CRYPTOHELPER

Architectuur Document

Pedro Marques | s31 | fhict

Documenthistorie

Versie	Wijzigingen	Auteur	Datum	Verzendlijst	Verzoek
0.1	Opzet document, Inleiding, Domeinmodel	Pedro Marques	27-11		
0.2	Componenten, Persistentie	Pedro Marques	28-11		
0.3	Communicatie	Pedro Marques	29-11		
0.4	Detailed Design, Deployment	Pedro Marques	30-11		
0.5	Persistentie hermaken, Coin Client klassendiagram	Pedro	31-11		
1.0	Spellcheck, last touches	Pedro Marques	02-11		

Inhoud

Inleiding

Context van het project

Om de waarde van mijn project duidelijk uit te leggen het is belangrijk om te snappen wat blockchain technologie is.

Blockchain technologie is een soort van databasesysteem (heel primitief gezegd) dat gebruikt kan worden om waarde (geld of digitale items) te sturen naar een derde partij op een veilige manier, zonder een middenman (iemand dat de transactie voor je uitvoert – denk aan banken of services zoals PayPal).

Tegenwoordig gebruiken mensen middenmannen omdat ze de derde partij aan wie het sturen van goederen is bedoeld niet vertrouwen. Middenmannen geven de zekerheid dat een transactie werd uitgevoerd, zodat de derde partij deze informatie niet kan falsificeren - bijvoorbeeld door het communiceren dat het overgeschreven bedrag niet genoeg of helemaal niet binnen gekomen is. Het klinkt allemaal goed en nuttig, maar dit betekent ook dat banken en zulke instituties ook controle hebben over het uitvoeren van transacties en kunnen dus servicekosten invoeren of zelfs transacties blokkeren.

Blockchain technologie betekent dus een “bypass” op de middenman bij het uitvoeren van een transactie, onder andere voordelen, zoals:

- Het publiek en transparant stellen van alle transacties die worden gemaakt door een gebruiker. Bijvoorbeeld: In het geval van een overheidsinstitutie is dit nuttig, omdat dan kan iedereen zien waar er geld wordt besteden. (Blockchain records zijn voor iedereen beschikbaar).
- Het makkelijk kunnen verifiëren van eigendom.
- Toepassing op letterlijk alle services die tegenwoordig bestaan. (Er bestaat tegenwoordig een blockchain project voor bijna alles – verkoop van games (Enjin), het maken van andere blockchain projecten (ARK), anoniem dingen kopen (Monero), etc.

Cryptocurrency is een elektronische eenheid dat wordt “gecreëerd” op het moment dat een miner (iemand die zijn computer gebruikt om te contribueren voor de werking van een blockchain network) een transactie confirmeert.

Een miner confirmeert een transactie door een block (een series wiskundige problemen) op te lossen met behulp van een krachtige computer. Een transactie is altijd geassocieerd aan een block, en het aantal confirmaties van een transactie is de hoeveelheid aangemaakte blocks sinds de block die geassocieerd is aan de transactie. Een transactie moet tenminste 6 keer worden geconfirmeerd om als geldig te worden beschouwd.

Applicatie

Tegenwoordig ervaart de cryptocurrency markt een explosieve groei. Bedrijven beseffen steeds meer de potentieel dat blockchain technologie te bieden heeft. Dit betekent dat er steeds meer significante investeringen in deze technologieën worden gemaakt, wat ook betekent dat er prijsfluctuaties ontstaan in de cryptocurrency markt.

Crytophelper is een project met als doel het maken van een gedistribueerde app dat gebruikt kan worden om:

- **Geïnformeerde investeringen binnen het cryptocurrency wereld te maken;**
    - Cryptohelper beschikt van een overzicht scherm die alleen maar **relevante** cryptocurrencies op de markt aan de gebruiker toont. Door informatie (prijzen, trends, etc.) over de cryptocurrencies aan de gebruiker te presenteren en alles bij elkaar in een fatsoenlijke GUI te presenteren wordt het maken van een slimme investering makkelijker. Daardoor wordt de kans om winst te maken ook groter.
- **De gebruikers portfolio te controleren en een overzicht daarvan te geven;**
    - Een overzicht over welke investeringen die al gemaakt zijn is noodzakelijk om goede beslissingen te maken. Cryptohelper helpt hierbij door dit makkelijk zichtbaar aan de user te stellen, zodat hij/zij makkelijk een accuraat analyse van de situatie van zijn IDatabaseHelper kan maken.
- **Up to date te blijven met de laatste nieuws over de crypto wereld.**
    - Cryptohelper beschikt van een News tab waar relevante nieuws over cryptocurrency aan de user worden getoond. Vaak hebben gebeurtenissen in de cryptowereld een heel opmerkelijk invloed op de prijs van cryptocurrencies en hun respectieve market cap.

Cryptohelper is een driedelige product: Het houdt in een server die informatie vanuit API rest calls haalt voor gebruik in de client, de client zelf en een database component die verbinding maakt met de database server (DAL).

Niet-functionele eisen
De volgende niet-functionele eisen zullen worden besproken in deze document:

Performance
Alle netwerk-afhankelijk handelingen worden met een vertraging van maximaal 1 seconden uitgevoerd (URS. Q.1). Daarnaast worden alle GUI handelingen uitgevoerd met een vertraging van maximaal 500ms (URS. Q.2).

Betrouwbaarheid
Het systeem moet 99% van de tijd beschikbaar zijn (URS Q.6).

Compatibiliteit
Documentatie van zowel interfaces als klassen is beschikbaar (URS Q.4)

Beveiligbaarheid
De componenten van het systeem delen geen niet-noodzakelijke klassen/gegevens met elkaar. (Q.9)

Database query's worden vooraf bereid in het systeem om injectie problemen te voorkomen. (Q.10)

Het systeem maakt gebruik van actuele gegevens (Q.11)

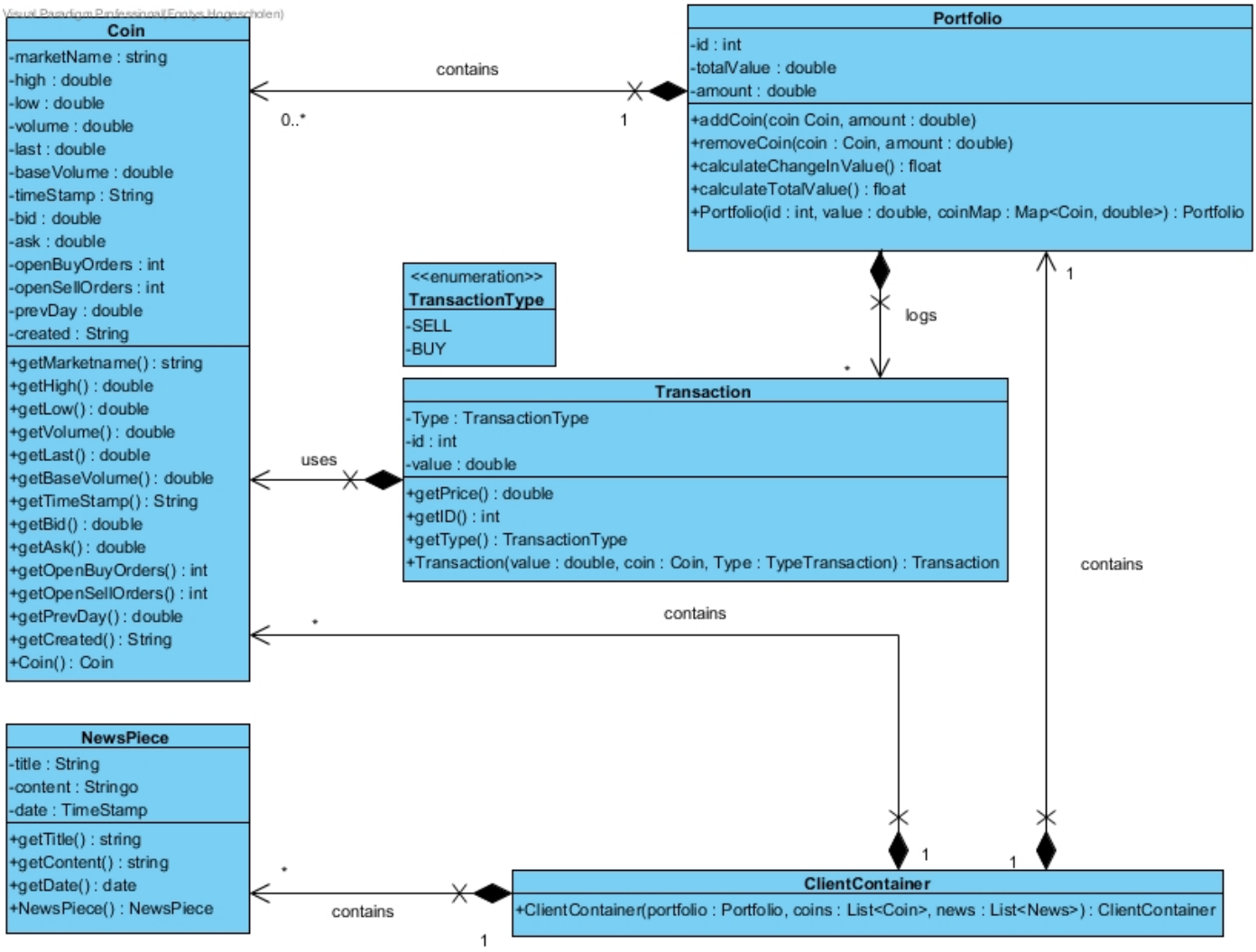
Onderhoudbaarheid
Het systeem is modulair opgebouwd.(Q.12)

