Eindverslag Bsc-project IN3700

4 October 2007

Instelling:
Technische Universiteit Delft
Faculteit Elektrotechniek Wiskunde en Informatica(EWI)

Instituut:
Data Archiving and Networked Services(DANS)

Opdrachtgever:
Drs. Milco Wansleeben

Begeleider DANS:
Ir. Rutger Kramer

Coördinator TUDelft:
Ir. Matthijs Sepers

Studenten TUDelft:
Ralph Matroos
Ramses Elisabeth

TU Delft
Delft University of Technology

Data Archiving and Networked Services
DANS

Voorwoord

Dit verslag is het laatste deel van een bachelorstage, als onderdeel van het derde studie jaar van de opleiding Technische Informatica aan de TU Delft. Dit project is voor ons een leuke uitdaging geweest en was in grote lijnen bedoeld om ons de kennis bij te brengen die nodig is om zelfstandig het gehele software-ontwikkeltraject te doorlopen.

Wij zijn heel veel dank verschuldigd aan onze begeleiders Ir. Rutger Kramer(DANS) en Ir. Matthijs Sepers(TUDelft), de opdrachtgever Drs. Milco Wansleeben en iedereen bij DANS die ons geholpen heeft gedurende de laatste 10 weken.

Delft, 4 oktober 2007

Ralph Matroos,
Ramses Elisabeth

Inhoudsopgave

Samenvatting

Het instituut DANS heeft als grootste doel het opslaan en beschikbaar stellen van onderzoeksgegevens. Sinds januari 2005 hebben zij een programma genaamd EASY in gebruik waarin dit soort gegevens als dataset worden opgeslagen en weer opgevraagd kunnen worden. Gebruikers kunnen zich voor deze service registreren en kunnen dan zelf hun eigen werk ook deponeren. Omdat de omvang van het gebruik zo snel gegroeid is heeft men de laatste tijd behoefte aan informatie betreffende het gebruik van de applicatie.

De enige manier om in het huidige systeem informatie hierover te kunnen vinden is door ingewikkelde database queries te schrijven wat veel tijd kost en niet handig is voor de beheerders van EASY omdat ze dan eerst de databasetaal SQL moeten gaan leren.
Het doel van onze opdracht is om een web-applicatie te maken die de beheerders van EASY alle nodige informatie d.m.v. een paar clickjes met de muis kan tonen. Dit hebben we gedaan door eerst een vooronderzoek te doen en daarna middels eXtreme Programming[6] en wekelijkse vergaderingen de webapplicatie te bouwen. Bij eXtreme Programming begin je na het vaststellen van de requirements gelijk met programmeren, je hebt wekelijkse vergaderingen(Planning games) waar samen met de opdrachtgever gediscussieerd wordt welke requirements de week daarop geïmplementeerd gaan worden.

Het uiteindelijke product(Easy Usage Statistics) werkt als volgt: Bij het inloggen op EUS krijg je een algemeen scherm te zien met daarin gelijk de belangrijkste informatie zoals b.v. meeste downloads, meeste deposits, soorten ingevoerde zoektermen, laatste 10 bekeken datasets etc. Verder hebben we aan de zijkant een menu waarin je kunt kiezen uit Home(startpagina), User statistics, AIP statistics, Category statistics, Help, About en Back to EASY.

Verder hebben wij nog nagedacht over onze webapplicatie binnen het nieuwe EASY project Fedora en hebben we gekeken naar mogelijke uitbreidingen op EUS in de toekomst.

Begrippen

DANS	Data Archiving and Networked Services[14] is het instituut waar onze stage heeft Plaatsgevonden. DANS is de nationale organisatie die zorgt voor de opslag en blijvende toegankelijkheid van onderzoeksgegevens in de alfa&gamma wetenschappen.
EASY	Electronic Archiving SYstem is een applicatie binnen DANS waar onderzoeksgegevens worden gedeponeerd en opgevraagd.
EUS	De applicatie die tijdens onze bachelorstage ontworpen en geimplementeerd is om gegevens over het gebruik van EASY gemakkelijk te kunnen weergeven.
DAO	Een Data Acces Object (DAO) [8] wordt gebruikt om toegang tot de data te scheiden van de business logica.
MVC	Model-view-controller (MVC) is een pattern dat gebruikt wordt in de software engineering . In complexe computer applicaties die veel data aan de gebruiker tonen is het vaak wenselijk om data te scheiden van de user interface(zie paragraaf 3.1.2).
XP	eXtreme Programming[6] is een ontwikkel methodiek waarbij niet uitgegaan wordt van een van te voren vast gesteld technisch ontwerp, maar waar door middel van iteraties naar een eindproduct wordt geconvergeerd.
AIP	Archival Information Package is de term voor een dataset binnen EASY. Deze term stamt uit het Open ARchives Information System(OAIS) reference model, wat door veel digitale archieven wordt gebruikt als raamwerk voor repository systemen.
MoSCoW	Een diagram waarmee men heel makkelijk de requirements kan groeperen en waar heel makkelijk de prioriteit van elke specifieke requirement uit af te lezen is. (Must have, Should have, Could have, Won't have)

1. Inleiding

DANS[14] is een instituut waar men zich bezig houdt met opslag en blijvende toegankelijkheid van onderzoeksgegevens in de alfa- en gammawetenschappen. In de zomer van 2005 is DANS met haar werk begonnen. Deels bestaat dat werk uit de activiteiten van data-archieven die al eerder bestonden; deels omvat het nieuwe activiteiten om een efficiënte infrastructuur voor opslag en toegang tot data van de grond te krijgen op gebieden waarvoor die nog niet bestaat. Een van de belangrijkste applicaties binnen DANS is EASY.

Met EASY kunnen onderzoeksgegevens snel en gemakkelijk worden gedeponeerd in het archief van DANS . Binnen een kort tijdsbestek kunnen zij toegankelijk worden gemaakt via het Internet. Hierdoor worden de gegevens, indien nodig, snel beschikbaar voor ander onderzoek. De data worden op een duurzame manier opgeslagen en onderhouden, zodat ze ook op de lange termijn volledig beschikbaar zullen blijven. EASY richt zich vooral op de 5 hoofdcategorieën(met subcategorieën) t.w. Life sciences, Humanities, Social sciences, Behavioral sciences en Socio cultural sciences. De directie van DANS heeft voor het opstellen van de jaarverslagen gegevens betreffende het gebruik van de EASY website nodig. Omdat het zoeken van deze informatie nu een heel langdradig proces is, heeft men besloten om hier een applicatie voor te maken waar alle informatie gemakkelijk te zien is, genaamd EASY Usage Statistics.

Het doel van dit rapport is om het ontworpen systeem te presenteren en om duidelijk te maken waarom we bepaalde beslissingen hebben genomen. Hierbij wordt rekening gehouden met de requirements die zijn opgesteld na gesprekken met de opdrachtgever en de andere belanghebbenden binnen DANS.

De opbouw van dit rapport is als volgt. In hoofdstuk 2 wordt de probleemstelling en analyse beschreven. Op basis hiervan hebben wij een ontwerp gemaakt die in hoofdstuk 3 te zien is, gevolgd door de implementatie in hoofdstuk 4. In hoofdstuk 5 gaan wij in op het testen van onze applicatie en daarna worden de resultaten in hoofdstuk 6 besproken. Als laatste volgen de conclusie en aanbevelingen in hoofdstuk 7 en 8.

2. Probleemstelling en analyse

In dit gedeelte van het verslag zullen wij de probleemstelling introduceren en ook zullen we het bestaande EASY gedetailleerd uitleggen. Verder zullen we de vastgestelde requirements introduceren. Eerst is het misschien handig om te weten wat EASY precies doet en hoe het in elkaar zit.

2.1 Probleem stelling

Sinds EASY in augustus 2005 in werking is gesteld is het nodig dat bepaalde informatie met betrekking tot het gebruik van dit systeem opgevraagd kan worden.
Informatie zoals welke/hoeveel datasets gedeponeerd worden per dag, het aantal bezoekers per dag/maand/jaar, het aantal gedownloade bestanden, het aantal aanwezige datasets, etc.
Om in EASY een beeld te krijgen van deze informatie wordt er sinds mei jl. een MySQL[12] database bijgehouden en uit de informatie die in deze database wordt opgeslagen valt af te leiden hoe gebruikers met de EASY website omgaan. Er is echter geen software voorhanden om de rauwe data op een menselijk interpreteerbare manier te presenteren. Als er nu wat van deze gegevens moet worden opgevraagd dient men zelf in de database te gaan zoeken en aangezien er de laatste tijd steeds meer vraag is naar deze informatie is er behoefte aan een applicatie die dit gemakkelijk kan weergeven en exporteren naar gewenste formaten.

Wat we dus moeten doen is, kort samengevat een applicatie ontwikkelen waarmee men gemakkelijk inzicht kan krijgen in de gebruiksgegevens van EASY.

2.2 Betrokken partijen

Het hele project is uitgevoerd onder toezicht van Ir Rutger Kramer, onze begeleider binnen DANS, en de coördinator van TU Delft Ir Matthijs Sepers. Verder hebben we regelmatig afspraken gehad met de opdrachtgever, drs. Milco Wansleeben, met de directeur en adjunct directeur van DANS respectievelijk Dr. Peter Doorn en Dr. Henk Harmsen en met de andere klanten van EASY t.w. Drs. Marjan Balkestein, Jetske van der Schaaf en Drs. Laurents Sesink. Bij het uitwerken van de requirements is rekening gehouden met de wensen van alle betrokken partijen binnen DANS

2.3 Product omgeving

Om ons project succesvol te kunnen afronden hebben we gebruik gemaakt van verscheidene programma's en tools. In dit deel van het verslag worden deze programma's en tools uitgelegd.

2.3.1 Eclipse Europa

Eclipse[13] is vrije open source software ontwikkeld door de onafhankelijke Eclipse foundation, die gesteund wordt door belangrijke bedrijven zoals IBM Rational, Borland, Red Hat en SuSE. Aangezien eclipse ook binnen DANS als programmeeromgeving gebruikt wordt, hebben wij om zo gemakkelijk mogelijk ons werk binnen EASY te kunnen integreren, ook dit programma gebruikt als basis ontwikkel omgeving.

2.3.2 MySQL

MySQL[12] is een opensource relationeel database management systeem(RDMS) dat gebruik maakt van SQL. SQL oftewel sructured query language is een gestandaardiseerde taal die gebruikt kan worden worden voor taken zoals het opvragen en het aanpassen van informatie in een relationele databank. Onze applicatie haalt met behulp van de java standaard package java.sql de gegevens uit 2 databases waarmee voortdurend gecommuniceerd wordt.

2.3.3 JFreeChart

Omdat er veel informatie weergeven moet worden zijn wij gaan nadenken over goede manieren om dit te representeren. Eén daarvan is door middel van grafieken. De beste oplossing voor ons was de tool JFreeChart[15] aangezien dit een opensource programma is en dus gratis geïmplementeerd mag worden. Om met JFreeChart een grafiek te maken moet je eerst een "dataset" creëren die je dan gebruikt om de grafiek te maken. Een dataset in JFreeChart bestaat uit de data die je wilt weergeven in je grafiek. Na een dataset te hebben gemaakt geef je deze mee aan de methode createChart die we dan zo hebben aangepast dat het gemaakte grafiek word opgeslagen als "jpeg".

2.3.4 CVS

Omdat er verschillende programmeurs aan EASY werken is het handig om een programma te hebben die alle veranderingen bijhoudt op een centrale plaats. Dit wordt bij DANS gerealiseerd met CVS (Concurrent Versions System) . CVS gebruikt een cliënt server architectuur en zit als volgt in elkaar: een server die bewaart de echte versie van het project en de geschiedenis hiervan op schijf. De cliënten kunnen dan inloggen hierop en een complete kopie van het project downloaden (check-out) naar hun eigen pc en kunnen veranderingen weer middels een "commit" laten aanbrengen op de server. Om te voorkomen dat mensen elkaars werk wissen wordt op de server bijgehouden waar er veranderingen zijn aangebracht. Stel dat bijvoorbeeld iemand een verandering wil opslaan in een class waar iemand anders vlak voor hem al iets had aangepast dan krijg je een conflict. Bij een conflict wordt een scherm geopend waarin je beide stukken code naast elkaar ziet en deze als nodig is kan samenvoegen. Verder is er nog een update commando waarmee je op elk willekeurig tijdstip het project op jouw systeem kan laten bijwerken tot de nieuwste versie(versie op de server).

2.3.5 XML

XML (eXtensible Markup Language) [5] is een standaard voor de representatie van gestructureerde gegevens in de vorm van tekst. Deze tekst is zowel voor een computer als voor een mens leesbaar en wordt niet alleen gebruikt om gegevens in op te slaan maar ook om gegevens via het Internet te versturen. Het gaat in dit bestandsformaat dus om de structuur van informatie.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<playlist name="mylist">
        <song>
                <title>Little Fluffy Clouds</title>
                <artist>the Orb</artist>
        </song>
        <song>
                <title>Goodbye mother Earth</title>
                <artist>Underworld</artist>
        </song>
</playlist>
```

Figuur 2.3.4.1

Zoals je hierboven ziet maak je door middel van tags (b.v. <playlist>) groeperingen en kan je binnen tags weer nieuwe tags aanmaken zodat je een duidelijke structuur hebt.

2.3.6 XSL en XSLT

XSL (Extensible stylesheet language) [5] en XSLT(Extensible stylesheet language transformations) [7] worden gebruikt om te beschrijven hoe XML documenten geformatteerd en weergeven moeten worden. Cliënts zoals web-browsers kunnen door XSLT opgevraagde XML bestanden omzetten in HTML. In de volgende figuur laten we een stukje xslt zien.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/> ...
 </xsl:stylesheet>
```

Figuur 2.3.5.1

2.4 Huidige situatie

Momenteel is de enige manier om in het huidige systeem wat aan informatie te vinden over het gebruik van EASY, het zelf zoeken in de managementdata, metadata en filedata (zie paragraaf 2.5.3) of door zelf queries te laten uitvoeren op de EasyStatistics database. Bij het bouwen van onze applicatie betreft het dus een compleet nieuw ontwerp en geen uitbreiding van een bestaand systeem.

2.5 Vooronderzoek

Het eerste wat wij moesten doen na Eclipse geïnstalleerd te hebben was nadenken over hoe wij te werk wilde gaan. De eerste vraag die we tegenkwamen was de vraag of we een standalone of webapplicatie gingen bouwen. Omdat dit niet aan ons lag hebben wij voor het volgende gesprek met de begeleider en opdrachtgever een schema met de voordelen en nadelen gemaakt die we hebben afgesloten met onze eigen conclusie.

2.5.1 Voor- en nadelen webapplicatie vs standalone applicatie

Weegfactor: +++ Zeer belangrijk
++ belangrijk
+ minder belangrijk

<u>Webapplicatie.</u>
Voordelen:

+++ Makkelijk bereikbaar voor iedereen, het enige wat je nodig hebt is een apparaat met webbrowser(b.v. desktop pc, laptop, pda, smartphone).
++ Kan in één keer aan de “server-side” ge-update worden en die updates zijn dan gelijk voor alle gebruikers actief.

Nadelen:

+ Minder makkelijk te beveiligen, er wordt veel informatie over het Internet gestuurd.
+ De connectiviteit is heel belangrijk d.w.z. als het Internet door een of andere reden uitvalt is het gelijk niet meer te gebruiken.

+ Werkt langzamer dan een standalone applicatie, alle informatie moet steeds opnieuw over het internet gestuurd worden.(Minder belangrijk bij breedband verbindingen maar denk aan mogelijk langzamere verbindingen dialup etc.)

Standalone applicatie
Voordelen:

+ Er bestaan veel meer plugins e.d. die we kunnen gebruiken bij onze applicatie.
Is gemakkelijker te beveiligen, er wordt minder informatie over het internet gestuurd.
+ De connectiviteit is hier minder belangrijk omdat er telkens informatie op de client pc wordt opgeslagen en deze nog te bereiken is als het internet onverwachts uitvalt.
Werkt sneller dan web-applicatie's omdat het programma dan lokaal draait.

Nadelen:

+++ Moet bij iedereen die het wil gebruiken apart geinstalleerd worden.
Moet aan de "client-side" geupdate worden, dus de gebruikers moeten zelf de updates installeren en introduceert mogelijk afhankelijkheden m.b.t. operating systems of interpreters.

Advies:

Gezien de score van de zwaarder wegende redenen (elke + kan als een punt geteld worden) krijg je als resultaat bij de webapplicatie (5-3) en bij de standaloneapplicatie (2-3). Gezien deze uitkomst raden wij aan een webapplicatie te gebruiken.

Na het gezamenlijk doornemen van onze analyse heeft de opdrachtgever besloten ons advies te volgen en zijn wij dus begonnen met het bouwen van een webapplicatie.

2.5.2. Instellingen Eclipse

Na het installeren van Eclipse hebben we middels CVS(zie paragraaf 2.3.3) een kopie uit-gechecked op onze eigen computers. Op de repository(server) hebben wij na het volgen van een korte handleiding(zie branche instructies bijlagen) een branch aangemaakt waarop wij voor de zekerheid ook ons eigen werk konden opslaan zodat er in geval van nood altijd een recente versie beschikbaar kon zijn. Verder moesten we zorgen dat alle dependencies van EASY beschikbaar waren door de benodigde libraries(.jar files) te downloaden en deze in het classpath op te nemen.

2.5.3 EASY

De volgende stap was het doornemen van EASY. EASY(Electronic Archiving SYstem,zie begrippen) is een webapplicatie waar men archieven kan opslaan, en is ontworpen door DANS(Data Archiving and Networked Services, zie begrippen). Zoals in de inleiding al kort is uitgelegd kunnen mensen bij EASY datasets deponeren, bekijken en downloaden.
EASY is gemaakt met de volgende drie hoofddoelen:

- Het zo gemakkelijk mogelijk maken voor onderzoekers om data te kunnen opslaan en opvragen.
- Het zo gemakkelijk mogelijk maken voor de archivarissen om deze data te beheren.
- Om er zeker van te zijn dat de data die er opgeslagen is voor een onbepaalde tijd beschikbaar zal blijven.

Het opslaan van data is binnen EASY gescheiden van het tonen, opslaan en downloaden hiervan. Dit is de reden waarom EASY uit 2 hoofdelementen bestaat:

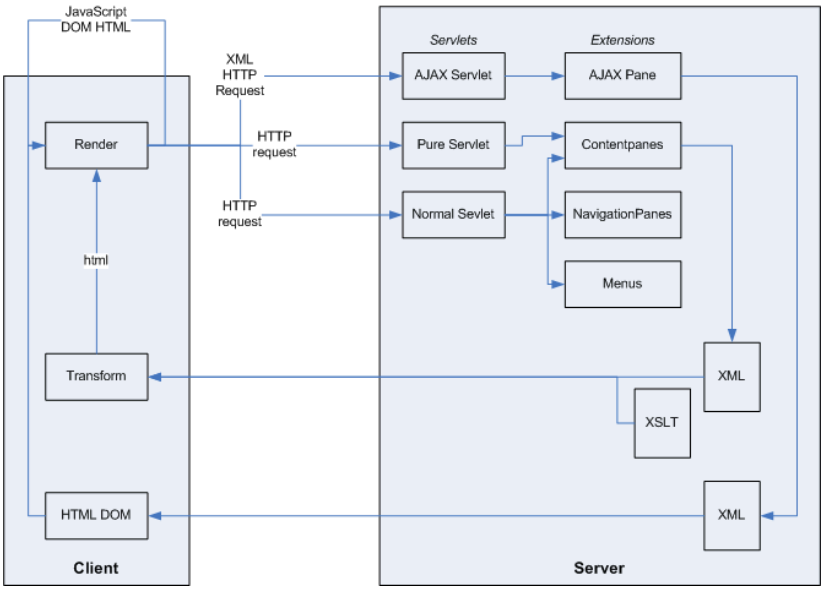
- AIPStore, voor het opslaan van de AIPs(Archival Information Packages)
- EASY, De front-end voor alle opgeslagen data.

De datasets(AIPs) worden bij EASY onderverdeeld in categorien(zie inleiding) die je in de applicatie aan de linkerkant van het scherm blijft zien.
De *aipstore* fungeert als een server waarmee via het socket systeem in Java datasets worden overgestuurd naar EASY. Een dataset(AIP) kan gezien worden als een pakket bestaande uit metadata, waarin het onderzoek beschreven staat, en een aantal losse bestanden waarin daadwerkelijke onderzoeksdata te vinden is. De bestanden in een AIP worden verdeeld over een drietal directories:

- Metadata, waarin de beschrijvingen van de bestanden en de totale dataset als XML bestanden te vinden zijn.
- Filedata, waarin de daadwerkelijke databestanden staan(zoals databases en codeboeken).
- Mgmdata, waarin managementdata over de dataset te vinden is, zoals de datum van invoer en publicatie, informatie over hoe de dataset in EASY gerepresenteerd moet worden.

