LIMBURGS UNIVERSITAIR CENTRUM
FACULTEIT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

De verdere ontwikkeling van een GIS-omgeving voor de bevraging van ruimtelijke mobiliteitsdata

Eindverhandeling voorgedragen tot het
behalen van de graad van
Handelsingenieur in de beleidsinformatica
door: Bruno BIJNENS
Promotor: Prof. dr. G. WETS
Copromotor: dr. T. BRIJS

2005

Woord vooraf

In het kader van mijn afstudeerrichting beleidsinformatica heb ik gekozen voor een computergerelateerd thesisonderwerp. Het leek mij interessant mee te helpen bij de ontwikkeling van een programma dat in de toekomst kan worden gebruikt. In het overleg met mijn thesispromotor en copromotor werd mij een onderwerp aangereikt wat mijn interesse wekte. Het betreft de verdere ontwikkeling van een applicatie die het mogelijk maakt om een databank te doorzoeken. De databank in kwestie bevat data over ongevallen die zich hebben voorgedaan op de Belgische wegen sinds het jaar 1989 tot en met 2000 en zijn opgenomen in een databanksysteem van het type IBM DB2. Deze werd gekozen voor zijn professesionele mogelijkheden met betrekking tot grootte en onderhoud. In de programmeertaal Java is een programma, AcciQuest genaamd, geschreven dat de gebruiker ervan toelaat, zonder enige kennis van DB2 of de SQL-taal een databank van dit type te doorzoeken. De omzetting naar DB2 en het schrijven van dit programma werd gerealiseerd door Geert Bens. Mijn opdracht bestond erin van de stand-alone-applicatie AcciQuest, een webapplicatie te maken die de gebruikers ervan toelaat, de gekoppelde databank te raadplegen over een netwerk.

Bruno Bijnens
Rekem, juni 2005

Samenvatting

De Systems Development Life Cycle is toegepast om de verdere ontwikkeling van een Geografisch Informatie Systeem [GIS]-omgeving voor het onderzoeken van ruimtelijke mobiliteitsdata te realiseren.
Het project werd als volgt geïdentificeerd: het ontwikkelen van een applicatie die mobiliteitsdata vanuit een netwerkgeoriënteerde invalshoek toegankelijk maakt. De systeemvereisten werden op het niveau van proces, logica en data beschreven. Deze vereisten leidden tot een Java2 Enterprise Edition [J2EE]-applicatie volgens *client*/server-architectuur met een driedelige configuratie. De serverzijde van de architectuur werd toevertrouwd aan een Tomcat 5.0.28 applicatieserver in combinatie met een Apache webcontainer. De data werden geplaatst in een IBM-DB2-serveromgeving en het zoekprogramma van de gebruiker deed dienst als *client*-machine.
De communicatie tussen de verschillende componenten van de architectuur werd toevertrouwd aan Java servlet-klassen met als taak: de invoer van de gebruiker doorgeven aan de achterliggende logica van de applicatie. De in deze logica gevormde Structured Query Language [SQL]-zoekopdracht wordt via *drivers* verstuurd naar de databank. De servlet genereert vervolgens de resultaten en stuurt deze naar de gebruiker door.
Om de organisatie voor te bereiden op het gebruik van het nieuwe informatiesysteem werden twee handleidingen en een installatieprocedure geschreven. Dit stadium van het SDLC-model omvatte onder meer het toetsen van de applicatie. Uit de toetsen bleek dat het nieuwe systeem correct werkt en de juiste data weergeeft.

INHOUDSOPGAVE

Glossarium

API	Application Programming Interface
ASP	Active Server Pages
BPP	Business Project Plan
BPR	Business Process Reengineering
CASE	Computer Aided Software Engineering
CGI	Common Gateway Interface
DB	DataBank
DBMS	DataBase Management System
DFD	Data Flow Diagram
DSS	Decision Support System
EIS	Enterprise Information System
ERD	Entiteiten-Relatie Diagram
GIS	Geografisch Informatie Systeem
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
IS	Informatie Systeem
J2EE	Java2 Enterprise Edition
J2SE	Java2 Standard Edition
JAD	Joint Application Design
JAR	Java Archive Resource
JDBC	Java DataBase Connectivity
JSP	Java ServerPages
MVC	Model View Controller model
ODBC	Open DataBase Connectivity
OOP	Object Oriented Programming
OSI	Open Systems Intercommunication

PC	Personal Computer
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
SDLC	System Development Life Cycle
SOW	Statement Of Work
SQL	Structured Query Language
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
WAR	Web Archive Resource
WWW	World Wide Web
XML	Extended Markup Language

1. Inleiding

1.1 Situering

De laatste jaren zijn grote stappen voorwaarts gezet op het domein van de computertechnologie. Vooral op het vlak van grafische gebruikersinterfaces, netwerken en het internet is de vooruitgang indrukwekkend geweest. Een wereld voorstellen zonder het Internet is tegenwoordig nog moeilijk te realiseren. Nieuwe technologieën op het vlak van informatiesystemen [IS] en de distributie van informatie leveren een steeds belangrijkere bijdrage in praktisch elke organisatie. Informatiesystemen en gerelateerde technologieën trachten de steeds grotere complexiteit van de informatiebehoeften van organisaties en gebruikers in te lossen.

Netwerken krijgen een steeds belangrijker wordende rol toebedeeld binnen informatiesystemen van organisaties. Steeds meer organisaties schakelen over van stand-alonesystemen naar, in netwerken verbonden systemen. Dergelijke systemen bieden mogelijkheden om tegemoet te komen aan problemen, veroorzaakt door de steeds grotere hoeveelheden data die in een organisatie omgaan. Op hetzelfde moment worden mainframe-applicaties vaker gedistribueerd over netwerken, met ondersteuning van enkele servers. Hiermee kunnen organisaties voordeel puren uit de grotere efficiëntie van dit type omgevingen. Netwerken kunnen worden uitgebreid naar het alles omvattende netwerk, gekend onder de naam world wide web [WWW] of ook wel, het internet.
Het internet laat toe programma's of data te delen over grotere groepen gebruikers die onderdeel uitmaken van de organisatie of daarbuiten opereren. Een organisatie kan kansen creëren door het toepassen van dergelijke technologieën. Soms maakt competitieve druk het noodzakelijk om concurrenten te volgen in het implementeren van deze nieuwe mogelijkheden. Alle typen van organisaties kunnen profiteren van nieuwe technologieën om effectiever of efficiënter de doelstellingen van de organisatie te verwezenlijken. In de commerciële wereld

duidt het succes van *e-commerce*, zowel naar de eindgebruikers als naar professionele klanten, op een naar alle verwachting mooie en geslaagde toekomst voor deze technologieën. [Hoffer et al., 2002]

1.2 Probleemstelling

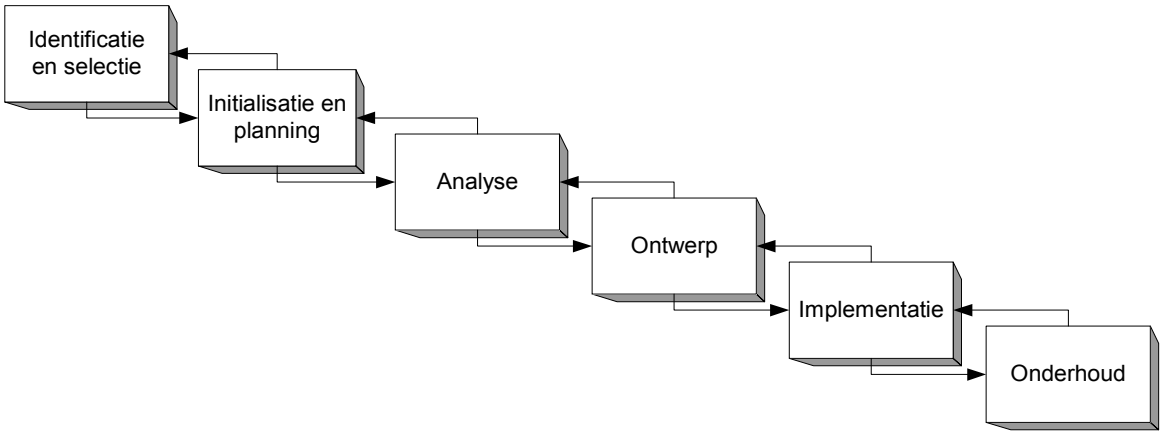
Het doel van de thesisopdracht wordt als volgt formeel verwoord:

De verdere ontwikkeling van een Geografisch Informatiesysteem [GIS]-omgeving voor het bevragen van ruimtelijke mobiliteitsdata.

Het beschikbaar maken van de mobiliteitsdata voor een groter publiek is een prioritaire activiteit binnen de ontwikkeling van de programmatuur. Een manier om dit te verwezenlijken is de ontwikkeling van een netwerkgeoriënteerd informatiesysteem. Het thesisproject kan meer specifiek worden omschreven als het ontwikkelen van een webapplicatie die analoge mogelijkheden biedt als het bestaande stand-aloneprogramma AcciQuest. AcciQuest betreft een in Java ontwikkeld programma dat, in opdracht van het Limburgs Universitair Centrum [LUC], tijdens een stage door Geert Bens werd ontworpen.
De nieuwe applicatie zal de gebruiker toelaten, de gekoppelde databank te raadplegen zonder dat de data, lokaal, beschikbaar moeten zijn. De data zijn over de periode van 1989 tot en met 2000 verzameld en hebben betrekking op de karakteristieken van ongevallen die plaatsvonden op de openbare hoofdwegen van België. Een webapplicatie laat de organisatie toe, de verzamelde data op een grotere schaal te gebruiken. De data hoeven daarenboven slechts op één plaats opgeslagen te worden. Dit kan kostenbesparingen realiseren op het vlak van onderhoud en beheer van de data en de applicatie. Hoe dit precies wordt bereikt is samen met het realiseren van het systeem de opdracht van deze thesis. In volgende hoofdstukken van de tekst komen de verdere vereisten die aan het systeem worden gesteld aan bod.

1.3 Werkwijze

Voor de realisatie van dit project wordt gebruik gemaakt van de SDLC-methode. SDLC staat voor de Systems Development Life Cycle en is een model om op een gestandaardiseerde wijze de ontwikkeling van een informatiesysteem, dat voldoet aan de wensen van de organisatie, te volbrengen. In de tekst wordt dezelfde opbouw gehanteerd als bij de ontwikkeling van het programma. De tekst heeft als taak de lezer te begeleiden in het doorgronden van het programma en uitleg te verschaffen bij het tot stand komen van het afgewerkte product. Elk stadium, dat tijdens de ontwikkeling van het programma wordt doorlopen, komt aan bod, ingeleid door een korte tekst in combinatie met een theoretische ondersteuning. Daarop volgt een beschrijving van de wijze waarop deze fase in de ontwikkeling van het programma tot uiting is gekomen.



figuur 1.1 De Systems Development Life Cycle (Hoffer et al., 2002)

Bovenstaande figuur 1.1 geeft de verschillende stappen van het SDLC-model weer. De SDLC-methode kan voor zeer uiteenlopende informaticagerelateerde projecten worden gebruikt. Dit thesisproject behoort hierbij eerder tot één van de meer eenvoudige toepassingen.

2. Projectidentificatie en -selectie

2.1 Inleiding

In het eerste stadium van de levenscyclus wordt, door bepaalde personen in de organisatie, een behoefte aan een nieuw of verbeterd informatiesysteem geïdentificeerd. Deze personen kunnen zijn: leidinggevenden, een stuurcomité, systeemanalisten of gebruikers. Een informatiebehoefte kan zeer kleine alsook zeer grote proporties aannemen. In overeenstemming met de behoefte wordt een project opgestart, gaande van een kleine applicatie tot de ontwikkeling van een Enterprise Information System [EIS].

In de fase projectidentificatie en -selectie onderscheidt Hoffer et al. [2002] de volgende activiteiten: a.het identificeren van potentiële projecten; b.het classificeren van deze projecten; c.het selecteren van de projecten die zullen worden uitgevoerd.

2.2 Projectidentificatie

Projecten kunnen worden geïdentificeerd via een top-down- of een bottom-upbenadering. Projecten die afkomstig zijn van een leidinggevende of een stuurcomité trachten aan een bredere nood tot informatieverstrekking tegemoet te komen. Dergelijke projecten kunnen invloed uitoefenen op de hele organisatie. Initiatieven vanuit de gebruiker zijn eerder gericht op het wegwerken van kleinere afwijkingen tussen informatienood en informatieverstrekking. Zij hebben, in de meeste gevallen, een kleinere invloed op de organisatie en worden eerder toegespitst op de noden van één of van enkele onderdelen binnen de organisatie.

2.3 Projectclassificatie en -selectie

De volgende taak in het eerste stadium van de cyclus betreft het indelen van de potentiële projecten. Aan alle geïdentificeerde projecten worden waarden toegekend volgens uiteenlopende methoden. De waarden duiden aan welke voor- en nadelen bepaalde projecten met zich meebrengen. Een veel gebruikte evaluatiemethode is de waardeketenanalyse [value chain analysis]. Bij deze methode worden de activiteiten van een organisatie ontleed om te bepalen bij welke van deze activiteiten waarde wordt toegevoegd. Hoe meer voordelen een informatiesysteem brengt tot de waardeketen, hoe hoger het project wordt gerangschikt.

De taak die nog rest, is de selectie van projecten voor verdere ontwikkeling. Een hele reeks factoren moeten onder de loep worden genomen bij projectselectie. De werkelijke noden, de lijst met potentiële projecten, de beschikbaarheid van middelen en de bereidheid ze op dit moment in te zetten, de huidige omgevingsfactoren en de gebruikte evaluatiecriteria hebben allen invloed bij de selectie van de toekomstige projecten.
Aan het einde van de eerste fase wordt een lijst gepresenteerd met alle geselecteerde projecten, in combinatie met een tijdsschema dat aangeeft wanneer bepaalde projecten kunnen evolueren naar de volgende stage van de levenscyclus. Er wordt een strategie van toenemende inspanning [incremental commitment] toegepast. Dit betekent dat na het afronden van een fase van het SDLC-model voor een bepaald project, dit project zal worden herbekeken. [Hoffer et al., 2002]

2.4 Toepassing op dit project

Het project is ontstaan op voorstel van de promotor en copromotor die een nood erkenden voor een netwerkgeoriënteerd programma dat het raadplegen van data mogelijk maakt. In de organisatie is een applicatie ontwikkeld onder de naam

AcciQuest. AcciQuest laat de gebruiker toe op een eenvoudige manier aan te geven aan welke criteria de data moeten voldoen. Het programma is echter enkel toe te passen indien de data geïnstalleerd zijn op hetzelfde station als de applicatie zelf. Hierdoor kan dit programma slechts door een, in het beste geval, kleine groep mensen worden gebruikt. De data, die worden gebruikt door deze applicatie, betreffen mobiliteitsdata die zich bevinden in een lokale IBM DB2-omgeving. Later in dit verslag wordt dieper ingegaan op de databank.

De doelstelling van het project is de verzamelde verkeersdata toegankelijk maken vanuit een netwerkgeoriënteerde, in plaats van lokale, invalshoek. Dit laat meerdere gebruikers, op meerdere locaties toe, de data te gebruiken zonder dat de programmatuur en de data verdeeld moeten worden over de verschillende gebruikersstations. De weergave van de gevraagde data moet geschieden in een voor de gebruiker beschikbaar formaat. Er wordt gedacht aan een weergave in tabelvorm en eventueel, in een later stadium, een weergave op één of andere geografische wijze. Een GIS-component kan een dergelijke taak uitvoeren. Het spreekt voor zich dat deze component verenigbaar moet zijn met de nieuwe netwerkgeoriënteerde applicatie. In de ontwikkeling van het project wordt rekening gehouden met het feit dat later eventueel een GIS-component wordt toegevoegd aan de applicatie. Wanneer het project volbracht is, moet een applicatie zijn ontstaan die gebaseerd is op netwerktechnologie en gebruikers toelaat, op afstand, verbinding te maken met een databank en de data hieruit te raadplegen.

3. Projectinitialisatie en -planning

3.1 Inleiding

In het tweede stadium van de levenscyclus wordt een meer gedetailleerde weergave gemaakt van een geselecteerd project, op basis van de initialisatie en de planning van het project. De waardering van een project is daarbij toegespitst, niet op de operationele werking van het toekomstige informatiesysteem, maar op het begrijpen van de invloed van het project op de organisatie. Verder wordt onderzocht welke middelen dienen ingezet te worden om het project te realiseren. De kans op succes, een geslaagde volbrenging van het project op basis van de ingezette middelen, speelt een rol. Dit stadium kan nog altijd, na gedetailleerde studie, leiden tot het afwijzen van een project.
De fase zal een duidelijke projectbeschrijving moeten opleveren. Goede communicatie tussen de systeemanalist, de leiding en de toekomstige gebruiker is noodzakelijk. De analist moet zich een duidelijk beeld kunnen vormen van de wensen en verwachtingen van de gebruikers en van de leiding. Deze fase en de volgende fase zijn moeilijk van elkaar te onderscheiden en vertonen enkele overeenkomstige taken. Er moet echter worden benadrukt dat in deze fase nog altijd een beeld moet worden gevormd van de voor- en nadelen van het project. Hier wordt nog geen tijd besteed aan de operationele werking van het informatiesysteem. Het project kan, zoals eerder aangehaald, nog altijd worden afgewezen. [Hoffer et al., 2002]

3.2 Projectinitialisatie

De projectinitialisatie is belast met het organiseren van een groep die de projectplanning gaat realiseren. Er wordt getracht de omvang, de invloed op de organisatie en complexiteit van het project te beoordelen. Verder worden procedures ontwikkeld die het projectteam ondersteunen bij de te realiseren

activiteiten. Er moet tevens een goede relatie worden opgebouwd met de gebruikers.

3.3 Projectplanning

Bij de projectplanning wordt getracht een aantal duidelijke, losstaande activiteiten binnen één enkel project te identificeren alsook het werk dat nodig is om deze individuele taken tot een goed einde te brengen. Het doel vormt de creatie van een Baseline Project Plan [BPP] en eventueel een Statement Of Work [SOW]. In het eerste plan wordt een beschrijving gegeven van het project in termen van kosten, opbrengsten, risico's en de vereisten op het vlak van middelen. Het wordt gebruikt tijdens de selectieprocedure om mee te bepalen of het project wordt verder gezet. Het tweede plan geeft de objectieven en de beperkingen van het project weer aan de klant, in combinatie met start- en einddata, performantiemaatstaven in termen van het budget en de beoogde mijlpalen. Voordat deze plannen kunnen worden opgesteld, moet de haalbaarheid van het project worden beoordeeld. [Hoffer et al., 2002; Chase et al., 2004]

"All projects are feasible given unlimited resources and infinite time." [Pressman, 2001]

Voor de meeste projecten gelden economische, technische of andere beperkingen. In volgende onderdelen wordt een idee gevormd over de verschillende beperkingen en opbrengsten van het project.

3.3.1 Economische haalbaarheid

Bij het bepalen van de economische kenmerken van een project gaat de aandacht uit naar de kosten die een project met zich meebrengt en de meeropbrengsten die het project zal genereren. Een onderscheid kan worden gemaakt tussen tastbare en ontastbare kosten en opbrengsten. Een aantal tastbare voordelen voor de

organisatie zijn: kostenvermindering, kleinere foutenmarges, toegenomen flexibiliteit, toegenomen snelheid, betere controle, minder uitval. Ook ontastbare voordelen kunnen zich manifesteren. Denk hierbij aan gelukkigere of meer gemotiveerde werknemers. Deze voordelen zijn nauwelijks kwantitatief in geld uit te drukken. Aan de kostenzijde kan ook het onderscheid worden gemaakt tussen tastbare en ontastbare kosten. Tastbare kosten zijn de kosten die goed meetbaar zijn in geldwaarde en met een bepaalde zekerheid kunnen worden vastgesteld. Denk hierbij aan de aankoopkosten van een nieuwe server. Ontastbare kosten zijn eerder moeilijk uit te drukken in een geldwaarde. Een voorbeeld voor deze categorie is het een verlies aan goodwill van een klant. [Hoffer et al., 2002]

Volgens Hoffer et al. [2002] kunnen kosten, bij informaticaprojecten, ook op een andere manier worden ingedeeld. Éénmalige instelkosten betreffen kosten die slechts één keer gemaakt hoeven te worden. Hieronder worden gegroepeerd, zowel de ontwikkelingskosten als de opstartkosten. Voorbeelden zijn aankoopkosten voor hardware en software, en de salarissen van de ontwikkelaars. Verder zijn er de periodiek terugkerende kosten. Denk hierbij aan onderhoudskosten of gebruikskosten. Beide kunnen verder worden onderverdeeld in een variabel en een vast gedeelte.

3.3.2 Technische haalbaarheid

De mogelijkheden van de organisatie om het vooropgestelde systeem te construeren, vallen onder deze categorie van haalbaarheid. Alle projecten hebben een risico en het is belangrijk om dit risico zo vroeg mogelijk te onderkennen. Hoffer et al. [2002] identificeert een aantal factoren die mee bepalen hoeveel risico geassocieerd wordt met een project: a.grote projecten kennen meer risico dan kleine projecten; b.duidelijke systeemvereisten leiden tot een lager risico; c.gebruik van standaard technologie is minder risicovol; d.minder risico indien gebruikers op de hoogte zijn van het systeemontwikkelingsproces.

