db "SELECT prevhash FROM nblock ORDER BY id DESC LIMIT 1;"
```

Für Newhash:

```
sqlite3 minibc.db "SELECT newhash FROM nblock ORDER BY id DESC LIMIT 1;"
```

Wir werden uns nun darauf konzentrieren, wie wir einen Antrag zur Einreichung eines neuen Handels und/oder einer neuen Transaktion stellen, die in die Transaktionswarteschlange aufgenommen werden sollen.

Es gibt zwei Schritte als Voraussetzung, um eine Transaktion an die Transaktionswarteschlange zu senden.

Die erste Voraussetzung ist, dass unser Kontostand über genügend "Guthaben" verfügt, um den zu überweisenden Betrag zu decken. Denken Sie daran, dass es sich bei den "Vermögenswerten" nicht nur um eine Anzahl oder einen Betrag handeln kann, sondern dass wir "Vermögenswerte" jeder Art schaffen können, je nachdem, welches System geschaffen werden soll.

Beispiele für Vermögenswerte:

- ✓ Intrinsischer Betrag einer "virtuellen oder Krypto-Währung" neuen Ursprungs.
- ✓ Daten, die sich auf digitale Daten beziehen (alle Arten von Dokumenten)
- ✓ Hash-Authentifizierungssignaturen für Zeichenfolgen oder alle Arten von Dateien.
- ✓ Jede Transaktion oder Übertragung digitaler Informationen oder deren Darstellung.

Der Saldo der "Assets" jedes Knotens wird mit Hilfe des Blocks (**GetBalance**) ermittelt.



Die zweite Anforderung ist die Angabe der Mindestparameter beim Senden einer Transaktion, nämlich

- ✓ Adresse der Quelle. - ist die Adresse, von der die Vermögenswerte versandt werden.
- ✓ Zieladresse. - ist die Adresse des Benutzers, der die gesendeten Assets erhalten wird.
- ✓ Aktiv. - Es handelt sich um die in jedem System angegebenen Daten mit immanentem Wert, die bei jeder Transaktion gesendet werden sollen.

Die obigen Adressen (Herkunft-Ziel) entsprechen den öffentlichen Schlüsseln der einzelnen Benutzer, als die Konten der einzelnen Benutzer erstellt wurden. Denken Sie daran, dass beim Erstellen eines Benutzerkontos zwei Arten von öffentlichen und privaten Schlüsseln erstellt werden.

Der öffentliche Schlüssel ist die Adresse, die im gesamten System freigegeben wird und die jeder Benutzer des Systems sehen und zum Senden oder Empfangen von Transaktionen verwenden kann.

Der private Schlüssel ist die Adresse, die lokal von jedem Knoten verwendet wird, und wie der Name schon sagt, ist er für den privaten Gebrauch bestimmt, mit dessen Hilfe digitale Signaturen erstellt werden, um die Sicherheit der gesendeten Transaktionen zu gewährleisten; dieser Schlüssel sollte niemals weitergegeben werden und ist für den lokalen Gebrauch bestimmt und eindeutig für den Quellknoten.

Um die vorherigen Parameter zu verwenden, werden wir uns auf zwei Repositories stützen, in denen die Adressen der "privaten Schlüssel", die sich in der keystore.db-Datenbank befinden, gespeichert sind und von lokalem Nutzen für jeden Knoten sein werden, ohne dass sie im System gemeinsam genutzt werden, und das andere Repository, in dem alle Adressen der "öffentlichen Schlüssel", die sich in der publickeys.db-Datenbank befinden, gespeichert sind, wird durch das "Peer-to-Peer"-Protokoll in allen Knoten des Systems gemeinsam genutzt.

Die Datenbanken "keystore.db" und "publickeys.db" wurden bereits im Anhang "KeyStore & PublicKeys Datenbankerstellung" erstellt.

Die Datenbank und das System für die Transaktionswarteschlange wurden bereits im Abschnitt "Installation und Konfiguration des RESTful Mini Sentinel-Netzes" erstellt. Wo wir bereits eine Datenbank für die Transaktionen mit dem Namen op.sqlite3 erstellt haben, haben wir hier zwei Tabellen, eine ist "trans", die sich um die Transaktionen kümmert (**Transaktionswarteschlange**), und die andere "sign", in der die digitalen Signaturen jeder Operation gespeichert werden.

Das SQLite RESTful-System ist das Backup-Netzwerk. Bei dieser Gelegenheit werden wir es der Einfachheit halber und zum Testen des Backup-Systems verwenden. Um jedoch die gleiche Datenbank op.sqlite3 in einem "Peer to Peer"-Modell zu ändern und zu verwenden, müssen wir die Datenbank nur in einem gemeinsam genutzten Modell wieder im Syncthing-System verwenden, dies werden wir später sehen.

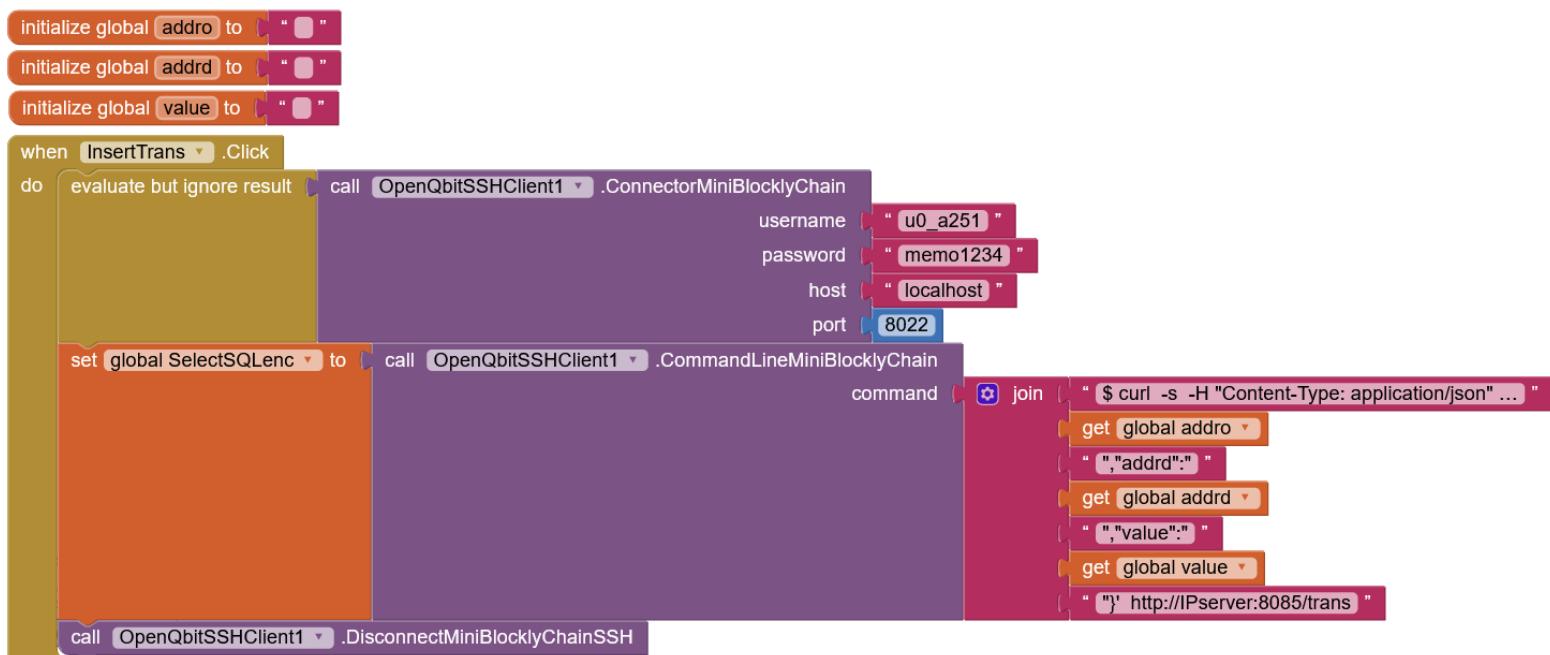
Wir werden damit beginnen, Operationen an die Transaktionswarteschlange zu senden, indem wir RESTful auf die op.sqlite-Datenbank anwenden.

Wir haben eine neue Transaktion in der "**trans**"-Tabelle erstellt

```
$ curl -s -H "Content-Type: application/json" -d
'{"addr": "MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",
"addrd": "MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh", "Wert": "999"}'
http://IPserver:8085/trans
```

```
# Ausgaben
{
  "id": 1,
  "addr": "MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",
  "addrd": "MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh",
  "Wert": "999".
}
```

Implementierung im App Inventor:



Die Eingabeparameter: addro, addrd, value sind Variablen, die in den Algorithmus eingegeben werden können und aus der Datenbank keystore.db extrahiert werden.

Siehe Anhang "KeyStore-Datenbankerstellung".

Wir fügen nun die digitalen Signaturen aus der vorherigen Transaktion ein. In der Tabelle "Zeichen"

```
$ curl -s -H "Inhaltstyp: Anwendung/Json" -d '{  
  "trans_id":1  
  "Zeichen": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",  
  "Hash": "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68"}'  
http://IPserver:8085/sign  
  
