Beurs

Rob Bonhof

Verslag over de beursopdracht waarin het protocol en het programma beschreven staan.
Dit voor het vak internet en netwerken van de opleiding technische informatica

Inhoudsopgave

H1 INLEIDING 3

H2 ONTWERP 4

H3 PROTOCOL 6

H4 TESTPLAN 8

H4.1 GEBRUIKERTEST 8
H4.2 VERENIGINGTEST 9
H4.3 AANDEELTET 10
H4.1 AANBIEDINGTEST 12

Versie	Inhoud	Datum	Auteur
1.0	Hoofdstukken toegevoegd	10-02-2015	Rob Bonhof

Tabel 1 Versiebeheer

H1 Inleiding

Voor het vak internet en netwerken van de opleiding technische informatica dient er een opdracht gemaakt te worden die een beurssysteem ontwerpt voor de studieverenigingen van Saxion. Dit systeem bestaat uit een server en client, waarbij de server de data bevat. Deze data kan de client opvragen volgens een te ontwerpen protocol en weergeven in zijn grafische user interface. In dit document is het ontwerp te vinden voor dit systeem samen met de gemaakte keuzes en een testplan. Het document eindigt met een beknopte handleiding hoe het product gebruikt dien te worden. Doel van deze opdracht is het verder bekend raken met socket programmeren en het ontwerpen van de bijbehorende protocollen.

H2 Ontwerp

In dit hoofdstuk wordt het ontwerp van de software behandeld, het gebruikte protocol volgt in een apart hoofdstuk. Allereerst zal er dieper op de server in worden gegaan. De server bestaat uit enkele dao's en een achterliggende database. In de database worden de gebruikers, verenigingen, aandelen en aanbiedingen opgeslagen. Het volgende strokendiagram geeft deze tabellen weer.

Gebruikersnummer	Gebruikersnaam	Wachtwoord	Naam	Balans

Tabel 2 Gebruiker

Verenigingsnummer	Naam

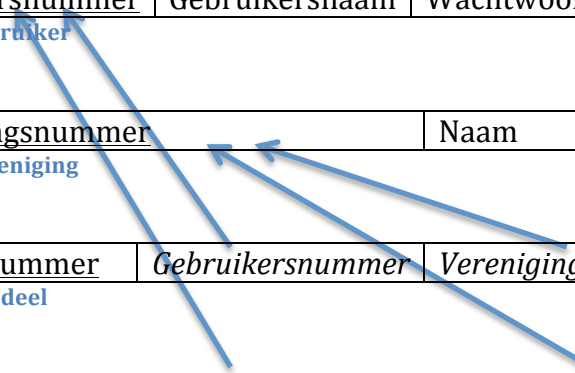
Tabel 3 Vereniging

Aandeelnummer	*Gebruikersnummer*	*Verenigingsnummer*	Aantal

Tabel 4 Aandeel

Aanbiedingsnummer	*Gebruikersnummer*	*Verenigingsnummer*	Aantal	Prijs

Tabel 5 Aanbieding

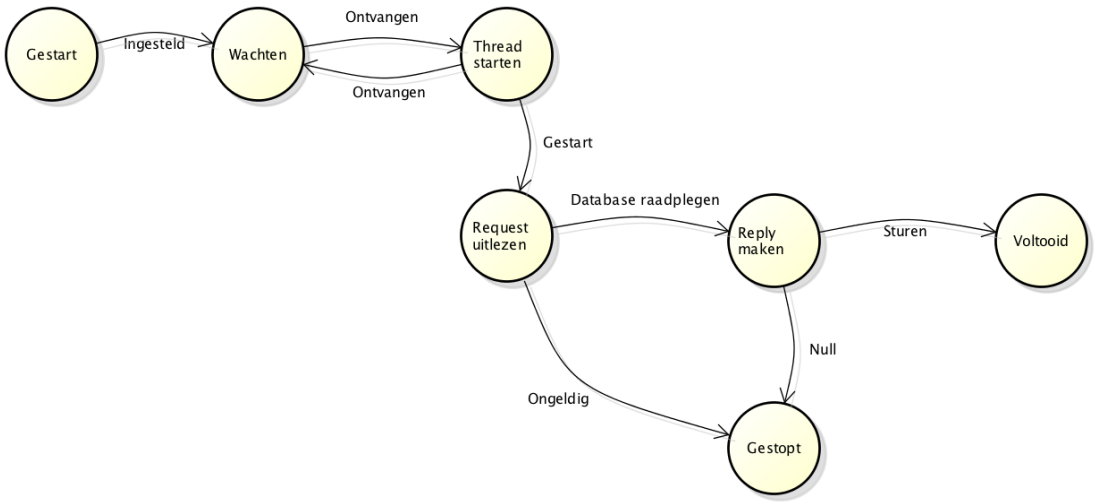


Op basis van het binnenkomende request wordt de juiste methode van de dao aangeroepen met de meegegeven parameters. Dit resultaat wordt vervolgens weer terug gestuurd naar de client verstuurd.