Doel van dit document
Het doel van deze document is het verantwoorden van de software/hardware architectuur van de CryptoHelper applicatie.

De volgende aspecten van de architectuur van CryptoHelper zullen worden beschreven:

- Domeinmodel
- Persistentie
- Communicatie
- Realisatie van niet-functionele eisen
- Componenten
- Deployment
- Specificatie van Interfaces

Domeinmodel



Figuur 1: Klassendiagram van het domein.

Klassenuitleg

Coin:

Representeert een cryptocurrency eenheid. Wordt vanuit een API call gevuld. Een coin heeft de volgende attributen:

- **MarketName**: de naam van de coin op de markt;
- **High**: de hoogste prijs van de coin op de uur waar de informatie werd aangevraagd;
- **Low**: de laagste prijs van de coin op de uur waar de informatie werd aangevraagd;
- **Volume**: de waarde van alle transacties van de blockchain waar de coin bij hoort in de laatste 24h;
- **BaseVolume**: de volume van de basis currency (BTC) in de laatste 24h;
- Last: de hoogste prijs van de coin in de laatste uur;
- **TimeStamp**: de tijdstip waarop de Coin informatie werd aangevraagd
- Bid: de hoogste koopprijs op de markt;
- **Ask**: de laagste verkooprijs op de markt
- **OpenBuyOrders**: hoeveel koop aanvragen actief zijn voor de coin;
- **OpenSellOrders**: hoeveel verkoop aanvragen actief zijn voor de coin;
- **PrevDay**: de hoogste prijs van de coin in de laatste dag;
- **Created**: de datum van wanneer de coin op de markt was gezet;

Portfolio:

Representeert de totale opsomming van alle investeringen dat de gebruiker heeft gemaakt. Bestaat uit een lijst van coins, een lijst met bedragen (amount), een id, een double (totalValue) om de totale waarde van de Portfolio op te slaan en een lijst van transacties.

Een Porfolio kan zijn totale waarde berekenen, de algemene verandering in waarde berekenen en coins toevoegen of verwijderen door middel van het aanmaken van transacties.

Transaction:

Representeert een verandering in de hoeveelheid coins in het Portfolio. Bestaat uit een type (E.TransactionType) die aangeeft of het een verkoop of een koop transactie is, een id en het bedrag coins waar de transactie over gaat. Transacties worden opgeslagen in een Portfolio.

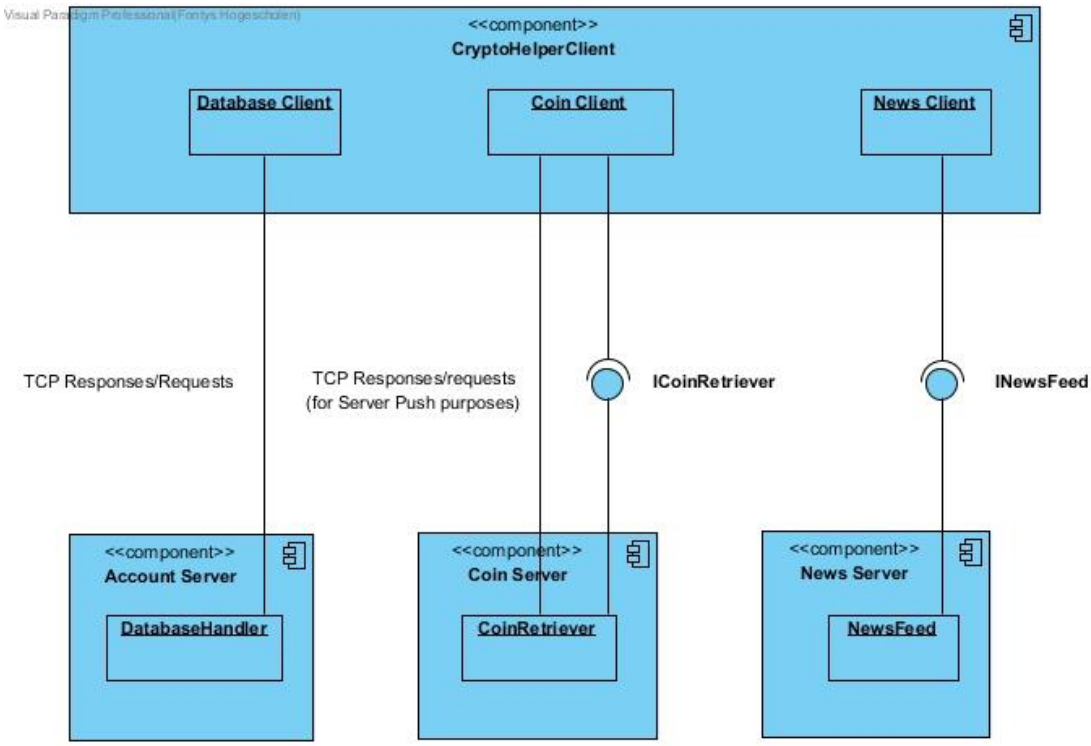
NewsPiece:

Representeert een nieuws artikel. Bevat een titel, inhoud en een datum.

ClientContainer:

Representeert een container voor een Portfolio, een lijst van coins en een lijst van nieuws.

Opdeling in Componenten



Component diagram

Figuur 2: Componentendiagram.

Het componentendiagram in Figuur 2 laat de 4 componenten van CryptoHelper zien met de klassen en interfaces die ze aanbieden/ontvangen. De componenten zijn:

- Account Server
  - Representeert de software applicatie die alle communicatie met de database behandelt. Dat houdt in het maken en verwijderen van accounts, het inloggen en uitloggen en het opslaan en laden van portfolio informatie. Bevat de class DatabaseHelper.
- Coin Server
  - Representeert de software applicatie die de cryptocurrency markt informatie ophaalt en vervolgens beschikbaar maakt. Bevat de class CoinRetriever.
- News Server
  - Representeert de software applicatie die e nieuws ophaalt en vervolgens beschikbaar maakt. Bevat de class NewsFeed.
- CryptoHelper Client
  - Representeert de software applicatie voor de gebruiker. Bevat de classes Database Client, Coin Client en News Client.

Koppeling tussen componenten

Om real-time communicatie uit te voeren worden er twee methodes toegepast: Het sturen van pakketten met requests en responses tussen Client en Account Server (Figuur 3), en Remote Method Invocation (Figuur 2). De RMI koppelingen tussen componenten worden gerealiseerd door gebruik te maken van twee interfaces: ICoinRetriever en INewsFeed.

Alle servers worden gekoppeld aan de bijbehorende Client class in CryptoHelper Client. Ik heb hiervoor gekozen om aan de Single Responsibility Principle te houden. Zo blijft het programma modulair en worden de data streams in aparte klassen gehouden. De classes in elke Client worden dan waar nodig in het programma gebruikt. Dit wordt niet in het diagram opgenomen om het schema overzichtelijk te houden.

Synchronisatie tussen componenten

Binnen de context van CryptoHelper is synchronisatie tussen de Coin Server en de Client belangrijk bij het berekenen van de waarde van een portfolio. Elke keer dat een berekening van portfolio waarde plaats vindt wordt er nieuwe informatie opgehaald zodat het prijs actueel is. Bovendien vindt er een automatisch update plaats elke 5 minuten.