# Ausgaben  
{  
  "id": 1,  
  "trans_id": 1,  
  "Zeichen": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62",  
  "Hash": "f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68"  
}
```

**HINWEIS:** Die digitale Signatur (Sign) muss vor dem Senden der Curl Command-Anweisung erhalten werden, sie wird mit dem Block (**GenerateSignature**) erzeugt.

Die Eingabeparameter: Trans\_id, sign, hash sind Variablen, die in den Algorithmus eingegeben werden können und die digitale Signatur der Transaktion wird mit dem Block erzeugt (**GenerateSignature**).

Wir werden nun das Verfahren zur Erzeugung einer digitalen Signatur sehen, die auf jede Transaktion angewendet wird, die durchgeführt wird.

Erzeugung einer digitalen Signatur für die Transaktion. Wenn wir den Block (**GenerateSignature**) verwenden, haben wir als Ergebnis einen **Dateityp .sig diese Datei, die wir verwenden werden**, um sie im Sign-Feld in der Tabelle "sign" zu speichern, diese Signatur wird verwendet, wenn die Transaktions-Warteschlange verarbeitet wird, und es ist ein weiterer Parameter, um Sicherheit zwischen dem Ursprung und dem Ziel, sowie den Betrag oder die Art der Transaktion zwischen ihnen zu geben.

Um Binärdaten wie die file.sig zu verarbeiten, müssen wir sie nach base64 konvertieren und dann in der Vorzeichenvariablen in der Tabelle "sign" speichern. Dazu werden wir den Block (**EncoderFileBase64**) verwenden.

Wie der Wert des Hash-Feldes für die "Vorzeichen"-Tabelle jeder Transaktion berechnet wird:

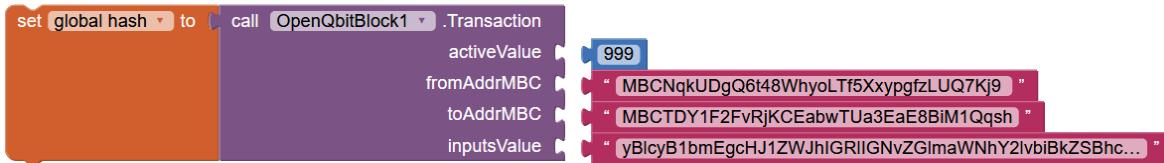
Transaktions-Hash = SHA256(Quelladresse + Zieladresse + Asset + fileBase64.sig)

Beispiel für den Hash-Typ SHA256:

**SHA256(MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9 + MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh + 999 + ).**

**Ausgabe:** 9d45198faaef624f2e7d1897dd9b3cede6ecca7fbac516ed1756b350fe1d56b4

Für die Berechnung des Hashes werden wir uns auf den Block(**Transaktion**) stützen müssen.



Der Ausgabeparameter ist der Hash, der für jede Transaktion gespeichert wird, die von einem beliebigen Knoten im System gesendet wird. Dieser wird im Vorzeichenfeld der Tabelle "sign" in der Basis op.sqlite

Bei der vorherigen Operation haben wir dafür gesorgt, dass nur ein eindeutiger und unwiederholbarer Hash jede an die Warteschlange gesendete Transaktion repräsentiert und dieser repräsentative Hash die Gesamtheit der übermittelten Informationen darstellt.

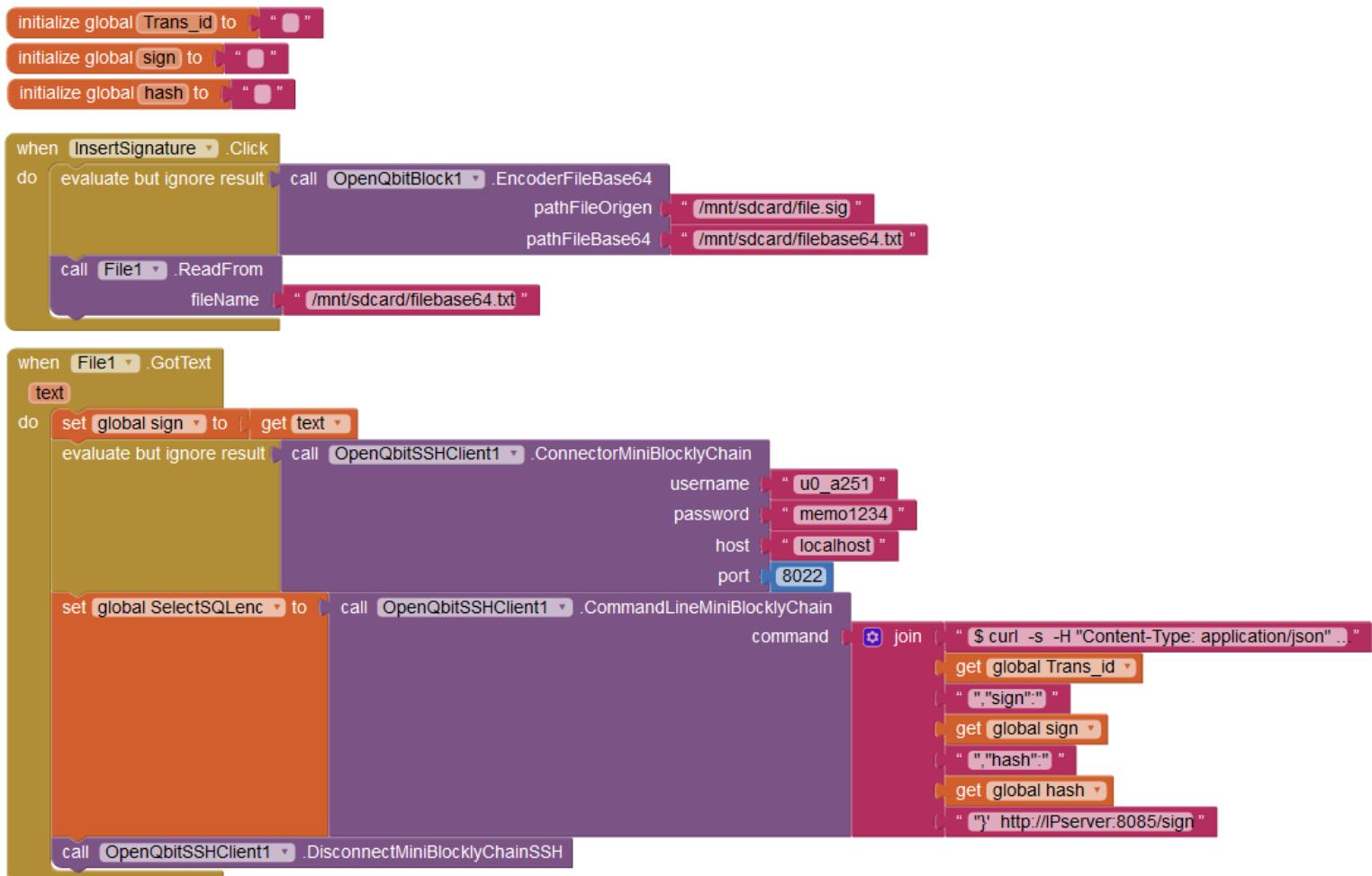
Das Einzige, was fehlt, ist die Trans\_id-Variable, die ein Identifikator sein kann, um zu unterscheiden, welcher Knoten für jede Transaktion gesendet wird. In unserem Beispiel werden wir die Trans\_id-Variable mit einem festen Wert von "1" setzen. Weil es der erste Knoten ist, der in unserem Netzwerk konfiguriert wird. Dieser Wert kann die Syntax je nach speziellem Design variieren, daher haben wir der Einfachheit halber einen einfachen Bezeichner gewählt.

Da wir alle Variablen definiert haben, werden wir die Struktur und Reihenfolge der Blockausführung innerhalb von App Inventor erstellen, um das EINFÜGEN in die Tabelle "sign" durchzuführen.

**ANMERKUNG:** Es ist wichtig zu berücksichtigen, dass jedes Senden einer Transaktion aus zwei EINFÜGUNGEN in der op.sqlite3-Datenbank besteht, von denen eine in der "trans"-Tabelle und ihre jeweiligen Werte in der "sign"-Tabelle vorgenommen werden müssen.

Beim Entwurf der op.sqlite3-Datenbank wurden zwei getrennte Tabellen erstellt, da es in Zukunft möglich ist, nur die "sign"-Tabelle zu verschlüsseln, da sie Informationen enthält, die nicht flach ins Netz gesendet werden sollen.

Wir werden die Ausführungsstruktur der Blöcke und Methoden innerhalb des App Inventor of the INSERT in der Tabelle "sign" erstellen.



Bis zu diesem Moment haben wir bereits das EINFÜGEN in die "sign"-Tabelle und in die "trans"-Tabelle abgeschlossen. Diese befinden sich innerhalb der **op.sqlite3-Datenbank**, und die in diese beiden Tabellen eingefügten Daten werden zur Erstellung der Transaktions-Warteschlange verwendet, die an alle Knoten zu ihrer Verarbeitung geschickt wird.

Zu diesem Zeitpunkt werden wir den Java SQLite-Redis Connector verwenden, der die Konvertierung der Daten aus der op.sqlite3-Datenbank in die Redis-Datenbank durchführt. Siehe Anhang "Java SQLite-Redis Code Connector".

Nach der Implementierung des SQLite-Redis Connectors wird die Transaktionswarteschlange über das Redis-System an alle Knoten des Systems geliefert.

Wir werden mit der Frage beginnen, wie wir die Transaktionswarteschlange ordnungsgemäß empfangen können, um sie verarbeiten zu können.

Die Transaktions-Warteschlange wird über den Redis-Dienst geliefert. Dies ist eine Echtzeit-Datenbank, die ihre jeweiligen Kontrollblöcke in der App Inventor-Umgebung hat.

Konfiguration der Redis-Umgebung innerhalb des blockweisen Systems von App Inventor.

Ein wichtiger Punkt ist, dass die Blöcke zur Steuerung eines Redis-Servers bis zum Zeitpunkt der Erstellung dieses Handbuchs nur im App Inventor-System verfügbar sind. Im Falle der Verwendung eines Blockly-Systems, das sich vom App Inventor unterscheidet, müssen Sie die Redis CLI (Command-Line) verwenden und dies durch das Senden von Befehlen an das Termux-Terminal über die Erweiterung (**OpenQbitSSHClient**) ausführen.

Beispiel für die Verwendung in Redis CLI (Command-Line):

Um alle **Etiketten** oder Schlüssel bei Redis zu erhalten.

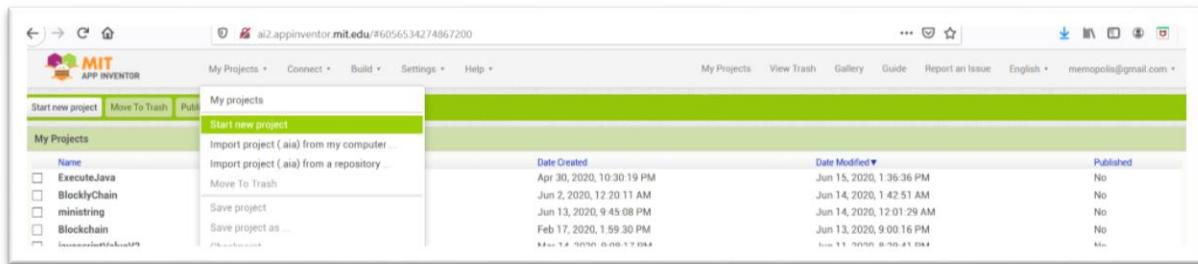
**\$ redis-cli SCHLÜSSEL '\*'**

Um einen Wert von einem Schlüssel zu erhalten.

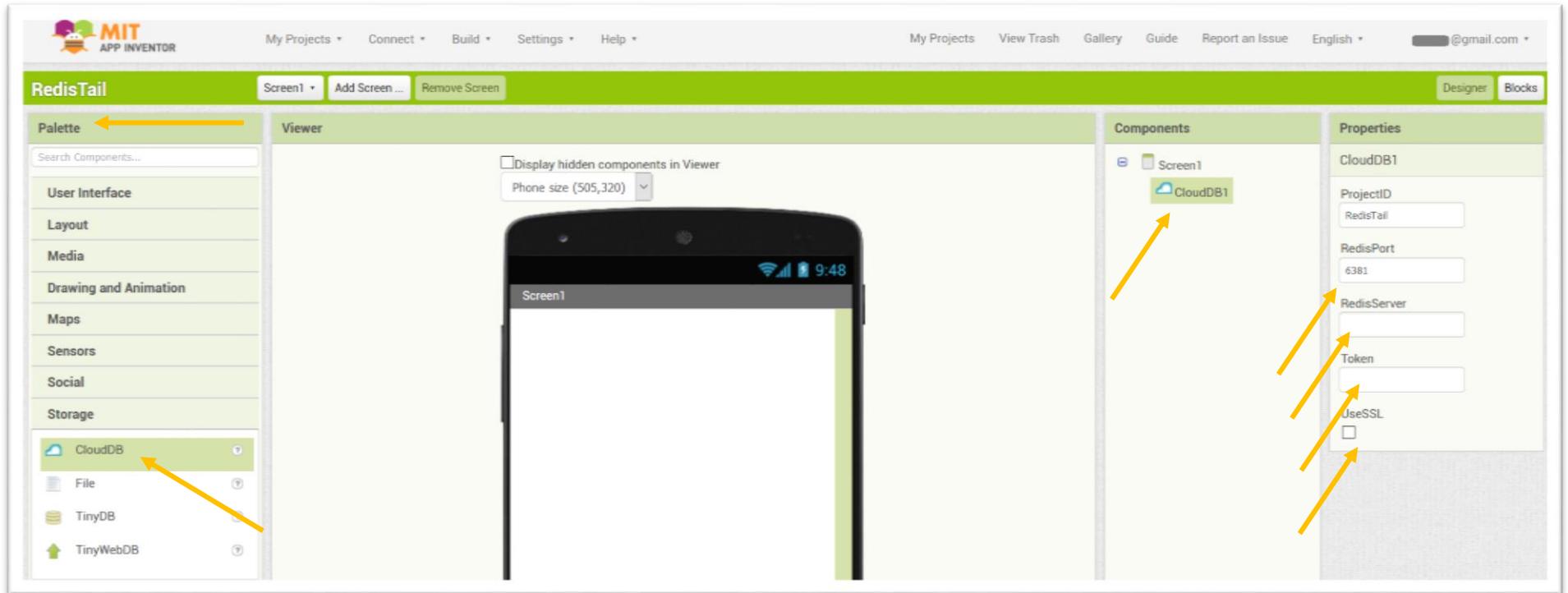
**\$ redis-cli GET <key>**

In unserem Fall, weil wir App inventor verwenden, werden wir prüfen, wie die Blöcke für die Arbeit mit Redis verwendet werden können.

Innerhalb von App Inventor haben wir mehrere Blöcke, die wir mit Redis verwenden können, wir haben damit begonnen, ein neues Projekt zu erstellen, das wir noch erstellen werden: Meine Projekte > Neues Projekt starten



Nachdem wir das Projekt erstellt haben, gehen wir nach links oben und im Abschnitt "Palette" klicken wir auf "Storage" und ziehen das Objekt "CloudDB". Dies wird alle Funktionen integrieren, um eine Verbindung zu einem Redis-Server zu konfigurieren.



Die Konfiguration ist sehr einfach, wir müssen nur die folgenden Parameter angeben, um eine Verbindung zu unserem Redis-Server (Lokal) herstellen zu können, der in unserem Knoten (Handy) läuft.

ProjektID. - Dies ist die ID, die das Label startet, das die Transaktionswarteschlange in Redis identifiziert.

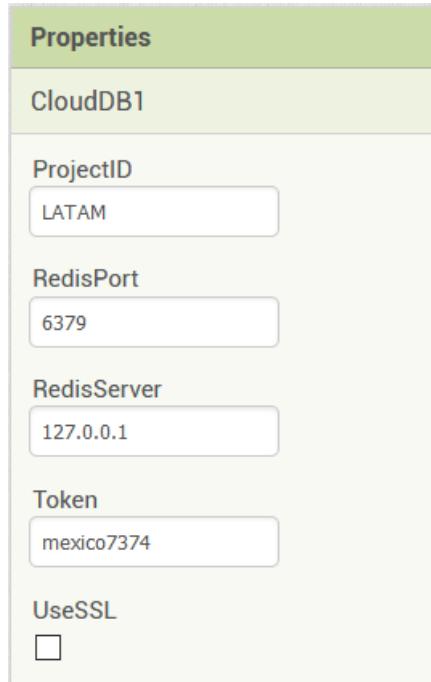
Umdisponieren. - Dies ist der Port des Redis-Servers, mit dem wir uns standardmäßig verbinden werden: 6379

RedisServer. - Dies ist in unserem Fall die Domäne oder lokale IP des Knotens, und die aller Knoten wird 127.0.0.1

Zeichen. - Dies ist das Passwort des Redis-Servers, mit dem die Verbindung hergestellt werden kann.

VerwendenSSL. - Diese Option gibt uns die Möglichkeit, SSL-Zertifikate zu verwenden, in unserem Fall lassen wir sie deaktiviert.

In unserem Modell- und Systementwurf werden wir in allen Netzwerknoten die folgenden Parameter haben.



Es ist an der Zeit, die Blöcke zu überprüfen, die uns die Kontrolle und Datenverarbeitung über eine Redis-Datenbankumgebung ermöglichen.

Wir beginnen mit dem Methodenblock (**DataChanged**)



Diese Methode liefert uns bei jeder Änderung des von uns konfigurierten Redis-Servers zwei Werte:

Tag. - Dieser Wert ist das Tag, das die Transaktionswarteschlange identifiziert.

Wert. - Dieser Wert enthält die Liste aller zur Verarbeitung gesendeten Transaktionen. Dieser Wert ist eine Stringliste vom Typ <String>.

Im Falle des "Wertes" ist das der Punkt, an dem wir beginnen werden, unsere Informationsverarbeitung wie folgt durchzuführen.

Da Sie uns eine Kette aller Hashwerttransaktionen zur Verfügung stellen, werden wir zunächst prüfen, ob diese Kette gültig ist und ob sie seit ihrer Entstehung nicht verändert wurde. Wir tun dies, indem wir einen sehr nützlichen Algorithmus zur Überprüfung großer Informationsmengen verwenden.

Wir werden den Merkle-Baum-Algorithmus verwenden. Die Anwendung dieses Algorithmus gibt uns eine Sicherheit in der Integrität der Informationen, die wir erhalten (Transaktionswarteschlange).

Sehen wir uns das folgende Beispiel einer Transaktions-Warteschlange in Redis an, wir führen die Redis CLI (Command-Line) von einem Termux-Terminal aus, um den "LATAM:HASH"-Schlüssel zu überprüfen, wir müssen bereits den SQLite-Redis Connector ausgeführt haben, um diesen Schlüssel abfragen zu können.