De client is ontworpen met behulp van JavaFX, deze bestaat uit controllers en views. Het model is de server en requests en reply's worden afgehandeld in de controller en weergegeven in de grafische user interface. De interface bestaat uit een menu, account en beurs view. Via het menu kan het programma worden gestopt en de gebruiker worden uitgelogd, ook kan het programma in de beginstand worden gebracht waarbij de 5000 aandelen gegenereerd worden van de verenigingen. In de accountview kan een gebruiker zich inloggen en kunnen de gegeven en balans worden ingezien. Ook kunnen de aandelen en aanbiedingen worden ingezien en de aanbiedingen worden geplaatst, gekocht of verwijderd. Deze aandelen en aanbiedingen worden in de beursview weergegeven.

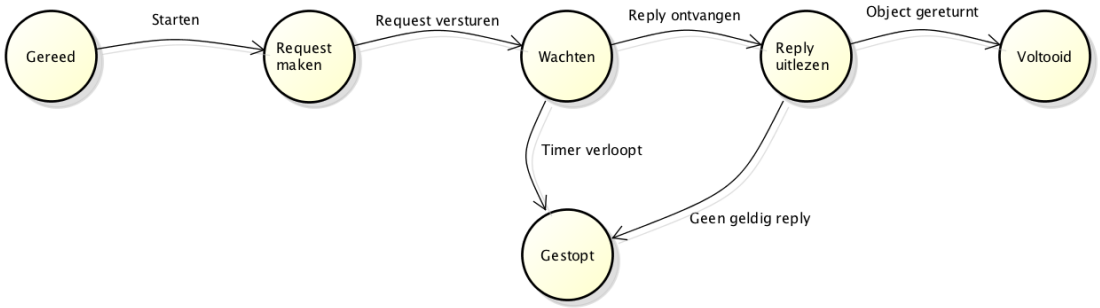
Voor elk datatype is er een clientthread, deze thread is een task. Het voordeel van een task is dat deze thread een object returnt, dit is het opgevraagde datatype. De task kan gemonitord worden en als deze klaar is kan het datatype object opgevraagd worden, dit gebeurd door middel van een listener. Er kunnen dus meerdere threads gestart worden en meerdere verbindingen tegelijk worden aangegaan met de server, waardoor er meerdere data tegelijk kan worden opgevraagd. Dit levert voordeel op bij grote batches.

De server en client zijn dus aparte threads buitenom het programma. Voor een enkele request wordt dus steeds een client opgestart. De client loop daarom van gereed tot voltooid, waarbij die laatste de data gereturnt is. In de tussentijd wordt de request voorbereid en verstuurt en de reply uitgelezen, bij fouten in dit proces wordt de thread gestopt en geen data terug gegeven. Dit is in het volgende diagram weergegeven.



Figuur 1 Client Statediagram

De server is een thread die altijd blijft lopen en op binnenkomende verzoeken verwacht. Wanneer er zo'n verzoek binnenkomt wordt er een vervolgthread gestart die het verzoek afhandelt. De server keert terug naar zijn originele staat en wacht weer op nieuwe verbindingen. Op deze manier kan de server meerdere verbindingen aangaan. De vervolgthread is voltooid wanneer er een reply terug is gestuurd, de verbinding wordt dan verbroken. Voor de server is dit hieronder weergegeven.



Figuur 2 Server Statediagram

H3 Protocol

Dit hoofdstuk gaat over het zelf ontworpen protocol en de manier waarop deze werkt. Het gebruikte transportprotocol is TCP aangezien het van belang is dat de data zonder fouten de client bereikt. Dit zou anders fouten in grafische user interface kunnen veroorzaken of verkeerde weergave van data. De data wordt textueel verstuurd wat extra bytes oplevert dan het binair versturen, echter bevorderd dit de leestbaarheid ondermeer door de flags gelijk te stellen aan enumeraties. Ook is deze data ook makkelijker uit te lezen in Java met read en write methodes en string makkelijk te splitten op een delimiter. Het protocol bestaat uit een request of een reply. De request worden door de client gestuurd en deze ontvangt een reply van de server met de gewenste data. Hieronder is dit protocol globaal weergegeven.