Services per component

De volgende services worden gerealiseerd door middel van de twee interfaces in het diagram:

- ICoinRetriever
    - Vraag referentie naar lijst van Coin
    - Vraag update van lijst van Coin
- INewsFeed
    - Vraag referentie naar lijst van News
    - Vraag update van lijst van News

De volgende services worden per component aangeboden:

- Coin Server: ICoinRetriever
- News Server: INewsFeed
- Account Server: IDatabaseHelper, LogInResponse, CreateAccountResponse, DeleteAccountResponse, ForceLogout.
- Database Client , LogInRequest, CreateAccountRequest, DeleteAccountRequest

In CryptoHelper de enige component die services aanvraagt is de CryptoHelper Client. Die vraagt namelijk voor alle aangeboden services door de andere componenten.

Verder vindt log in communicatie plaats tussen de Database Server en de Crypto Client in de vorm van het uitwisselen van TCP pakketten. Voor meer informatie hierover zie de hoofdstuk Communicatie.

Allocatie van objecten
Hier worden beschreven waar welke instanties van welke objecten zijn gealloceerd.

- Coin Server: Instantie van CoinRetriever.
- News Server: Instantie van NewsFeed.
- Database Server: Instantie van DatabaseHelper.
- CryptoHelper Client: Instanties van DatabaseClient, CoinClient en NewsClient;

Remote objecten
Alle remote objecten (CoinRetriever en NewsFeed) worden in CryptoHelper Client geïnstantieerd.

Packagestructuur
Elke component heeft zijn eigen package. Daarnaast is er een Shared package voor klassen die in meer dan een component worden gebruikt, en een Network Package voor de requests en responses tussen twee connections.

Klassen/Interfaces per package:

- Shared: Coin, NewsPiece, ICoinRetriever, INewsFeed. Network (Bevat alle Requests en Responses)
- Client: CoinClient, NewsClient, DatabaseClient.
- CoinServer: CoinServer, CoinRetriever.
- NewsServer: NewsServer, NewsFeed.
- AccountServer: DatabaseServer, DatabaseHelper;

Communicatie tussen componenten

Communicatie Library

Communicatie binnen de componenten van CryptoHelper wordt gerealiseerd op twee manieren: Door middel van RMI en het gebruik van uitwisseling van pakketten.

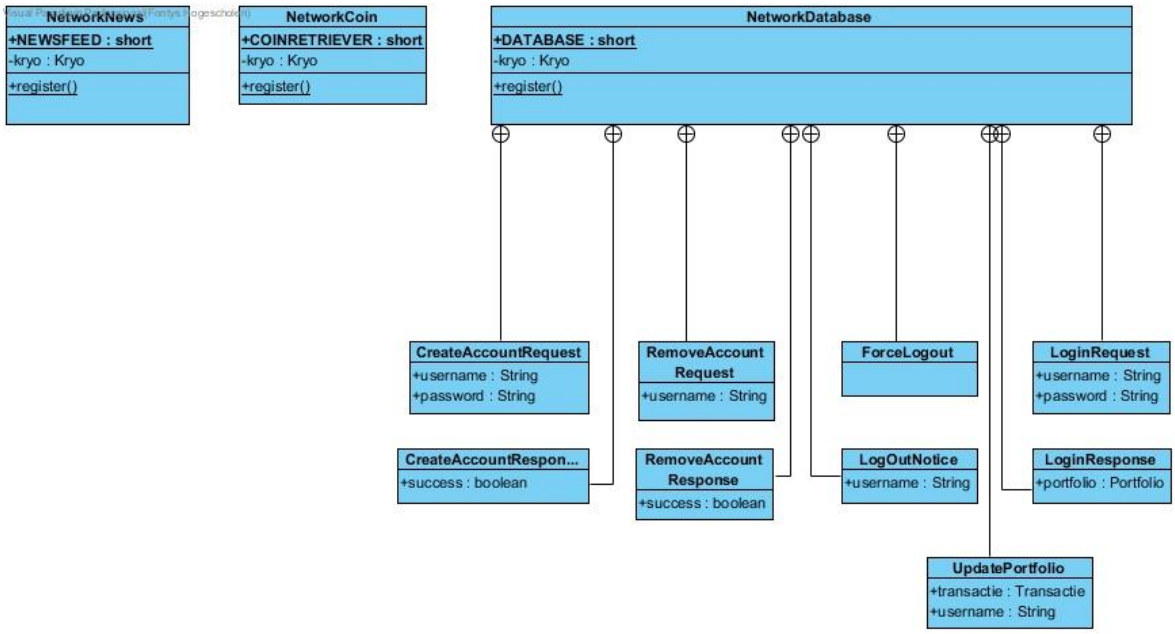
Om dit te coderen er wordt gebruik gemaakt van de Kryonet networking library (https://github.com/EsotericSoftware/kryonet). Er werd hiervoor gekozen omdat Kryonet versimpelt het opstellen van RMI en pakketten-communicatie door interfaces en serialisatie automatisch te doen, en maakt gebruik van methoden die goed te combineren zijn met de observable pattern.

Communicatie tussen Componenten

Tijdens het gebruik van CryptoHelper staat de CryptoHelper Client verbonden aan de andere drie componenten (Database, News en CoinRetriever Servers). Deze drie comuniceren tussen elkaar niet. Alle servers zijn stateless.

Er wordt Client-Pull RMI toegepast tussen de News/Coins Server en de Client. Er wordt Server Push RMI gebruikt tussen de Coin Server en de Client.

Ik zal per component uitleggen welke eigenschappen van welke objecten worden gecommuniceerd.

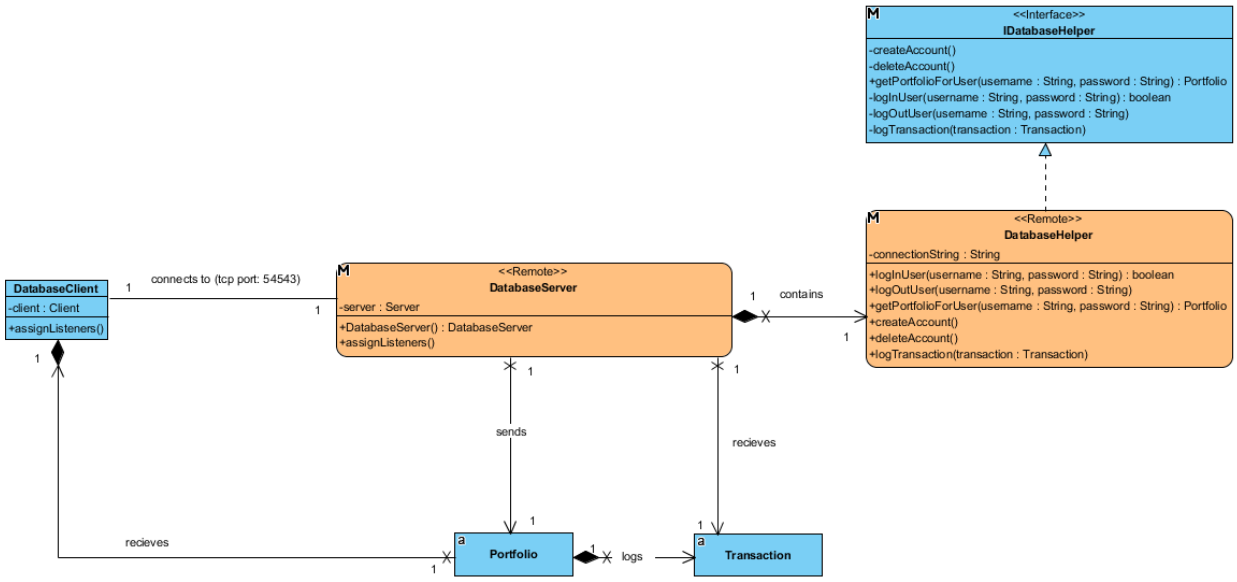


Figuur 4: Network klassen. Gebruikt om klasses/return types op de registry te registreren op twee endpoints van een verbinding, zodat ze op beide kanten van een verbinding beschikbaar zijn. Requests en responses worden als inner klassen gerepresenteerd, zoals beschreven en gesuggereerd in de Kryonet Github voorbeelden.

