Uitwisselingssysteem historische persoonsgegevens

Bachelor scriptie over het ontwerp en de bouw van een uitwisselingssysteem voor historische persoonsgegevens.

Maurits Meijer

1-5-2012

Inhoud

1. Inleiding 4
De opdracht 4
Het onderwerp 5
De opbouw van dit verslag 7
2. het systeem 8
Overzicht 8
Harvester 9
Repository 10
Beheer 10
3. Algemene aanpak 11
De twee problemen 11
Het systeem in de wereld 11
Centraal systeem 11
Onderlinge uitwisseling 12
Uitwissel protocol 13
OAI-PMH 13
Het request 13
De response 14
Het systeem 14
De Data Layer 14
De Data Access Layer 15
De Business Layer 16
De Interface Layer 16
4. Bestaande dataformaten 17
Genealogische data vs. akte data 17
Transcriptie 17
Data formaat Digitale stamboom 18
Genealogische dataformaten 18
Bestaande formaten 19

5. Het Dataformaat 21
Actoren 21
Archiefinstellingen 21
Onderzoeker 21
Derde partijen 21
Bouwer/Onderhouder dataformaat 21
Doelstellingen 22
Klein en flexibel 22
Betekenisvol 22
Intuïtief 22
Uitbreidbaar 22
Techniek 22
Uitwerking 23
6. Dataformaat naar data en database 24
Database 24
Indexen 24
B-Tree 25
Data omzetting 25
Overige stored procedures 26
7. Protocollen en beveiliging 27
Beveiliging 27
OAuth 27
HMAC 28
SHA1 29
De kwaliteit van HMAC-SHA1 29
Diffie-Hellman 29
8. Tools 31
Ontwikkeltools 31
Gebruikte libaries 31
9. Slot 33
10 Bijlagen 36

1. Inleiding

De opdracht

Het doel van dit project is om een ontwerp en implementatie te maken van een systeem voor de uitwisseling van historische persoonsgegevens.

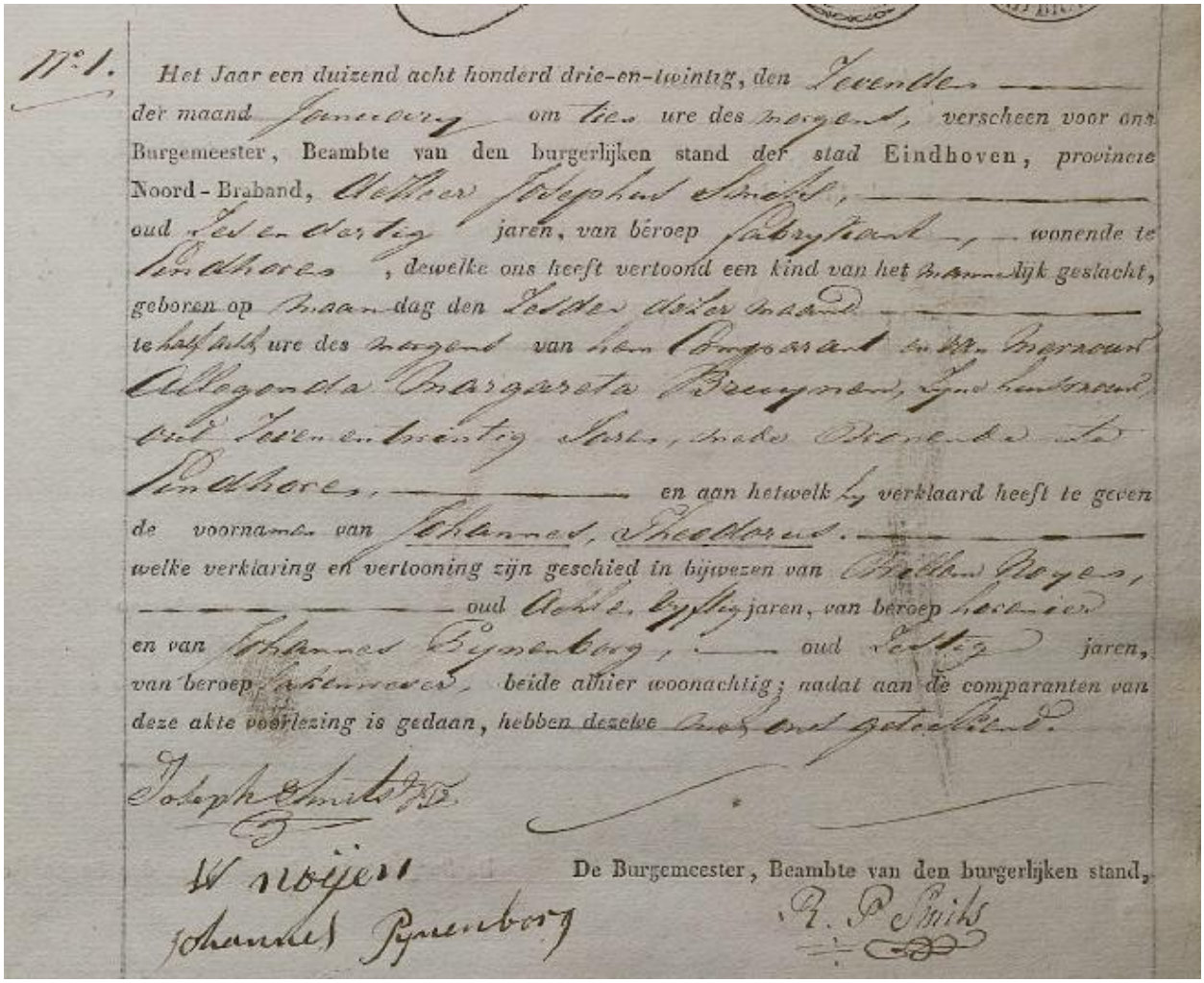
Het idee en het verzoek om dit project te gaan doen is afkomstig van Mindbus[1]. Mindbus is onder andere de bouwer en ontwerper van diverse webtoepassingen voor het beheer en de ontsluiting van archiefgegevens. Een bijzondere vermelding verdient het dat Mindbus de bouwer is van de Digitale Stamboom[2]. De Digitale Stamboom is een applicatie voor het beheren en ontsluiten van historische persoonsgegevens. Er zijn 7 archief instellingen aan gesloten bij de digitale stamboom. Deze hebben samen ongeveer 6 miljoen akten gedigitaliseerd waarop in totaal 19 miljoen personen.

Ondergetekende is als werknemer bij Mindbus belast met het onderhoud van deze applicaties. De Digitale Stamboom is eigendom van erfgoed Delft en omstreken en wordt gebruikt door verschillende erfgoedinstellingen door heel Nederland.

---

[1] Website Mindbus BV http://mindbus.nl [gezien 3 mei 2011]
[2] Website Digitale Stamboom http://digitalestamboom.nl [gezien 3 mei 2011

Het onderwerp



Figuur 1 Akte uit de burgerlijke stand

De historische persoonsgegevens binnen archiefdiensten hebben, door het gebruik van verschillende applicaties, verschillende formaten. Ook worden ze op verschillende manieren openbaar gemaakt. Naast de verschillen in applicaties zijn de archieven ook afhankelijk van het formaat waarin data door derde partijen, bijvoorbeeld kerkarchieven of particulieren, wordt aangeleverd. De historische persoonsgegevens zelf bestaan daarnaast ook nog eens uit een ruime verzameling van verschillende soorten. Zo zijn er bijvoorbeeld doop-, trouw- en begraafboeken, notariële akten, verklaringen van armoede enz.

Het gevolg hiervan is dat Archiefdiensten deze gegevens, die in het algemeen gebruikt worden voor genealogisch onderzoek, op verschillende websites of op verschillende plaatsen binnen een website aanbieden. Dat is een belangrijk struikelblok voor de onderzoeker die zijn zoekopdracht op verschillende plaatsen binnen een archief moet herhalen. In de huidige praktijk probeert men dit soms te ondervangen door de zoekvraag simultaan aan de verschillende applicaties te stellen, het zogenaamde gedistribueerd zoeken, en de resultaten per applicatie weer te geven. Deze oplossing komen we

bijvoorbeeld tegen op de website van het Gemeentearchief Rotterdam[3]. Dit is in veel gevallen onpraktisch als het gaat om genealogisch onderzoek. Onder andere doordat het zoekscherm beperkt wordt door het totaal van beperkingen van de zoekinterface van de diverse applicaties. Een zoekvraag met bijvoorbeeld een sterfdatum als parameter kan, normaal gesproken, niet aan een applicatie voor doopakten worden gesteld.

