IN2805-B ST4 PROJECT

GROEP 4

Eindverslag

Erik Ammerlaan	4005341
Jan Elffers	4014340
Wilson Ko	4005686
Quinten Stokkink	4016270
Erwin Walraven	4013166

14 juni 2011

Voorwoord

Het eindverslag dat voor u ligt, is het resultaat van het ST4-project aan de Technische Universiteit Delft. Gedurende vijf maanden hebben wij gewerkt aan een gedistribueerd softwaresysteem voor het Molvanisch Genootschap van Boekhandelaren (MGB). Het project bestond uit twee onderwijsperiodes. In de eerste periode werd het systeem ontworpen. Dit resulteerde in requirements, architectuur, ontwerp, implementatieplan en testplan. In de tweede periode werd het systeem geïmplementeerd volgens de eerder gemaakte documenten. Dit verslag bevat een korte beschrijving van het ontwikkelde systeem en het ontwikkelproces, onze reflecties, planningsdocumenten en instructies om het systeem te installeren.

We willen de mensen bedanken die hebben bijgedragen aan het succes van dit project: de docent Martin Pinzger en de studentassistenten Thomas Schaap en Aad van Buuren.

Delft, juni 2011
Erik Ammerlaan
Jan Elffers
Wilson Ko
Quinten Stokkink
Erwin Walraven

Samenvatting

Dit document is onderdeel van het ST4-project aan de Technische Universiteit Delft. Gedurende dit project werd er een systeem ontwikkeld voor het Molvanisch Genootschap van Boekhandelaren(MGB). Het project duurde twee kwartalen, waarbij er in het eerste kwartaal software werd ontworpen en in het tweede kwartaal software werd geïmplementeerd.

Het systeem dat door ons is gebouwd, is bedoeld voor de boekhandelaren, beheerder van het MGB en de koerier die betrokken is bij het transporteren van de bestelde boeken. Met het systeem is het voor de beheerder mogelijk om het genootschap te beheren, daarnaast is het voor de handelaren o.a. mogelijk om elkaars voorraden in te zien en boeken bij elkaar te bestellen. De koerier kan zien waar hij zijn boeken moet ophalen en waar hij ze moet afleveren. Bij het bouwen van het systeem is gebruik gemaakt van onze eerder geschreven documenten, zoals de requirements-analyse en het architectural design.

Het systeem moest ook bepalen hoe de bestellingen geplaatst moesten worden zodanig dat de transportkosten het laagst waren. Het optimaliseren werd als volgt gedaan:

1. Verzamel bestellingslijst en voorraadlijst van alle boekhandelaars. Vraag de lijst met routes voor de komende week op van de koerier. Dit samen noemen we de invoer.
2. Voer de berekeningen uit op de invoer en sla de resulterende order-lijst op.
3. Stel boekhandelaars en koerier op de hoogte van aangemaakte orders.

Tijdens de implementatiefase werd gewerkt met wekelijkse sprints. Per sprint werden een of twee iteraties uitgevoerd, afhankelijk van de voortgang en de lengte van de iteraties. Aan het begin van iedere week werd een korte sprintmeeting georganiseerd waarbij ieder groepslid aanwezig was. Bij de sprintmeeting werd er gesproken over zaken als het gemaakte werk, de algehele voortgang en de planning. Deze manier van werken werd door de hele groep als prettig ervaren, omdat het voor structuur zorgde.

Bij het implementeren werkte iedereen zoveel mogelijk aan zijn eigen code, waardoor elk groepslid gespecialiseerd was in zijn eigen deel. Door deze manier van werken raakte iedereen geroutineerd in zijn eigen deel, waardoor de ontwikkeling van het systeem snel vorderde. Nadat de bugtracker was geïnstalleerd verliep het proces nog sneller, wat een grote bijdrage heeft geleverd aan het goede verloop van het project. Later ging het implementeren moeizamer vanwege het doorvoeren van een aantal veranderingen. Dit zorgde er ook voor dat het moeilijker werd om test-driven te implementeren. Nadat de veranderingen waren doorgevoerd verliep het proces weer soepel, waardoor we weer op schema zaten.

De veranderingen die waren doorgevoerd zorgden ervoor dat de verschillende lagen als objecten werden gerepresenteerd in plaats van klassen met alleen statische functies. Hierdoor representeren de klassen in de model package puur de business objecten en ze hebben geen functionaliteit meer om met de database te communiceren. Door deze veranderingen werd de geschreven code overzichtelijker, daarnaast werd het unit-testen veel makkelijker.

Over het algemeen kunnen we tevreden zijn met het groepsproces gedurende de twee periodes. De samenwerking verliep vanaf het begin goed en er traden geen grote problemen op. In het begin werd er wel onnodig veel overlegd, maar naarmate de tijd vorderde werd dit steeds minder.

Inhoudsopgave

Voorwoord **i**

Samenvatting **ii**

1 Inleiding **1**

2 Het opgeleverde systeem **2**
2.1 Algemeen . . . 2
2.2 Boekhandelapplicatie . . . 2
2.3 Koeriersapplicatie . . . 3
2.4 MGB-applicatie . . . 3
2.5 Server . . . 3

3 Ontwikkelproces **4**
3.1 Werkwijze . . . 4
3.2 Veranderingen in het ontwerp . . . 5
3.3 Optimalisering . . . 5
3.3.1 Relevante requirements . . . 6
3.3.2 Overzicht . . . 6
3.3.3 Oplosmethode . . . 6
3.3.4 Stap 1 . . . 7
3.3.5 Stap 2 . . . 8

4 Reflectie **10**
4.1 Erik Ammerlaan . . . 10
4.2 Jan Elffers . . . 10
4.3 Wilson Ko . . . 11
4.4 Quinten Stokkink . . . 11
4.5 Erwin Walraven . . . 12

Appendix A: Installatie-instructies **13**

Appendix B: Planning **19**

1 Inleiding

Dit document is onderdeel van het ST4-project aan de Technische Universiteit Delft. Gedurende dit project werd er een systeem ontwikkeld voor het Molvanisch Genootschap van Boekhandelaren (MGB). Tijdens de eerste periode van dit project zijn er requirements voor het systeem opgesteld. Ook is er een ontwerp gemaakt van de architectuur, een technisch ontwerp en een implementatie- en testplan. Gedurende de tweede periode werd het ontworpen systeem geïmplementeerd. In dit document wordt teruggeblikt op het project en worden wijzigingen ten opzichte van eerdere documenten gedocumenteerd.

In hoofdstuk 2 wordt een overzicht gegeven van de ontwikkelde applicaties. Hierbij worden voornamelijk de highlights van iedere applicatie besproken. Hoofdstuk 3 bespreekt de werkwijze, veranderingen in het ontwerp en de optimaliseringsproblemen. In hoofdstuk 4 wordt gereflecteerd op het project, zowel algemeen als individueel. Installatie-instructies en planningsdocumenten zijn bijgevoegd als appendices.