Bepaalde acties van de EASY gebruikers worden gelogd in de EasyStatistics database(zie bijlage EasyStatistics) , en die informatie kunnen wij dan weer gebruiken om weer te geven. Na het vergelijken van de requirements met de gegevens in de database hebben we besloten onze eigen database te ontwerpen aangezien niet alle benodigde informatie gelogd wordt. Voor een beschrijving van deze database wordt u verwezen naar hoofdstuk 3 paragraaf 3.2.2.

In EASY heb je contentPanes, navigationPanes en Menu's. De NavigationPanes en Menu's vormen het navigatie gedeelte van de webapplicatie en in de ContentPanes wordt de opgevraagde informatie weergeven. Deze panes worden omgezet naar XML wat weer geconverteerd wordt naar XSLT. De XSLT bestanden worden door de webbrowser van de cliënt weer als HTML getoond.
Hieronder treft u een flow-diagram aan van EASY(hier is het deel van de AIPStore weggelaten). Links heb je een cliënt die dan een http request opstuurt naar EASY, Dit wordt via de contentPanes, NavigationPanes en Menu's omgezet naar XML en vervolgens XSLT, wat weer getoond word bij de cliënt.



Figuur 2.5.3.2: werking van EASY

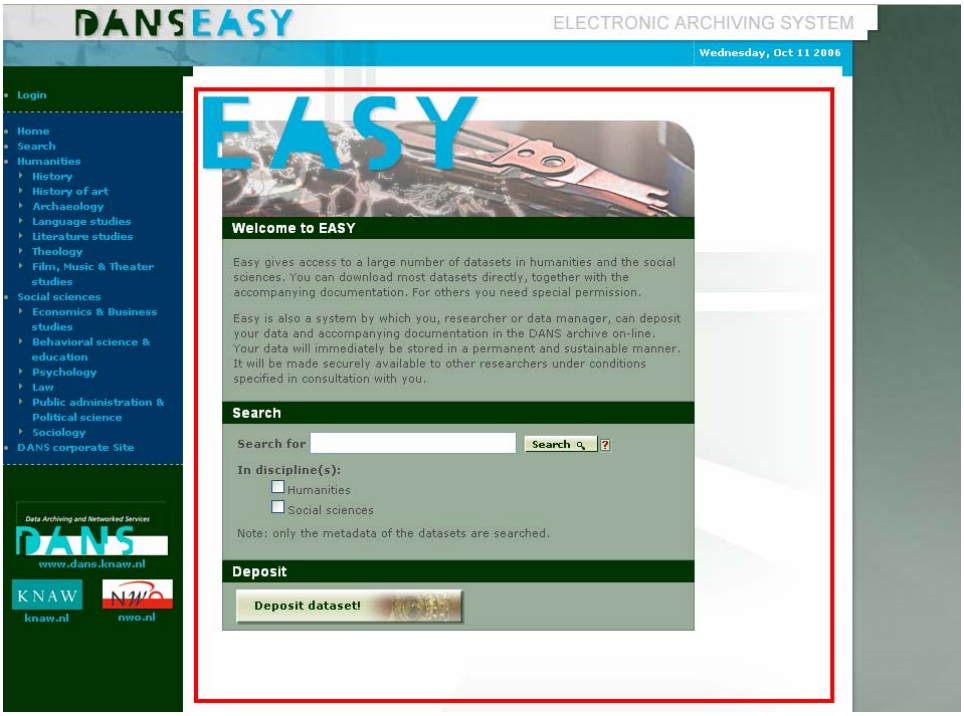


Fig. 2.5.3.3 Voorbeeld ContentPane(afgebakende gedeelte)

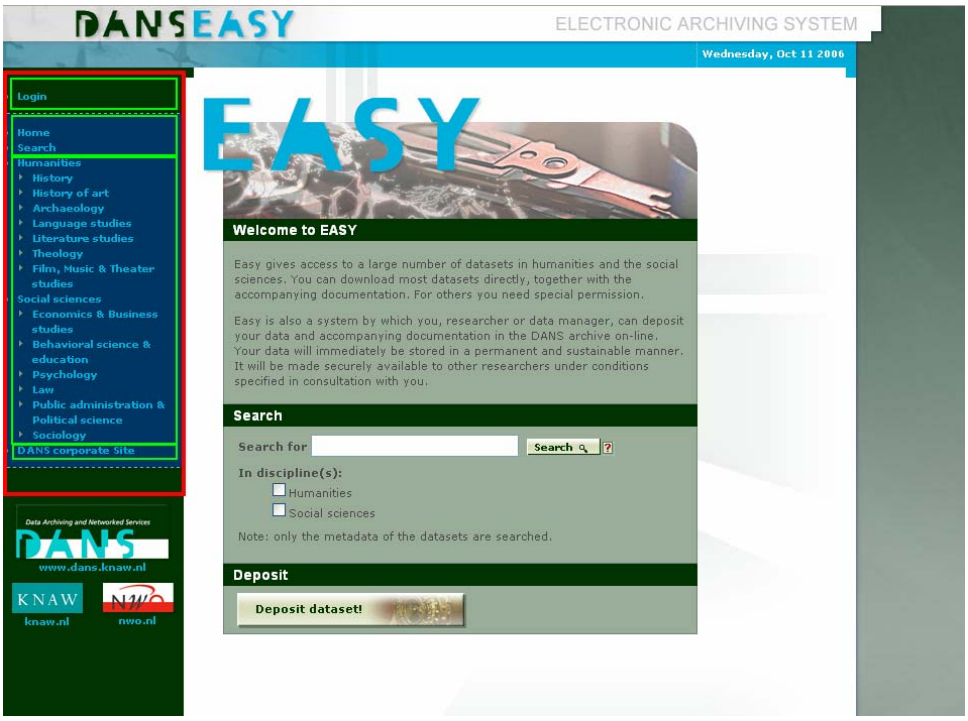


Fig. 2.5.3.4 Voorbeeld navigatiePane(afgebakende gedeelte)

In figuur 2.5.3.4 zie je binnen het afgebakende gedeelte tevens de categorien waarin bij EASY Datasets(AIPs) te vinden zijn en in figuur 2.5.3.3 zie je binnen het afgebakende gedeelte de informatie die je opvraagt.

2.6 Requirements

In de eerste week van het project hebben wij gesprekken gehad met onze opdrachtgever Drs. Milco Wansleeben, de Directeur van dans Dr. Peter Doorn en adjunct directeur Dr. Henk Harmsen. Deze informatie hebben wij samengebracht tot één geheel die we hebben uitgewerkt tot een MoSCoW Diagram(zie bijlagen). Een MoSCoW diagram is een diagram waarmee men heel makkelijk de requirements kan groeperen en waar heel makkelijk de prioriteit van elke specifieke requirement uit af te lezen is. (Must have, Should have, Could have, Won't have). De eisen na de gesprekken voor de omzetting naar MoSCoW treft u in de volgende paragraaf (Voor een uitleg van de termen die in de eisen gebruikt worden verwijzen wij u naar http://easy.dans.knaw.nl 2007).

2.6.1 Functionele eisen

+++ Must Have
++ Should Have
+ Could Have
- Won't Have

1. *Gebruikers:*
   - *1.1.* +++ Het aantal gebruikers moet kunnen worden opgevraagd.(bv archeoloog of researcher)
   - *1.2.* +++ Welke gebruikers welke soort dataset(bv archeologische dataset) gedeponeerd hebben en aantal per depositor.
   - *1.3.* +++ Totaal aantal depositors per jaar/kwartaal/maand.
   - *1.4.* +++ Totaal aantal bezoekers van EASY onderverdeeld in wel/niet gedownload per maand/kwartaal/jaar.
   - *1.5.* ++ Aantal geregistreerde/niet geregistreerde gebruikers(nieuw)per maand/kwartaal/jaar/totaal.
   - *1.6.* ++ Permission requests die nog open staan per gebruikers type.
   - *1.7.* **- Per gebruiker het aantal datasets in de drafts en sinds wanneer, de date created wordt niet bijgehouden(advies; bijhouden als het echt nodig is).**

2. *Dataset(AIP's):*
   - *2.1.* +++ Aantal gedownloade files per AIP per category.(zonder metadata.pdf's).
   - *2.2.* +++ Welke aips hebben nog geen gepubliceerde files.
   - *2.3.* +++ (totaal)Aantal aanwezige AIP's (tot nu toe) per gebruiker/categorie.
   - *2.4.* +++ aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.5.* +++ Totaal aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.6.* +++ De verstreken tijd tussen het submitten van een AIP en het publishen van een AIP.(Als het binnen 2(5) dagen niet gebeurt waarschuwing verzenden.)
   - *2.7.* +++ Aantal bekeken datasets jaar/kwartaal/maand.(ook aantal hits per dataset)
   - *2.8.* ++ Welke hebben nog geen jumpoff pages.(evt een extra kolom).
   - *2.9.* ++ Gebuikersrechten en dateavailable per aip.
   - *2.10.* ++ De "best permission" van resticted datasets weergeven.
   - *2.11.* ++ Aantal datasets die terug zijn naar Draft(niet gepublished zijn na workflow).
   - *2.12.* ++ Verlopen restricties van AIP's, verzonden waarschuwings mail.
   - *2.13.* ++ Welke datasets zitten nog in de drafts en -hoe lang.
   - *2.14.* + Welke bestanden in een AIP hebben geen extra metadata.
   - *2.15.* + Waar vandaan wordter naar welkeAIP's/easy gelinkt???
   - *2.16.* + Aantal ingevulde metadata velden per aip.
   - *2.17.* + veranderingen in toegangscategorieen.

2.18. - Wat is de responsetijd per AIP.

3. *Categorie:*
   - *3.1.* +++ Aantal depositors per jaar/kwartaal/maand per categorie
   - *3.2.* ++ Welke AIP's zijn per categorie gepublished op volgorde van datum.
   - *3.3.* ++ Files per categorie:
     - *3.3.1.* +++ Welke bestanden zijn het meest gedownload per categorie.
     - *3.3.2.* +++ (Totaal)Aantal gedownloade bestanden per jaar/kwartaal/maand per categorie.
     - *3.3.3.* ++ (totaal)Aantal gedeponeerde bestanden(aantal files) per jaar/kwartaal/maand.
     - *3.3.4.* + Welke bestandstypen komen per categorie voor.
     - *3.3.5.* + Aantal files in de AIP's verdeeld per category. En ook aantal MB's en + bestandsstype.(ext of bestandsherkenning).

4. *Export functies:*
   - *4.1.* +++ Naar excel
   - *4.2.* **- Xml export van de project (dublin core) metadata waarin alle AIP's van een vooraf bepaalde groep zitten.**

5. Top/Laatste 10 per /allertijden/jaar/kwartaal/maand:
   - *5.1.* +++ Gebruikers die het meeste geupload hebben .
   - *5.2.* +++ Gebruikers die het meeste gedownload hebben.
   - *5.3.* **+++ Meest aangevraagde AIP's en datafiles.**
   - *5.4.* +++ Laatste 10 gedeponeerde bestanden + acces categorie(als extra kolom).
   - *5.5.* +++ Laatste 10 bekeken AIP's.
   - *5.6.* +++ Actiefste gebruiker(down+upload).
6. *Algemeen:*
   - *6.1.* +++ Zoektermen per dag/ maand en meest gezochte termen.
   - *6.2.* – browse acties per dag/maand.
   - *6.3.* - hoe vaak mislukt het deponeren, gaan mensen verder nadat het een keer mislukt is?
   - *6.4.* - hoe lang zijn mensen bezig met het deponeren?(zie punt 1.7)

2.6.2 Niet-functionele eisen

User Interface

De user-interface moet te gebruiken zijn door de opdrachtgever en andere klanten en er moet bij aanmelden een overzicht te zien zijn van de basis informatie (aantal downloads, aantal uploads etc.)

Documenten

Er moet een handleiding gemaakt worden die moet helpen als de gebruiker niet weet hoe hij bepaalde informatie kan vinden.

Systeemaanpassingen

Het systeem moet zo in elkaar zitten dat het gemakkelijk implementeerbaar is op een nieuw systeem d.w.z. het moet zo onafhankelijk mogelijk gemaakt worden zodat het niet uitmaakt op wat voor manier EASY in elkaar zit.

3. Ontwerp

Belangrijk voor het software ontwikkeltraject is het kiezen van een ontwikkel methodiek die het beste past bij het bedrijf en het te ontwikkelen product. DANS hanteert een aangepaste versie van de eXtreme Programming ontwikkelmethodiek[11]. Hierdoor was er in het begin geen gedetailleerd ontwerp. In dit hoofdstuk wordt het ontwerp besproken, in de eerste paragraaf wordt een globale ontwerp van ons systeem beschreven en daarna het gedetailleerd ontwerp.

3.1 Globaal Ontwerp

3.1.1 Extreem Programming

Voor het ontwikkelen van Easy Usage Statistics wordt ook de XP methodiek[6] gebruikt. De methodiek wordt aangepast zodat deze beter past bij het project, rekening houdend met de beschikbare tijd en omvang. De belangrijkste richtlijnen zijn als volgt;

- Software wordt ontwikkeld in iteraties van 1 week.
- Bij het begin van elke week vindt een planning game plaats. Tijdens de planning games worden de resultaten van de afgelopen iteratie besproken. Tevens wordt er beoordeeld in hoeverre de resultaten aan de wensen van de klant voldoen en worden de taken voor de volgende iteratie ingepland.
- Bij elke iteratie worden de geplande functionaliteiten uitgewerkt.
- Code moet 'test-first' ontwikkeld worden.
- Code 'review' wordt door andere ontwikkelaars gedaan.
- Code 'refactoring' (herstructurering) moet regelmatig toegepast worden om zoveel mogelijk fouten uit het systeem te halen.

Het is niet altijd gelukt om "test-first" te ontwikkelen, de tests werden daarna geschreven. De reden hiervoor was dat bepaalde klassen moeilijker te testen waren dan anderen.Dit kwam grootendeels omdat er veel classes gebruik maakten van methodes uit andere classes en omdat niet overal evenmakkelijk testdata gemaakt kon worden om de assert-regels van het testen(zie hoofdstuk 5) konden toepassen. Verder hebben we uiteindelijk te veel tijd gestoken in het afmaken van de requirements en wat minder in het schrijven van tests. Volgens de XP methodiek behoor je als je de wekelijkse planning niet haalt gewoon je planning aan te passen en alle requirements die wel af zijn goed te testen en documenteren. Maar omdat wij wat te enthousiast bezig waren wilden we ook alle requirements van de planning afhebben waardoor we uiteindelijk tijd tekort kwamen.
Bij de XP methodiek behoort ook het zogenaamde 'pair' programming. Dit houdt in dat er in paren geprogrammeerd wordt. Een alternatief hierop is de 'peer' programming[11]. Er wordt eerst met elkaar besproken hoe de functionaliteit geïmplementeerd wordt, gevolgd door een code review.

In het begin konden we niet meteen een gedetailleerd ontwerp van het systeem maken. Wel hebben we onderzocht welk design pattern het beste zou passen bij dit project. Het gebruikte pattern wordt in de volgende paragraaf beschreven.

3.1.2 Design Patterns

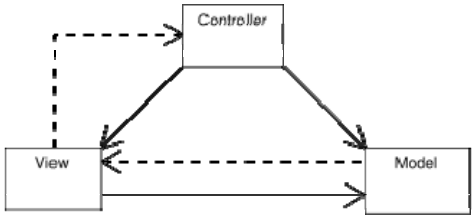
Voor het ontwikkelen van webapplicaties is het belangrijk om een goede design pattern te gebruiken eventueel in combinatie met andere patterns. Bij dit project wordt het MVC pattern in combinatie met de DAO en Abstract Factory pattern gebruikt.

Model View Controller Pattern

Het Model View Controller Pattern[16] werd gekozen omdat deze goed past bij webapplicaties. Door dit pattern te gebruiken wordt de applicatie opgedeeld in drie lagen met verschillende verantwoordelijkheden, hierdoor wordt de code beter leesbaar en herbruikbaar. Hieronder volgt een beschrijving van de verschillende componenten van dit design pattern:

- Model: Een representatie van de data/tabellen in de database.
- View: De informatie wordt via de views aan de gebruikers getoond.
- Controller: Verantwoordelijk voor het afhandelen van de requests die via de view binnen komen.

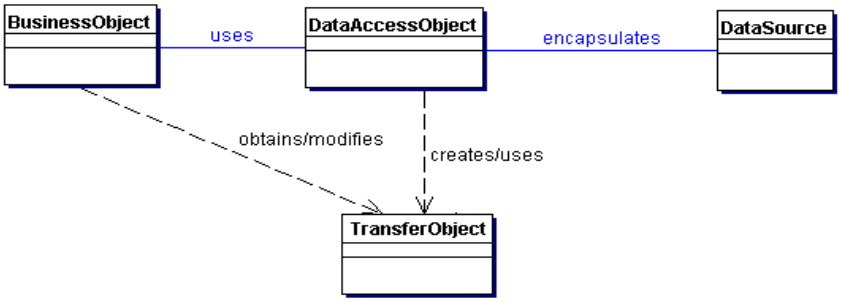
In paragraaf 3.2.1 kunt u zien welke packages bij dit pattern horen. Hieronder treft u een kleine figuur die de MVC pattern toont.



Figuur 3.1.2.1: MVC Model View Controller Pattern.[16]

Data Access Object Pattern

Het Data Access Object (DAO) Pattern is nodig om de toegang tot de data bron te ontkoppelen van de business logica. Het DAO is dan verantwoordelijk voor de verbinding met de data bron en voor het ophalen en opslaan van de gegevens[7]. Uit de volgende illustratie is te zien hoe de verschillende componenten van het DAO pattern met elkaar in verbinding staan.

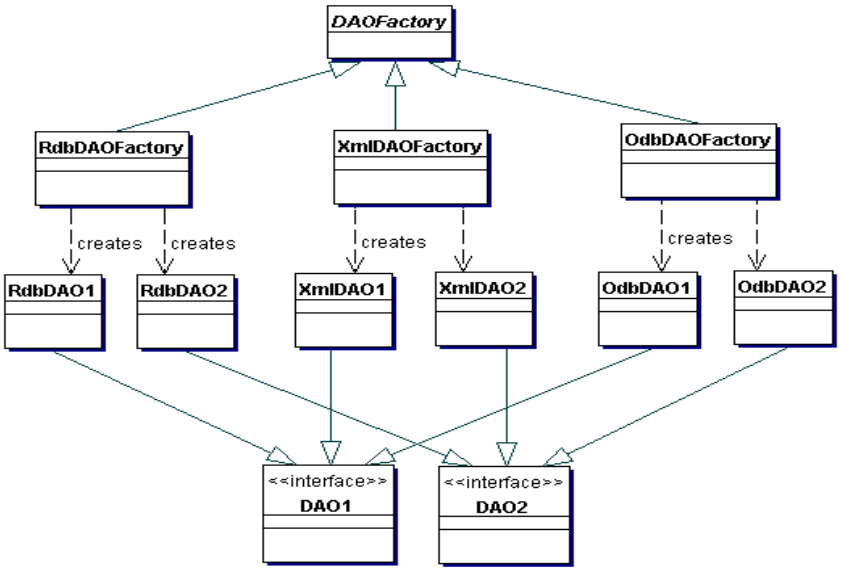


Figuur 3.1.2.2: DAO pattern principe.[7]

Abtract Factory Pattern

Naast het DAO pattern wordt ook gebruik gemaakt van het Abtract Factory Pattern(AFP)[2]. Het AFP is heel belangrijk gebleken omdat we 2 databases gebruiken en er hiermee gemakkelijk tussen de twee geschakeld kan worden(zie hoofdstuk 4). Omdat we te maken hebben met twee databases is het gebruik van dit patroon belangrijk. Als later andere databases gebruikt zullen worden kan deze makkelijker toegevoegd worden aan het gehele systeem. Het enige wat er gedaan moet worden is het

implementeren van een nieuwe concrete DAOFactory en het implementeren van de onderliggende DAO's.



Figuur 3.1.2.3: Abstract Factory Pattern.[7]

3.2 Gedetailleerd Ontwerp

Na een aantal iteraties werd het steeds duidelijker hoe het systeem eruit zou moeten zien. In de volgende paragraaf worden de onderdelen van het systeem opgedeeld per package weergeven.

3.2.1 Systeem ontwerp

Het systeem is ontworpen volgens de MVC, DAO en Abstact Factory ontwerp patronen. Hier volgt een overzicht van de pattern componenten en de bijbehorende packages.