Naast de twee aangehaalde haalbaarheidsstudies bestaan er nog een hele reeks andere studies die een waardering van het project kunnen weergeven. Deze studies kunnen behandelen: operationele haalbaarheid, schemahaalbaarheid, wettelijke en contractuele haalbaarheid, politieke en eventueel ethische haalbaarheid. Verdere informatie over deze studies is terug te vinden in de gespecialiseerde literatuur. [Hoffer et al., 2002]

3.4 Toepassing op het project

Het project wordt gerealiseerd in het teken van de thesisopdracht. In het kader van dit project wordt minder aandacht besteed aan de eerste twee fasen van de SDLC dan normaal het geval moet zijn. Niettegenstaande zal worden getracht bovenstaande stadia uit te werken. De opdracht werd in overleg met de promotoren vastgelegd. Projectinitialisatie is minder van toepassing aangezien de thesisopdracht een individueel werk betreft. Een evenwichtige taakverdeling hoeft uiteraard niet uitgewerkt te worden. Er werd duidelijk gecommuniceerd met de opdrachtgevers over de mogelijkheden die het programma moest bevatten en over de gebruikersvereisten. Hier wordt kort overlopen wat in het BPP kan worden opgenomen.

3.4.1 Economische haalbaarheid

Economische redenen voor het realiseren van dit project zijn talrijk. Dataverzameling is belangrijk voor onderzoeksinstellingen. Veel tijd en geld worden geïnvesteerd in het verzamelen van de juiste data. De data vormen de basis voor het bedenken van oplossingen voor bestaande problemen of het bedenken van nieuwe mogelijkheden. Dataverzameling is noodzakelijk maar heeft weinig nut indien de data niet op een effectieve manier, kan worden geraadpleegd.

Dit project bereikt op het vlak van de distributie haar meerwaarde ten opzichte van het huidige systeem. Na uitvoering van het project zou de mobiliteitsdata

bereikbaar moeten zijn voor meerdere gebruikers van op afstand. Bovendien zouden de data op een meer gebruiksvriendelijke manier moeten worden weergegeven. Dit kan bijdragen tot een hogere effectiviteit van de investeringen die worden gedaan om de data te verzamelen. Verder kan dit leiden tot het beter doorgronden van ongelukken en de factoren die hier mede verantwoordelijk voor zijn. Indien het project hiertoe een bijdrage kan leveren, dan kunnen oplossingen door verkeersdeskundigen worden bedacht die effectiever zijn in het minimaliseren van het aantal ongelukken. De overheid en de maatschappij in zijn geheel zouden minder financieel en emotioneel verlies lijden, veroorzaakt door ongelukken en alle gevolgen die deze met zich meebrengen.

Het project zal tevens toelaten op slechts één welbepaalde locatie de data te beheren en te onderhouden. Ook de applicatie zelf kan op één welbepaalde locatie worden geplaatst. Dit kan leiden tot een efficiënter gebruik van de onderhoudsdiensten doordat deze slechts op één locatie hun werkzaamheden dienen te verrichten, in plaats van op elke locatie waar data- en programmabestanden zijn gelocaliseerd.

Op het vlak van kosten wordt kort ingegaan. Naast de opportuniteitskosten, die verloren zijn gegaan door het ter beschikking stellen van de nodige apparatuur, zijn verder geen kosten te onderkennen.

3.4.2 Technische haalbaarheid

Aangezien de nieuwe applicatie op een analoge wijze moet opereren als AcciQuest, gaat in dit project vooral de aandacht uit naar de eisen van de leidinggevenden. Deze wensen een over een netwerk gedistribueerd systeem dat voor de gebruiker niet anders werkt dan het huidige systeem, maar veel meer gebruikers toelaat. Het moet meer mogelijkheden bieden en toelaten van op afstand (van de data) gebruik te maken van de applicatie. Aangezien dit project

slechts een geringe omvang heeft, zijn vele initialisatie-activiteiten die in deze fase uitgevoerd zouden kunnen worden, op dit project minder van toepassing.

De hoeveelheid risico die met dit project wordt geassocieerd, hangt af van zes factoren die inherent zijn aan het project. Een tijdslimiet waarin het project afgewerkt moet zijn, dient natuurlijk gerespecteerd te worden. Door de nodige ondersteuning, die geleverd wordt door de promotors, zullen ook de technische uitdagingen zonder problemen kunnen worden volbracht. De vereisten voor het project zijn daarenboven zeer duidelijk gestructureerd. De risico's worden verder geminimaliseerd door het beschikbaar stellen van het programma AcciQuest. Sommige componenten van deze applicatie kunnen eventueel hergebruikt worden in de nieuwe toepassing. Hierdoor is verdere kostenbesparing in de vorm van tijdswinst mogelijk. Aangezien de gebruikersinterface een analoge opbouw kent als die in AcciQuest, zal hier verder het risico, op het niet slagen van het project, worden verminderd. De technologie die wordt gebruikt bij de ontwikkeling is standaardtechnologie. De ontwikkelaar heeft echter de vrijheid deze zelf te bepalen. De opdrachtgevers hebben geen verplichtingen opgelegd. Een voorstel werd echter gedaan om de programmeertaal Java te gebruiken. Verdere uitleg over de technologie volgt in de volgende fase van de SDLC.

4. Analyse

4.1 Inleiding

Analyse is het onderdeel van de systeemontwikkelingscyclus [SDLC] waarin de werking van het huidige systeem en de verwachtingen van het nieuwe systeem gedetailleerd worden beschreven. In dit stadium van de cyclus worden een aantal taken onderscheiden. Een grondige analyse van de verwachtingen en de noden van de organisatie is het basiselement voor een formele bepaling van de systeemvereisten. De verzamelde gegevens moeten daarenboven op een gestructureerde wijze worden weergegeven. Een laatste activiteit die wordt onderscheiden in deze fase, is de selectie van een gepaste ontwerpstrategie.
Vervolgens wordt een korte theoretische beschrijving gegeven op het vlak van communicatie, hardware en software. Het hoofdstuk wordt afgesloten met de toepassing van deze concepten op het project.

4.2 Bepalen van de systeemvereisten

De taak van de analist bestaat uit het determineren van de eisen die worden gesteld aan het nieuwe systeem. Alvorens hiermee van start te gaan, onderzoekt de analist welke kenmerken het huidige systeem heeft. Deze kenmerken voldoen niet meer aan de wensen van de gebruiker waardoor het initiatief is ontstaan voor de creatie van een nieuw systeem. Om het nieuwe systeem beter te laten aansluiten bij de wensen van de gebruikers en de leidinggevenden, moet worden onderzocht welke precies de verwachtingen van deze twee doelgroepen zijn. Deze informatie kan worden verkregen door een studie van de huidige infrastructuur en door diepgaande gesprekken met zowel de gebruikers als de leidinggevenden. Een gedetailleerde lijst van functionaliteiten van het nieuwe systeem wordt in deze fase bepaald. [Hoffer et al., 2002]

Hoffer et al. [2002] identificeert verschillende werkwijzen om de bepaling van de systeemvereisten met succes te volbrengen. Enerzijds kan gebruik worden gemaakt van traditionele methodes zoals individuele interviews, enquêtes, groepinterviews en eventueel observatie van de gebruikers in hun werkomgeving. Tevens kan een studie van formele documenten worden georganiseerd. Uit deze documenten kunnen regels en richtlijnen over ondernemingsbeleid worden afgeleid naast voorbeelden van gebruik van data. Anderzijds zijn nieuwere technieken ontwikkeld om de vereisten van het systeem te definiëren. Dit kunnen zijn: Joint Application Design [JAD], Computer Assisted Software Engineering [CASE], Group Support Systems, Prototyping en in de meer extreme gevallen, Business Process Reengineering [BPR].

4.3 Structureren van de systeemvereisten

4.3.1 Procesweergave

Procesweergave betreft de eerste activiteit in het structureren van de systeemvereisten. Bij deze activiteit wordt getracht de beweging van de data doorheen het systeem tussen externe entiteiten, de processen en de dataopslag weer te geven. Het opstellen van een Data Flow Diagram [DFD] is een manier om informatie over de beweging van de data te structureren. Deze en de twee volgende structureringsonderdelen geven de systeemanalist een duidelijk beeld van het verwachte systeem. Op basis van deze informatie kan worden gestart met het ontwerp.

4.3.2 Logicaweergave

Bij het modelleren van de logica van een applicatie wordt dieper ingegaan op de manier waarop de verschillende processen data transformeren. DFD's zijn geen geschikt middel om dergelijke informatie weer te geven. De logicaweergave tracht de logica, die in elk proces aanwezig is, te structureren. Dit kan worden

gerealiseerd via talloze technieken. Hoffer et al. [2002] noemt volgende mogelijkheden: gestructureerd Engels, beslissingstabellen en beslissingsbomen.

4.3.3 Conceptuele dataweergave

Naast voorgaande technieken om bepaalde systeemvereisten te structureren, moet nog een kritisch onderdeel worden behandeld: de structuur van de data. De bewegingen van de data en de transformatie van de data door processen zijn al duidelijk. DFD's en beslissingstabellen geven weer hoe, waar en wanneer data gebruikt of veranderd worden in een informatiesysteem maar zij verschaffen geen klare kijk op de definitie, de structuur van, en de relaties tussen data. Deze informatie is noodzakelijk bij de bouw van een informatiesysteem. Op verschillende momenten in de ontwikkeling worden de kenmerken van de data gebruikt om de data correct weer te geven of op te slaan. Het meest gebruikte formaat voor het structuren van data is het entiteiten-relatiediagam [ERD]. Bij ERD wordt de datastructuur weergegeven, onafhankelijk van de manier waarop ze wordt opgeslagen in het computergeheugen. [Hoffer et al., 2002]

4.4 Selecteren van de beste ontwerpstrategie

De laatste activiteit bij de analysefase heeft volgens Hoffer et al. [2002] betrekking op het formuleren van een aantal ontwerpalternatieven. Het beste alternatief om de doelstellingen te verwezenlijken wordt geselecteerd. De ontwerpstrategie bevat algemene informatie over de aanpak die gevolgd zal worden bij de ontwikkeling van het IS. Onderdeel hiervan vormen de functionaliteiten van het systeem en de hardware- en softwareplatforms.

4.5 Theoretisch concept: communicatie

De succesvolle overdracht van berichten tussen applicaties over een netwerk vraagt communicatie tussen verschillende hardware- en softwarecomponenten. De

communicatie tussen overeenkomstige processen op verschillende machines wordt beheerd door specifieke standaarden die gekend zijn onder de noemer protocollen. Er zijn twee belangrijke mogelijkheden op het vlak van communicatieprotocollen bij dergelijke toepassingen. Enerzijds is er de TCP/IP-configuratie. TCP staat voor Transmission Control Protocol en vereist het gebruik van IP, het Internet Protocol. Een andere mogelijkheid is de OSI [Open Systems Intercommunication] die in het leven werd geroepen door de Internationale Organisatie voor Standaarden [ISO]. In de meeste netwerken wordt een combinatie gebruikt van de twee standaarden waarbij de OSI domineert bij de onderste lagen en de TCP/IP-configuratie domineert bij de bovenste communicatielagen. De TCP/IP-architectuur is noodzakelijk voor communicatie via het internet op de internet- en transportlagen. [Panko, 2003]

4.6 Theoretisch concept: hardware architectuur

4.6.1 Inleiding

Een IT-architectuur kan worden beschouwd als een blauwdruk van het informatiesysteem dat aangeeft hoe het systeem eruit gaat zien en hoe alle componenten van het systeem aan elkaar gerelateerd zijn. [McNurlin et al., 2004]

Gegevensverwerking werd in de meer traditionele vormen van informatiesystemen centraal georganiseerd. In een dergelijke architectuur voert een centraal geplaatste computer of groep van computers de *data processing* of gegevensverwerking uit. Activiteiten worden aangestuurd vanuit de locatie zelf of, via terminals, vanuit andere locaties. De gebruiker wordt voorzien van een interactiemiddel dat verbonden is met de centrale gegevensverwerkingseenheid via een communicatievoorziening. Deze systemen worden ook wel aangeduid met de naam Host-Terminal. Gecentraliseerde faciliteiten voor gegevensverwerking kunnen op meerdere aspecten gecentraliseerd zijn. Denk hierbij aan

gecentraliseerde computers, gecentraliseerde verwerking of gewoon gecentraliseerde gegevens. [Stallings, 2001]

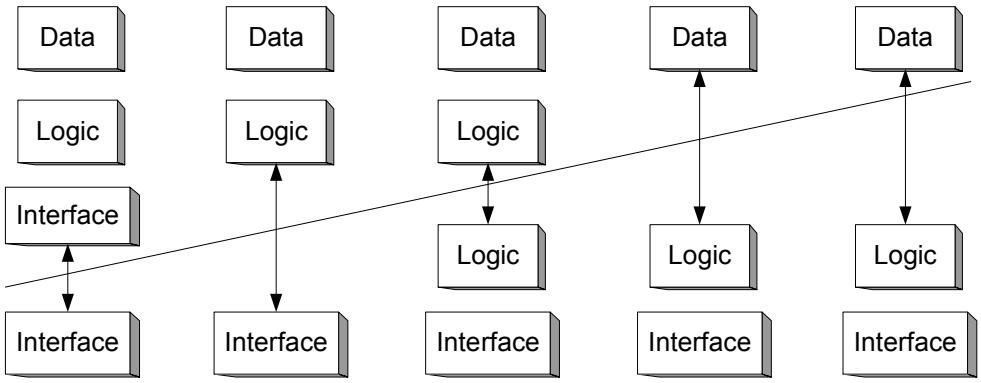
De tegenhanger van een dergelijke gecentraliseerde aanpak wordt door Stallings [2001] geïdentificeerd als *distributed data processing* of gedistribueerde gegevensverwerking. De decentralisatie van de gegevens-verwerking kan op uiteenlopende niveaus worden geïmplementeerd. Het scenario omvat de verspreiding en verwerking van gegevens, en de toestellen die deze mogelijkheden kunnen ondersteunen. Twee toepassingen van gedistribueerde gegevensverwerking worden toegelicht. Eerst wordt de familie van *client*/server-verwerking van nader bekeken. Vervolgens worden de kenmerken van een *peer-to-peer*-architectuur onderzocht.

4.6.2 *Client*/serverarchitectuur

Definitie *client*/serververwerking:

"*Client/serververwerking impliceert een coöperatieve verwerking van verzoeken die worden ingediend door een client bij de server, die de verzoeken verwerkt en de resultaten teruggeeft aan de client. In dit model wordt de toepassingsverwerking (niet noodzakelijk even) gedeeld tussen de client en de server. De verwerking wordt feitelijk gestart en gedeeltelijk bestuurd door de client, maar niet in een master/slave-verhouding. In plaats daarvan werken client en server samen om de toepassing met succes uit te voeren.*" [Berson, 1996]

Het *client*/servermodel is een veel gebruikte vorm van gedistribueerde gegevensverwerking. Een applicatie bestaat uit drie componenten: data, logica en een presentatiegedeelte. Figuur 4.1 (p.26) biedt een overzicht van de verschillende mogelijkheden, naargelang de taakverdeling. De gegevensverwerking wordt gedeeld tussen de computer van de gebruiker en de servers die met elkaar via een netwerk verbonden zijn.



figuur 4.1 Client/Server model, server(boven) en client(beneden) (Laudon et al., 2004)

Bij een *client*/serverarchitectuur zijn twee stations te onderscheiden. Beide stations krijgen de taken toegewezen die zij het beste kunnen uitvoeren. De *client*-zijde van een dergelijk netwerk betreft vaak de *point-of-entry* van de gebruiker. In de meeste gevallen is deze *client*-zijde een werkstation of een gewone computer. Indien een toegangspunt in het netwerk aanwezig is, kunnen eveneens draadloos verbonden machines zoals een Personal Digital Assistant [PDA] of een draagbare computer met draadloze ontvanger/verzender de functie van *client* in het model vervullen. De server levert op vraag van de gebruiker, een aantal diensten. De servermodule kan een ander werkstation, een gewone computer, een mainframecomputer of een gespecialiseerde servermachine zijn. Deze netwerkcomputers nemen tevens het beheer van het netwerk, de opslag van gegevens en andere functies voor hun rekening. Het meest gebruikte type server is de databankserver, die in de meeste gevallen een relationele databank beheert. De server biedt aan veel gebruikers de mogelijkheid de toegang tot dezelfde databank te delen. [Laudon, 2002]

Normaal moeten *client*/serversystemen aan de gebruikerszijde beschikken over de nodige programma's om contact te verzorgen met de serverzijde. Dergelijke systemen maakten vroeger gebruik van gespecialiseerde programma's. Deze aanpak vroeg echter veel tijd en investeringen om te implementeren.

Tegenwoordig kan gebruik worden gemaakt van het ene *client*-programma dat zo goed als elke PC heeft: de browser of zoekmachine. Vele *client*/serverapplicaties maken gebruik van de zoekmachine als interactieprogramma. Hierdoor wordt het mogelijk via het internet te communiceren en hoeft geen apart *client*-zijde-programma ontwikkeld te worden.

4.6.3 *Peer-To-Peer*-architectuur

De gegevensverwerking komt bij *peer-to-peer computing* terug terecht bij de computers van de gebruikers. Echter nu wordt de verwerkingskracht van de in netwerken verbonden computers gebundeld. De stations kunnen hierdoor data, verwerkingskracht en geheugen delen met alle andere computers die onderdeel uitmaken van het netwerk. Een contrast met het bovenstaande *Network Computing (client/*server*)* wordt aangegeven door het feit dat bij *peer-to-peer computing* geen sprake is van servers, mainframes of een andere centrale controlerende instantie. Al de verwerkingskracht is gehuisd bij de PC's zelf. Servers voor de opslag van bestanden kunnen natuurlijk wel voorkomen. Deze manier van werken laat toe extra capaciteit, die normaal in elke computer aanwezig is, te laten gebruiken door anderen. Door deze verwerkingskracht te combineren kunnen taken worden uitgevoerd die eerder enkel door zeer krachtige servercomputers of supercomputers konden worden geklaard. [McNurlin et al., 2004]

Één bepaalde vorm van *peer-to-peer* wordt ook wel *grid computing* genoemd. Speciale software wordt ingeschakeld om ongebruikte capaciteit in het netwerk op te sporen en dit samen te voegen zodat een virtuele 'supercomputer' wordt bekomen. Grote taken worden opgesplitst in kleine afzonderlijk te verrichten deeltaken die gedistribueerd worden over de beschikbare verwerkingscapaciteit, verspreid over verschillende machines. [Laudon, 2004]

4.7 Theoretisch concept: software

4.7.1 Java2 Enterprise Editie

Een mogelijke programmeertaal voor het schrijven van applicaties is Java. Een gepaste specificatie voor het construeren van webapplicaties is beschikbaar binnen deze programmeertaal en is gekend onder de naam: Java2 Platform, Enterprise Edition [J2EE]. J2EE is een uitbreiding van de standaard Java2 Standard Edition [J2SE] waarbij Application Programming Interface [API]-compatibiliteit is toegevoegd ter ondersteuning van applicatieservers. Deze ondersteuning laat toe servlets, Java ServerPages [JSP] en *deployment descriptors* te schrijven. Naast deze worden nog een aantal andere zaken zoals *Web Services* en *enterpriseBeans* door dit platform ondersteund. J2EE is platformonafhankelijk en biedt hiermee de mogelijkheid te kiezen welke technologie van welke fabricanten verder wordt gebruikt.

Op het J2EE-platform worden programma's ontwikkeld in de Java-programmeertaal naar analogie met een losstaande applicatie binnen J2SE. Echter het verhaal is gecompliceerder door de toevoeging van servers en de spreiding van de applicatie over meerdere machines. Communicatie tussen verschillende componenten moet tot stand worden gebracht om het beoogde resultaat te realiseren. Een J2EE-applicatie wordt over het algemeen gezien als een driedelige [three-tier] applicatie omdat het verspreid wordt over drie locaties: gebruikerstoestel, J2EE-server en een databankmachine. De uitbreiding ten opzichte van een tweedelige [two-tier] applicatie betreft de toevoeging van een *multithreaded* applicatieserver tussen het gebruikerstoestel en de DB-server. Indien wordt geredeneerd vanuit aparte taken, dan is er sprake van een vierdelige [four-tier] toepassing. Hierbij worden de uit te voeren taken, in plaats van de machines die de taken uitvoeren, onderscheiden als *tier*. [Armstrong et al, 2004]

4.7.2 Andere mogelijkheden

Andere mogelijkheden op het vlak van software worden aangevoerd door het MicroSoft product Visual Basic voor het logica-aspect, in combinatie met Active Server Pages [ASP] dat de communicatie met de server verzorgt. Het presentatiegedeelte wordt gerealiseerd door HTML-pagina's [HyperText Markup Language]. Een andere mogelijkheid is het gebruik van Hypertext Preprocessor [PHP] voor de communicatie met de server. In bijna alle alternatieven worden HTML-bestanden gebruikt als webpagina's. Bij Java kunnen echter ook JSP's worden gebruikt voor het presentatiegedeelte. JSP zijn een combinatie van Java Servlets en HTML. Zij hebben de mogelijkheid dynamische inhoud weer te geven op de pagina. In volgend hoofdstuk wordt dit verder besproken.

4.7.3 Databankconnectiesoftware

De meeste programmeringstalen voorzien methodes of functies die een applicatie toelaten te communiceren met een databank. Open DataBase Connectivity [ODBC] is een standaard databanktoegangsmethode die werd ontwikkeld door de SQL Access Group in 1992. Het doel van ODBC is het mogelijk maken om data te raadplegen vanuit elke applicatie, ongeacht welk DataBase Management System [DBMS] de data beheert. ODBC realiseert deze mogelijkheid door *drivers* toe te voegen tussen de applicatie en het beheersysteem van de databank. De *driver* vertaalt de applicatie-opdracht naar commando's die door het DBMS begrepen kunnen worden. Deze *drivers* zijn specifiek voor praktisch elke combinatie databank/programmeertaal. Een ODBC-*driver* voor Java-applicaties wordt ook wel eens een Java DataBase Connectivity [JDBC]-*driver* genoemd.