```
C:memor>redis-cli
127.0.0.1:6379> get LATAM:HASH
"9d45198faaef624f2e7d1897dd9b3cde6ecca7fbac516ed1756b350fe1d56b4,"
f71c801a5fd25fc303ebc8c616204b4877ffb93006ec6a88bc30acf43ec250f5,"\60f8a3bcac1
ea7d38e86efbc3e3e3e00480807d23f980391000766e804ed14ecb2,
8a6dfe1d38c22e0f9212052efa6136da3edf1fb1b2a3e25224ac3d689124b754]
127.0.0.1:6379>
```

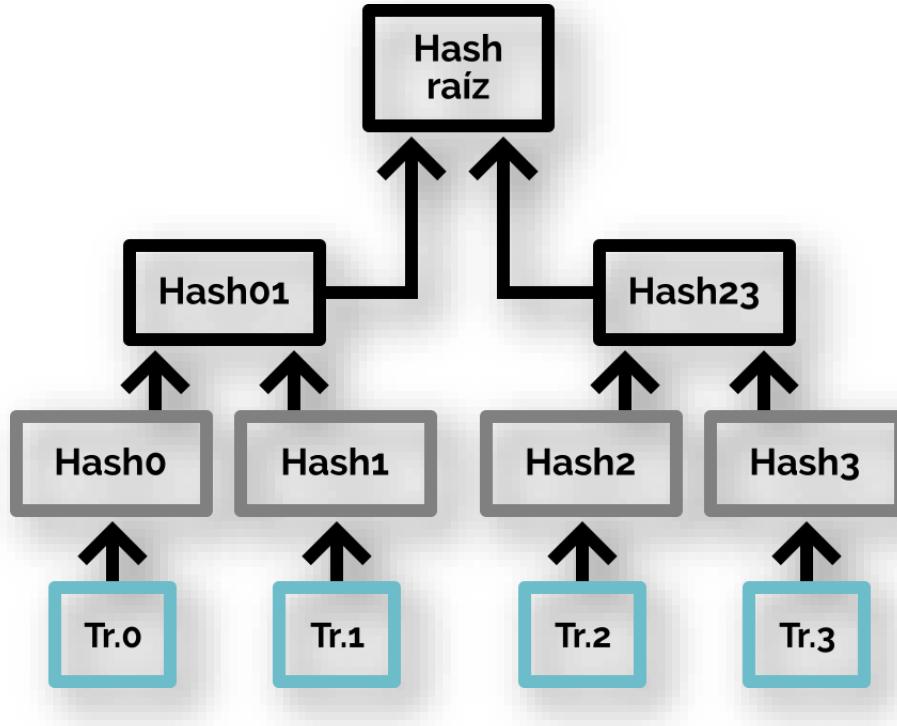
Die vorherige Transaktions-Warteschlange hat nur drei Elemente, da wir vier Hashes sehen, die eine einzelne Transaktion repräsentieren, die sie ausmachen, und sie sind die Hashes jeder Transaktion, die ursprünglich in die op.sqlite3-Datenbank innerhalb der "trans"- und "sign"-Tabellen injiziert wurde.

Jetzt ist es an der Zeit zu verstehen, wie der Merkle-Baum funktioniert.

Ein Hash-Merkle-Baum ist eine binäre oder nicht-binäre Datenbaumstruktur, in der jeder Knoten, der kein Blatt ist, mit dem Hash der Verkettung der Labels oder Werte seiner Kindknoten versehen ist. Sie sind eine Verallgemeinerung von Hash-Listen und Hash-Strings.

In unserem Fall werden wir die folgende Berechnung durchführen, um den Merkle-Baum für vier Elemente zu berechnen, indem wir das Ergebnis der Verkettung von Datenpaaren berechnen und ihre jeweiligen Hashes erhalten. Die Ergebnisse der ersten Ebene werden auf die Ergebnisse der zweiten Ebene angewendet, bis es nur noch ein einziges abschließendes Element gibt, das als Hash-Merkleroot (Wurzelhash) bezeichnet wird.

Sehen wir uns das folgende Diagramm an, das diesen Prozess beschreibt.



Die hashbasierte Transaktionswarteschlange wird über den Block (**GetMerkleRoot**) herausgezogen



Hash-Wurzel:

51431822de7c94b90dc06d47b8f6275f315a4976c8479d30c32747fa90325432

Das Ergebnis wird dann mit dem LATAM:merkleroot-Schlüssel im lokalen Redis-System verglichen, und beide Schlüssel müssen übereinstimmen, um die Datenintegrität zu überprüfen.

Ein weiterer Sicherheitspunkt, den der Merkle-Baum bietet, ist die Bestätigung, dass eine bestimmte Transaktion von ihrem Ursprung her in die Transaktionswarteschlange

aufgenommen wurde und dass sie nicht durch ein externes oder internes Kommunikationsmittel auf betrügerische Weise eingeführt wurde.

Für den Fall, dass die Kette in ihrer Summierung aus ungeraden Elementen besteht, dupliziert sich das letzte Element, um eine Anordnung für den Algorithmus zu haben und die Ausführung des Algorithmus zu starten.

Ein weiterer, nicht minder wichtiger Punkt ist der Zeitpunkt, zu dem überprüft werden muss, ob jede Transaktion mit ihrem Ursprungs-Ziel übereinstimmt und ob es sich bei dem Vermögenswert um denjenigen handelt, der von der Ursprungsadresse gesendet wurde.

Hier befindet sich der Block (VerifySignature).



Vor der Ausführung des vorherigen Blocks müssen Sie zunächst die Datei (file.sig) im Binärformat aus der Tabelle "sign" herunterladen:

