Pflichtenheft Projekt Kyas Coin

Auftraggeber:

- Name: Andreas Villscheider
- Email Adresse: andreas.villscheider@schuele.suedtirol.it
- Telefonnummer: +39 334 804 2516
- Adresse: Unterdrittelgasse 16, Brixen

Auftragnehmer:

- Alexander Plaikner
  - Email Adresse: alexander02plaikner.ap@gmail.com
  - Telefonnummer: +39 340 127 8474
  - Adresse: Im Peuren 28, St.Sigmund, Kiens
- Matteo Planker
  - Email Adresse: plankermatteo23@gmail.com
  - Telefonnummer: +39 331 871 0241
  - Adresse: Daunei 11, Wolkenstein
- Peter Mantinger
  - Email Adresse: mantingerpeter2@gmail.com
  - Telefonnummer: +39 366 115 0413
  - Adresse: Gufidaun 122

Inhaltsverzeichnis

1 Anforderungen und Ziele **7**
1.1 Begriffserklärung . . . 7
1.1.1 Blockchain . . . 7
1.2 Überblick . . . 7
1.3 Details Muss-Anforderungen . . . 7
1.3.1 Transaktionen erstellen . . . 7
1.3.2 Mining . . . 8
1.3.3 Cross NetWork Unterstützung . . . 8
1.3.4 Coin Burning . . . 8
1.3.5 Masternode aussuchen . . . 8
1.3.6 User registrieren . . . 8
1.4 Zielgruppe und Anforderungen . . . 9

2 Use Cases **9**
2.1 Akteure . . . 9
2.1.1 User mit JavaFX Client . . . 9
2.1.2 Lokale Node . . . 9
2.1.3 Masternode . . . 9
2.2 Use Case Diagramm . . . 10
2.3 Use Case "Anfrage User registrieren" . . . 11
2.3.1 Kurze Beschreibung . . . 11
2.3.2 Auslöser . . . 11
2.3.3 Akteure . . . 11
2.3.4 Vorbedingungen . . . 11
2.3.5 Schritte . . . 11
2.3.6 Alternative Schritte . . . 11
2.3.7 Inkludierte Use Cases . . . 11
2.4 Use Case "Anfrage User Balance" . . . 12
2.4.1 Kurze Beschreibung . . . 12
2.4.2 Auslöser . . . 12
2.4.3 Akteure . . . 12
2.4.4 Vorbedingungen . . . 12
2.4.5 Schritte . . . 12
2.4.6 Alternative Schritte . . . 12
2.4.7 Inkludierte Use Cases . . . 12
2.5 Use Case "Anfrage User registrieren" . . . 13
2.5.1 Kurze Beschreibung . . . 13
2.5.2 Auslöser . . . 13
2.5.3 Akteure . . . 13
2.5.4 Vorbedingungen . . . 13
2.5.5 Schritte . . . 13
2.5.6 Alternative Schritte . . . 13
2.5.7 Nachbedingungen . . . 13
2.5.8 Inkludierte Use Cases . . . 13
2.6 Use Case "Anfrage PoW Berechnung" . . . 14
2.6.1 Kurze Beschreibung . . . 14

2.6.2 Auslöser . . . 14
2.6.3 Akteure . . . 14
2.6.4 Vorbedingungen . . . 14
2.6.5 Schritte . . . 14
2.6.6 Alternative Schritte . . . 14
2.6.7 Inkludierte Use Cases . . . 14
2.7 Use Case "Anfrage Transaktionen senden" . . . 15
2.7.1 Kurze Beschreibung . . . 15
2.7.2 Auslöser . . . 15
2.7.3 Akteure . . . 15
2.7.4 Vorbedingungen . . . 15
2.7.5 Schritte . . . 15
2.7.6 Alternative Schritte . . . 15
2.7.7 Inkludierte Use Cases . . . 15
2.8 Use Case "Anfrage Masternode setzen" . . . 16
2.8.1 Kurze Beschreibung . . . 16
2.8.2 Auslöser . . . 16
2.8.3 Akteure . . . 16
2.8.4 Vorbedingungen . . . 16
2.8.5 Schritte . . . 16
2.8.6 Alternative Schritte . . . 16
2.8.7 Inkludierte Use Cases . . . 16
2.9 Use Case "PoW Berechnung" . . . 17
2.9.1 Kurze Beschreibung . . . 17
2.9.2 Auslöser . . . 17
2.9.3 Akteure . . . 17
2.9.4 Vorbedingungen . . . 17
2.9.5 Schritte . . . 17
2.9.6 Alternative Schritte . . . 18
2.9.7 Inkludierte Use Cases . . . 18
2.10 Use Case "Transaktion erstellen" . . . 18
2.10.1 Kurze Beschreibung . . . 18
2.10.2 Auslöser . . . 18
2.10.3 Akteure . . . 18
2.10.4 Vorbedingungen . . . 18
2.10.5 Schritte . . . 19
2.10.6 Alternative Schritte . . . 19
2.10.7 Inkludierte Use Cases . . . 19
2.11 Use Case "Chain synchronisieren" . . . 20
2.11.1 Kurze Beschreibung . . . 20
2.11.2 Auslöser . . . 20
2.11.3 Akteure . . . 20
2.11.4 Vorbedingungen . . . 20
2.11.5 Schritte . . . 20
2.11.6 Alternative Schritte . . . 20
2.11.7 Inkludierte Use Cases . . . 20
2.12 User Case "Anfrage Chain" . . . 21
2.12.1 Kurze Beschreibung . . . 21

2.12.2 Auslöser . . . 21
2.12.3 Akteure . . . 21
2.12.4 Vorbedingungen . . . 21
2.12.5 Schritte . . . 21
2.12.6 Alternative Schritte . . . 21
2.12.7 Inkludierte Use Cases . . . 21
2.13 Use Case "Balance synchronisieren" . . . 22
2.13.1 Kurze Beschreibung . . . 22
2.13.2 Auslöser . . . 22
2.13.3 Akteure . . . 22
2.13.4 Vorbedingungen . . . 22
2.13.5 Schritte . . . 22
2.13.6 Alternative Schritte . . . 22
2.13.7 Inkludierte Use Cases . . . 23
2.14 Use Case "Anfrage Balance" . . . 23
2.14.1 Kurze Beschreibung . . . 23
2.14.2 Auslöser . . . 23
2.14.3 Akteure . . . 23
2.14.4 Vorbedingungen . . . 23
2.14.5 Schritte . . . 23
2.14.6 Alternative Schritte . . . 24
2.14.7 Inkludierte Use Cases . . . 24
2.15 Use Case "Anfrage User Balance" . . . 25
2.15.1 Kurze Beschreibung . . . 25
2.15.2 Auslöser . . . 25
2.15.3 Akteure . . . 25
2.15.4 Vorbedingungen . . . 25
2.15.5 Schritte . . . 25
2.15.6 Alternative Schritte . . . 25
2.15.7 Inkludierte Use Cases . . . 25
2.16 Use Case "Masternode setzen" . . . 26
2.16.1 Kurze Beschreibung . . . 26
2.16.2 Auslöser . . . 26
2.16.3 Akteure . . . 26
2.16.4 Vorbedingungen . . . 26
2.16.5 Schritte . . . 26
2.16.6 Alternative Schritte . . . 26
2.16.7 Inkludierte Use Cases . . . 26
2.17 Use Case "PoW Berechnung" . . . 27
2.17.1 Kurze Beschreibung . . . 27
2.17.2 Auslöser . . . 27
2.17.3 Akteure . . . 27
2.17.4 Vorbedingungen . . . 27
2.17.5 Schritte . . . 27
2.17.6 Alternative Schritte . . . 27
2.17.7 Inkludierte Use Cases . . . 27
2.18 Use Case "Transaktionen erstellen" . . . 28
2.18.1 Kurze Beschreibung . . . 28

2.18.2 Auslöser . . . 28
2.18.3 Akteure . . . 28
2.18.4 Vorbedingungen . . . 28
2.18.5 Schritte . . . 28
2.18.6 Alternative Schritte . . . 28
2.18.7 Inkludierte Use Cases . . . 28
2.19 Use Case "Blöcke erstellen" . . . 29
2.19.1 Kurze Beschreibung . . . 29
2.19.2 Auslöser . . . 29
2.19.3 Akteure . . . 29
2.19.4 Vorbedingungen . . . 29
2.19.5 Schritte . . . 29
2.19.6 Alternative Schritte . . . 29
2.19.7 Inkludierte Use Cases . . . 29
2.20 Use Case "Anfrage User Balance" . . . 30
2.20.1 Kurze Beschreibung . . . 30
2.20.2 Auslöser . . . 30
2.20.3 Akteure . . . 30
2.20.4 Vorbedingungen . . . 30
2.20.5 Schritte . . . 30
2.20.6 Alternative Schritte . . . 30
2.20.7 Inkludierte Use Cases . . . 30
2.21 Use Case "Anfrage User Transaktionen" . . . 31
2.21.1 Kurze Beschreibung . . . 31
2.21.2 Auslöser . . . 31
2.21.3 Akteure . . . 31
2.21.4 Vorbedingungen . . . 31
2.21.5 Schritte . . . 31
2.21.6 Alternative Schritte . . . 31
2.21.7 Inkludierte Use Cases . . . 31
2.22 Use Case "Anfrage User Transaktionen" . . . 32
2.22.1 Kurze Beschreibung . . . 32
2.22.2 Auslöser . . . 32
2.22.3 Akteure . . . 32
2.22.4 Vorbedingungen . . . 32
2.22.5 Schritte . . . 32
2.22.6 Alternative Schritte . . . 32
2.22.7 Inkludierte Use Cases . . . 32
2.23 Use Case "User registrieren" . . . 33
2.23.1 Kurze Beschreibung . . . 33
2.23.2 Auslöser . . . 33
2.23.3 Akteure . . . 33
2.23.4 Vorbedingungen . . . 33
2.23.5 Schritte . . . 33
2.23.6 Alternative Schritte . . . 33
2.23.7 Inkludierte Use Cases . . . 33

3 Klassendiagramme **34**

3.1 Klassendiagramm JavaFX Client 34
3.2 Klassendiagramm Blockchain lokale Node 35
3.3 Klassendiagramm Blockchain Masternode 36

4 Sequenzdiagramme **37**
4.1 Sequenzdiagramm Mining Operation 37
4.2 Sequenzdiagramm Transaktion erstellen 38
4.3 Sequenzdiagramm Masternode setzen 39
4.4 Sequenzdiagramm Balance synchronisieren 40
4.5 Sequenzdiagramm User registrieren 41
4.6 Sequenzdiagramm Block erstellen 42

5 Umsetzungsdetails **43**
5.1 Zielplattform 43
5.2 Datenhaltung 43
5.3 Oberfläche 43
5.4 Systemarchitektur 43
5.5 Hardwareanforderungen 44
5.5.1 Masternode 44
5.6 Node 44

6 Qualitätsmerkmale **45**
6.1 Benutzbarkeitsanforderungen 45
6.2 Effizienzanforderungen 45
6.3 Wartbarkeits- und Portierbarkeitsanforderung 45
6.4 Sicherheitsanforderung 45
6.5 Gesetzliche Anforderung 46

7 Projektplan **46**
7.1 Zeitplan 46
7.2 Kostenschätzung 46

1 Anforderungen und Ziele

1.1 Begriffserklärung

1.1.1 Blockchain

Eine Blockchain ist eine Kette aus Datensätzen, welche aus Transaktionen und verschiedenen anderen Komponenten (Hashes etc.) besteht. Da alle Blöcke auf dem vorherigen Block aufbauen, ist es unmöglich Blöcke im Nachhinein zu verändern, da sich so die ganze weitere Blockchain verändern würde. So ist es unmöglich eine vergangene Transaktion zu verändern, ohne alle weiteren Transaktionen zu zerstören. Eine manipulierte Blockchain würde außerdem durch die Inkonsistenz in den Berechnungen auffallen.

