Blockchain: Identity Management en Distributed Network

Architectuurdocument

Jeffrey van Hoven

30 mei 2018

Inhoudsopgave

1 Inleiding

Dit document is opgesteld ter behoeve van het ontworpen architectuur voor de onderdelen Identity Management en Peer-to-Peer netwerk. Het maakt gebruik van het 4+1 architectural view model om logischerwijs de verschillende geïnteresseerden te informeren over de keuzes die gemaakt zijn.

2 Systeem stakeholders en requirements

2.1 Stakeholders

Er zijn meerdere stakeholders die baat hebben bij de realisatie van dit project:

Quintor De opdrachtgever en tevens de eigenaar van het project. De organisatie heeft baat bij het opdoen van kennis gedaan door dit project. Tevens zal het de eindgebruiker zijn van het systeem.

Kevin Bos Heeft belang bij de realisatie van het onderdeel Distributed Network en Identity Management gezien de het gedeelte dat gerealiseerd wordt hem samen dient te werken met de componenten die voorgesteld zijn binnen dit document.

2.2 Requirements

2.2.1 Business rules

BR01	Berichten dienen van type req(uest), inv(entory), data en auth(entication) te zijn.
BR02	Transactietypes zijn: account – om een account te registreren in het netwerk, data – arbitraire data dat nog niet gedefinieerd is.

2.2.2 Functional requirements

Id	Beschrijving	Prioritering
FR01	Als gebruiker wil ik een transactie kunnen aanmaken.	Must have
FR02	Als gebruiker wil ik mijn data kunnen synchroniseren.	Should have
FR03	Als gebruiker wil ik connectie kunnen leggen met een deelnemer uit het Peer-to-Peer netwerk.	Must have
FR04	Als gebruiker wil ik mijn openstaande connecties kunnen inzien.	Could have
FR05	Als gebruiker wil ik kunnen toetreden in het Peer-to-Peer netwerk.	Must have
FR06	Als gebruiker wil ik een block kunnen aanmaken.	Must have
FR07	Als beheerder wil ik een gebruiker kunnen aanmaken.	Must have