Naast het uitwisselen van informatie binnen de archiefinstellingen wordt er ook informatie uitgewisseld tussen archiefinstellingen. Door onderling informatie uit te wisselen kan de informatie van de verschillende instellingen in een keer aan de onderzoekers worden aangeboden. Naar de mogelijkheid om door meerdere collecties van verschillende instellingen te zoeken is een grote vraag. Dit blijkt uit de populariteit van sites als Genlias[4], de centrale zoekingang van de Digitale Stamboom en Thematis[5].

Naast de service die het verzamelen van de informatie van meerdere instellingen op een plek aan de genealogen biedt is het ook wetenschappelijk van belang om een centrale database te hebben, dan wel aan te kunnen leggen. Hierbij valt bijvoorbeeld te denken aan het Links project. Maar op tal van gebieden is dergelijke informatie zeer nuttig. Denk bijvoorbeeld aan historische demografie, sociaal historische economie, naamkunde, epidemiologie, antropologie, historisch sociologische genetica.

Het verenigen van deze verschillende data uit verschillende bronnen en van verschillende archiefdiensten is dus wenselijk.

Historische persoonsgegevens worden met name via websites aangeboden. De informatie wordt door de aanbieders van de verschillende systemen voor de digitalisatie van akten telkens op een eigen manier opgeslagen. Hoe nauwkeurig of uitgebreid de informatie uit een akte wordt gedigitaliseerd is aan de individuele archieven. Ook hier in zijn grote verschillen. De ene instelling neemt bijvoorbeeld in een transcriptie van een huwelijksakte het beroep van de vader van de bruid op, terwijl een andere instelling dit achterwegen laat.

Op het gebied van genealogische data zijn er verschillende standaard dataformaten. Voor de transcriptie van akten met historische persoonsgegevens zijn er echter geen standaard dataformaten. Om optimaal gebruik te kunnen maken van de beschikbare data is het belangrijk dat er een dataformaat wordt gevonden dat alle getranscribeerde historische persoonsgegevens in welk systeem dan ook kan bevatten.

Vaak is de database van historische persoonsgegevens binnen een instelling onderhevig aan vele veranderingen. De bestaande data moet soms verbeterd worden op fouten die in de transcriptie zijn geslopen. En aangezien nog lang niet alle historische persoonsgegevens zijn getranscribeerd is er regelmatig aanwas van data.

Bij het beschikbaar stellen van een grote hoeveelheid, regelmatig veranderende, informatie is het belangrijk te letten op het reguleren van de beveiliging van het systeem. Het is bijvoorbeeld niet wenselijk dat iemand het systeem zwaar belast door alle informatie op te vragen zonder dat daar een

---

[3] Website Gemeentearchief Rotterdam online: http://www.gemeentearchief.rotterdam.nl [gezien 3 juli 2009]

[4] Genlias online: http://genlias.nl [gezien 10 november 2010]

[5] Thematis online: http://www.thematis.nl [gezien 12 april 2011]

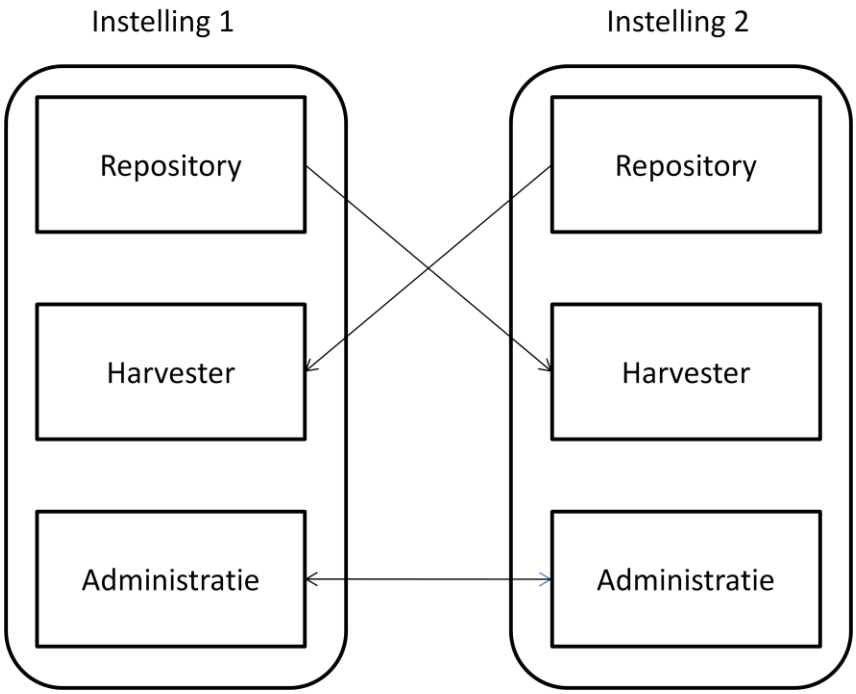
redelijke reden voor is. Ook kan het bezwaarlijk zijn als derden de volledige dataset herpubliceren. Hierom wordt in dit project ook aan de beveiliging van het systeem enige aandacht geschonken.

De opbouw van dit verslag

De rest van dit verslag is als volgt opgebouwd. In hoofdstuk 2 wordt een beschrijving gegeven van het systeem dat het resultaat is van dit project. In hoofdstuk 3 wordt verteld hoe het systeem tot stand is gekomen. In de hoofdstukken 4 en 5 wordt nader ingegaan op het dataformaat. Vervolgens worden in hoofdstukken 6 en 7 nader ingegaan op respectievelijk de implementatie van de dataopslag en van de beveiliging. En in hoofdstuk 8 wordt een overzicht gegeven van de ontwikkelomgeving en de hulpmiddelen. Tot slot is er een samenvatting en een conclusie in hoofdstuk 9. De code en de technische documentatie en handleidingen zijn als bijlage aan dit verslag toegevoegd.

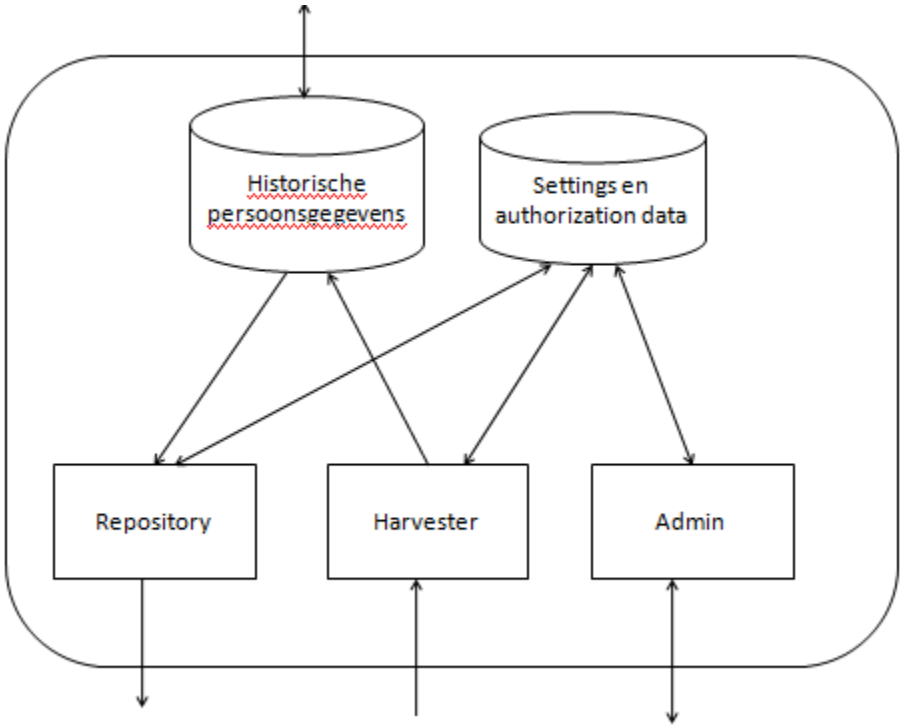
2. Het systeem

Overzicht



Figuur 2 Overzicht van het systeem en de interactie tussen twee instanties van het systeem.

Het systeem bestaat uit 3 losse onderdelen, de harvester, het repository en een administratie deel. De harvester is een deel van het systeem dat informatie binnen haalt. Het repository (opslagplaats) is het deel van het systeem dat de informatie aanbied. Het administratie deel is een interface voor het aanpassen van instellingen en autorisatie gegevens. De instellingen communiceren de data met elkaar van het repository van de ene instelling naar de harvester van de andere. Tussen de administratieve gedeeltes word gecommuniceerd om authenticatie en autorisatie informatie door te geven.