4.8 Toepassing op het project

4.8.1 Bepalen van de systeemvereisten

Het toekomstige systeem moet breed gezien dezelfde functionaliteiten bevatten als het oude systeem. Daarom is het belangrijk dat voldoende aandacht wordt besteed aan het onderzoeken van de huidige applicatie. Beide zullen een aantal overeenkomsten vertonen, zeker op het vlak van de gebruikersinterface. Het doel van de nieuwe toepassing is: de gebruikers van de applicatie een manier bieden waarop zij eenvoudig de databank kunnen raadplegen. Dit doel komt overeen met het doel waarvoor het vorige systeem, AcciQuest, werd gecreëerd. Eerst wordt dit systeem nader bekeken. Vervolgens wordt ingegaan op de punten waarin de beide systemen van elkaar moeten verschillen.

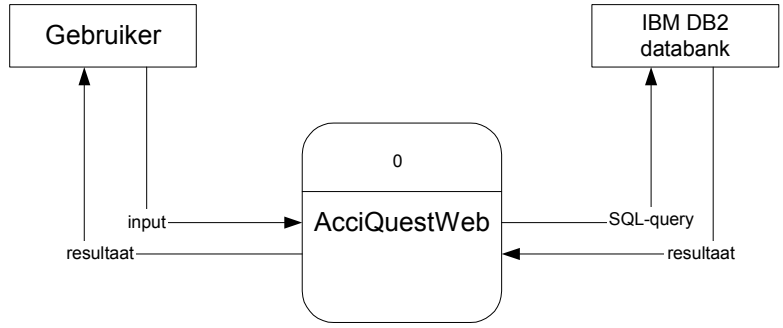
Het huidige systeem bestaat uit een lokaal draaiende applicatie die geschreven is in de programmeertaal Java. De applicatie wordt gestart vanuit MS-DOS. Eens het programma draait, kan een verbinding worden gemaakt met de databank. Deze is van het type DB2. Op de ter beschikking gestelde computer draait een IBM DB2 omgeving die instaat voor het databeheer. In het programma AcciQuest kunnen de gebruikers een aantal criteria kiezen waarop zij de data willen selecteren. Het programma zet hierbi,j achter de schermen, deze informatie om naar een Standard Query Language [SQL]-zoekopdracht. De SQL-code wordt vervolgens doorgegeven aan de databank. De zoekopdracht wordt uitgevoerd en de resultaten worden in een eenvoudig tekstformaat opgeslagen op de harde schijf van het station. Het programma bouwt een SQL-opdracht die rekening houdt met alle voorkeuren van de gebruiker. Het is tevens zo geschreven, door gebruik van *arrays* van variabelen, dat op een eenvoudige wijze, componenten kunnen worden toegevoegd zonder dat de achterliggende logica moet worden herschreven. Het programma biedt tevens een overzichtelijke gebruikersinterface en de mogelijkheid om met een andere databank te connecteren. Deze moet, gezien de eenvoudige architectuur, wel op hetzelfde station als de applicatie aanwezig zijn.

Het nieuwe systeem moet alle mogelijkheden bieden die ook in de huidige applicatie voorkomen. Maar waar verschillen beide van elkaar? Het grote onderscheid tussen beide systemen komt tot uiting op drie vlakken. Het nieuwe programma moet gebruikers, die op de hoogte zijn van SQL, toelaten zelf volledige zoekopdrachten te schrijven en die eenvoudig via het interactiemedium van de applicatie tot uitvoer te brengen. Ten tweede moet het nieuwe systeem tegemoet komen aan de eisen van de gebruikers om op een meer overzichtelijke manier de data te raadplegen. Een mogelijkheid bestaat uit het weergeven van de data in een tabelformaat dat in het programma zelf beschikbaar wordt voor de gebruiker. Een derde verschil heeft betrekking op de opbouw van de applicatie zelf. Om een netwerkgecentreerd systeem te maken, moet een andere architectuur worden toegepast. In ieder geval wordt een databankserver met de verkeersdata geïntegreerd in het systeem. Verder moet toegang tot het informatiesysteem verlopen via het zoekprogramma van de gebruiker. Toegang via het internet moet namelijk tot de mogelijkheden behoren. Hiermee wordt bedoeld dat internetstandaarden moeten worden toegepast voor de communicatie tussen de verschillende stations.

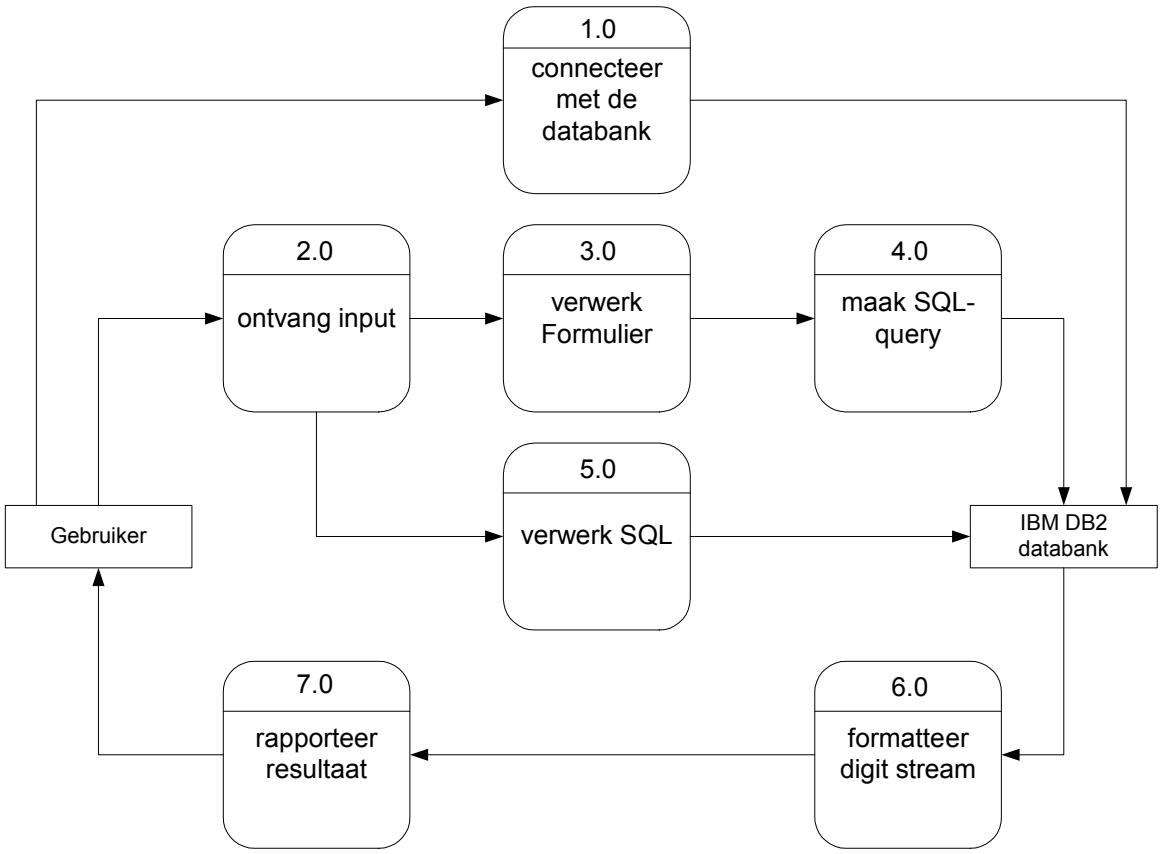
Bovenstaande informatie is verzameld door gesprekken met de opdrachtgevers van het project. Een grote vrijheid is gegeven aan de uitvoerder om de architectuur van het informatiesysteem zelf te bepalen. Door de eisen die gesteld worden aan het IS kunnen natuurlijk slechts een aantal architecturen mogelijk.

4.8.2 Procesweergave: DFD

De procesweergave wordt gerealiseerd door een DFD: zowel een contextdiagram (figuur 4.2) als een Level-0-diagram (figuur 4.3). Beide figuren zijn gesitueerd op de volgende pagina. De processen die zijn weergegeven op de figuren worden nader uitgelegd in de bijbehorende beschrijving.



figuur 4.2 Context diagram AcciQuestWeb



figuur 4.3 Level-0-diagram AcciQuestWeb

4.8.3 Procesweergave: beschrijving

Connecteer met de databank (1.0)

Deze stap betreft een speciale keuze voor de gebruiker. De gebruiker moet, voordat het programma beschikbaar wordt gesteld, een verbinding realiseren met de databank. Hiervoor worden methodes ontworpen die door de gebruiker zullen worden aangeroepen. De gebruiker zal waarschijnlijk bepaalde invoer moeten verschaffen over de coördinaten van de databank waarmee connectie gemaakt dient te worden.

Ontvang invoer (2.0)

In deze stap wordt de invoer van de gebruiker, afhankelijk van de pagina die is ingevuld, geïmporteerd in de applicatie. De gebruiker zal op verschillende plaatsen invoer kunnen ingeven in het programma. Deze informatie zal allemaal via dit proces worden beheerd.

Verwerk Formulier (3.0)

In dit proces moeten acties worden ondernomen die eigen zijn aan de keuze die de gebruiker heeft gemaakt wat betreft de invoermogelijkheden. In dit scenario heeft de gebruiker gekozen voor een invoer met behulp van een formulier waarop gekozen wordt welke criteria onderdeel moeten uitmaken van de zoekopdracht. Tevens wordt in dit formulier ingegeven aan welke waarden de criteria moeten voldoen.

Maak SQL-zoekopdracht (4.0)

In deze stap wordt de input van de gebruiker verwerkt indien deze gekozen heeft voor het invulformulier. Op basis van deze data wordt door de applicatie een SQL-opdracht, met eventuele deelopdrachten, gecreëerd die bestaat uit een reeks variabelen en SQL-code. De variabelen komen overeen met de keuzes die door de gebruiker zijn aangegeven.

Verwerk SQL (5.0)

Het SQL-proces zorgt voor de acties die worden ondernomen indien de gebruiker kiest voor het zelf ingeven van SQL-code. Hiervoor moet de gebruiker natuurlijk op de hoogte zijn van de SQL-procedures en opbouw. In dit scenario kunnen perfect complexe opdrachten worden ingevoerd. Dan wordt de ingegeven zoekopdracht [query] doorgegeven aan de IBM DB2-server *sink*. De databank zal op basis van de geleverde opdracht bepaalde records selecteren.

Formatteer digit stream (6.0)

Dit proces ontvangt, ongeacht of de zoekopdracht door de gebruiker zelf of door de applicatie is geschreven, het resultaat van de databankserver. Dit resultaat bestaat gewoon uit een stroom van *bits*. Deze stroom wordt door de applicatie correct ontvangen en omgezet naar symbolen en letters. De gegevens worden gerangschikt op de manier die door de gebruiker is gekozen. De gebruiker kan hiervoor kiezen uit een aantal weergavemogelijkheden.

Rapporteer resultaat (7.0)

Het resultaat wordt teruggezonden naar de computer van de gebruiker. In dit proces zal alles gedecodeerd en weergegeven worden op de computer van de gebruiker in het gewenste formaat.

4.8.4 Logicaweergave

In dit project wordt gebruik gemaakt van het Model View Controller [MVC]-model. Het is een techniek die een complete afzonderlijke opbouw van logica en presentatie nastreeft.

De *view* geeft de gebruikersinterface weer. De taak van de *controller* bestaat onder andere uit het verzorgen van de coördinatie tussen de interface en het achterliggende model. De *controller* is verantwoordelijk voor de overdracht van de in de presentatie ingevulde gegevens aan het model. Het model voert haar opdracht uit en geeft de resultaten terug aan de *controller* die vervolgens uitzoekt

wat de gebruiker wenst te zien. De gebruiker ontvangt een nieuw scherm met de gewenste informatie.
De kracht van deze opbouw komt naar voren in de mate van flexibiliteit die wordt gerealiseerd. De opsplitsing van model, presentatie en *controller* laat toe alle onderdelen apart te ontwikkelen. Het model hoeft daarbij geen weet te hebben van specifieke presentatie-aangelegenheden. Het model bevat een reeks van functies die enkel aangeroepen worden door de *controller*. Je kunt dus perfect een nieuwe weergave ontwikkelen met overeenkomstige *controller* en deze laten samenwerken met het bestaande model. [Buschmann, 2001]

Objectgeoriënteerd programmeren [OOP] alsook de MVC-techniek promoten het hergebruik van code voor nieuwe doeleinden. Bij de creatie van deze webapplicatie wordt getracht dit gegeven toe te passen. In het project wordt een nieuwe presentatie gemaakt voor een bestaand *business*-model. De logica in de applicatie bestaat vooral uit regels die de opbouw van de SQL-opdracht, op basis van de invoer van de gebruiker, realiseren. Deze logica voor het samenstellen van SQL-zoekopdrachten wordt overgenomen van AcciQuest aangezien dezelfde SQL-opbouw zal worden toegepast in de nieuwe applicatie. De presentatie wordt gemaakt in een programmeertaal die leesbaar is voor de zoekmachine van de gebruiker, zodoende internetgebruik toe te laten. De *controllers* moeten geschreven worden in het teken van de vereisten van de nieuwe applicatie. Zij zullen de communicatie met het achterliggende model verzorgen. In beide gevallen kan geen gebruik gemaakt worden van componenten uit AcciQuest.

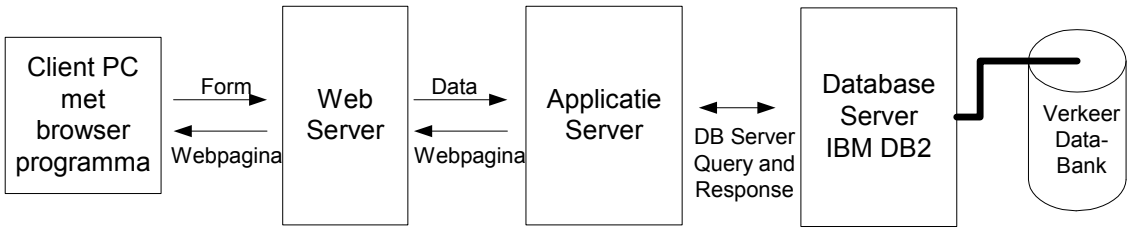
4.8.5 Conceptuele dataweergave: ERD

De verkeersdata, die in de applicatie wordt gebruikt, is ondergebracht in een IBM DB2-omgeving. Dezelfde databank wordt nu al gebruikt door de applicatie AcciQuest. Omdat deze databank bestaat, hoeft in dit project geen aandacht besteed te worden aan het structureren van de data en het bepalen van de kenmerken van de data. De, in de ontwikkelomgeving van de applicatie, actieve

databank betreft een servereditie van voorgenoemd programma dat een overeenkomstige structuur vertoont met de professionele DB2-servers van IBM. Hierdoor kan de databank zonder problemen worden overgeheveld naar een gespecialiseerde databank [DB]-server. Een databankspecifieke server zal, wanneer het systeem operationeel wordt, leiden tot betere prestaties

4.8.6 Selecteren van de beste ontwerpstrategie: hardware

Voor deze opdracht wordt de voorkeur gegeven aan de *client*/serverarchitectuur. Dit model is de meest uitgewezen keuze omdat enkele gewenste functionaliteiten, zoals connectie over het internet, het beste via deze architectuur kunnen worden verwezenlijkt.



figuur 4.4 Three-Tier Architectuur (Panko, 2003)

De gewenste configuratie, zoals aangegeven in figuur 4.4 is opgebouwd uit drie onderdelen die enige verdere toelichting behoeven. Deel één van dit model wordt ingevuld door de computer van de gebruiker. Deze bevat een zoekmachine die tracht contact op te nemen met de webserver via een netwerk. Het tweede deel omvat de webserver en de applicatieserver waarop de logica van de applicatie wordt geplaatst, alsook de *controllers*. Deze netwerkcomputer heeft als taak de coördinatie te verzorgen tussen de zoekmachine van de gebruiker en de data. De server bevat alle HTML-pagina's die in het programma voorzien zijn. Op het vlak van applicatieservers zijn meerdere mogelijkheden beschikbaar. Applicatieservers worden daarbij regelmatig gecombineerd met een webserver. De derde

component kan een ondernemingsnetwerk, een mainframe, een DB-server of een andere server zijn. In dit project wordt dit onderdeel verzorgd door een IBM DB2-server. Aangezien de databank al bestaat in het type DB2, ligt de keuze voor dit type server vast.

4.8.7 Selecteren van de beste ontwerpstrategie: software

De keuze voor programmeertaal is gevallen op Java. Een belangrijke reden hiervoor is het feit dat AcciQuest werd geschreven in Java. Het opnieuw kiezen voor Java biedt de ontwikkelaar de mogelijkheid bepaalde delen van de code van AcciQuest, in het teken van objectgeoriënteerd programmeren en de MVC-techniek, te hergebruiken. Daarnaast biedt Java met de J2EE-specificatie een uitstekende basis voor de ontwikkeling van webapplicaties. Java is bovendien één van de meest gebruikte talen voor het ontwerpen van internetgebaseerde applicaties.

5. Ontwerp

5.1 Inleiding

In het ontwerpstadium van het SDLC-model worden de technische aspecten van het toekomstige informatiesysteem gerealiseerd. Deze kunnen volgens Hoffer et al. [2002] worden gegroepeerd in drie activiteiten. Ten eerste worden de in- en uitvoervelden geschetst aan de hand van verkregen informatie van de toekomstige gebruikers. Hierbij wordt rekening gehouden met een reeks richtlijnen die trachten de schermen en weergave van informatie zo overzichtelijk mogelijk te realiseren. Een tweede activiteit, eigen aan deze fase is de concrete uitwerking van de datastructuur tot een volwaardige databank. Echter dit project hoeft niet te worden voorzien van een databank aangezien het programma gebruik maakt van een bestaande databank. Er moet wel aandacht worden besteed aan het zeker stellen dat de verschillende onderdelen van de applicatie kunnen communiceren met de databank. Bovendien moet vanuit de applicatie een verbinding worden gelegd met de databank. Een derde activiteit die binnen deze fase wordt gesitueerd, omvat het beschrijven van de componenten en objecten, samen met hun functie. Voordat hiermee wordt begonnen, zal in volgende paragraaf een korte toelichting worden gegeven over Java servlets en JSP [Java ServerPages].

5.2 Theoretisch kader: servlets en JSP

De kenmerken van de programmeertaal Java en de J2EE-specificatie werden in de analysefase al bondig weergegeven. J2EE is een interessante keuze voor de ontwikkeling van webapplicaties omdat een bepaald type klasse gebruikt kan worden. Deze klasse is bekend onder de naam servlet. Servlets vormen de kern van elke J2EE-specificatie. Alle machines die de J2EE-specificatie ondersteunen, zullen servlets accepteren en implementeren. Servlets kunnen volgens Hall [2000] worden beschouwd als het antwoord van Java op Common Gateway Interface

[CGI]. Dergelijke programma's opereren vanuit netwerkcomputers en maken webapplicaties een realiteit. Een webserver wordt geplaatst tussen de gebruiker enerzijds en ondernemingsapplicaties met bijbehorende data anderszijds. Algemeen gesteld, verzorgen servlets de communicatie tussen een zoekmachine of een andere HyperText Transfer Protocol [HTTP]-*client* enerzijds en databanken of applicaties op de HTTP-server anderzijds. Volgende alinea gaat gedetailleerd in op de activiteiten van servlets.