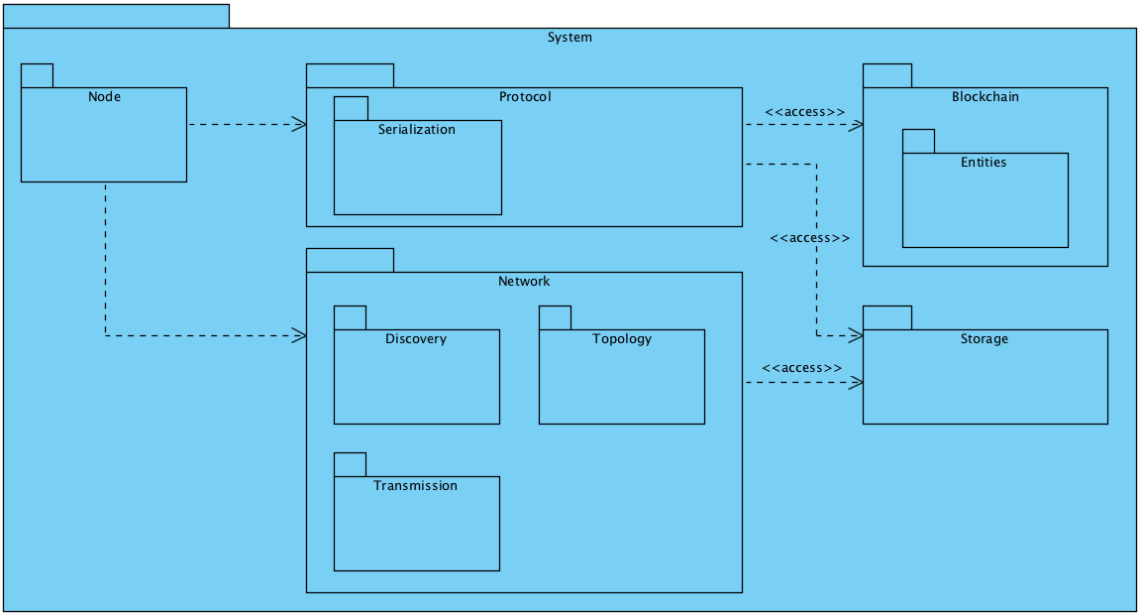
2.2.3 Non-functional requirements

Id	Beschrijving	ISO
NFR01	Het systeem dient om te kunnen gaan met deelnemers die de performance van het Peer-to-Peer netwerk proberen te verstoren.	Securability
NFR02	Het systeem dient om te kunnen gaan met het vervalsen van transacties.	Securability
NFR03	Het systeem dient makkelijk uitgebreid te worden door de kerncomponenten modulair op te stellen.	Maintainability
NFR04	Het systeem dient rekening te houden met protocol updates, en dient interactie met verouderde versies niet te ondersteunen.	Maintainability, Securability
NFR05	Het systeem dient makkelijk ingezet te kunnen worden.	Deployment

3 Architectuur views

3.1 Logical view

In de logische weergave wordt de architectuur benaderd vanuit het oogpunt van de eindgebruiker. Hierin komen de functionaliteiten van de verschillende componenten aan bod om de functionaliteit te ondersteunen.

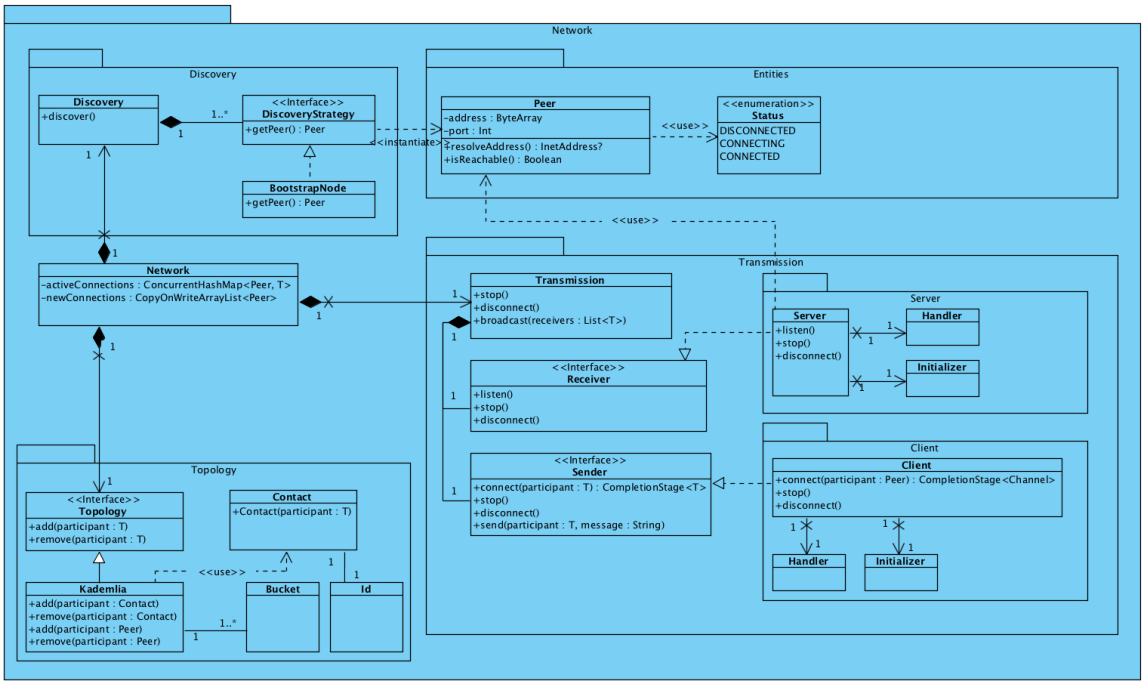


Figuur 3.1: Overzicht van het systeem

In fig. 3.1 is een overzicht te zien van de verschillende onderdelen van het systeem. De node is het startpunt van het systeem en maakt gebruik van een protocol specificatie om entiteiten uit de Blockchain op te maken in berichten die geschikt zijn voor het geïmplementeerde protocol.