1.1.2 Block

Die Blockchain setzt sich aus Blöcken zusammen. Jeder Block enthält eine historische Datenbank aller Transaktionen, die durchgeführt wurden, bis der Block voll ist. Es handelt sich um eine permanente Aufzeichnung, wie eine Tasche mit Daten, die jederzeit geöffnet und angesehen werden kann.

1.1.3 Node/Knotenpunkt

Jeder Rechner, der im Netzwerk einer Blockchain hängt, wird als Node/Knotenpunkt bezeichnet.

1.1.4 Masternode

Spezielle Nodes welche Blöcke erstellen können. Manche ältere Blockchains unterscheiden nicht zwischen Node und Masternode.

1.1.5 Adresse

Jede Einheit einer Kryptowährung hat eine eindeutige Adresse, die angibt, wo sie auf der Blockchain sitzt. Es ist diese Adresse, dieser Ort, an dem die Eigentumsdaten der Münze gespeichert sind und an dem alle Änderungen registriert werden, wenn die Münze gehandelt wird. Diese Adressen unterscheiden sich je nach Krypto-Währung, sind aber in der Regel eine Zeichenkette von mehr als 30 Zeichen.

1.1.6 Balance

Die Anzahl der Coins welche eine bestimmte Adresse besitzt.

1.1.7 Wallet

Eine Wallet wird durch einen eindeutigen Code definiert, der ihre "Adresse" auf der Blockchain darstellt. Die Adresse der Wallet ist öffentlich, aber in ihr befindet sich eine Reihe privater Schlüssel, die den Besitz der Balance und die Balance selbst bestimmen. Sie kann in Software, Hardware, Papier oder anderen Formen vorliegen.

1.1.8 Mining

Der Begriff für den Prozess der Verifizierung von Transaktionen auf einer Blockchain. Bei der Lösung der Verschlüsselungsprobleme werden der Person, die die Computerleistung spendet, neue Bruchteile

der Kryptowährung zugeteilt.

1.1.9 Double-spending

Dies geschieht, wenn jemand versucht, eine Kryptowährung gleichzeitig an zwei verschiedene Wallets oder Orte zu senden.

1.1.10 Coin Burn

Wenn ein Coin in einer bestimmten Kryptowährung unbrauchbar gemacht wurde, wird sie als verbrannt bezeichnet.

1.1.11 Blockzeit/Blocktime

Die Zeit nachdem ein Block geschlossen, verifiziert und der Blockchain hinzugefügt wird.

1.2 Überblick

Es soll eine Kryptowährung mittels Blockchaintechnologie entwickelt werden. Sogenannte *Masternodes* sollen die wichtigsten Arbeiten erledigen wie z.B. neue Blöcke zu erstellen. Normale *Nodes*, die von den einzelnen Usern verwaltet werden sollen die *Proof of Work* für den nächsten Block berechnen und diese an die Masternode schicken können. Sie sollen auch neue Transaktionen erstellen und diese nach Bestätigung des Masternodes zu ihren unbestätigten Transaktionen hinzufügen können. Die Blockchain soll durch JavaFX und Android clients ansteuerbar sein. Außerdem soll jeder User auf einem Rechner einen lokalen Blockchain Algorithmus laufen haben, der wiederum die entfernte Masternode Blockchain ansteuert.

Folgende Funktionen müssen durch das System abgedeckt werden:

- Transaktionen an andere User senden
- Blöcke minen
- Operationen ausführen auch wenn man sich in einem anderen Netzwerk aufhält
- Coin Burning Algorithmus
- Masternode aussuchen
- User registrieren

Die Anforderungen werden nachfolgend detailliert.

1.3 Details Muss-Anforderungen

1.3.1 Transaktionen erstellen

Das System soll es Usern erlauben Transaktionen zu erstellen. User sollen auf mittels einer JavaFX bzw. Android Applikation Transaktionen erstellen können. Will der User nun eine Transaktion erstellen, so soll zuerst der derzeitige *Kontostand*, oder von nun an *Balance* genannt, abfragen. Ist die Balance kleiner als der Wert, den der User senden will, so soll das System die Transaktion abarbeiten. Dies soll folgendermaßen geschehen: Die lokale Node soll die Transaktion an die entfernten Masternode schicken, welcher die Transaktion vor Ort abarbeiten soll. Die Masternode soll die Transaktion

zum Mempool (Liste aus unbestätigten Transaktionen) hinzufügen. Dann soll die Masternode eine HTTP Bestätigung an die lokale Node schicken, welche dann die Transaktion auch zum lokale Mempool hinzufügen soll. Sollte die Balance nicht ausreichend sein, so soll die lokale Node eine HTTP Response mit einen Error Code an den JavaFX Client schicken.

1.3.2 Mining

Der JavaFX Client soll im Stande sein über die lokale Node Blöcke zu minen. Das heißt, die lokale Node soll eine HTTP Request vom JavaFX Client bekommen und daraufhin eine Abfolge von Befehlen ausführen. Zuerst soll die Node eine HTTP Request an die Masternode schicken und abfragen ob die nächste Proof of Work schon berechnet wurde. Falls die Masternode antwortet, dass sie schon berechnet wurde, soll die Node eine Antwort an den JavaFX Client schicken. Falls die Masternode antwortet, dass die nächste Proof of Work noch nicht berechnet wurde, soll die Node die lokale Blockchain mit der Blockchain der Masternode synchronisieren und dann anfangen die Proof of Work zu berechnen. Wurde die PoW berechnet, so soll sie zusammen mit dem Namen des Users an die Masternode mittels HTTP Request gesendet werden. Die Masternode soll die PoW auf ihre Validität überprüfen. Ist die PoW valide, so soll dem User ein gewisser Betrag überwiesen werden, ist sie nicht valide soll ein Error Message zurückgeschickt werden. Je nach empfangener Nachricht der Masternode soll der JavaFX Client eine andere Alert Nachricht anzeigen.

1.3.3 Cross-Network Unterstützung

Jeder User soll im Stande sein Transaktionen auszuführen oder Blöcke zu minen, egal in welchem Netzwerk sie sich befinden. Der User muss aber eine stabile Internetverbindung haben um mit den anderen Nodes kommunizieren zu können.

1.3.4 Coin Burning

Es soll ein Coin Burning Algorithmus im System laufen, welcher nach jeder Blocktime (Zeit, nach der ein Block erstellt wird, in diesem Fall 2 Minuten), sich 10% der registrierten Benutzer aussucht und je 10% ihrer Balance "verbrennt". Der maximale Wert, welcher pro Block insgesamt verbrennt werden kann, sollen 0.5% der Balance der Masternode sein. Der insgesamt verbrannte Wert soll auch von der Balance der Masternode verbrannt werden. Mit "verbrennen" ist gemeint, dass diese Beträge an eine Adresse geschickt werden sollen, wo sie dann für immer bleiben sollen und nicht mehr wegzuschicken gehen. Somit sind diese Coins dann vom offenen Markt genommen. Dadurch soll ein deflationärer Effekt erzeugt werden, wo die Anzahl der nutzbaren Coins sinkt, aber der Wert eines einzelnen Coins steigt.

1.3.5 Masternode aussuchen

Jeder User soll in seinem JavaFX Client im Stande sein die Masternode auszusuchen, mit welcher die lokale Node kommunizieren soll. Die lokale Node soll dann eine HTTP Request vom JavaFX Client empfangen und die Masternode setzen. Die Android Applikation hingegen, verwendet nur den Hauptledger als Peer.

1.3.6 User registrieren

Meldet sich ein User im JavaFX bzw. Android Client an, so soll eine HTTP Request an die lokale Node gesendet werden, wo der Name der Users enthalten ist. Dieser Namen soll dann an die Mas-

ternode weitergeleitet werden, welche nachschauen soll, ob der User neu ist. Ist er neu, soll die Masternode einen kleinen Betrag an den neuen User überweisen, als Willkommensbonus. Ist der User nicht neu, soll nichts passieren, außer, dass die lokale Node den Benutzernamen setzt. Der Benutzername soll dann hergenommen werden, wenn es Mining Operationen zu erledigen gibt, ober wenn eine neue Transaktion erstellt wird.

1.4 Zielgruppe und Anforderungen

Potentielle Benutzer dieses Systems, oder besser, dieser neuen finanziellen Umgebung, sind Personen, welche Transaktionen schnell abwickeln möchten und sich außerdem ein wenig mit der Bedienung von Computerprogrammen auskennen. Der User muss im Grunde nur ein Python Script ausführen und eine JAR öffnen / eine App installieren, alles Weitere verläuft über ein GUI.

2 Use Cases

2.1 Akteure

2.1.1 User mit JavaFX Client

Der User mit dem JavaFX Client kann im Grunde genommen nur Anfragen an den zweiten Akteur im Use Case Diagramm schicken, der lokalen Node. Der Client fungiert als Benutzeroberfläche für den User.

2.1.2 User mit Android Client

Die Android App fungiert rein als Client und besitzt keine Kopie der Blockchain. Die App kann nur an die Masternode Anfragen schicken und die davon gelieferten Antworten visuell repräsentieren.