De taken die servlets binnen een J2EE-applicatie vervullen zijn niet alleen talrijk, ze zijn ook uiteenlopend. Servlets ontvangen en bekijken alle informatie die door de gebruiker wordt verstuurd. Meestal is deze informatie ingegevoerd in een formulier op een website. Ook kan deze informatie afkomstig zijn van een Java applet of zelfs van een specifiek ontwikkeld programma. Een servlet bekijkt alle additionele informatie die samen met het verzoek werd verzonden. De servlet geeft bepaalde parameters mee aan de berichten die het verzendt. De zoekmachine van de gebruiker bepaalt op basis van deze waarden welk soort pagina wordt teruggestuurd. Welke informatie de servlet in de berichten meezendt, wordt vastgelegd tijdens de ontwikkeling. Deze berichten geven bijvoorbeeld weer welke opties zijn geactiveerd. Bepaalde informatie kan naar de gebruiker worden verstuurd in de vorm van een *cookie*. Een *cookie* zal, de volgende keer dat de gebruiker de site bezoekt, door de applicatie worden opgehaald. De informatie uit de *cookie* wordt gebruikt voor het faciliteren van de diensten aan de gebruiker. De servlet verzorgt vervolgens de communicatie met de databank en genereert hieruit de gevraagde resultaten. Een servlet kan overigens ook berekeningen uitvoeren met data afkomstig van de gebruiker, zonder dat een databank dient te worden geraadpleegd. In de code van de servlet wordt weergegeven hoe de opmaak van het terug te sturen document met de resultaten eruit moet zien. In de meeste gevallen zal hiervoor HTML-code worden opgenomen in de servlet-code. Het document kan in verschillende soorten formaten gestuurd worden. Enkele mogelijkheden zijn: tekst, binair of een gecomprimeerd formaat. [Hall, 2000]

Servlets worden geïmplementeerd in een webapplicatie om de dynamische inhoud van de webpagina's te genereren. De gebruiker bepaalt met andere woorden welke informatie wordt getoond op het scherm in de applicatie.
Servlets worden niet beschouwd als een goed alternatief voor het creëren van statische inhoud binnen een webpagina. HTML-code is hiervoor meer geschikt. Dit belet niet dat beiden overigens perfect kunnen worden gecombineerd. Sommige pagina's, zoals de openingspagina, worden bijvoorbeeld geschreven in HTML. Nadat de gebruiker zich aanmeldt in het systeem, wordt gebruikerspecifieke informatie getoond die via servlets wordt gegenereerd. HTML-code kan echter niet gecombineerd worden met servlet-specifieke Java-code op eenzelfde pagina. Een pagina wordt in deze opstelling dus ofwel voorzien van statische ofwel van dynamische inhoud. Een kleine hoeveelheid dynamische inhoud impliceert dus een volledige pagina in Java-code. [Hall, 2000]

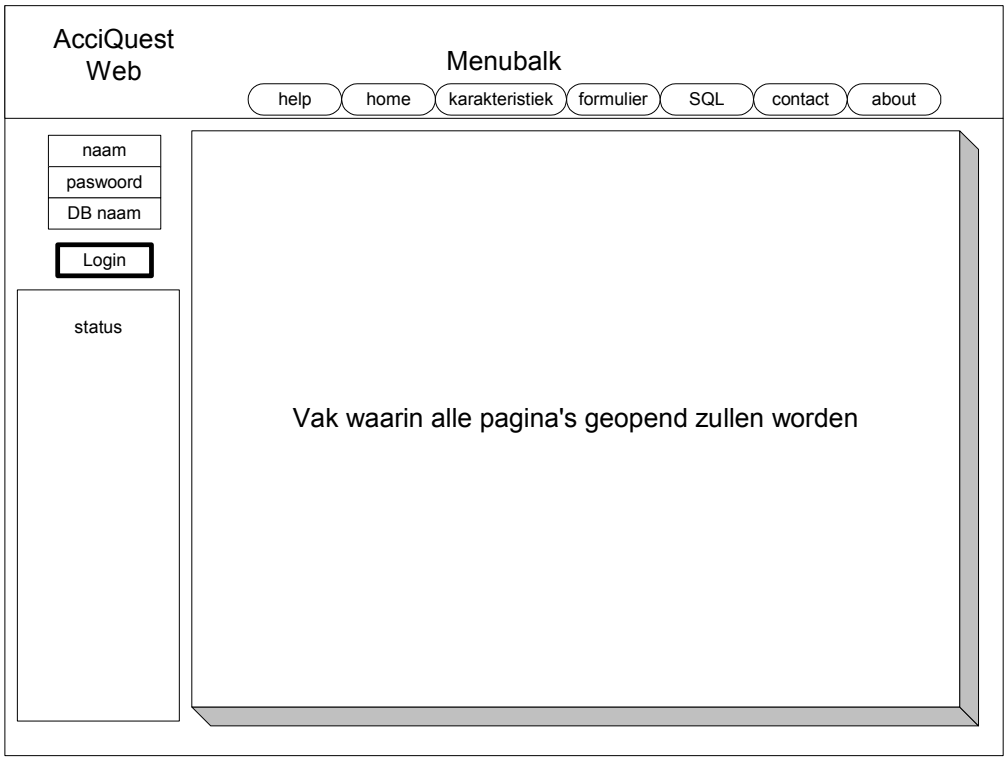
De ontwerpers van de J2EE-specificatie zijn aan deze problematiek tegemoet gekomen door de creatie van JSP. JSP zijn een verdere evolutie van servlets en werken op dezelfde wijze als servlets maar laten toe statische HTML code te combineren met dynamische inhoud die gegenereerd wordt door Java servlets. Op één en dezelfde pagina kan dus zowel Java-code als HTML-code voorkomen. Een JSP komt nog het meeste overeen met een standaard HTML-pagina. Deze manier van werken laat toe, beide onderdelen afzonderlijk te creëren. Het merendeel van de pagina bestaat uit statische HTML die onveranderd naar de *client* wordt gestuurd. De onderdelen die dynamisch gegenereerd worden, zijn gemarkeerd met special HTML-achtige *tags*. Deze delen worden door de webserver vertaald voordat de pagina wordt doorgestuurd naar de gebruiker. [Hall, 2000]

5.3 Toepassing op het project

5.3.1 Schetsen van de schermen

De homepagina van de applicatie

De eerste pagina van de website, zoals in onderstaande figuur 5.1 wordt weergegeven, ontvangt de gebruiker met algemene informatie. Bovenaan de pagina bevindt zich een menubalk. Deze zal altijd beschikbaar zijn voor de gebruiker. Alle pagina's zullen in vakken [frames] geopend worden die links en onder gesitueerd zijn. Door gebruik van de menubalk kan de gebruiker zich bewegen doorheen de applicatie. De gebruiker kan altijd terugkeren naar de startsituatie door de knop home in te drukken. De status geeft weer of een verbinding wordt onderhouden met de databank.



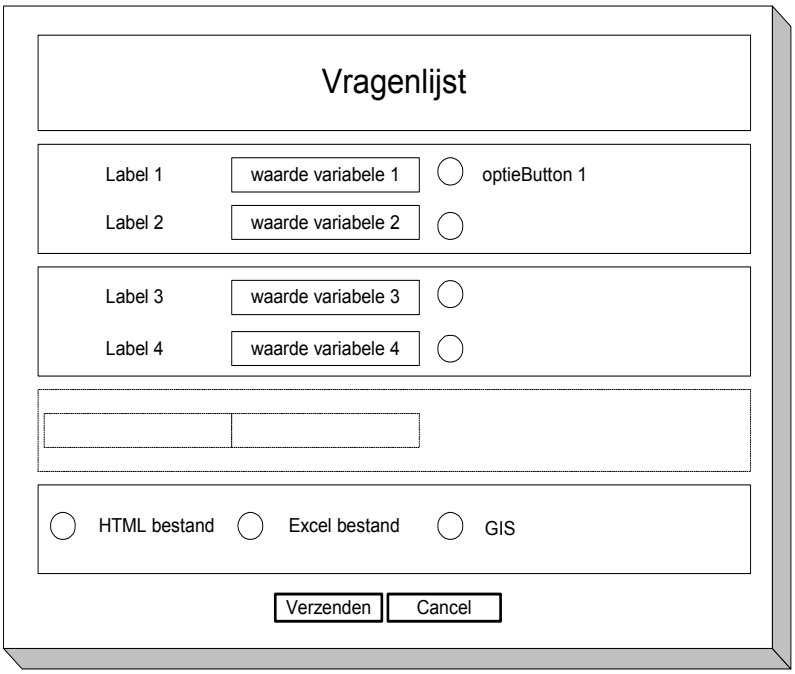
figuur 5.1 Scherm dat de homepagina van de applicatie voorstelt

Boven het statusvak wordt een vak voorzien dat de gebruiker de mogelijkheid geeft om verbinding te maken met de databank. Hier geven de gebruikers enkele parameters in zoals gebruikersnaam, paswoord en eventueel karakteristieken van de databank. Verbinding met de databank wordt gerealiseerd door de login-knop in te drukken, die geplaatst is in het gespecifieerde vak. Op deze pagina zijn drie invulvelden terug te vinden: txtNaam, txtPaswoord, txtDBnaam en een de login-knop: B1. De variabelen zijn in eerste instantie gepositioneerd in een HTML-pagina die de opmaak voor dit vak verzorgt.
Indien de gebruiker kiest voor één van de laatste twee knoppen in de menubalk, dan worden pagina's geopend zonder enige dynamische inhoud. Het zijn de contact- en about-knoppen die aanleiding geven tot het openen van pagina's die zich bevinden op de webserver en geen informatie uitwisselen met andere delen van de applicatie of de databank. Er bevinden zich geen knoppen, invulvelden of een resultatenweergave op deze pagina's. Bij de contact-pagina zal worden getracht een *mail*-component via een *hyperlink* te integreren. Deze component activeert in een apart venster de e-mailapplicatie van de gebruiker met in het 'To:' veld het opgegeven e-mailadres van de IT afdeling en als onderwerp: 'contact AcciQuestWeb'. De *about*-knop geeft een pagina met een aantal kenmerken van de applicatie zelf.

De formulier-pagina

De formulier-pagina, zoals weergegeven in figuur 5.2 op pagina 43, wordt geopend in het centrale vak van de openingspagina en biedt aan de gebruiker een intuïtief interactiemedium in de vorm van een ongevallenformulier. De gebruiker kan aangeven welke criteria gelden bij de vorming van een zoekopdracht en welke waarden hieraan worden toegekend. De gebruiker kiest in de meeste gevallen tussen een aantal voorgeprogrammeerde mogelijkheden. Dit is gedaan om verschillende redenen. Om zo weinig mogelijk foutmeldingen te genereren zullen de meeste keuzes gelimiteerd zijn tot de echte mogelijkheden die ook in de databank zelf zijn opgenomen. Hierdoor heeft de gebruiker een idee van welke data hij wenst te zien. De gebruiker hoeft niet in te schatten op welke wijze de data

in de databank is opgeslagen en welke waarden mogelijk zijn. Alle omzettingen naar het formaat zoals in de databank gebruikt, zal gebeuren door de applicatie. In de databank wordt gebruik gemaakt van cijfers die bepaalde situaties voorstellen. De gebruiker kan een situatie kiezen uit een lijst. De gemaakte keuze wordt door de applicatie omgezet in een cijfer dat overeenkomt met de waarden in de databank. De gebruiker bepaalt ook, door het selecteren van de optieknoppen, welke variabelen moeten worden weergegeven op het scherm.



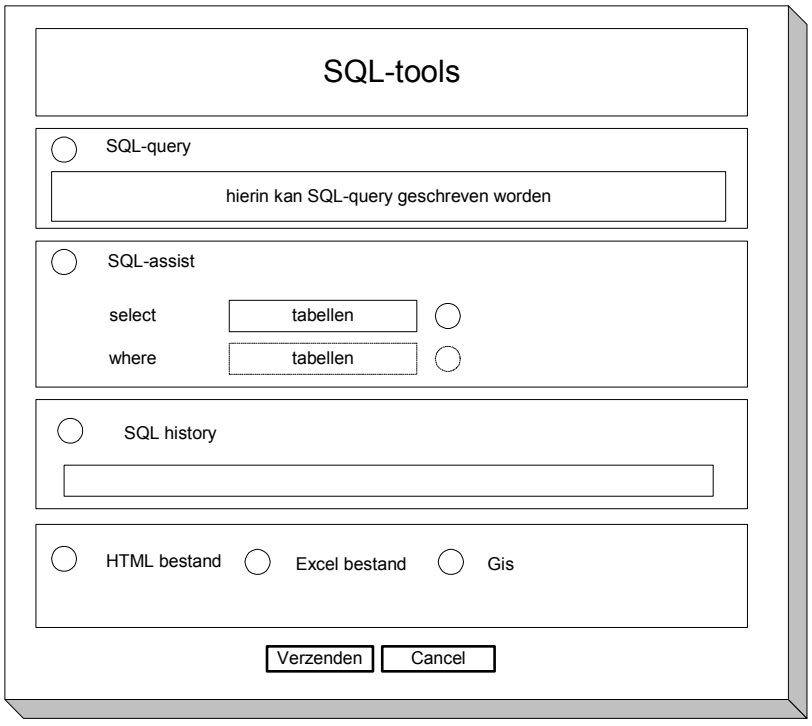
figuur 5.2 Scherm dat de formulierpagina voorstelt

Onderaan de pagina bevinden zich twee knoppen die het verdere verloop bepalen. De cancel-knop maakt alle keuzes die net gemaakt zijn ongedaan. De verzenden-knop gaat de keuzes en de overeenkomstige waarden die werden ingevoerd, meegeven aan het bericht dat teruggestuurd wordt naar de onderliggende applicatie. Dit wordt gerealiseerd door een servlet te starten die de data ophaalt uit het bericht en deze distribueert over de onderdelen van de applicatie. De servlet zal methodes aanroepen van andere klassen die in Java zijn geschreven. Het

maakt hiertoe objecten aan van deze klassen. Door deze methodes zal een SQL-zoekopdracht geformuleerd worden op basis van de invoer van de gebruiker.

De SQL-pagina van de applicatie

De keuze SQL activeert in het onderste vak rechts de SQL-pagina (zie onderstaande figuur 5.3). Deze pagina zal de gebruiker een keuze laten maken uit drie manieren om zelf een SQL-zoekopdracht samen te stellen. De eerste geeft, door middel van een invoerveld, de mogelijkheid aan de gebruiker om een volledige SQL-opdracht te schrijven. Daaronder kan een kleine hulpfunctie gebruikt worden om een SQL-opdracht te realiseren.



figuur 5.3 Scherm dat de SQL-pagina van de applicatie voorstelt

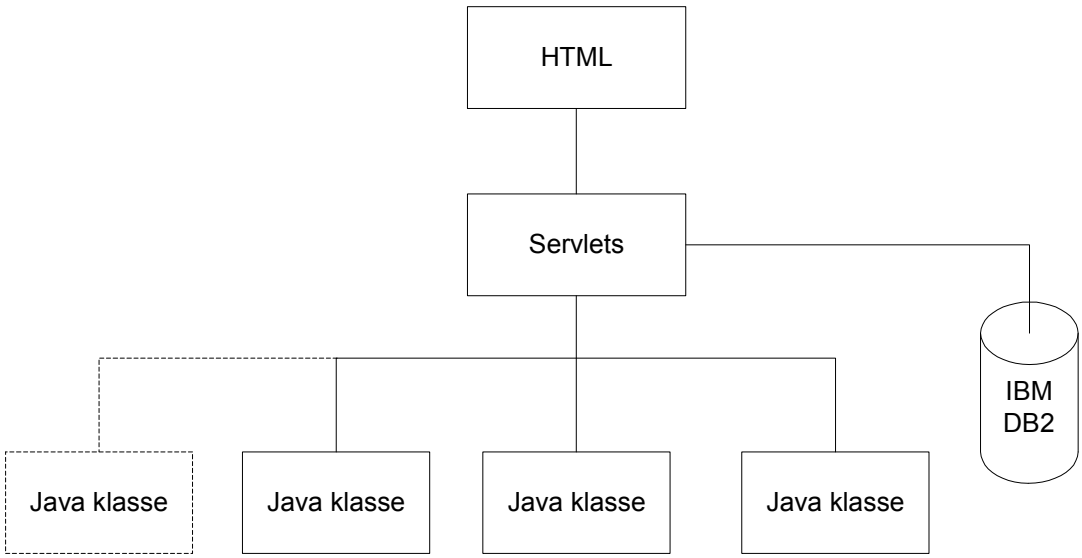
De hulpfunctie geeft de structuur weer zodat de gebruiker nog enkel de gewenste tabellen moet ingeven en de criteria met een waarde. Let wel dat ook hier de gebruiker kennis moet hebben van SQL. De gevormde code moet voldoen aan de

vereisten van de SQL-taal. Anders zal de database niet kunnen ingaan op het verzoek.
Een derde optie biedt de gebruiker de mogelijkheid te kiezen voor een zoekopdracht die tijdens de sessie werd uitgevoerd. Hier kan dus snel opnieuw één van de vorige zoekopdrachten geselecteerd en uitgevoerd worden. Het betreffen zowel opdrachten die tot stand zijn gekomen via de SQL- als de formulier-pagina.

Op beide pagina's, net boven de knoppen, kunnen de gebruikers aangeven op welke manier zij graag de resultaten willen bekijken. De keuze wordt via een variabele meegegeven aan de onderliggende klassen. De geactiveerde servlet zal naargelang de waarde van deze variabele de juiste actie ondernemen. De mogelijkheden zijn voor de twee invoerpagina's identiek: HTML-tabel of Excel-bestand. Alle resultaten worden weergegeven in het vak onder de titel en menubalk. De gebruiker kan kiezen tussen het bestand openen of het bestand opslaan op een bepaalde locatie, indien de Excel-optie wordt gekozen. De knoppen op deze pagina werken op analoge wijze als op het andere invoerscherm. De verzenden-knop zal waarschijnlijk ook dezelfde servlet activeren als bij het andere invoerscherm het geval is.

5.3.2 Impliciete opbouw van het systeem

Zoals in figuur 5.4 (p.46) is aangegeven, is het programma opgebouwd rond een aantal Java-klassen. De presentatie wordt in eerste instantie georganiseerd door HTML-pagina's die zich bevinden op de webserver. De gebruiker beweegt zich doorheen het programma via de menubalk. Indien de gebruiker verzoeken indient bij de applicatie, dan worden de antwoorden teruggegeven door de servlets. De servlets zullen de data uit de databank doorzoeken en de gewenste resultaten in een bepaald formaat terugkoppelen aan de gebruiker. De servlets vormen de kern van de applicatie en maken gebruik van andere, gewone Java-klassen, om de doelstellingen van de applicatie te bereiken.



figuur 5.4 Structuur van de applicatie AcciQuestWeb

De gewone klassen vormen de logica binnen de applicatie. Zij bereiden, op aansturen van de servlets, de SQL-componenten voor elke tabel in de databank voor. De componenten worden door de servlets samengevoegd en verder verwerkt, met behulp van andere methodes van de gewone Java-klassen. De antwoorden op de verzoeken van de gebruiker worden geformuleerd door de servlets. Deze klassen creëren hiervoor nieuwe HTML-pagina's waarbij de dynamische inhoud toegevoegd is, in de pagina zelf. De dynamische inhoud is afkomstig uit de databank en wordt samengesteld door de servlet op basis van een SQL-zoekopdracht. Deze kan door de gebruiker zelf worden ingegeven of, via het ongevallenformulier door de applicatie worden samengesteld.

6. Implementatie en onderhoud

6.1 Inleiding

De laatste twee stadia van de systeemontwikkelingcyclus [SDLC] worden samen in dit hoofdstuk behandeld. Het betreffen de stappen: implementatie en onderhoud. Beide fasen worden gedetailleerd uitgelegd, waarna het theoretische concept van webapplicaties aan bod komt. Dit theoretische concept wordt toegepast op het informatiesysteem en de fasen worden concreet voor deze applicatie uitgewerkt.

6.2 Implementatie

Hoffer et al. [2002] onderscheidt in de implementatiefase vier taken. De hoofdtaak wordt geïdentificeerd als de creatie van een werkend informatiesysteem dat is gebaseerd op alle informatie en uitgewerkte schema's uit voorgaande stappen. Dit houdt in: het schrijven van de code van het programma. De implementatiefase moet tevens een installatieprocedure opleveren die toelaat de applicatie te initialiseren in de bestaande of nieuw aangekochte informatica-infrastructuur. Een volgend onderdeel van de fase betreft het uitvoerig toetsen van de code waarbij de meeste fouten die nog in de applicatie voorkomen, geneutraliseerd moeten worden. Een vierde taak bestaat uit het voorbereiden van de organisatie op het gebruik van het programma. Een gebruikershandleiding maakt de toekomstige gebruikers wegwijs in het programma. Daarenboven wordt een technische handleiding geschreven die toekomstige programmeurs binnen de organisatie vertrouwd moet maken met de meer technische aspecten van de applicatie. De technische handleiding geeft aan hoe het programma is gestructureerd, welke de verschillende componenten zijn, welke procedures en functies onderdeel uitmaken van de toepassing.

6.3 Onderhoud

De onderhoudsfase groepeert een aantal activiteiten die betrekking hebben op de opvolging van het programma, nadat deze in gebruik wordt genomen. Het doel van de fase is het oplossen van problemen die zich kunnen voordoen bij het gebruik. Er wordt getracht de fouten, die nog kunnen voorkomen in het systeem, te ondervangen. Een andere activiteit van de onderhoudsfase betreft het aanpassen van het systeem aan veranderende omstandigheden waarmee de organisatie geconfronteerd zal worden in de toekomst. De andere stadia van het SDLC-model kunnen hier opnieuw worden toegepast om deze vereisten in concrete programma-eigenschappen om te zetten. Nieuwe mogelijkheden worden onderzocht op hun haalbaarheid, kosten en opbrengsten worden geschat, het ontwerp van het systeem wordt aangepast en als laatste wordt de implementatie van het vernieuwde systeem doorgevoerd.

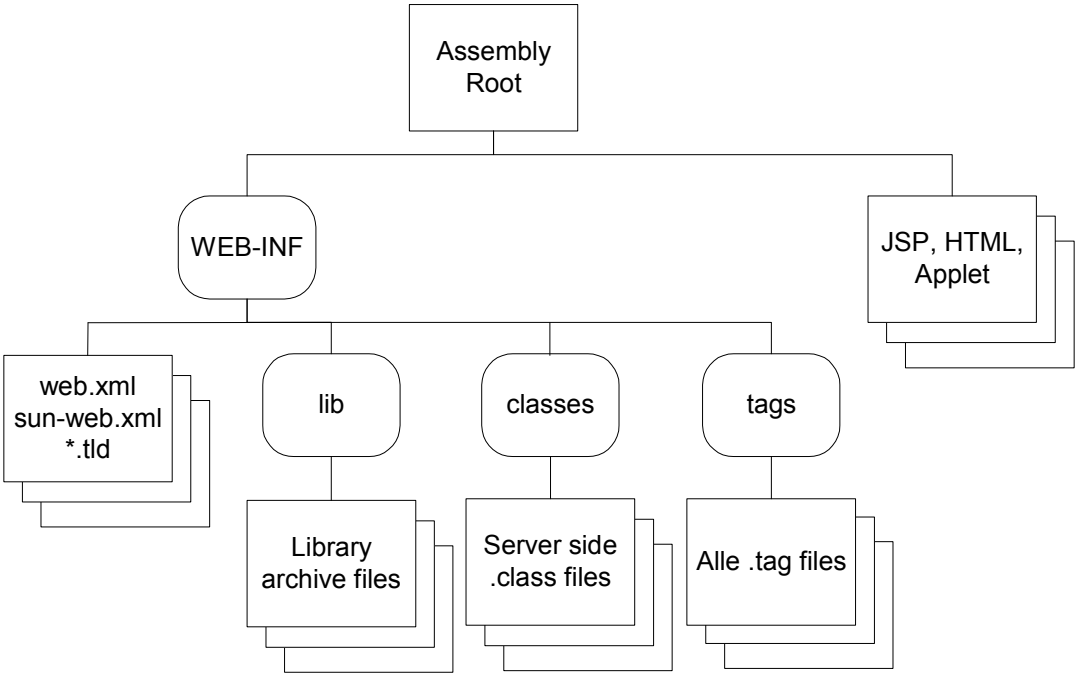
6.4 Theoretisch aspect: webapplicatie

6.4.1 Structuur

Het woord webapplicatie is al een aantal keren voorgekomen in deze tekst. Een algemene definitie wordt gegeven door Armstrong et al. [2004]: *“Een webapplicatie is een dynamische uitbreiding van een web- of applicatieserver*”.