Daarnaast maakt het gebruik van de netwerk specificatie om het toe te treden, de topologie te creëren en berichten die gemaakt zijn door het protocol te versturen.

Op de volgende pagina's zijn de verschillende componenten in detail gemodelleerd.



Figuur 3.2: Gedetailleerd overzicht van het Network component

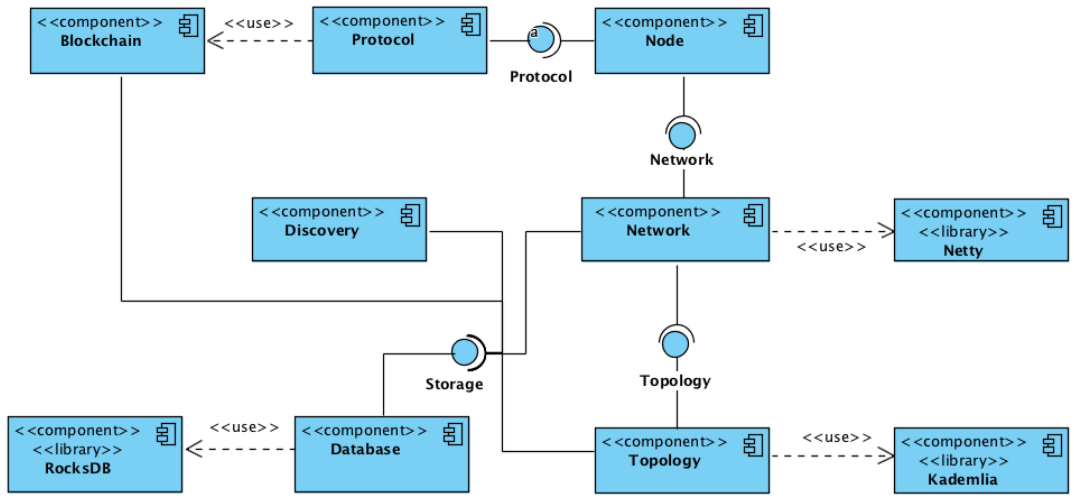
In fig. 3.2 is een gedetailleerd overzicht te van van het Network component. Het Discovery component is verantwoordelijk voor het uitvoeren van het Peer Discovery Protocol dat gebruikt wordt ten tijde van het toetreden van het netwerk. Er wordt hierbij gebruik gemaakt van een strategie, namelijk het opzoeken van een BootstrapNode.

Het Topology component is verantwoordelijk voor de structuur van het netwerk. Dit is modulair opgebouwd zodat het makkelijk gewisseld kan worden door een andere implementatie. De default topology is het Kademlia protocol.

Het Transmission component is verantwoordelijk voor het versturen en ontvangen van berichten. Dit is opgesplitst in een *Receiver* en *Sender* interface zodat het niet protocol specifiek geïmplementeerd hoeft te zijn.

3.2 Development view

De development weergave illustreert het systeem van een programmeur perspectief en omvat het Software Management gedeelte.



Figuur 3.3: Component Diagram waarin de diverse componenten en de samenwerking daartussen te zien is.

In fig. 3.3 is het component diagram te zien waarin de kerncomponenten van de applicatie staan. Hieronder zijn alle component individueel besproken:

- **Blockchain**
  Het Blockchain component bevat de logica en cryptografie om de structuur van een Blockchain op te bouwen. Een belangrijk onderdeel van het Blockchain component zijn de identiteiten die benodigd zijn voor de communicatie tussen verschillende participanten van het netwerk.

- **Protocol**
  Het Protocol component stelt de regels op met betrekking tot het gebruik van de Blockchain data.

- **Node**
  Het Node component bevat de functionaliteit waarmee de eindgebruiker kan interacteren.