MVC pattern:

- Model: usageStatistics.model
- View: usageStatistics.view
- Controller: usageStatistics.content

DAO pattern:

- Bussiness objects: usageStatistics.business
- DAO: usageStatistics.dao
- Transfer objects: usageStatistics.model
- Datasource: easyStatistics en additional_information databases

Abstract Factory pattern:

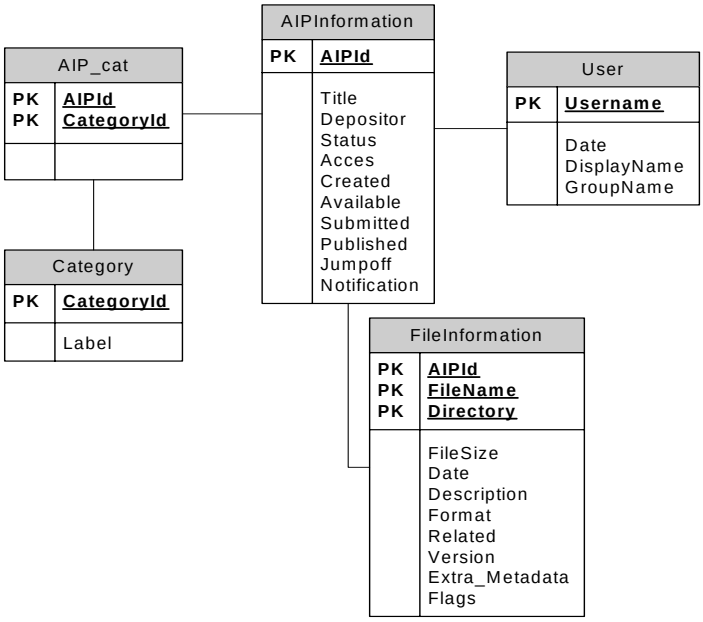
- Factory: usageStatistic.dao

Voor een gedetailleerd overzicht van de plugins en de klassen die daarin een rol spelen wordt u verwezen naar bijlage 2.

3.2.2 Database Ontwerp

In deze paragraaf wordt de additional_information database beschreven die gebruikt werd om extra informatie op te slaan en op te halen. De easyStatitics database was ontworpen door Taner Gedikoglu, een ontwikkelaar bij DANS, en de beschrijving daarvan is te vinden in de bijlage.

Additional_information database model:



Figuur 3.2.2.1: Database model

Beschrijving:

Hier volgt een beschrijving van de verschillende tabellen in de database:

- AIPInformation: beschrijving van een dataset volgens de Dublin Core standaard.
- aip_cat: aipId en categoryId koppelen, Hieruit kan afgeleid worden welke dataset tot welke category behoord.
- FileInformation: hier in zit de informatie over alle files en tot welke dataset deze files behoren.
- user: de user informatie.
- category: alle categorieën in EASY.

4. Implementatie

In dit hoofdstuk wordt beschreven hoe de applicatie geïmplementeerd is, de problemen die zich voor deden en hoe deze problemen opgelost zijn. Zoals eerder besproken wordt de XP ontwikkelmethodiek gebruikt. In iteraties van één week werden de functionaliteiten aangepakt, dus in het begin was er geen gedetailleerd ontwerp.

De volgende informatie was wel bekend. Het moet een webapplicatie worden die toegankelijk is voor de archivarissen via EASY. De statistieken zitten allemaal in de easyStatistics database, en de rest van de informatie zoals gegevens over de AIP's zitten in xml bestanden.

In de volgende paragraaf wordt de basis opzet van EUS beschreven. Daarna wordt er dieper ingegaan op de implementatie details met daarin zaken zoals het ophalen en tonen van de statistieken en de database Job voor de extra database.

4.1 Basis opzet

In de tweede week van het project zijn we begonnen met het opzetten van een basis voor EUS. Zoals eerder besproken wordt EASY dmv het eclipse framework ontwikkeld. Het EASY project bestaat uit verschillende plugins/packages en EUS wordt een project/plugin binnen dat project. Dus het eerste dat gedaan moet worden is het creëren van een nieuwe plugin, namelijk, nl.knaw.dans.easy.usageStatistics.

De plugins/packages binnen EUS beginnen allemaal met nl.knaw.dans.easy.usageStatistics en deze zijn verdeeld volgens het Model View Controller en de Data Acces Object pattern principe. In het begin was de MVC indeling niet helemaal duidelijk maar naarmate het project vordert kregen we een beter overzicht van het systeem. Het volledige begin indeling is te zien aan het einde van deze paragraaf.

Naast de MVC packages hadden we ook de usageStatistics.bussiness en de usageStatistics.dao. De laatste twee kwamen voort uit het DAO pattern principe, beschreven in hoofdstuk 3.
De bedoeling van het gebruik van DAO is om alle toegang tot de data bron te ontkoppelen van de business logica. Naast de Data Acces-, Business objecten en de data bron zijn Transfer objecten ook nodig en die zijn bedoeld als data carrier. Deze Transfer objecten zitten in de easyUsageStatistics.model package.

Na een gesprek met Henk van de Berg, ontwikkelaar bij DANS, bleek dat de controller, dus de package content, ook voor een gedeelte als view optreedt omdat die ook het xml(inhoud) genereert. Niet alleen de xslt treedt als view op maar ook als een klasse die de inhoud moet genereren. Later werden deze twee van elkaar gescheiden. Voor een gedetailleerde versie van de gehele EUS plugin wordt u verwezen naar hoofdstuk 3 paragraaf 3.2.1.

Begin indeling volgens de MVC en DAO patterns:

MVC pattern:

- Model
  - usageStatistics.model
- View
  - Src_xslt
- Controller
  - usageStatistics.content

DAO pattern:

- Business Objects
  - easy.usageStatistics.business
- Data Access Objects
  - easy.usageStatistics.dao
- Data source
  - easyStatistics database
- Transfer Objects
  - easy.usageStatistics.model

In de volgende paragraaf worden de implementatie details beschreven.

4.2 Implementatie details

In deze paragraaf worden de implementatie details beschreven. Het systeem kan in het algemeen als volgt verdeeld worden: Het ophalen van de statistieken van easyStatistics database, het ophalen en invoeren van de extra data en het tonen van de statistieken aan gebruikers.

4.2.1 Ophalen easy statistieken

De statistieken zitten in de easyStatistics database, in hoofdstuk 3 staat de specificatie. De database heeft in totaal drie tabellen, statistics, statisticvalues en statisticevents. Als er een bepaalde event nodig is, bijvoorbeeld een lijst van deposits, dan moet eerst de statistic event id van de event, dataset_deposit, opgezocht worden. Vervolgens worden in de tabel statistics de rijen opgehaald waarvan de statistic event id overeenkomt met die van de deposit event. In het tabel statisticvalues zit extra informatie over een bepaald gelogde event. Deposits kunnen dus als volgt worden opgehaald;

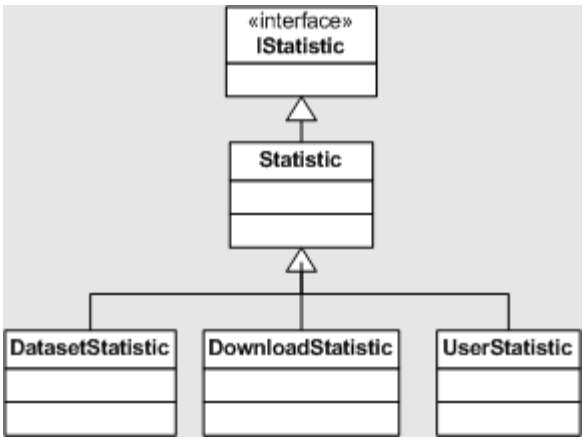
```
SELECT * FROM statistics WHERE statisticEvent=’9’
SELECT * FROM statisticvalues WHERE statisticid='342'
```

De eerste query haalt alle deposit statistics uit de database, het statistic tabel heeft naast statisticEvent nummer, user name, date en ip-addres ook een statistic id. Deze statisticic id wordt in de tweede query gebruikt om de rest van de informatie op te halen in de statisticvalues tabel. In deze tabel zit extra informatie zoals aipId van de gedeponeerde dataset of voor het geval van een download, ook de filenaam en file grootte.

Zoals al eerder genoemd wordt het DAO pattern gebruikt om de toegang tot de data bron te ontkoppellen van de business objecten. Dus voor het ophalen van de statistieken zijn data access objecten gebruikt. De verschillende events hebben we in een aantal groepen verdeeld en voor elk groep één data acces object gedefinieerd. Hier volgen de verschillende data access objecten.

- Statistics. Voor het ophalen van alle statistieken en is afhankelijk van het statisticEvent nummer. Deze moet meegegeven worden in de constructor van de klasse. In het voorbeeld hierboven komt het deposit event overeen met nummer 9.
- DatasetStatistics. Voor het ophalen van de deposit, view en publish events.
- DownloadStatistics. Voor het ophalen van de download events.
- UserStatistics. Voor het ophalen van de events die te maken hebben met een user, dus logins, logouts, visits en registration.
- SearchTermsStatistics. Voor het ophalen van de search questions.

Een ander onderdeel van het DAO pattern is het gebruik van Transfer objecten. Deze worden gebruikt als een soort informatie drager. Dus voor elk van de bovenstaande DAO ’s behalve de SearchTermsStatistics is er een transfer object geïmplementeerd. Voor alle statistieken geldt het Statistics transfer object, dit bezit standaard informatie over een statistiek. Er zijn ook andere statistieken die extra informatie nodig hebben, bijvoorbeeld een deposit event heeft onder andere ook een aipId van de gedeponeerde dataset. Dus de rest van de transferobjecten zijn als subklassen van het Statistics transfer object geïmplementeerd. Ter illustratie is er een eenvoudige versie van een klassendiagram hieronder opgenomen.



Figuur 4.2.1.1: klassendiagram

Het ophalen van de benodigde statistieken wordt uiteindelijk geregeld door het business object. Het business object, UsageStatistics, maakt gebruik van een Transaction klasse voor het correct afhandelen van een transactie. De Transaction klasse verzorgt de transactie met transaction begin, commit en rollback. De transactie wordt gestart met begin, commit wordt aangeroepen als alles goed verloopt en als er problemen optreden een rollback. Ter illustratie volgt een code fragment uit de klasse UsageStatistics. In de bijlage wordt het volledige klassendiagram geïllustreerd met de bijbehorende methodes. Hier is duidelijk te zien hoe de transacties afgehandeld worden.

```
public List<IStatistic> getDatasetStatistics(int statisticEventId){

        List<IStatistic> ds = new ArrayList<IStatistic>();

        try {
                daoFactory.begin();

                IDatasetStatisticsDAO d = daoFactory.getDatasetStatisticsDAO();
                ds = d.getDatasetStatisticsList(statisticEventId);

                daoFactory.commit();

        } catch (Exception e) {
                daoFactory.rollback();
        }

        return ds;
}
```

Uit het code fragment is ook het gebruik van een daoFactory object te zien. De Factory klasse is een abstracte klasse die de Transaction klasse implementeert. De Abstract Factory Pattern wordt hier gebruikt omdat we te maken hebben met verschillende databases, de easyStatistics en additional_information database. Elke database heeft zijn eigen DAOFactory implementatie, concrete factory, en de abstract DAOFactory klasse heeft een methode getDAOFactory. Hiermee kan de concrete factory klasse die nodig is, gekozen worden. Voor meer informatie over Abstract Factory Pattern klassen wordt u verwezen naar de klassendiagrammen in de bijlage.

4.2.2 Ophalen extra gegevens

Na het ophalen van de easy statistieken moeten deze worden bijgevuld met extra gegevens. Gegevens als de titel van een dataset en display name van een user worden niet gelogd. Bij het tonen van een tabel met deposits bijvoorbeeld, is het beter om in plaats van de aipId van een dataset de titel weer te

geven. Naast het tonen van de gebruiksstatistieken moet het ook mogelijk zijn om gegevens over datasets, files in de datasets, gebruikers en de verschillende disciplines te zien. Deze gegevens zitten opgeslagen in verschillende xml files in AIPStore.

In het begin hebben we besloten, gezien de tijd, zoveel mogelijk gebruik te maken van de functies van EASY. Het ophalen van de gegevens ging via die bestaande functies. Lokaal hadden we niet meer dan 30 datasets om mee te testen en alles verliep soepel en zonder problemen. Later besloten we om met de data van de productie server te testen en de problemen kwamen meteen naar voren. We hadden te maken met een lijst van meer dan 3000 datasets. EASY heeft een methode om de metadata op te vragen. De data zitten in AIPStore en EASY communiceert via sockets met AIPStore. Er wordt telkens een nieuwe socket geopend voor elke metadata overdracht. Dat gaat behoorlijk traag als we temaken hebben met zoveel datasets.

Naast de performance problemen kregen we ook te horen dat de AIPStore over een paar maanden niet meer gebruikt zal worden. AIPStore wordt vervangen door FEDORA, voor meer informatie hierover wordt u verwezen naar hoofdstuk 8. Om deze redenen werd er besloten om een database te gebruiken. De benodigde gegevens worden direct gehaald uit de xml files en opgeslagen in een database. Deze gegevens kunnen dan veel sneller opgevraagd worden en bovendien is EUS niet meer afhankelijk van AIPStore. In paragraaf 4.2.5 wordt verder uitgelegd hoe de xml files ingelezen worden en hoe de database gevuld wordt met deze gegevens.

Nog een ander reden om een extra database te gebruiken is het feit dat de easyStatistics database pas in mei 2007 in werking trad. Dus de gebruiksstatistieken voor deze datum zijn niet gelogd terwijl deze gegevens ook zeer belangrijk zijn. Er zijn nog steeds gegevens dat onbekend blijven, bijvoorbeeld de registratie datum van een gebruiker dat zich voor mei 2007 geregistreerd heeft of het datum van het publiceren van een dataset. De ontbrekende gegevens worden expliciet in de handleiding beschreven, die te vinden is in de bijlage.

Het ophalen van de gegevens uit de extra database gaat in principe op dezelfde manier als het ophalen van de statistieken. Hiervoor moet wel de juiste DAOFactory gekozen worden, zie paragraaf 4.2.1.

4.2.3 Statistieken tonen

Na het ophalen van de benodigde gegevens moeten deze getoond worden aan de gebruiker. De statistieken komen meestal in de vorm van een lijst met objecten terug. Afhankelijk van het soort object wordt er een view klasse aangemaakt die verantwoordelijk is voor het genereren van de xml die door de xslt stylesheet omgezet wordt naar HTML. Neem als voorbeeld het tonen van een lijst met gebruikers. Met deze lijst wordt een view object aangemaakt en met behulp van de methode toXML() wordt de nodige xml gegenereerd. Voor elke view moet er een xslt stylesheet gemaakt worden. Dus voor een userView.java is er ook een UserView.xslt. Hieronder volgt een fragment van de xml en xslt van de userView en het resultaat, dus wat de gebruiker te zien krijgt. Hier is te zien hoe het tabel gevormd wordt. In de tweede xslt staat als het ware dat voor elke user in de xml een nieuwe rij wordt aangemaakt met daarin de cellen userName, displayName, group, date registered, deposits en aantal aips. Alle tabellen in EUS worden op deze manier gemaakt, dus aan de hand van een lijst met objecten.

```xml
<users total="312" pages="16">
   <user page="1">
       <userName>adrie</userName>
       <displayName>Adrie Ufkes</displayName>
       <group>archaeologists /researchers</group>
       <dateReg>2007-05-03</dateReg>
       <deposits>0</deposits>
       <aips>0</aips>
   </user>
   <user page="1">
       <userName>jurate249</userName>
       <displayName>jurate</displayName>
       <group>researchers</group>
       <dateReg>2007-05-04</dateReg>
       <deposits>0</deposits>
       <aips>0</aips>
   </user>
</users>
```

Figuur 4.2.3.1: xml fragment gegenereert door userView.toXml() methode.

```xml
<xsl:template name="users">
    <xsl:for-each select="user">
        <tr id="{@page}" style="display:none">
            <td>
                <xsl:value-of select="userName"/>
            </td>
            <td>
                <xsl:value-of select="displayName"/>
            </td>
            <td>
                <xsl:value-of select="group"/>
            </td>
            <td>
                <xsl:value-of select="dateReg"/>
            </td>
            <td>
                <xsl:value-of select="deposits"/>
            </td>
            <td>
                <xsl:value-of select="aips"/>
            </td>
        </tr>
    </xsl:for-each>
</xsl:template>
```

Figuur 4.2.3.2: fragment uit userView.xslt.

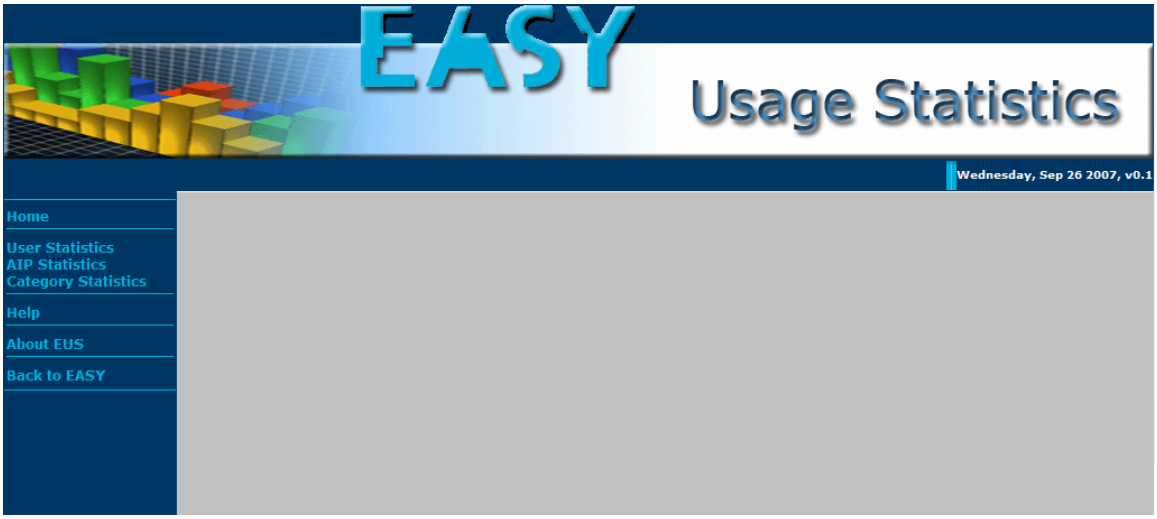
adrie	Adrie Ufkes	archaeologists /researchers	2007-05-03	0	0
jurate249	jurate	researchers	2007-05-04	0	0

Figuur 4.2.3.3: screenshot van een tabel van users.

Naast de views moeten ook contentPanes voorzien worden van een xslt. De tabellen en grafieken komen op deze contentPanes terecht. De requirements werden vanaf het begin ingedeeld in verschillende categorieën, hoofdstuk 1. De categorieën worden hier nogmaals genoemd.

- Algemeen, de algemene informatie zoals top 10's en laatste gedeponeerde datasets.
- Gebruikers, informatie per gebruiker.
- Datasets, informatie per dataset.
- Disciplines, informatie per discipline

Voor elke categorie is er een contentpanes geïmplementeerd, deze zijn gelijk de vier hoofd contentpanes waar de gebruiker naar toe kan navigeren. Hier volgt een screenshot van de menu structuur van EUS:



Figuur 4.2.3.4: screenshot van de hoofd pagina

Aan de hand van het gekozen menu item wordt een van de vier contentpanes getoond in het grijze gebied. De menu structuur is onderdeel van de stylesheet eusmain.xslt. Zoals eerder genoemd is EUS een onderdeel van EASY en met de parameter windowStyle kan aangegeven worden welk soort scherm getoond moet worden. EASY kent een aantal soort schermen en voor dit project hebben we ervoor gekozen om een nieuwe windowStyle te gebruiken. Het opvragen van een pagina in EASY gebeurt door middel van de volgende Internet adres:

http://*clienturl*/dms/command=*command*&windowStyle=*windowStyle*

Door het meegeven van de parameter windowStyle=EUSContent in het adres wordt de bovenstaande pagina standaard getoond. En aan de hand van het command parameter wordt de gewenste contentpane in het grijze gebied zichtbaar. EUS kent de volgende commando's: showUsageStatistics, showUserStatistics, showAIPStatistics en als laatste showCategoryStatistics.

De contentPane klasse is dus verantwoordelijk voor het afhandelen van de commando's die binnen komen bij een gebruikersactie en worden hierdoor ook gezien als Controllers in het MVC Pattern. Er zijn hier drie belangrijke methodes, namelijk isVisible, doProcessing en generateContent. De methode isVisible wordt gebruikt om aan te geven of een contentPane zichtbaar is of niet en aan de hand van het commando wordt waar of onwaar terug gegeven. In doProcessing worden de business objecten aangemaakt, de gewenste lijsten binnengehaald en ook verwerkt. Deze worden opgeslagen in de session voor verder gebruik. Als laatste worden in generateContent de lijsten uit de session gehaald om met behulp van de views de gegevens aan de gebruiker te laten zien.