Figuur 3 overzicht van het systeem

Een enkele instantie van de applicatie is te zien in figuur 3. De communicatie tussen de harvester, het repository en de administratie module binnen een instantie van het systeem verloopt via de database. De database is in figuur 3 weergegeven als twee conceptueel losse delen, een die de historische persoonsgegevens bevat en een die de instellingen bevat. In de implementatie is dit een enkele database hoewel de persoonsgegevens en instellingtabellen onafhankelijk van elkaar zijn.

Harvester

De harvester haalt op gezette tijden de nieuwe en gewijzigde data op van de diverse repositories. Om de opzet van het hele systeem zo homogeen en portable mogelijk te houden is dit onderdeel ook geïmplementeerd als een webservice (in plaats van bijvoorbeeld een Windows service).

Om een webapplicatie regelmatig automatisch te laten starten gebruik ik de cache items van .Net. Deze items verlopen na een in te stellen tijd en doen bij het verlopen een callback naar de applicatie. Die callback wordt gebruikt om het harvesting proces inwerking te zetten. Zodra het proces in werking is gezet wordt er opnieuw een cache item geplaatst die de volgende callback gaat veroorzaken. Als het proces nog draait op het moment dat er een callback komt, loopt het proces gewoon door en wordt alleen een nieuw callback item gezet.

Het proces haalt alle repositories waarvan geharvest kan worden uit de database. Vervolgens wordt er met één of meerdere requests alle data opgehaald van dat repository. De binnengehaalde data wordt per record in de A2A (Archive2Archive) class opgeslagen in de database.

Als alle records van alle repositories zijn opgehaald sluit de applicatie zichzelf af.

Repository

Het repository gedeelte van het systeem biedt de data volgens het A2A (Archive2Archive) model aan harversters aan. Het is een webservice die een http-request ontvangt. Er zijn twee soorten requests mogelijk. Een request voor autorisatie om te harvesten en een OAI-PMH request voor data.

Alle requests komen binnen via webrequests naar oai2.aspx of DHke.aspx.

Metadata afhandeling

Er zijn twee klassen om de data afhandeling te doen. Dat aantal kan eenvoudig worden uitgebreid. Beide klassen zijn hetzelfde opgebouwd. Eén klasse is voor data die is opgevraagd in het A2A-formaat. De andere is voor data opgevraagd in het Dublin-Core formaat. Het aanbieden van data in dit formaat dient louter om formeel te voldoen aan de OAI-PMH standaard.

In de klasse wordt bij de constructie een XML-document gegenereerd. Hiervoor wordt de data betreffende het opgevraagde record uit de verschillende tabellen gehaald en opgeslagen in XML-tags. Na de constructie kan de XML worden opgevraagd. Daarnaast kan het worden opgevraagd in het record formaat van OAI-PMH.

Beheer

Om de autorisaties, de aan- en afmeldingen te regelen en de statistieken te kunnen bekijken is er een beheersmodule. Dit deel van het systeem bestaat uit een aantal webpagina’s, een login-module en een Ajax-module.

3. Algemene aanpak

In dit hoofdstuk kijk ik hoe het uitwisselingssysteem in de bestaande infrastructuur gaat worden ingepast. Vervolgens hoe instanties van het systeem met elkaar gaan communiceren. En ten slotte welke onderdelen het systeem heeft en hoe die onderdelen met elkaar communiceren.

De twee problemen

Bij het ontwikkelen van het uitwisselingssysteem zijn er twee grote uitdagingen. Het eerste is het maken van een precieze definitie van de data die uitgewisseld moet worden, of te wel het data formaat. In hoofdstuk 4 wordt uitgebreid ingegaan op de eisen waaraan het dataformaat moet voldoen en in hoofdstuk 5 wordt uitgelegd tot welk datamodel die eisen hebben geleid.

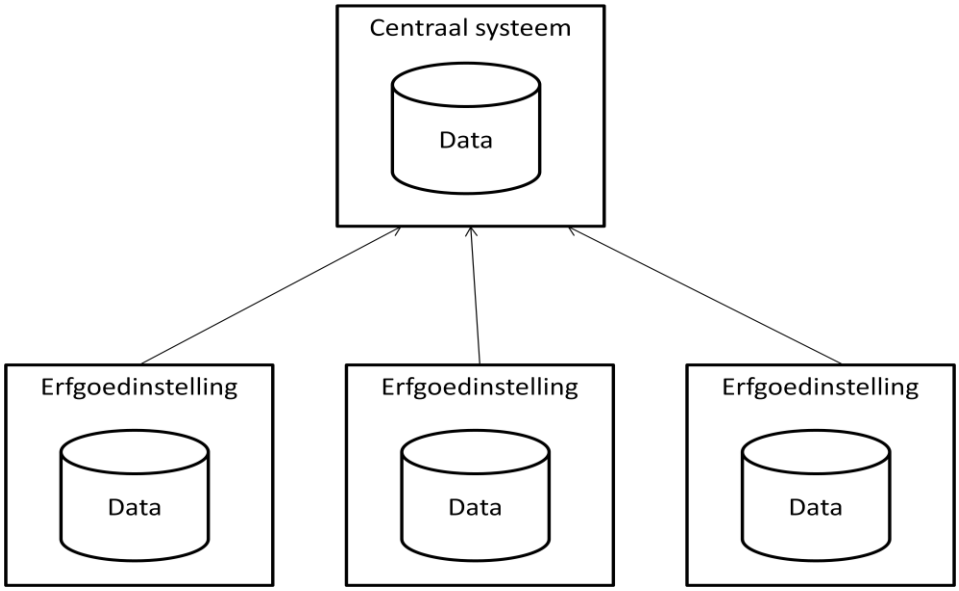
De andere uitdaging is de ontwikkeling van het systeemgedeelte. Voor dat het systeem ontwikkeld wordt moet er eerst een ontwerp van het systeem worden gemaakt. De rest van dit hoofdstuk gaat daarover. Daarnaast wordt in hoofdstuk 8 ingegaan op de werkwijze bij de implementatie aan de hand van de gebruikte tools.

Het systeem in de wereld

Er zijn twee belangrijke scenario's voor het gebruik van een uitwisselingssysteem voor historische persoonsgegevens. Het eerste is om de data centraal beschikbaar te maken. En het tweede is om een bestaande collectie aan te vullen met informatie uit een ander systeem.

Centraal systeem

Een centraal systeem beschikt zelf niet over data maar haalt deze op bij verschillende andere partijen. Een nuttige toepassing hiervan is het mogelijk maken van het zoeken door een verzameling data uit verschillende instellingen (zie figuur 4).



Figuur 4 Data-uitwisseling ten behoeven van een centraal systeem

In dit scenario ontvangt het centrale systeem de data van de erfgoedinstellingen en slaat het op.

Distibuted search

Voor zoektoepassingen vervult dit scenario dezelfde functie als een zgn. distributed search systeem, waar in aan het centrale systeem een zoekvraag wordt gesteld en het centrale systeem vervolgens die zoekvraag stelt aan elk van de aangesloten erfgoedinstellingen en de resultaten verzameld en als antwoord terug geeft. Het systeem van Thematis is een voorbeeld van distributed search.

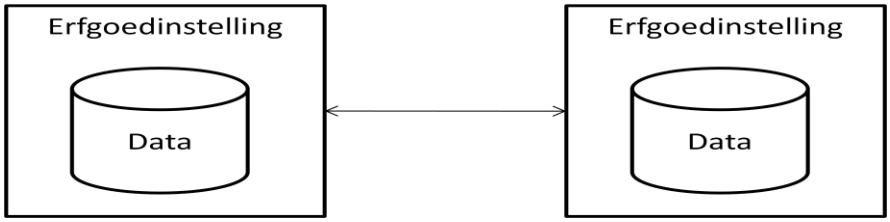
Distributed search heeft een aantal grote nadelen. Zo ben je voor een volledig antwoord op je zoekvraag afhankelijk van de werking van alle systemen waarin gezocht wordt. Verder is een dergelijk systeem zo snel als de langzaamste component.

Het voorgestelde systeem met de data centraal in een systeem heeft deze problemen niet. Het beschikt namelijk altijd zelf over de data. Het systeem met een centrale opslag heeft als nadeel dat de informatie 2 keer word opgeslagen, namelijk een keer bij de instellingen en een keer centraal. Dit nadeel weegt echter niet op tegen de genoemde problemen met distributed search. Naast zoektoepassingen is dit scenario ook bruikbaar voor andere toepassingen die betrekking hebben op de verzamelde data. Hierbij kan men bijvoorbeeld denken aan onderzoek naar naam frequenties.