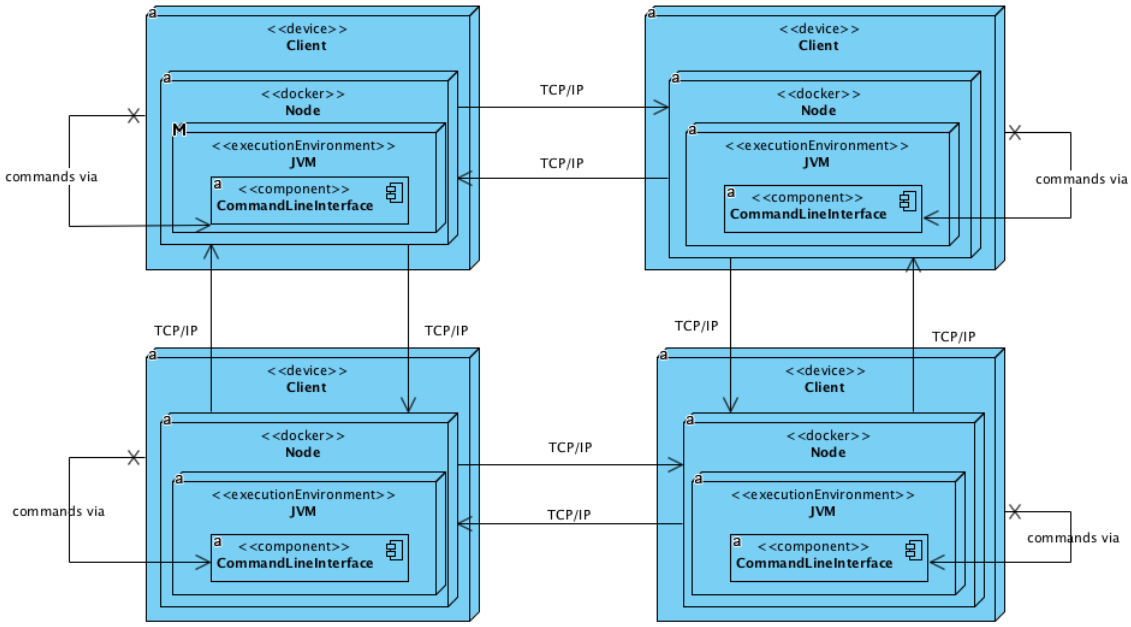
- **Network**
  Het Network component is een encapsulatie van de verschillende componenten die hier deel van uitmaken. Het is verantwoordelijk voor het opzetten van het gehele Peer-to-Peer netwerk.

- **Discovery**
  Het Discovery component bevat de Peer Discovery mechanisme die gebruikt om toe te treden in het netwerk.

- **Topology**
  Het Topology component bepaald de infrastructuur van het Peer-to-Peer netwerk.

- **Database**
  Het Database component bevat de logica om te interacteren met de gekozen database implementatie.

3.3 Physical view



Figuur 3.4: Deployment Diagram

In het Deployment Diagram is te zien dat er gebruik gemaakt wordt van Docker om de Blockchain client te draaien. Een vereiste hiervan is dat de Docker container beschikking heeft over de Java Virtual Machine. Communicatie tussen Blockchain clients gebeurt over TCP/IP waardoor een internetconnectie een vereiste is.