2 Het opgeleverde systeem

2.1 Algemeen

Het systeem dat door ons is gebouwd, is bedoeld voor de boekhandelaren, beheerder van het MGB en de koerier die betrokken is bij het transporteren van de bestelde boeken. Met het systeem is het voor de beheerder mogelijk om het genootschap te beheren (d.w.z. de winkels beheren die lid zijn van de MGB). Daarnaast is het voor de handelaren o.a. mogelijk om elkaars voorraden in te zien en boeken bij elkaar te bestellen. De koerier kan zien waar hij zijn boeken moet ophalen en waar hij ze moet afleveren.

Het systeem is geïmplementeerd volgens de architectuur zoals behandeld in het ontwerp van de architectuur: er zijn meerdere gebruikers (clients) die met elkaar communiceren via een server. Het voldoet aan de eisen zoals genoemd in de requirements-analyse. Het is dus mogelijk om alle gewenste scenario's uit de must en should haves van het implementatieplan uit te voeren. Daarbij zijn er tijdens het implementeren nieuwe functies bij gekomen om de hoofdfuncties te ondersteunen, denk hierbij aan functies die het gebruikersgemak van de applicatie verhogen (zoals waarschuwingsschermen en controle tegen netwerkstoringen).

2.2 Boekhandelapplicatie

Wanneer de boekhandelapplicatie wordt gestart, wordt er gevraagd naar een gebruikersnaam en een wachtwoord. De functies die vervolgens aangeboden worden in het hoofdscherm zijn afhankelijk van het type gebruiker dat is ingelogd. Zo hebben boekhandelaren meer rechten dan hun medewerkers (req. 7). Boekhandelaren kunnen, eenmaal ingelogd, nieuwe gebruikers toevoegen, gebruikers verwijderen of gebruikers aanpassen (req. 8). Bij het aanpassen van andere gebruikers is het alleen mogelijk om het type van de gebruiker te veranderen; het wachtwoord en de gebruikersnaam kunnen alleen veranderd worden door de eigenaar van de account. Bij het verwijderen is het natuurlijk niet mogelijk om je eigen account te verwijderen, andersom is het bij het toevoegen niet mogelijk om twee accounts met dezelfde gebruikersnamen toe te voegen. In het hoofdscherm is het mogelijk om als gebruiker uit te loggen.

De boekbeheerder kan ook zijn eigen bestellingen en transacties beheren (req. 4, 5 en 6). Bij het bestellen van een boek wordt er een verzoek naar de server gedaan, die kijkt of het gezochte boek ergens beschikbaar is. Er wordt dan een uitgaande bestelling geplaatst in de lijst van geplaatste bestellingen. De server berekent vervolgens de optimale plaatsing van bestellingen en wijst dan bestellingen toe aan de betrokken winkels en de koerier. Op de lijst van binnengekomen bestellingen kan een winkel zien welke boeken er bij hem besteld zijn, zodat hij weet dat hij de koerier die het boek komt ophalen, binnenkort kan verwachten. Het is echter niet mogelijk om bestellingen te accepteren of te weigeren, aangezien dat besloten wordt door de server.

Zoals al eerder is genoemd is het voor de boekhandelaar mogelijk om zijn boeken te beheren (req. 1). Denk hierbij aan het toevoegen, aanpassen en verwijderen van boeken. Bij het toevoegen van boeken wordt er gecheckt of het boek wel echt bestaat door het ISBN te controleren met de checkdigit. Daarbij is het niet mogelijk om twee dezelfde boeken toe te voegen. Bij het toevoegen of wijzigen kan ook aangegeven worden hoeveel exemplaren je van een bepaald boek wilt behouden. Zo kan voorkomen worden dat alle exemplaren door andere winkels worden besteld, terwijl de boekhandelaar nog exemplaren wilde behouden voor zijn eigen verkoop.

De applicatie is overigens gemaakt om overweg te kunnen met netwerkstoringen. Wanneer er een netwerkstoring optreedt, worden de functies die een netwerkverbinding nodig hebben uitgeschakeld; het is dan alleen mogelijk om lokale functies te gebruiken zoals het beheren van boeken (req.

14). Wanneer er weer een verbinding is, worden de uitgeschakelde functies automatisch ingeschakeld.

2.3 Koeriersapplicatie

In de koeriersapplicatie kunnen eenvoudig routes langs winkels worden toegevoegd, die op bepaalde dagen gereden worden (req. 11). Deze routes worden automatisch door de server gebruikt bij het optimaliseren van bestellingen. De koerier kan een overzicht opvragen van bestellingen die hij moet ophalen en afleveren, compleet met een overzicht van welke routes gereden moeten worden en bij welke winkels op kruisende routes een boek eventueel moet blijven liggen (req. 10). De koerier kan bestellingen afvinken, wanneer ze verstuurd zijn. Deze orderstatus kan ook door de betrokken boekhandelaars worden bekeken. Tot slot is er ook ondersteuning voor gebruikersbeheer.

2.4 MGB-applicatie

In de applicatie die voor het MGB zelf is ontwikkeld kan men de winkels bijhouden die zijn aangesloten bij het MGB (req. 9). Van elke winkel wordt een fysiek adres en IP-adres opgeslagen. Ook kan met deze applicatie het IP-adres van de koeriersdienst worden ingesteld. Al deze IP-adressen zijn nodig voor de server, die op die manier met de juiste winkels kan communiceren. De MGB-applicatie is een gebruiksvriendelijke manier om te reguleren wie van het MGB-systeem gebruik mag maken.

2.5 Server

De server verzorgt de communicatie tussen alle applicaties. Verder neemt de server het optimaliseren op zich; meer hierover in §3.3. De server beschikt over een GUI waarmee de server gestart en gestopt kan worden. Daarnaast bevat de GUI grafieken waarop het geheugengebruik en het aantal lopende threads is uitgezet. Indien gewenst kan de server ook zonder GUI worden gestart via de commandoregel.

3 Ontwikkelproces

Dit hoofdstuk beschrijft het ontwikkelproces. Eerst wordt stilgestaan bij de gehanteerde manier van werken. Vervolgens komen de veranderingen in het ontwerp aan de orde. Ten slotte worden de optimaliseringsproblemen besproken en de methode waarmee deze zijn opgelost.

3.1 Werkwijze

Tijdens de implementatiefase werd gewerkt met wekelijkse sprints. Per sprint werden een of twee iteraties uitgevoerd, afhankelijk van de voortgang en de lengte van de iteraties. Aan het begin van iedere week werd een korte sprintmeeting georganiseerd waarbij ieder groepslid aanwezig was. Hierbij werd de sprint van de vorige week nabesproken en werd een planning gemaakt voor de komende sprint. Tijdens de sprints werd de planning regelmatig bijgewerkt, zodat ieder groepslid kon zien wat de algehele voortgang was. Deze aanpak hebben wij als prettig ervaren, omdat het voor structuur zorgde en het voor iedereen duidelijk was wat er moest gebeuren.

Ieder groepslid heeft zich zo veel mogelijk beziggehouden met dezelfde onderdelen van de code. Hierdoor was iedereen dus min of meer gespecialiseerd in zijn deel. De interfaces werden bijvoorbeeld steeds gemaakt door dezelfde personen, net als het IO-gedeelte en de databaseklassen. Dit was een goede manier van werken, omdat iedereen geroutineerd raakte in zijn onderdeel. Door het gebruik van deze aanpak verliep de ontwikkeling naarmate het kwartaal vorderde ook steeds beter. We hadden er ook voor kunnen kiezen om groepsleden allround inzetbaar te laten zijn, maar vanwege de strakke tijdsplanning vonden wij dit niet verstandig.