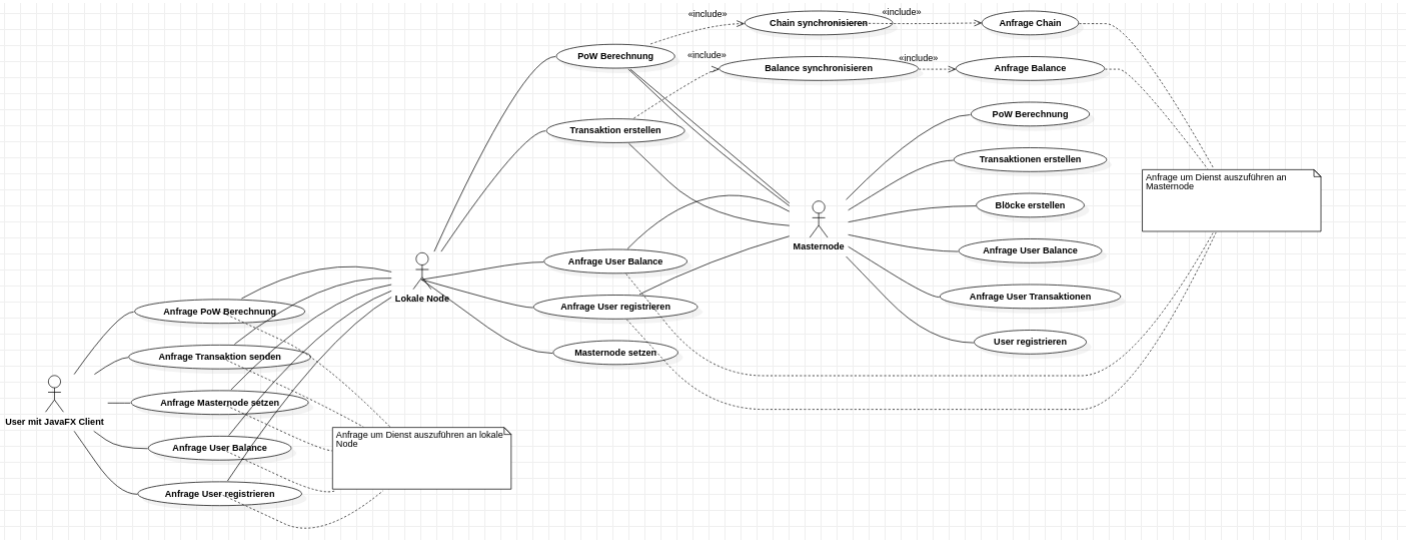
2.1.3 Lokale Node

Die lokale Node gilt als Weiterleiter von gewissen Anfragen des JavaFX Clients an die Masternode. Die lokale Node kann jedoch auch selbst Aufgaben ausführen wie z.B. die Berechnung der nächsten PoW.

2.1.4 Masternode

Die Masternode ist das Herzstück dieses Systems. Sie erstellt Blöcke und kann alles, was eine Masternode auch kann. Durch die Masternode ist es möglich die Balances und Transaktionen zu synchronisieren.

2.2 Use Case Diagramm



2.3 Use Case "Anfrage User registrieren"

2.3.1 Kurze Beschreibung

Der User schickt eine Anfrage um den User zu registrieren.

2.3.2 Auslöser

Der User meldet sich an und klickt im Client auf Synchronisieren.

2.3.3 Akteure

User mit JavaFX/Android Client, Lokale Node

2.3.4 Vorbedingungen

Lokale Node bereit. JavaFX/Android Client bereit. Masternode bereit.

2.3.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize/Refresh

2.3.6 Alternative Schritte

1. User ändert die Addresse im "Settings" Menü in der Android Applikation.
2. Keine alternativen Schritte
3. (a) User klickt auf Send (nur JavaFx Client)
   (b) User klickt auf Mining (nur JavaFx Client)

2.3.7 Inkludierte Use Cases

-

2.4 Use Case "Anfrage User Balance"

2.4.1 Kurze Beschreibung

Der User schickt eine Anfrage um seine Balance zu erhalten.

2.4.2 Auslöser

Der User meldet sich an und klickt im Client auf Synchronisieren.

2.4.3 Akteure

User mit JavaFX/Android Client, Lokale Node

2.4.4 Vorbedingungen

Lokale Node bereit. JavaFX/Android Client bereit. Masternode bereit.

2.4.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus. (nur JavaFx Client)
3. User klickt auf Synchronize/Refresh

2.4.6 Alternative Schritte

1. User ändert die Addresse im "Settings" Menü in der Android Applikation.
2. Keine alternativen Schritte
3. (a) User klickt auf Send (nur JavaFx Client)
   (b) User klickt auf Mining (nur JavaFx Client)

2.4.7 Inkludierte Use Cases

-

2.5 Use Case "Anfrage User registrieren"

2.5.1 Kurze Beschreibung

Der User schickt eine Anfrage um den User zu registrieren.

2.5.2 Auslöser

Der User meldet sich an und klickt im Client auf Synchronisieren.

2.5.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.5.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.5.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. Lokale Node sendet der Masternode den User zur Registrierung

2.5.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining

2.5.7 Nachbedingungen

-

2.5.8 Inkludierte Use Cases

-

2.6 Use Case "Anfrage PoW Berechnung"

2.6.1 Kurze Beschreibung

Der User schickt eine Anfrage die nächste PoW zu berechnen.

2.6.2 Auslöser

User klickt im Tab Mining auf Mine!

2.6.3 Akteure

User mit JavaFX Client, Lokale Node

2.6.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.6.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Mining
5. User klickt auf Mine!

2.6.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte

2.6.7 Inkludierte Use Cases

-

2.7 Use Case "Anfrage Transaktionen senden"

2.7.1 Kurze Beschreibung

Der User schickt eine Anfrage um eine Transaktion zu senden.

2.7.2 Auslöser

Der User meldet sich an und versucht eine Transaktion abzuschicken.

2.7.3 Akteure

User mit JavaFX/Android Client, Lokale Node

2.7.4 Vorbedingungen

Lokale Node bereit. JavaFX/Android Client bereit. Masternode bereit.

2.7.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus. (nur JavaFx)
3. User klickt auf Synchronize/Refresh
4. User klickt im Seitenpaneel auf Send/Plus-Zeichen
5. User gibt Zieladresse an
6. User gibt Betrag an
7. User klickt auf Send

2.7.6 Alternative Schritte

1. User gibt seine Addresse unter "Settings" ein (Android)
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte

2.7.7 Inkludierte Use Cases

-

2.8 Use Case "Anfrage Masternode setzen"

2.8.1 Kurze Beschreibung

Der User schickt eine Anfrage um die Masternode zu setzen.

2.8.2 Auslöser

Der User meldet sich an und wählt eine Masternode aus.

2.8.3 Akteure

User mit JavaFX Client, Lokale Node

2.8.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.8.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize

2.8.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining

2.8.7 Inkludierte Use Cases

-

2.9 Use Case "PoW Berechnung"

2.9.1 Kurze Beschreibung

Die lokale Node berechnet die nächste PoW.

2.9.2 Auslöser

Der User sendet über den JavaFX Client eine Anfrage um die nächste PoW zu berechnen.

2.9.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.9.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.9.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Mining
5. User klickt auf Mine!
6. Lokale Node synchronisiert Blockchain
7. Lokale Node checkt bei Masternode ob PoW schon berechnet wurde
8. Lokale Node berechnet PoW und schickt PoW an Masternode

2.9.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte
8. (a) Wurde die PoW schon berechnet, wird ein Error an den JavaFX Client zurückgegeben

2.9.7 Inkludierte Use Cases

Chain synchronisieren, Anfrage Chain

2.10 Use Case "Transaktion erstellen"

2.10.1 Kurze Beschreibung

Die lokale Node erstellt eine neue Transaktion.

2.10.2 Auslöser

Der User sendet über den JavaFX/Android Client eine Anfrage um eine neue Transaktion zu erstellen.

2.10.3 Akteure

User mit JavaFX/Android Client, Lokale Node, Masternode

2.10.4 Vorbedingungen

Lokale Node bereit. JavaFX/Android Client bereit. Masternode bereit.

2.10.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus. (nur JavaFx)
3. User klickt auf Synchronize/Refresh
4. User klickt im Seitenpaneel auf Send/Plus-Zeichen
5. User gibt Zieladresse an
6. User gibt Betrag an
7. User klickt auf Send
8. Lokale Node checkt bei der Masternode ob die Balance des Users ausreichend ist
9. Lokale Node sendet neue Transaktion an Masternode
10. Lokale Node erstellt selbst neue Transaktion

2.10.6 Alternative Schritte

1. User gibt Adresse unter "Settings" ein (Android)
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte
8. Keine alternativen Schritte
9. (a) Ist die Balance ausreichen wird eine Error Message an den JavaFX Client geschickt
10. (a) Ist die Balance nicht ausreichen wird eine Error Message an den JavaFX Client geschickt und die Transaktion nicht erstellt
    (b) Schickt die Masternode eine Error Message zurück wird eine Error Message an den Client gesendet und die Transaktion nicht erstellt

2.10.7 Inkludierte Use Cases

Balance synchronisieren, Anfrage Balance

2.11 Use Case "Chain synchronisieren"

2.11.1 Kurze Beschreibung

Die lokale Node synchronisiert die Blockchain mit der Masternode.

2.11.2 Auslöser

Der User will einen neuen Block minen.

2.11.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.11.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.11.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Mining
5. User klickt auf Mine!
6. Lokale Node synchronisiert die lokale Blockchain mit der der Masternode

2.11.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte

2.11.7 Inkludierte Use Cases

Anfrage Chain

2.12 User Case "Anfrage Chain"

2.12.1 Kurze Beschreibung

Die lokale Node fragt bei der Masternode um eine Kopie der Blockchain an.

2.12.2 Auslöser

Der User will einen neuen Block minen.

2.12.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.12.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.12.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Mining
5. User klickt auf Mine!
6. Lokale Node synchronisiert die lokale Blockchain
7. Lokale Node fragt bei der Masternode um eine Kopie der Blockchain an

2.12.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte

2.12.7 Inkludierte Use Cases

-

2.13 Use Case "Balance synchronisieren"

2.13.1 Kurze Beschreibung

Die lokale Node synchronisiert die Balance.

2.13.2 Auslöser

Der User will eine neue Transaktion erstellen.

2.13.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.13.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.13.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Send
5. User gibt Zieladresse an
6. User gibt Betrag an
7. User klickt auf Send
8. Lokale Node synchronisiert Balance

2.13.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte
8. Keine alternativen Schritte

2.13.7 Inkludierte Use Cases

Anfrage Balance

2.14 Use Case "Anfrage Balance"

2.14.1 Kurze Beschreibung

Die lokale Node fragt bei der Masternode um die Balance des Users an.

2.14.2 Auslöser

Der User will eine neue Transaktion erstellen.