```
sqlite3 op.sqlite3 "Zeichen von Zeichen auswählen where=id_addr;"
```

**id\_addr:** Dies ist die ID der Quelladresse der "trans"-Tabelle.

Die vorherige Suchanfrage liefert uns die Daten der digitalen Signatur der aktuellen Transaktion im Base64-Format. Um diese in ihr ursprüngliches Format (binär) zu konvertieren, benötigen wir den Block(**DecoderFileBase64**).

Da wir unsere binäre Datei (file.sig) haben, müssen wir den öffentlichen Schlüssel des Ursprungs und den öffentlichen Schlüssel des Empfängers ebenfalls in binärem Format in das System laden, wobei die vier Daten in ihren jeweiligen Formaten vorliegen.

- ✓ Heimadresse mit öffentlichem Schlüssel
- ✓ Öffentliche Schlüsseladresse des Empfängers
- ✓ Aktiv gesendet.
- ✓ Digitale Unterschrift (file.sig)

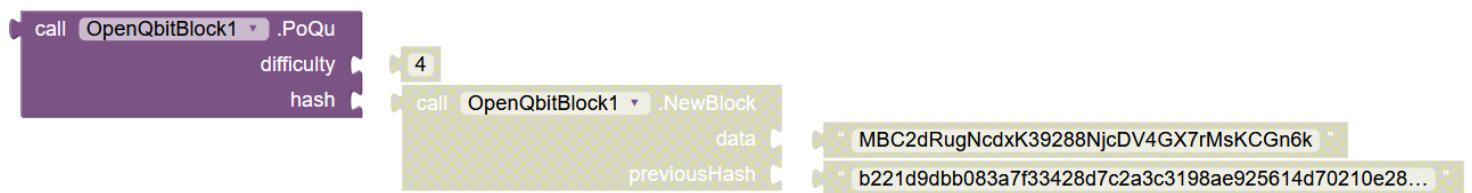
Öffentliche Schlüssel im Binärformat können im entsprechenden Format (binär) von der gemeinsam genutzten Datenbank publickeys.db heruntergeladen werden.

Dieser Prozess muss für jeden einzelnen Vorgang in der Transaktionswarteschlange ausgeführt werden.

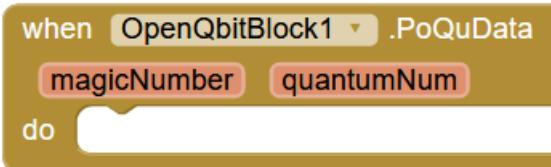
Abschließend werden wir überprüfen, wie das System einen Knoten einvernehmlich auswählt, um ausgewählt werden zu können, den nächsten Block der Blockkette hinzuzufügen und derjenige zu sein, der die Transaktionswarteschlange verarbeitet.

Diese Art der Auswahl des Gewinnerknotens für die Verarbeitung der Transaktionswarteschlange basiert auf unserem Algorithmus, der für ein Mini-BlocklyChain-System entwickelt wurde.

Wie wir den PoQu "Proof of Quantum"-Konsens umgesetzt haben. Dieses Konsensverfahren basiert auf der Erzeugung von Quantenzufallszahlen und wird mit Hilfe des (**PoQu**)-Blocks angewendet.



Zuerst erhalten wir zwei Parameter, die uns der Block (**PoQu**) in der Methode (**PoQuData**) liefert. Die Parameter sind die magische Zahl und das QuantumNum.



Die **magischeZahl** ist die Zahl "nonce", ist eine ganze Zahl, die erhalten wird, indem man ein internes PoW mit Schwierigkeit nicht größer als 5 macht, diese Zahl ist dafür zuständig, eine erste Anforderung an den Knoten mit der magischenZahl zu stellen, die der Knoten in der Lage sein wird, eine Anforderung zu stellen, um eine Zahl des Systems der Generierung von Quantenzufallszahlen zu erhalten, die im Bereich von 0 bis 1 liegt, diese Zahl wird eine Wahrscheinlichkeit geben, die zufällig für den Knoten in der Ausführung festgelegt wird, die in der Tabelle namens "Abstimmung" gespeichert wird.

Die "Vote"-Tabelle wird das von jedem Knoten berechnete QuantumNum speichern, diese Speicherung erfolgt mit einer Anzahl von Knoten bis zu einer bestimmten Zeit, die in jedem Systemdesign festgelegt wird, für das es empfohlen wird, Einträge  $((N/2) + 1)$  zu haben, wobei "N" die Anzahl der im System verfügbaren Knoten ist und durch eine Aktion im "cron"-Taskmanagement-Tool jedes Knotens festgelegt oder kontrolliert werden kann.

Ein wichtiger Punkt ist, dass Sie zu diesem Zeitpunkt bereits eine lokale Zeitsynchronisation jedes Knotens durch das automatische System des Mobiltelefons etabliert haben sollten, falls Sie das "Peer to Peer"-Netzwerk bei der Übertragung der Transaktionswarteschlange verwenden.

Die Konfiguration des Cron-Agenten in den Knoten. Siehe Abschnitt "Zeitsynchronisation in den Systemknoten (Mobiltelefon) in Minuten und Sekunden".

Da wir in diesem Beispiel das Backup-Kommunikationsnetzwerk verwenden, brauchen wir die Synchronisation von Minuten und Sekunden für die Knoten nicht, da unser Beispiel ein "Client-Server"-Schema belegt - diese Art der Kommunikation findet nur im Übertragungsprozess der Transaktionswarteschlange statt. Alle anderen Prozesse zwischen den Knoten erfolgen über eine "Peer-to-Peer"-Kommunikation.

Nun werden wir uns die Struktur, das Design und die Erstellung der "Abstimmungs"-Tabelle ansehen, die sich in der Datenbank "quorum.db" befindet, die wir in diesem Beispiel ebenfalls erstellen werden.

**\$ sqlite3**

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

Geben Sie ".help" für Benutzungshinweise ein.

Verbunden mit einer transienten In-Memory-Datenbank.

Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.

sqlite> .open quorum.db

sqlite> CREATE TABLE-Abstimmung (id ganzzahliger Primärschlüssel AUTOINCREMENT NOT NULL, node\_imei VARCHAR NOT NULL, quantumNum INTEGER NOT NULL, nonce INTEGER NOT NULL);

sqlite> .quit

Die Erstellung derselben **Abstimmungstabelle** ist unten dargestellt, allerdings durch Einführung der SQL-Anweisung in einer segmentierten Form:

**\$ sqlite3**

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

Geben Sie ".help" für Benutzungshinweise ein.

Verbunden mit einer transienten In-Memory-Datenbank.

Verwenden Sie ".open FILENAME", um eine persistente Datenbank erneut zu öffnen.

sqlite> .open quorum.db

sqlite> TABELLE SCHAFFEN Abstimmung (

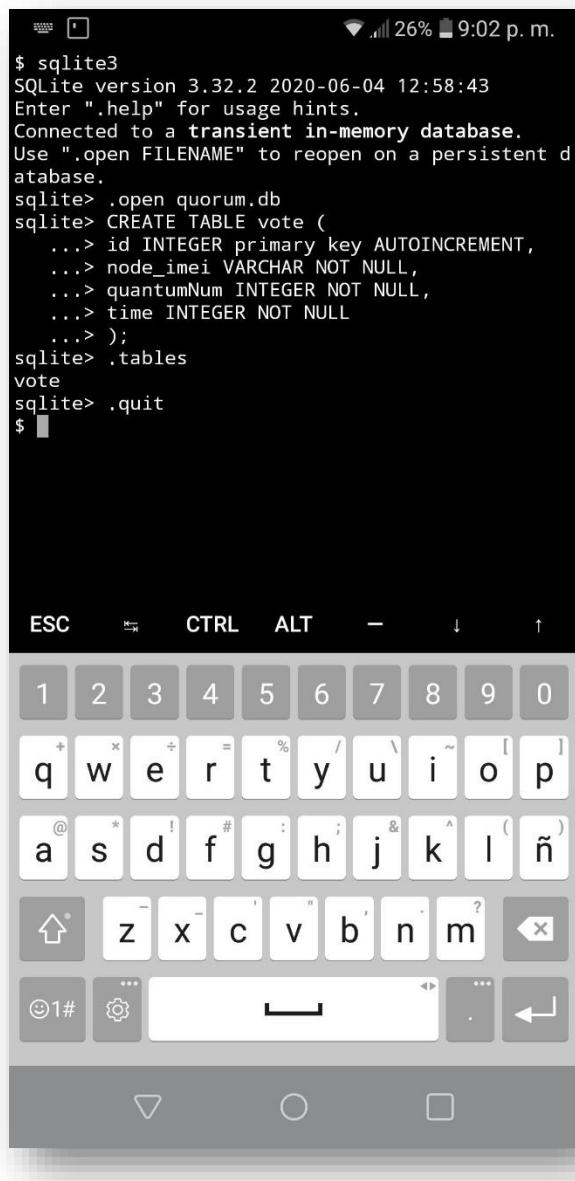
...> id ganzzahliger Primärschlüssel AUTOINKREMENT

...> knoten\_imei VARCHAR NICHT NULL,

...> quantumNum INTEGER NICHT NULL,

```
...> nonce INTEGER NICHT NULL  
...> );  
sqlite> .tabellen  
Abstimmung  
sqlite> .quit
```

Wir werden etwas Ähnliches bei der Schaffung der Basis "quorum.db" und der Tischabstimmung erleben.

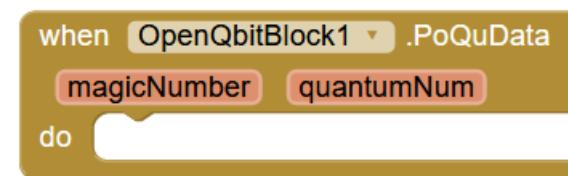


Nun wollen wir sehen, wie wir die Werte aus der "Abstimmungstabelle" erhalten.

knoten\_imei. - Diesen Wert **erhalten** wir über den Block (**GetDeviceID**).



quantumNum - Dieser Wert ist eines der Ergebnisse der (**PoQuData**)-Methode, die unter Verwendung des (**PoQu**)-Blocks erhalten wird.



nonce. - Dieser Wert ergibt sich daraus, dass der Block (**PoQu**) bereits integral und gleich der magischen Zahl der Methode (**PoQuData**) ausgeführt wurde.

Nach dem Ausfüllen der EINFÜGUNGEN in der Tabelle "Abstimmung" ist es notwendig, eine Kopie der Datenbank anzufertigen.

Nach einer bestimmten Zeit und basierend auf dem Design jedes Systems wird der "cron"-Dienst auf jedem Netzwerknoten ausgeführt und hat das nächste SELECT, das in der "Vote"-Tabelle verarbeitet wird:

```
SELECT node_imei FROM Abstimmung WHERE magicNumber= (SELECT max(magicNumber) FROM Abstimmung);
```

Das vorherige SELECT gibt das IMEI-Ergebnis mit höherer Wahrscheinlichkeit zurück, jetzt vergleicht jeder Knoten, der SELECT ausführt, seine IMEI mit dem Ergebnis IMEI, und nur der Knoten, der übereinstimmt, erstellt eine Datei mit der höchsten Wahrscheinlichkeitszahl, die im Netzwerk "Peer to Peer" durch eine Datei mit dem Format IMEI.mbc, die die IMEI des Gewinnerknotens enthält, repliziert wird.

Der Gewinnerknoten kann mit der Verarbeitung der Transaktionswarteschlange beginnen. Basierend auf allen oben genannten Blöcken.

Zwei wichtige Punkte sind, dass abhängig von jeder Systemerstellung drei Prozesse von jedem Designer überprüft und angepasst werden müssen.

1.- Wenn der siegreiche Knoten mit der Verarbeitung der Transaktionswarteschlange beginnt, muss eine Methode oder ein Prozess implementiert werden, bei dem geprüft

werden muss, ob der siegreiche Knoten online ist und die Kommunikation mit dem Netzwerk nicht verloren gegangen ist.

2.- Wenn die Verarbeitung der Transaktionswarteschlange beginnt, muss der siegreiche Knoten zwei Steuerflags starten, die den Beginn der Verarbeitung anzeigen, und ein weiteres, um zu bestätigen, dass die Verarbeitung der Transaktionswarteschlange abgeschlossen ist. Diese beiden Flags müssen im Netzwerk von allen Knoten gemeinsam genutzt werden, um Verbindungsaußfälle oder Verarbeitungsfehler oder zu viel Verarbeitungszeit zu lokalisieren.

3.- Im Falle eines Kommunikationsfehlers oder eines anderen Ereignisses, bei dem der gewinnende Knoten die Transaktionswarteschlange nicht verarbeiten konnte, muss der nächste Knoten im unmittelbaren Wahrscheinlichkeitsbereich gewählt werden.

Der vorherige Punkt kann mit einem Dienst kontrolliert werden, der überprüft, ob der Gewinnerknoten online ist und den "cron"-Dienst nutzen kann. Das Skript muss für jeden entworfenen Fall entwickelt werden, jedoch wird unten ein generisches Beispiel eines Shell-Skripts gezeigt, so dass es entsprechend den Bedürfnissen jedes Mini-Blocklychain-Systems modifiziert werden kann.