(https://github.com/EsotericSoftware/kryonet/tree/master/examples/com/esotericsoftware/kryonet/examples)

In de volgende diagrammen wordt er aangenomen dat de Client class leidend is in het realiseren van communicatie. Er zijn enkele uitzonderingen hiervoor.

Communicatie tussen Client en Database Server



Figuur 5: Klassendiagram van communicatie tussen Client en Account Server

Communicatie tussen de Client en Account Server component wordt gerealiseerd door middel van uitwisseling van requests en responses. De requests en responses bevatten attributen die over het netwerk verstuurd worden. In dit geval worden er strings, booleans, Transactie en Portfolio objecten verstuurd.

Het ontvangen van een request/response triggert een event in de connection endpoint waar het plaatsvindt. Deze events worden gebruikt om methoden uit te voeren binnen de klasse, om een response terug te sturen, of allebei. De Network klassen, requests en responses die hiervoor gebruikt worden staan uitgebeeld in Figuur 4.

Er vindt communicatie plaats tussen de Client en de Database Server op de volgende momenten:

Note: De standaard uitwisseling van pakketten staat beschreven in detail in de hoofdstuk interface specificatie, in de DatabaseHelper sectie.

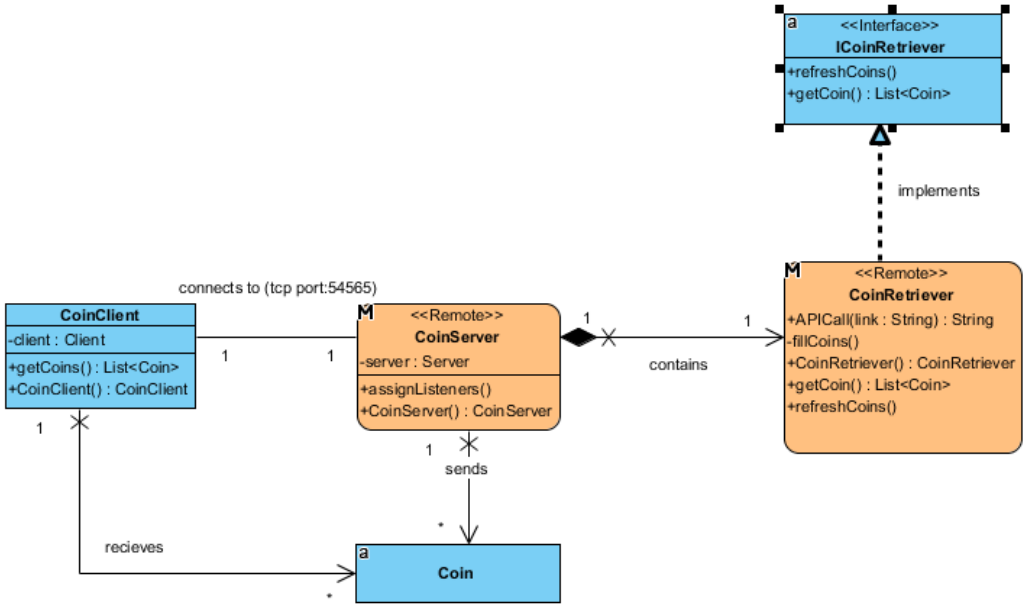
- Bij het inloggen van een user (Client Pull):
    - Er wordt een pakket verstuurd met een username en een password en een ontvangen met de Portfolio die gebonden is aan de inloggegevens.
    - Als de login succesvol is dan krijgt de actor een bericht hierover en krijgt hij/zij toegang tot de rest van het programma. Er wordt ook een sessie booolean naar true gezet en een portfolio opgeladen.
    - Als de login niet succesvol is dan krijgt de actor een bericht met het fout tijdens het inloggen.
    - Bij het inloggen wordt er een boolean veld in de database als true gezet. Zolang als deze veld true is kan niemand anders zich inloggen met de net ingevoerde gegevens.

- Bij het uitloggen van een user (Client pull):
    - Er wordt een pakket verstuurd met een username bij het uitloggen in de app. Het ontvangen van het pakket triggert een event die de user uitlogt.
    - Als er geen handmatig logout wordt verstuurd dan wordt de disconnect van de client gedetecteerd door een listener op de server en wordt de client alsnog uitgelogd.
- Bij het aanmaken van een account(Client pull):
    - Net als bij het inloggen, er wordt een pakket verstuurd naar de database server met een username en een wachtword. Het ontvangen van het pakket triggert een event op de server die een methode van DatabaseHelper uitvoert om een account te maken met de gekregen gegevens.
    - Als het aanmaken van een account niet lukt (bijv. door al bestaande gegevens) krijgt de user een message met de foutmelding.

- Bij het verwijderen van een account (Client pull):
    - Er wordt een pakket verstuurd met een username. Het ontvangen van het pakket triggert een event op de server die een methode van DatabaseHelper uitvoert om een account te verwijderen met de gekregen username.
    - Als het verwijderen van een account niet lukt (geen verbinding) krijgt de user een message met de foutmelding.

- Bij het updaten van een portfolio (nieuwe transactie) (Client pull):
    - Er wordt een pakket verstuurd met een transactie en een username. Het ontvangen van het pakket triggert een event op de server die een methode van DatabaseHelper uitvoert om een transactie toe te voegen aan het portfolio van de gekregen username.
    - Als de update niet gelukt is krijgt de user een foutmelding.
- Bij het detecteren van client idling (geen activiteit) (SERVER PUSH):
    - Hierbij is de server leidend.
    - Als er geen client activiteit is gedetecteerd voor 20 minuten wordt een ForceLogOut pakket verstuurd naar de betreffende client. Het ontvangen van het pakket triggert een logout in de client.
    - De gebruiker wordt verstuurd naar het inlogscherm.

Communicatie tussen Coin Client en Coin Server



Figuur 6: Klassendiagram van communicatie tussen CoinClient en CoinServer

Het klassendiagram in figuur 6 representeert de relaties tussen betrokken klassen en interfaces gebruikt in het versturen van lijsten van Coin instanties. In het geval van het updaten van een lijst door de server, moeten de remote objecten beschouwd worden als lokale, en de lokale als remote.

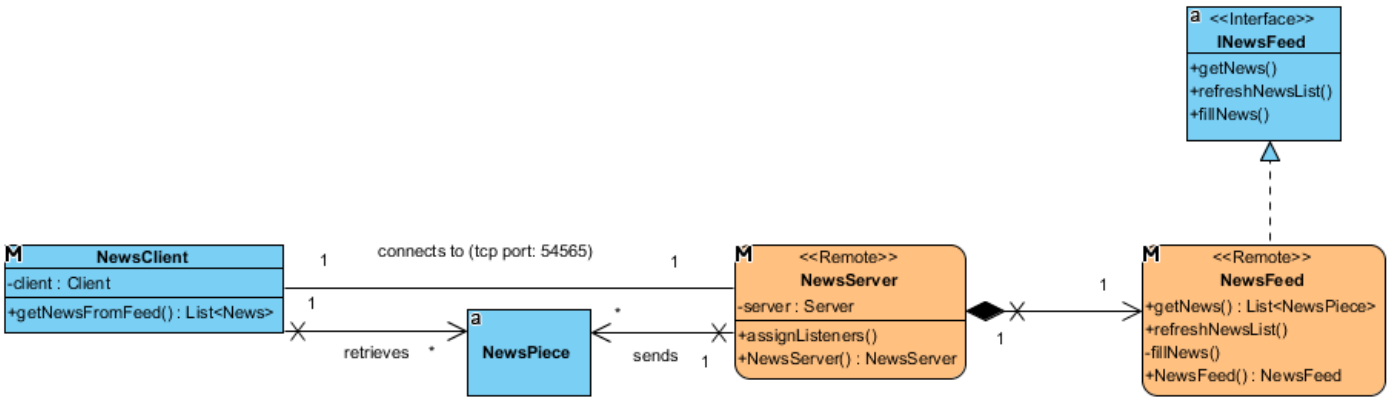
Tijdens gebruik van CryptoHelper wordt communicatie tussen deze twee componenten gerealiseerd door middel van RMI. Markt informatie wordt opgehaald met een API call. Er worden lijsten van Coin gecommuniceerd.

Coins worden gestuurd als een client zich inlogt, een refresh aanvraagt (client pull) of een automatisch server refresh actie (server push) die elk 5 minuten de informatie (JSON format) van de markt weer ophaalt via een API GET call. (Zie detailed design per component hoofdstuk)

Door gebruik te maken van een propertyChanged() event wordt de lijst van coins op de client automatisch geüpdatet elke keer dat de CoinRetriever zelf de methode refreshCoins() uitvoert.