2.14.3 Akteure

User mit JavaFX Client, Lokale Node, Masternode

2.14.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit. Masternode bereit.

2.14.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. User klickt im Seitenpaneel auf Send
5. User gibt Zieladresse an
6. User gibt Betrag an
7. User klickt auf Send
8. Lokale Node synchronisiert Balance
9. Lokale Node sendet eine Request an die Masternode wo die Balance des derzeitigen Users abgefragt wird

2.14.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. (a) User klickt auf Send
   (b) User klickt auf Mining
4. Keine alternativen Schritte
5. Keine alternativen Schritte
6. Keine alternativen Schritte
7. Keine alternativen Schritte
8. Keine alternativen Schritte
9. Keine alternativen Schritte

2.14.7 Inkludierte Use Cases

-

2.15 Use Case "Anfrage User Balance"

2.15.1 Kurze Beschreibung

Die lokale Node schickt der Masternode eine Anfrage um die Balance eines Users zu erhalten.

2.15.2 Auslöser

Der User will seine Balance synchronisieren.

2.15.3 Akteure

User mit JavaFX/Android Client, Lokale Node, Masternode

2.15.4 Vorbedingungen

Lokale Node bereit. JavaFX/Android Client bereit. Masternode bereit.

2.15.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus. (nur JavaFx)
3. User klickt auf Synchronize, jedes späteres Drücken auf Synchronize wird diesem Anwendungsfall entsprechen
4. Lokale Node sendet HTTP Request an Masternode um die Balance des Users zu erhalten

2.15.6 Alternative Schritte

1. User gibt seine Addresse unter "Settings" ein (Android)
2. Keine alternativen Schritte
3. Keine alternativen Schritte
4. Keine alternativen Schritte

2.15.7 Inkludierte Use Cases

-

2.16 Use Case "Masternode setzen"

2.16.1 Kurze Beschreibung

Die lokale Node setzt die Adresse der Masternode, zu der sie sich verbinden will.

2.16.2 Auslöser

Der User synchronisiert zum ersten Mal.

2.16.3 Akteure

User mit JavaFX Client, Lokale Node

2.16.4 Vorbedingungen

Lokale Node bereit. JavaFX Client bereit.

2.16.5 Schritte

1. User meldet sich im JavaFX Client an.
2. User sucht eine Masternode aus dem Einstellungen Screen aus.
3. User klickt auf Synchronize
4. JavaFX Client übergibt Namen des Masternodes an die lokale Chain mittels HTTP Request
5. Lokale Node speichert die Masternode ein

2.16.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte
4. Keine alternativen Schritte
5. Keine alternativen Schritte

2.16.7 Inkludierte Use Cases

-

2.17 Use Case "PoW Berechnung"

2.17.1 Kurze Beschreibung

Masternode berechnet die nächste Proof of Work.

2.17.2 Auslöser

Blocktime ist um und keine Node hat nächste PoW berechnet.

2.17.3 Akteure

Masternode

2.17.4 Vorbedingungen

Masternode bereit. PoW wurde noch nicht berechnet.

2.17.5 Schritte

1. Blocktime ist um
2. PoW berechnen

2.17.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte

2.17.7 Inkludierte Use Cases

-

2.18 Use Case "Transaktionen erstellen"

2.18.1 Kurze Beschreibung

Masternode erstellt eine neue Transaktion.

2.18.2 Auslöser

Masternode erhält einen API Call eine neue Transaktion zu erstellen.

2.18.3 Akteure

Masternode

2.18.4 Vorbedingungen

Masternode bereit.

2.18.5 Schritte

1. Masternode erhält einen API Call eine neue Transaktion zu erstellen
2. Masternode erstellt eine neue Transaktion
3. Masternode fügt die neue Transaktion dem Mempool der Masternode hinzu

2.18.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte

2.18.7 Inkludierte Use Cases

-

2.19 Use Case "Blöcke erstellen"

2.19.1 Kurze Beschreibung

Masternode erstellt einen neuen Block.

2.19.2 Auslöser

Blocktime ist um.

2.19.3 Akteure

Masternode

2.19.4 Vorbedingungen

Masternode bereit.

2.19.5 Schritte

1. Blocktime um
2. Masternode führt Burning Algorithmus aus
3. Masternode erstellt einen neuen Block
4. Masternode fügt neuen Block zur Blockchain hinzu

2.19.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte
4. Keine alternativen Schritte

2.19.7 Inkludierte Use Cases

-

2.20 Use Case "Anfrage User Balance"

2.20.1 Kurze Beschreibung

Masternode gibt Balance eines Users zurück.

2.20.2 Auslöser

Masternode erhält einen API Call die Balance eines Users zurückzugeben.

2.20.3 Akteure

Masternode

2.20.4 Vorbedingungen

Masternode bereit.

2.20.5 Schritte

1. API Call geht ein
2. Masternode iteriert durch alle Blöcke um Balance zu berechnen
3. Masternode gibt Balance zurück

2.20.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte

2.20.7 Inkludierte Use Cases

-

2.21 Use Case "Anfrage User Transaktionen"

2.21.1 Kurze Beschreibung

Masternode gibt Transaktionen eines Users zurück.

2.21.2 Auslöser

Masternode erhält einen API Call die Transaktionen eines Users zurückzugeben.

2.21.3 Akteure

Masternode

2.21.4 Vorbedingungen

Masternode bereit.

2.21.5 Schritte

1. API Call geht ein
2. Masternode iteriert durch alle Blöcke und speichert alle Transaktionen eines Users in eine Liste
3. Masternode gibt Liste zurück

2.21.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte

2.21.7 Inkludierte Use Cases

-

2.22 Use Case "Anfrage User Transaktionen"

2.22.1 Kurze Beschreibung

Masternode gibt Transaktionen eines Users zurück.

2.22.2 Auslöser

Masternode erhält einen API Call die Transaktionen eines Users zurückzugeben.

2.22.3 Akteure

Masternode

2.22.4 Vorbedingungen

Masternode bereit.

2.22.5 Schritte

1. API Call geht ein
2. Masternode iteriert durch alle Blöcke und speichert alle Transaktionen eines Users in eine Liste
3. Masternode gibt Liste zurück

2.22.6 Alternative Schritte

1. Keine alternativen Schritte
2. Keine alternativen Schritte
3. Keine alternativen Schritte

2.22.7 Inkludierte Use Cases

-

2.23 Use Case "User registrieren"

2.23.1 Kurze Beschreibung

Masternode registriert einen neuen User.

2.23.2 Auslöser

Masternode erhält einen API Call um einen neuen User zu erstellen.

2.23.3 Akteure

Masternode

2.23.4 Vorbedingungen

Masternode bereit.

2.23.5 Schritte

1. API Call geht ein
2. Masternode schaut nach ob es den User schon in der Liste der registrierten User gibt, falls ja wird er nicht neu registriert, falls nein wird er registriert und ein Signup Bonus wird dem User gutgeschrieben.

2.23.6 Alternative Schritte

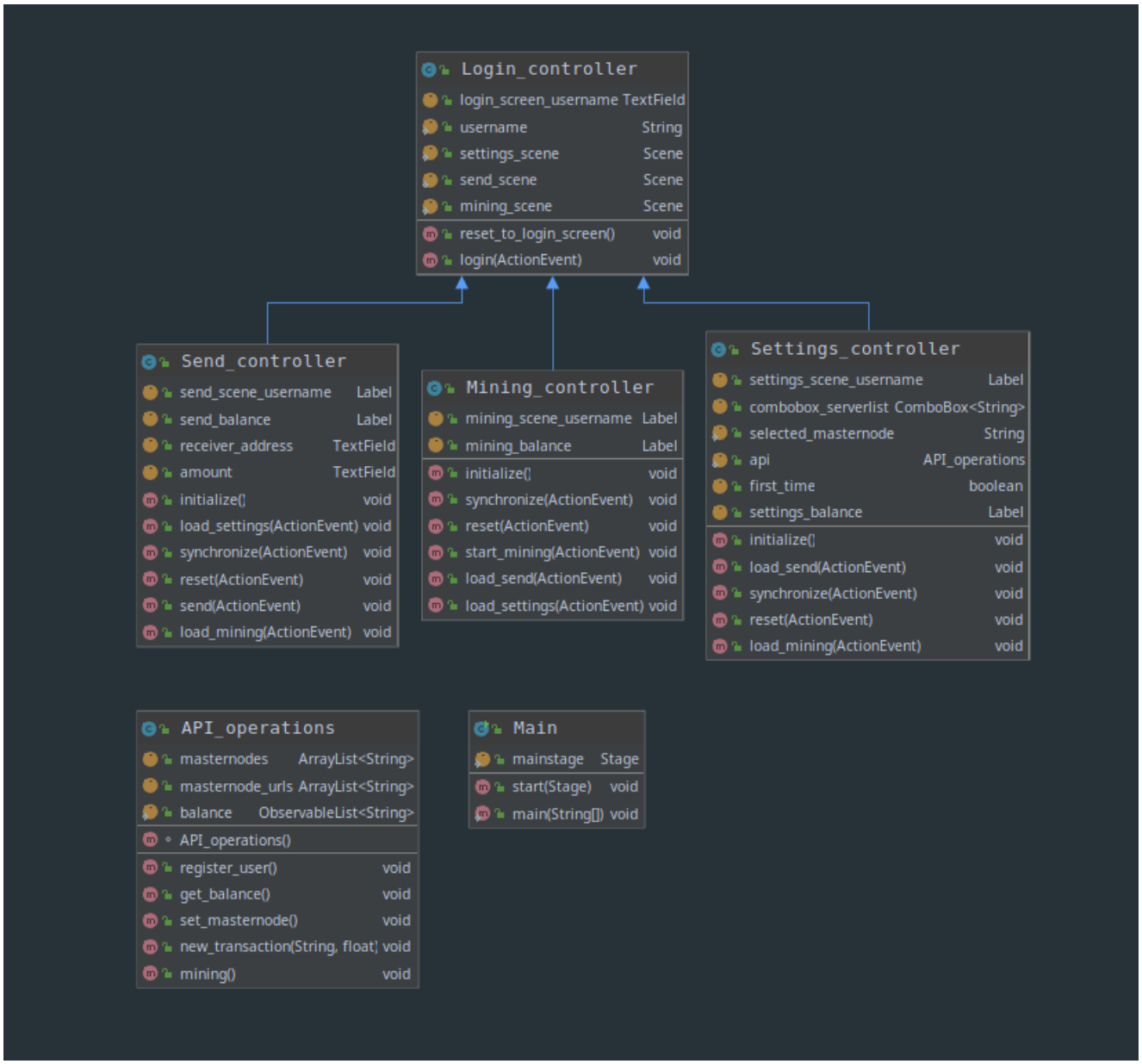
1. Keine alternativen Schritte
2. Keine alternativen Schritte