Een ander vereiste was dat de gegevens ook per maand, kwartaal en jaar opgevraagd moeten worden. Hiervoor is er een date selector ontwikkeld, die in alle contentPanes geplaatst kan worden. De date selector is als een view geïmplementeerd, dus een dateSelector klasse en zijn bijbehorende datSelector.xslt. De informatie voor de xslt stylesheet worden door de klasse gegenereerd. Hier volgt een screenshot van de dateSelector:

Figuur 4.2.3.5: screenshot van de dateSelector

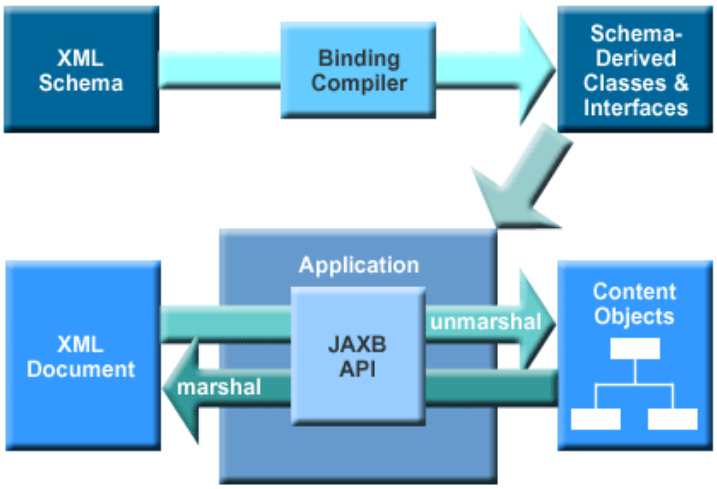
4.2.5 Database Job

Zoals eerder genoemd wordt er ook gebruik gemaakt van een database voor extra gegevens zoals de titel van een AIP, aantal files behorend bij een dataset en welke dataset tot welke categorie behoort. De nodige informatie wordt opgeslagen in XML formaat en deze XML documenten moeten dus ingelezen worden. In deze paragraaf worden enkele belangrijke onderdelen van de Job beschreven die de xml documenten inlezen en de nodige informatie opslaat in de database.

Elke AIP heeft zijn eigen folder met data zoals beschreven in paragraaf 2.5.3, hier wordt de inhoud van deze folder nogmaals weergeven;

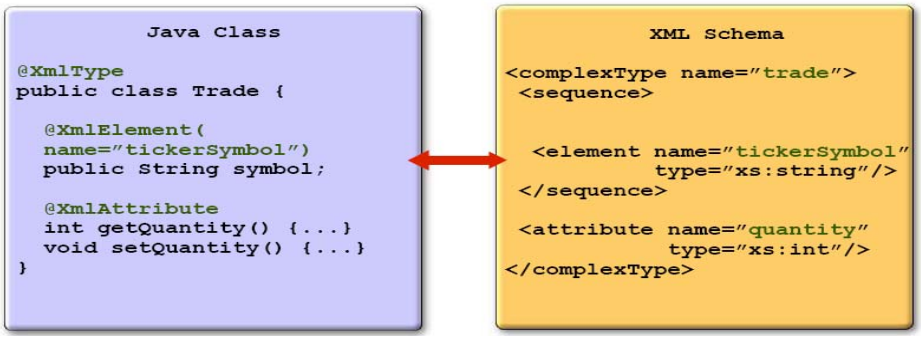
- filedata folder; de files die bij een AIP horen.
- metadata folder met;
    - dfm folder, hierin zit voor elke file de filemetadata. Dit is een xml bestand in de vorm van *file*-md.xml met informatie over de file zelf.
    - dc-simple of dc-arch folder. Hierin zit de data.xml file met informatie over een AIP volgens de Dublin Core standaard.
- mgmdata(management data) folder;
    - mgmdata.xml, informatie over de AIP die door AIPStore en EASY gebruikt wordt.
    - filerights.xml, informatie over de toegankelijkheid van de files.
    - permissiondata.xml

Voor het inlezen van de bovenstaande XML documenten wordt JAXB gebruikt. JAXB staat voor Java Architecture for XML Binding. Dit is een Java API die het gemakkelijk maakt om XML documenten te manipuleren vanaf applicaties geschreven in de programmeertaal Java. De onderstaande illustratie is gedownload van de SUN Developer Network site en is een beschrijving van hoe een XML document met behulp van JAXB gemanipuleerd kan worden[9].



Figuur 4.2.5: gebruik van JAXB[9]

Om informatie uit een XML document te halen is alleen het onderste gedeelte van het figuur van belang, vanaf het XML document gedeelte tot aan de Content Objects. Wel zijn er zogenaamde Java Annotated Classes nodig, deze zijn Java representaties van het XML schema. In de bovenstaand figuur kunt u zien hoe dat deze klassen gegenereerd kunnen worden met behulp van de Binding Compiler van JAXB. Hieronder volgt een illustratie van een Java Annotated Class met bijbehorend XML Shema, die genomen is uit een presentatie van de SUN JavaOne vergadering[10].



Figuur 4.2.6: XML Schema met zijn Java Respresentatie[9]

De Java Annotated Classes die nodig waren hebben wij zelf geïmplementeerd, dus niet met behulp van een Binding Compiler. Dus voor elk XML document is een dergelijk klasse gemaakt en hieronder volgt een voorbeeld van een metadata van een AIP, data.xml en de bijbehorende Java representatie.

```
@XmlRootElement(name = "EasyMetadata")
@XmlAccessorType(AccessType.NONE)
public class EUSAIPMetadata implements IEasyMetadata {

        @XmlElement(name = EMD_ELEMENTNAME, namespace = EMD_NAMESPACE)
        private MetaTag metaTag = new MetaTag();

        @XmlElement(name = "title", namespace = "http://purl.org/dc/elements/1.1/")
        private StringList titles = new StringList();

        @XmlElement(name = "creator", namespace = "http://purl.org/dc/elements/1.1/")
        private StringList creators = new StringList();

        @XmlElement(name = "created", namespace = "http://purl.org/dc/terms/")
        private DateStringList created = new DateStringList();
```

Figuur 4.2.7: simple metadata Java Annotated Class

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EasyMetadata xmlns:emd="http://easy.dans.knaw.nl/dms/easymetadata"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <emd:meta>
        <shortname>dc</shortname>
        <breed>simple</breed>
        <description>a simple metadata format based on Dublin Core describing datasets</description>
        <containerClassname>nl.knaw.dans.dms.qdc.simple.core.SimpleDCContainer</containerClassname>
    </emd:meta>
    <dc:creator>
        <value>res2</value>
    </dc:creator>
    <dc:title>
        <value>testing something part two</value>
    </dc:title>
    <dc:description>
        <value>testing testing part two</value>
    </dc:description>
    <dcterms:created>
        <value dcterms:W3CDTF="true" format="yyyy">
            <stringValue>2007</stringValue>
            <date>2007-01-01T00:00:00+01:00</date>
        </value>
    </dcterms:created>
```

Figuur 4.2.8: simple metadata(data.xml)

Hierboven is te zien hoe de elementen dc:creator, dc:title, dc:description en dcterms:created van het XML document gedefinieerd worden in de Java Annotated Class met het annotation *@XMLElement*. Voor een volledig overzicht van de klassen wordt u verwezen naar de bijlage 4.

Naast de Java Annotated Classes wordt de JAXBContext class gebruikt om de JAXB API te benaderen. Er moet een nieuwe instantie van deze klasse gemaakt worden via de methode *newInstance(contextPath)* en als argument kunnen er verschillende contextPaths meegegeven worden. Deze zijn de paths naar de eerder besproken Java Annotated classes. Na een nieuwe instantie gemaakt te hebben worden de methodes *createUnMarshaller* en *unMarshall(de xml file)* achtereenvolgens aangeroepen om de XML Document in te kunnen lezen. Hierna wordt de opgehaalde informatie naar de database gestuurd.

5.Software kwaliteit en testen

Om er zeker van te zijn dat onze uitgeschreven methoden naar behoren werkten hebben we onze code grondig getest. Deels door tijdens het schrijven van de code elke methode afzonderlijk met testdata te testen en deels door het maken van unit-tests. Een bekende unit test is die van JUnit wat eigenlijk niet meer dan een simpel framework is om herhaalbare tests te maken.

5.1 Testplan

We zullen alle code testen met behulp van unit-testing. Voor de code geschreven in Java gebruiken we JUnit. Omdat we de code zo generiek mogelijk hebben gemaakt gebruiken we in veel klassen dezelfde methodes . De veelgebruikte methodes gaan we dan ook testen met JUnit. Met JUnit heb je binnen een klasse die dan de package TestCase extend de methodes setUp(); en tearDown();. In setUp() declareer je alles wat je wil gaan gebruiken en met tearDown breek je het weer af().

```
protected void setUp() throws Exception {
                    Class.forName("com.mysql.jdbc.Driver").newInstance();
    conn = DriverManager.getConnection(STR_CONNECT);

    if(conn != null) {
        // insert the test user in the test db
```

```
            try {
                        stmnt = conn.createStatement();
                        stmnt.execute(INSERT9);
                        stmnt.execute(INSERT5);
            } catch (Exception me) {
                        fail(me.toString());
            }

            ps = conn.prepareStatement(LIST + depositId + "' ;");
            rs = ps.executeQuery();
            ps = conn.prepareStatement(LIST + downloadId + "' ;");
            rs2 = ps.executeQuery();

            createTestData();


      } else{
            fail("couldn't connect to the database");
      }
            super.setUp();
}
```

v.b. van de methode setUp() in klasse statisticsTest.

```
/* (non-Javadoc)
             * @see junit.framework.TestCase#tearDown()
             */
            protected void tearDown() throws Exception {
                        if(conn!=null) {
                                    //remove test user from db
                                    ps = conn.prepareStatement(REMOVE9);
                                    ps.execute();
                                    ps = conn.prepareStatement(REMOVE5);
                                    ps.execute();
                                    conn.close();
                        }
                        super.tearDown();
            }
```

v.b. van de methode teardown() in klasse statisticsTest.

Met de assert methodes van JUnit kan je nagaan als de data die je verwacht gelijk is aan de data die je daadwerkelijk verkregen hebt.

```
/**
             * Test method for {@link nl.knaw.dans.easy.usageStatistics.dao.StatisticsDAO#createStatisticsList()}.
             */
            public void testCreateStatisticsList() {
                        StatisticsDAO ls = new StatisticsDAO();
                        try {
                                    List<IStatistic> actualList = ls.createStatisticsList(rs);
                                    List<IStatistic> actualList2 = ls.createStatisticsList(rs2);
                         assertEquals(expStatisticList.get(0).getStatisticEvent(),
                         actualList.get(0).getStatisticEvent());
            assertEquals(expStatisticList.get(0).getIpAdres(),
            actualList.get(0).getIpAdres());
            assertEquals(expStatisticList.get(0).getUserName(),
            actualList.get(0).getUserName());
            assertFalse("",expStatisticList.get(0).getStatisticEvent() ==
            actualList2.get(0).getStatisticEvent());
            assertTrue(expStatisticList.size() == actualList.size());