Gedurende de eerste twee sprints werd er gewerkt zonder bug tracker en verliep de communicatie buiten de projectsessies via e-mail. In het begin verliep dit goed, omdat de hoeveelheid informatie die werd verstuurd beperkt was en het overzicht behouden bleef. Naarmate er meer kleine problemen ontstonden, werden de conversaties langer en verloren we het overzicht. De efficiëntie ging hierdoor achteruit. In de derde week werd besloten om een online bug tracker te installeren, zodat problemen gestructureerd gemeld en besproken konden worden. Dit heeft een grote bijdrage geleverd aan het goede verloop van het project. Iedereen controleerde de bugtracker regelmatig en problemen werden hierdoor sneller opgelost, vooral wanneer er buiten de projectsessies thuis werd gewerkt.

Na de eerste feedback van de docent werd besloten om enkele wijzigingen door te voeren in het technische ontwerp. Dit was erg ingrijpend, omdat bijna alle code afhankelijkheden had met de code die aangepast moest worden. De wijzigingen werden uitgevoerd door één persoon in een aparte branch directory. Hierdoor werd de overlast voor de andere groepsleden beperkt en kon er gewoon doorgewerkt worden. Omdat er tijdens het maken van de aanpassingen ook werd doorgewerkt in de bestaande code, was het mergen tijdsintensief. Achteraf gezien hadden we dit beter moeten aanpakken, omdat we hierdoor enigszins achter kwamen te lopen op schema. Uiteindelijk hebben we deze tijd kunnen inhalen, dus het heeft geen negatieve invloed gehad op het eindresultaat.

Vooraf was het de bedoeling om zo veel mogelijk testgestuurd te ontwikkelen. In de praktijk viel dit tegen omdat we te maken kregen met enkele wijzigingen in het technisch ontwerp. Ook bleek dat het uitgebreide testen, zoals besproken in het testplan, heel erg tijdrovend was. Omdat de onderliggende functionaliteit van elke databaseklasse hetzelfde is, hebben we uiteindelijk één van deze klassen getest. De requesthandlers die worden gebruikt op de server zijn getest, omdat deze een cruciale rol vervullen bij de werking van het systeem. Requesthandlers die slechts een aanroep naar een databaseklasse doen, zijn niet onderworpen aan een test, omdat de kans op fouten hier zeer klein is. De controllerklassen maken gebruik van IO- en databasefunctionaliteiten. De manier waarop dit gebeurt, is op veel plaatsen echter nagenoeg hetzelfde, dus dit is niet voor alle klassen uitvoerig getest. Om scenario's te kunnen testen hebben we de software uitgerold op

meerdere systemen. Op deze manier zijn praktijksituaties uitgeprobeerd en kwamen er nog enkele problemen aan het licht. Verder hebben we gebruikgemaakt van asserties om fouten eenvoudig op te sporen en de debugtijd te beperken.

3.2 Veranderingen in het ontwerp

- De grote refactor-wijziging die we hebben doorgevoerd, betrof het wijzigen van de representatiewijze van de verschillende lagen. We hebben besloten ze als objecten te representeren en niet als klassen met alleen statische functies. Aanleiding was de opmerking van de docent bij de demo halverwege, een oorzaak was ook dat de source steeds minder beheers- en dus debugbaar werd. We hebben dit opgelost door uit het design af te leiden wat de dependencies van de componenten waren en deze via de constructor mee te geven. De controllerlaag krijgt bij constructie bijvoorbeeld pointers naar de datalaag en I/O-laag mee. De opbouw en afhankelijkheden blijven hetzelfde. Een ander voordeel van deze aanpak is dat unit-testen hiermee makkelijker is.
- De klassen in de `model` package (en subpackages) representeren nu puur de business-objecten en hebben geen functionaliteit om met de database te communiceren. De manier om een business-object te updaten (in een van de client-applicaties), is de bijbehorende update-functie in de controller aan te roepen. Deze roept dan bijv. de datalaag of IO-laag aan.

3.3 Optimalisering

Deze paragraaf beschrijft op welke manier optimalisatie is toegepast in de implementatie. Eerst wordt uitgelegd wat onder het optimalisatieprobleem verstaan wordt, daarna hoe we dit probleem onderverdeeld hebben in subproblemen en ten slotte de technische oplossing van deze subproblemen.

We definiëren eerst een aantal termen die we verderop gebruiken. Hier herdefiniëren we geen business-objects, alleen termen die we verderop gebruiken bij het uitleggen van het optimalisatieproces.

- Een **tijdelijke bestelling** is de combinatie van een orderID, een winkelID en een ISBN. De **bestellingslijst** van een winkel is de lijst tijdelijke bestellingen die de winkel heeft gedaan.
- De **voorraadlijst** van een winkel is de lijst met aantallen boeken die een winkel wil afstaan.
- Een **order** is de combinatie van een orderID, een ISBN en twee winkelIDs: die van zender en ontvanger.
- Een **route** is een lijst winkels, gekoppeld aan een dag van de week (de dag dat hij wordt gereden). Elke route wordt dus op maar één dag per week gereden. De representatie voor de koerier is anders, maar voor de optimalisator is het zojuist beschreven formaat beter. We splitsen elke koerier-route dus in een aantal één-dag-routes vooraf aan het optimalisatieproces.
- De **supply** van een winkel voor een bepaald ISBN, is het aantal boeken dat hij kan afstaan.
- De **demand** van een winkel voor een bepaald ISBN, is het aantal boeken dat hij wil ontvangen.

3.3.1 Relevante requirements

- Een order moet snel geleverd kunnen worden. Er is gekozen voor een maximale levertijd van 7 dagen.
- (requirement 12) De kosten van het MGB moeten, onder de vorige voorwaarde, zo laag mogelijk blijven.
- Optimalisatie moet aan het einde van elke werkdag *centraal* (i.p.v. *gedistribueerd*) plaatsvinden. Hierover is niets gezegd in het requirements-document, maar dit volgt uit later overleg met de klant.

3.3.2 Overzicht

Het optimalisatieproces bestaat uit de volgende stappen:

1. Verzamel bestellingslijst en voorraadlijst van alle boekhandelaars. Vraag de lijst met routes voor de komende week op van de koerier. Dit samen noemen we de invoer.
2. Voer het optimalisatieproces uit op de invoer en sla de resulterende order-lijst op.
3. Stel boekhandelaars en koerier op de hoogte van aangemaakte orders.

De rest van sectie 3.3 zal gewijd zijn aan stap 2.