2.23.7 Inkludierte Use Cases

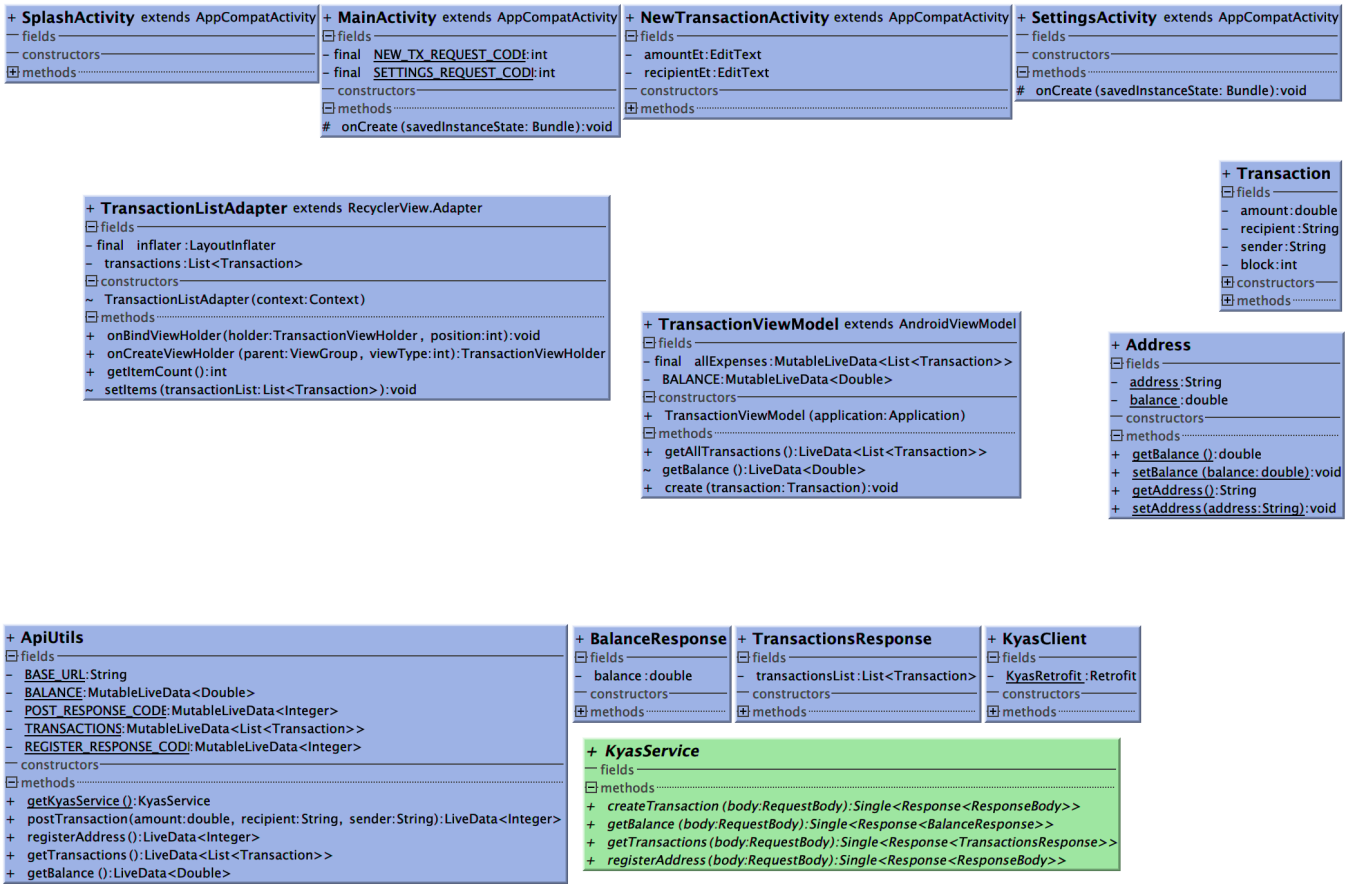
-

3 Klassendiagramme

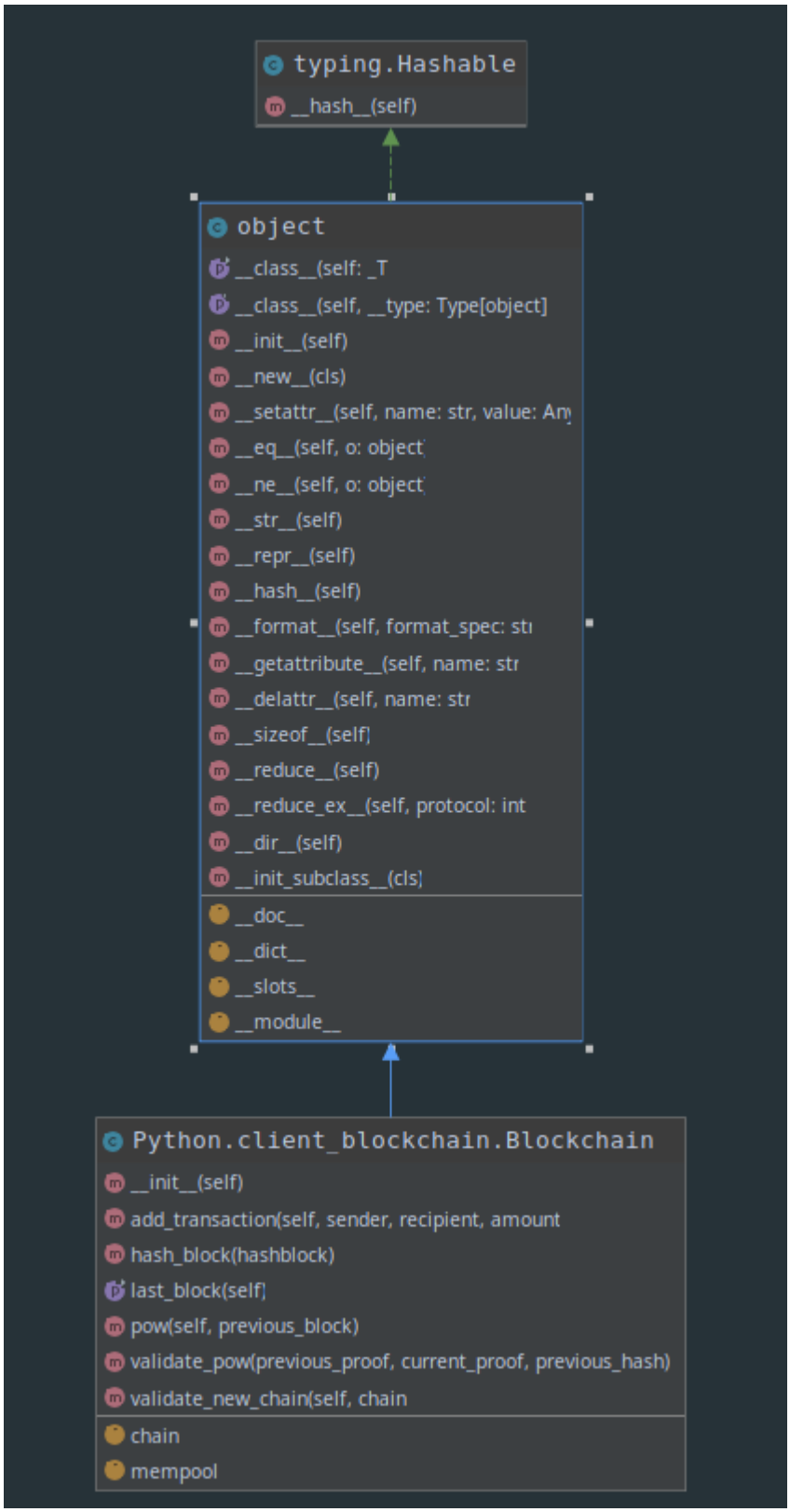
3.1 Klassendiagramm JavaFX Client



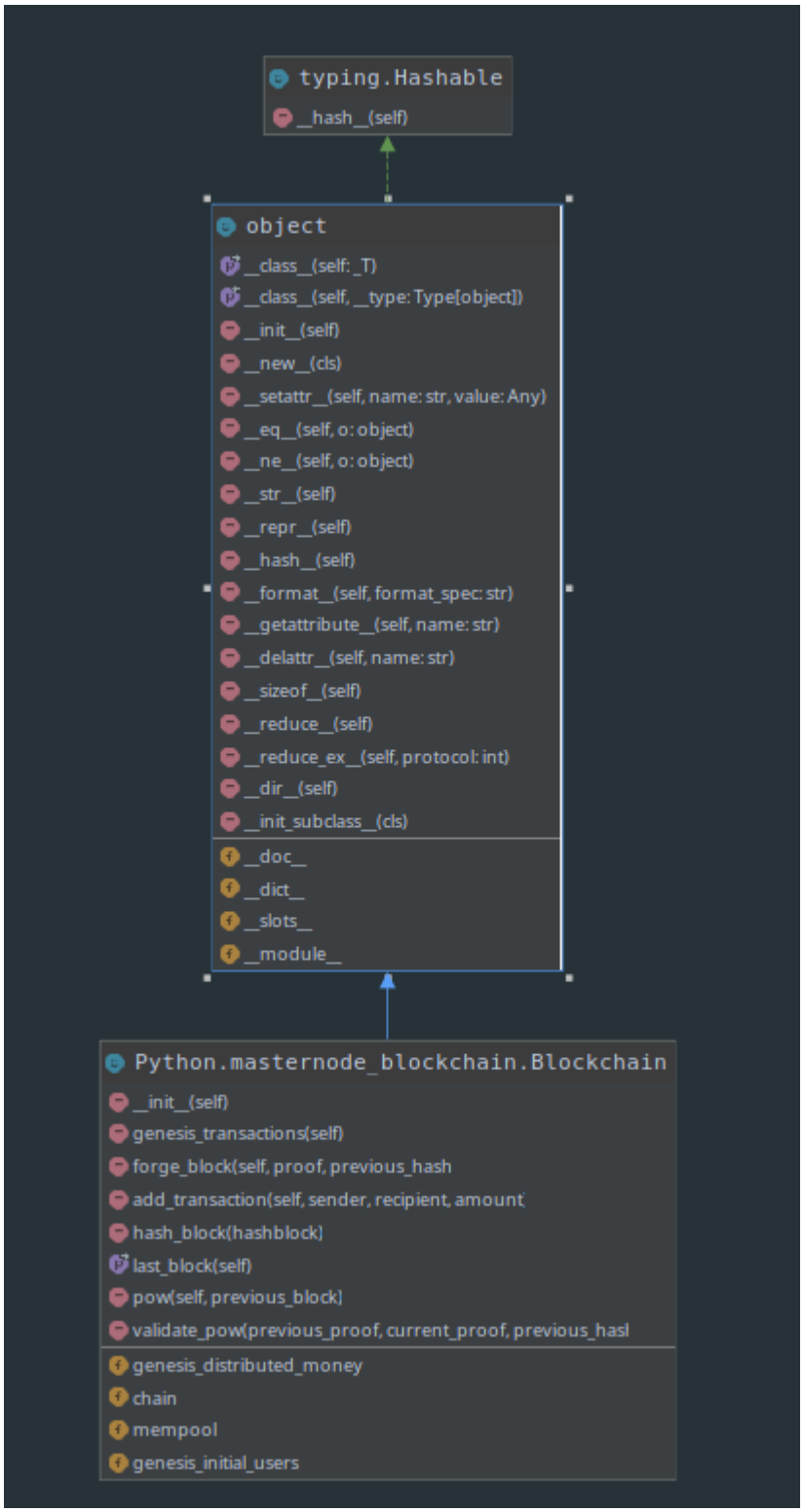