Er bestaan twee typen van webapplicaties. Een presentatiegeoriënteerde webapplicatie genereert interactieve webpagina’s waarin meerdere *markup*- talen zoals HTML of Extended Markup Language [XML] kunnen voorkomen. Tevens mogelijk is dynamische inhoud die wordt gerealiseerd als antwoord op verzoeken vanuit de gebruiker. De dynamische uitbreiding van webapplicaties wordt binnen de programmeertaal Java verzorgd door Java servlets of JSP. Bij dienstengeoriënteerde applicaties wordt deze rol vervuld door interactiepunten van een *Web Service*. [Armstrong et al., 2004]

Onderstaande figuur 6.1 geeft aan uit welke componenten een webapplicatie is opgebouwd. Een webmodule komt overeen met het kleinst mogelijke onderdeel dat kan worden geïnstalleerd [deploy]. Een J2EE-module komt overeen met een webapplicatie zoals gedefinieerd in de Java servlet specificatie.



figuur 6.1 Folderhiërarchie van een webapplicatie (Armstrong et al., 2004)

6.4.2 Applicatiepakket

Twee mogelijkheden worden aaangeboden, naargelang het type server, om een webmodule te installeren. Een eerste methode betreft het plaatsen van een losse bestandsstructuur van de applicatie in een specifieke folder van een server. Een andere mogelijkheid wordt geboden door het Web Archive Resource [WAR]-bestandstype. Hierbij wordt de webmodule ingepakt in een speciaal type Java Archive Resource [JAR]-bestand, gekend onder de naam WAR. Het grote voordeel van het werken met WAR-bestanden is de overdraagbaarheid. WAR-bestanden zijn webapplicaties, in Java-taal geschreven, die door alle ontwikkelaars van

serverhardware worden erkend. Het bestand kan dus zonder enige wijzigingen worden geplaatst in om het even welke webomgeving die conform de Java servlet-specificaties opereert.

6.5 Toepassing op het project: technische handleiding

6.5.1 Inleiding

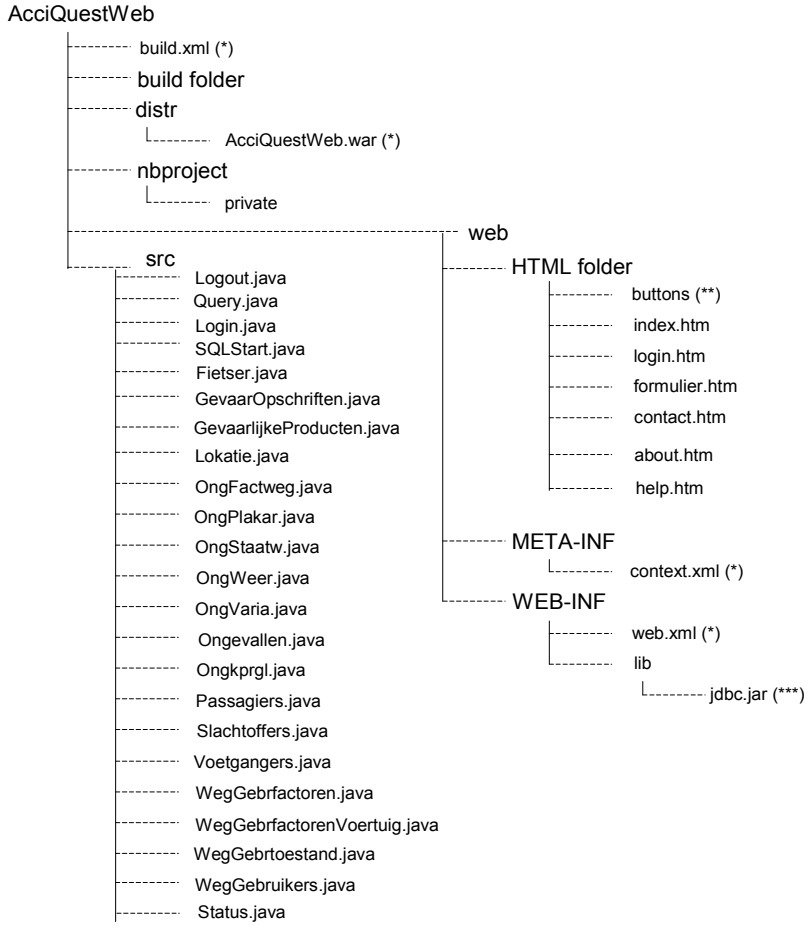
De technische handleiding die hier wordt gepresenteerd, levert een deel van de benodigde informatie voor programmeurs. Andere delen van de uitleg zijn terug te vinden in de vorige hoofdstukken. Daarenboven is een gedetailleerde beschrijving van de methodes, functies, objecten en klassen toegevoegd aan de code in de vorm van commentaarregels. Deze drie onderdelen vormen samen één complete technische handleiding. De structuur van de webapplicatie wordt weergegeven in figuur 6.2 op pagina 51.

In de applicatie komen Java-klassen voor, een aantal HTML-pagina's en enkele bijzondere bestanden. In volgende paragrafen worden al deze typen van bestanden beschreven. De Java-klassen kunnen worden onderverdeeld in twee categorieën. Een eerste categorie betreft de servlets. De tweede groep bevat klassen die de logica van de applicatie voorstellen. Zij worden geactiveerd vanuit de servlets die bij de aanroeping, de waarden van de parameters meegeven. Alle Java-klassen die de logica beschrijven hebben een analoge opbouw. De algemene opbouw van deze categorie van klassen wordt aangegeven, in de plaats van elke klasse afzonderlijk te bespreken.

6.5.2 HTML-pagina's

De HTML-pagina's verzorgen de gebruikersinterface van de applicatie. Zij werden toegelicht in hoofdstuk 4 bij het beschrijven van de schermen. De HTML-pagina's

bevatten een reeks variabelen die in de pagina zelf aangemaakt worden. De waarden die deze variabelen aannemen hangen af van de keuzes van de gebruiker. De servlets zullen deze waarden ophalen van de HTML-pagina's.



figuur 6.2 Folderhiërarchie AcciQuestWeb (*) speciale bestanden, (**) verzameling, (***) driver

6.5.3 Java-klassen: servlets

Servlets zijn specifiek aan applicaties die voldoen aan de J2EE-specificatie. In de applicatie AcciQuestWeb worden vijf servlets gebruikt. Servlets zijn uitbreidingen van de HTTPServlet-klasse uit de bibliotheek javax.servlet.*. Bij alle gebruikte servlets worden zowel de doPost- als doGet-methodes geïmplementeerd. Beide methodes gebruiken twee parameters, request en response. De methode

processRequest werd in het leven geroepen om beide ‘doXx’-methodes op te vangen. De ‘doXx’-methodes roepen de processRequest-methode op waarin alle uit te voeren code wordt geschreven. De ‘doXx’-methodes zelf worden aangeroepen wanneer de gebruiker de servlet activeert. De code van de knop die dit realiseert in de HTML-pagina’s bevat de naam van de methode die gebruikt dient te worden door de zoekmachine.

Servlet implementeren uitzonderingen [exceptions]. Indien bepaalde code niet kan worden uitgevoerd, zal in plaats van een foutmelding een bericht verschijnen. De applicatie hoeft hierdoor niet stopgezet te worden. Bepaalde acties binnen Java verplichten het gebruik van uitzonderingen.

6.5.3.1 Login-servlet

Deze servlet verzorgt de verbinding met de databank. De Login-servlet begint aan haar taak met de creatie van een sessie. De sessie wordt gedetermineerd door de machine van de gebruiker en de server. Enkel de gebruiker die zich heeft aangemeld, kan gebruik maken van zijn of haar sessie. De sessie-identificatie wordt gerealiseerd door de toekenning van een code. Deze code verandert iedere keer dat een gebruiker zich aanmeldt. De servlet haalt vervolgens van de pagina login.htm, de parameters die werden ingevuld in de velden door de gebruiker. Deze waarden worden geplaatst in drie variabelen met de namen: txtNaam, txtPaswoord en txtDBnaam. Aan de aangemaakte sessie worden een aantal attributen verbonden die vanuit andere servlets opgevraagd kunnen worden. Indien men geen gebruik zou maken van sessie-opvolging [session tracking], dan zou voor elk verzoek van de gebruiker een nieuwe connectie met de databank aangemaakt moeten worden. De connection-variabele wordt via de Login-servlet aangemaakt, alsook een statement-variabele van de connectie. Deze gegevens worden als attributen van de sessie meegegeven bij elk bericht dat verzonden wordt. Een resultSet-variabele wordt hier nog niet aangemaakt. Dit vindt plaats in de Query-klasse zelf.

In de code van de servlet wordt meegegeven welke *driver* gebruikt dient te worden bij de verbinding met de databank en waar deze *driver* zich binnen de applicatiestructuur bevindt. Nieuwe bibliotheken worden bovenaan in de code geïmporteerd om de werking met de algemene Java-klassen mogelijk te maken. De servlet tracht een verbinding met de gekozen databank tot stand te brengen door implementatie van een aantal methodes van deze algemene Java-klassen.

6.5.3.2 Logout-servlet

Indien de applicatie een verbinding heeft kunnen verzorgen met de databank, zal dit meegedeeld worden aan de gebruiker. De Login-servlet zal een nieuwe HTML-pagina genereren, die opnieuw voorzien is van een knop die de gebruiker de mogelijkheid biedt zich af te melden. Deze knop activeert een tweede servlet met de naam Logout.java. Deze servlet verbreekt de verbinding met de databank via een aantal close()-methodes. Tevens zal de sessie afgebroken worden. De nieuwe HTML-pagina die door deze servlet gemaakt wordt, heeft dezelfde opbouw als het initiële Login-venster en zal in hetzelfde vak geopend worden.

6.5.3.3 Query-servlet

De kern van de applicatie is de Query.java-servlet. Deze servet ontvangt alle gebruikersinvoer die te maken heeft met de formulier- en de SQL-pagina. Vanuit de Query.java-klasse worden methodes van alle gewone Java-klassen van de applicatie aangeroepen. De servlet begint met het importeren van een reeks Java-bibliotheken. Vervolgens worden een reeks variabelen gedeclareerd. Deze hebben betrekking op klassen die aanwezig zijn in de applicatie. Later in de code zullen objecten van deze klassen worden aangemaakt. Een reeks hulpvariabelen worden hier ook gedeclareerd. Zes methodes of groepen van methodes worden hier beschreven. Het betreffen de belangrijkste methodes die in deze klasse voorkomen.

Methode: processRequest()

De eerste taak binnen de processRequest()-methode is het controleren of een sessie met de gebruiker bestaat, en indien dit het geval is, hiermee contact op te nemen. Indien geen sessie bestaat, kan de Query-servlet niet verder gaan. Ze zal aan de gebruiker vragen om zich aan te melden. Indien wel een sessie bestaat, dan zal de servlet de sessie-attributen raadplegen. Op de twee pagina's, zijnde formulier of SQL, die de Query.java-servlet kunnen activeren, zijn verborgen variabelen aangebracht die aangeven welke pagina de Query.java-servlet heeft geactiveerd. Naar gelang de initialiserende pagina zal actie worden ondernomen. Indien het verzoek afkomstig is van de formulier-pagina, dan zal de methode queryIntuitief() worden aangeroepen. In het andere geval zal de methode queryZelfSql() worden aangeroepen. Hierna zal in beide gevallen de methode execute() starten.

Methode: content()

De methode content() legt de keuze van de gebruiker vast op het vlak van het formaat die de teruggezonden documenten moeten aannemen. Op elke pagina kunnen de gebruikers aangeven in welk formaat zij de gegevens willen bekijken.

Methode: queryIntuitief()

Een volgende methode betreft queryIntuitief() die in eerste instantie objecten aanmaakt van bijna alle andere klassen in het programma. Via deze objecten zullen methodes, die gedefinieerd zijn in hun respectievelijke klassen, kunnen worden uitgevoerd vanuit de servlet. Niet alle methodes zullen gegevens terugsturen naar de servlet. Bij een aantal methodes worden gegevens intern in de klasse gemanipuleerd en op een later tijdstip pas aangeroepen door een andere methode van de klasse of door een methode van de servlet. De methode queryIntuitief() tracht alle invoer van de gebruikers te beheren. Naargelang de keuze van de gebruikers zullen methodes worden aangeroepen. Dit is voor alle invoer analoog. Invoer kan geschieden op basis van tekstvelden, selectievelden en keuzemenu's. Door het gebruik van parameters wordt deze informatie verdeeld

over de verschillende klassen. Voor elk invoerveld kan de gebruiker twee acties ondernemen. Enerzijds kunnen de argumenten worden geselecteerd die de gebruikers op hun scherm willen terugzien. Deze horen thuis in het 'Select'-gedeelte van een SQL-zoekopdracht. Anderzijds kunnen gebruikers voor elk veld aangeven of zij dit veld als criterium wensen te gebruiken en welke waarde hieraan moet worden toegekend. Indien de gebruiker de default-keuze laat staan of het tekstveld leeg maakt, dan zullen deze velden niet worden opgenomen als criteria. De mogelijkheden in de keuzemenu's zijn gecodeerd in de HTML-pagina. Op deze pagina zijn variabelen aangemaakt die de informatie doorspelen aan de servlet. Het laatste feit binnen deze methode is het aanroepen van een andere methode van de klasse namelijk de doQuery()-methode.

Methode: queryZelfSql()

De queryZelfSql()-methode vangt alle verzoeken op, afkomstig van de SQLStart-servlet. Hier wordt verder geen andere methode aangeroepen. De code hier wordt geacht een SQL-conforme zoekopdracht af te leveren. Nadat deze methode is uitgevoerd kan direct worden overgegaan tot de uitvoer van de zoekopdracht via de methode 'execute()'.

Methodes voor opbouw SQL-zoekopdracht

Het doel van deze groep van methodes is het creëren van een SQL-zoekopdracht uit alle informatie die de gebruiker heeft ingegeven. Eerst worden van alle andere klassen methodes aangeroepen om te controleren of de gebruikers wel iets hebben geselecteerd uit de mogelijkheden. Indien dit het geval is, wordt een nieuwe variabele aangemaakt van het type StringBuffer. De naam die wordt meegegeven aan deze variabele is query. De keuze is op een StringBuffer-variabele gevallen omdat hiermee aan een al bestaande StringBuffer nieuwe strings kunnen worden toegevoegd via de methode append(). Deze variabele zal echter van type moeten wijzigen voordat ze tot uitvoer aan de execute()-methode wordt meegegeven. De reden hiertoe is eenvoudig: de methode voor het doorgeven van de opdracht aan de databank gebruikt een String-variabele als

parameter. Drie methodes worden vervolgens aangeroepen. Zij zorgen samen voor de creatie van de SQL-zoekopdracht. Het betreffen de methodes: createSelectQuery(), createWhereQuery(), createFromQuery(). Zij realiseren elk een bepaald deel dat zal terugkomen in de finale SQL-opdracht.

Methode: execute()

Een volgend luik in de code wordt gewijd aan de uitvoer van de servlet. De execute()-methode werd in de processRequest()-methode aangeroepen. De SQL-opdracht wordt in deze methode verzonden naar de DB-server. De oorsprong van de SQL-opdracht is hierbij niet relevant. Beide mogelijkheden worden op dezelfde wijze afgehandeld. In de execute()-methode worden een aantal lussen gedefinieerd die alle data, teruggezonden door de databank in antwoord op het verzoek, doorlopen en weergeven in een overzichtelijke tabel.

6.5.3.4 SQLStart-servlet

De servlet wordt gestart als de gebruiker de knop SQL uit het menu indrukt. De reden om hier een servlet te gebruiken, in plaats van een statische HTML-pagina, is dat een keuzemenu van zoekopdrachten, die al door de gebruiker werden uitgevoerd in deze sessie, op deze pagina moet worden gegenereerd. Andere onderdelen van deze pagina zijn statisch.

6.5.3.5 Status-servlet

Een laatste servlet is de Status.java-klasse. Deze servlet heeft als taak het weergeven van karakteristieken van zoekopdrachten, indien de gebruiker deze aanvraagt. Ook hier wordt gecontroleerd of een sessie actief is. Zoniet kan natuurlijk geen opdracht worden afgeleverd en kunnen ook geen gegevens worden teruggekoppeld.

6.5.4 Andere Java-klassen

Alle andere klassen in de webapplicatie organiseren de logica van het programma. Elk van deze klassen vertegenwoordigt een tabel uit de databank. Ze zullen instaan voor het beheer van de invoer van de gebruiker. Bovenstaande servlets halen de invoer van de gebruikers op uit de invoervelden van de HTML-pagina's. Deze waarden worden toegekend aan variabelen. Vervolgens geeft de servlet deze waarden door aan de juiste klasse. In de klassen worden de delen van de zoekopdracht samengesteld die betrekking hebben op de met de klasse geassocieerde tabel. In de methodes createSelectQuery(), createWhereQuery() en createFromQuery() van de Query.java-klasse worden al deze onderdelen samengevoegd tot één grote SQL-opdracht. Eventueel kunnen subopdrachten worden geformuleerd door de applicatie. In wat volgt wordt een algemene beschrijving gegeven van de opbouw van deze klassen. Zij hebben namelijk alle een overeenkomstige opbouw. Deze manier van werken zorgt voor een eenvoudig uit te breiden logica die goed van pas kan komen indien de databank wordt uitgebreid. Het zorgt tevens voor een overzichtelijke code. Deze klassen zijn terug te vinden in de map src van de applicatiestructuur zoals weergegeven door figuur 6.2 op pagina 51.

Dit type klasse vangt aan met de declaratie van een aantal variabelen waaronder een vector variabele. Drie *booleans* worden ook aangemaakt: m_select, m_touched, m_ingevuld. Eerst wordt een algemene methode geschreven voor de klasse met als naam, de naam van de klasse. Van hieruit wordt een aantal andere methodes aangeroepen. Voor elke kolom in de tabel van de databank wordt een methode geschreven die tracht zo goed mogelijk de invoer van de gebruiker te beheren. Elke kolom, zijnde een attribuut van de tabel, krijgt twee variabelen die van een waarde worden voorzien door de gebruiker. Elke variabele (en dus kolom van de tabel die overeenkomt met de specifieke klasse) heeft daarenboven drie methodes: een set()-methode, een change()-methode, een select()-methode. Verder worden nog een aantal methodes geplaatst die voor de hele klasse

optreden. Deze methodes creëren het aandeel van deze klasse tot de grote SQL-opdracht die geformuleerd dient te worden. Het betreffen een set()- en een set'naam'()-methode. Het 'naam'-gedeelte staat terug op de naam van de klasse.

6.5.5 Andere bestanden

6.5.5.1 build.xml

Het build.xml-bestand levert een aantal doelen [targets] voor het compileren van de applicatie, het installeren van de applicatie, het draaien van de server en het verwijderen van oude kopieën van de applicatie. Bij het compileren van de broncode wordt een /build-map aangemaakt door de build.xml in de structuur van de applicatie. Deze mapstructuur [directory] bevat een exacte kopie van de binaire distributie van de webapplicatie. Ant wordt in het project gebruikt om de compilatie van de Java-broncode en de hierarchie van de applicatie te beheren. Ant is een programma dat werd ontwikkeld door de Apache Software Foundation. Het werkt onder de controle van een build.xml bestand.

6.5.5.2 web.xml

Het web.xml-bestand, ook wel de *depoyment descriptor* genoemd, levert aan de J2EE-server de nodige informatie over de webapplicatie die in hetzelfde webapplicatie-archief vervat zit. Het geeft bijvoorbeeld aan welke pagina de zoekmachine eerst moet openen wanneer de gebruiker de Uniform Resource Locator [URL] ingeeft. Verder wordt in dit bestand aangegeven welke servlets in de webapplicatie worden gebruikt. Een naam wordt geassocieerd met een servlet-klasse. Daarna worden initialisatieparameters verbonden aan de naam.

6.6 Toepassing op het project: gebruikershandleiding

Een volgende activiteit bij de implementatiefase is het schrijven van een handleiding die de toekomstige gebruiker wegwijs maakt doorheen de applicatie. Bij de ontwikkeling van deze applicatie is getracht bij elke stap na te gaan of de

aangeboden interface duidelijk te begrijpen is door een gebruiker. Hoe beter het programma hiermee rekening houdt, hoe eenvoudiger de gebruikershandleiding kan zijn. Doorslaggevend in een dergelijke werkwijze is de intuïtieve opbouw van het programma. Het moet de gebruiker van de eerste keer een gevoel geven dat al jaren met het programma wordt gewerkt. Het scherm is met opzet eenvoudig gehouden zodat de aandacht van de gebruiker niet wordt afgeleid van de essentie. In onderstaande paragrafen wordt elke activiteit die de gebruiker kan ondernemen bondig beschreven.