                        } catch (SQLException e) {
```

```
                fail(e.toString());
            }
        }
```

v.b. van de methodes assert in klasse StatisticstTest

5.2 Documentatierichtlijnen

Als commentaar hebben wij voor de get en setmethodes javaDoc gebruikt en voor de specifieke methodes zelf het commentaar aangepast omdat javaDoc niet duidelijk genoeg was.

We hanteren de volgende richtlijnen:
- Iedere klasse krijgt een beschrijving
- Voor iedere methode worden de parameters beschreven.
- Het resultaat van een methode wordt beschreven
- De excepties en foutmeldingen die te voorschijn kunnen komen worden gelogd en als informatie getoond in de java console van eclipse.
- In complexe code wordt tussen de regels commentaar gezet met uitleg.

6. Resultaten

In dit hoofdstuk willen we een overzicht geven van de ontwikkelde webapplicatie en nagaan welke requirements uiteindelijk geimplementerd zijn.Als eerste laten we Easy Usage Statistics zien en daarna zullen we ingaan op de requirements.

6.1 Easy Usage Statistics

Het eerste wat opvalt als de klanten inloggen in hun bestaande systeem is dat er een knopje easy usage statistics in hun lijstje staat(Fig 6.1.1).

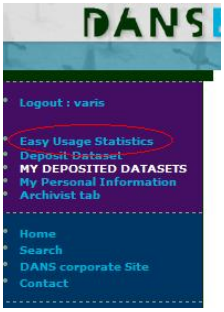


Fig 6.1.1

Door hierop te klikken opent de applicatie EUS, en wordt al gelijk de algemene informatie getoond(Fig 6.1.2).

Fig 6.1.2 eerste gedeelte home pagina

Zoals u hierboven zit heeft onze applicatie net als in EASY aan de linkerkant een menu waaruit gekozen kan worden. Daarnaast is er nog een dateselector bovenin, waarmee alle informatie op de betreffende pagina na het klikken van de knop show ververst wordt naar de geselecteerde datum. Als u wat meer naar beneden schuift binnen de contentPane(zie fig 2.5.3.3) is er nog informatie te zien over de downloads(fig 6.1.3).

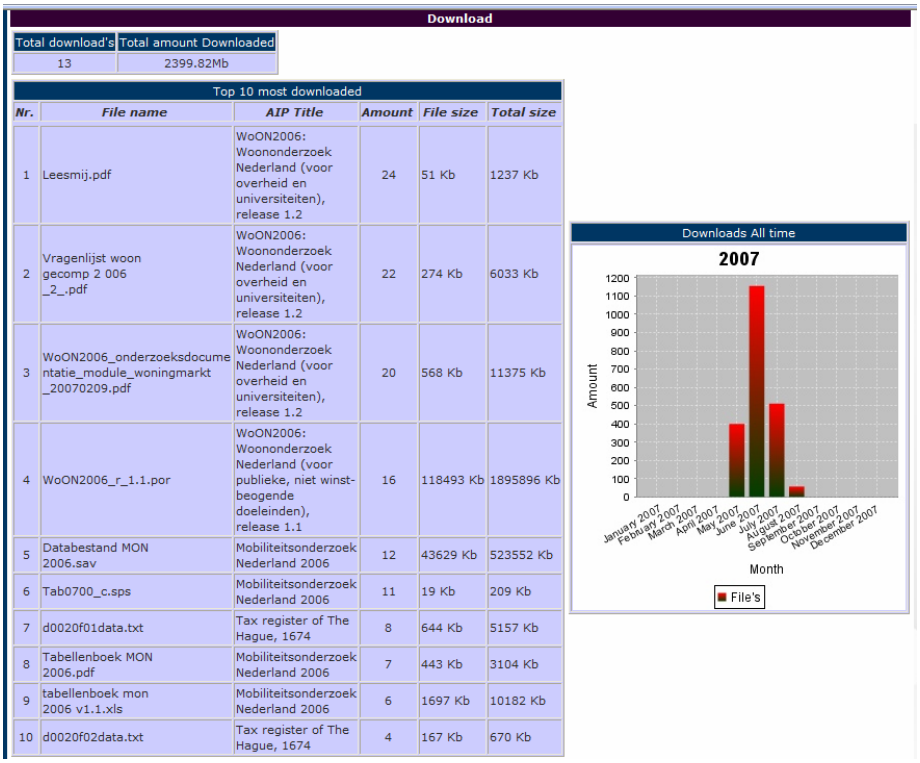
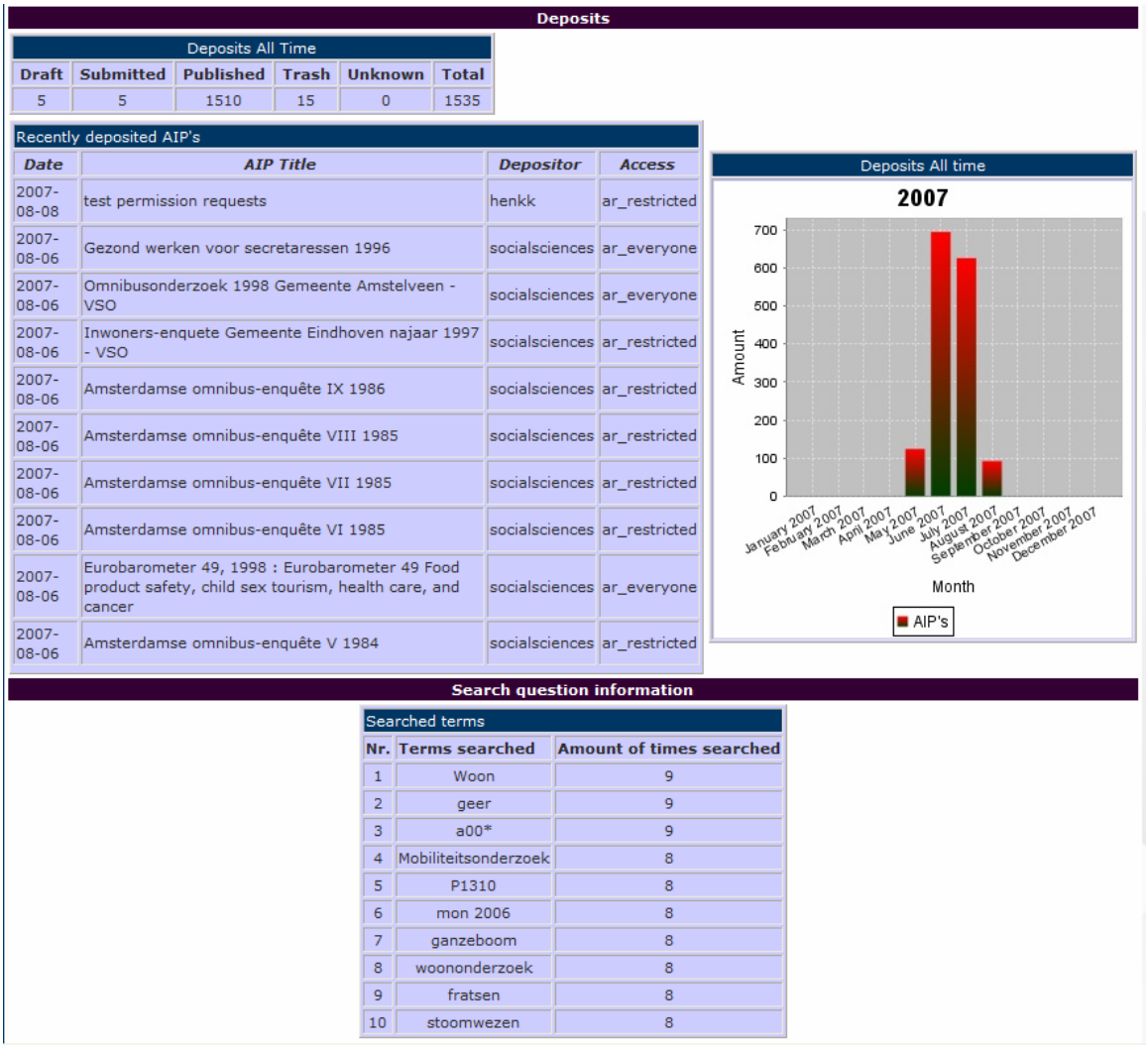


Fig 6.1.3 tweede gedeelte home pagina

Verder is er op de homepagina ook informatie over uploads en zoek-termen te zien(Fig 6.1.4).



Deposits

Deposits All Time

Draft	Submitted	Published	Trash	Unknown	Total
5	5	1510	15	0	1535

Recently deposited AIP's

Date	AIP Title	Depositor	Access
2007-08-08	test permission requests	henkk	ar_restricted
2007-08-06	Gezond werken voor secretaressen 1996	socialsciences	ar_everyone
2007-08-06	Omnibusonderzoek 1998 Gemeente Amstelveen - VSO	socialsciences	ar_everyone
2007-08-06	Inwoners-enquete Gemeente Eindhoven najaar 1997 - VSO	socialsciences	ar_restricted
2007-08-06	Amsterdamse omnibus-enquête IX 1986	socialsciences	ar_restricted
2007-08-06	Amsterdamse omnibus-enquête VIII 1985	socialsciences	ar_restricted
2007-08-06	Amsterdamse omnibus-enquête VII 1985	socialsciences	ar_restricted
2007-08-06	Amsterdamse omnibus-enquête VI 1985	socialsciences	ar_restricted
2007-08-06	Eurobarometer 49, 1998 : Eurobarometer 49 Food product safety, child sex tourism, health care, and cancer	socialsciences	ar_everyone
2007-08-06	Amsterdamse omnibus-enquête V 1984	socialsciences	ar_restricted

Deposits All time

Search question information

Searched terms

Nr.	Terms searched	Amount of times searched
1	Woon	9
2	geer	9
3	a00*	9
4	Mobiliteitsonderzoek	8
5	P1310	8
6	mon 2006	8
7	ganzeboom	8
8	woononderzoek	8
9	fratsen	8
10	stoomwezen	8

Fig 6.1.4 derde gedeelte home pagina

Als er op de knop userstatistics geklikt wordt ziet u de volgende pagina waarop alle informatie over de easy gebruikers te zien is(fig 6.1.5).

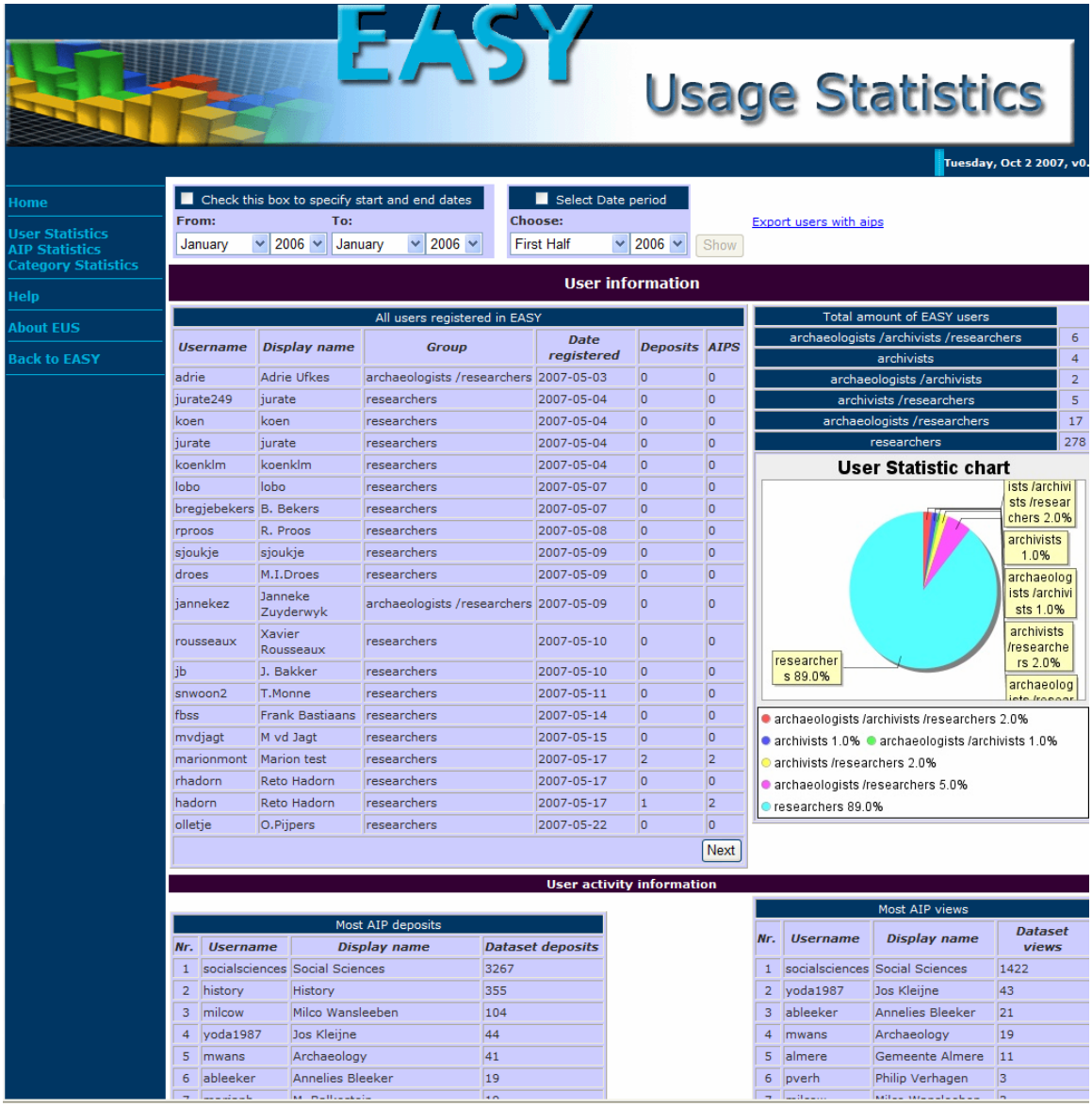


Fig 6.1.5 user statistics pagina

En bij de knoppen aipstatistics en category statistics is er informatie te vinden van de aips(fig 6.1.6 en 6.1.7).

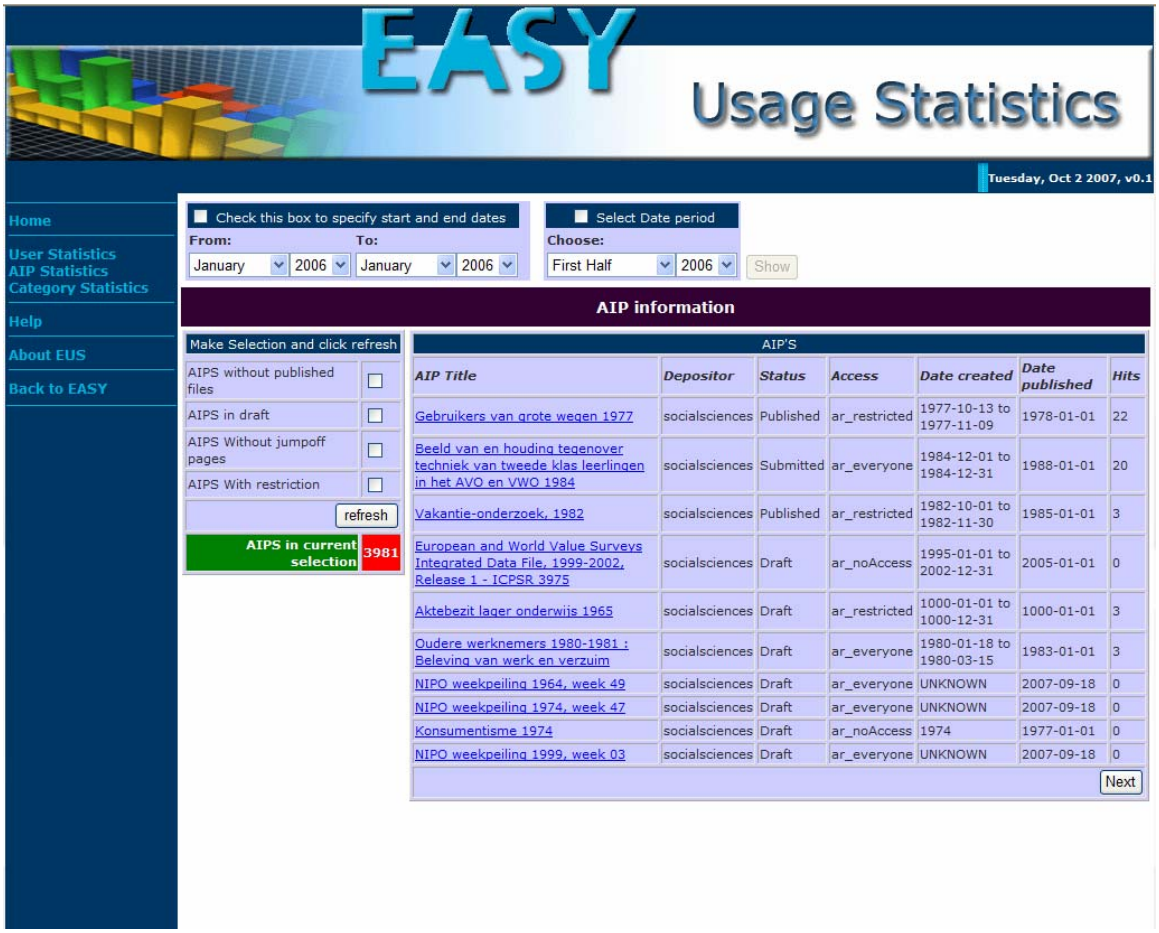


Fig 6.1.6 AIP statistics pagina

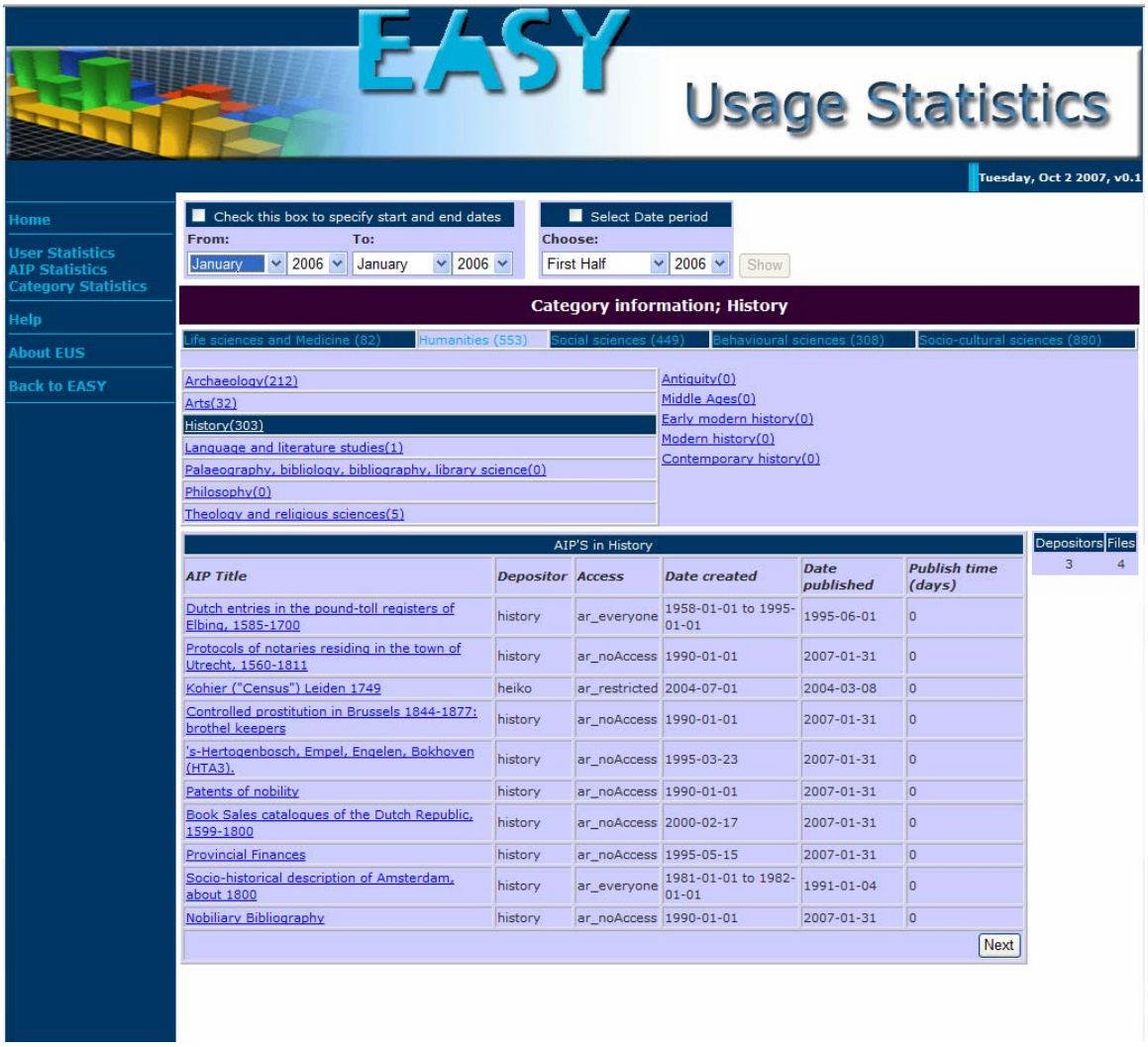


Fig 6.1.7 category statistics pagina

6.2 Requirements gehaald/niet gehaald

Na de requirements te hebben gesorteerd m.b.v. het MoSCoW diagram hebben we al van te voren gezegd welke requirements we wel zouden implementeren en welke niet. In het uiteindelijke systeem hebben wij alle must-haves, alle shoud haves en bijna alle could haves geimplementeerd.

Aan het einde van onze 10 weken hebben wij in totaal vier werkende contentpanes opgeleverd met daarop gegevens over het gebruik van EASY.

Home:
Dit is de startpagina met daarop de volgende gegevens:
EASY User Totals : Dit is een tabel met algemene informatie over de gebruikers van EASY. Er zijn 6 kolommen zichtbaar t.w. de Visits, registrations , Downloaders, Depositors, Logins en als laatste Logouts. Verder is er een tabel met de laatste 10 bekeken datasets en een tabel met het totaal aantal bekeken datasets.

Download:

Hier zijn er in totaal twee tabellen te zien en een histogram. De eerste tabel heeft twee kolommen, total downloads in aantallen en total amount downloaded in megabytes. De tweede tabel geeft de Top 10 meest gedownloade files. Hierin vindt men informatie als file name, aip title, amount file size en total amount. Als laatste het histogram, hier kan men het totale aantal gedownloade files zien per maand.

Deposits:
Hier vindt men de deposit informatie. Er zijn hier ook twee tabellen en een histogram te zien. Het eerste bevat alle deposits tot nu toe onderverdeeld in AIP statussen. De tweede bevat de recent gedeponeerde aips met als kolommen de datum aip titel en de depositor. Het histogram geeft een overzicht van het totale aantal deposits per maand weer.

Search question information:
Hier kan men zien welke zoektermen de laatste tijd gebruikt zijn en ook staat er een top 10.

User Infomation:
De gebruikers gegevens van EASY.
Twee tabellen en een Pie Chart.De eerste tabel bevat alle geregistreerde gebruikers van EASY met hun username, displayname group en date registered. Daarnaast is er een tabel met de totale user gegevens onderverdeeld in groepen. Dit overzicht is ook te zien in de Pie Chart met percentages erbij.

AIP Information:.
De gegevens m.b.t. de aips in EASY.
Hier zijn verschillende requirements geïmplementeerd. Aan de linkerkant heb je een box met opties die je kan aanvinken. Door op de knop refresh te drukken wordt de lijst van alle aips ververst met de gefilterde opties.

Category information:
De gegevens m.b.t. de categorieën in EASY.
Tabellen die uit te breiden zijn per categorie met daarin desbetreffende aips. Naast de kopjes van de categorieën vind je ook gelijk het aantal aips in deze categorie.
Als je een subcategorie aanklikt verschijnt daaronder een lijstje met alle aips in die categorie. Verder kan je dan nog op de naam van de aip klikken en opent deze in een nieuw scherm.

Afgemaakt:
Op de volgende twee pagina's zijn de requirements genummerd op volgorde van prioriteit. De volgende nummers zijn afgerond;
Gebruikers: 1.1 t/m 1.5. Punt 1.4 niet onderverdeeld in wel/niet gedownload.
Dataset: 2.1 t/m 2.9 en 2.14. Punt 2.6, verzenden waarschuwingsemail niet geïmplementeerd. Punt 2.12, er wordt momenteel in EASY een email verzonden naar de depositors van de AIPS waarvan de restrictie binnen twee maanden verloopt, de verzenddatum van de email wordt bijgehouden. De waarschuwingsemail kan dus wel worden opgevraagd. Punt 2.13, datasets die in draft zitten kunnen worden opgevraagd maar hoe lang ze in de draft toestand gebleven zijn kan niet opgevraagd worden. De begindatum wordt in easy niet bijgehouden.
Category: 3.1 t/m 3.3
Export functies: 4.1, exporteren naar excel is voor een gedeelte klaar.
Top/Laatste 10: 5.1 t/m 5.6
Alegemeen: 6.1

Requirements

+++ Must Have
++ Should Have
+ Could Have
- Won't Have

1. *Gebruikers:*
   - *1.1.* +++ Het aantal gebruikers moet kunnen worden opgevraagd.(bv archeoloog of researcher)
   - *1.2.* +++ Welke gebruikers welke soort dataset(bv archeologische dataset) gedeponeerd hebben en aantal per depositor.
   - *1.3.* +++ Totaal aantal depositors per jaar/kwartaal/maand.
   - *1.4.* +++ Totaal aantal bezoekers van EASY onderverdeeld in wel/niet gedownload per maand/kwartaal/jaar.
   - *1.5.* ++ Aantal geregistreerde/niet geregistreerde gebruikers(nieuw)per maand/kwartaal/jaar/totaal.
   - *1.6.* ++ Permission requests die nog open staan per gebruikers type.
   - *1.7.* **- Per gebruiker het aantal datasets in de drafts en sinds wanneer, de date created wordt niet bijgehouden(advies; bijhouden als het echt nodig is).**

2. *Dataset(AIP's):*
   - *2.1.* +++ Aantal gedownloade files per AIP per category.(zonder metadata.pdf's).
   - *2.2.* +++ Welke aips hebben nog geen gepubliceerde files.
   - *2.3.* +++ (totaal)Aantal aanwezige AIP's (tot nu toe) per gebruiker/categorie.
   - *2.4.* +++ aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.5.* +++ Totaal aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.6.* +++ De verstreken tijd tussen het submitten van een AIP en het publishen van een AIP.(Als het binnen 2(5) dagen niet gebeurt waarschuwing verzenden.)
   - *2.7.* +++ Aantal bekeken datasets jaar/kwartaal/maand.(ook aantal hits per dataset)
   - *2.8.* ++ Welke hebben nog geen jumpoff pages.(evt een extra kolom).
   - *2.9.* ++ Gebuikersrechten en dateavailable per aip.
   - *2.10.* ++ De "best permission" van resticted datasets weergeven.
   - *2.11.* ++ Aantal datasets die terug zijn naar Draft(niet gepublished zijn na workflow).
   - *2.12.* ++ Verlopen restricties van AIP's, verzonden waarschuwings mail.
   - *2.13.* ++ Welke datasets zitten nog in de drafts en -hoe lang.
   - *2.14.* + Welke bestanden in een AIP hebben geen extra metadata.
   - *2.15.* + Waar vandaan wordter naar welkeAIP's/easy gelinkt???
   - *2.16.* + Aantal ingevulde metadata velden per aip.
   - *2.17.* + veranderingen in toegangscategorieen.
   - *2.18.* - Wat is de responsetijd per AIP.

3. *Categorie:*
   - *3.1.* +++ Aantal depositors per jaar/kwartaal/maand per categorie
   - *3.2.* ++ Welke AIP's zijn per categorie gepublished op volgorde van datum.
   - *3.3.* ++ Files per categorie:
     - *3.3.1.* +++ Welke bestanden zijn het meest gedownload per categorie.
     - *3.3.2.* +++ (Totaal)Aantal gedownloade bestanden per jaar/kwartaal/maand per categorie.
     - *3.3.3.* ++ (totaal)Aantal gedeponeerde bestanden(aantal files) per jaar/kwartaal/maand.
     - *3.3.4.* + Welke bestandstypen komen per categorie voor.
     - *3.3.5.* + Aantal files in de AIP's verdeeld per category. En ook aantal MB's en

+ bestandsstype.(ext of bestandsherkenning).

4. *Export functies:*
   - *4.1.* +++ Naar excel
   - *4.2.* **- Xml export van de project (dublin core) metadata waarin alle AIP's van een vooraf bepaalde groep zitten.**

5. Top/Laatste 10 per /allertijden/jaar/kwartaal/maand:
   - *5.1.* +++ Gebruikers die het meeste geupload hebben .
   - *5.2.* +++ Gebruikers die het meeste gedownload hebben.
   - *5.3.* **+++ Meest aangevraagde AIP's en datafiles.**
   - *5.4.* +++ Laatste 10 gedeponeerde bestanden + acces categorie(als extra kolom).
   - *5.5.* +++ Laatste 10 bekeken AIP's.
   - *5.6.* +++ Actiefste gebruiker(down+upload).
6. *Algemeen:*
   - *6.1.* +++ Zoektermen per dag/ maand en meest gezochte termen.
   - *6.2.* – browse acties per dag/maand.
   - *6.3.* - hoe vaak mislukt het deponeren, gaan mensen verder nadat het een keer mislukt is?
   - *6.4.* - hoe lang zijn mensen bezig met het deponeren?(zie punt 1.7)

7. Conclusie

De laatste 10 weken hebben we heel hard gewerkt aan dit project en toch kwamen we uiteindelijk tijd te kort omdat niet al onze plannen zijn verlopen zoals wij gedacht hadden. We hebben geleerd over de voor en nadelen van de xp ontwikkel methodiek en we hebben veel geleerd over het indelen van je tijd en het maken van afspraken. Ook hebben we onze Java kennis uitgebreidt met nieuwe technieken zoals DAO en MVC(zie hoofdstuk 4), en hebben we meer geleerd van XML en XSLT. Verder is deze stage ook nog handig geweest voor het leren onderhandelen met de opdrachtgever en andere belanghebbenden.

Het ontwerpen van de statistics web-applicatie bleek een interesante opdracht te zijn waar we eigenlijk nog tijd voor tekort kwamen. We hebben uiteindelijk bijna alle vooraf afgesproken requirements geimplementeerd maar zijn niet toegekomen aan het implementeren van de applicatie op de server en het testen daarvan. Met behulp van de database tasks(Zie paragraaf 4.2.5) is het momenteel wel mogelijk de applicatie te gebruiken om actuele informatie te verkrijgen, maar het oorspronkelijke plan was het implementeren op de server met het oog op snelheid. Tijdens ons laatste voortgangs gesprek hebben we samen met de begeleider en opdrachtgever afgesproken dat gezien het gebrek aan tijd de applicatie voorlopig niet op de server geimplementeerd wordt maar wordt gebruikt met de databasetasks. Dit zal zo blijven tot EASY op het nieuwe systeem fedora(zie hoofdstuk 8) draait.

Ook op database-gebied hebben we veel van onze kennis kunnen toepassen omdat er naast de bestaande statisticsdatabase nog een extra database nodig was om extra informatie op te slaan. Dit is de reden waarom we uiteindelijk onze eigendatabase(zie paragraaf 3.2.2) hebben ontworpen en geimplementeerd. Tijdens ons project is er binnen EASY zelf om snelheids redenen ook een database in gebruik genomen, en omdat deze database veel gelijkenissen vertoonde met die van ons hebben wij besloten onze ontbrekende tabellen ook daar te implementeren. Uiteindelijk zal dus de easystatistics

database gebruikt worden in combinatie met die van EASY zelf. In het huidige systeem gebruiken we echter nog onze eigen variant.

Door het volgen van de XP ontwikkelmethodiek heeft de opdrachtgever(Milco Wansleeben) iedere week de geplande requirements moeten goedkeuren. Hierdoor kunnen we concluderen dat de functionaliteit die voor de opdrachtgever het belangrijkst was geïmplementeerd is.

8. Aanbevelingen

Voordat we het gaan hebben over de aanbevelingen en de uitbreidingen die misschien kunnen plaatsvinden in de toekomst willen we het eerst hebben over een ontwikkeling bij EASY(EASY op Fedora) waar men al mee begonnen is.

8.1 Fedora

Omdat men bij DANS streeft naar perfectie wordt er voortdurend gewerkt aan het verbeteren van EASY. Een van de nieuwste ontwikkelingen is het plan om de gehele architectuur van EASY om te gooien en te laten werken op Fedora. Het Project genaamd Easy On Fedora (EOF) heeft verschillende punten als doel:

- Added value by design.
- Duurzame software en software ontwikkeling.
- EASY gebruikers gelukkig maken met o.a. snelheid, betrouwbaarheid, schaalbaarheid en handigheid van werken.
- Etc.

De reden waarom deze ontwikkelingen hier kort behandeld worden is omdat men van plan is om de EUS plugin ook mee te nemen naar dit nieuwe systeem(zie figuur 8.1.1).

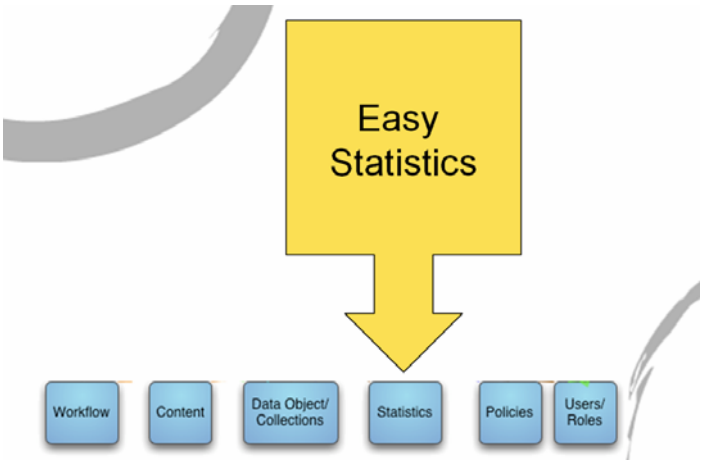


Fig 8.1.1 Easy On Fedora(EOF) met EUS

In feite zijn er dus alleen 2 databases(die d.m.v. logs in de nieuwe code bijgevuld worden) nodig om EUS in EOF te gebruiken. Meer informatie hierover vind u in paragraaf 8.2.