Request
xyz[parameters]
x = datatype (gebruiker, vereniging, aandeel, aanbieding)
y = actiontype (get, add, delete)
z = parametertype (none, gebruiker, vereniging, aandeel, aanbieding, vergen)

Reply
[parameters]

De parameters in de requests en reply's zijn gescheiden door een delimiter, de keuze is gevallen op het '|' character, wat een algemeen gebruikte character is voor het scheiden van data. Bij de request worden eerst nog flags gestuurd, waaraan de server kan zien welke methodes hij moet aanroepen in de dao's. De x is het datatype wat de client opvraagt oftewel het returntype van de dao-methode. De y flag staat voor het type actie wat op de database wordt uitgevoerd, bij get wordt er een query uitgevoerd, bij add wordt een datatype toegevoegd en bij delete wordt een tuple verwijdert uit de database. De z parameter is enkel voor een get actie, deze geeft aan met welke parameters er gezocht dient te worden. Deze komen overeen met de parameters van de dao methodes.

Het protocol levert de volgende interactie op, hier is als voorbeeld gebruikt het opvragen van een aanbieding op basis van zijn aanbiedingnummer. Als laatste wordt de opgevraagde aanbieding geprint in de console. Voor de volledigheid is deze op de volgende pagina weergegeven.

```
CLIENT Maak verbinding met server
CLIENT Aanvraag voor aanbieding gestuurd:
AANBIEDINGGETAANBIEDING1|
SERVER Verbinding geaccepteerd
SERVER Message ontvangen: AANBIEDINGGETAANBIEDING1|
SERVER Sluit de verbinding
CLIENT reply ontvangen:
1|1|Archimedes|Archimedes|voorzitter|37.0|1|Archimedes|4985|5.
0
CLIENT Verbinding gesloten
Aanbieding [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 37.0], aandelenlijst=[]]], vereniging=ObjectProperty
[value: Vereniging [PRIMARYKEY=1, naam=StringProperty [value:
Archimedes], aandelenLijst=[]]], aantal=IntegerProperty
[value: 4985], prijs=DoubleProperty [value: 5.0]]
```

H4 Testplan

Voor het ophalen van de data zijn er dus verschillende clients, namelijk een voor gebruiker, vereniging, aandeel en aanbieding. In deze test worden de clients getest op de respons die zijn ontvangen van de server, daarmee is gelijk de werking van zowel server, client als protocol aangetoond. Alle methoden worden daarnaast van de dao's aangeroepen door server, waardoor de werking van de database ook aangetoond is. Er wordt op elke parametertype getest, ook al heeft de server hiervoor geen bijbehorende query. In dat laatste geval wordt er een error ontvangen door de client en een lege lijst teruggeven. Hieronder is een tabel met de verwachtte resultaten en de daadwerkelijk output voor de test.

H4.1 Gebruikertest

Parametertype	Verwacht resultaat	Geslaagd
None	1 gebruiker per vereniging (10)	V
Gebruiker	Gebruiker met key 1 (Archimedes)	V
Vereniging	Lege lijst	V
Aandeel	Lege lijst	V
Aanbieding	Lege lijst	V
VerGen	Lege lijst	V

Tabel 6 Gebruikertest

```
GebTest None:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]
Gebruiker [PRIMARYKEY=2, gebruikersnaam=StringProperty [value:
Balans], wachtwoord=StringProperty [value: Balans],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 100.0]]
…… ENZEVOORT ….
------------------
GebTest Gebruiker:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]
------------------
GebTest Vereniging:
------------------
GebTest Aandeel:
------------------
GebTest Aandeel:
------------------
GebTest VerGen:
```

H4.2 Verenigingtest

Parametertype	Verwacht resultaat	Geslaagd
None	10 verenigingen	V
Gebruiker	Lege lijst	V
Vereniging	Vereniging met key 1(Archimedes)	V
Aandeel	Lege lijst	V
Aanbieding	Lege lijst	V
VerGen	Lege lijst	