6.6.1 Aanmelden van de gebruiker

Het aanmelden in het AcciQuestWeb-systeem is een heel eenvoudige activiteit. De gebruiker heeft nodig: een identificatienaam, een paswoord en de naam van de databank waarmee men wenst een verbinding aan te leggen. De gebruiker begeeft zich naar de linkerbovenhoek van het openingsscherm en geeft daar de parameters in. Vervolgens drukt de gebruiker op de knop aanmelden. Vanaf hier neemt het programma het over en zal de gebruiker melden aan het systeem. De gebruiker ontvangt een bericht over de status van het systeem. Indien het aanmelden is geslaagd, dan wordt een nieuwe pagina in deze linkerbovenhoek geopend. Deze pagina geeft aan dat een succesvolle verbinding is gerealiseerd en voorziet de gebruiker van een knop om zich terug uit het systeem af te melden. Indien het programma geen succesvolle connectie tot stand heeft kunnen brengen, zal de applicatie de gebruiker hiervan op de hoogte stellen. De gebruiker krijgt opnieuw de kans om zich aan te melden. Vanaf het moment dat de gebruiker is aangemeld kan van alle mogelijkheden van het programma gebruik worden gemaakt zonder opnieuw de identificatie-kenmerken in te brengen.

6.6.2 Knop: home

Deze knop laat de gebruiker toe, terug te keren naar de openingspagina vanuit welke andere pagina die op dat moment actief is.

6.6.3 Knop: karakteristiek

Deze knop biedt de gebruiker de mogelijkheid karakteristieken van een gevraagde zoekopdracht weer te geven in het statusgedeelte van de pagina. Dit is vooral handig indien complexere criteria worden opgegeven voor het doorzoeken van de data.

6.6.4 Knop: formulier

Deze knop activeert de kernpagina van de applicatie. Het biedt de gebruikers een formulier aan, naar analogie met een ongevalformulier, waar zij aangeven welke criteria belangrijk zijn voor de selectie van de data. Twee aspecten moeten worden aangegeven. Enerzijds duidt de gebruiker aan of bepaalde variabelen moeten worden getoond in de resultatenpagina. Dit gebeurt door het selectievak aan te duiden dat langs elk invoerveld aanwezig is. Anderzijds moeten criteria worden aangegeven op basis waarvan data wordt geselecteerd. Dit kan de gebruiker bereiken door een waarde in het tekstveld of het invoerveld van de variabele in te vullen. Enkel indien een waarde door de gebruiker wordt aangegeven, dus geen leeg veld of de default-waarde, wordt deze variabele meegenomen als criterium in de SQL-zoekopdracht. Op het einde van de pagina vindt de gebruiker twee weergavemogelijkheden waarvan de HTML-pagina als default-keuze is aangegeven. Met deze keuze kan de gebruiker bepalen in welk formaat het resultaat wordt weergegeven. Indien gekozen wordt voor het Excel-formaat, dan kunnen de resultaten van de zoekopdracht opgeslagen worden als xls-bestand op de computer van de gebruiker.

Onderaan de pagina staan twee knoppen: één om de invoer te wissen en één om de door de gebruiker geselecteerde informatie door te sturen naar het achterliggende model van het programma. De gebruiker hoeft geen actie meer te ondernemen. Vanaf het moment dat de knop verzenden wordt ingedrukt, begint het programma met het genereren van een SQL-zoekopdracht die overeenkomt

met de wensen van de gebruiker. Deze zal aan de databank worden meegedeeld. Vervolgens wordt het resultaat naar het scherm van de gebruiker gestuurd. Via de terug-toets van de zoekmachine kan de gebruiker nagaan welke selectie en criteria waren opgegeven. De gebruiker kan ook via de knop ‘karakteristiek’ in de menubalk de karakteristieken van de laatst uitgevoerde zoekopdracht bekijken. Het programma kan meerdere opdrachten uitvoeren zonder te verzoeken naar identificatiegegevens van de gebruiker.

6.6.5 Knop: SQL

Deze knop opent een pagina die de gebruiker toelaat, zelf SQL-opdrachten in te geven. Deze zullen aan het achterliggende model worden verstuurd nadat de verzenden-toets op de pagina wordt ingedrukt. Ook op het scherm aanwezig is een annuleer-knop die de invoer verwijdert. Net zoals bij de formulier-pagina kan hier worden aangegeven in welk formaat de resultaten van de zoekopdracht moeten worden weergegeven. Alle code die hier wordt ingegeven moet trouwens voldoen aan de specificaties van de SQL-taal. Indien dit niet het geval is voor de ingegeven opdracht, dan verschijnt een foutmelding in het status-onderdeel van het scherm.

6.6.6 Knop: help

De help-knop verziet de gebruiker van de gebruikershandleiding terwijl het programma in gebruik is. De tekst zal verschijnen in het algemene venster in het centrum van het scherm. De menuopties zullen hierbij beschikbaar blijven. Indien de gebruiker het help-bestand liever opent in een ander venster, dan kan met de rechter muisknop de optie ‘open pagina in nieuw venster’ worden geselecteerd. Op deze wijze kan het help-bestand worden geraadpleegd zonder de resultaten van de zoekopdracht uit te uitschakelen.

6.6.7 Knop: contact

Deze mogelijkheid laat de gebruiker toe contact op te nemen met de onderhoudsploeg van de applicatie voor eventuele vragen of opmerkingen. De standaard e-mailapplicatie van de gebruiker zal worden gestart met de karakteristieken van de onderhoudsploeg en het onderwerp ingevuld.

6.6.8 Knop: about

De pagina ‘about’ die verschijnt, verschaft de gebruiker extra informatie over het programma en hoe het tot stand is gekomen. Tevens worden medewerkers aangegeven en wordt een lijst meegegeven van mogelijke verbeteringen die aan het programma kunnen worden doorgevoerd.

6.7 Toepassing op het project: testen

Het testen van de applicatie vormt een zeer belangrijke activiteit binnen de implementatiefase van het SDLC-model. Drie soorten problemen worden onderscheiden.

Een eerste categorie betreffen de fouten die in het programma voorkomen door toedoen van de programmeur. Dergelijke fouten in de code worden opgespoord door tests te ontwikkelen die nagaan of de juiste resultaten worden gegenereerd door de applicatie. De tests in dit project concentreerden zich op het vergelijken van de resultaten, gegenereerd door AcciQuest, AcciQuestWeb en de databank zelf, op grond van dezelfde criteria. Dergelijke toetsen werden meermaals uitgevoerd met verschillende criteria en selecties. De resultaten van de verschillende methoden werden vergeleken en bleken overeen te komen.
Uit het testen bleek dat de logica van de applicatie AcciQuest enkele fouten bevatte, waardoor de applicatie voor sommige criteria geen resultaten kon genereren. Het toepassen van de MVC-techniek en dus het overnemen van de

logica van de applicatie AcciQuest zorgden ervoor dat deze fouten tevens voorkwamen in de applicatie AcciQuestWeb. Een lijst van deze fouten is terug te vinden in bijlage A. Aanpassingen van de logica werden in de applicatie AcciQuestWeb doorgevoerd zodat deze fouten hierin niet meer voorkomen.

Een tweede categorie problemen wordt veroorzaakt door de gebruikers van de applicatie. De gebruikers kunnen gegevens invoeren die niet mogelijk zijn of in de code van de applicatie problemen veroorzaken. Deze categorie van problemen wordt aangepakt tijdens het ontwerp van de applicatie door het opnemen van foutafhandelingsprocedures [error-trapping] enerzijds en het beperken van de invoermogelijkheden van de gebruiker anderzijds. Potentiële fouten worden vermeden door de expliciete invoer van de gebruikers tot een minimum te beperken. In de meeste gevallen werd geopteerd voor keuzemenu's in de HTML-pagina's die voorzien in de invoermogelijkheden van de gebruiker.

Een derde categorie fouten is eigen aan de structuur van webapplicaties en kan moeilijk worden aangepakt vanuit de applicatie zelf. Fouten kunnen zich manifesteren op tal van locaties. Denk hierbij aan een weigering van toegang tot de server of de onmogelijkheid om een verbinding te realiseren met de databank of server waarop de databank is gelocaliseerd. De applicatie, de server of de databank kan voor onderhoud onbeschikbaar zijn voor gebruikers. Tevens kunnen problemen ontstaan in verband met de J2EE-server waarop de webapplicatie draait. Problemen die gesitueerd zijn bij de internetleverancier van de gebruiker of bij één van de *backbone-carriers* die de communicatie over lange afstanden verzorgen, zijn ook mogelijk.

Voordat deze applicatie operationeel kan worden ingezet, dient nog een testronde te worden georganiseerd. Enerzijds moeten hierin controletesten van de applicatie worden uitgevoerd door andere personen dan de ontwikkelaars. Anderzijds moet de professionele omgeving waarin de applicatie wordt geplaatst, worden getest.

6.8 Toepassing op het project: installatieprocedure

Installatie van een webapplicatie kan een gecompliceerde aangelegenheid zijn die betrekking heeft op meerdere machines. In het kort wordt de omgeving weergegeven waarin dit project werd ontwikkeld. Vervolgens zal een operationele omgeving worden beschreven. De paragraaf wordt afgesloten met de installatie-procedure.

6.8.1 Ontwikkelomgeving

De omgeving waarin deze applicatie werd ontwikkeld, bestond uit drie onderdelen. De data waar het programma gebruik van maakt, worden geplaatst op een databankserver. In dit project wordt deze rol vervuld door een IDM DB2® Universal Database™ 8 Server Editie. Het tweede onderdeel van een webapplicatie, de web- en applicatieserver, wordt in dit project door de Tomcat 5.0.28 uitgevoerd. Dit is een applicatieserver voor Java-applicaties die tevens de Apache-webcontainer bevat die de taak van webserver op zich neemt. Het is een gratis systeem dat ideaal is voor de ontwikkeling van een webapplicatie en wordt geïnstalleerd op een desktoptoestel. Om een zo breed mogelijke garantie te geven dat de applicatie werkt, zijn daarnaast een aantal andere servers getest. Op deze servers werkt het systeem ook naar behoren. Het betreffen de servers: Sun Java™ System Application Server Platform Edition 8 en de Java™ Web Services Developer Pack 1.5. Hoe deze systemen precies worden bestuurd, is terug te vinden in de gebruikershandleidingen van deze servers. De server werd vervolgens verbonden aan een Integrated Development Environment [IDE]-programma dat de ontwikkelaar assisteert bij het schrijven van programma's. In dit project werd Sun NetBeans IDE 4.0 gebruikt. De derde component in de webapplicatie is de zoekmachine van de gebruiker. Op het vlak van zoekmachines zijn de meest gebruikte programma's getest. Het gaat hier over Internet Explorer van Microsoft, Netscape en Mozilla FireFox. In de ontwikkelomgeving is de webapplicatie terug te

vinden op het volgende adres: http://localhost:8084/ AcciQuestWeb/. ‘Localhost’ verwijst naar de server die op de machine geïnstalleerd is.

6.8.2 Operationele omgeving

Een geschikte plaats wordt geïdentificeerd voor elke component van de webapplicatie. De data dient geplaatst te worden op een professionele databank-server die het IBM DB2-formaat ondersteunt. Een mogelijkheid is hier een professionele DB2 server van IBM. De applicatie kan worden geplaatst op een commerciële web/applicatieservercombinatie. Enkel de gebruikerscomponent ondervindt geen veranderingen. De coördinaten van de website worden doorgegeven aan de gebruikers die ze vervolgens ingeven in hun zoekmachines. Indien een andere omgeving wordt gebruikt dan de ontwikkelomgeving, dan dienen de ROOT- en HOME-coördinaten (CLASSPATHS) van alle andere onderdelen die de applicatie nodig heeft, zoals de Java installatiefolder en de serverinstallatiefolder, te worden aangepast.

6.8.3 Installatie AcciQuestWeb

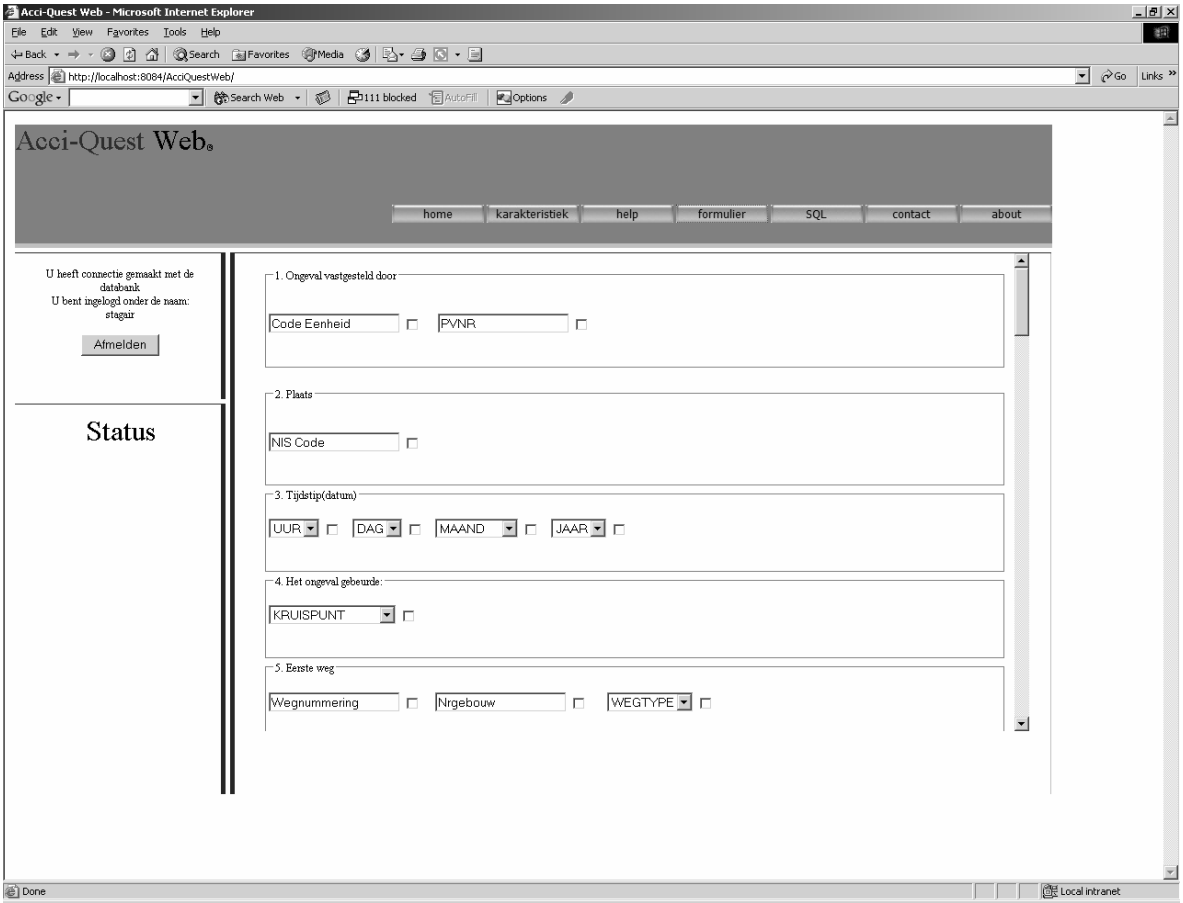
Installatie van het programma wordt gerealiseerd door in de geselecteerde server aan te geven waar de losse applicatiestructuur zich bevindt in het geheugen van de computer. Een alternatief is de creatie van een WAR-bestand dat geplaatst wordt in een speciale folder van de serverinstallatie. Tijdens de installatie van dit bestand op een applicatieserver zal een *runtime deployment descriptor* worden gestart. Dit is een XML-bestand dat informatie bevat over: het in kaart brengen van de middelen die door de applicatie worden gebruikt en de *context root* van de applicatie.

6.9 Onderhoud

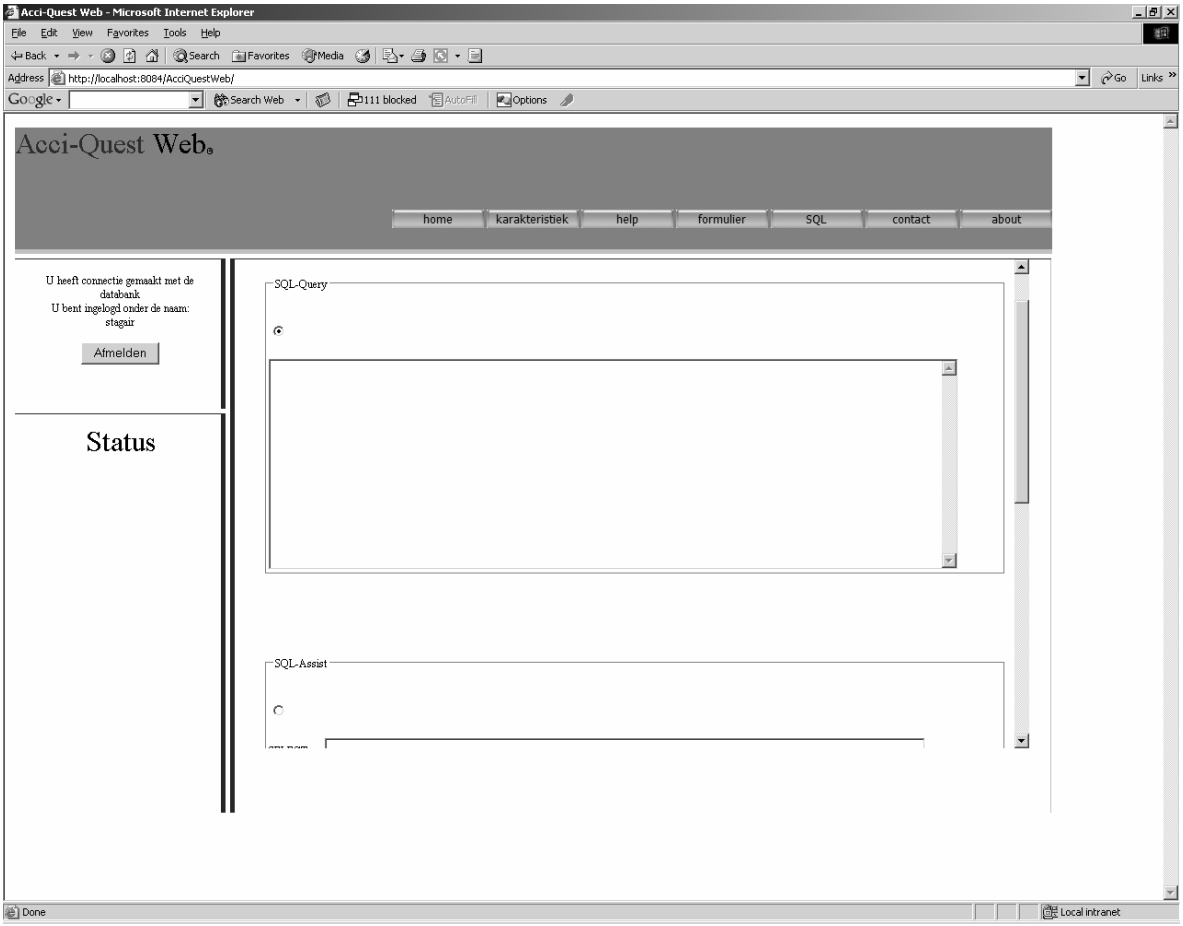
Op de laatste fase van het SDLC-model wordt hier slechts kort ingegaan. Een mogelijkheid is in het systeem opgenomen voor gebruikers om direct contact op te nemen met de diensten die instaan voor het onderhoud van de applicatie. Indien gebruikers suggesties hebben over nieuwe mogelijkheden die toegevoegd kunnen worden of eventueel verbeteringen, dan kan via dezelfde wijze contact worden gezocht. Bij de keuze van programmeertaal is rekening gehouden met het onderhoudsaspect. In Java geprogrammeerde applicaties hebben een goede historie van betrouwbaarheid [Hall, 2000]. Door deze taal te selecteren is gekozen voor een taal die één van de meest gebruikte talen is bij de ontwikkeling van webapplicaties. Dit zal ervoor zorgen dat onderhoud van de applicatie kan worden uitgevoerd door een bestaande onderhoudsgroep zonder dat extra opleiding nodig is. De belangrijkse taak binnen de onderhoudsfase betreft de verdere ontwikkeling van de applicatie om deze te laten voldoen aan veranderende omstandigheden waarin de organisatie zich bevindt. Een SDLC-stappenplan kan bij deze veranderingen opnieuw toegepast worden om deze aanpassingen door te voeren. In het slotstuk van deze tekst worden enkele voorstellen gepresenteerd ter uitbreiding van AcciQuestWeb.