3.3.3 Oplosmethode

In het interview met de koerier werd duidelijk dat de kosten van een route per boek constant zijn (7 sjekl). Dus, de kosten van een order in sjekl zijn gelijk aan zeven maal het aantal afgelegde routes. De totale kosten van alle orders zijn dus de som van al deze kosten. We willen de totale kosten minimaliseren en zo veel mogelijk orders door laten gaan. Om het probleem op te kunnen lossen moet eerst nog het begrip *geldige orderverzameling* gedefinieerd worden:

Een verzameling orders is **geldig**, als:

1. Elke order een levertijd van maximaal 7 dagen heeft.
2. Voor elke winkel en elk ISBN geldt dat de winkel niet meer dan zijn supply van dat ISBN levert.
3. Voor elke winkel en elk ISBN geldt dat de winkel niet meer dan zijn demand van dat ISBN ontvangt.

Nu kan het optimaliseringsprobleem als volgt worden geformuleerd:

"Vind een orderverzameling met minimale totaalkosten uit alle geldige orderverzamelingen van maximale grootte."

We zullen nu onze methode van oplossen uitleggen. De eerste simplificatie is, dat we het probleem afzonderlijk kunnen oplossen voor elk ISBN. Voor het probleem voor een enkel ISBN hebben we de volgende invoer:

1. Van elke boekhandelaar, zijn supply (hoeveel boeken hij kan afstaan) of zijn demand (hoeveel hij er wil inkopen).
2. Van de koerier onthouden we de lijst met routes voor de komende week (hetzelfde voor elk ISBN).

We kunnen het resulterende probleem in twee stappen oplossen:

1. Bereken voor elk geordend paar winkels (verzender, ontvanger) de minimale leverkosten en een routeschema met die kosten (als dat er niet is, sla dat op). Hoe het routeschema uit alle routeschema's precies gekozen wordt, wordt beschreven in de volgende sectie.
2. Los het volgende probleem op: gegeven de supply/demand van elke winkel voor dit boek, en de leverkosten voor elk geordend paar winkels, bereken de minimale totale leverkosten om een zo groot mogelijk aantal bestellingen door te laten gaan.

Het resultaat hiervan kunnen we koppelen met de bestellingslijst uit de invoer, waarmee we de orders kunnen maken. Omdat de uitvoer van stap 1 onafhankelijk is van het ISBN, zullen we stap 1 slechts één keer uitvoeren per uitvoering van optimalisering. De oplossingen van de twee stappen worden hieronder nader uitgelegd.

3.3.4 Stap 1

Als invoer krijgen we hier de huidige dag van de week, en het overzicht van routes die de koerier aankomende 7 dagen (vanaf morgen) zal rijden. De uitvoer zal zijn, voor elk geordend paar winkels (verzender, ontvanger) in het MGB-netwerk, een routeschema van minimale kosten en levertijd maximaal 7 dagen van verzender naar ontvanger. Als zo een routeschema niet bestaat moet dit ook worden aangegeven. Er kunnen meerdere routeschema's zijn van minimale kosten. In dat geval zal het algoritme er een kiezen met minimale levertijd. In het vervolg zal gedetailleerd worden uitgelegd hoe we precies deze optimale routeschema's berekenen.

Technische beschrijving

We modelleren het probleem als een kortste paden-probleem in een gerichte, gewogen graaf G. De beschrijving van de graaf is te vergelijken met een in– en uitstapsysteem zoals in het openbaar vervoer, met haltes.

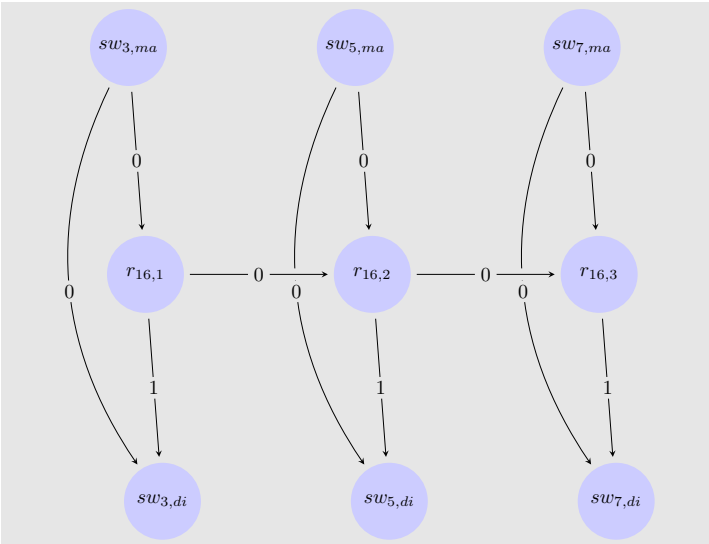
We definiëren twee typen vertices, switch– en route-vertices:

- Een switch-vertex $sw_{w,d}$ hoort bij een winkel w en een dag $d \in [0 \dots 7]$. Vanaf switch-vertices kan men instappen voor routes.
- Een route-vertex hoort bij een winkel en een route.
  Voor elke route $R = [store_1, store_2, \dots, store_{|R|}]$ zijn er $|R|$ route-vertices, voor elke winkel die op de route ligt één.

De vertex set van de graaf is opgebouwd uit deze twee typen vertices. We willen de graaf als volgt opbouwen: "elk pad van een switch-vertex naar een andere switch-vertex heeft gewicht precies gelijk aan het aantal routes dat gevolgd is". We beschrijven nu hoe de graaf is opgebouwd.

- De switch-vertices in de graaf zijn als volgt: voor elke winkel zijn er $7 + 1$ switch-vertices, voor elke dag $d \in [0 \dots 7]$ één. Voor elke switch-vertex (w, d), als $d < 7$ dan voeg een edge toe met gewicht 0 naar switch-vertex $(store, dag + 1)$. Dit komt overeen met het één dag blijven liggen van een pakket op een bepaalde locatie.
- De route-vertices van de graaf zijn als volgt: voor elke route r op dag $0 \leq d < 7$, maak $|r|$ route-vertices, voor elke winkel die op de route ligt één. Maak nu twee typen edges voor elke winkel w op de route:
  - Een edge van switch-vertex $sw_{w,d}$ naar de route-vertex op deze route behorende bij die winkel (gewicht 0).
  - Een edge van de route-vertex op deze route behorende bij die winkel, naar de switch-vertex $sw_{w,d+1}$. Intuïtief betekent dit dat je betaalt voor de route bij het uitstappen.

Daarnaast zullen er edges van de route-vertices naar hun volgende winkel zijn (behalve voor de laatste winkel op de route).



Figuur 3.1: *Overzicht van de manier waarop routes de switch-vertices verbinden. Route 16 rijdt op maandag en gaat langs winkels [3, 5, 7].*

Zo komt het gewicht van een pad in de resulterende graaf overeen met het aantal afgelegde routes. We kunnen in deze graaf kortste paden uitrekenen en hieruit de routes halen die tot de optima leiden, extracten. Nu we de graaf opgebouwd hebben doen we het volgende voor elke winkel w in het MGB-netwerk:

1. Draai het single-source shortest path algoritme vanaf switch-vertex $sw_{w,0}$ (behorende bij deze winkel en dag 0).
2. Voor elke andere winkel w': Als $sw_{w',0}$ niet bereikbaar is, sla geen route op (er is geen routeschema binnen 7 dagen). Anders, vind de switch-vertex *met de laagst mogelijke dag die bereikbaar is en het minimale aantal routes nodig heeft over alle switch-vertices van deze winkel die bereikbaar zijn.* Haal het kortste pad van s naar deze switch-vertex op uit de graaf en maak hieruit het routeschema.