Een hiërarchische structuur zoals hierboven beschreven past uitstekend in hoe de archiefwereld ten opzichte van de zoekportalen in elkaar zit. De archiefinstellingen zijn redelijk autonoom in welke informatie ze digitaal willen delen en met wie. Een model met meer dan 2 lagen leidt daarentegen tot verschillende problemen. Voorbeelden hiervan zijn bijvoorbeeld juridische problemen: kan een instantie in een tussen laag zomaar de informatie van een instelling met een derde partij delen, maar ook logische problemen: hoe voorkom je dubbele data als een instelling bij twee instanties in de tussen laag is aangesloten enz. Aangezien de oplossing met meer dan twee lagen geen functionele voordelen lijkt te hebben is het systeem ingericht met als uitgangspunt, dat elke instelling alleen haar eigen originele data deelt.

Onderlinge uitwisseling

Een tweede vorm van uitwisseling is een onderlinge uitwisseling tussen twee erfgoedinstellingen. Dit gebeurt bijvoorbeeld als een collectie over de grenzen van instellingen heen gaan. Dat kan gebeuren als de gemeentegrenzen in de loop van de jaren zijn veranderd of dat de collecties zijn gemaakt door een instituut dat de gemeentegrenzen overschrijdt. Een voorbeeld hier van is het archief van een rechtbank.



Figuur 5 Data-uitwisseling ten behoeven van een andere instelling

Bij uitwisseling in deze vorm beschikken, anders dan de toepassing in het centrale systeem, alle partijen zelf over gegevens en vullen ze deze aan met gegevens van een andere partij.

Het is in dergelijke systemen wenselijk dat er een selectie plaats kan vinden op de collecties of delen ervan die worden uitgewisseld, zodat de niet relevante data niet wordt uitgewisseld.

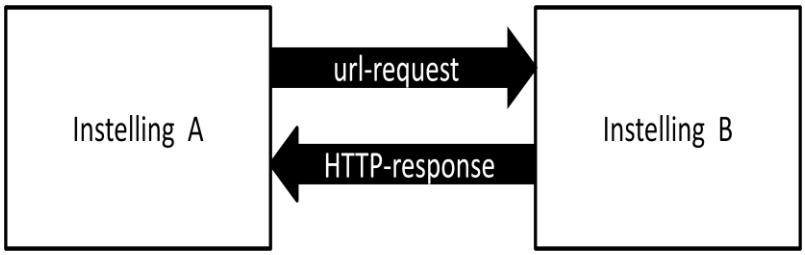
Voor de onderlinge uitwisseling van gegevens is distributed search ook een alternatief. Dat wil zeggen dat zowel in het eigen systeem wordt gezocht als dat er een zoekvraag wordt gesteld aan een ander systeem. Distributed search heeft in dit scenario dezelfde nadelen die bij een centraal systeem worden genoemd.

Zowel het centralesysteem als het onderlinge uitwisselen worden door het ontworpen systeem ondersteund.

Uitwissel protocol

OAI-PMH

Voor het uitwisselen van data zelf zijn goede afspraken nodig. Elke geschikte akte moet vanaf de publicatie aangeboden worden. Ook moeten er correcties en verwijderingen worden gecommuniceerd. Het Open Archives Initiative(OAI)[6] heeft hier voor een standaard ontworpen. Deze standaard Open Archives Initiative- Protocol for Metadata Harvesting (OAI-PMH) bestaat al sinds 2001. De laatste versie, versie 2.0 kwam 14 juni 2002 uit. OAI-PMH wordt met name gebruikt om de inhoud van bibliotheekcollecties uit te wisselen. In OAI-PMH doet de harvester een request naar een URL en specificeert het in de query string parameters voor dat request. Deze parameters zijn bijvoorbeeld in welke periode de records zijn toegevoegd of bij welke subset van de data deze horen. Het antwoord in OAI-PMH is een http-bericht met in de body XML met de opgevraagde data.



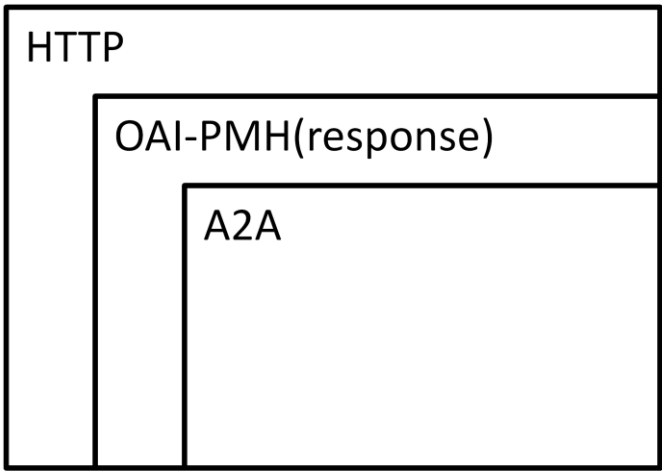
Figuur 6 communicatie tussen twee instanties van het systeem

Het request

Een request in OAI-PMH is een http request naar het url van het repository met in de query Sting de parameters voor het request. De eerste parameter in een request is the commando. OAI-PMH kent commando's om ondermeer de een record op te vragen een groep records op te vragen of een groep identifiers of te vragen. De parameters voor een commando worden ook in de query string mee gegeven. Voorbeelden van dit soort parameters zijn in welke periode een record gewijzigd is, of in welk formaat het antwoord moet zijn.

[6] OAI-PMH http://www.openarchives.org/OAI/openarchivesprotocol.html [gezien 14 januari 2010]

Volgens het protocol moet een repository minstens het Dublin Core (DC)[7] formaat ondersteunen.



Figuur 7 de hiërarchische opbouw van het antwoord

Dublin Core

Het Dublin Core formaat is een data formaat dat een aantal beschrijvende dingen over de aangeboden data zegt. Onder andere de titel, het onderwerp, het type, de bron en het formaat worden beschreven. Het Dublin Core formaat is voor historische persoonsgegevens niet erg belangrijk. Het tweede dataformaat dat we gebruiken, A2A, is een dataformaat dat de historische persoonsgegevens zelf kan bevatten.

De response

Naast de content wordt er ook een header mee gestuurd. De header bevat informatie over het record zoals wanneer het record is toegevoegd of verwijderd.

Het OAI-PMH biedt de mogelijkheid om informatie te comprimeren. Dat is voor ons belangrijk, het gaat immers dagelijks om veel informatie. De compressie gebeurt op het HTTP[8] niveau met de daarvoor geldende standaard.

Het systeem

In figuur 8 is te zien hoe een enkele instantie van het systeem er uit komt te zien. Het systeem is weergegeven in 4 lagen. In de onderste laag zit de data. Deze laag is voor heel het systeem hetzelfde. Boven deze laag zit de data access laag. Deze laag is een verbinding tussen de logica laag en de data laag. Boven de data access laag zit de logica laag. In deze laag worden alle functionele bewerkingen gedaan. Bijvoorbeeld het vertalen van een request naar een vraag aan de data access layer en het vertalen van het antwoord van de data access layer naar een response. De bovenste laag is de interface laag. Dit is de laag waar externe objecten, bijvoorbeeld een harvester in een andere instantie, mee kunnen communiceren.