7. Resultaten van het project

In dit hoofdstuk wordt aandacht besteed aan het bespreken van de resultaten die in dit project werden gerealiseerd. Dit gebeurt door enkele voorbeelden uit te werken. Vier zoekopdrachten worden samengesteld, drie op basis van de formulier-pagina en één op basis van de SQL-pagina. De figuren [screenshots] 7.1 op pagina 67 en 7.2 op pagina 68 geven een voorstelling van de verschillende pagina's die worden gebruikt. Tevens zijn figuren opgenomen die een voorbeeld geven van de gegenereerde resultaten. Niet alle *records* van het resultaat worden weergegeven aangezien de databank omvangrijk is en zoekopdrachten kunnen leiden tot omvangrijke resultaten.



figuur 7.1 Screenshot van de formulier-pagina van de applicatie AcciQuestWeb



figuur 7.2 Screenshot van de SQL-pagina van de applicatie AcciQuestWeb

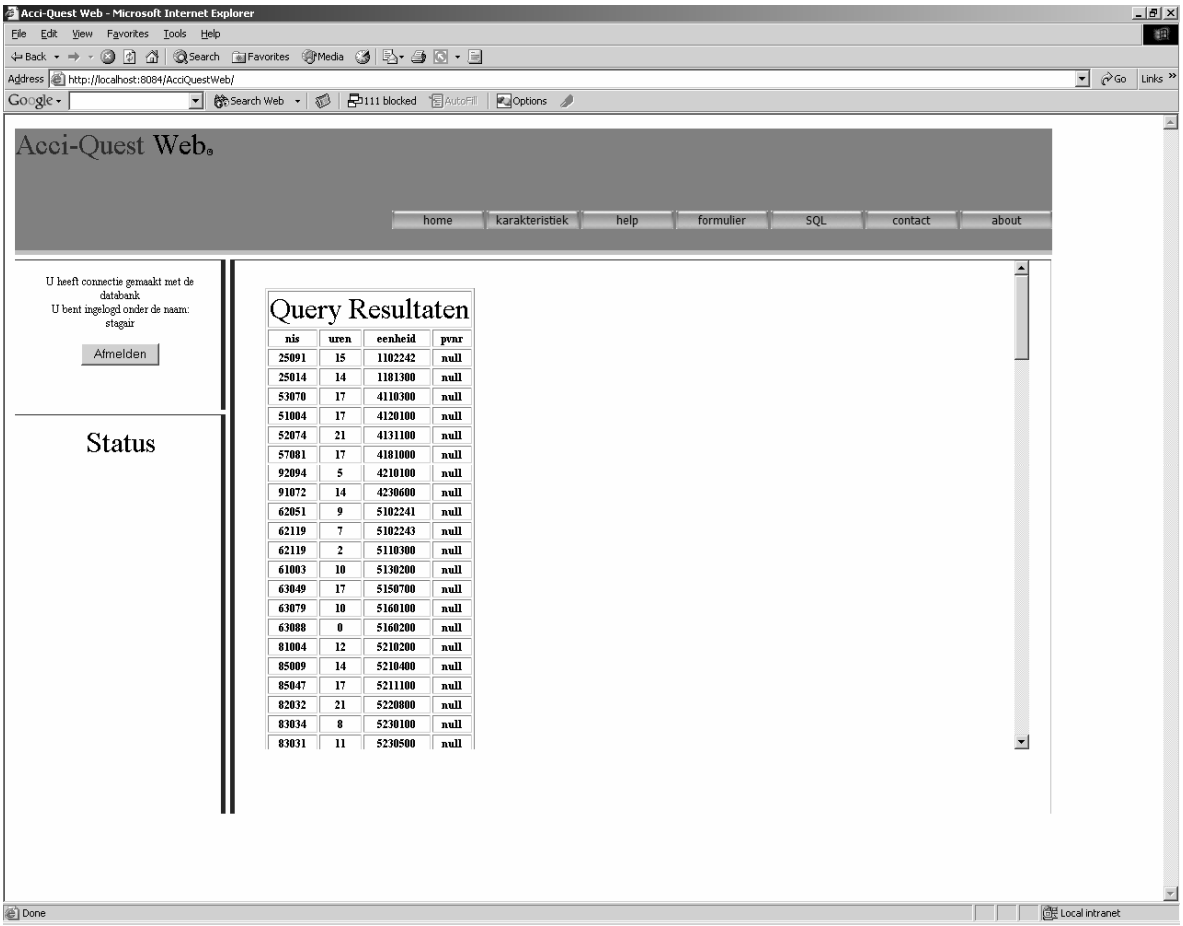
De criteria en selecties van de voorbeelden zijn in de tekst opgenomen. Een screenshot geeft daarenboven een weergave van de resultaten van de zoekopdracht. Als de gebruiker het voorbeeld volgt, dan genereert de applicatie een SQL-zoekopdracht die overeenkomt met deze die in de tekst is opgenomen. Vervolgens wordt nog aangegeven in deze tekst welke de beperkingen zijn van de applicatie.

7.1 Zoekopdracht één

De eerste zoekopdracht omvat alle ongevallen die hebben plaatsgevonden op 4 september 1996. De informatie die voor deze *records* moet worden weergegeven is: Code Eenheid, PVNR, NIS code en het uur waarop het ongeval heeft

plaatsgevonden. De gebruiker maakt deze selectie op de formulier-pagina en geeft aan dat de resultaten moeten worden getoond in het HTML-formaat. Dan drukt de gebruiker de knop ‘verzenden’ in en het programma creëert een SQL-zoekopdracht. Deze zoekopdracht is als volgt:

SELECT o.nis,o.uren,o.eenheid,o.pvnr FROM (SELECT id, nis, jaar, maand, dag, uren, eenheid, pvnr FROM Ongeval WHERE jaar = 1996 AND maand = '9' AND dag = '4') o

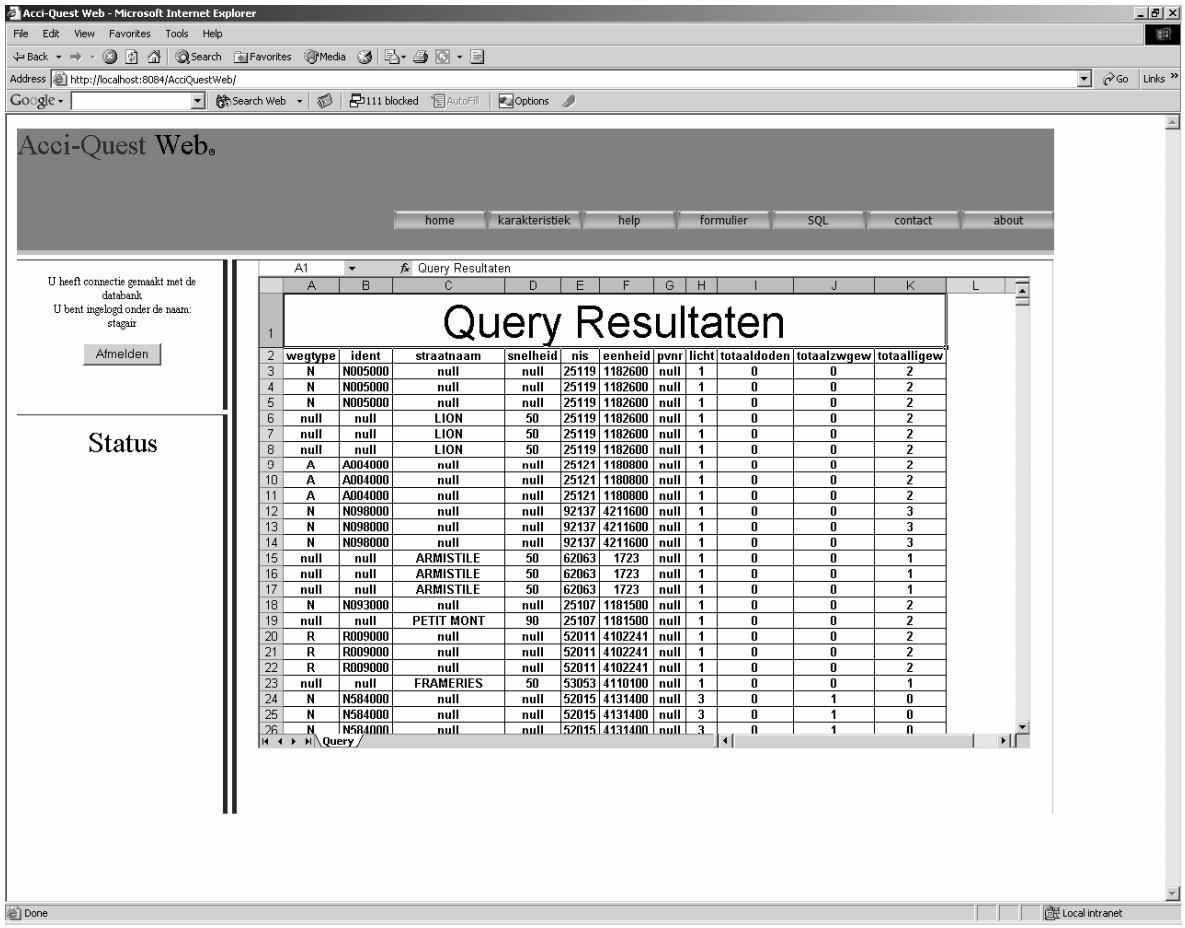


figuur 7.3 Screenshot van de resultaten van query1

De gebruiker ontvangt de resultaten in het programma. Deze komen overeen met figuur 7.3. Indien de gebruiker de karakteristiek-knop indrukt, verschijnt in het statusvenster de zoekopdracht en het aantal resultaten dat werd gegenereerd.

7.2 Zoekopdracht twee

Het criterium van de tweede opdracht wordt gevormd door het tijdstip: het jaar 1995 en het type botsing: kettingsbotsingen. De kenmerken die moeten worden weergegeven zijn: Code Eenheid, PVNR, NIS, wegtype, wegnummering, straatnaam, maximum toegelaten snelheid, lichtgesteldheid op het moment van de botsing, het aantal doden, licht- en zwaargewonden. Het resultaat wordt weergegeven in een Excel-bestand. Deze keuze wordt door de gebruiker aangegeven op de formulier-pagina net boven de knop verzenden.



wegtype	ident	straatnaam	snelheid	nis	eenheid	pvnr	licht	totaaldoden	totaalzwgew	totaalligew
N	N005000	null	null	25119	1182600	null	1	0	0	2
N	N005000	null	null	25119	1182600	null	1	0	0	2
N	N005000	null	null	25119	1182600	null	1	0	0	2
null	null	LION	50	25119	1182600	null	1	0	0	2
null	null	LION	50	25119	1182600	null	1	0	0	2
null	null	LION	50	25119	1182600	null	1	0	0	2
A	A004000	null	null	25121	1180800	null	1	0	0	2
A	A004000	null	null	25121	1180800	null	1	0	0	2
A	A004000	null	null	25121	1180800	null	1	0	0	2
N	N098000	null	null	92137	4211600	null	1	0	0	3
N	N098000	null	null	92137	4211600	null	1	0	0	3
N	N098000	null	null	92137	4211600	null	1	0	0	3
null	null	ARMISTILE	50	62063	1723	null	1	0	0	1
null	null	ARMISTILE	50	62063	1723	null	1	0	0	1
null	null	ARMISTILE	50	62063	1723	null	1	0	0	1
N	N093000	null	null	25107	1181500	null	1	0	0	2
null	null	PETIT MONT	90	25107	1181500	null	1	0	0	2
R	R009000	null	null	52011	4102241	null	1	0	0	2
R	R009000	null	null	52011	4102241	null	1	0	0	2
R	R009000	null	null	52011	4102241	null	1	0	0	2
null	null	FRAMERIES	50	53053	4110100	null	1	0	0	1
N	N584000	null	null	52015	4131400	null	3	0	1	0
N	N584000	null	null	52015	4131400	null	3	0	1	0
N	N584000	null	null	52015	4131400	null	3	0	1	0

figuur 7.4 Screenshot van de resultaten van query2

De databank bevat 1.609 *records* die aan deze criteria voldoen. Een deel van de resultaten is terug te vinden in bovenstaande figuur 7.4. De SQL-zoekopdracht die

door het programma wordt gegenereerd op basis van deze invoer van de gebruiker is als volgt:

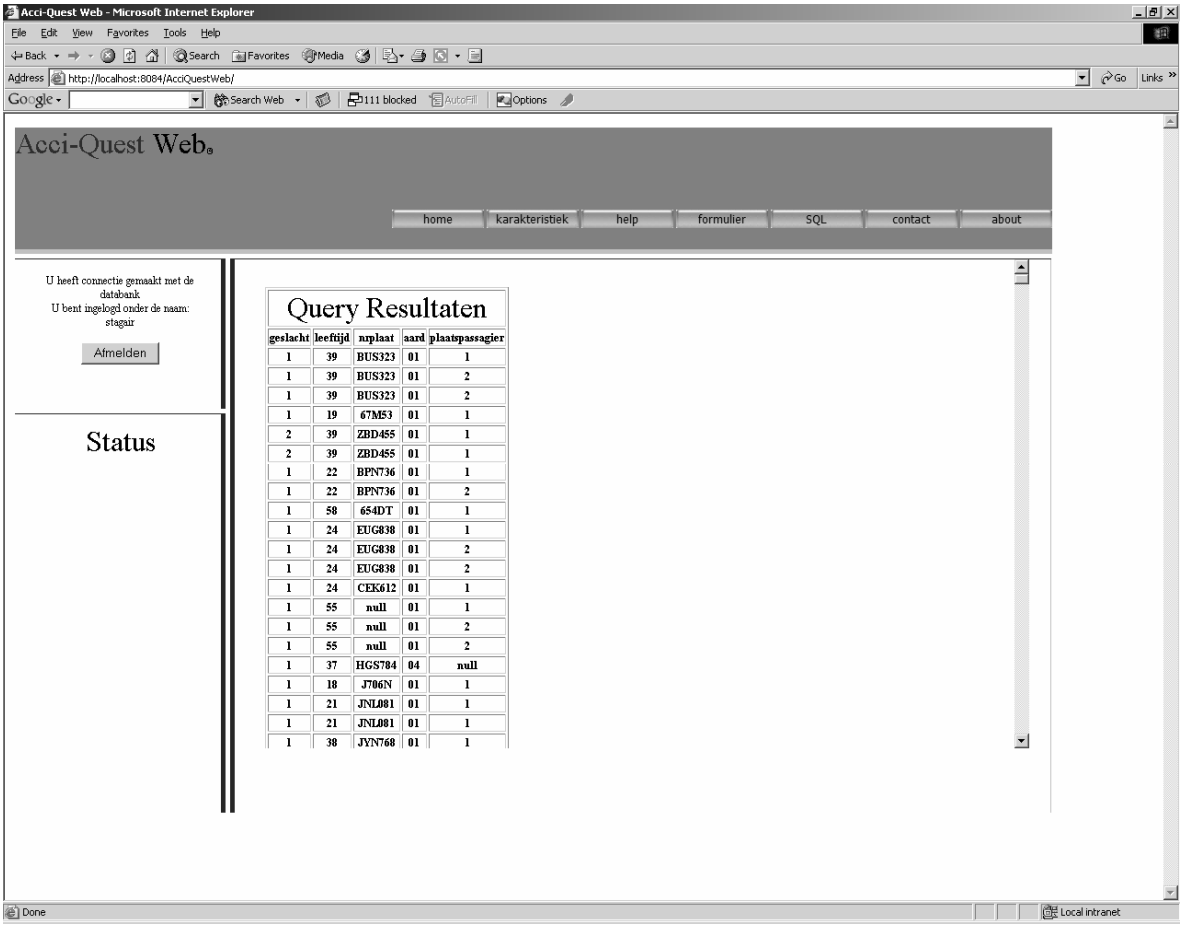
SELECT l.wegtype, l.ident, l.straatnaam, l.snelheid, o.nis, o.eenheid, o.pvnr, o.licht, o.totaaldoden, o.totaalzwgew,o.totaalligew FROM (SELECT id, wegtype, ident, straatnaam, snelheid FROM Lokatie WHERE) l,(SELECT id, nis, jaar, eenheid, pvnr, licht, totaaldoden, totaalzwgew, totaalligew FROM Ongeval WHERE jaar = 1995) o, (SELECT id, typeaanrijding FROM WegGebruikers WHERE typeaanrijding = '1') w WHERE l.id = o.id AND l.id = w.id

7.3 Zoekopdracht drie

De derde zoekopdracht bevat alle ongevallen van het type frontale botsing die plaatsvonden op het moment dat de dag aanbreekt (dageraad). De kenmerken die moeten worden weergegeven in het resultaat zijn: aard en nummerbord van het voertuig, geslacht en leeftijd van de chauffeur en de plaats van de passagier in het voertuig. De opdracht ziet eruit als volgt:

SELECT w.geslacht, w.leeftijd, w.nrplaat, w.aard, pas.plaatspassagier FROM (SELECT id,licht FROM Ongeval WHERE licht = '2') o, (SELECT id,geslacht, leeftijd, nrplaat, typeaanrijding, aard FROM WegGebruikers WHERE typeaanrijding = '2') w, (SELECT id, plaatspassagier FROM Passagiers WHERE) pas WHERE o.id = w.id AND o.id = pas.id

De databank bevat 1.336 *records* die aan deze criteria voldoen. Figuur 7.5 op pagina 72 geeft aan hoe deze resultaten in de applicatie worden weergegeven.



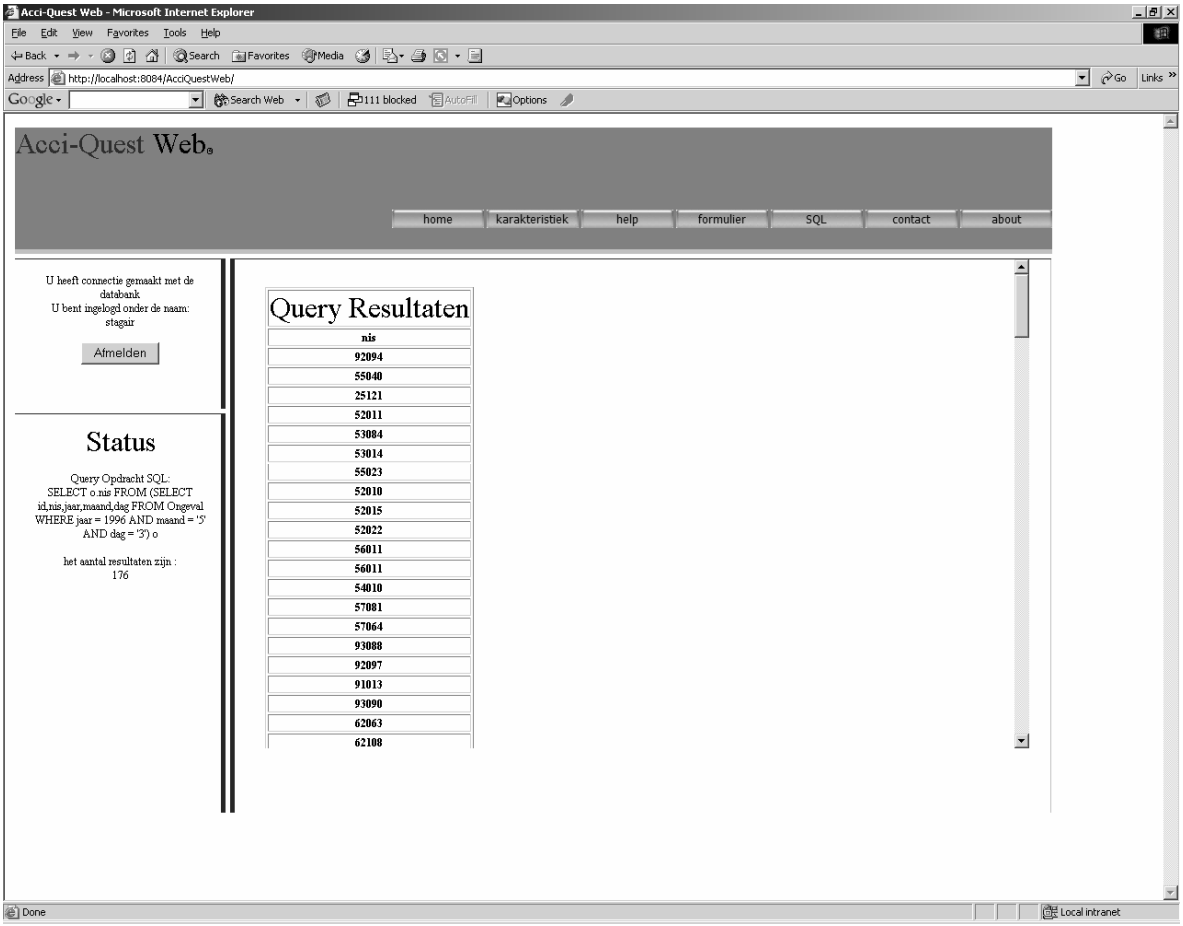
figuur 7.5 Screenshot van de resultaten van query3

7.4 Zoekopdracht vier

De vierde opdracht wordt in SQL-code ingegeven in de SQL-pagina. Via deze weg worden dezelfde resultaten bekomen als via de formulier-pagina.

SELECT o.nis FROM (SELECT id,nis,jaar,maand,dag FROM Ongeval WHERE jaar = 1996 AND maand = '5' AND dag = '3') o

Het aantal records bedraagt 176. Figuur 7.6 op pagina 73 geeft aan hoe deze resultaten in de applicatie worden weergegeven.



figuur 7.6 Screenshot van de resultaten van query4

7.5 Beperkingen

Beperking één:

Indien de gebruiker zelf zoekopdrachten wenst in te geven, dient dit te gebeuren op basis van de SQL-taal. Deze kunnen naargelang de databank licht verschillen. Daarenboven moet de gebruiker werken met subopdrachten en niet alles in één zoekopdracht trachten te vatten.

Beperking twee:

De geschiedenis bevat niet de laatst uitgevoerde zoekopdracht. Dus een nieuwe zoekopdracht moet uitgevoerd te worden vooraleer de vorige onder *history* wordt opgenomen.

Beperking drie:

In de formulierpagina is de keuze van de gebruiker beperkt tot alle velden die zijn opgenomen op het formulier. Op het vlak van de ID-velden kan bijvoorbeeld niet worden geselecteerd. Hiervoor moet gebruik worden gemaakt van de SQL-pagina.

Beperking vier:

Doordat de data in de databank vooral uit cijfercodes bestaat, zijn dit ook de gegevens die over het algemeen worden weergegeven in de tabel. Voor een gebruiker die de codes niet kent en deze moet gaan opzoeken kan dit veel werk betekenen. Een oplossing kan zijn, de gegevens terug te decoderen voordat ze aan de gebruiker worden getoond. Indien echter een andere weergavecomponent aan de applicatie wordt toegevoegd, dan moeten de cijfercodes gebruikt kunnen worden.

Beperking vijf:

Op de formulierpagina is het niet mogelijk om te werken met intervallen voor het selecteren van *records.* Indien men dus alle records wil selecteren die kleiner zijn dan een bepaalde waarde, dan moet de SQL-pagina worden gebruikt. Een oplossing is de formulier-pagina te gebruiken om de opdracht uit te voeren met een gelijkheidsteken en dan via status de opdracht te kopiëren naar de SQL-pagina en daar het symbool '=' te vervangen door het gewenste '<' of '>'.