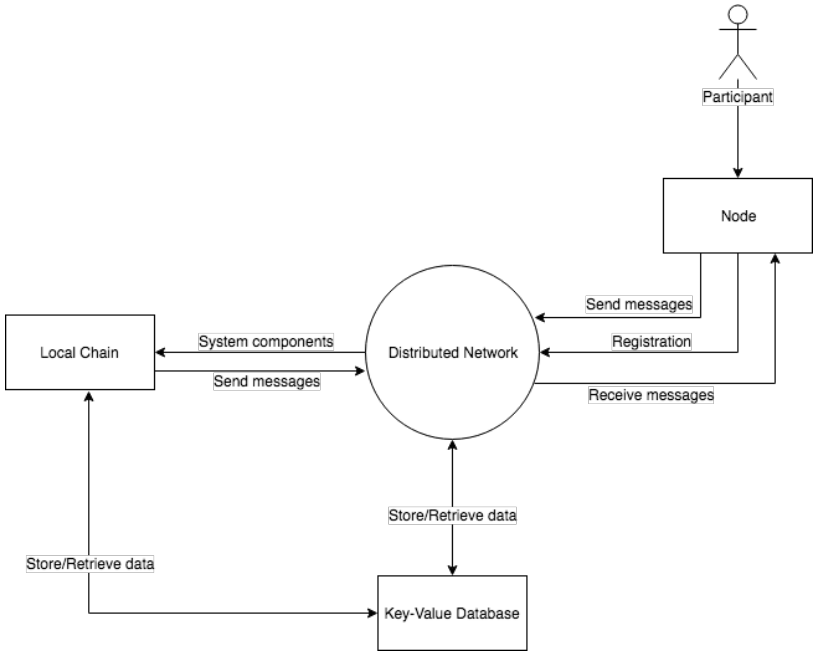
3.3.1 Software Dependencies

Om de applicatie te laten werken in de Docker container zijn er een aantal software modules nodig:

Module	Beschrijving
Maven	Wordt gebruikt om alle dependencies op te halen, en tevens het build proces uit te voeren.
RocksDB	Verzorgd de opslag binnen de applicatie.

3.4 Process view

De contextweergave van het systeem beschrijft de relaties, afhankelijkheden en interacties tussen het systeem en zijn omgeving (de mensen, systemen, en externe identiteiten waarmee het communiceert).