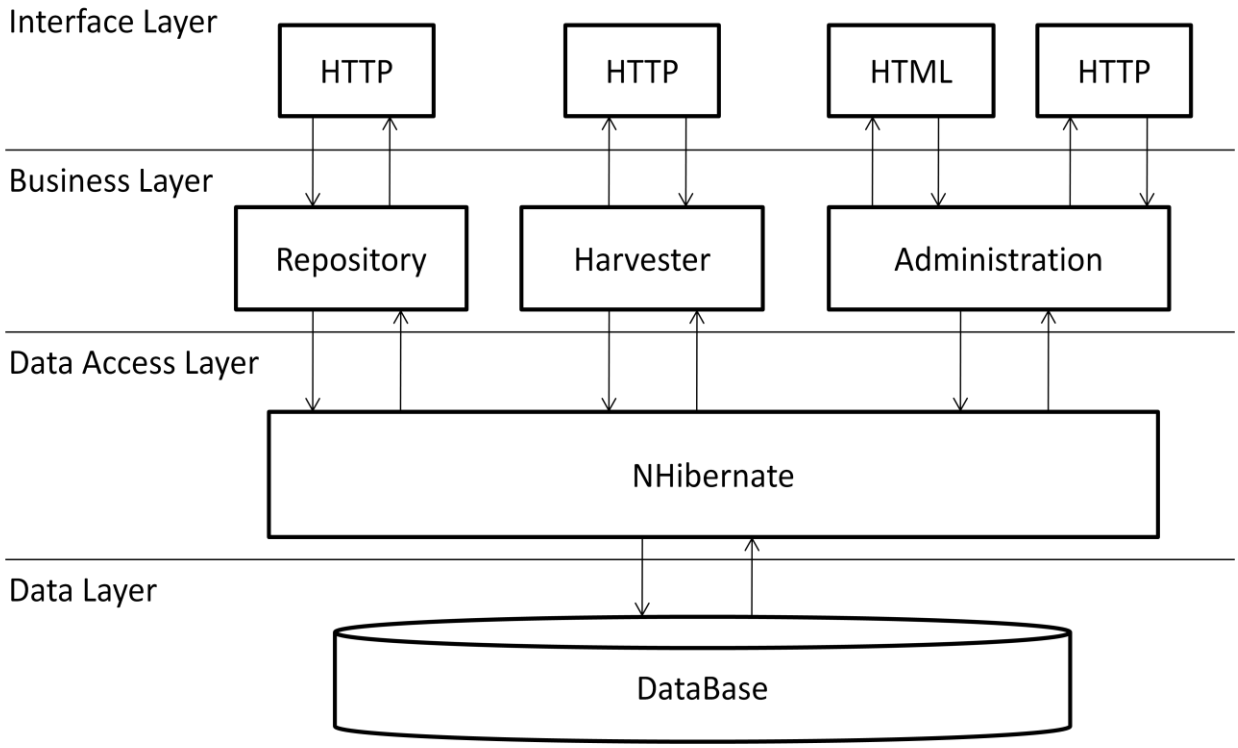
De data layer

De data layer bevat een enkele database. In deze database worden drie dingen opgeslagen. Als eerste de configuratie gegevens. De configuratie gegevens bestaan voornamelijk uit de identificatie, authenticatie en autorisatie gegevens van andere instanties van het systeem. Deze gegevens worden gemaakt door de administration module en worden gebruikt door zowel de harvester, namelijk om te

[7] Dublin Core http://dublincore.org/specifications/ [gezien 20 april 2011]
[8] http http://tools.ietf.org/html/rfc2616 [gezien 20 april 2011]

weten waar en wanneer de data geharvest moet worden en hoe de harvester zich identificeert. Ook het repository maakt gebruik van deze data voornamelijk om te controleren of een andere instantie die data opvraagt daar toe geautoriseerd is.

In dezelfde database vinden we ook de historische persoonsgegevens die via het repository worden aangeboden. Hoewel het systeem geen strikte beperkingen bevat om informatie die via de harvester is verkregen weer te delen via het repository is het conceptueel juist om beiden te scheiden. Dat betekent uiteraard niet dat de te delen informatie en de geharveste informatie niet in de zelfde tabellen zouden kunnen staan, waarbij deze bijvoorbeeld door de gegevens over de herkomst worden gescheiden.



Figuur 8 layer architectuur van een instantie van het systeem

De data access layer

Omdat de informatie in de database tussen alle modules wordt gedeeld en de database ingewikkeld van structuur is, is er gekozen om NHibernate als data access layer te gebruiken. Met behulp van NHibernate worden de records in de database gemapped naar classes in C# en kunnen de bewerkingen op de database gemakkelijker worden uitgevoerd (zie hoofdstukken 6 en 8). De data access layer is voor alle modules vrijwel gelijk, zodat een wijziging in de structuur van de database gemakkelijk van worden vertaald naar de C# implementatie.

De business layer

In de business layer vinden we de logische gedeeltes van het systeem. Deze gedeeltes vallen uiteen in drie delen, het repository, de harverster en het administratie gedeelte. Deze 3 delen communiceren met elkaar alleen via de database.

Het repository controleert voor ontvangen requests of ze de juiste autorisatie hebben, vraagt de gegevens op en verstuurd ze.

De Harvester bepaald aan de hand van de instellingen in de database wanneer het aan welke andere instantie van het systeem welk request voor data verstuurt en slaat de data die het terug krijgt op in de database.

De administratie module haalt de instellingsinformatie op uit de database of slaat deze op. Ook zorgt de administratie module voor het afhandelen van autorisatie requests van andere instanties van het systeem of van het versturen van deze requests.

De interface layer

De interface layer van het systeem praat met de buiten wereld. Voor het repository betekent dit het ontvangen van requests voor data en het versturen van de antwoorden. Het .Net framework zorgt er voor dat de door de applicatie verzonden data in de vorm van http berichten wordt verstuurd. Voor de harvester geldt min of weer hetzelfde, de op gevraagde URLs worden door het .Net framework omgezet in http, en het antwoord dat terug komt wordt door het framework omgezet in een object.

De administratie module communiceert op twee manieren. De eerste is de communicatie tussen deze module en de andere instanties van het systeem voor bijvoorbeeld het uitwisselen van autorisatie gegevens. Deze vorm van communicatie verloopt gelijk aan die van de harvester en het repository via http.

De module dient echter ook als interface voor gebruikers die de instellingen van het systeem willen wijzigen of een systeem aan de lijst van de te harveste systemen wil toevoegen.

Samenvattend heeft kan het voorgestelde systeem zowel voor uitwisseling met een centraal systeem, als voor onderlinge uitwisseling worden ingezet. Voor de uitwisseling van data tussen instanties van het systeem wordt gebruik gemaakt van het bestaande OAI-protocol. Verder heb ik in dit hoofdstuk de lagen en de modules binnen een enkele instantie van het systeem beschreven. Aan de hand van deze beschrijving is het hier voorgestelde systeem geïmplementeerd.

4. Bestaande dataformaten

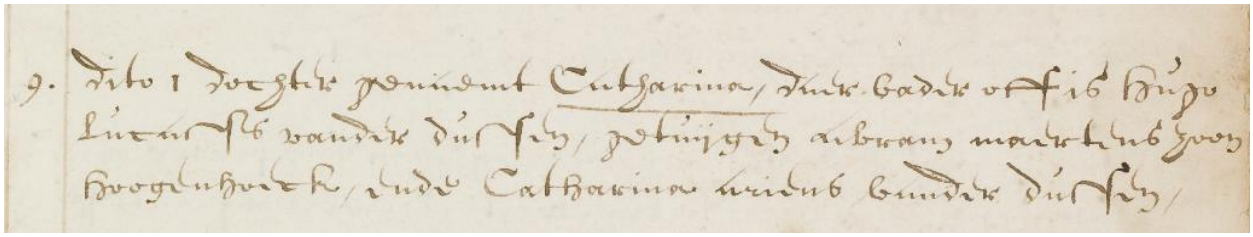
Voor het uitwisselen van data in het systeem moet er een uitwisselformaat worden vastgesteld. Het heeft de voorkeur om de data-uitwisseling in een bestaand dataformaat te doen om zo eenvoudige aansluiting bij bestaande systemen te vinden en geen energie in het ontwikkelen van een nieuw dataformaat te hoeven steken. Omdat er vrijwel geen open dataformaten zijn voor historische persoonsgegevens is er ook gekeken naar genealogische dataformaten.

Genealogische data vs. akte data

Historische persoonsgegevens lijken op het eerste gezicht veel op genealogische gegevens. Er zijn echter enkele belangrijke verschillen. In de eerste plaats richten de historische persoonsgegevens zich op historische objecten, zoals akten en niet op historische personen zoals genealogische data. Een ander belangrijk verschil is dat genealogische gegevens het resultaat zijn van onderzoek en interpretatie van data. De gegevens zijn daarom minder absoluut dan de historische persoonsgegevens die een exacte neerslag van de akte zijn.

Transcriptie

Wat betreft de exacte neerslag van de akte is wanneer men dat nader bekijkt nog wel meer te zeggen. Een akte komt niet zomaar uit de lucht vallen. In het algemeen wordt een akte opgemaakt aan de hand van een gebeurtenis. De meeste akten die getranscribeerd worden, zijn tussen de 50 en 500 jaar oud.



Figuur 9 een doopakte uit 1616[9]

Met namen in oudere akten kan het lastig zijn de exacte woorden te ontcijferen. Een transcriptie van een naam op een akte is daarom soms niet 100% zeker. Maar de onzekerheid in deze ontcijfering is anders van natuur dan de claims die binnen genealogie worden gemaakt. Het is natuurlijk wel belangrijk dat dergelijke onzekerheden een plaats hebben in het datamodel. Wanneer een akte wordt getranscribeerd word niet de hele tekst getranscribeerd maar alleen de gegevens die belangrijk zijn: in bovenstaande akte de dopeling, de vader, de getuigen en de datum.