Het berekenen van de kortste paden vanaf een vaste source vertex is geïmplementeerd met een gemodificeerde versie van breadth-first search in lineaire tijd en ruimte in de grootte van de graaf (aantal vertices plus aantal edges). De looptijd totaal is $O(n \cdot (7n + \sum_{r \in ROUTES} |r|))$ (n is aantal stores).

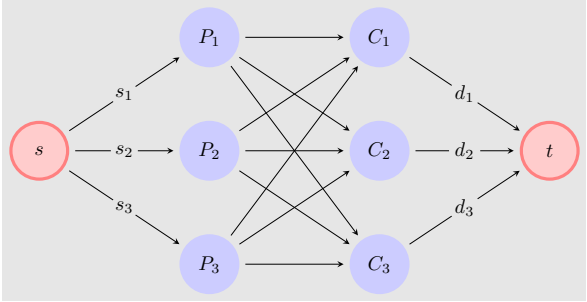
3.3.5 Stap 2

Als invoer krijgen we hier een aantal winkels met positieve supply (*producers*), en een aantal winkels met positieve demand (*consumers*). Verder krijgen we uit stap 1 een kostenmatrix van transportkosten per order. De uitvoer moet dus een lijst orders zijn die de minimale totale leverkosten heeft onder alle orderlijsten van maximale grootte.

Technische beschrijving

We hebben het probleem als volgt via flow theorie opgelost: definieer $P = \{store_i \mid supply(store_i) > 0\}$ en $C = \{store_i \mid supply(i) < 0\}$ (de producers en consumers). Maak een gerichte graaf met vertices $\{s\} \cup P \cup C \cup \{t\}$ (source, producers en consumers, sink). Voeg de volgende edges toe:

- s naar elke producer $p \in P$ (capaciteit $supply(p)$, kosten 0)
- voor elke consumer $c \in C$ naar t (capaciteit $demand(c)$, kosten 0)
- voor alle producers $p \in P$ naar alle consumers $c \in C$ waarnaar een geldig routeschema bestaat (capaciteit ∞, kosten zijn de transportkosten $p \to c$).



Figuur 3.2: *Voorbeeld flowgraph met 3 producers en 3 consumers.*

Het is duidelijk hoe elke geldige $s-t$ flow correspondeert met een keuze van orders. Dus, we zoeken een **maximale flow met minimale kosten** (een goedkoopste maximale flow) in deze graaf. Dit is een standaardprobleem in de lineaire optimalisering. We hebben het dan ook opgelost met een standaardalgoritme, het successive shortest-path algoritme. De looptijd hiervan is maximaal $O(n^3)$ per order in de output, waarbij n het totale aantal winkels in dit transportatieprobleem is. We hebben niet geëxperimenteerd of het algoritme in de praktijk beter schaalt dan deze bovengrens.

4 Reflectie

Over het algemeen kunnen we tevreden zijn met het groepsproces gedurende de twee periodes. De samenwerking verliep vanaf het begin goed en er traden geen grote problemen op. Aanvankelijk duurde overleg vaak onnodig lang door het bespreken van veel details. Later in het project verliep dit beter en duurden de besprekingen minder lang. Doordat dit project, in tegenstelling tot vorige projecten, twee onderwijsperiodes duurde raakten we beter op elkaar ingewerkt. Hierdoor wisten we ook beter wat we aan elkaar hadden en wat we van elkaar konden verwachten. Iedereen heeft zich gehouden aan de gemaakte afspraken en alles was altijd op tijd klaar. Dit heeft ervoor gezorgd dat we zonder stress konden werken en de teamspirit goed was.

4.1 Erik Ammerlaan

Ik ben erg tevreden met hoe we dit project voltooid hebben. Vanaf het begin hebben we ons goed aan de planning gehouden en ook veel werk buiten de ingeroosterde projecturen gedaan. Daardoor zijn we nooit in tijdnood gekomen en konden we het de laatste week zelfs rustiger aandoen, omdat het systeem al klaar was. Verder heeft iedereen zijn best gedaan om de software te perfectioneren en is er telkens kritisch gekeken naar hoe we dingen gingen implementeren. Het was ook fijn dat we al snel een werkende client-serverarchitectuur op poten hadden gezet, waar we op konden bouwen.

In het derde kwartaal heb ik geleerd op welke manier je met een klant een ontwerp kunt bespreken. Ik realiseer me nu hoe belangrijk het is om o.a. een goede lijst requirements en een MoSCoW-document te hebben, waar zowel de klant als de programmeur het over eens zijn, zodat je daar altijd op terug kunt vallen. Van het programmeren heb ik weinig nieuws geleerd op het Unit-testen na. Het is handig om te weten hoe het xUnit-framework werkt, aangezien ik dat vast nog wel eens zal gebruiken. Hoewel ik verder weinig nieuwe programmeerkennis heb opgedaan, heb ik wel mijn best gedaan om kennis door te geven aan anderen.

Het was wel nieuw voor mij om met een team aan een softwareproject als dit te werken. In het begin ging nog flink wat tijd verloren, doordat teamleden elkaar op de hoogte moesten brengen van tegengekomen problemen. Uiteindelijk hebben we een bugtracker gebruikt en dat blijkt eigenlijk wel een onmisbaar gereedschap, zelfs voor een relatief kleine groep. Opvallend is dat bij elke iteratie vaak dezelfde personen weer dezelfde soort taken deden als de vorige iteratie. Dat was niet zo afgesproken, maar dat was een natuurlijk proces. Zo was uiteindelijk iedereen gespecialiseerd in een bepaald gedeelte van het systeem. Daarnaast is goede communicatie vanzelfsprekend erg belangrijk, mede omdat de code die je schrijft vaak afhangt van functies die anderen implementeren. Het moet daarom goed helder zijn wie wat van iemand anders nodig heeft, zodat je op tijd kunt integreren. Dit ging niet altijd vlekkeloos, maar wel steeds beter.

Al met al vond ik samenwerking goed verlopen en hebben we een mooi product af kunnen leveren!

4.2 Jan Elffers

Ik denk dat we een goed product hebben gemaakt en dat we terug kunnen kijken op een goed proces. Vooral tijdens de implementatiefase was er veel werk te verrichten en iedereen had zijn taken meestal op tijd voltooid. Zo konden we ons iteratieschema, dat we aan het begin van de implementatiefase hebben gemaakt, blijven aanhouden. Hierdoor konden we ook fatsoenlijke tussenproducten laten zien tijdens de demo's.

Ondanks dat het design de basis vormt van het implementatieproces, was het programmeren van bijv. de GUI, niet triviaal. Daarmee was naast inzet de achtergrondkennis van de teamleden dus belangrijk. Ik denk dat ik wel veel heb geleerd, maar waarschijnlijk vooral over software

development in Java en in het algemeen. Ik heb design patterns bestudeerd om een keuze te maken hoe we onze code het best konden structureren. Daarnaast heb ik geleerd hoe je code eenvoudiger debugbaar kunt maken: veel overleggen met de anderen over programmeerkeuzes en development best practices, componenten met elkaar laten communiceren via interfaces, en inhoudelijk: warnings niet negeren, gebruikmaken van een bug detector zoals `FindBugs`, Java features zoals Exceptions gebruiken om optimaal duidelijk te maken wat jouw code doet.