Tabel 7 Verenigingtest

```
VerTest Vereniging:
Vereniging [PRIMARYKEY=1, naam=StringProperty [value:
Archimedes]]
------------------
VerTest VerGen:
------------------
VerTest None:
Vereniging [PRIMARYKEY=1, naam=StringProperty [value:
Archimedes]]
Vereniging [PRIMARYKEY=2, naam=StringProperty [value: Balans]]
Vereniging [PRIMARYKEY=3, naam=StringProperty [value:
Construct]]
Vereniging [PRIMARYKEY=4, naam=StringProperty [value:
Cementi]]
Vereniging [PRIMARYKEY=5, naam=StringProperty [value: Connect
Terzake]]
Vereniging [PRIMARYKEY=6, naam=StringProperty [value:
IManage]]
Vereniging [PRIMARYKEY=7, naam=StringProperty [value: ISLINK]]
Vereniging [PRIMARYKEY=8, naam=StringProperty [value:
Syntaxis]]
Vereniging [PRIMARYKEY=9, naam=StringProperty [value:
TriasLo]]
Vereniging [PRIMARYKEY=10, naam=StringProperty [value: Watt]]
------------------
VerTest Aandeel:
------------------
VerTest Gebruiker:
------------------
VerTest Aanbieding:
------------------
```

H4.3 Aandeeltet

Parametertype	Verwacht resultaat	Geslaagd
None	500 per vereniging (10 hits)	V
Gebruiker	Aandelen van gebruiker met key 1 (voorzitter Archimedes)	V
Vereniging	Aandelen van vereniging met key 1 (Archimedes)	V
Aandeel	Aandelen met key 1(Archimedes)	V
Aanbieding	Lege lijst	V
VerGen	Aandelen van vereniging met key 1 van gebruiker met key 1 (voorzitter Archimedes)	

Tabel 8 Aandeeltest

```
AdeelTest Vereniging:
Aandeel [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 500011]]
------------------
AdeelTest None:
Aandeel [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 5000]]
Aandeel [PRIMARYKEY=2, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=2, gebruikersnaam=StringProperty [value:
Balans], wachtwoord=StringProperty [value: Balans],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 100.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=2, naam=StringProperty [value: Balans]]],
aantal=IntegerProperty [value: 5000]]
… ENZEVOORT …
------------------
AdeelTest Aandeel:
Aandeel [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 5000]]
------------------
```

```
AdeelTest VerGen:
Aandeel [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 5000]]
------------------
AdeelTest Aanbieding:
------------------
AdeelTest Gebruiker:
Aandeel [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 5000]]
------------------
```

H4.1 Aanbiedingtest

Parametertype	Verwacht resultaat	Geslaagd
None	1 aanbieding per vereniging (10, alle aandelen)	V
Gebruiker	Lege lijst	V
Vereniging	Aanbieding van vereniging met key 1(Archimedes)	V
Aandeel	Lege lijst	V
Aanbieding	Aanbieding met key 1(Archimedes)	V
VerGen	Lege lijst	V

Tabel 9 Aanbiedingtest

```
AbieTest None:
Aanbieding [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 4842], prijs=DoubleProperty
[value: 5.0]]
Aanbieding [PRIMARYKEY=2, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=2, gebruikersnaam=StringProperty [value:
Balans], wachtwoord=StringProperty [value: Balans],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 100.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=2, naam=StringProperty [value: Balans]]],
aantal=IntegerProperty [value: 5000], prijs=DoubleProperty
[value: 5.0]]
Aanbieding [PRIMARYKEY=3, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=3, gebruikersnaam=StringProperty [value:
Construct], wachtwoord=StringProperty [value: Construct],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 100.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=3, naam=StringProperty [value: Construct]]],
aantal=IntegerProperty [value: 5000], prijs=DoubleProperty
[value: 5.0]]
…. ENZEVOORT ….
------------------
AbieTest Gebruiker:
------------------
AbieTest Vereniging:
Aanbieding [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
```

```
aantal=IntegerProperty [value: 4842], prijs=DoubleProperty
[value: 5.0]]
------------------
AbieTest Aandeel:
------------------
AbieTest Aanbieding:
Aanbieding [PRIMARYKEY=1, gebruiker=ObjectProperty [value:
Gebruiker [PRIMARYKEY=1, gebruikersnaam=StringProperty [value:
Archimedes], wachtwoord=StringProperty [value: Archimedes],
naam=StringProperty [value: voorzitter], balans=DoubleProperty
[value: 625.0]]], vereniging=ObjectProperty [value: Vereniging
[PRIMARYKEY=1, naam=StringProperty [value: Archimedes]]],
aantal=IntegerProperty [value: 4842], prijs=DoubleProperty
[value: 5.0]]
------------------
AbieTest VerGen:
------------------
```