8.2 Toekomstplannen en aanbevelingen

Om nu de applicatie te gebruiken is het handig om te weten waar allemaal rekening mee moet worden gehouden.

Als eerste zijn er verschillende libraries, *.jar files(Java ARchive) nodig voor EUS:

- Jfreechart-1.0.6.jar en jcommon-1.0.10.jar, nodig voor het maken van de grafieken.
- Mysql-connector-java-5.0.5-bin.jar, nodig voor de communicatie tussen de applicatie en de database.
- Jxl.jar , nodig voor het exporteren van tabellen naar Excel formaat.

Als eerste moet je een database aanmaken volgens het model dat beschreven is in paragraaf 3.2.2. Hiervoor moet de database Job beschreven in paragraaf 4.2.5 van hoofdstuk 4 uitgevoerd worden als Java Applicatie. Door deze Job uit te voeren worden alle gegevens die voor EASY nodig zijn opgeslagen in de 2 databases. Wanneer dit gedaan is kan je EASY starten en de link volgen naar Usage Statistics, waar dan alle informatie beschikbaar zal zijn.

Tijdens het ontwikkelen van EUS is er bij EASY ook veel veranderd. Een van de grootste problemen was dat EASY traag reageerde bij het ophalen van de AIP's waardoor men naar andere alternatieven ging zoeken om de performance te verbeteren. Henk van de Berg, één van de ontwikkelaars voor EASY, kwam op het idee om een database in te schakelen zodat het ophalen van AIP's wat sneller kan gaan. Ondertussen is de database in werking gesteld en de ontwikkeling daarvan liep parallel met het ontwikkelen van de database die bij EUS gebruikt wordt.
Als later EUS gereleased wordt dan zou het beter zijn om in plaats van twee databases, die grotendeels hetzelfde zijn, één database te gebruiken omdat je zo redundantie vermijdt. Hiervoor moet de EASY database uitgebreid worden met de extra tabellen en kolommen van de additional_information database van EUS.
Verder moet in de code een nieuwe concrete DAOFactory en de onderliggende DAO's geïmplementeerd worden. Voor meer informatie over DAO's en DAOFactory wordt u verwezen naar paragraaf 3.1.2 en 4.2 van hoofdstuk 3 en 4.

9. Literatuurlijst

[1] David J. Barnes, *Object-Oriented programming with Java: An Introduction*(2000)

[2] Alan Shalloway, James R. Trott, *Design Patterns Explained: A new perspective on Object-Oriented Design*(2002)

[3] Timothy C. Lethbridge, Robert Laganiere, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*(2004)

[4] Rien Elling, Bas Andeweg, Jaap de Jong, Christine Swankhuisen, *Rapportagetechniek*(2000)

[5] David Hunter, *Beginning XML*(2000)

[6] James NewKirk, Robert C. Martin, *Extreme Programming in practice*(2001)

[7] W3 Schools, *http://w3schools.com(2007)*

[8] SUN Developer Network; Core J2EE Patterns - Data Access Object
http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html (okt-2007)

[9] SUN Developer Network; Article Java Architecture for XML Binding (JAXB)
http://java.sun.com/developer/technicalArticles/WebServices/jaxb/index.html(okt-2007)

[10] 2005 JavaOne Conference | Session 3636, Sekhar Vajjhala;
http://gceclub.sun.com.cn/java_one_online/2005/TS-3636/ts-3636.pdf (okt-2007)

[11] Handleiding eXtreem Programming voor DANS versie 2. (2007)

[12]Elmasri & Navathe, *Fundamentals of DATABASE SYSTEMS*(2007)

[13] Eclipse Foundation, http://www.eclipse.org/org/(2007)

[14] Data Archiving and Networked Services, http://www.dans.knaw.nl/nl/(2007)

[15] JFreeChart, http://www.jfree.org/index.html(2007)

[16] Model View Controller, http://msdn2.microsoft.com/en-us/library/ms978748.aspx(2007)

Bijlagen

Bijlage 1: RAD

Requirements Analysis Document

R.D. Elisabeth
R.E. Matroos

1. Inleiding

DANS is een gezamenlijk initiatief van de Koninklijke Nederlandse Akademie van Wetenschappen (KNAW) en de Nederlandse organisatie voor Wetenschappelijk Onderzoek (NWO). DANS is opgericht om op nationaal niveau de toegankelijkheid van gegevensbestanden voor onderzoekers in de maatschappij- en gedragswetenschappen en in de geesteswetenschappen te verbeteren en voor de lange termijn te garanderen. DANS beoogt (interdisciplinaire) samenwerking met en tussen onderzoekers te bevorderen.
De sectie O&A initieert en begeleidt projecten op het gebied het gebruik van ICT voor de lange-termijn toegang van wetenschappelijk onderzoeksmateriaal. Onlangs is door DANS het Electronic Archiving System (EASY, http://easy.dans.knaw.nl/) in gebruik genomen, waar onderzoekers in de maatschappij- en gedragswetenschappen en in de geesteswetenschappen onderzoeksdata kunnen deponeren en downloaden.

2. De huidige situatie.

Sinds EASY in augustus 2005 in werking is gesteld is het nodig dat bepaalde informatie met betrekking tot het gebruik van dit systeem opgevraagd kan worden.
Informatie zoals welke/hoeveel datasets gedeponeerd worden per dag, het aantal bezoekers per dag/maand/jaar, het aantal gedownloade bestanden, het aanwezige
aantal datasets, etc.
Om in EASY een beeld te krijgen van deze informatie wordt er sinds kort een MySQL database bijgehouden en uit de informatie die in deze database wordt opgeslagen valt af te leiden hoe gebruikers met de EASY website omgaan. Er is echter geen software voorhanden om de rauwe data op een menselijk interpreteerbare manier te presenteren. Als er nu wat van deze gegevens moet worden opgevraagt dient men zelf in de database te gaan zoeken en aangezien er de laatste tijd steeds meer vraag is naar deze informatie is er behoefte aan een applicatie dit makkelijk kan weergeven en exporteren naar gewenste formaten.

3. Het te bouwen systeem

3.1 Overzicht

Er moet een systeem gebouwd worden waarbij verscheidene personen bij DANS gemakkelijk informatie kunnen opvragen over het gebruik van EASY.

3.2 Functionele eisen

We onderscheiden twee soorten gebruikers van het systeem:

- Geregistreerde gebruiker
- Anonieme gebruiker

Geregistreerde gebruiker

Onder geregistreerde gebruiker verstaan we de gebruikers die al een account hebben voor EASY. Het is de bedoeling dat iedereen die op EASY kan inloggen ook automatisch toegang heeft tot EUS.

1.1 Een geregistreerde gebruiker kan alle management informatie bekijken en kan evt de gegevens exporteren naar gewenste formaten.
De gebruiker kan bijvoorbeeld een specifiek de top 10 meest bekeken datasets in een maand opvragen;

- 1.1.1 Top <aantal> datasets
- 1.1.2 Per <discipline>, waarbij <discipline> gekozen kan worden uit een lijst.
- 1.1.3 Soort dataset
- 1.1.4 In de maand <maand>

Anonieme gebruiker

Een anonieme gebruiker is een gebruiker die geen account heeft en toch de website bezoekt.

2.1 Kan de basis gegevens op de website zien, zoals de top 10 gedownloade/uploade datasets en de recentste toevoegingen inzien.

Hieronder treft u de eisen aan die zijn samengesteld na gesprekken met de opdrachtgever en klanten. Deze requirements zijn opgesteld in MoSCoW formaat.

+++ Must Have
++ Should Have
+ Could Have
- Won't Have

1. *Gebruikers:*
   - *1.1.* +++ Het aantal gebruikers moet kunnen worden opgevraagd.(bv archeoloog of researcher)
   - *1.2.* +++ Welke gebruikers welke soort dataset(bv archeologische dataset) gedeponeerd hebben en aantal per depositor.
   - *1.3.* +++ Totaal aantal depositors per jaar/kwartaal/maand.
   - *1.4.* +++ Totaal aantal bezoekers van EASY onderverdeeld in wel/niet gedownload per maand/kwartaal/jaar.
   - *1.5.* ++ Aantal geregistreerde/niet geregistreerde gebruikers(nieuw)per maand/kwartaal/jaar/totaal.
   - *1.6.* ++ Permission requests die nog open staan per gebruikers type.
   - *1.7.* **- Per gebruiker het aantal datasets in de drafts en sinds wanneer, de date created wordt niet bijgehouden(advies; bijhouden als het echt nodig is).**

2. *Dataset(AIP's):*
   - *2.1.* +++ Aantal gedownloade files per AIP per category.(zonder metadata.pdf's).
   - *2.2.* +++ Welke aips hebben nog geen gepubliceerde files.
   - *2.3.* +++ (totaal)Aantal aanwezige AIP's (tot nu toe) per gebruiker/categorie.
   - *2.4.* +++ aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.5.* +++ Totaal aantal gedeponeerde/gepubliceerde AIP's per jaar/kwartaal/maand.
   - *2.6.* +++ De verstreken tijd tussen het submitten van een AIP en het publishen van een AIP.(Als het binnen 2(5) dagen niet gebeurt waarschuwing verzenden.)

2.7. +++ Aantal bekeken datasets jaar/kwartaal/maand.(ook aantal hits per dataset)
2.8. ++ Welke hebben nog geen jumpoff pages.(evt een extra kolom).
2.9. ++ Gebuikersrechten en dateavailable per aip.
2.10. ++ De "best permission" van resticted datasets weergeven.
2.11. ++ Aantal datasets die terug zijn naar Draft(niet gepublished zijn na workflow).
2.12. ++ Verlopen restricties van AIP's, verzonden waarschuwings mail.
2.13. ++ Welke datasets zitten nog in de drafts en -hoe lang.
2.14. + Welke bestanden in een AIP hebben geen extra metadata.
2.15. + Waar vandaan wordter naar welkeAIP's/easy gelinkt???
2.16. + Aantal ingevulde metadata velden per aip.
2.17. + veranderingen in toegangscategorieen.
2.18. - Wat is de responsetijd per AIP.

3. *Categorie:*
   3.1. +++ Aantal depositors per jaar/kwartaal/maand per categorie
   3.2. ++ Welke AIP's zijn per categorie gepublished op volgorde van datum.
   3.3. ++ Files per categorie:
      3.3.1. +++ Welke bestanden zijn het meest gedownload per categorie.
      3.3.2. +++ (Totaal)Aantal gedownloade bestanden per jaar/kwartaal/maand per categorie.
      3.3.3. ++ (totaal)Aantal gedeponeerde bestanden(aantal files) per jaar/kwartaal/maand.
      3.3.4. + Welke bestandstypen komen per categorie voor.
      3.3.5. + Aantal files in de AIP's verdeeld per category. En ook aantal MB's en
      + bestandsstype.(ext of bestandsherkenning).

4. *Export functies:*
   4.1. +++ Naar excel
   4.2. **- Xml export van de project (dublin core) metadata waarin alle AIP's van een vooraf bepaalde groep zitten.**

5. Top/Laatste 10 per /allertijden/jaar/kwartaal/maand:
   5.1. +++ Gebruikers die het meeste geupload hebben .
   5.2. +++ Gebruikers die het meeste gedownload hebben.
   5.3. **+++ Meest aangevraagde AIP's en datafiles.**
   5.4. +++ Laatste 10 gedeponeerde bestanden + acces categorie(als extra kolom).
   5.5. +++ Laatste 10 bekeken AIP's.
   5.6. +++ Actiefste gebruiker(down+upload).
6. *Algemeen:*
   6.1. +++ Zoektermen per dag/ maand en meest gezochte termen.
   6.2. – browse acties per dag/maand.
   6.3. - hoe vaak mislukt het deponeren, gaan mensen verder nadat het een keer mislukt is?
   6.4. - hoe lang zijn mensen bezig met het deponeren?(zie punt 1.7)

3.3 Niet-functionele eisen

User Interface

De user-interface moet te gebruiken zijn door de opdrachtgever en andere klanten en er moet bij aanmelden een overzicht te zien zijn van de basis informatie (aantal downloads, aantal uploads etc.)

Documenten

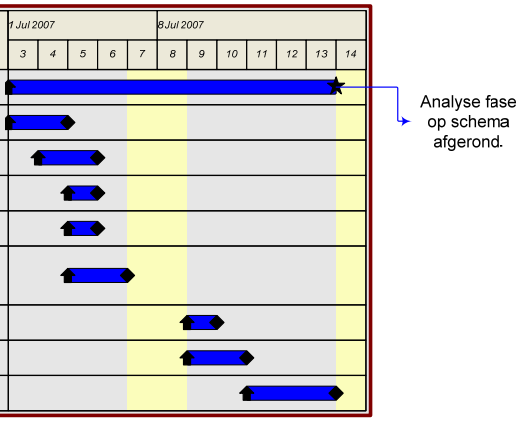
Er moet een handleiding gemaakt worden die moet helpen als de gebruiker niet weet hoe hij bepaalde informatie kan vinden.

Systeemaanpassingen

Het systeem moet zo in elkaar zitten dat het gemakkelijk implementeerbaar is op een nieuw systeem d.w.z. het moet zo onafhankelijk mogelijk gemaakt worden zodat het niet uitmaakt op wat voor manier EASY in elkaar zit.

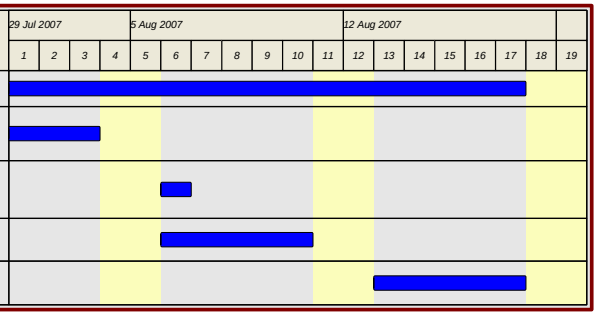
Bijlage 2: Gantt charts

ID	Task Name	Start	Finish	Duration
1	Analyse fase	7/3/2007	7/13/2007	1w 4d
2	Opdracht bekijken en onderzoek doen naar mogelijkheden	7/3/2007	7/4/2007	2d
3	Requirements-analyse, begin	7/4/2007	7/5/2007	2d
4	Requirements verder uitbreiden	7/5/2007	7/5/2007	1d
5	Voor-en nadelen webapplicatie vs standalone zoeken	7/5/2007	7/5/2007	1d
6	Documentatie lezen over WTP in eclipse en de andere software tools die we gaan gebruiken	7/5/2007	7/6/2007	2d
7	Planning game met de klant en begeleider	7/9/2007	7/9/2007	1d
8	Begin makekn aan het verslag en maken MoSCoW diagram	7/9/2007	7/10/2007	2d
9	Begin klassendiagram	7/11/2007	7/13/2007	3d



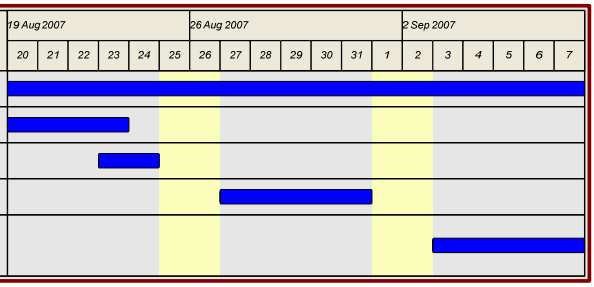
ID	Task Name	Start	Finish	Duration
1	Implentatie fase	7/16/2007	7/31/2007	12d
2	Eerste iteratie	7/16/2007	7/20/2007	5d
3	Planning game met rutger. MySQL database downloaden en installeren.	7/16/2007	7/16/2007	1d
4	Instellen van de database en brainstormen over het database model(nieuwe of aanpassen etc.)	7/17/2007	7/17/2007	1d
5	Beginnen aan de basis (framework) van onze applicatie.	7/18/2007	7/20/2007	3d
6	Tweede iteratie	7/23/2007	7/27/2007	5d
7	Applicatie design. Maken logo EASY Usage Statistics(EUS) in photoshop. Planning game met Rutger	7/23/2007	7/23/2007	1d
8	Onderzoek doen naar DAO(Data Acces Object) en verder programmeren	7/23/2007	7/25/2007	3d
9	Testdata aanmaken(in database) en klassen schrijven(contentPanes) om deze testdata te weergeven	7/25/2007	7/27/2007	3d
10	Testklassen schrijven met JUNIT voor de eerste klassen	7/30/2007	7/31/2007	2d

ID	Task Name	Start	Finish	Duration
1	Implementatie fase	8/1/2007	8/17/2007	13d
2	Werken aan het databasemodel en de java weergave hiervan voor de klasse(en het tabel) aipinformation.	8/1/2007	8/3/2007	3d
3	Kijken naar de mogelijkheden voor weergave van informatie m.b.v. grafieken etc.	8/6/2007	8/6/2007	1d
4	De requirements van het “users” gedeelte van het MoSCoW diagram	8/6/2007	8/10/2007	5d
5	De requirements van het “algemene” gedeelte in de applicatie.	8/13/2007	8/17/2007	5d

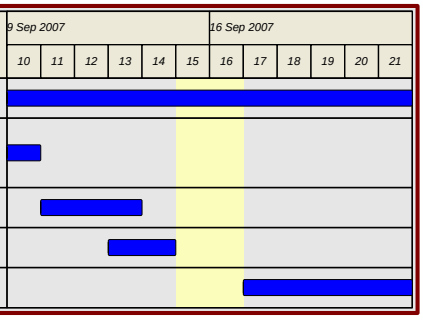


Hier wijkt de planning af van wat er echt gebeurt is omdat we gezien hebben dat je de requirements niet in vaste blokken kan programmeren omdat er steeds weer wat anders bij komt kijken wat nog makkelijker is om eerst mee te nemen.

ID	Task Name	Start	Finish	Duration
1	Implementatie fase	8/20/2007	9/7/2007	15d
2	Testklassen schrijven voor het afgemaakte deel	8/20/2007	8/23/2007	4d
3	Requirements van het MoSCoW diagram maken voor het AIP gedeelte	8/23/2007	8/24/2007	2d
4	Requirements maken voor het AIPgedeelte en voor categories	8/27/2007	8/31/2007	5d
5	Requirements voor de Categories en het schrijven van testklassen voor het tot nu toe gemaakte gedeelte	9/3/2007	9/7/2007	5d



ID	Task Name	Start	Finish	Duration
1	Verslaglegging en afmaken van kleine dingetjes	9/10/2007	9/21/2007	10d
2	Mergen van gegevens tijdens de afwezige periode(wel doorwerken aan requirements)	9/10/2007	9/10/2007	1d
3	Doornemen van gant charts en doorgaan met het eindverslag	9/11/2007	9/13/2007	3d
4	Resterende code programmeren en handleiding maken voor de applicatie	9/13/2007	9/14/2007	2d
5	Resterende code programmeren en Werken aan verslag	9/17/2007	9/21/2007	5d



Bijlage 3: Project log

Begeleider TU:

Ir. M Sepers

Begeleider DANS:

Ir. R Kramer

Groepsleden:

Ramses Elisabeth 1015052
Ralph Matroos 1193791

Dinsdag 3-07-07

De eerste werkdag. We hebben de opdracht doorgenomen en zijn begonnen met het maken van het requirements analyse verslag.

Woensdag 4-07-07

We zijn begonnen met de requirements door te nemen en hebben een afspraak gemaakt en gehad met Peter. De ideeen van Peter hebben we verwerkt in een verslag met de bedoeling deze uit te breiden naarmate er meer afspraken gemaakt worden.