```
#!/bin/bash
dir="/data/data/com.termux/files/home/Sync/imei";
wenn [ !$(ls $Verzeichnis) ] ]
dann
sqlite3 quorum.db "UPDATE-Abstimmung SET magicNumber=0 WHERE magicNumber=
(SELECT max(magicNumber) FROM Abstimmung);";
sonst
MAX_NUM=$(sqlite3 quorum.db "SELECT max(magicNumber) FROM Abstimmung;")
IMEI_quorum=$(sqlite3 quorum.db "SELECT node_id FROM vote WHERE=MAX_NUM")
IMEI_local=$(cat device_imei) // Den Block verwenden (GetDevice)
if [ IMEI_quorum -eq IMEI_local ][ IMEI_quorum -eq IMEI_local ]
dann
    Berühren Sie $MAX_NUM > IMEI.mbc
fi
fi
Ausfahrt
```

### 31. Anhang "Integration mit Ethereum & Bitcoin-Umgebungen".

Jetzt werden wir sehen, wie wir die beiden weltweit bekanntesten, auf Kryptomonie spezialisierten Blockkettensysteme wie Ethereum und Bitcoin integrieren können. Beginnen wir mit der Installation der Software, die uns helfen wird, alle möglichen Transaktionen in der Ethereum-Umgebung durchzuführen.

Was ist Ethereum?

Ethereum ist eine Open-Source-Plattform, dezentralisiert im Gegensatz zu anderen Blockketten, Ethereum kann viel mehr tun. Es ist programmierbar, was bedeutet, dass Entwickler damit neue Arten von Anwendungen erstellen können.

Diese dezentralisierten Anwendungen (oder "dapps") profitieren von den Vorteilen der Kryptomontage- und Blockkettentechnologie. Sie sind zuverlässig und vorhersehbar, was bedeutet, dass sie, sobald sie einmal in das Ethereum "geladen" sind, immer termingerecht laufen. Sie können digitale Vermögenswerte kontrollieren, um neue Arten von Finanzanwendungen zu erstellen. Sie können dezentralisiert sein, was bedeutet, dass sie nicht von einer einzigen Entität oder Person kontrolliert werden.

Gegenwärtig erstellen Tausende von Entwicklern auf der ganzen Welt Anwendungen auf Ethereum und erfinden neue Arten von Anwendungen, von denen Sie heute viele nutzen können:

- Krypto-Währungsportfolios, die es Ihnen ermöglichen, günstige, sofortige Zahlungen mit ETH- oder anderen Vermögenswerten zu tätigen
- Finanzanwendungen, die es Ihnen ermöglichen, Ihre digitalen Vermögenswerte zu leihen, auszuleihen oder zu investieren
- Dezentralisierte Märkte, die es Ihnen ermöglichen, digitale Bestände auszutauschen oder sogar "Vorhersagen" über Ereignisse in der realen Welt auszutauschen.
- Spiele, bei denen Sie Vermögenswerte im Spiel haben und sogar echtes Geld gewinnen können.
- Sie verfügt über intelligente Verträge, d.h. Programme mit Vereinbarungen, die auszuführen sind, wenn die Voraussetzungen, mit denen sie ausgearbeitet oder geschaffen wurde, erfüllt sind.

Intelligente Verträge weisen Ähnlichkeiten mit **DApps** (dezentralisierte Anträge) auf, trennen sie aber auch von einigen wichtigen Unterschieden.

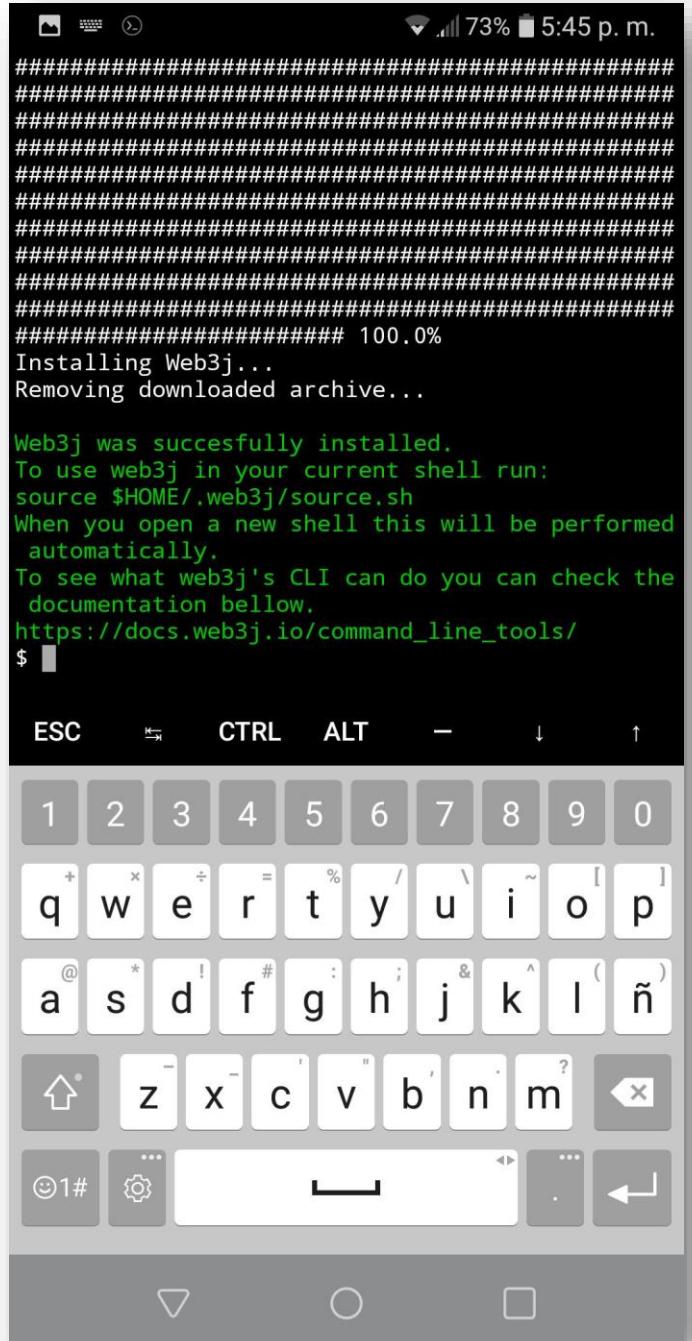
Wie intelligente Verträge ist ein DApp eine Schnittstelle, die einen Benutzer über ein dezentralisiertes Peer-Netzwerk mit einem Dienst eines Anbieters verbindet. Aber während bei intelligenten Verträgen eine feste Teilnehmerzahl erforderlich ist, gibt es bei DApps keine Begrenzung der Nutzerzahl. Darüber hinaus sind sie nicht nur auf finanzielle Anwendungen

wie intelligente Vertrage beschrnkt: ein DApp kann jeden erdenklichen Zweck erfllen. In unserem Fall werden wir die Bibliothek "Web3j" verwenden, die in Java entwickelt wurde und die eine einfache und intuitive Interaktion mit der Ethereum-Blockkette ermglicht.

Führen Sie den folgenden Befehl aus, um "Web3j" zu installieren:

```
$ kringeln -L qet.web3j.io | sh
```

```
$ curl -L https://get.web3j.io | sh
% Total      % Received   % Xferd  Average Speed
Time      Time      Time  Current
Dload  Upload
Total  Spent    Left  Speed
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0
:--:--  0:00:05 :--:--:-- 0
0       0       0       0       0       0       0       0
100  5143  100  5143  0       0       821  0
:00:06  0:00:06 :--:--:-- 20572
% Total      % Received   % Xferd  Average Speed
Time      Time      Time  Current
Dload  Upload
Total  Spent    Left  Speed
0       0       0       0       0       0       0       0
:--:--  :--:--  :--:-- 0
```



Dann müssen wir die Umgebungsvariable **\$JAVA\_HOME** mit dem Pfad erstellen, in dem sich die ausführbare Datei "java" befindet, da die Bibliothek diese Variable durchsuchen wird, um erfolgreich ausgeführt werden zu können.

```
$ JAVA_HOME= /data/data/com.termux/files/usr/bin
```

Sobald die Variable erstellt ist, müssen wir in das Verzeichnis gehen, in dem die Bibliothek "Web3j" installiert wurde, indem wir den folgenden Befehl ausführen, ein wichtiger Punkt ist, dass das Verzeichnis versteckt wird, nachdem dr der Befehl von "cd" einen Punkt "." Und dann das Verzeichnis ohne Leerzeichen, wie folgt:

```
$ cd .web3j
```

Sobald wir drin sind, testen wir mit dem folgenden Befehl, ob die Bibliothek ordnungsgemäß läuft:

```
./web3j-Version
```

Es führt zu etwas sehr Ähnlichem:

The screenshot shows a Termux terminal window. At the top, there are system status icons: battery level (9%), signal strength, and the time (12:49 a.m.). Below the status bar, the terminal prompt is '\$'. The user runs the command '\$ ls', which lists three files: 'source.sh', 'web3j', and 'web3j-4.5.16'. Then, the user runs '\$ ./web3j version', which outputs a large, stylized version string composed of various symbols like '|', '—', '(', ')', and '/'. Below this, the output continues with 'Version: 4.5.16' and 'Build timestamp: 2020-03-06 14:13:49.943 UTC'. The bottom of the screen shows standard terminal navigation keys: ESC, CTRL, ALT, and arrow keys.

Später werden wir eine Brieftasche erstellen, die in der Blockkettenumgebung von Ethereum wie folgt verwendet werden soll:

## § ./web3j Brieftasche erstellen

Der vorherige Befehl gibt uns die Adresse von 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
$
```

Das Ergebnis ist eine Datei im JSON-Format, die die Adresse und die verschlüsselten Daten enthält, die später zur Generierung des privaten Schlüssels für das gerade erstellte Konto verwendet werden sollen.

Diese Datei vom Typ JSON wird in einem Standardverzeichnis namens keystore erstellt.

Von hier aus können wir bereits Operationen durchführen, die den Befehl Web3j verwenden und/oder ausführen.

Wir können dies durch die Verwendung der Erweiterung (**ConnectorSSHClient**) erreichen.

Um zu wissen, wie die verschiedenen Parameter und Operationen der "Web3j"-Bibliothek zu verwenden sind, können wir uns helfen, indem wir die Dokumentation auf ihrer offiziellen Website konsultieren.

<https://docs.web3j.io/>

Für die Bitcoin-Umgebung haben wir zwei Möglichkeiten: Wir können den Block () verwenden, der das Bitcoin-Konto und seine jeweiligen öffentlichen und privaten Schlüssel generiert, und wir können diese Schlüssel zur Integration in die Bitcoin-Umgebung verwenden, indem wir die Java-Bibliothek "Bitcoij" installieren.

\$ npm bitcoinj installieren



```
$ npm install bitcoinj
+ bitcoinj@0.0.0
added 1 package from 1 contributor and audited 2
09 packages in 20.528s
$
```

Um diese Bibliothek zu nutzen, werden wir uns auf ihre offizielle Website stützen.