Uiteindelijk ben ik tevreden over het ontwikkelde systeem en het proces waarmee dit gemaakt is.

4.3 Wilson Ko

Ik vond het project erg goed gaan, iedereen deed actief mee en er werd door iedereen veel tijd gestoken in het project. Hierdoor werkten we altijd mooi op schema en hoefden we nooit te stressen om iets af te krijgen. Bovendien was het geleverde werk altijd van goede kwaliteit, omdat iedereen zijn best deed. Daarnaast werd er goed samengewerkt in de groep. Er werd veel gecommuniceerd en overlegd waardoor het project erg soepel verliep. Verder vond ik het proces in sprints erg handig; het zorgde ervoor dat we allemaal wisten wat er gedaan moest worden en wat er van ons verwacht werd.

Tijdens het project heb ik heel veel geleerd, niet alleen bij het ontwerpen, maar ook bij het programmeren. Bij zo een groot project vind ik het altijd moeilijk om te starten, want ik weet nooit waar ik moet beginnen. Van dit project heb ik geleerd dat het heel handig is om dingen van tevoren uit te zoeken en te documenteren, zoals de requirements, software-architectuur etc.. Ik heb gemerkt dat het veel makkelijker wordt om code te schrijven als dit soort dingen van tevoren gedocumenteerd worden, want dan heb je een veel beter beeld van wat er geprogrammeerd moet worden. Daarnaast heb ik bij dit project heel veel programmeerervaring en -kennis opgedaan, wat ik hiervoor nog een beetje miste omdat ik over het algemeen niet veel programmeer.

Ik vond het project naast leerzaam ook erg leuk. Het was voor mij de eerste keer om zo een groot systeem te bouwen van 'scratch' en het was ook nieuw om met zoveel mensen samen te werken aan zo een groot project. Het leuke was ook dat er heel veel verschillende aspecten zijn bij zo een systeem, bijv. servercommunicatie, databeheer, user interface etc.. Het was erg interessant om aan deze aspecten te werken en deze te integreren tot een geheel.

4.4 Quinten Stokkink

Het project is zonder grote tijdsdruk afgerond en is daarmee naar mijn mening een groot succes. Alle ingeplande functionaliteiten zijn tijdig (of met zeer kleine uitloop) afgerond. Er was zelfs tijd om extra functionaliteit te implementeren. Over het algemeen was de originele inschatting van het project in de design-fase goed in overeenstemming met wat er daadwerkelijk van terecht is gekomen tijdens de implementatiefase.

Dit was niet mijn eerste grote project in Java, maar wel het eerste in een teamverband. Ook is het tijdsbestek van dit project korter dan wat ik zelf zou maken. Deze twee punten brachten dus de problemen van communicatie en efficiënt werken met teamleden met zich mee. De communicatie tussen teamleden werd naarmate het project liep steeds soepeler. Ook werd het samenwerken in het team steeds vloeiender. Daar waar eerst wat meer apart gewerkt werd, was aan het einde iedereen – zonder problemen – door elkaar aan het werken.

Qua technische vaardigheden heb ik meer geleerd van de ontwerpfase dan van de implementatie fase. Mijn ontwerpvaardigheden waren lichtelijk weggezakt en deze heb ik tijdens de eerste fase weer helemaal op kunnen halen. Vooral het modelleren van dataopslag was voor mij een nieuw

punt. Tijdens de implementatiefase heb ik een paar trivialiteiten van Java uitgevonden, maar daar was niet zo veel meer om te leren.

4.5 Erwin Walraven

Ik heb het ST4-project als leerzaam ervaren en vond het leuk om deel te nemen. Vooral het werken in teamverband voor een langere periode is mij goed bevallen. Bij eerdere projecten was de beschikbare tijd meestal beperkt en werd het pas leuk op het moment dat het project bijna was afgelopen. Nu was er genoeg tijd beschikbaar waar we goed gebruik van hebben gemaakt.

De eerste periode van het project is voor mij het meest leerzaam geweest. Vooral het denken in termen van businessobjecten was nieuw en dit kostte in het begin soms wat moeite. Het maken van het technische ontwerp vond ik in deze periode het leukste om te doen, omdat er rekening gehouden moest worden met alle aspecten van het systeem. Uiteindelijk zijn enkele details van het ontwerp aangepast tijdens het implementeren. Dit is echter alleen maar leerzaam geweest en een nuttige ervaring voor een volgende keer.

Wat programmeren betreft heb ik in de tweede periode niet zo veel nieuwe dingen geleerd. Het was wel leerzaam om met meerdere personen tegelijk aan een groot systeem te werken. Het was dus een uitdaging om ervoor te zorgen dat alles goed verliep. Vooral de gestructureerde manier van werken bleek nuttig te zijn en ik denk dat ik hier veel van opgestoken heb. Ook de goede samenstelling van de groep heeft hierbij een rol gespeeld omdat we geen energie hebben gestoken in irrelevante dingen.

Appendix A: Installatie-instructies

Installatie-instructies voor boekhandelaren

Deze applicatie draait op Java. Wellicht moet u van te voren een computer expert inschakelen om dit werkend te krijgen up uw systeem.
Voordat u begint met het gebruiken van de applicatie moet u zorgen dat u de volgende gegevens heeft:

1. Een database-adres (bijvoorbeeld: sql.ewi.tudelft.nl/st4g4_boekw1)
2. Een database-gebruikersnaam (bijvoorbeeld: st4g4_boekw1)
3. Een database-wachtwoord (bijvoorbeeld: wacht_woord1)
4. Het IP-adres van de server (bijvoorbeeld: 83.102.159.5)
5. De poort van de server (bijvoorbeeld: 4321)
6. Uw eigen poort (bijvoorbeeld: 1234)
7. Uw eigen ID (bijvoorbeeld: 519)

Een database-adres
Dit is uw eigen database. Het adres zelf is het pad naar de database. Indien u dit pad niet weet moet u dit uw database ontwerper vragen.

Een database-gebruikersnaam
De gebruikersnaam die bij uw eigen database kwam. Indien u dit niet weet, moet u dit uw database-ontwerper vragen.

Een database-wachtwoord
Het wachtwoord voor de gebruikersnaam voor uw eigen database. Indien u dit niet weet, moet u dit uw database-ontwerper vragen.

Het ip adres van de server
Dit is een IPv4-adres van de server. Dit adres wordt u aangeleverd door de MGB. Indien u dit IP niet weet, moet u contact opnemen met de mgb.

De poort van de server
Dit is een communicatiepoort van de server. Deze poort wordt u aangeleverd door de MGB. Indien u deze poort niet weet, moet u contact opnemen met de mgb.

Uw eigen poort
Dit is een poort waarover communicatie geschiedt. Over het algemeen zijn de poorten van 8192

tot 32768 beschikbaar. Deze waarde is arbitrair maar kan bij toeval niet werken.