3.2 Klassendiagramm Android Client



3.3 Klassendiagramm Blockchain lokale Node

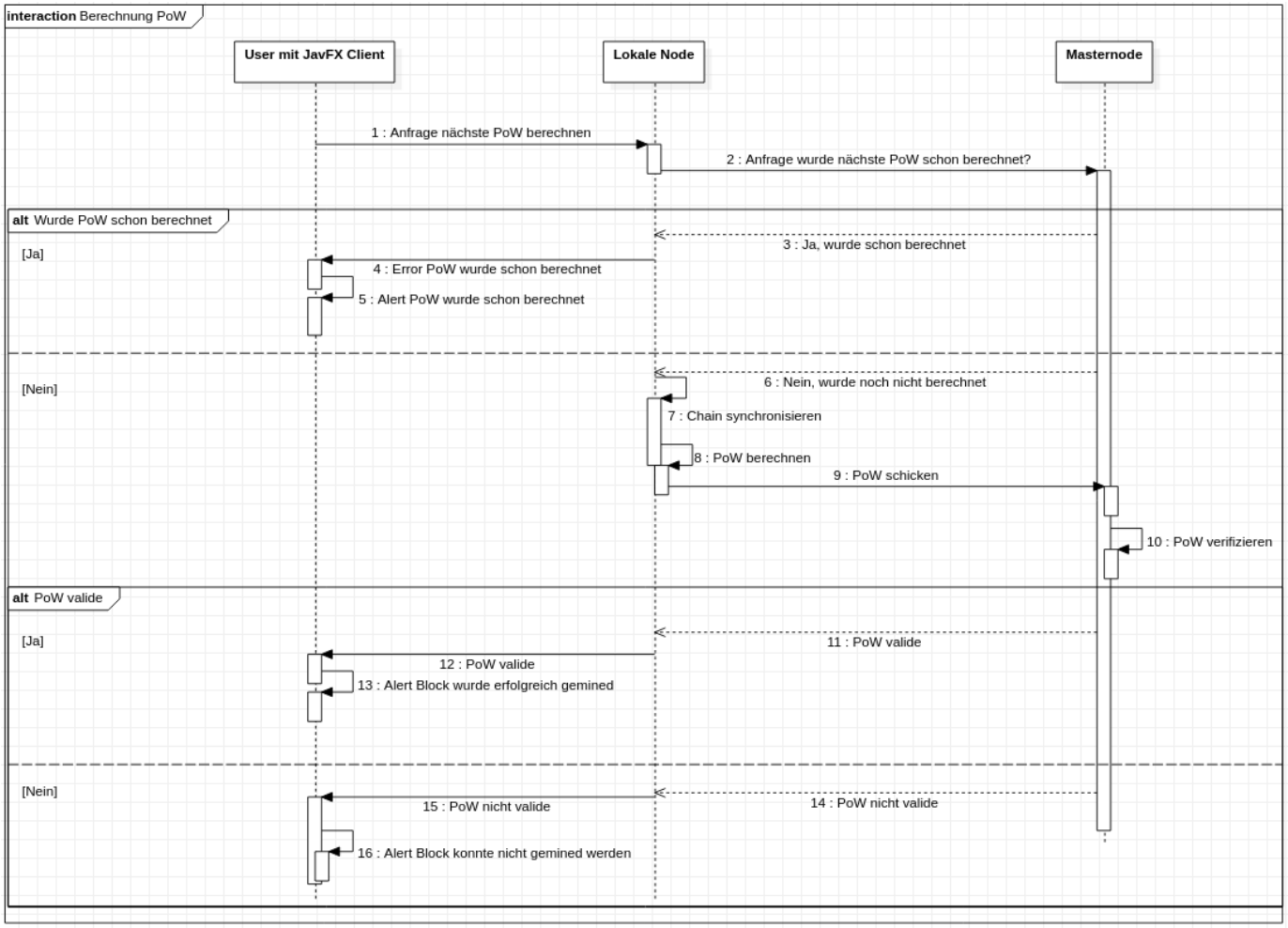


3.4 Klassendiagramm Blockchain Masternode

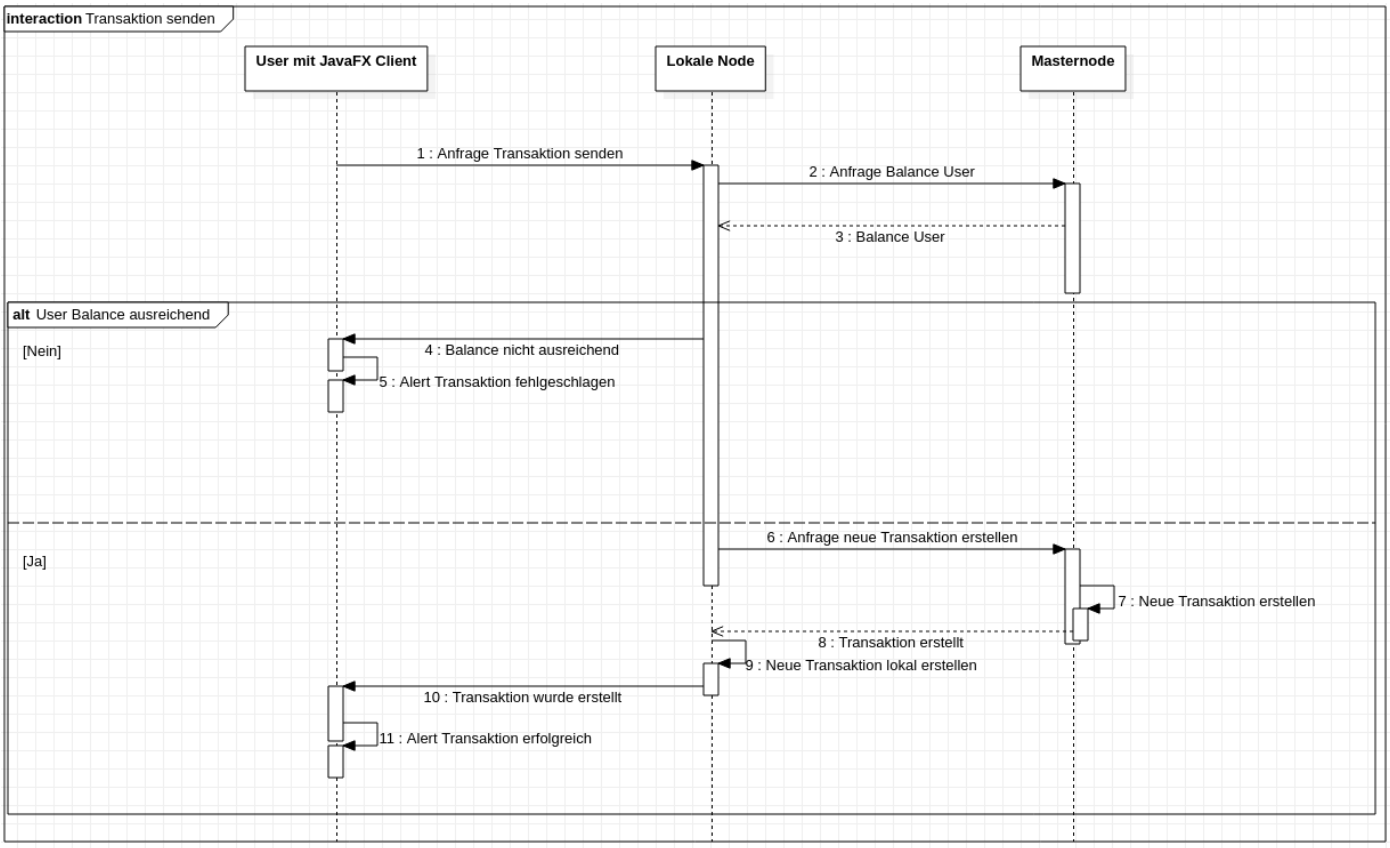