Coins worden geserialiseerd door middel van gebruik van de gson library. (https://github.com/google/gson).

Communicatie tussen News Client en News Server



Figuur 7: Klassendiagram van communicatie tussen NewsClient en NewsServer

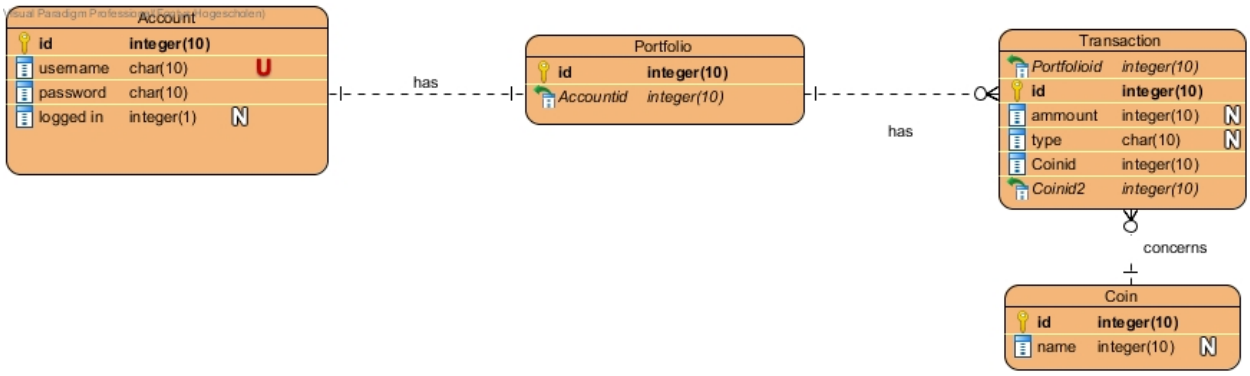
Het klassendiagram in figuur 6 representeert de relaties tussen betrokken klassen en interfaces gebruikt in het versturen van lijsten van NewsPiece instanties.

De communicatie tussen de News Client en de News Server componenten wordt gerealiseerd door middel van RMI volgens het Client pull principe. Hier worden NewsPiece lijsten verstuurd.

Zoals bij de communicatie tussen de Coin Client en Coin Server wordt er gebruik gemaakt van de client pull principe voor als de gebruiker de newsfeed vernieuwt. Er wordt geen gebruik gemaakt van server push bij deze twee componenten.

De NewsFeed klasse gebruikt een RSS feed (Really Simple Syndication, XML bestanden uit de web) als resource om de NewsPiece instanties op te bouwen. (Zie Detailed Design Per Component Hoofdstuk)

Persistentie per component



Figuur 8: Database Ontwerp Diagram van dataopslag in CryptoHelper

Persistentie binnen CryptoHelper

Om de login gegevens, sessie, transacties en de Portfolio van een klant op te slaan wordt er gebruik gemaakt van een MySQL database.

In Figuur 8 wordt beschreven wat wordt opgeslagen in de context van de CryptoHelper applicatie. **Alle dataopslag vindt plaats in de Account Server component.**

Er zijn 4 entiteiten:

- Account: Representeert een user account.
    - Username: de gebruikersnaam van een account. Deze is uniek.
    - Password: de wachtwoord van een account.
    - Logged in: een integer veld die aangeeft of de account in gebruik is of niet. 1 staat voor ingelogd en 0 voor niet ingelogd.

- Portfolio: Representeert een Portfolio. Is verbonden aan een account via foreign key.
- Transaction: Representeert een Transaction. Is verbonden aan een Portfolio via foreign key.
    - Amount: bedrag coins bij de transactie
    - Type: soort van transactie. Verkoop of Koop.
    - Coinid : soort coin waar de transactie over gaat. FK naar Coin.
- Coin: Representeert een coin. Is verbonden aan transacties via foreign key. Bevat een naam.

Data opslag tijdens gebruik van CryptoHelper

Data opslag gebeurt binnen CryptoHelper op de volgende momenten:

Bij het aanmaken van een account:

Als een account wordt aangemaakt worden de username en password aan een id nummer geassocieerd en opgeslagen in de database. Een Portfolio wordt gemaakt en verbonden aan de accountid.

Bij het inloggen:

Om log in sessies bij te houden en ongeoorloofde gebruik tegen te gaan, wordt het logged in veld in de database naar 1 gezet. Als iemand probeert met gegevens van een al ingelogde account in te loggen krijgt hij/zij een error message en wordt de toegang geweigerd.

Bij het uitvoeren van transacties:

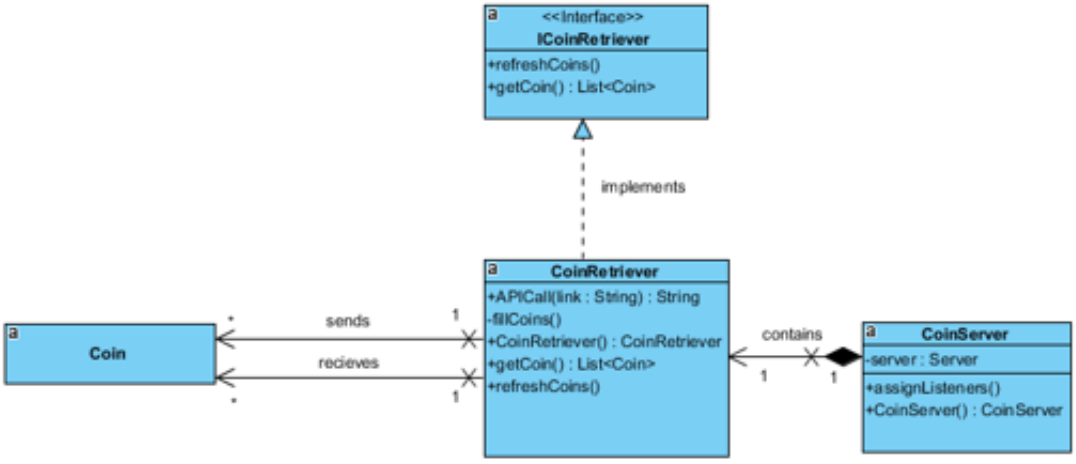
Tijdens gebruik van CryptoHelper worden alle updates aan de Portfolio van de klant opgeslagen op het moment dat ze uitgevoerd zijn. Dus elke keer dat de klant een transactie uitvoert wordt de Portfolio data entry van de user op de database meteen geüpdatet.

Bij het uitloggen:

Op het moment dat een user zich uitlogt wordt de login veld naar 0 gezet.

Dit kan ook gebeuren als de user 20 minuten inactief is.

Detailed Design per Component

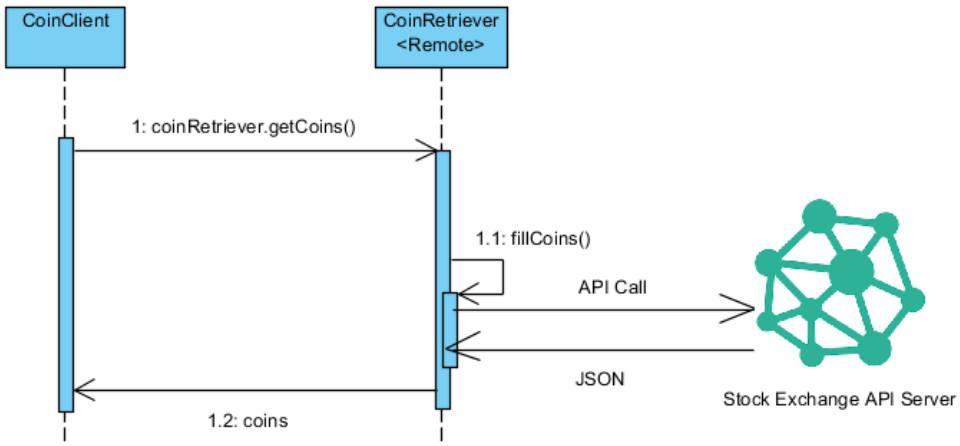


Figuur 9: Klassendiagram van Coin Server component

Coin Server

De klassendiagram in figuur 9 is de representatief voor de klassen van het Coin Server component. Coin Server heeft als doel het ophalen en versturen van een lijst van Coins naar de Client. Dit wordt gerealiseerd door gebruik te maken van de ICoinRetriever interface.