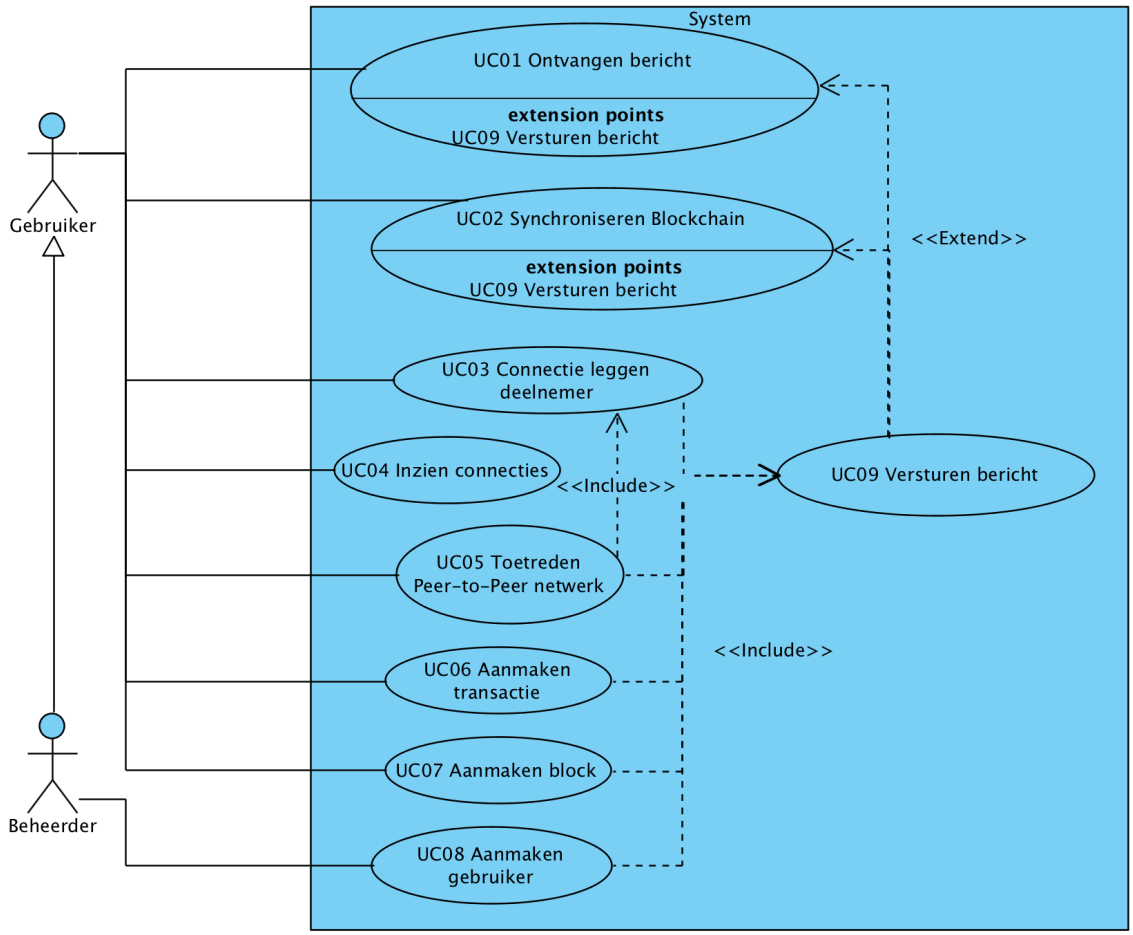


Figuur 3.5: Context Diagram waarin de interacties te zien is tussen het systeem en zijn omgeving.

De gebruiker draait een Node die gebruik maakt van het Peer-to-Peer netwerk om berichten te versturen. Een van de berichten is specifiek weergegeven aangezien het gaat om de registratie van een nieuwe gebruiker in het systeem. Het Distributed Network maakt gebruik van entiteiten uit de Local Chain om de benodigde data te versturen.

Zowel het Local Chain gedeelte als het Distributed Network maken gebruik van een Key-Value database om data op te slaan. In het geval van het Distributed Network gaat dit om informatie over connecties.

3.5 Scenarios



Figuur 3.6: Use-case diagram waarin de rollen binnen het systeem te zien zijn en de acties die zij kunnen uitvoeren.

Tabel 3.1: Use-case: Ontvangen bericht

Use-case	**Ontvangen bericht**
Id	UC01
Requirements	FR03, FR02, FR01
Beschrijving	Gebruiker ontvangt een bericht van een deelnemer uit het Peer-to-Peer netwerk
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	De gebruiker is verbonden met het Peer-to-Peer netwerk
Main flow	1. Systeem ontvangt bericht 2. Systeem valideert bericht type 3. Systeem deserialiseert bericht 4. Systeem controleert of er antwoord verstuurd dient te worden 5. Use-case eindigt (Postconditie: Success1)
Postconditie	Success1: Systeem heeft een bericht verstuurd naar verzender Failure1: Systeem is ongewijzigd
Alternatieve flows	1. Bericht is van type *req* (na MF4) 1.1. Systeem valideert dat gevraagde data aanwezig is 1.2. Systeem creëert *data* bericht 1.3. Systeem voert *UC09 - Versturen bericht* uit 1.4. Use-case eindigt (Postconditie: Success1) 2. Bericht is van type *inv* (na MF4) 2.1. Systeem valideert dat aangeboden data niet aanwezig is 2.2. Systeem creëert *req* bericht 2.3. Systeem voert *UC09 - Versturen bericht* uit 2.4. Use-case eindigt (Postconditie: Success1)

Tabel 3.2: Use-case: Synchroniseren Blockchain

Use-case	**Synchroniseren Blockchain**
Id	UC02
Requirements	FR02
Beschrijving	Gebruiker haalt Blockchain informatie op van een verbonden deelnemer
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	De gebruiker is verbonden met het Peer-to-Peer netwerk
Main flow	1. Systeem maakt *req* bericht 2. Systeem voert *UC09 - Versturen bericht* uit 3. Systeem voert *UC01 - Ontvangen bericht* uit 4. Systeem hercreëert Blockchain van ontvangen data 5. Use case eindigt (Postconditie: Success1)
Postconditie	Success1: Gebruiker is up-to-date met de laatste Blockchain data

Tabel 3.3: Use-case: Connectie leggen deelnemer

Use-case	**Connectie leggen deelnemer**
Id	UC03
Requirements	FR03
Beschrijving	Gebruiker maakt connectie met een deelnemer uit het Peer-to-Peer netwerk
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	De gebruiker is verbonden met het Peer-to-Peer netwerk
Main flow	1. Systeem vraagt om adresgegevens(ip, poort) van deelnemer 2. Actor vult informatie in 3. Systeem valideert adresgegevens 4. Systeem valideert dat deelnemer bereikbaar is 5. Systeem creëert *auth* bericht 6. Systeem voert *UC09 - Versturen bericht* uit 7. Use-case eindigt (Postconditie: Succes1)
Postconditie	Succes1: De gebruiker is verbonden met de deelnemer Failure1: Systeem is ongewijzigd
Alternatieve flow	1. Invalide adresgegevens (na MF3) 1.1. Use-case gaat verder bij MF1 2. Deelnemer is niet bereikbaar (na MF4) 2.1. Systeem toont foutmelding 2.2. Use-case eindigt (Postconditie: Failure1) 3. Actor annuleert (Overal)