4 Sequenzdiagramme

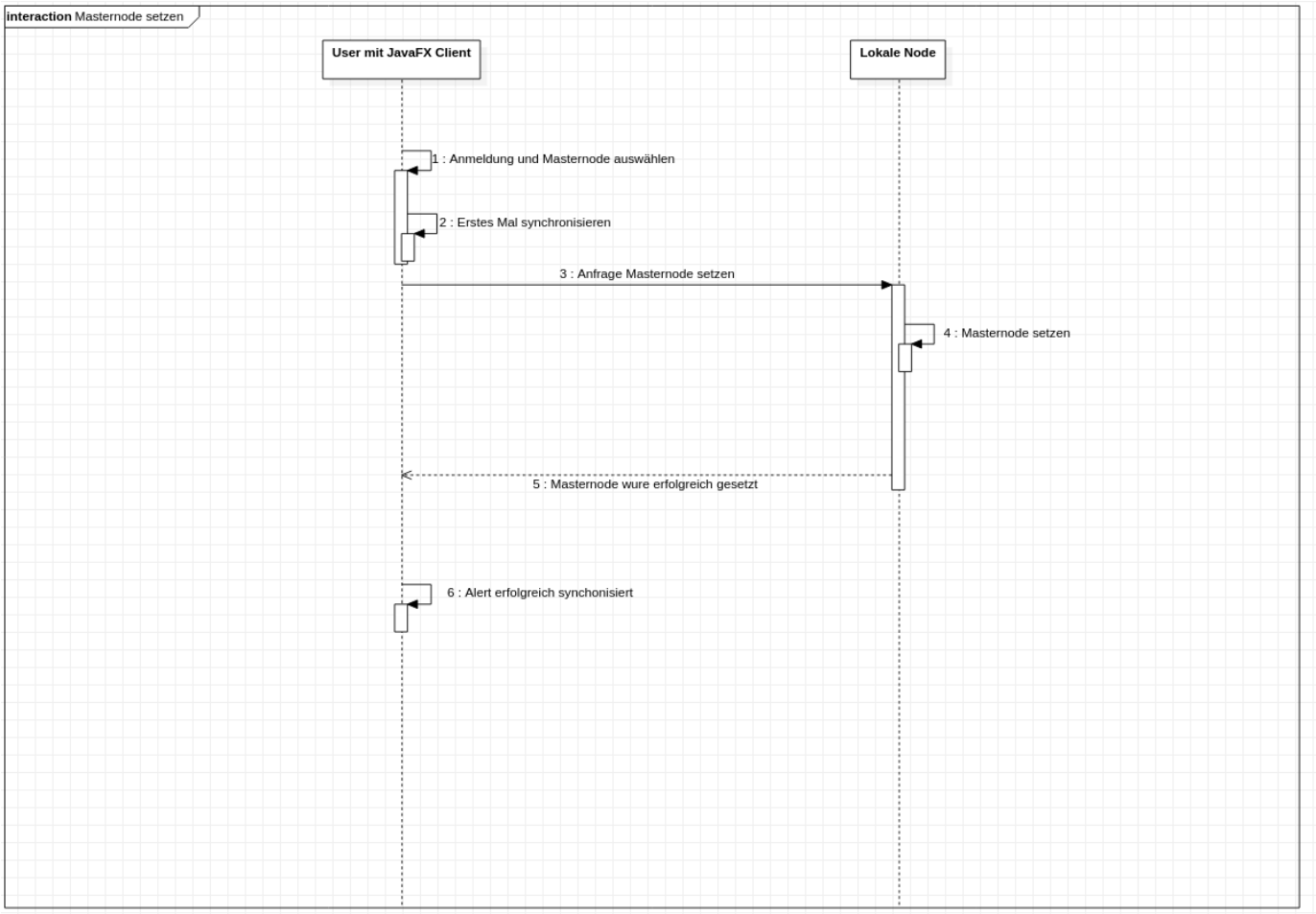
4.1 Sequenzdiagramm Mining Operation



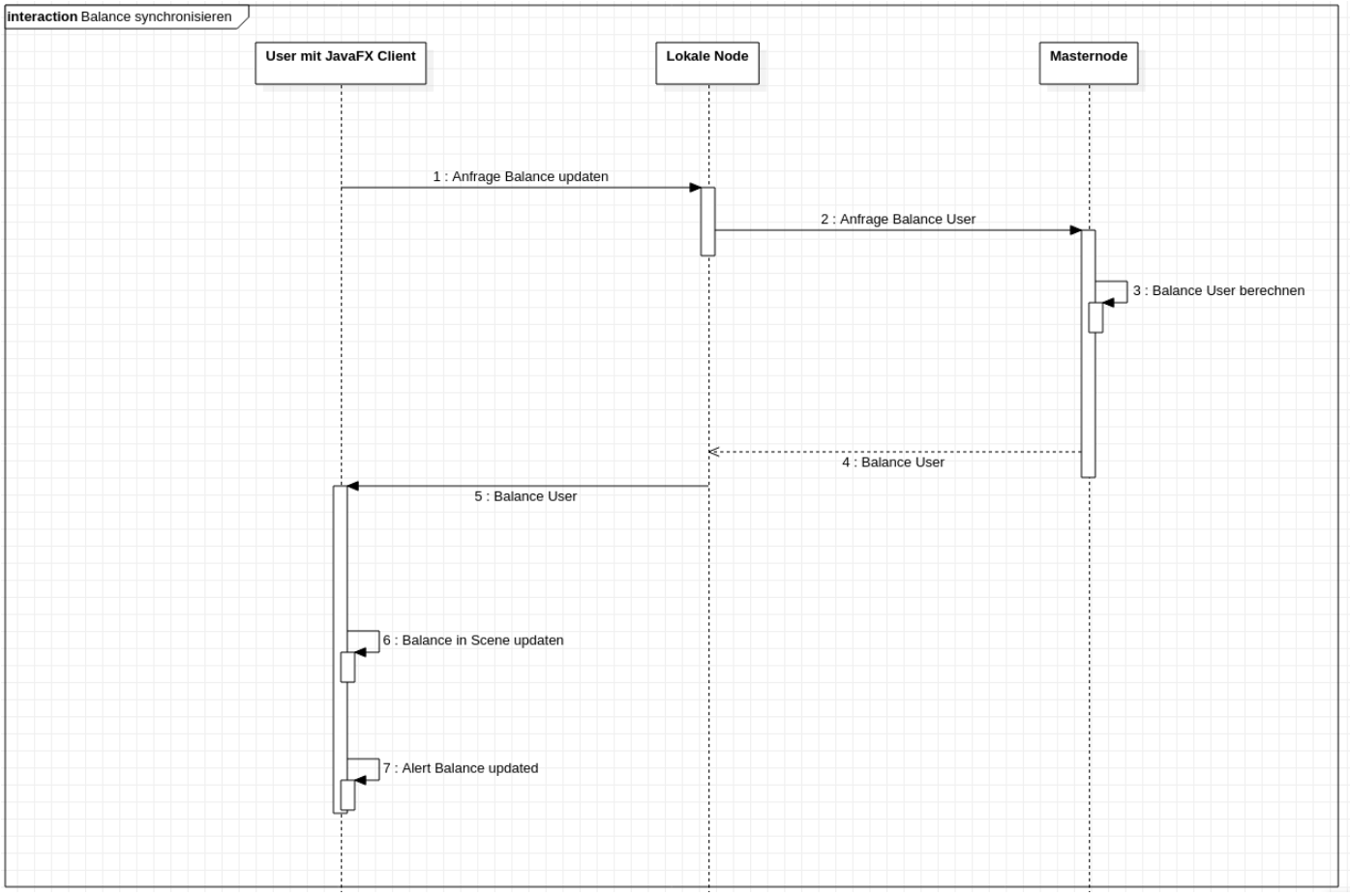
4.2 Sequenzdiagramm Transaktion erstellen