<https://bitcoinj.github.io/>

Eine zweite Option zur Integration in die Bitcoin-Blockchain-Umgebung ist die Installation des Bitcoind-Pakets für Termux, wie unten gezeigt.

\$ apt bitcoin installieren



```
$ pkg install bitcoin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bitcoin
0 upgraded, 1 newly installed, 0 to remove and 1
9 not upgraded.
Need to get 3601 kB of archives.
After this operation, 15.0 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm bitcoin arm 0.20.0 [3601
kB]
Fetched 3601 kB in 2s (1253 kB/s)
Selecting previously unselected package bitcoin.
(Reading database ... 19110 files and directo
ries currently installed.)
Preparing to unpack .../bitcoin_0.20.0_arm.deb .
..
Unpacking bitcoin (0.20.0) ...
Setting up bitcoin (0.20.0) ...
$
```

Wenn wir nun den bitcoind-Agenten auf dem Termux-Terminal ausführen, erstellt er automatisch eine Adresse im Verzeichnis:

/data/data/com.termux/files/hme/.bitcoin

In diesem Fall von Bitcoin werden wir uns auf die folgende Dokumentation des "Bitcoind"-Agenten stützen.

In diesem Fall können wir auch die Erweiterung (**ConnectorSSHClient**) verwenden, um den "bitcoind"-Agenten je nach Bedarf auszuführen.

Beschreibung der Parameter für die Verwendung mit Bitcoind.

## NAME

bitcoind - Start des Bitcoin Core Daemon

## SYNOPSIS

**bitcoind** [*Optionen*] *Bitcoin Core Daemon* starten

## BESCHREIBUNG

Bitcoin-Kerndämon starten

## OPTIONEN

-?

Drucken Sie diese Hilfemitteilung aus und beenden Sie

### **-alertnotify=<cmd>**

Führen Sie den Befehl aus, wenn eine entsprechende Warnung empfangen wird oder wir einen sehr langen Fork sehen (%s in cmd wird durch die Nachricht ersetzt)

### **-annahmegültig=<hex>**

Wenn sich dieser Block in der Zeichenfolge befindet, gehen Sie davon aus, dass er und seine Vorfahren gültig sind, und überspringen Sie möglicherweise Ihre Schreibprüfung (0 für alle prüfen, Standard:

0000000000000000f1c54590ee18d15ec70e68c8cd4cfbadb1b4f11697eee, testnet:  
000000000000000037a8cd3e06cd5edbfe9dd1dbcc5dacab279376ef7fcf2b4c75)

### **-blocknotify=<cmd>**

Führen Sie den Befehl aus, wenn sich der beste Block ändert (%s in cmd wird durch den Blockhash ersetzt)

**-Blockwiederaufbauextratxn=<n>**

Zusätzliche Transaktionen, die für kompakte Blockrekonstruktionen im Speicher gehalten werden müssen (Standard: 100)

**-blocksdir=<dir>**

Geben Sie das Blockverzeichnis an (Standard: <datadir>/blocks)

**-nur im Block**

Wenn Sie Transaktionen von Netzwerkpartnern ablehnen. Portfolio- oder RPC-Transaktionen sind nicht betroffen. (Voreinstellung: 0)

**-conf=<archivieren>**

Geben Sie die Konfigurationsdatei an. Den relativen Pfaden wird der Ort des Datadir vorangestellt. (Standard: bitcoin.conf)

**-Dämon**

Es läuft im Hintergrund wie ein Dämon und nimmt die Befehle

**-datadir=<dir>**

Geben Sie das Datenverzeichnis an

**-dbcache=<n>**

Maximale Datenbank-Cache-Größe <n> MiB (4 bis 16384, Voreinstellung: 450). Darüber hinaus wird für diesen Cache ungenutzter Mempool-Speicher gemeinsam genutzt (siehe -maxmempool).

**-debuglogfile=<file>**

Geben Sie den Speicherort der Debug-Protokolldatei an. Relativen Pfaden wird ein netzwerkspezifischer Datenadir-Speicherort vorangestellt. (-nodebuglogfile zu deaktivieren; Standard: debug.log)

**-einschließlichf=<Datei>**

Geben Sie eine zusätzliche Konfigurationsdatei an, relativ zum Pfad **-datadir** (kann nur aus der Konfigurationsdatei heraus verwendet werden, nicht von der Befehlszeile aus)

**-loadblock=<Datei>**

Importieren von Blöcken aus der externen Datei blk000???.dat beim Start

**-maxmempool=<n>**

Halten Sie den Transaktionsspeicherpool unter <n> Megabyte (Standard: 300)

**-maxorphantx=<n>**

Maximum <n> abschaltbare Transaktionen im Speicher halten (Standard: 100)

**-mempoolexpiry=<n>**

Behalten Sie Transaktionen in mempool nicht länger als <n> Stunden (Standard: 336)

**-par=<n>**

Legt die Anzahl der Verifizierungs-Threads im Bindestrich fest (-6 bis 16, 0 = automatisch, <0 = lässt alle Kerne frei, Standard: 0)

**-Persistenzmempool**

Wenn Sie den mempool beim Herunterfahren speichern und beim Neustart laden (Standard: 1)

**-pid=<Datei>**

Geben Sie die PID-Datei an. Relativen Pfaden wird ein netzwerkspezifischer Datenadir-Speicherort vorangestellt. (Standard: bitcoind.pid)

**-punza=<n>**

Reduzieren Sie den Lagerbedarf, indem Sie das Beschneiden (Entfernen) alter Blöcke ermöglichen. Dadurch kann die RPC-Schnittkette aufgerufen werden, um bestimmte Blöcke zu entfernen, und sie ermöglicht das automatische Beschneiden alter Blöcke, wenn eine Zielgröße in MIB angegeben ist. Dieser Modus ist nicht kompatibel mit **-txindex** und **-rescan**. Warnung: Um diese Einstellung rückgängig zu machen, ist es notwendig, die gesamte Blockkette erneut herunterzuladen (Standard: 0 = Blockbeschneidung deaktivieren, 1 = manuelle Beschneidung über RPC zulassen, >=550 = Blockdateien automatisch beschneiden, um unter der in MIB angegebenen Zielgröße zu bleiben)

**-reindizieren**

Stellt den Zeichenkettenstatus und den Blockindex von blk\*.dat-Dateien auf der Festplatte wieder her

### **-reindex-Kettenstaat**

Rekonstruieren Sie den Zustand der Kette aus den aktuell indizierten Blöcken. Wenn Sie sich im Bereinigungsmodus befinden oder wenn die Blöcke auf der Platte beschädigt sein könnten, verwenden Sie stattdessen den vollständigen **Neuindex**.

### **-Seufz**

Erstellen neuer Dateien mit Standard-Systemberechtigungen statt umask 077 (nur wirksam bei deaktivierter Portfolio-Funktionalität)

### **-txindex**

Pflegen Sie einen vollständigen Transaktionsindex, der vom getrawtransaction rpc-Aufruf verwendet wird (Standard: 0)

### **-Version**

Druck & Go-Version

Verbindungsoptionen:

#### **-addnode=<ip>**

Fügen Sie einen Knoten hinzu, zu dem Sie eine Verbindung herstellen wollen, und versuchen Sie, die Verbindung offen zu halten (weitere Informationen finden Sie in der RPC-Befehlshilfe des "Addendums"). Diese Option kann mehrmals angegeben werden, um mehrere Knoten hinzuzufügen.

#### **-bancore=<n>**

Schwellenwert für das Trennen der Verbindung von sich schlecht benehmenden Kollegen (Standard: 100)

#### **-In der Zwischenzeit...**

Anzahl von Sekunden, um zu verhindern, dass sich schlecht benehmende Kollegen wieder verbinden (Voreinstellung: 86400)

#### **-bind=<addr>**

Binden Sie sich an die angegebene Adresse und hören Sie sich diese immer an. Verwenden Sie die Notation [host]:port für IPv6

#### **-Verbindung=<ip>**

Nur mit dem angegebenen Knoten verbinden; **-noconnect** deaktiviert automatische Verbindungen (die Regeln für dieses Paar sind die gleichen wie für **-addnode**). Diese Option kann mehrmals angegeben werden, um eine Verbindung zu mehreren Knoten herzustellen.

### **-entdecken**

Ermitteln Sie Ihre eigenen IP-Adressen (Standard: 1 beim Abhören und nicht **-externalip** oder **-proxy**)

### **-dns**

Erlauben Sie DNS-Suchen nach **-addnode**, **-seednode** und **-connect** (Standard: 1)

### **-dnsseed**

Paarweise Adressabfrage durch DNS-Lookup, falls nur wenige Adressen vorhanden sind (Standard: 1, es sei denn, es wird **-connection** verwendet)

### **-befähigenbip61**

Senden von Ablehnungsmeldungen durch BIP61 (Standard: 1)

### **-externalip=<ip>**

Geben Sie Ihre eigene öffentliche Adresse an

### **-erzwungennSaatgut**

Die Adressen von Kollegen immer über DNS-Lookup abfragen (Standard: 0)

### **-hören**

Verbindungen von außen akzeptieren (Standard: 1, wenn kein **-Proxy** oder **-Verbindung**)

### **-Listenonionion**

Automatisch den versteckten Dienst Tor erstellen (Standard: 1)

### **-maxconnections=<n>**

Höchstens <n> Verbindungen mit Kollegen pflegen (Standard: 125)

### **-maxreceivebuffer=<n>**

Maximaler Empfangspuffer pro Verbindung, <n>\*1000 Bytes (Standard: 5000)

**-maxsendbuffer=<n>**

Maximaler Sendepuffer pro Verbindung, <n>\*1000 Bytes (Standard: 1000)

**-maximale Zeiteinstellung**

Das Maximum, das für die Anpassung des durchschnittlichen Zeitausgleichs der Paare zulässig ist. Die lokale Zeitperspektive kann durch die Vorwärts- oder Rückwärtspaare um diesen Betrag beeinflusst werden. (Standard: 4200 Sekunden)

**-maxuploadtarget=<n>**

Versuchen Sie, den ausgehenden Verkehr unter dem vorgegebenen Ziel zu halten (in MiB für 24 Stunden), 0 = keine Beschränkung (Standard: 0)

**-onion=<>ip:port>**

Verwenden Sie einen separaten SOCKS5-Proxy, um Gleichgesinnte über die versteckten Dienste von Tor zu erreichen, setzen Sie **-noonion** auf disable (Standard: **-proxy**)

**-onlynet=<net>**

Stellen Sie ausgehende Verbindungen nur über das <net> Netzwerk her (ipv4, ipv6 oder onion). Eingehende Verbindungen sind von dieser Option nicht betroffen. Diese Option kann mehrmals angegeben werden, um mehrere Netzwerke zuzulassen.

**-Paarfilter**

Unterstützt das Blockieren von Filtern und Transaktionen mit blühenden Filtern (Standard: 1)

**-erlaubtbaremultisig**

Relais nicht P2SH-Multisig (Standard: 1)

**-port=<port>**

Hören Sie die Verbindungen in 'port' ab (Standard: 8333, testnet: 18333, regtest: 18444)

**-proxy=<ip:port>**

Verbindung über SOCKS5-Proxy herstellen, **-noproxy** auf off setzen (Standard: off)

**-proxyrandomisieren**

Randomisierung der Anmeldeinformationen für jede Proxy-Verbindung Dies ermöglicht die Isolierung des Tor-Flusses (Standard: 1)

**-seednode=<ip>**

Verbinden Sie sich mit einem Knoten, um die Paaradressen abzurufen, und trennen Sie die Verbindung. Diese Option kann mehrmals angegeben werden, um eine Verbindung zu mehreren Knoten herzustellen.