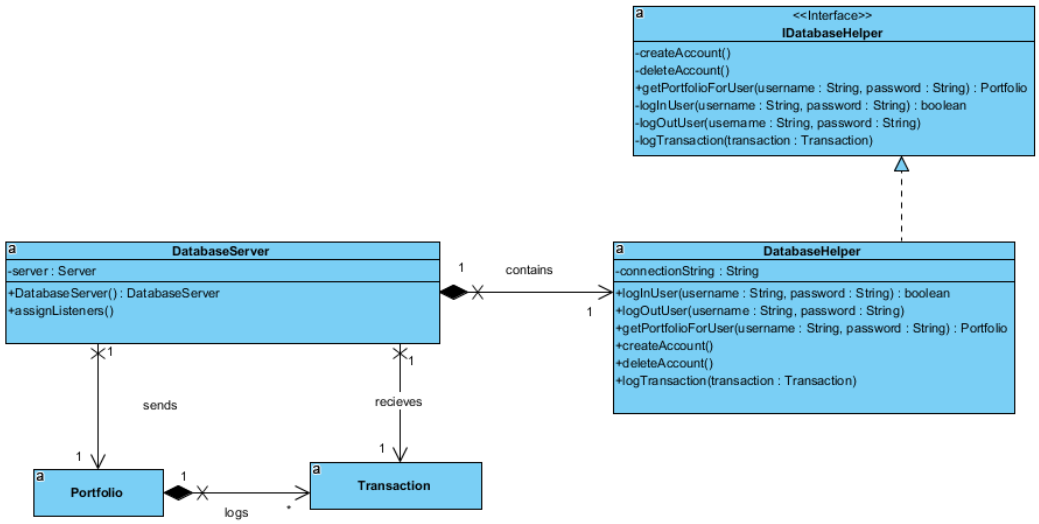
Zoals elke server component in CryoHelper, Coin Server bestaat uit een Kryonet Server object, die alle networking beheert door middel van een combinatie van het beschikbaar stellen van de server op een netwerk en event listening (events bij een nieuwe verbinding, disconnect, , etc.).



Figuur 10: Sequence Diagram van het ophalen van coins bij de log in. (Client Pull)

Naast de server bevat de CoinServer klasse een instantie van CoinRetriever, die een realizatie is van ICoinRetriever. Deze instantie wordt gegeven aan de client via RMI en gebruikt om lijsten van coins te sturen (Figuur 10) en vernieuwen. Bij een automatisch vernieuwing (gebeurt elke 5 minuten) wordt de lijst coins gepusht naar de client via een event. Vernieuwingen kunnen ook door de client aangevraagd worden.

Database Server

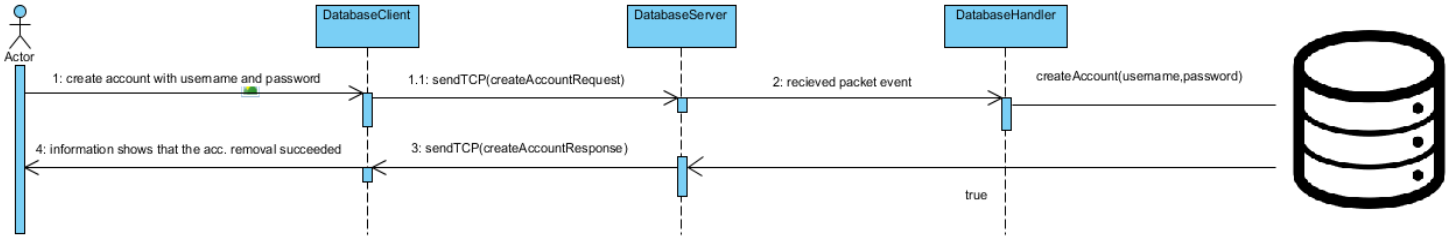


Figuur 11: Klassendiagram van Account Server component

De klassendiagram in figuur 9 is de representatief voor de klassen van het Account Server component.

Het Account Server component heeft als doel de communicatie tussen de Client component en de database beheren. Dit houdt in het maken van parameterized queries naar een database server en het bijhouden van login sessies.

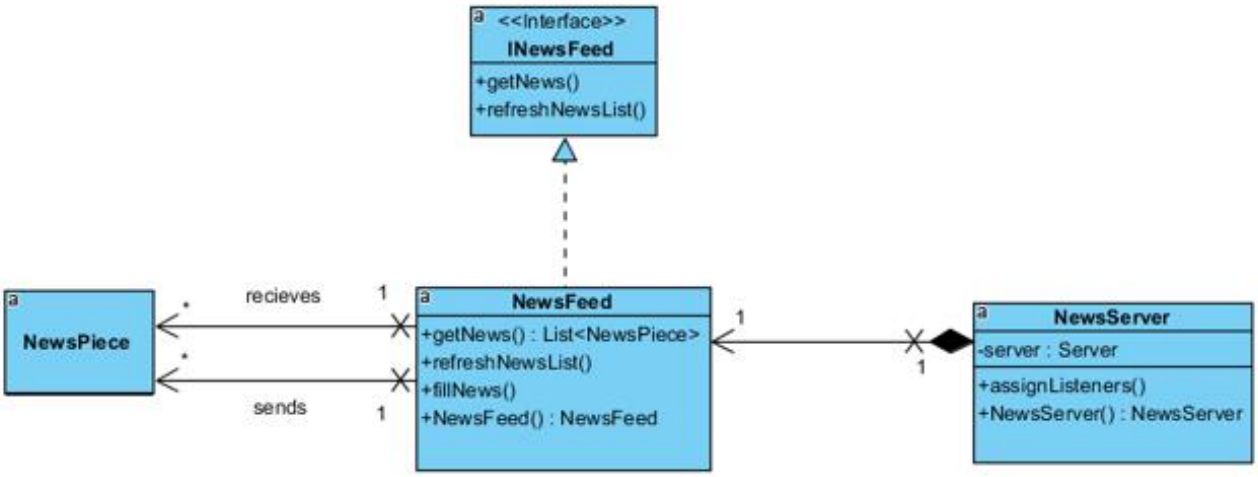
Zoals elke server component in CryptoHelper, de DatabaseServer klasse bestaat uit een Kryonet Server object, die alle networking beheert door middel van een combinatie van het beschikbaar stellen van de server op een netwerk en event listening (events bij een nieuwe verbinding, disconnect, etc.).



Figuur 12: Sequence diagram van het versturen en behandelen van pakketten door de Account Server component in het geval van een succesvol query. removeAccount kan vervangen worden door elk andere Request van de NetworkDatabase klasse (Figuur 4, blz. 14).

De DatabaseHelper klasse voert alle communicatie tussen database en de component in de vorm van SQL queries met parameters. Het manier waarop de communicatie gerealiseerd wordt staat beschreven in de Communicatie hoofdstuk (blz. 15-17).
De sequence diagram op Figuur 12 beschrijft hoe de event listener gebonden aan de kryonet.Server object binnen de DatabaseServer klasse reageert op het ontvangen van pakketten.

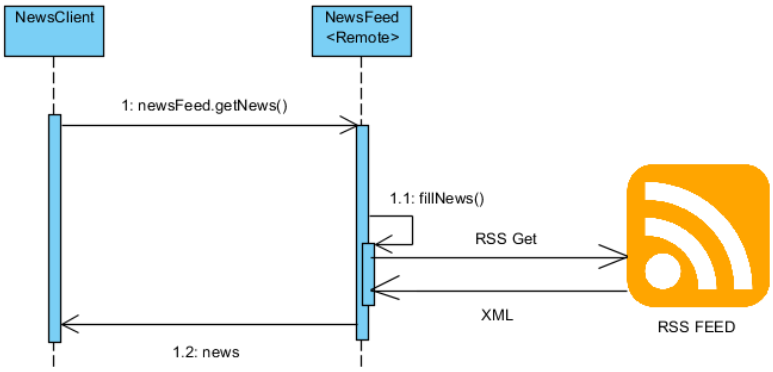
News Server



Figuur 13: Klassendiagram van News Server component

De klassendiagram in figuur 13 is de representatief voor de klassen van het News Server component. De News Server component heeft als doel het versturen van een lijst van News naar de Client. Dit wordt gerealiseerd door gebruik te maken van de INewsFeed interface.