[9] DTB Delft inv. 54 , folio 1 http://delft.digitalestamboom.nl/scan.aspx?book=D&id=95569 [gezien 6 september 2011]

Data formaat Digitale Stamboom

Het eerste dataformaat waar we naar kijken is dat van de Digitale Stamboom. Meer precies de akten uit de kerkboeken, boeken waar in de kerkgenootschappen akten opmaken van de gebeurtenissen dopen, trouwen, begraven en de akten uit de burgerlijke stand, opgemaakt aan de hand van de gebeurtenissen geboorte, huwelijk en overlijden. In de Digitale Stamboom, en voor zover ik heb kunnen nagaan in alle andere vergelijkbare systemen, worden deze op geslagen in de vorm van een voornamelijk plat model. Dat wil zeggen er is een enkele tabel waarin kolommen stukken data representeren die in alle akten van dat type voorkomen.

Voornaam kind	Patroniem kind	Voorvoegsel achternaam kind	Achternaam kind	doopdatum
Marcellus		van	Rest	15-06-1603

Figuur 10 voorbeeld van een deel van een doopakte in een plat datamodel

De kolommen in een dergelijk dataformaat zijn voor de verschillende aktetypen verschillend, een doopakte heeft andere kolommen dan een huwelijksakte. Dat betekend dat voor elk nieuw type akte het data model moet worden aangepast. Het platte datamodel heeft een ander groot nadeel namelijk velden die zich herhalen. Zo kan bijvoorbeeld een persoon meerdere beroepen hebben, een gebeurtenis meerdere getuigen hebben enz. Het aantal keer dat deze velden zich kunnen herhalen is in principe onbeperkt. Het gebruik van een plat datamodel betekent dat het aantal herhaling van bepaalde velden moet worden beperkt en vastgesteld.

Voor de boven genoemde akten is het, hoewel niet fraai, nog wel in redelijkheid te doen deze beperking in te bouwen. Zo kent de Digitale Stamboom bijvoorbeeld een veld voor het beroep van een persoon en velden voor maximaal 5 getuigen bij een doop. Die oplossing geeft in de praktijk weinig problemen. In notariële akten, bijvoorbeeld een akte van een veiling, zijn er veel personen genoemd op een akte. In de Digitale Stamboom is er daarom voor een meer hiërarchisch model gekozen waar in de personen in een losse tabel worden opgeslagen met verwijzingen naar de rij in de aktetabel waar ze bij horen. Dit lost echter niet het probleem op van de verschillende formaten.

Genealogische dataformaten

Op het internet wordt volop gedaan aan genealogie. Dit is bijvoorbeeld te zien aan de populariteit van de diverse mailinglijsten en fora. Ook is er een grote variëteit aan websites en desktop applicaties. Niet zo in het oog springend, maar wel degelijk, typerend is de hoeveelheid dataformaten die er zijn voor genealogische data. Helaas hebben al deze dataformaten een manco dat ze voor onze doelen onpraktisch of onbruikbaar maakt. Het belangrijkste probleem is dat deze formaten zich centreren rond personen en relaties tussen personen, terwijl de akten juist spreken over gebeurtenissen en actoren bij gebeurtenissen. Niettemin is het leerzaam om de bekendste dataformaten langs te lopen.

Bestaande formaten

Gedcom

Dit formaat, waarvan de meest recente versie (5.5 standaard+errata sheet)[10][11] in januari 1996 uitkwam (de eerste versie dateert van 1984), is de meest gebruikte standaard op het gebied van genealogische data. Het formaat wordt echter niet meer onderhouden. Gedcom wordt in veel genealogische applicaties gebruikt. Dit zijn zowel web als desktop applicaties. Het formaat bestaat uit tekstregels die zijn opgebouwd uit een nummer voor niveau, eventueel een identifier, een 4-letterige afkorting als keyword en eventueel data.

Het model is hiërarchisch. Met op het bovenste niveau een header en een trailer en daarnaast ingangen voor individuals, families, sources, notes, repositories, multimedia, submiter. We missen hier een ingang op gebeurtenis, het centrale punt van een akte, en houden buiten de individual weinig nuttigs over. Als we een niveau dieper kijken naar sources zien we dat daarin ruimte is voor gebeurtenissen. De gebeurtenissen in Gedcom dienen ter ondersteuning van de relaties tussen personen. Bij historische persoonsgegevens stellen we echter geen relaties tussen personen vast, maar welke rol een persoon speelt bij de gebeurtenis waarvan een akte is opgemaakt.

```
0 HEAD
1 SOUR Reunion
2 VERS V8.0
2 CORP Leister Productions
1 DEST Reunion
1 DATE 11 FEB 2006
1 FILE test
1 GEDC
2 VERS 5.5
1 CHAR MACINTOSH
0 @I1@ INDI
1 NAME Bob /Cox/
1 SEX M
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @I2@ INDI
1 NAME Joann /Para/
1 SEX F
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @I3@ INDI
1 NAME Bobby Jo /Cox/
1 SEX M
1 FAMC @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @F1@ FAM
1 HUSB @I1@
1 WIFE @I2@
1 MARR
1 CHIL @I3@
0 TRLR
```

Figuur 11 Voorbeeld van een deel van een GEDCOM bestand

Genealogical Data Model van Gentech(gdm) in de vorm van gdmxml[12]

Het Genealogisch data model van Gentech is een data model dat zich richt op zowel de opslag van genealogische data als op het opslaan van gegevens van het researchproces. Het datamodel heeft een uitwerking in xml onder de naam gdmxml.

Het grootste probleem van dit formaat is het enorme gebied dat dit formaat probeert te beslaan. Zij probeert namelijk naast persoonsgegevens ook onder andere de volledige financiële administratie van de onderzoeker te bewaren. Het datamodel valt in 3 delen uiteen: een administratie deel, een bewijs deel en een conclusie deel. Het conclusie deel zou het meest interessant zijn voor onze doelstellingen. In het doelstellingen gedeelte staan de aannames centraal. De aannames vullen telkens delen in van de informatie van een historisch persoon, bijvoorbeeld welke levensloop gebeurtenissen er zijn geweest etc. Hoewel het formaat interessant en uitgebreid is blijft het hangen in het genealogisch gebied. Wel leerzaam is de manier waarop dit formaat met tijd en plaats om gaat. Zij gebruiken hier voor ondermeer een hard/zacht formaat. Dat wil zeggen dat het mogelijk is om zowel een precieze datum(hard) op te slaan, maar ook om een datum op te slaan als een tijdsperiode waar binnen de gebeurtenis plaats heeft

---

[10] De GEDCOM standaard http://homepages.rootsweb.ancestry.com/~pmcbride/gedcom/55gctoc.htm [gezien 3 mei 2011]
[11] Wikipedia entry GEDCOM http://en.wikipedia.org/wiki/GEDCOM [gezien 7 september 2011]
[12] De Gentech gdm standaard http://www.ngsgenealogy.org/cs/GenTech_Projects [gezien 3 mei 2011]

gevonden(zacht). Dat laatste is een oplossing voor een probleem dat binnen de historische persoonsgegevens ook voorkomt, namelijk gebeurtenissen zonder precieze datum.

```
<PersonalName>
  <NamePart Type="given">Catherine</NamePart>
  <NamePart Type="given">Zeta</NamePart>
  <NamePart Type="surname">Jones</NamePart>
</PersonalName>
```

Figuur 12 een persoonsnaam in Gedc

Gedc(2.0)[13]

Deze standaard van februari 2009 is het nieuwste op het gebied van xml genealogische dataformaten. Het gaat echter uit van grote datasets, in de zin van dat bijvoorbeeld een kwartierstaat 1 Gedc record is. (Een kwartierstaat is een binaire boom met daar in de voorouders van diegene waarvan men uitgaat. Elke persoon in de kwartierstaat wordt breadth first een index toegekend. Diegene van wie men uit gaat krijgt index ,1 de vader index 2, de moeder index 3, de vader van de vader index4 enz.)

Elk Gedc record bevat tevens de definities van types die gebruikt worden (bijvoorbeeld hoe een achternaam eruit mag zien). Dat is te veel overlap wanneer je voor elke akte een record hebt. Verder willen wij zo min mogelijk lettercodes gebruiken. Gedc bevat bijvoorbeeld letter codes voor de personstatus D=dead en u=died unmarried enz. Dit is niet wenselijk van uit het oogpunt van de leesbaarheid.

Samenvattend sluit geen van de hiergenoemde formaten goed aan bij het transcriberen van akten. De formaten kunnen veel informatie opslaan die buiten ons domein liggen. Waar door het in het uitwisseling systeem overbodige informatie kan bevatten. Informatie die we wel willen vast leggen heeft geen plaats (bijvoorbeeld een akte datum, of een opmerking over een onleesbare letter) of moet vervormd worden om in het dataformaat te passen.