**-Zeitüberschreitung=<n>**

Geben Sie den Verbindungs-Timeout in Millisekunden an (Minimum: 1, Standard: 5000)

**-torcontrol=<ip>:<port>**

Tor-Kontrollport zur Verwendung, wenn Zwiebelhören aktiviert ist (Standard: 127.0.0.1:9051)

**-passwort=<pass>**

Tor-Kontrollport-Passwort (Standard: leer)

**-upnp**

Verwenden Sie UPnP, um den Abhörport zuzuweisen (Standard: 0)

**-whitebind=<addr>**

Link zu einer bestimmten Adresse und den auf der weißen Liste aufgeführten Partnern, die eine Verbindung zu dieser Adresse herstellen. Verwenden Sie die Notation [host]:port für IPv6

**-whitelist=<IP-Adresse oder Netzwerk>**

Die Whitelisted Pairs werden über die angegebene IP-Adresse (z.B. 1.2.3.4) oder das kommentierte Netz des CIDR (z.B. 1.2.3.0/24) verbunden. Sie kann mehrfach angegeben werden. Whitelisted Pairs können nicht vom DoS verboten werden

Portfolio-Optionen:

**-Adressentyp**

Welche Art von Adressen verwendet werden soll ("Legacy", "p2sh-segwit" oder "bech32", Standard: "p2sh-segwit")

#### **-vermeiden Sie teilweise Kosten**

Gruppieren Sie die Ausgaben nach Richtung, indem Sie alle oder keine auswählen, anstatt nach jeder Ausgabe zu selektieren. Der Datenschutz wird verbessert, da eine Adresse nur einmal verwendet wird (es sei denn, jemand verschickt sie, nachdem sie ausgegeben wurde), kann aber zu etwas höheren Raten führen, da die Auswahl der Währungen aufgrund der zusätzlichen Einschränkung suboptimal sein kann (Standard: 0)

#### **-Typ-Änderung**

Welcher Wechselkurs soll verwendet werden ("Legacy", "p2sh-segwit" oder "bech32"). Der Standard ist derselbe wie **-addresstype**, außer dass **-addresstype=p2sh-segwit** die systemeigene Segwit-Ausgabe verwendet, wenn an eine systemeigene Segwit-Adresse gesendet wird)

#### **-aus der Brieftasche**

Laden Sie die Brieftasche nicht und deaktivieren Sie RPC-Anrufe aus der Brieftasche

#### **-Entgelt=<amt>**

Der Satz des Tarifs (in BTC/kB), der die Toleranz angibt, die Änderung durch Hinzufügen zum Tarif zu verwerfen (Standard: 0,0001). Hinweis: Eine Ausgabe wird verworfen, wenn es sich bei dieser Rate um Staub handelt, aber wir werden immer bis zur Staub-Wiederübertragungsrate verwerfen, und eine darüber hinausgehende Verwerfungsrate wird durch die Ratenschätzung für das längere Ziel begrenzt.

#### **-fallbackfee=<amt>**

Ein Gebührensatz (in BTC/kB), der zu verwenden ist, wenn die Gebührentschatzung unzureichende Daten enthält (Standard: 0,0002)

#### **-Taste=<n>**

Setzen Sie die Größe des Schlüsselpools auf <n> (Standard: 1000)

#### **-mintxfee=<amt>**

Niedrigere Sätze (in BTC/kB) als dieser werden bei der Erstellung der Transaktion als Nullsatz betrachtet (Standard: 0,00001)

#### **-paytxfee=<amt>**

Kurs (in BTC/kB) zum Hinzufügen zu den von Ihnen gesendeten Transaktionen (Standard: 0,00)

### **-Suchen Sie**

Scannen Sie die Blockkette erneut, um am Anfang nach fehlenden Portfolio-Transaktionen zu suchen

### **-Rettungsgeldbörse**

Versuch, die privaten Schlüssel zu einer korrupten Brieftasche im Kofferraum zurückzuholen

### **-Verschwendungen des Informationsaustauschs**

Geben Sie das unbestätigte Wechselgeld beim Versenden der Transaktionen aus (Standard: 1)

### **-txconfirmtarget=<n>**

Wenn keine Zahlungsgebühr festgelegt wird, fügen Sie eine ausreichende Gebühr für Transaktionen hinzu, um die Bestätigung im Durchschnitt innerhalb von n Blöcken zu beginnen (Standard: 6)

### **-Portfolioverwaltung**

Aktualisieren Sie das Portfolio zu Beginn auf das neueste Format

### **-Brieftasche=<Pfad>**

Geben Sie den Pfad der Portfoliodatenbank an. Sie können ihn mehrmals angeben, um mehrere Portfolios zu laden. Der Pfad wird in Bezug auf <walletdir> interpretiert, wenn er nicht absolut ist, und wird erstellt, wenn er nicht existiert (z.B. ein Verzeichnis, das eine wallet.dat-Datei und Protokolldateien enthält). Aus Gründen der Abwärtskompatibilität akzeptiert es auch die Namen bestehender Datendateien in <walletdir>).

### **-Wallensendung**

Machen Sie die Portfolio-Diffusionstransaktionen (Standard: 1)

### **-walletdir=<dir>**

Geben Sie das Verzeichnis zum Speichern der Portfolios an (Standard: <datadir>/portfolios, falls vorhanden, sonst <datadir>)

**-walletnotify=<cmd>**

Befehl ausführen, wenn sich eine Portfolio-Transaktion ändert (%s in cmd wird durch TxID ersetzt)

**-walletrbf**

Senden Sie die Transaktionen mit der Option, die gesamte RBF zu aktivieren (nur RPC, Standard: 0)

**-zapwallettxes=<mode>**

Löschen Sie alle Transaktionen aus dem Portfolio und rufen Sie nur die Teile der Sperrkette durch **-rescan** am Anfang ab (1 = tx-Metadaten behalten, z.B. Zahlungsanforderungsinformationen, 2 = tx-Metadaten löschen)

ZeroMQ-Benachrichtigungsoptionen:

**-zmqpubhashblock=<Adresse>**

Aktivieren Sie die Veröffentlichungssperre in 'Adresse'

**-zmqpubhashblockhwm=<n>**

Veröffentlichungsblock für ausgehende Nachrichten mit einem hohen Wasserzeichen festlegen (Standard: 1000)

**-zmqpubhashtx=<Adresse>**

Aktivieren Sie die Veröffentlichung der Haschisch-Transaktion in 'Adresse'

**-zmqpubhashtxhwm=<n>**

Legen Sie die Veröffentlichung der Ausgabemeldung der Hash-Transaktion mit hohem Wasserzeichen fest (Standard: 1000)

**-zmqpubrawblock=<Adresse>**

Aktivieren Sie den rohen Veröffentlichungsblock in 'Adresse'

**-zmqpubrawblockhwm=<n>**

Raw Block Outgoing Message High Watermark setzen (Standard: 1000)

**-zmqpubrawtx=<Adresse>**

Aktivieren Sie die Veröffentlichung der Rohtransaktion in 'Adresse

**-zmqpubrawtxhwmt=<n>**

Setzen Sie das hohe Wasserzeichen für die Nachricht zur Ausgabe der Bruttotransaktion veröffentlichen (Standard: 1000)

Debugging-/Test-Optionen:

**-debug=<Kategorie>**

Debugging-Informationen ausgeben (Standard: **-nodebug**, die Angabe der 'Kategorie' ist optional). Wenn <category> nicht angegeben wird, oder wenn <category> = 1, werden alle Debug-Informationen ausgegeben. <category> können sein: net, tor, mempool, http, bench, zmq, db, rpc, estimatefee, addrman, selectcoins, reindex, cmpctblock, rand, prune, proxy, mempoolrej, libevent, coindb, qt, leveldb.

**-debugexclude=<Kategorie>**

Debugging-Informationen aus einer Kategorie ausschließen. Kann in Verbindung mit **-debug=1** verwendet werden, um Debug-Datensätze für alle Kategorien außer einer oder mehreren angegebenen Kategorien zu erzeugen.

**-Hilfe-Debug**

Drucken der Hilfemitteilung mit Debugging-Optionen und Beenden

**-Logbücher**

Die IP-Adressen in die Debug-Ausgabe einschließen (Standard: 0)

**-Logzeitstempel**

Bereiten Sie die Debugging-Ausgabe mit dem Zeitstempel vor (Standard: 1)

**-maxtxfee=<amt>**

Maximale Gesamtgebühren (in BTC) für die Verwendung bei einer einzelnen Portfolio-Transaktion oder bei einer Brutto-Transaktion; wenn sie zu niedrig angesetzt wird, kann sie große Transaktionen abbrechen (Standard: 0,10)

**-Drucken für die Konsole**

Trace-/Debugging-Informationen an die Konsole senden (Standard: 1, wenn kein **-daemon** vorhanden ist. Um die Protokollierung in eine Datei zu deaktivieren, setzen Sie **-nodebuglogfile**)

### **-shrinkdebugfile**

Reduzieren Sie die Datei debug.log beim Start des Clients (Standard: 1, wenn kein **-debug**)

### **-uacomment=<cmt>**

Einen Kommentar zur User-Agent-Zeichenfolge hinzufügen

Optionen für die Kettenauswahl:

### **-testnet**

Verwenden Sie die Testkette...

Optionen für die Weiterleitung von Knoten:

### **-bytespersigop**

Byte-Äquivalente pro Sigop bei Rundfunk- und Bergbautransaktionen (Standard: 20)

### **-Datenträger**

Relais- und Minen-Datenträger-Transaktionen (Standard: 1)

### **-datacarrierisieren**

Maximale Größe der Daten in den Transaktionen der Datenträger, die wir weiterleiten und extrahieren (Standard: 83)

### **-Mempool-Ersatz**

Aktivieren des Ersetzens von Transaktionen im Speicherpool (Standard: 1)

### **-minrelaytxfee=<amt>**

Gebühren (in BTC/kB), die darunter liegen, gelten als Nullgebühr für die Weiterleitung, Extraktion und Erstellung von Transaktionen (Standard: 0,00001)

### **-Whitelistforcerelay**

Erzwingen der Weiterleitung von Transaktionen der auf der Whitelist aufgeführten Partner, auch wenn die Transaktionen bereits im Mempool waren oder gegen die lokale Weiterleitungsrichtlinie verstößen (Standard: 0)

### **-Whitelistrelay**

Akzeptieren von übertragenen Transaktionen, die von Paaren der Whitelist empfangen wurden, auch wenn keine Transaktionen übertragen werden (Standard: 1)

Blockieren Sie die Erstellungsoptionen:

### **-blockmaxGewicht=<n>**

Legen Sie das maximale Gewicht des BIP141-Blocks fest (Standard: 3996000)

### **-blockmintxfee=<amt>**

Legen Sie den niedrigsten Provisionssatz (in BTC/kB) für Transaktionen fest, die bei der Blockerstellung berücksichtigt werden. (Standard: 0,00001)

RPC-Server-Optionen:

### **-Restaurant**

Öffentliche REST-Anfragen akzeptieren (Standard: 0)

### **-rpcallowip=<ip>**

Lassen Sie JSON-RPC-Verbindungen von der angegebenen Quelle zu. Sie sind gültig für <ip> eine einzelne IP (z. B. 1.2.3.4), ein Netzwerk/eine Netzwerkmaske (z. B. 1.2.3.4/255.255.255.0) oder ein Netzwerk/CIDR (z. B. 1.2.3.4/24). Diese Option kann mehrfach angegeben werden