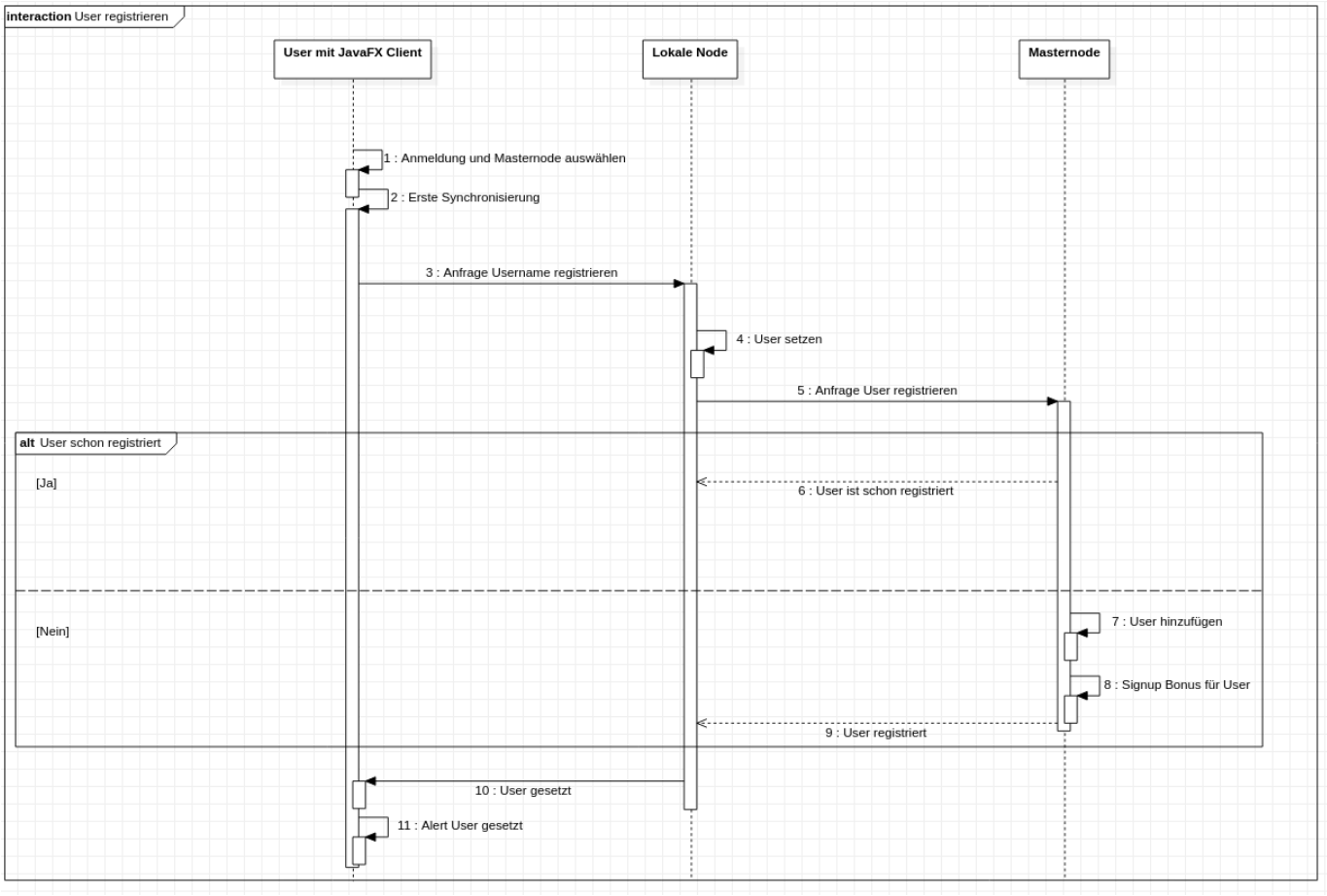
4.3 Sequenzdiagramm Masternode setzen



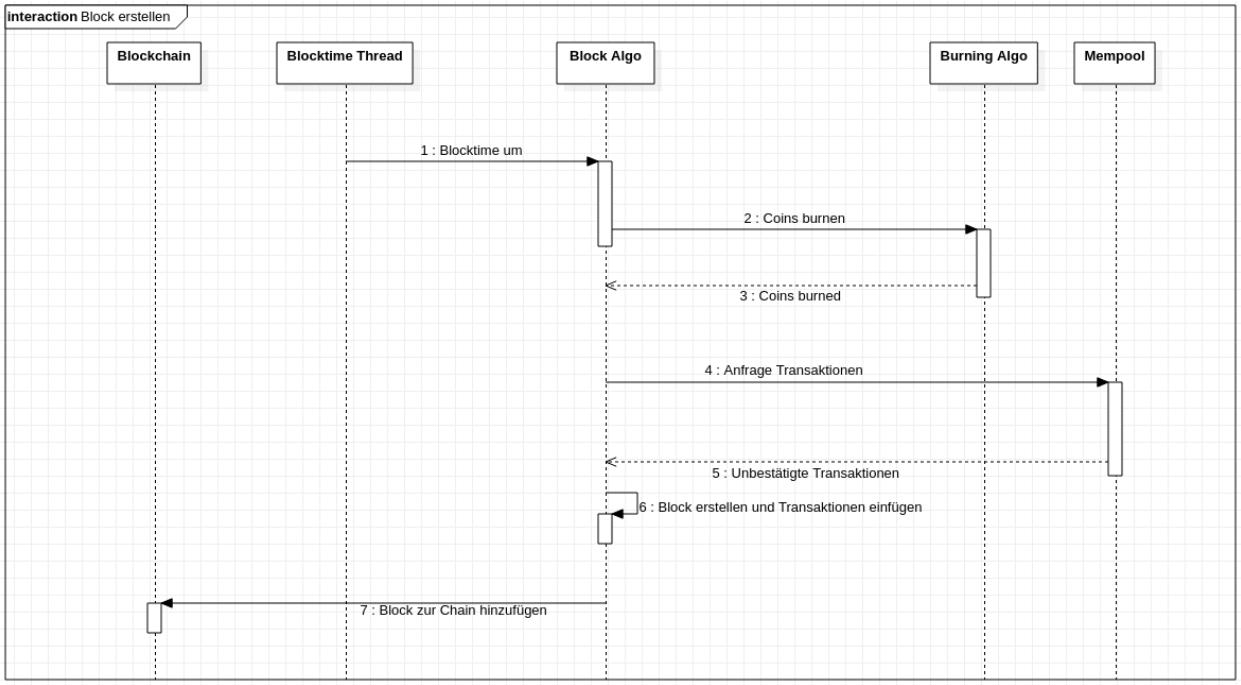
4.4 Sequenzdiagramm Balance synchronisieren



4.5 Sequenzdiagramm User registrieren



4.6 Sequenzdiagramm Block erstellen



5 Umsetzungsdetails

5.1 Zielplattform

Die Kryptowährung Kyas Coin wird auf verschiedenen Plattformen umgesetzt. Zielplattform für Masternode und normale Node ist Python (wenn möglich über Version 3). Somit können Nodes auf allen möglichen Linux, Windows und MacOS Systemen laufen. Der JavaFX Client benötigt eine Java Version gleich oder größer als Java 10, da sie spezielle Java 10 Funktionen nutzt. Der Android Client benötigt mindestens Android Lollipop (Android 5.0), da spezielle AppCompat-Bibliotheken verwendet werden. Der Betrieb der Clients und der Python Scripts für lokale Nodes ist nicht sehr rechenaufwendig und kann somit auch auf älteren Computer-/ Handymodellen stattfinden. Masternodes jedoch sollten jedoch in Umgebungen mit mehr Rechenleistung laufen, da sie viel mehr Überprüfungen und Berechnungen erledigen müssen. Masternodes sind auch ein viel wichtigerer Bestandteil des gesamten Systems. Eine funktionierende Internetverbindung wird überall vorausgesetzt.

5.2 Datenhaltung

Die Datenhaltung erfolgt im RAM des jeweiligen Geräts. Die Speicherung der Blockchain ist zwar möglich, jedoch muss das Speichermedium dann eine ähnliche Geschwindigkeit wie Arbeitsspeicher haben um Verzögerungen zu vermeiden. Zu diesem Zeitpunkt wird die Speicherung der Blockchain in Medien außer dem Arbeitsspeicher und Swap Speicher nicht unterstützt. Bei Masternodes, welche immer die gesamte Blockchain lokal im RAM haben, wird geraten je nachdem wie viel RAM die Blockchain auf dem Gerät verbraucht, entweder mehr RAM einzubauen oder Swap Speicher bereitzustellen.

5.3 Oberfläche

Die Oberfläche wird mit JavaFX implementiert. Die einzelnen Szenen werden mit Gluon Scene Builder mittels FXML Dateien erstellt. Gewisse Aspekte des Look and Feel werden mittels CSS angepasst. Mittels dieser Oberfläche sind lokale Nodes ansteuerbar, welche sonst nur mittels API Calls angesteuert werden könnten. Masternodes gehen nur mittels API Calls anzusteuern und sollen generell als Daemon Prozesse im Hintergrund laufen um nahezu lückenlose Uptime sicherzustellen.

5.4 Systemarchitektur

Das System soll im Grunde in Masternodes, welche idealerweise auf leistungsstarken Servern laufen, und normalen Nodes, welche auch auf leistungsschwächeren Computern laufen können, aufgeteilt werden. Da lokale Nodes sich immer mit den Masternodes synchronisieren müssen, kann das System nicht ohne Masternodes funktionieren.

5.5 Hardwareanforderungen

5.5.1 Masternode

Arbeitsspeicher Minimum: 8GB

Arbeitsspeicher empfohlen: 32GB

CPU Minimum: 1 Core

CPU empfohlen: 4 Cores

5.5.2 Node

Arbeitsspeicher Minimum: 4GB

Arbeitsspeicher empfohlen: 16GB

CPU Minimum: 1 Core

CPU empfohlen: 4 Cores

5.5.3 Android Client

Arbeitsspeicher Minimum: 500MB

Arbeitsspeicher empfohlen: 1GB

6 Qualitätsmerkmale

6.1 Benutzbarkeitsanforderungen

Das System soll flexibel für unterschiedliche Arbeitsweisen einsetzbar sein. Die mitgelieferte Oberfläche muss nicht der einzige Weg sein, wie der User mit dem System, oder der Blockchain, interagieren kann. Da das Projekt Open Source ist, kann der User auch eine eigene Oberfläche entwickeln, mir welcher er die API Endpoints der lokalen Node ansteuern kann. Die mitgelieferte Oberfläche ähnelt anderen sogenannten Wallets von bekannten Kryptowährungen. Bei dieser Oberfläche wurde sogar eine Mining Funktion eingebaut, was nicht der Norm entspricht. Diese Funktion erspart dem User noch eine weitere Oberfläche benutzen zu müssen.

Die Android Applikation ist sehr minimalistisch gebaut, mit 3 einfache Activities (Main, Send und Settings). Die App besitzt aber keine Kopie der Blockchain und ist somit kein Node und kein Netwerkpeer.

6.2 Effizienzanforderungen

Man kann sagen, dass je weniger Transaktionen in der Blockchain sind, desto effizienter läuft sie. Jedoch ist es nicht das Ziel wenig Transaktionen zu haben, das Ziel ist es viele Transaktionen zu haben und sie mittels schnellem Speicher schnell abzuarbeiten. Damit das System gut laufen kann, werden die Daten der Nodes direkt im RAM gespeichert. Reicht der RAM nicht mehr aus, werden die Daten auf Swap Speicher gespeichert. Dieser Swap Speicher sollte minimal eine SSD sein. Geraten wird zu NVME bzw. Optane Speicher. Die einzige Operation, welche nicht ein gewisses Zeitlimit überschreiten sollte ist die Blockerstellung. Alle 120 Sekunden (plus maximal 10 Sekunden) muss ein neuer Block erstellt werden. Darin enthalten sind die Ausführung der Burning Operation, das Verschieben aller unbestätigten Transaktionen in den zu erstellenden Block und das Hinzufügen des erstellten Blocks zu Blockchain.

Die App ist so gebaut, dass alle Operation über LiveData/Observer Stacks laufen. Somit laufen alle API calls asynchron.

6.3 Wartbarkeits- und Portierbarkeitsanforderung

Das Produkt soll später in vielen verschiedenen Sprachen verfügbar sein und noch andere Funktionen wie z.B. Multi Masternode Betrieb, enthalten. Da das Produkt einer Open Source Lizenz unterliegt, ist es jedem User freigestellt neue Funktionen zu implementieren und diese dem Entwicklerteam als Erweiterung vorzuschlagen. Die Oberfläche ist frei änderbar, das heißt sollte einem User das Design nicht passen, kann er es selber ändern. Da die Userbase dieses Produktes wahrscheinlich ein gewisses Maß an Programmierfähigkeiten besitzt, sollten Änderungen zu implementieren für die Willigen kein Problem darstellen.

Die Android Applikation basiert auf Retrofit2 und RxJava/RxAndroid, Bibliotheken welche für lange Zeit Backwards-Compatible bleiben werden.

6.4 Sicherheitsanforderung

Da das System auf Blockchaintechnologie aufbaut ist die Sicherheit der Transaktionen und Blöcke sehr gut gewährleistet. Sollte jemand versuchen Transaktionen zu ändern, so würde er die ganze nachfolgende Chain verändern und die Inkorrektheit der Berechnungen würde mit Sicherheit der ersten

Node auffallen, welche sich zu synchronisieren versucht und die neue Chain verifiziert. Die lokalen Nodes können ausfallen, ohne dass es zu Problem kommen könnte. Falls es mehrere Masternodes gibt, bereitet der Ausfall einer Masternode keine großen Probleme wenn die übrigen die umgeleiteten User der ausgefallenen Blockchain bedienen können. Hier spielen die Faktoren Rechenkapazität und Internet Bandbreite eine große Rolle. Die Blockchain wird immer weiterlaufen solange es eine aktive Masternode gibt.

Aufgrund des begrenzten Zeitrahmens und der begrenzten Ressourcen des Projekts konnten viele grundlegende Sicherheits- und Datenschutzfunktionen, die eine Blockchain haben sollte, wie z.B. double-spending Kontrollen, Datenschutzkontrollen mit View Keys und Datengeschützte Ausgaben mit geheimen Spend Keys, nicht implementiert werden.

6.5 Gesetzliche Anforderung

Dieses Produkt unterliegt der GNU General Public License v3.0.

7 Projektplan

7.1 Zeitplan

- Erster Sprint: 16. April 2020 bis 20. April 2020
- Zweiter Sprint: 20. April 2020 bis 22. April
- Dritter Sprint: 22. April 2020 bis 24 April 2020

7.2 Kostenschätzung

- Kosten Designer für Logo: 550 €
- Kosten Blockchain, API und JavaFX Development: 3000 €
- Kosten App Development: 3000 €
- Kosten Server: 20 €

Somit würde sich die geschätzte Summe zuzüglich Nebenkosten auf ca. 7000 € belaufen.