[13] De Gedc standaard http://www.sunflower.com/~billk/GEDC/index.htm [gezien 3 mei 2011

5. Het Dataformaat

Voor het dataformaat voor uitwisseling wordt uitgegaan van de gegevens op de historische objecten. Deze objecten bijvoorbeeld een akte van overlijden of een volkstelling, komen allemaal tot stand naar aanleiding van een officiële gebeurtenis. In het datamodel staan gebeurtenissen dan ook centraal. Uiteraard zijn ook de personen en hun relaties tot de gebeurtenis belangrijk. Een vierde belangrijk onderdeel is de context van het record. Bij context valt ondermeer te denken aan het jaar waar in het stuk werd opgemaakt of de plek waar het stuk tegenwoordig bewaard wordt.

Actoren

Bij het ontwerp van het dataformaat is rekening gehouden met een aantal actoren.

Archiefinstellingen

De medewerkers van archiefinstellingen die de akten willen digitaliseren. Voor hen is het van belang dat zij al de data die ze willen digitaliseren in het dataformaat kwijt kunnen. Omdat de set van objecten die we willen digitaliseren zeer divers is en er objecten binnen kunnen vallen waarvan we de daar op vermelde informatie onmogelijk kunnen voorspellen, is het van belang dat het datamodel flexibel is. Een ander belangrijk punt is dat de het lang niet altijd duidelijk is wat er precies op het object staat.

Onderzoeker

In de inleiding zijn al een aantal onderzoeksgebieden genoemd waar historische persoonsgegevens van belang zijn. Voor de onderzoeker is het van belang dat het datamodel semantisch eenduidig is. Verder is het met name voor de genealogische onderzoeker van belang dat er verwezen kan worden naar de bron van informatie.

Derde partijen

Derde partijen die in een implementatie willen A2A gebruiken hebben vooral technische eisen. Zo is het belangrijk dat de definitie het mogelijk maakt met diverse tools te werken. Een voorbeeld hiervan is de intellisense en autocomplete functies die beschikbaar komen in diverse editors bij het gebruik van xml op basis van xsd of om bij het voorbeeld van xml te gebruiken tools als xsd, schematron etc. Verder is het uiteraard van belang dat het dataformaat eenvoudig te begrijpen is.

Bouwer/Onderhouder dataformaat

Het bouwen en onderhouden van het dataformaat heeft voor een groot deel dezelfde eisen als de derde partij. Het is daarnaast noodzakelijk voor het onderhoud en eventuele uitbreidingen van het dataformaat. Wat betreft de aanpassingen aan het datamodel is het belangrijk dat er gestreefd kan worden naar backwards compatibility. Backwards compatibility is van belang voor diegenen die het datamodel implementeren. De wensen voor uitbreiding kunnen ook voor een deel in de data in het datamodel worden verwerkt. Bijvoorbeeld door het gebruik van key-value pairs.

Doelstellingen

Aan de hand van de bovenstaande beschrijving van de wensen van diverse partijen die met het datamodel te maken krijgen kunnen de volgende principes worden gehanteerd bij het maken van het datamodel.

Klein en flexibel

Een klein datamodel maakt het werk voor met name de derde partijen en de bouwer van het datamodel eenvoudiger. Om toch aan de wens van de archiefinstellingen te kunnen voldoen om hun historische persoonsgegevens in het datamodel kwijt te kunnen moet het datamodel een zekere mate van flexibiliteit krijgen. Zodat data die relatief zeldzaam op een akte voorkomen, bijvoorbeeld de kleur van iemands ogen, ook kan worden getranscribeerd.

Betekenisvol

Met name voor de onderzoekers is het van belang te weten wat de getranscribeerde data betekent. Dit betekent dat elk stuk data in het datamodel voorzien moet worden van een ondubbelzinnige beschrijving. Dit principe concurreert soms met de flexibiliteit.

Intuïtief

Met name voor de technici van de derde partijen en de bouwer is het van belang dat het datamodel goed en eenvoudig te begrijpen is in het kader van de wensen van de archiefinstelling en de onderzoeker. Wanneer men activiteiten in de richting van het datamodel ontwikkelt kan er immers vanuit worden gegaan dat dergelijke kennis aanwezig is. Goede documentatie bij het datamodel is onmisbaar om dit voor elkaar te krijgen.

Uitbreidbaar

Het is naar mijn mening een illusie om in een keer een gebalanceerd datamodel op te zetten. De informatie is daarvoor te divers. Het is daarom belangrijk dat het eenvoudig is om zowel van een dynamisch veld naar een vast veld, als van een vast veld naar een dynamisch veld te veranderen.

Techniek

Technisch gezien zijn er een aantal opties om een dataformaat te definiëren. Gangbaar zijn vooral xml, SQL, JSON flattext, YAML. Ik heb gekozen om xml te gebruiken. De tooling voor JSON, flattext en YAML is veel minder dan die voor xml of SQL. SQL is als uitwisselformaat tussen verschillende systemen, met allicht verschillende databases niet echt geschikt. Daar komt bij dat je er niet vanuit kan gaan dat een aangesloten systeem zich op een SQL database, baseert. Daarnaast heeft xml het voordeel dat het een begrepen en geaccepteerde techniek is binnen de archiefwereld. De goede tooling van xml geeft ook uitstekende mogelijkheden om zonder speciale software met het datamodel te werken. De intellisense van Visual Studio is hier een goed voorbeeld van.

Uitwerking

Voor de bouw van het datamodel is uitgegaan van gebeurtenissen. Gebeurtenissen zoals een huwelijk, een overlijden enz. vormen de reden voor het opmaken van een akte. Daarnaast zijn de personen op de akte uiteraard belangrijk. De personen en de gebeurtenissen worden aan elkaar gekoppelt door relaties, deze relaties beschrijven onder andere de rol die een persoon bij een gebeurtenis vervulde. Een persoon kan net als in een akte relaties hebben met meerdere gebeurtenissen. Een vierde belangrijk element in het datamodel is de bronvermelding. Dit element verteld onder andere over waar en wanneer de akte is opgemaakt en waar die nu te vinden is.

Deze vier elementen bevatten elk hun eigen sub-elementen. Aan de hand van het element persoon zal ik kort de manier van het opzetten van deze items doorlopen.

```
<Person pid="Person5">
        <PersonName>
                <PersonNameFirstName TranscriptionRemark="onduidelijk geschreven">
                        Mes
                </PersonNameFirstName>
                <PersonNameLastName>Jans Dr</PersonNameLastName>
        </PersonName>
</Person>
```

Figuur 13 Een persoon in A2A

Een persoon op een akte kan vele verschillende eigenschappen hebben. Voorbeelden hiervan zijn uiteraard de naam, het beroep, de lengte enz. De frequentie waarmee deze eigenschappen voorkomen kan erg verschillen. Vrijwel alle akten bevatten een naam. Terwijl lengte van de persoon slechts in enkele akten is terug te vinden. Omdat we het datamodel klein willen houden en omdat er geen uitputtende lijst is van eigenschappen die op een akte voor kunnen komen is er voor gekozen om alleen de belangrijkste eigenschappen een vaste plaats te geven in het datamodel. Voor de overige eigenschappen is gekozen om deze op te slaan in key value pairs. Het opslaan on key value pairs betekent dat het wat moeilijker wordt om te achterhalen wat de betekenis van bepaalde data is. De zelfde eigenschap kan immers op meerdere manieren omschreven worden.

Voor informatie die in een andere taal op de akte staat is er bij de velden een eigenschap gevoegd die aangeeft welke taal de data heeft. Dit is met name van belang voor akten uit de Franse tijd. Verder blijkt uit het opmerkingenveld in de Digitale Stamboom dat er soms behoefte is om een opmerking bij een transcriptie te maken, bijvoorbeeld dat iets slecht leesbaar is. Hiervoor is er een eigenschap opmerking bij de transcriptie velden gemaakt.

De volledige beschrijving van het datamodel is te vinden in de bijlage.

6. Dataformaat naar data en database

Database

Om de data op te slaan is een SQL Server database gebruikt. De data bestaat uit twee delen. Het eerste deel is de configuratie data van het uitwisselingssysteem, zoals welke repositories geharvest moeten worden en hoe vaak. Het tweede gedeelte is de akte data. Dit is het gedeelte dat niet triviaal is en daarom wat uitgebreider beschreven zal worden.

De tabellen in de database volgen ongeveer de structuur van de xml. Dat maakt het makkelijker om data in XML-formaat uit de database te halen en om de data in XML-formaat in de database op te slaan.

Op een aantal punten wordt er afgeweken van het xml-formaat. De belangrijkste afwijkingen zijn de opmerkingen en de taalattributen die bij veel elementen in de XML voorkomen. Deze hebben een eigen tabel met een verwijzing naar de tabel, rij en kolom. Voor de opmerkingen is een soortgelijke constructie toegepast.