8. Besluit

In deze tekst werd getracht een beschrijving te geven van de ontwikkeling van de applicatie AcciQuestWeb. Het stand-aloneprogramma AcciQuest werd omgevormd tot een netwerkgeoriënteerd programma waarbij gebruik werd gemaakt van de J2EE-specificatie. De mogelijkheden voor het gebruik werden hierdoor uitgebreid. Meer gebruikers kunnen van uit verschillende locaties, tegelijkertijd, via een netwerk de verkeersdata onderzoeken.

Door het toepassen van de SDLC-methode is een gestructureerd stappenplan gevolgd bij de ontwikkeling van het informatiesysteem AcciQuestWeb. In de tijdspanne die voor deze opdracht ter beschikking was, bleek de ontwikkeling van een GIS-component niet mogelijk. Het is om die reden opgenomen in de lijst van mogelijke uitbreidingen. Het programma kan een extra dimensie krijgen indien de data worden weergegeven in een geografisch kader. Dit zal de gebruikers toelaten sneller en eenvoudiger de data te analyseren.

Het informatiesysteem is op een wijze geprogrammeerd die relatief eenvoudig aanpassingen en uitbreidingen mogelijk maakt. Enkele mogelijke uitbreidingen van deze applicatie worden in onderstaande opsomming gepresenteerd:

- het toevoegen van een GIS-component
- het installeren in een professionele omgeving
- het aanvullen met administratiefunctie zodat onderhoud via de applicatie kan verlopen en aanpassingen aan de code mogelijk zijn
- het toevoegen van een databank met gebruikernamen en paswoorden
- het toelaten aan nieuwe gebruikers om zichzelf online te registreren

Hieronder volgt ter afsluiting bij elk van de uitbreidingsmogelijkheden een korte beschrijving.

GIS-omgeving

Het toevoegen van een GIS-component kan worden omschreven als de belangrijkste volgende activiteit in de verdere ontwikkeling. Het zou de applicatie omvormen tot een echt Decision Support Systeem [DSS].

Een DSS ondersteunt de leidinggevenden bij het nemen van management-beslissingen door het combineren van data, analytische modellen en gebruikersvriendelijke software. De gebruikers van DSS worden voorzien van flexibele instrumenten voor het analyseren van grote hoeveelheden data. De componenten van een DSS bestaan uit een databank, een softwaresysteem met modellen en een gebruikersinterface. Het DSS-softwaresysteem bevat de instrumenten die worden gebruikt voor de analyse van de data zoals: a.verschillende OnLine Analytical Processing [OLAP]-instrumenten; b.traditionele zoekopdrachten; c.*datamining*; d.multidimensionele-data-analyse-instrumenten; e.een collectie van wiskundige en analytische modellen. [Laudon et al., 2004]

Data uit informatiesystemen kan eenvoudiger opgenomen worden door de gebruikers indien deze informatie in een of ander grafisch platform wordt weergegeven. Hierbij spreken we over tabellen, grafieken, kaarten, drie-dimensionele presentaties, animatie en andere datavisualiseringstechnologieën. Het presenteren van data in een grafische vorm laat gebruikers toe, sneller en eenvoudiger patronen en relaties te ontdekken in deze data. Sommige datavisualiseringstechnieken zijn daarenboven interactief zodat de gebruiker kan bepalen welke informatie wordt weergegeven en kan achterhalen welke de effecten zijn van manipulaties van de criteria door de gebruiker. [Laudon et al., 2004]

GIS kunnen worden gedefinieerd als een speciale categorie van DSS die gebruik maken van datavisualiseringstechnieken. Deze technieken worden gebruikt voor het analyseren en weergeven van data voor planning en beslissingneming in de vorm van gedigitaliseerde kaarten. GIS-software is bedacht om geografisch verbonden [referenced] informatie te assembleren, op te slaan, te manipuleren en weer te geven door deze data te linken aan punten, lijnen en gebieden op een kaart. GIS kan worden gebruikt ter ondersteuning van beslissingen die nood hebben aan informatie betreffende de geografische context van problemen. Dit impliceert het gebruik van GIS voor een hele waaier van problemen gaande van wetenschappelijk onderzoek over middelenallocatie tot ontwikkelings-ondersteuning. GIS kan perfect gecombineerd worden met een web gerealiseerde interface en een server gelocaliseerde databank. [Laudon et al., 2004]

Installeren in een professionele omgeving

De te nemen stappen om de installatie te realiseren werden al in de tekst zelf opgenomen. Geschikte machines moeten worden geselecteerd voor het uitvoeren van de verschillende taken binnen de applicatie. Voordat de applicatie operationeel kan worden gebruikt, dient nog een grondige testfase te worden doorlopen.

Administratiefunctie toevoegen zodat onderhoud via de applicatie kan verlopen en aanpassingen aan de code mogelijk zijn

Een administratiefunctie zal administrators toelaten zich apart aan te melden aan de applicatie om bepaalde werkzaamheden uit te voeren.

Extra databank met gebruikernamen en paswoorden

Dit zal toelaten om gegevens van nieuwe gebruikers op te nemen in deze databank vanaf welk moment de applicatie rekening zal houden met deze waarden. De applicatie zal, indien iemand zich wenst aan te melden, gewoon de databank raadplegen met daarin alle gebruikersnamen en paswoorden. Indien die van de huidige gebruiker erin voorkomt, dan zal deze toegang krijgen tot de applicatie. Een veiligheidsrapport [security rapport] dient opgesteld te worden om

het niveau van beveiliging vast te stellen. Indien de organisatie een beleid heeft op dit vlak, dan moet dit beleid ook op deze applicatie worden toegepast.

Toelaten aan nieuwe gebruikers om zichzelf online te registreren

De mogelijkheid wordt geboden aan gebruikers om *online,* in een venster van de applicatie, hun karakteristieken in te vullen. Indien zij die verzenden, dan zal in een databank met gebruikerinformatie een nieuwe instantie worden aangemaakt. Daarenboven wordt een nieuw paswoord gecreëerd, geïnitialiseerd en doorgestuurd naar de gebruiker. Hierdoor hoeft de administratie of het onderhoud zich niet meer bezig te houden met het distribueren van paswoorden en deze te initialiseren.

Literatuur

ARMSTRONG, E. et al., *The Java Web Services Tutorial*. Palo Alto, 2002, Online, Sun Microsystems, Internet, 7 november 2004.
Beschikbaar: http://java.sun.com/webservices/docs/1.2/tutorial/doc/index.html

ARMSTRONG, E. et al., *J2EE 1.4 Tutorial: for Sun Java System Application Server Platform edition 8 2004Q4Beta*. Santa Clara, 2004, Online, Sun Microsystems, Internet, 7 november 2004.
Beschikbaar: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf

BELL, D. en M. PARR, *Java voor studenten*. New Jersey: Pearson Education, 2002.

BERGSTEN, H., *JavaServer Pages*. Sebastopol: O'Reilly & Associates, 2001.

BERSON, A, *Client/Server Architectecture*. New York: McGraw-Hill, 1996.

BUSCHMANN, F. et al., *Pattern-oriented software architecture: a system of patterns.* Chichester: John Wiley & Sons, 2001.

CHASE, R.B., F.R. JACOBS en N.J. AQUILANO, *Operations Management for Competitive Advantage*. New York: McGraw-Hill/Irwin, 2004.

HALL, M., *Core Servlets and JavaServer Pages*. Indianapolis: Prentice Hall Professional Technical Reference, 2000.

HAYES-ROTH, F. en D. AMOR, *Radical Simplicity: Transforming Computers Into Me-Centric Appliances*. New Yersey: Prentice Hall Professional Technical Reference, 2003.

HOFFER, H.A., J.F. GEORGE en J.S. VALACICH, *Modern systems analysis & design*. New Jersey: Pearson Education, 2002.

HUNTON, J.E., S.M. BRYANT, N.A. BAGRANOFF, *Core concepts of technology auditing*. New York: John Wiley & Sons, 2004.

INTERNATIONAL BUSINESS MACHINES COORPORATION, *Developing Enterprise Java Applications Using DB2 Version 8*. s.l., 2000-2002, Online, IBM prduct manuals, Internet, 15 oktober 2004.
Beschikbaar: http://www-306.ibm.com/software/data/db2/udb/support/manuals

INTERNATIONAL BUSINESS MACHINES COORPORATION, *Installing DB2 Universal Database Enterprise Server Edition*. s.l., 1993-2004, Online, IBM prduct manuals, Internet, 10 oktober 2004.
Beschikbaar: http://www-306.ibm.com/software/data/db2/udb/support/manuals

INTERNATIONAL BUSINESS MACHINES COORPORATION, *Quick Beginnings for DB2 Servers: Version 8*. s.l., 1993-2002, Online, IBM prduct manuals, Internet, 20 oktober 2004.
Beschikbaar: http://www-306.ibm.com/software/data/db2/udb/support/manuals

LAUDON, K.C. en J.P. LAUDON, *Management Information Systems: Managing the digital firm*. New Jersey: Pearson Education, 2004.

LEFEBVRE, E.R.J., *Tekst en organisatie: Ideeën en beschouwingen voor het management van academisch denken en schrijven.* Leuven: Acco, 1997.

MCNURLIN, B.C. en R.H. SPRAGUE jr., *Information systems management in practice*. New Jersey: Pearson Education, 2004.

NEDERHOED, P., *Helder rapporteren: een handleiding voor het schrijven van rapporten, scripties, nota's en artikelen in wetenschap en techniek*.
Houten/Diegem: Bohn Stafleu Van Loghum, 1999.

PANKO, R.R., *Business Data Networks and Telecommunications*. New Jersey: Pearson Education, 2003.

PRESSMAN, R.S., *Software Engineering*. New York: McGraw-Hill, 2001.

STALLINGS, W., *Operating Systems: Achtergronden, werking en ontwerp*.
Schoonhoven: Academic Service, 2001.

VAN HEMMEN, L. et al., *Bedrijfsvoering en ICT op één lijn*. Den Haag: Academic Service, 2004.

Bijlage A: Fouten in de logica

tabel A.1 Weergave van de fouten in de logica; C: criterium; S: selectie

Vraag	variabele	situering	omschrijving van de fout	uitgevoerde oplossing
1	code eenheid	C	nullen vooraan niet meegenomen	string ipv. Integer gebruiken
1	pvnr	C	nullen vooraan niet meegenomen	string ipv. Integer gebruiken
5	soort_een	C	foutmelding	setLocation case4 --> String
5	soort_twee		foutmelding	parameterfout bij setkind2
5	soort_een	S	foutmelding	
8B	weggebruiker	S, C	fout in de query	verkeerde tabelnaam
9	weer	S	niets geselecteerd	false --> true
11	staat weg	S,C	fout in de query	os ipv. s voor OngStaatw
16	beweging gebr1	S, C	fout in de query	false --> true
19	voetganger	S	niets geselecteerd	parameterfout logica
19	overst	S	niets geselecteerd	parameterfout logica
19	afstand	C	fout in de query	parameterfout logica
20	tweewieler	S, C	fout in de query	parameterfout logica
21	varia	S, C	niet opgenomen HTML	wel opgenomen in HTML
22	borden	S, C	niet opgenomen HTML	wel opgenomen in HTML
22	lading	S, C	nooit resultaten	captionfout+parameterfout
24	aard	C	foute query	string ipv. Integer gebruiken

Bijlage B: Code uit formulier.htm

```
   <fieldset style="padding: 2">
          <legend>26. Andere slachtoffers</legend> <p> 
         <select size="1" name="geslacht_drie">
         <option>GESLACHT</option>
         <option value="1">man</option>
         <option value="2">vrouw</option>
         </select>
      <input type="checkbox" name="check_geslacht_drie"
value="0">     
         <select size="1" name="gevolg_drie">
         <option>GEVOLGEN</option>
         <option value="1">dood</option>
         <option value="2">zwaar gewond</option>
         <option value="3">licht gewond</option>
         <option value="4">ongedeerd</option>
         </select>
      <input type="checkbox" name="check_gevolg_drie"
value="0">        
      <input type="text" name="leeftijd_drie" size="5" value="0">
      <input type="checkbox" name="check_leeftijd_drie" value="0">
          leeftijd</p>
          <p> </p>
          <p> </p>
   </fieldset>

<!-- optie keuzemenu voor weergave resultaten-->
   <fieldset style="padding: 2">
      <legend><span lang="en-us">Weergave query resultaten</span></legend>
      <input type="radio" value="1" name="weergave"><span lang="en-us"> Excel formaat</span>
      <p><input type="radio" checked name="weergave" value="2"><span lang="en-us"> HTML
formaat</span></p>
      <p> </p>
   </fieldset>

<!-- weergave knoppenbalk-->
   <input type="hidden" name="queryX" value="1">
   <p align="center">
   <input type="reset" value="Invoer wissen" name="B2">
   <input type="submit" value="Verzenden" name="B1"></p>
```

code B.1 HTML-code uit de formulier.htm pagina

Bijlage C: Code uit servlet Status.java

```
public class Status extends HttpServlet {

   private String connectie;
   private String queryString;
   private Long count;

   public void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Status";
      connectie = "";
      HttpSession session = request.getSession(false);

      try{
         connectie = (String)session.getAttribute("connectie");
         queryString = (String)session.getAttribute("queryString");
         count = (Long)session.getAttribute("count");

         if (connectie.equals(session.getId())){
            out.println(ServletUtilities.headWithTitle(title));
            out.println("<BODY><TITLE>" + title + "</TITLE>");
            out.println("<p align ='center'><font size='10'>" + title + "</font><br><br>");
            out.println("Query Opdracht SQL: <br>"+ queryString + "<br><br>");
            out.println("het aantal resultaten zijn : <br>"+ count +"<br><br>");
            out.println("</p></BODY></html>");
         }
      }catch(Exception e){
         out.println(ServletUtilities.headWithTitle(title));
         out.println("<BODY><TITLE>"+ title + "</TITLE>");
         out.println("<p align = 'center'><font size='10'>" + title + "</font><br><br>");
         out.println("U bent niet aangemeld en kan dus geen karakteristieken van de gevraagde
query bekijken <br>");
         out.println("Gelieve in te loggen.");
         out.println("</p></BODY></html>");
      }
      out.close();
   }
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }
}
```

code C.1 Java-code uit de servlet Status.java

Bijlage D: Code uit servlet Query.java

```
public void processRequest(HttpServletRequest request, HttpServletResponse response)throws
ServletException, IOException {

      content(request, response);

      out = response.getWriter();
      HttpSession session = request.getSession(false);
      connectie = "1";

      try{
         Integer accessCount = (Integer)session.getValue("accessCount");
         accessCount = new Integer(accessCount.intValue() + 1);
         connectie = (String)session.getAttribute("connectie");

         if (connectie.equals(session.getId())){
            conn = (Connection)session.getAttribute("conn");
            statement = (Statement)session.getAttribute("statement");

            if (request.getParameter("queryX").equals("1")){
               queryIntuitief(request, response);
               }else if(request.getParameter("queryX").equals("2")){
               queryZelfSql(request, response);
            }
            session.setAttribute("queryString", queryString);
            session.setAttribute("accessCount",  accessCount);
            int i = accessCount.intValue();
            session.setAttribute("sql"+i, queryString);
            execute();
            Long aantal = new Long(count);
            session.setAttribute("count", aantal);
          }
      }catch(Exception e){
            out.println("Deze opdracht kan niet uitgevoerd worden.");
            out.println("U bent niet geconnecteerd met de databank.");
            out.println("Gelieve in te loggen");
      }
   }
```

code D.1 Deel methode processRequest uit de servlet Query.java

```
public void queryIntuitief(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

      ow = new OngWeer(0);
      os = new OngStaatw(0);
      o = new Ongevallen( 0, 0, 0, 0, 0, 0, 0, "", "", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
…

if(request.getParameter("check_code_eenheid")== null){
         o.selectCopUnit(false);
```

```
        }else{
            o.selectCopUnit(true);
        }
        if (request.getParameter("code_eenheid").equals("") ||
request.getParameter("code_eenheid").equals("Code Eenheid")){// voldoetEenheid?
            mode =1; y="";
        }else{
            mode = 0; y = (request.getParameter("code_eenheid"));
        }
        o.changeCopUnit(y,mode);
…
```

code D.2 Deel methode queryIntuitief uit de servlet Query.java

```
public void doQuery() {

if(!l.ietsGeselecteerd() && !w.ietsGeselecteerd() && !o.ietsGeselecteerd() &&
!ow.ietsGeselecteerd() && !os.ietsGeselecteerd()&& !f.ietsGeselecteerd() && !v.ietsGeselecteerd()
&& !ops.ietsGeselecteerd() && !p.ietsGeselecteerd()&& !plak.ietsGeselecteerd() &&
!var.ietsGeselecteerd() && !k.ietsGeselecteerd()  && !pas.ietsGeselecteerd()&&
!slacht.ietsGeselecteerd() && !wgebrf.ietsGeselecteerd() && !fv.ietsGeselecteerd() &&
!t.ietsGeselecteerd()) {

                        String bericht = "Er is niets geselecteerd !!";
                        String titel = "Foutmelding";
                out.println("<br>" + bericht + "<br>" + titel + "<br>");
                 }
                 clean();
            query = new StringBuffer();
                 createSelectQuery();
                 createFromQuery();
                 createWhereQuery();
            queryString = query.toString();
  }
```

code D.3 Deel methode doQuery uit servlet Query.java

```
public void execute(){
        try{
            rs = statement.executeQuery(queryString);
            int z = m_attributes.size();
            count = 0;
            out.println("<br>");
            out.println(ServletUtilities.headWithTitle(title));
            out.println("<TABLE BORDER='1'>");
            out.println("<TR align=center><td colspan =" + z + "><font size ='7'>"+ title +
            "</font></td> </tr>");
            out.println("<tr>");
            int i = 0;
            for ( i=0;i<z;i++){
                out.println("<th>"+ m_attributes.elementAt(i)+ "</th>");
            }
```

```
            out.println("</tr>");
            i = 0;
            while (rs.next()){
               for (i=0;i<z;i++){
                  String attr = rs.getString(m_attributes.elementAt(i).toString());
                  out.println("<th>" + attr +"</th>");
               }
               out.println("</tr>");
               count ++;
            }
            out.println("</table>");
            out.println ("</body></html>");
            out.close();
            rs.close();

         }catch(SQLException e)
                  {
                           String bericht = "Fout in de query";
                           String titel = "Foutmelding";
                           out.println("<br>" + bericht + "<br>");
               out.println(titel);
         }
}
```

code D.4 Methode execute uit servlet Query.java

```
public void content(HttpServletRequest request, HttpServletResponse response)throws
ServletException, IOException{
   try{
      if (request.getParameter("weergave").equals("1")){
         response.setContentType("application/vnd.ms-excel");
      }else if (request.getParameter("weergave").equals("2")){
         response.setContentType("text/html");
        // of loaden van de applet of starten van het GIS gedeelte
      }else {
         response.setContentType("text/html");
         // loaden van de Applet of het starten van het GIS gedeelte
      }
   }catch(Exception e){
      out.println("nullpointerException");
      response.setContentType("text/html");
   }
```

code D.5 Methode content uit servlet Query.java

Bijlage E: Code uit web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd" version="2.4">
 <servlet>
  <servlet-name>Login</servlet-name><servlet-class>Login</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>Query</servlet-name><servlet-class>Query</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>Logout</servlet-name><servlet-class>Logout</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>Status</servlet-name><servlet-class>Status</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>SQLStart</servlet-name><servlet-class>SQLStart</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Login</servlet-name><url-pattern>/Login</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>Query</servlet-name><url-pattern>/Query</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>Logout</servlet-name><url-pattern>/Logout</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>Status</servlet-name><url-pattern>/Status</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>SQLStart</servlet-name><url-pattern>/SQLStart</url-pattern>
 </servlet-mapping>
 <session-config><session-timeout>30</session-timeout></session-config>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
 </welcome-file-list>
</web-app>
```

code E.1 Code het web.xml bestand

Bijlage F: Code uit klasse Ongevallen.java

```
public Ongevallen(int nis,int prov,int year,int month,int day,int hours,int r_p,String cop_unit,String
pv_nr,int crossroads,int inCity,int light,int number_of_drivers,int number_of_passengers, int
number_of_victims,int number_of_people,int number_of_deaths,int number_of_heavy_wounded,
int number_of_light_wounded,int number_of_hw_deaths,int number_of_lw_deaths){
                m_aantal = 21;
                m_select = new boolean[m_aantal];
                m_ingevuld = new boolean[m_aantal];
                for(int i= 0;i<m_aantal;i++)
                {
                        m_select[i] = false;
                        m_ingevuld[i] = false;
                }
                m_touched = false;
                setNis(nis);
…
                setCopUnit(cop_unit);
…
```

code F.1 Deel van de methode Ongevallen uit Ongevallen.java

```
Private void setOngeval(int mode,int i,StringBuffer query,Vector m_attributes){
        switch(i)
        {
                case 0:query.append("nis");
                        if(mode == 0)m_attributes.add("nis");
                        if(mode == 1)query.append(" = '" +
                        ((Integer)m_ongevallen.elementAt(i)).intValue()+ "'");
                break;
…
                case 6:query.append("eenheid");
                        if(mode == 0)m_attributes.add("eenheid");
                        if(mode == 1)query.append("= '" +
                        ((String)m_ongevallen.elementAt(i)) +"'");
                break;
…
```

code F.2 Deel van de methode setOngeval uit Ongevallen.java

```
public void changeCopUnit(String x,int mode){
        String m_cop_unit =x;
        m_ongevallen.setElementAt(m_cop_unit,6);
        if(mode == 0)
                m_ingevuld[6] = true;
        else
                m_ingevuld[6] = false;
}
```

code F.3 Methode changeCopUnit uit Ongevallen.java