Donderdag 5-07-07

Afspraak met onze begeleider(Rutger). We hebben de verwachtingen onderling besproken en we hebben het gehad over wat voor systeem het wordt(web applicatie of standalone applicatie).
Verder hebben wij nog een document gemaakt met de voor/nadelen bij het gebruik van een web vs. standalone applicatie.
Afspraak gemaakt met de klant(Milco).
We hebben gesproken met de makers van de huidige database en zijn alvast begonnen met het vergelijken van dingen die erin staan en die we nog nodig hebben.

Maandag 9-07-07

Planning-game gehad met de klant(Milco) en een Mediator(Henk B), hier hebben wij het gehad over de requirements en de vragen die we daar nog over hadden en hebben wij het over het te maken systeem gehad. Uit de requirements is er duidelijk geworden wat er nu al mogelijk is en wat pas na aanpassing van de database mogelijk is, en ook wat er minder belangrijk/niet mogelijk is. Hieruit gaan wij een MoSCoW diagram gemaakt die we dan gaan gebruiken bij de implementatie van het programma.(prioriteiten e.d.)

We hebben ook besloten het systeem een onderdeel te maken van het bestaande EASY(webapplicatie) dat alleen toegankelijk is voor mensen met een bepaalde rol.

Dinsdag 10-07-07

Easy Usage Statistics, dat is de naam die we hebben gevonden voor ons gedeelte van de statistieken. Na het installeren van de benodigde software hebben wij een branch gemaakt waarop wij gaan beginnen ons programma te schrijven.

Woensdag 11-07-07

Probleem: In de database staat niet alle informatie die wij nodig zullen hebben. Moeten wij onze eigen database maken? Of mogen wij de bestaande database aanpassen?

We hebben het een en ander nagevraagd en hebben dus besloten de aanpassingen gewoon aan te brengen in de bestaande database.(zonder hun systeem te wijzigen)

MySQL database gedownload en geïnstalleerd op beide pc's we hebben de bestaande statistics database nagemaakt en hebben een extra tabel User toegevoegd omdat we die nodig zullen hebben.

Vrijdag 13-07-07

We hebben m.b.t. statistieken in de database besloten dat alleen Ralph een locale database draait en hebben de configfile van ramses aangepast zodat hij zijn gegevens ook in Ralph's database logt.(enige nadeel hiervan is dan dat Ramses alleen kan werken als Ralph er ook bij is)

Maandag 16-07-07

Planning game gehad met Rutger. We hebben gesproken over het verloop van het ontwerp en implementatie tot nu toe. Verder heeft hij ons verteld waar we op moeten letten en heeft hij gevraagd als we goed hebben gedocumenteerd wat nou eigenlijk voorrang heeft(requirements).

Dinsdag 17-07-07

De basis voor veel dingen hebben wij nu geschreven en hebben het een en ander al geïmplementeerd. Verder is er een banner gemaakt voor Easy Usage Statistics in Photoshop en hebben we gebrainstormed over de layout van ons programma.

Donderdag 19-07-07

Wat klassen geschreven in E.U.S. en onderzoek gedaan naar DAO(Data Acces Object) en we hebben gekeken waarop de huidige statiestieken gelogd worden. Verder hebben we de eerste tabellen gemaakt voor EUS en hebben gelijk in CSS een standaard layout bedacht voor de tabellen met informatie.

Vrijdag 20-07-07

Nog wat klassen gemaakt om informatie te weergeven(ContentPanes) en we hebben in EASY wat test data aangemaakt die we dan gelijk hebben gelogt in onze database. Dus het maken van aips(deposits) het publishen hiervan het maken van nieuwe users het downloaden van files etc.

Maandag 23-07-07

Planning game gehad met Rutger. We hebben gesproken over de zaken die nu al af zijn. En hij heeft ons gevraagd goed op te schrijven welke puntjes in het requirements verslag nu al af zijn. We kregen de tip elke dag wat tijd te besteden aan het loggen van dingen en dat we ons moesten proberen te houden aan de volgorde van het MoSCoW diagram.

Dinsdag 24-07-07

Wat gewerkt aan JUnit voor downloads en gewerkt aan de list methodes om die generiek te maken. Ook hebben we gekeken naar de flow van onze logger om te zien hoeveel keer de database wordt opgeroepen etc.(statistics van de statistics). Na dit te hebben gedaan hebben we vastgesteld wat we kunnen veranderen om het allemaal sneller te laten verlopen.

Woensdag 25-07-07

We hebben heel wat methodes aangepast in EASY om bv connections niet steeds opnieuw te maken maar om deze door te sturen naar de methode die het nodig heeft. Ook hebben wij i.p.v. steeds nieuwe lists te maken met databasecalls de lists doorgestuurd naar de betreffende methode zodat het geheel wat sneller werkt

Vrijdag 27-07-07

Gewerkt aan het "onder de motorkap " gedeelte van AIP's . Dus er is een modelclass gemaakt en een klasse AIPStatistic om informatie op te vragen en te kunnen bijhouden. Enige problemen opgelopen met het linken van een AIP aan een Category(minder makkelijk dan eerste instantie gedacht) Tree maar dat gelukkig opgelost.

Maandag 30-07-07

Probleem: Iedere keer als op een link geclicked wordt in EASY Usage Statistics dan wordt de informatie opnieuw opgezocht in de database waardoor er later als er heel veel gegevens in de database staan het systeem (te)traag kan worden.

We zijn aan het kijken naar mogelijkheden de tabellen op te slaan in een session of iets dergelijks zodat alle informatie maar 1 keer opgeroepen wordt

We zijn gaan zoeken naar mogelijkheden om grafieken te weergeven in EUS. Na lang zoeken hebben wij verscheidene mogelijkheden gevonden in java en na het discussiëren hierover hebben wij gekozen voor JFreeChart Omdat dit zoals de naam al doorgeeft freeware is en het redelijk simpel te implementeren is. Het weergeven van grafieken is in EUS geen prioriteit(maar het moet er alsnog wel strak uitzien).

Dinsdag 31-07-07

We hebben in JFreeChart wat demo's doorgenomen en hebben een PieChart gemaakt voor de users. Verder hebben we ook wat verder gewerkt aan het menu.

Woensdag 01-08-07

Er is gewerkt in EASY aan het generiek maken van bepaalde methodes en er is ook gekeken naar de mogelijkheden en wat we in de toekomst ook nog kunnen gaan aanpassen.

Vrijdag 03-08-07

We hebben gewerkt aan EASY wat facts aangepast bij General information en we hebben Histogrammen gemaakt voor downloads en deposits. Verder zijn bij de pie chart procenten toegevoegd .

Maandag 06-08-07

We Hebben wat gewerkt aan het log bestand en zijn precies gaan kijken welke requirements we nu precies af hebben. Gesprek gehad met Rutger en we zijn langs geweest bij Henk Harmsen(1 van de klanten) om wat inzicht te verkrijgen vanuit gebruikerspunt. We hebben alvast gekeken naar hoe we de queries en methodes wat kunnen versnellen en we hebben van henk ideeen gekregen over hoe we kunnen sorteren op verschillende data(maand , kwartaal, jaar).

Dinsdag 07-08-07

We hebben vandaag allebei een ander deel genomen, Ramses is gaan werken aan het van tot gedeelte bij de algemene informatie en Ralph heeft zich bezig gehouden met Aipstatistics .

Woensdag 08-08-07

Een dump gemaakt van de productie database en hebben de inforrmatie in ons systeem vervangen door de dump. De users uit het dmsaa.xml bestand ingelezen in onze eigen database.

Vrijdag 10-08-07

Bijna de heledag bezig geweest met het overzetten van de aips in de productie versie op onze eigen computers. Het overzetten van de aips(zonder de datafiles) duurde heel lang en na wat testen bleek dat niet alles was meegenomen in de transfer van linux naar windows.

Maandag 13-08-07

Gesprek gehad met Rutger en Milco, we hebben de stand van zaken besproken en we hebben het gehad over de beste aanpak voor de rest van de periode.

Dinsdag 14-08-07

Door gewerkt aan EUS. AipstatisticsContentPane en er is een begin gemaakt aan category statistics. Ook is date selector verder uitgebreid naar de andere contentpanes.

Woensdag 15-08-07

AipstatisticsContentPane is afgemaakt en het eerste deel van categories is gemaakt.Nog wat andere requirements aangepakt.

Vrijdag 17-08-07

Gewerkt aan het database model en aan een beter systeem om aips en hun metadata te weergeven. Nieuwe database “additional_info” met tabellen en relaties gemaakt.

Maandag 20-08-07 t/m Vrijdag 7-09-07 Tentamens+vakantie(Nog wel thuis gewerkt aan problemen en requirements)

Gewerkt aan een DBtask om alle aips in te lezen en bezig geweest met refactoring. Ook informatie gezocht over het maken van ons verslag.

Maandag 10- 09 - 07

Veel bezig geweest met het mergen van werk omdat we elkaar een lange periode niet gezien hebben.Daarna een inventarisatie gedaan van de requirements en precies afgestreept mat we al hebben.

Dinsdag 11-09 -07

Afspraak met Rutger en Milco, hier hebben we besproken hoe we EUS in easy gaan implementeren. Omdat er niet genoeg tijd meer is en omdat EASY binnenkort overgaat op Fedora gaat er hoogstwaarschijnlijk een soort standalone versie van EUS gedraait worden wanneer er informatie nodig is.

Woensdag 12-09-07

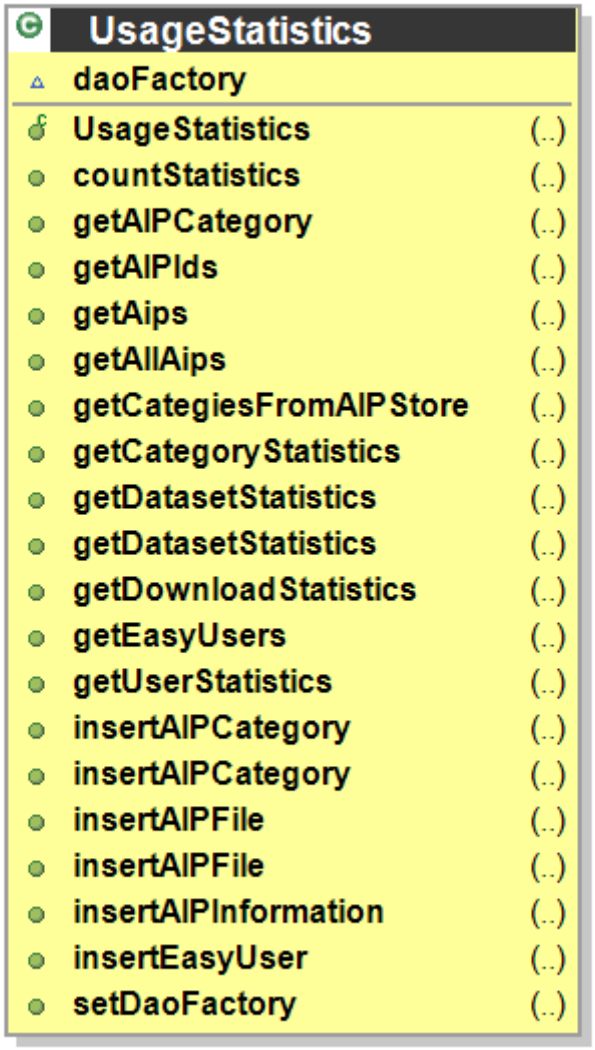
Een verdeling gemaakt van de resterende aips en door gegaan met werken aan de requirements.

Vrijdag 14-09-07

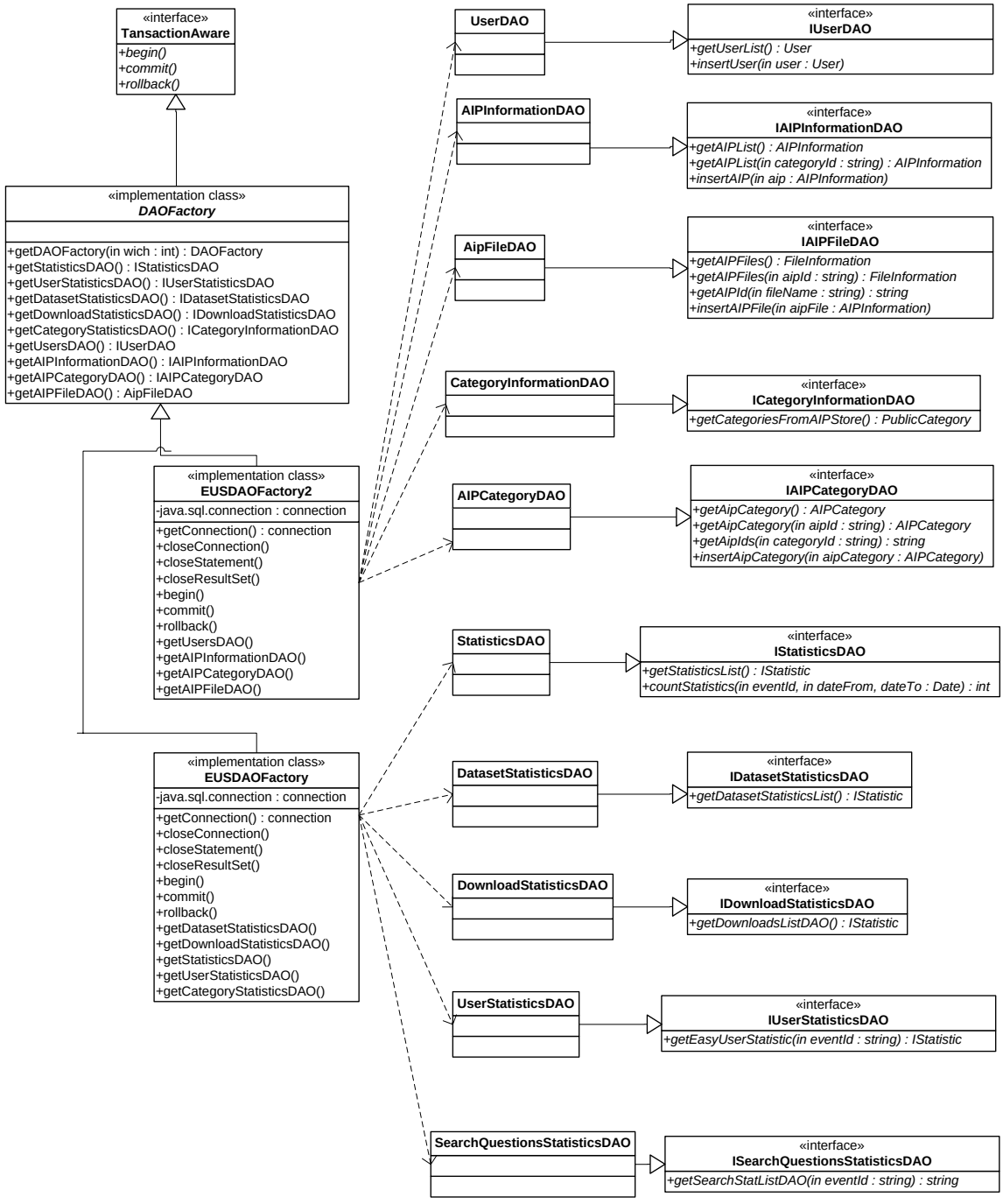
Wat nieuwe requirements afgerond en door gewerkt aan wat laatste dingetjes. Verder is er gisteren(Donderdag) ook een afspraak geweest met Peter.

Bijlage 4: Klassendiagrammen

nl.knaw.dans.easy.usageStatistics.business

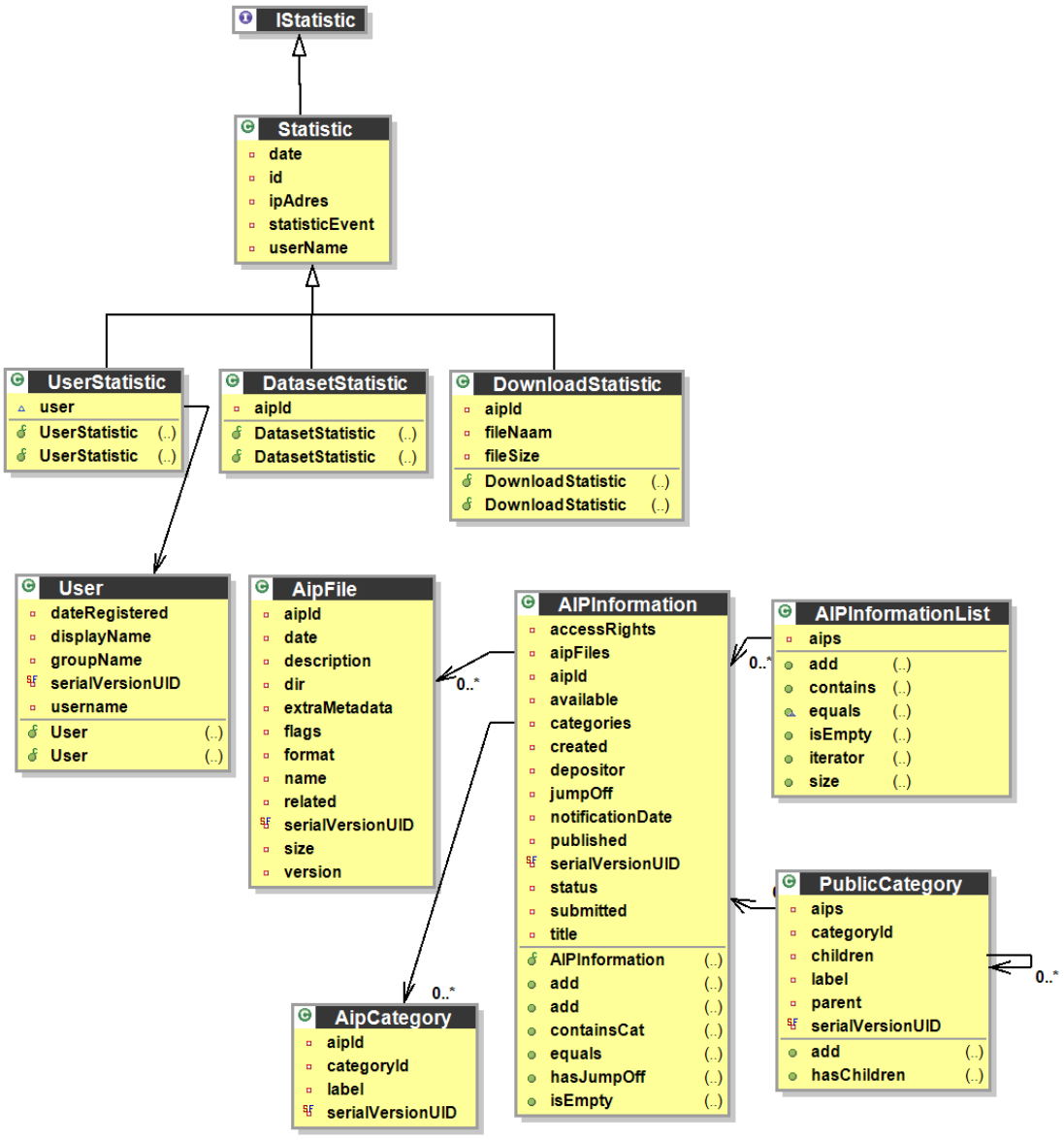


nl.knaw.dans.easy.usageStatistics.dao:



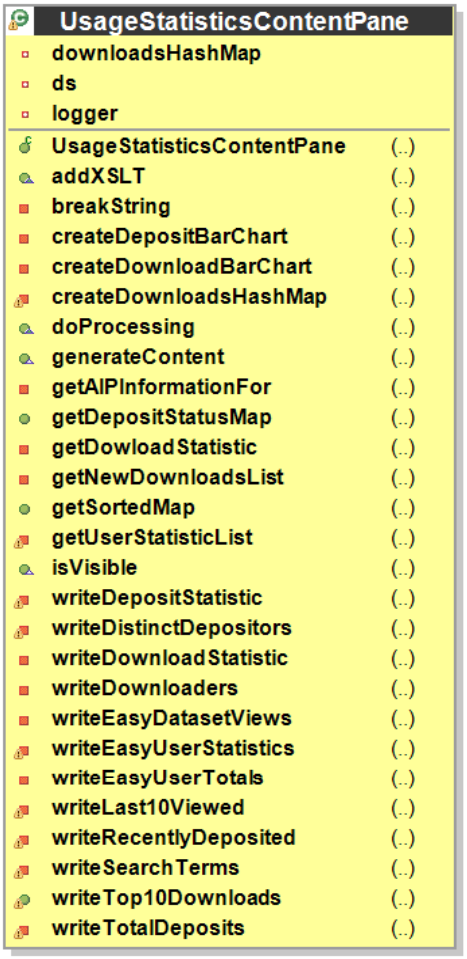

StatisticsDAO
LIST
conn
logger
ps
rs
size
statisticList
StatisticsDAO (..)
StatisticsDAO (..)
countStatistics (..)
createStatistic (..)
createStatisticsList (..)
getList (..)
getList (..)
getSize (..)
getStatisticsList (..)
SearchQuestionStatisticsDAO
LIST
LIST2
logger
ps
ps2
rs
rs2
searchIdList
statisticEventId
SearchQuestionStatisticsDAO (..)
getSearchStatListDAO (..)
getSearchTerm (..)
UserStatisticsDAO
logger
login
loginEventId
logout
logoutEventId
ps
registered
registrationEventId
rs
visit
visitEventId
UserStatisticsDAO (..)
getAdditionalUserInfo (..)
getEasyUserStatistic (..)
getLoginEventId (..)
getLogoutEventId (..)
getRegistrationEventId (..)
getVisitEventId (..)
DownloadStatisticsDAO
LIST
conn
dlist
logger
ps
rs
statisticEventId
DownloadStatisticsDAO (..)
DownloadStatisticsDAO (..)
createDownloadsList (..)
getDList (..)
getDownloadStatistic (..)
getDownloadsListDAO (..)
DatasetStatisticsDAO
conn
depositEventId
logger
ps
publishEventId
rs
viewedEventId
DatasetStatisticsDAO (..)
DatasetStatisticsDAO (..)
getAipId (..)
getDatasetStatistic (..)
getDatasetStatisticsList (..)
getDatasetStatisticsList (..)
getDepositEventId (..)
getPublishEventId (..)
getViewedEventId (..)
AIPFileDAO
INSERT
conn
createAIPFileList (..)
getAIPFiles (..)
getAIPFiles (..)
getAIPIds (..)
insertAIPFile (..)
AIPInformationDAO
INSERT
SELECT
SELECT2
conn
logger
createAIPList (..)
getAIPList (..)
getAIPList (..)
insertAIP (..)
UsersDAO
INSERT
SELECT
conn
logger
ps
rs
size
uList
UsersDAO (..)
UsersDAO (..)
contains (..)
createUser (..)
createUserList (..)
getGroupHash (..)
getSize (..)
getUList (..)
getUserList (..)
insertUser (..)
setSize (..)
setUList (..)
AIPCategoryDAO
INSERT
conn
createAIPCategoryList (..)
getAipCategory (..)
getAipCategory (..)
getAipIds (..)
insertAipCategory (..)
CategoryDAO
conn
logger
CategoryDAO (..)
addChildren (..)
getCategoriesFromAIPStore (..)

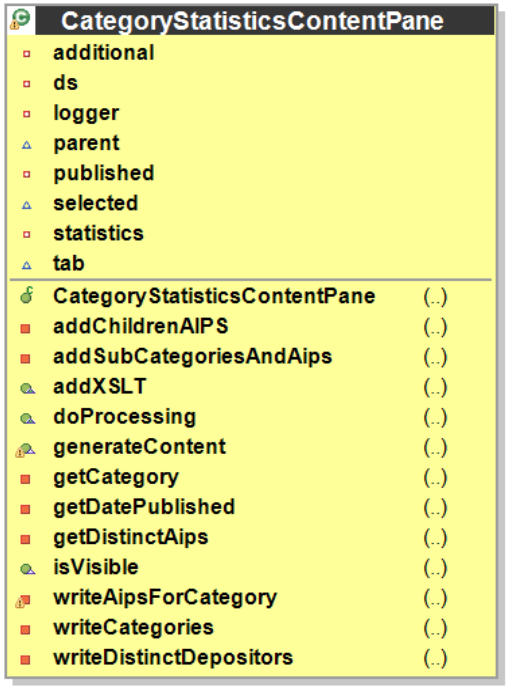
nl.knaw.dans.easy.usageStatistics.model:

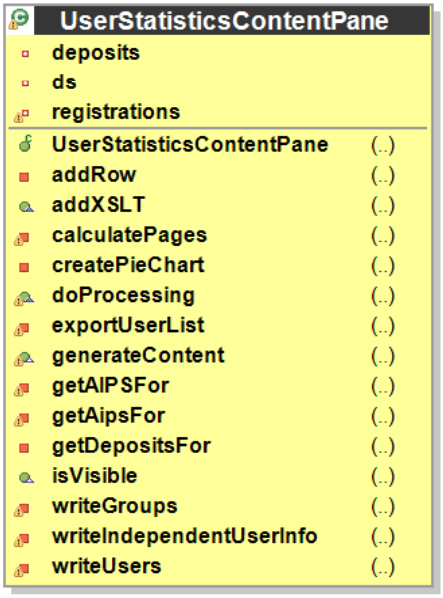


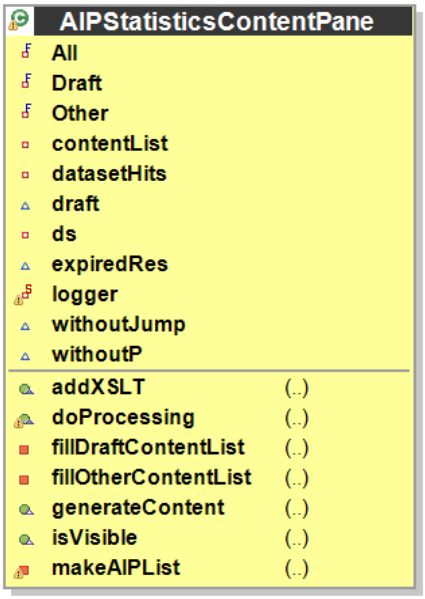
nl.knaw.dans.easy.usageStatistics.view:

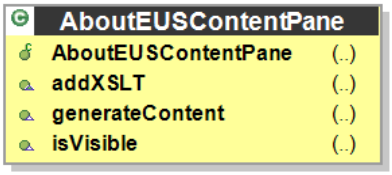
nl.knaw.dans.easy.usageStatistics.content:



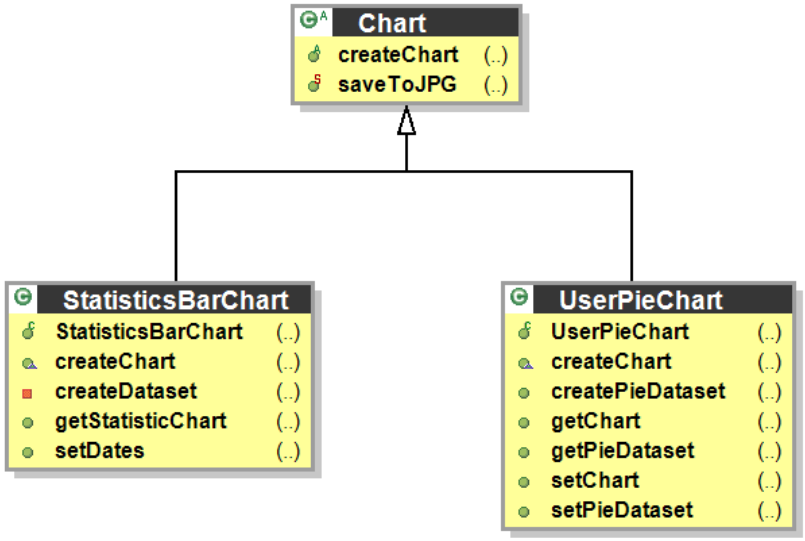




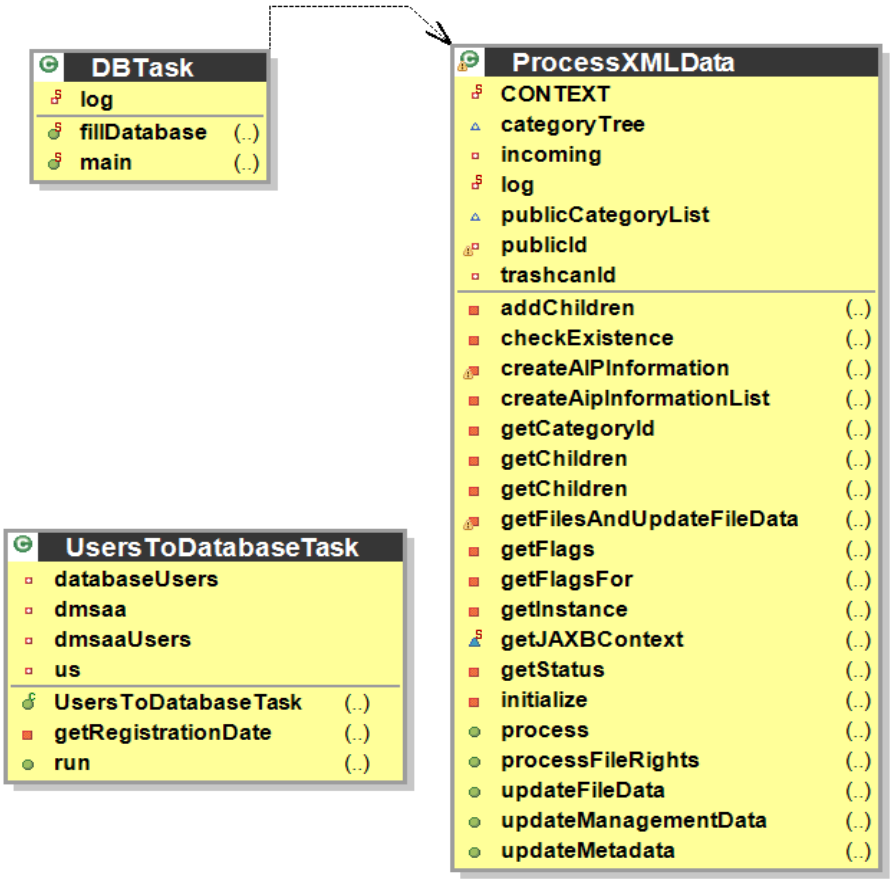




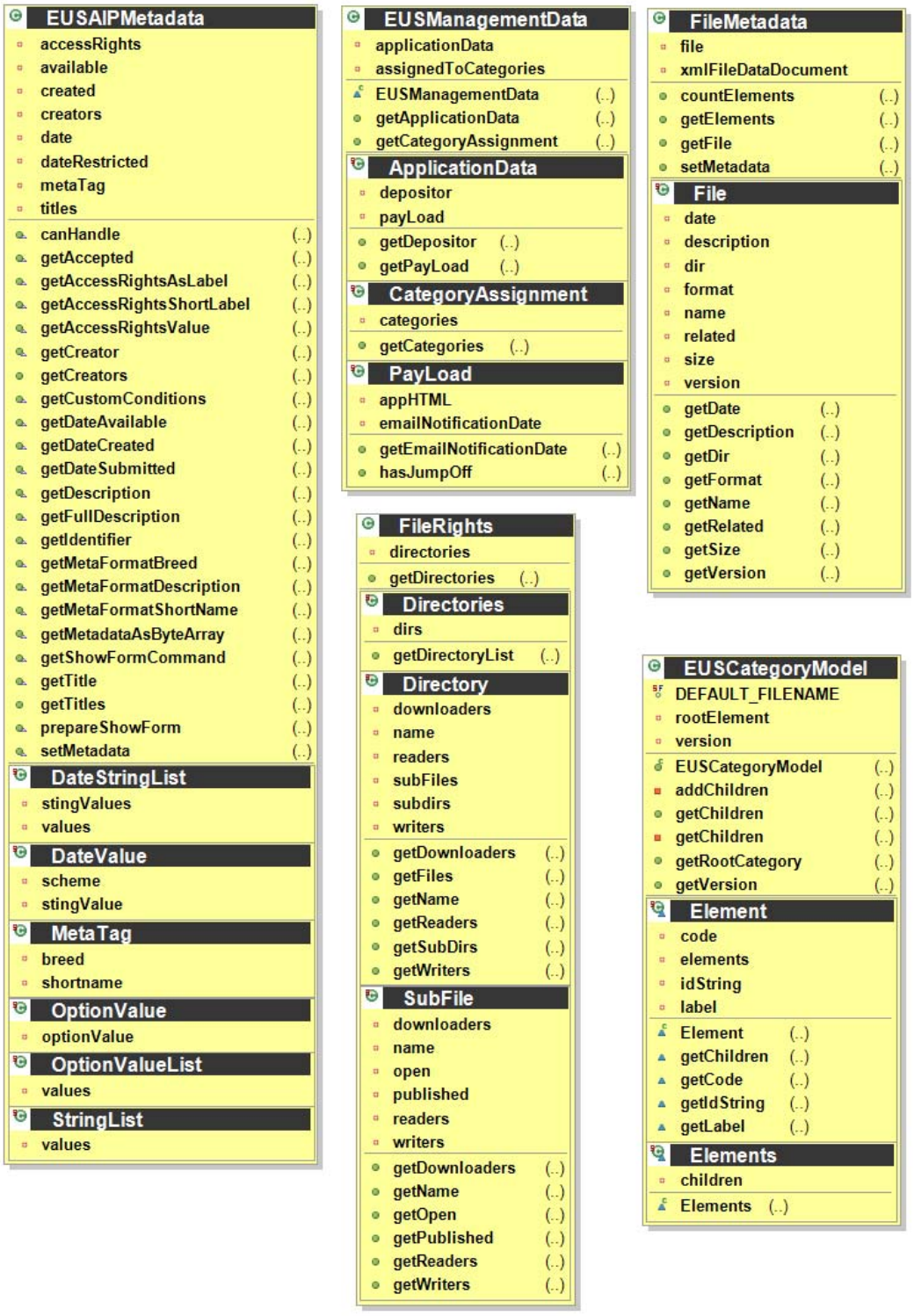
nl.knaw.dans.easy.usageStatistics.chart



nl.knaw.dans.easy.usageStatistics.task



nl.knaw.dans.easy.usageStatistics.helper:



Bijlage 5: Specificatie EasyStaistics database

Statistics Plugin

Voor het management is er een “Statistics” plugin toegevoegd, om gegevens over het gebruik van EASY te kunnen loggen. Vanwege de huidige opzet van EASY is de “Statistics” plugin opgedeeld in twee plugins:

1. Een “Statistics” plugin die een extension point heeft
2. Een “Standard statistics” plugin die het extension point van de “Statistics” plugin implementeert .

De reden hiervoor is dat de plaatsen waar verschillende gebruiksgegevens verzameld moet worden verspreid zijn over verschillende plugins. Verder moet EASY ook zonder een implementatie van de “Statistics” plugin extension point blijven werken.

Loggen van gebruiksgegevens

Voor het management worden de volgende gegevens in een database vastgelegd om later met een ander programma geanalyseerd te worden.

1. Bezoeker die de hoofdpagina van EASY bezoekt.
2. Alle requests van een bezoeker/gebruiker.
3. Bezoekers die inloggen
4. Gebruikers die uitloggen
5. Bestanden van een dataset die worden gedownload inclusief de grootte ervan
6. Zoektermen die ingevoerd worden in het zoekscherm
7. Gedeponeerde datasets
8. Gepubliceerde datasets
9. Bekeken datasets
10. Bezoekers die zich registreren

Naaste de bovengenoemde gegevens worden standaard bij elke gegeven de tijd, gebruikersnaam en IP-adres vastgelegd.

Configuratie parameters

Onder de map “conf” is de configuratie bestand te vinden.

1. “Statistics” plugin

Parameter	Omschrijving
nl.knaw.dans.dms.statistics.StatisticsManager.logStatistics	Boolean waarde om vastleggen van gegevens aan of uit te zetten. Mogelijke waarden “true” of “false”. Standaard is “true”.

2. “Standard statistics” plugin

Parameter	Omschrijving
nl.knaw.dans.dms.statistics.standard.dao.MysqlStatisticsDAOFactory.driver	String waarde geeft de driver aan die gebruikt moet worden voor de database. Standaard waarde is “com.mysql.jdbc.Driver”.
nl.knaw.dans.dms.statistics.standard.dao.MysqlStatisticsDAOFactory.url	String waarde die het pad aangeeft van de database die gebruikt moet worden. Standaard waarde is “jdbc:mysql://localhost/easystatistics”
nl.knaw.dans.dms.statistics.standard.dao.MysqlStatisticsDAOFactory.userName	String waarde met het gebruikersnaam die moet worden gebruikt voor de database. Standaard waarde is “easystatistics”
nl.knaw.dans.dms.statistics.standard.dao.MysqlStatisticsDAOFactory.password	String waarde met het password die moet worden gebruikt voor de database. Standaard waarde is “easystatistics”
nl.knaw.dans.dms.statistics.standard.DatabaseWriter.bufferTreshold	Integer waarde die aangeeft bij hoeveel gegevens die in de buffer zitten gedumpt wordt naar disk. Standaard waarde is 2000. Dit is voorzorgsmaatregel voor het geval er iets mis is met de database
nl.knaw.dans.dms.statistics.standard.FileWriter.dumpPath	String waarde die het pad aangeeft waar de dump bestanden moeten worden opgeslagen. Standaard waarde is “/” **Note: Het pad moet wel een bestaande pad zijn**

Gebruik maken van gebruiksgegevens loggen

Om gebruiksgegevens te loggen kun je dat op de volgende manier doen:

```
StatisticsManager.getInstance().logStatistic(StatisticEvent.AIPSTORE_DATASET_VIEWED, _session.getUserName(), _session.getRemoteAddress(), _aip.getId());
```

De methode “logStatistic” heeft drie vaste parameters en een variabele parameter.

Parameter	Type	Omschrijving
event	StatisticEvents	Vast te leggen gebruiksgegeven
userName	String	Naam van de ingelogde gebruiker
ipAddress	String	IP-adres van de gebruiker
values	String (vararg)	0 of meer gegevens die horen bij de vast te leggen gebruiksgegeven

Databasemodel gebruiksgegevens

statisticEventsPKid description fullDescriptionstatisticsPKid date userName ipAddressFK1statisticEventstatisticValuesPK,FK1orderIdPKstatisticId

Tabel “statisticsEvents”

Veld	Type	Omschrijving
id	INT(11)	Autonummer veld voor identificatie
description	VARCHAR(50)	Korte omschrijving van de gebruiksgegeven
fullDescrpition	TEXT	Human readable omschrijving van de gebruiksgegeven

Tabel “statistics”

Veld	Type	Omschrijving
id	INT(11)	Autonummer veld voor identificatie
date	DATETIME	Datum en tijdstip van het gelogde gebruiksgegeven
userName	VARCHAR(50)	Gebruikersnaam
ipAddress	VARCHAR(50)	IP-adres van de gebruiker
statisticEvent	INT(11)	Verwijzing naar het type gelogde gebruikersgegeven

Tabel “statisticValues”

Veld	Type	Omschrijving
orderId	INT(11)	Volgnummer van de gelogde waarde van een gebruiksgegeven
statisticId	INT(11)	Verwijzing naar het gelogde gebruikersgegeven

Om de database aan te maken en te verwijderen is er een ant build script gemaakt. Er zijn twee targets “local” en “deployment”. - “local” kan gebruikt worden om lokaal of op test server een database aan te maken en te verwijderen. - “deployment” kan gebruikt worden om de productie server een database aan te maken en te verwijderen Beide kan men d.m.v. van twee properties files configureren. De properties files zijn te vinden onder database waar ook de database script te vinden is.

Note: De “Standard statistics” plugin is gebouwd voor een MySQL database. Wil je lokaal kunnen testen zul je MySQL moeten installeren.

Author: Taner Gedikoglu.