Uw eigen ID
Dit is de ID die u is toegewezen door de MGB. Deze ID wordt u aangeleverd door de MGB. Indien u uw ID niet weet, moet u contact opnemen met de mgb.

Als u al deze gegevens heeft, kunt u uw software op uw computer zetten.
U krijgt twee bestanden. Zowel een applicatie `mgbboekhandelaar.jar` als `config-booktrader-client` zijn meegeleverd.
De informatie in `config-booktrader-client` zou gelijk moeten zijn aan de gegevens die u hierboven heeft bepaald.

Wanneer u gevalideerd heeft dat de instellingen in `config-booktrader-client` goed zijn, kunt u simpelweg `mgbboekhandelaar.jar` starten en aan de slag met de MGB-boekhandelarenapplicatie.

Installatie-instructies voor MGB-beheerders

Deze applicatie draait op Java. Wellicht moet dit nog geïnstalleerd worden op uw systeem.
Voordat u begint met het gebruiken van de applicatie, moet u zorgen dat u de volgende gegevens heeft:

1. Het IP-adres van de server (bijvoorbeeld: 83.102.159.5)
2. De poort van de server (bijvoorbeeld: 4321)
3. Uw eigen poort (bijvoorbeeld: 1234)
4. Uw eigen ID (bijvoorbeeld: 519)

Het IP-adres van de server
Dit is een IPv4-adres van de server. Dit adres wordt u aangeleverd door de MGB. Indien u dit IP niet weet, moet u contact opnemen met de MGB.

De poort van de server
Dit is een communicatiepoort van de server. Deze poort wordt u aangeleverd door de MGB. Indien u deze poort niet weet, moet u contact opnemen met de MGB.

Uw eigen poort
Dit is een poort waarover communicatie geschiedt. Over het algemeen zijn de poorten van 8192 tot 32768 beschikbaar. Deze waarde is arbitrair maar kan bij toeval niet werken.

Uw eigen ID
Dit is de ID die u is toegewezen door de MGB. Deze ID wordt u aangeleverd door de MGB. Indien u uw ID niet weet, moet u contact opnemen met de mgb.

Als u al deze gegevens heeft, kunt u uw software op uw computer zetten.
U krijgt twee bestanden. Zowel een applicatie `mgbadministratie.jar` als `config-mgb-client` zijn meegeleverd.
De informatie in `config-mgb-client` zou gelijk moeten zijn aan de gegevens die u hierboven heeft bepaald.

Wanneer u gevalideerd heeft dat de instellingen in `config-mgb-client` goed zijn, kunt u simpelweg `mgbadministratie.jar` starten en aan de slag met de MGB-applicatie.

Installatie-instructies voor de koerier

Deze applicatie draait op Java. Wellicht moet u van te voren een computerexpert inschakelen om dit werkend te krijgen up uw systeem.
Voordat u begint met het gebruiken van de applicatie, moet u zorgen dat u de volgende gegevens heeft:

1. Een database-adres (bijvoorbeeld: sql.ewi.tudelft.nl/st4g4_boekw1)
2. Een database-gebruikersnaam (bijvoorbeeld: st4g4_boekw1)
3. Een database-wachtwoord (bijvoorbeeld: wacht_woord1)
4. Het IP-adres van de server (bijvoorbeeld: 83.102.159.5)
5. De poort van de server (bijvoorbeeld: 4321)
6. Uw eigen poort (bijvoorbeeld: 1234)
7. Uw eigen ID (bijvoorbeeld: 519)

Een database-adres
Dit is uw eigen database. Het adres zelf is het pad naar de database. Indien u dit pad niet weet moet u dit uw database-ontwerper vragen.

Een database-gebruikersnaam
De gebruikersnaam die bij uw eigen database hoort. Indien u dit niet weet, moet u dit uw database-ontwerper vragen.

Een database-wachtwoord
Het wachtwoord voor de gebruikersnaam voor uw eigen database. Indien u dit niet weet, moet u dit uw database-ontwerper vragen.

Het IP-adres van de server
Dit is een IPv4-adres van de server. Dit adres wordt u aangeleverd door de MGB. Indien u dit IP niet weet, moet u contact opnemen met de mgb.

De poort van de server
Dit is een communicatiepoort van de server. Deze poort wordt u aangeleverd door de MGB. Indien u deze poort niet weet, moet u contact opnemen met de mgb.

Uw eigen poort
Dit is een poort waarover communicatie geschiedt. Over het algemeen zijn de poorten van 8192 tot 32768 beschikbaar. Deze waarde is arbitrair maar kan bij toeval niet werken.

Uw eigen ID
Dit is de ID die u is toegewezen door de MGB. Deze ID wordt u aangeleverd door de MGB. Indien u uw ID niet weet, moet u contact opnemen met de mgb.

Als u al deze gegevens heeft, kunt u uw software op uw computer zetten.
U krijgt twee bestanden. Zowel een applicatie `mgbkoerier.jar` als `config-koerier` zijn meegeleverd.
De informatie in `config-koerier` zou gelijk moeten zijn aan de gegevens die u hierboven heeft bepaald.

Wanneer u gevalideerd heeft dat de instellingen in `config-koerier` goed zijn, kunt u simpelweg `mgbkoerier.jar` starten en aan de slag met de koeriersapplicatie.

Installatie-instructies voor serverbeheerders

Vergeet niet dat Java geïnstalleerd moet zijn op het systeem, voordat u de server kunt draaien.
Voordat u begint met het gebruiken van de applicatie, moet u zorgen dat u de volgende gegevens heeft:

1. Een database-adres (bijvoorbeeld: sql.ewi.tudelft.nl/st4g4_boekw1)
2. Een database-gebruikersnaam (bijvoorbeeld: st4g4_boekw1)
3. Een database-wachtwoord (bijvoorbeeld: wacht_woord1)
4. De poort van de server (bijvoorbeeld: 4321)

Een database-adres
Dit is uw eigen database. Het adres zelf is het pad naar de database. Indien u dit pad niet weet, moet u dit uw database-ontwerper vragen.

Een database-gebruikersnaam
De gebruikersnaam die bij uw eigen database hoort. Indien u dit niet weet, moet u dit uw database-ontwerper vragen.

Een database-wachtwoord
Het wachtwoord voor de gebruikersnaam voor uw eigen database. Indien u dit niet weet, moet u dit uw database ontwerper vragen.

De poort van de server
Dit is een poort waarover communicatie geschiedt. Over het algemeen zijn de poorten van 8192 tot 32768 beschikbaar. Deze waarde is arbitrair maar kan bij toeval niet werken.

Als u al deze gegevens heeft kunt u uw software op uw computer zetten.
U krijgt twee bestanden. Zowel een applicatie `mgbserver.jar` als `config-server` zijn meegeleverd.
De informatie in `config-server` zou gelijk moeten zijn aan de gegevens die u hierboven heeft bepaald.

Wanneer u gevalideerd heeft dat de instellingen in `config-server` goed zijn, kunt u simpelweg `mgbserver.jar` starten en aan de slag met de MGB-server.