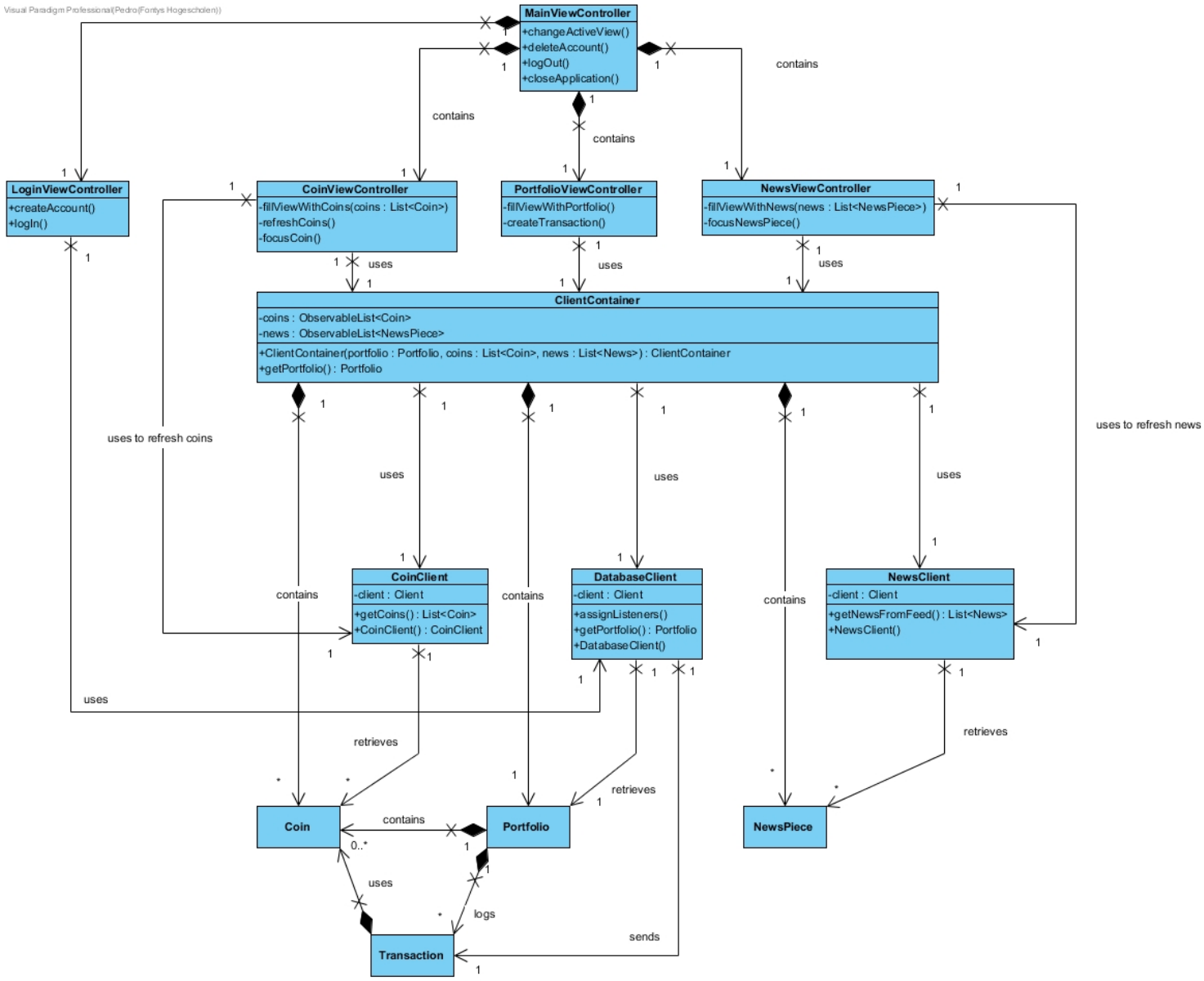
Zoals elke server component in CryptoHelper, de NewsServer klasse bestaat uit een Kryonet.Server object, die alle networking beheert door middel van een combinatie van het beschikbaar stellen van de server op een netwerk en event listening (events bij een nieuwe verbinding, disconnect, etc.).



Figuur 14: Sequence diagram van het ophalen van nieuws uit een RSS Feed.

Naast de server bevat de NewsServer klasse een instantie van NewsFeed, die een realisatie is van INewsFeed. Deze instantie wordt gegeven aan de client via RMI en gebruikt om lijsten van NewsPiece te sturen (Figuur 14). Vernieuwingen kunnen door de client aangevraagd worden.

CryptoHelper Client



Figuur 15: Klassendiagram van de CryptoHelper Client Component.

De klassendiagram in figuur 15 is de representatief voor de klassen van het CryptoHelper Client component. Deze component heeft als functies:

- Het samenstellen van opgehaalde data (Coin, Portfolio, Transaction en NewsPieces) tot een zinnige en makkelijk te begrijpen format voor de gebruiker.
- Een manier aan de gebruiker geven om de functionaliteit van het programma te kunnen gebruiken via GUI.
- Het communiceren van door de gebruiker uitgevoerde handelingen aan de Account Server component.

In de context van de CryptoHelper Client bestaan er drie lagen:

- De communicatielaag (verder beschreven in de Communicatie hoofdstuk)
    - Bestaat uit de drie Client klassen. Hier worden alle businessobjecten opgehaald en naar andere componenten verstuurd (bijvoorbeeld bij het uitvoeren van een transactie).
- De business objecten laag (beschreven in de Domein Model hoofdstuk)
    - Bestaat uit de Transaction, Portfolio, Coin, NewsPiece en de ClientContainer. De container groepeert alle business objecten en maakt ze beschikbaar via een observable list in het geval van de lijsten van Coin en NewsPiece, en via een getter voor Portfolio.
    - De observables worden gebruikt door de controlers voor automatische updates elke keer dat een lijst ververst wordt.
- De controller laag, die uit 4 verschillende controllers bestaat
    - Bestaat uit alle Controllers in figuur 15. Elke controller beheert een view voor een bepaalde deel van het programma. De controllers beheren de functionaliteit van de GUI en verbinden de View aan de Logic van het programma. Om de View op te bouwen wordt er gebruik gemaakt van FXML bestanden. Dit komt omdat het bewerken van FXML bestanden met SceneBuilder meer design vrijheid geeft dan het maken van een scene in code.
    - De MainViewController serveert als parent voor de andere views. Hier vindt het uitwissellen van main view plaats. Deze class bevat ook methoden om de user uit te loggen en een account te verwijderen.
    - De PortfolioViewController beheert de GUI deel waar er veranderingen aan een portfolio kunnen worden uitgevoerd. Hier vindt het uitvoeren van transacties plaats.
    - De CoinViewController beheert de GUI deel waar Coins kunnen worden gezocht en geanalyseerd door de gebruiker. Hier komt alle marktinformatie (list van Coin) in. Hier kan worden aangevraagd om de lijst van Coin te verversen.
    - De NewsViewController is dezelfde als de CoinViewController, maar voor NewsPiece.

In deze component worden de controllers zijn subscribers van de ClientContainer lijsten. De lijsten in ClientContainer kunnen worden ververst, en dat veroorzaakt dat de controllers meteen nieuwe lijsten krijgen.

Realisatie Niet-Functionele Eisen

Betrouwbaarheid

1. *Het systeem is tenminste 99% van de tijd beschikbaar:*
    a. Door alle servercomponenten stateless te maken wordt de load van de server laag gehouden, omdat de server houdt geen instanties van client gerelateerde objecten (denk aan Portfolio per klant, etc). Het gevolg hiervan is dat alle communicatie een stuk sneller loopt. Dit leidt tot een kleinere kans tot crashes en connection overload, en dus naar een langere functioneel status.
2. *Het systeem maakt gebruik van actuele gegevens:*
    a. Door de refresh functionaliteit in de Server Componenten kunnen er continu actuele gegevens worden aangevraagd door de Klant. De Coin server bevat zelf een automatisch-refresh functionaliteit om elke 5 minuten de coin gegevens actueel te houden.
3. *Het systeem slaat veranderingen in een portfolio op:*
    a. Elke keer dat een transactie plaatsvindt wordt de betreffende portfolio opgeslagen in de database zodat er geen veranderingen kwijtgeraakt kunnen worden.