Tabel 3.4: Use-case: Toetreden Peer-to-Peer netwerk

Use-case	**Toetreden Peer-to-Peer netwerk**
Id	UC05
Requirements	FR05
Beschrijving	Gebruiker wilt deel uitmaken van het Peer-to-Peer netwerk
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	Actor heeft een account tot zijn beschikking
Main flow	1. Actor start systeem 2. Systeem controleert of de actor niet reeds connectie heeft gemaakt 3. Systeem zoekt bootstrap node op 4. Systeem verstuurd authenticatie bericht naar bootstrap node 5. Systeem voert *UC01 - Ontvangen bericht* uit 6. Systeem ontvangt lijst van andere deelnemers die verbinding gemaakt hebben met het netwerk 7. Systeem voert *UC03 - Connectie leggen deelnemer* uit 8. Systeem voert *UC01 - Ontvangen bericht* uit 9. Systeem slaat adresgegevens (ip, port) op van deelnemer 10. Systeem voert *UC02 - Synchroniseren Blockchain* uit 11. Use-case eindigt (Postconditie: Success1)
Post conditie	Success1: Actor is actief in het netwerk. Failure1: Systeem is ongewijzigd
Alternatieve flows	1. AF1: Gebruiker heeft reeds connectie gemaakt (na MF2) 1.1. Systeem haalt lijst van opgeslagen deelnemers op 1.2. Systeem probeert verbinding te maken met deelnemers 1.3. Systeem voert *UC01 - Ontvangen bericht* uit 1.4. Use-case eindigt (Postconditie: Success1) 2. AF2: Actor gebruikt verkeerde identificatie (na MF5) 2.1. Systeem toont foutmelding 2.2. Use-case eindigt (Postconditie: Failure1)

Tabel 3.5: Use-case: Aanmaken transactie

Use-case	**Aanmaken transactie**
Id	UC06
Requirements	FR01
Beschrijving	Gebruiker wilt een transactie opslaan in de Blockchain
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	De gebruiker is verbonden met het Peer-to-Peer netwerk
Main flow	1. Systeem vraagt om public key ontvanger 2. Actor vult public key in 3. Systeem valideert public key 4. Systeem vraagt om type transactie 5. Actor selecteert type transactie 6. Systeem vraag aanvullende informatie gebaseerd op geselecteerde type 7. Actor vult aanvullende informatie in 8. Systeem valideert aanvullende informatie 9. Systeem maakt transactie van geselecteerde transactietype aan 10. Systeem creëert een *inv* bericht 11. Systeem voert *UC09 - Versturen bericht* uit 12. Use-case eindigt (Postconditie: Success1)
Post conditie	Success1: Systeem heeft een transactie aangemaakt
Alternatieve flows	1. Actor annuleert (Overal)

Tabel 3.6: Use-case: Versturen bericht

Use-case	**Versturen bericht**
Id	UC09
Requirements	FR01
Beschrijving	Gebruiker verstuurd bericht over het netwerk
Primaire actor	Gebruiker
Secundaire actor	-
Precondition	De gebruiker is verbonden met het Peer-to-Peer netwerk
Main flow	1. Systeem controleert bericht type 2. Systeem verstuurd bericht naar deelnemer 3. Use-case eindigt (Postconditie: Success1)
Postconditie	Success1: Systeem heeft bericht verstuurd naar deelnemer Success2: Systeem heeft bericht verstuurd naar alle verbonden deelnemers
Alternatieve flows	1. Bericht is van type *inv* (na MF1) 1.1. Systeem haalt lijst van alle verbonden deelnemers op 1.2. Systeem verstuurd bericht naar alle verbonden deelnemers 1.3. Use-case eindigt (Postconditie: Success2)