Appendix B: Planning

ST4 Planning

Groep: 4
Iteratie: 1

Onderdeel	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Datalaag voor winkel database	Community en Store klassen	80 min	3 uur	Erik	voltooid
	DBRecord klasse en child klassen	180 min	1 uur	Erwin	voltooid
	DBConnection, DBTable klasse en child klassen	180 min	5 uur	Jan	voltooid
Serverfunctionaliteiten	IO klassen	240 min	12 uur	Quinten	voltooid
	Request handlers, automatische herstart, kleine bugfixes		7 uur	Erwin	voltooid
Gebruikersinterface	Parent frame en winkel panel	105 min	10 uur	Erik	voltooid
	Overige panels en debuggen	45 min	10 uur	Wilson	voltooid
	Debuggen laadbalk		2 uur	Erwin	voltooid
Controllers voor winkels	StoreController klasse	150 min	10 min	Wilson	voltooid

ST4 Planning

Groep: 4
Iteratie: 2

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Gebruikersinterface boekbeheer	Uitbreiding datalaag, controller	4 uur	10 uur	Jan	voltooid
	Hoofdinterface (boekenlijst)	5 uur	9 uur	Erik	voltooid
	Interface voor toevoegen, bewerken	4 uur	10 uur	Wilson	voltooid
	Zoekfunctionaliteit	4 uur	4 uur	Quinten	voltooid
	Testen	3 uur	3 uur	Erwin	voltooid
Serverfunctionaliteiten	Verbeteren robuustheid server	3 uur	5 uur	Jan	voltooid
	Testen server	2 uur	4 uur	Erwin	voltooid
	Verbeteren robuustheid server		3 uur	Quinten	voltooid
Overig	Robuustheid interfaces verbeteren (Linux)	1 uur	2 uur	Erwin	voltooid

ST4 Planning

Groep: 4
Iteratie: 3

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Bestelfunctionaliteit boeken	Bestelfunctie, orders opvragen	10 uur	10 uur	Erik	voltooid
	Accepteren / weigeren bestellen	11 uur	9 uur	Wilson	voltooid
	Aanpassingen client voor verzoeken bij winkels (handlers etc)	10 uur	10 uur	Erwin	voltooid
	Aanpassingen server om clients om informatie te vragen	12 uur	13 uur	Quinten	voltooid
	Uitbreiden servertest, datalaag	8 uur	8 uur	Jan	datalaag voltooid
	Aanpassen design mbt orders	nvt	6 uur	Jan	voltooid
		nvt	4 uur	Wilson	voltooid
	Opzetten bugtracker	nvt	2 uur	Erwin	voltooid
	Algemeen debuggen	nvt	1 uur	Erwin	voltooid
			2 uur	Erik	voltooid
			2 uur	Jan	voltooid
			6 uur	Quinten	voltooid

ST4 Planning

Groep: 4
Iteratie: 4

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Koerierapplicatie	Interface koerier: orders, hoofdscherm	10 uur	7 uur	Erik	voltooid
	Interface MGB en booktrader aanpassen	12 uur	7 uur	Wilson	voltooid
	Handlers en serverfunctionaliteit, interface	8 uur	9 uur	Quinten	voltooid
	Aanpassingen singletons naar aanleiding van feedback	8 uur	12 uur	Jan	voltooid
	Testsuite verbeteren en aanvullen	10 uur	9 uur	Erwin	voltooid
	Overgebleven problemen iteratie 3	1 uur	1 uur	Erik	voltooid
	Error handling client		6 uur	Wilson	naar sprint 5
	Testen met meerdere systemen		2 uur	Erwin	voltooid
			2 uur	Erik	voltooid
			2 uur	Quinten	voltooid

ST4 Planning

Groep: 4
Sprint: 5

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Routes beheren door koerier	gebruikersinterface	8 uur	9 uur	Erik	voltooid
	datalaag uitbreiden voor routes (o.a. Kruistabel)	4 uur	4 uur	Jan	voltooid
Inlogfunctionaliteit	gebruikersbeheer	10 uur	11 uur	Wilson	voltooid
	datalaag uitbreiden voor gebruikers	2 uur	2 uur	Erwin	voltooid
Overig	openstaande problemen bugtracker	2 uur	3 uur	Quinten	voltooid
	verificatie van stores	5 uur	4 uur	Jan	voltooid
	tests uitbreiden (library splitsen, nieuwe tests)	8 uur	5 uur	Erwin	voltooid
	mergen van de gemaakte branch	4 uur	4 uur	Jan	voltooid
	uitwerken optimaliseringsproblemen	4 uur	2 uur	Wilson	voltooid,
	consistente error handling in huidige code	8 uur		Wilson	niet voltooid
	debuggen koerierapplicatie		3 uur	Erik	voltooid
			3 uur	Quinten	voltooid
	herschrijven handlers mbt koerier<->booktrader		4 uur	Erwin	voltooid
			4 uur	Quinten	voltooid
	gebruik timeouts en laadbalken verbeterd		1 uur	Erwin	voltooid

ST4 Planning

Groep: 4
Sprint: 6

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Optimalisatie en transfers	Uitwerken en implementeren optimalisatie	14 uur	12 uur	Jan	voltooid (deel naar sprint 7)
	Requests en handlers voor optimalisatie	10 uur	10 uur	Quinten	voltooid
	Datalaag voor transacties, requests en handlers	6 uur	5 uur	Erwin	voltooid
	Oplossen openstaande problemen bugtracker	6 uur	7 uur	Erik	voltooid
	Testsuites opnieuw compatibel maken en aanvullen	3 uur	2 uur	Erwin	naar sprint 7
	Boekbeheer beschikbaar zonder netwerkverbinding	4 uur	0 uur	Erik	uitgesteld
	Gebruikersbeheer afmaken	4 uur	5 uur	Wilson	voltooid
	Verzamelen data, optimizer aanroepen, afhandelen	nvt	2 uur	Erwin	voltooid
	aanpassen boekscherm + datalaag voor extra attribuut	nvt	1 uur	Erik	voltooid
	datalaag voor geoptimaliseerde route bij order	nvt	4 uur	Erik	voltooid
	GUI voor transacties	nvt	8 uur	Wilson	voltooid

ST4 Planning

Groep: 4
Sprint: 7

Feature	Taken	Verwacht	Tijd	Ontwikkelaar	Status
Optimalisatie en transfers	Transfers aanpassen	4 uur	7 uur	Wilson	voltooid
	Aanmaken van alle orders na optimaliseren	6 uur	4 uur	Erwin	voltooid
		6 uur	8 uur	Jan	voltooid
	Functionaliteit tijdens netwerkstoring	6 uur	5 uur	Quinten	voltooid
	Winkels op inactief ipv verwijderen	2 uur	2 uur	Erik	voltooid
	Aanvraag annuleren	1 uur	1 uur	Erik	voltooid
	Controleren of time-outs netjes opgevangen worden	1 uur	1 uur	Erik	voltooid
				Quinten	
	GUI verbeteren	4 uur	4 uur	Erik	voltooid
	Tests opnieuw compatibel maken	4 uur	4 uur	Erwin	voltooid