Performance

1. *De netwerk responsetijd bedraagt minder dan 1s*
    a. Zie Betrouwbaarheid 1.a
2. *De UI wordt niet voor langer dan 500ms geblokkeerd tijdens het uitvoeren van taken*
    a. Dit wordt opgelost door threading te gebruiken tijdens het updaten van de View. Er zal altijd feedback ontstaan tijdens het uitvoeren van taken.

Beveiliging

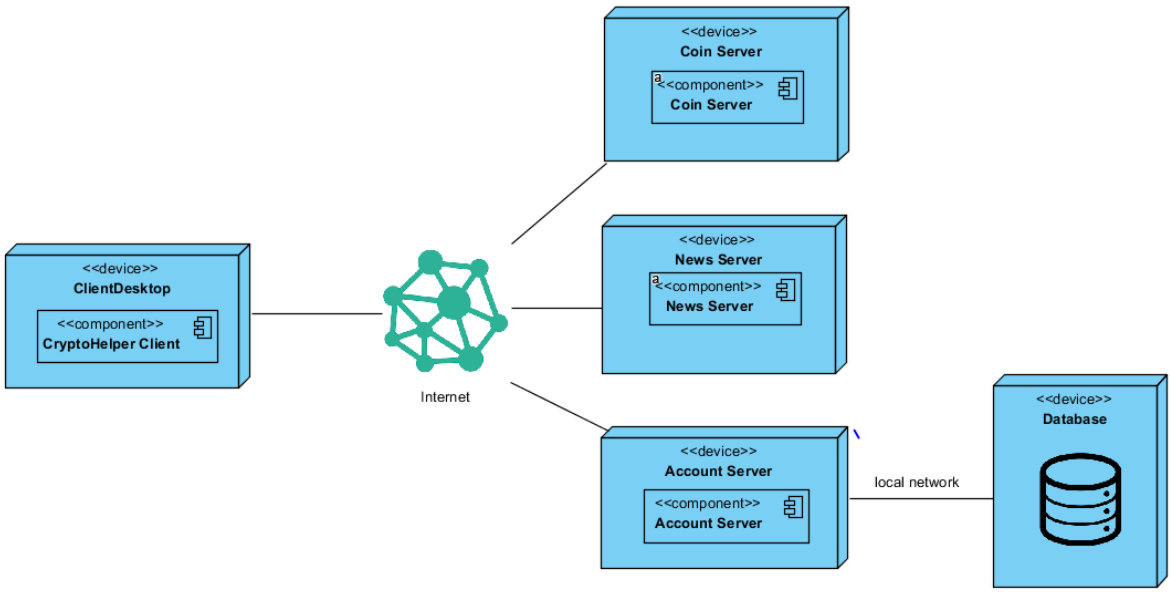
1. *Gegevens worden niet gedeeld tussen Servers:*
    a. Alle servers werken onafhankelijk van elkaar. Zo wordt de flow van informatie beperkt naar waar de informatie absoluut nodig is. Dit leidt naar minder kans op security lekken.
2. *Queries worden vooraf binnen het systeem gemaakt om SQL injectie te voorkomen:*
    a. Er wordt gebruik gemaakt van parameters tijdens het maken van database queries, zodat er geen SQL injectie mogelijk is.

Onderhoudbarheid

1. *Het systeem is modulair opgebouwd:*
    a. Omdat het systeem in verschillende componenten is verdeeld, alle componenten zijn te vervangen.

Deployment

Deploymentdiagram



Figuur 18: Deploymentdiagram.

Deployment van de gedistribueerde app is afgebeeld in Figuur 18. Het bestaat uit de volgende onderdelen:

- Een client desktop: Computer waar de CryptoHelperClient .jar executable is aanwezig.
- Coin Server: Server waar de CryptoHelperCoinServer .jar executable runt.
- News Server: Server waar de CryptoHelperNewsServer .jar executable runt.
- Account Server: Server waar de CryptoHelperAccountServer.jar executable runt. Deze server staat verbonden via local network met de database.

Specificatie van interfaces

IDatabaseHelper

Naam methode: logInUser(username : String, password : String) : bool

- Precondities:
  - De gebruiker heeft de CryptoHelper .jar gestart.
  - Een account bestaat al in zijn naam.
- Beschrijving:
  - Voert een SQL query in om de gebruikers gegevens te vergelijken met wat op de database opgeslagen staat. Als de gegevens goed zijn, return true. Zo niet, return false.
- Aanleiding voor excepties:
  - MySQLException als verbinding niet gelukt is gefaald.

Naam methode: createAccount(username : String, password : String) : bool

- Precondities:
  - De gebruiker heeft de CryptoHelper .jar gestart.
- Beschrijving:
  - Voert een SQL query in om een nieuwe account te maken. Return true als het gelukt is. Return false als de gegevens al bestaan.query niet gelukt is.
- Aanleiding voor excepties:
  - MySQLException als verbinding niet gelukt is of de query is gefaald

Naam methode: logOutUser(username : String) : bool

- Precondities:
  - De gebruiker is ingelogd.
- Beschrijving:
  - Voert een SQL query in om de gebruiker's session te beëindigen. Return true als dit gelukt is. Returnt false als de session al beëindigd is.
- Aanleiding voor excepties:
- MySQLException als verbinding niet gelukt is is gefaald.

Naam methode: deleteAccount(username : String, password : String) : bool

- Precondities:
  - De gebruiker is ingelogd
- Beschrijving:
  - Voert een SQL query in om de gebruikers account op de database te verwijderen. Hiervoor is zijn geldige inloggegevens nodig. Als de gegevens goed zijn, return true. Zo niet, return false.
- Aanleiding voor excepties:
  - MySQLException als verbinding niet gelukt is of query is gefaald

Naam methode: getPortfolioForUser(username : String) : Portfolio

- Precondities:
  - logIn(username) is true
- Beschrijving:
  - Voert een SQL query in om de gebruikers portfolio in de database op te halen bij een succesvolle login. Return Portfolio van username.
- Aanleiding voor excepties:
  - MySQLException als verbinding niet gelukt is of query is gefaald.

Naam methode: logTransaction(username : String, transactie: Transaction) : bool

- Precondities:
  - De gebruiker heeft net een transactie in zijn portfolio uitgevoerd.
- Beschrijving:
  - Voert een SQL query in om een transactie in de database op te slaan. Als de gegevens goed zijn, return true. Zo niet, return false.
- Aanleiding voor excepties:
  - MySQLException als verbinding niet gelukt is of query is gefaald.

ICoinRetriever

Naam methode: APICall(link : String) : String

- Precondities:
  - De CoinRetriever .jar is gestart
- Beschrijving:
  - Voert een API call om een JSON array met marktinformatie op te halen. Returnt een JSON array in de vorm van een String.
- Aanleiding voor excepties:
  - Gson.ConnectionFailed als verbinding niet gelukt (geen internet of api server down)

Naam methode: fillCoins() : void

- Precondities:
  - APICall is gelukt
- Beschrijving:
  - Serialiseert de resultaat van APICall(string) en vult een lijst van Coin in CoinRetriever. Wordt niet uitgevoerd als de APICall niet succesvol is, vanwege exception handling.

Naam methode: getCoin() : List<Coin>

- Precondities:
  - De CoinRetriever object staat geregistreerd in de objectspace en er worden coins aangevraagd.
- Beschrijving:
  - Returnt de lijst van Coin.
- Aanleiding voor excepties:
  - NotInitializedException als lijst niet bestaat.

Naam methode: refreshCoins() : void

- Precondities:
  - De CoinRetriever bevat een instantie van een lijst van Coin.
- Beschrijving:
  - Ververst de lijst van Coin.
- Aanleiding voor excepties:
  - NotInitializedException als lijst niet bestaat

INewsFeed

Naam methode: getNews() : List<NewsPiece>

- Precondities:
  - De NewsFeed bevat een instantie van een lijst van Coin.
- Beschrijving:
  - Returnt de lijst van coin.
- Aanleiding voor excepties:
  - NotInitializedException als lijst niet bestaat

Naam methode: refreshNews() : void

- Precondities:
  - De CoinRetriever bevat een instantie van een lijst van Coin.
- Beschrijving:
  - Ververst de lijst van Coin.
- Aanleiding voor excepties:
  - NotInitializedException als lijst niet bestaat

Naam methode: fillNews() : void

- Precondities:
  - Het programma is gestart.
- Beschrijving:
  - Genereert een lijst NewsPieces vanuit een RSS resource (XML file) van de internet.
- Aanleiding voor excepties:
  - ConnectionError als de resource niet beschikbaar is of geen verbinding.