### **-rpcauth=<benutzerpw>**

Benutzername und Kennwort HMAC-SHA-256 für JSON-RPC-Verbindungen. Das Feld 'userpw' hat folgendes Format: 'Benutzername': 'SALT': '\$ HASH'. Eine kanonische Pythonschrift ist in share/rpcauth enthalten. Der Client verbindet sich dann ganz normal mit dem Argumentpaar rpcuser=<USERNAME>/rpcpassword=<PASSWORD>. Diese Option kann mehrfach angegeben werden

### **-rpcbind=<addr>[:port]**

Link zu einer bestimmten Adresse, um JSON-RPC-Verbindungen abzu hören. Setzen Sie den RPC-Server nicht unzuverlässigen Netzwerken wie dem öffentlichen Internet aus! Diese

Option wird ignoriert, es sei denn, es wird auch **-rpccallowip** übergeben. Der Port ist optional und überschreibt **-rpcport**. Verwenden Sie die Notation [host]:port für IPv6. Diese Option kann mehrfach angegeben werden (Standard: 127.0.0.1 und ::1 d.h. localhost)

**-rpccookiefile=<loc>**

Speicherort des Autorisierungs-Cookies. Relativen Pfaden wird ein netzwerkspezifischer Datenadir-Speicherort vorangestellt. (Standard: Datenverzeichnis)

**-rpcpassword=<pw>**

Passwort für JSON-RPC-Verbindungen

**-rpcport=<port>**

Abhören von JSON-RPC-Verbindungen in 'port' (Standard: 8332, testnet: 18332, regtest: 18443)

**-rpcserialversion**

Setzt die rohe Transaktions-Serialisierung oder den Block Hex, der im nonverbalen Modus zurückgegeben wird, nicht segwit(0) oder segwit(1) (Standard: 1)

**-rpcthreads=<n>**

Legen Sie die Anzahl der Threads zur Behandlung von RPC-Aufrufen fest (Standard: 4)

**-rpcuser=<Benutzer>**

Benutzername für JSON-RPC-Verbindungen

**-Server**

Kommandozeilenbefehle und JSON-RPC akzeptieren

## 32. Lizenzierung und Nutzung von Software.

Android

<https://source.android.com/setup/start/licenses>

Termux

<https://github.com/termux/termux-app/blob/master/LICENSE.md>

Knotenpunkt

<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-Netzwerk

<https://github.com/torproject/tor/blob/master/LICENSE>

Syncthing Netzwerk

<https://forum.syncthing.net/t/syncthing-is-now-mpfv2-licensed/2133>

OpenSSH

<https://www.openssh.com/features.html>

Kitt SSH

<https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

MIT App Inventor 2 Companion und App Inventor Blockly

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

SQLite Expert Personal -freeware

<http://www.sqliteexpert.com/download.html>

Apachen-Ameise

<https://ant.apache.org/license.html>

WGET

<https://www.gnu.org/software/wget/>

OpenJDK

<https://openjdk.java.net/legal/>

Externe Erweiterungen:

JSONTOOLs

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

Die Lizenzierung von Open-Source- und kommerziellen Versionen des Mini BlocklyChain-Systems finden Sie auf der offiziellen Website <http://www.openqbit.com>. Mini BlocklyChain, MiniBlockly, BlocklyCode, MiniBlockMiniChain, QBlockly Sohn marcas registradas por OpenQbit.

Mini BlocklyChain ist öffentlich zugänglich.

Der gesamte Code und die Dokumentation in Mini BlocklyChain wurde von den Autoren der Public Domain gewidmet. Alle Code-Autoren und Vertreter der Unternehmen, für die sie arbeiten, haben eidesstattliche Erklärungen unterzeichnet, in denen sie ihre Beiträge der Öffentlichkeit zur Verfügung stellen, und die Originale dieser eidesstattlichen Erklärungen werden in einem Safe in der Hauptniederlassung von OpenQbit Mexiko aufbewahrt. Jedem steht es frei, die originalen Mini BlocklyChain (OpenQbit)-Erweiterungen zu veröffentlichen, zu verwenden oder zu verteilen, entweder als Quellcode oder als kompilierte Binärdateien, für jeden Zweck, ob kommerziell oder nicht-kommerziell, und mit allen Mitteln.

Der vorstehende Absatz bezieht sich auf den Code und die Dokumentation, die in Mini BlocklyChain geliefert werden, d. h. in den Teilen der Mini BlocklyChain-Bibliothek, die tatsächlich mit einer größeren Anwendung gruppiert und ausgeliefert werden. Einige Skripte, die als Teil des Kompilierungsprozesses verwendet werden (z.B. von autoconf generierte "Konfigurations"-Skripte), können in anderen Open-Source-Lizenzen enthalten sein. Keines dieser Kompilationsskripte schafft es jedoch in die endgültige Mini BlocklyChain-Bibliothek, so dass die mit diesen Skripten verbundenen Lizenzen bei der Bewertung Ihrer Rechte zum Kopieren und Verwenden der Mini BlocklyChain-Bibliothek keine Rolle spielen sollten.

Der gesamte lieferbare Code in Mini BlocklyChain wurde von Grund auf neu geschrieben. Es wurde kein Code von anderen Projekten oder aus dem offenen Internet übernommen. Jede Codezeile kann bis zu ihrem ursprünglichen Autor zurückverfolgt werden, und alle diese Autoren haben Public-Domain-Widmungen in den Akten. Daher ist die Codebasis von Mini BlocklyChain sauber und nicht durch Code verunreinigt, der von anderen Open-Source-Projekten lizenziert wurde, nicht durch offene Beiträge

Mini BlocklyChain ist Open Source, was bedeutet, dass Sie so viele Kopien machen können, wie Sie wollen, und mit diesen Kopien machen können, was Sie wollen, ohne Einschränkung. Aber Mini BlocklyChain ist nicht Open Source. Um Mini BlocklyChain in der öffentlichen Domäne zu halten und um sicherzustellen, dass der Code nicht durch proprietäre oder lizenzierte Inhalte verunreinigt wird, akzeptiert das Projekt keine Patches von unbekannten Personen. Der gesamte Code in Mini BlocklyChain ist ein Original, da er speziell für die Verwendung von Mini BlocklyChain geschrieben wurde. Es wurde kein Code aus unbekannten Quellen im Internet kopiert.

Mini BlocklyChain ist öffentlich zugänglich und erfordert keine Lizenz. Dennoch wollen einige Organisationen einen legalen Beweis für ihr Recht, Mini-BlocklyChain zu verwenden. Zu den Umständen, unter denen dies geschieht, gehören die folgenden:

- Ihr Unternehmen will eine Entschädigung für Ansprüche wegen Urheberrechtsverletzungen.
- Sie verwenden Mini BlocklyChain in einer Gerichtsbarkeit, die den öffentlichen Bereich nicht anerkennt.
- Sie verwenden Mini BlocklyChain in einer Gerichtsbarkeit, die das Recht eines Autors, sein Werk öffentlich zugänglich zu machen, nicht anerkennt.
- Sie möchten ein greifbares Rechtsdokument als Beweis dafür haben, dass Sie das Recht haben, Mini BlocklyChain zu verwenden und zu verteilen.
- Ihre Rechtsabteilung teilt Ihnen mit, dass Sie eine Lizenz kaufen müssen.

Wenn einer der oben genannten Umstände auf Sie zutrifft, wird OpenQbit, die Firma, die alle Mini BlocklyChain-Entwickler beschäftigt, Ihnen eine Mini BlocklyChain-Titelgarantie verkaufen. Eine Titelgarantie ist ein Rechtsdokument, das besagt, dass die beanspruchten Autoren von Mini BlocklyChain die wahren Autoren sind, und dass die Autoren das Recht haben, die Mini BlocklyChain der Public Domain zu widmen, und dass OpenQbit sich energisch gegen die Lizenzansprüche verteidigen wird. Alle Erlöse aus dem Verkauf der Titelgarantien von Mini BlocklyChain werden zur Finanzierung der kontinuierlichen Verbesserung und Unterstützung von Mini BlocklyChain verwendet.

#### Beisteuernder Code

Um Mini BlocklyChain völlig frei und gebührenfrei zu halten, akzeptiert das Projekt keine Patches. Wenn Sie einen Änderungsvorschlag machen und einen Patch als Proof-of-Concept beifügen möchten, wäre das großartig. Seien Sie jedoch nicht beleidigt, wenn wir Ihren Patch von Grund auf neu schreiben. Die Art der nicht-kommerziellen oder Open-Source-Lizenz, die es in dieser und ähnlichen Modalität ohne den Kauf von Support für Einzelpersonen oder Unternehmen unabhängig von der Größe des Unternehmens verwendet, wird durch die folgenden rechtlichen Prämissen geregelt.

Gewährleistungsausschluss. Sofern nicht durch geltendes Recht gefordert oder schriftlich vereinbart, stellt der Lizenzgeber den Schutzgegenstand (und jeder Beitragende stellt seine Beiträge zur Verfügung) "SO WIE SIE SIND", **OHNE GEWÄHRLEISTUNGEN ODER BEDINGUNGEN IRGENDERART, weder** ausdrücklich noch stillschweigend, einschließlich, aber nicht beschränkt auf Garantien oder Bedingungen des TITELS, der NICHTVERLETZUNG, der MARKTGÄNGIGKEIT ODER DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Sie sind allein verantwortlich für die Bestimmung der korrekten Verwendung oder Weitergabe des Schutzgegenstandes und für die Übernahme aller Risiken, die mit der Ausübung der Berechtigungen unter dieser Lizenz verbunden sind.

Alle finanziellen oder sonstigen Verluste, die durch die Verwendung dieser Software entstehen, werden von der betroffenen Partei getragen. Alle Rechtsstreitigkeiten werden von den Parteien nur den Gerichten in der Gerichtsbarkeit von Mexiko-Stadt, Land Mexiko, vorgelegt. Für kommerzielle Unterstützung, Nutzung und Lizenzierung muss eine Vereinbarung oder ein Vertrag zwischen OpenQbit oder seinem Unternehmen und dem Interessenten abgeschlossen werden.

Die Bedingungen des Vertriebsmarketings können sich ohne Vorankündigung ändern. Bitte gehen Sie auf die offizielle Website [www.openqbit.com](http://www.openqbit.com), um alle Änderungen der Support- und Lizenzbestimmungen für nicht-kommerzielle und kommerzielle Zwecke zu sehen.

Jede Person, Benutzer, private oder öffentliche Einrichtung jeglicher rechtlicher Art oder aus irgendeinem Teil der Welt, die die Software einfach nur benutzt, akzeptiert ohne Bedingungen die in diesem Dokument festgelegten Klauseln und diejenigen, die jederzeit im Portal von [www.openqbit.co](http://www.openqbit.co) ohne vorherige Ankündigung geändert werden können und nach dem Ermessen von OpenQbit in nicht-kommerzieller oder kommerzieller Nutzung angewendet werden können.

Alle Fragen und Informationen über Mini BlocklyChain sollten an die App Inventor-Gemeinschaft oder an die verschiedenen Blockly-System-Gemeinschaften gerichtet werden, so wie sie sind: AppBuilder, Trunkable, etc. und/oder an die Mail [opensource@openqbit.com](mailto:opensource@openqbit.com) für die Anforderung von Fragen kann die Antwort von 3 bis 5 Werktagen dauern.

Unterstützung bei der kommerziellen Nutzung.

[support@openqbit.com](mailto:support@openqbit.com)

Verkauf zur kommerziellen Nutzung.

[sales@openqbit.com](mailto:sales@openqbit.com)

Rechtliche Informationen und Lizenzfragen oder Bedenken

[legal@openqbit.com](mailto:legal@openqbit.com)