Achteraf gezien is het relationeel opslaan van de xml data misschien niet de beste oplossing. Het originele idee dat men in een relationele representatie eenvoudiger databewerkingen kan doen komt slecht tot haar recht omdat de statements voor deze bewerkingen vaak een join over 3 of meer tabellen betekent. Het komt de eenvoud van het systeem ten goede om de xml als xml in de database op te slaan of om een NoSQL[14] oplossing toe te passen.

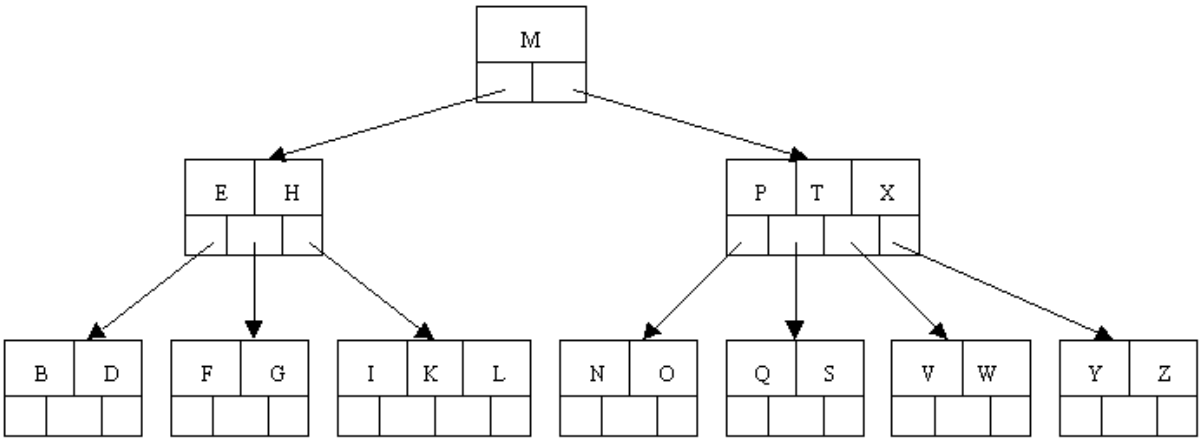
Indexen

Om de zoeksnelheid van de records te verhogen staan er geclusterde indices op elke tabel. Deze indices staan telkens op de velden die het meest worden gebruikt om het record te benaderen. Zoals bijvoorbeeld op fk_record in tblEvent of op een combinatie van TableName fk_row en FieldName voor tblTransRemark.

[14] NoSQL entry wikipedia http://en.wikipedia.org/wiki/NoSQL [gezien 3 mei 2011]

B-Tree

Geclusterde indexen in SQL Server zijn geïmplementeerd als een B-Tree[15].



Figuur 14 B-Tree[16]

Een B-Tree is een datastructuur die geoptimaliseerd is voor het snel zoeken en ophalen van data. Een van de datavelden is aangewezen als sleutel. Een node in een B-Tree bevat een of meerdere sleutels. Dat zijn de letters in figuur 14. Elke node, behalve de bladeren, heeft kinderen. Het aantal kinderen is altijd één meer dan het aantal sleutels dat de node bevat. Het eerste kind bevat sleutels die kleiner zijn dan de eerste sleutel, het tweede kind bevat sleutels die kleiner zijn dan de tweede sleutel maar niet kleiner zijn dan de eerste enzovoort. In de bladeren van de B-Tree bevinden zich ten slotte de echte records. Het idee achter het gebruik van een B-Tree is om snel te kunnen zoeken in een datastructuur die zich (gedeeltelijk) op de harde schijf bevind. Door de boom zich sterk te laten vertakken neemt de diepte van de boom af en daarmee het aantal keren dat we moeten lezen van de harde schijf.

Data omzetting

De database moet initieel en eventueel als aanvulling op de bestaande data inhoud krijgen. Voor dit project is er voor data van het oud notarieel archief van Delft en voor data van de doopboeken van het Regionaal Archief Leiden een mapping gemaakt van het originele digitale formaat van de akte naar de A2A Database. De omzetting vindt plaats met behulp van stored procedures, een routine in de database.

De data uit de doopboeken bestaat uit twee gebeurtenissen, namelijk een geboorte en een doop. Het aantal personen dat op een doop akte staat is beperkt tot een dopeling, een vader een moeder en 5 getuigen. Al deze data staan in de bron database in een enkel record. De data van dat record kan dus in één keer worden ingelezen en in de A2A database worden opgeslagen.

De data uit het notarieel archief is dynamischer dan de data van doopboeken. Het aantal personen dat op een dergelijke akte voor kan komen is in principe onbeperkt. In de bron database staat de data over

[15] Clustered Index Structures http://msdn.microsoft.com/en-us/library/ms177443.aspx [gezien 6 september 2011]

[16] Software Design Using C++ http://cis.stvincent.edu/carlsond/swdesign/btree/btree.html [gezien 6 september 2011]

de personen dan ook in een aparte tabel. Voor het omzetten van de data betekent dit dat de stored procedure door de collectie van personen bij een akte moet lopen en per record een nieuw persoon aan moet maken in de A2A database.

Overige stored procedures

Voor het uitvoeren van diverse onderhoudstaken op de database, zoals het importeren van de externe data, het vullen van de standaardwaarden van tabellen en het resetten van de inhoud van de database, is er een aantal stored procedures ontwikkeld. Een en ander is te vinden in de bijlagen.

7. Protocollen en beveiliging

Beveiliging

Vanwege de grote hoeveelheid data en de zoekbelasting die webservices van de Digitale Stamboom op de verbindingen en de servers van de aanbieder geven, heeft de Digitale Stamboom de wens geuit een toegangsbeleid te kunnen voeren op het gebruik van webservices.

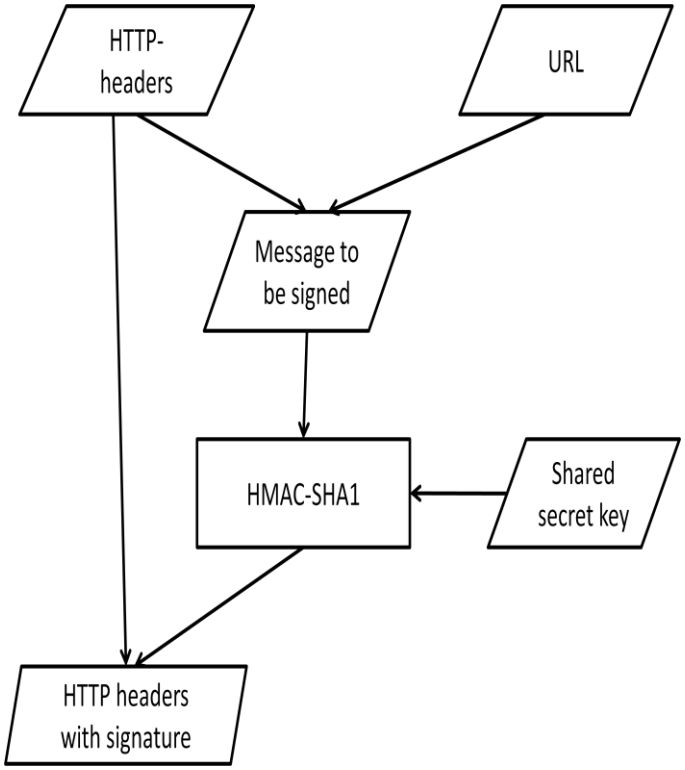
De Digitale Stamboom heeft op dit moment al enkele webservices in gebruik. De belangrijkste zijn de zoekwebservices. Eén van deze webservices is origineel ontwikkeld voor archiefplein dat oorspronkelijk verschillende archief platformen gelijktijdig doorzocht. Recenter is er met de komst van de centrale zoekingang van de Digitale Stamboom een nieuwe, in functionaliteit vergelijkbare, zoekwebservice gebouwd. Al het zoekwerk bij het gebruik van deze services wordt door de aanbieder gedaan en belast de infrastructuur van de aanbieder. Voor de nieuwe webservice betekent dit ongeveer 30.000 zoekvragen per maand. Deze zoekvragen zijn min of meer legitiem. Maar deze webservice is slechts bij enkele partijen bekend. Het is een slecht beveiligingsprincipe om te rekenen op de onbekendheid van een service. Naast de serverbelasting is het ook voorstelbaar dat een instelling niet zondermeer de inhoud van haar complete database wil weggeven.

Om de instellingen op een veilige manier te laten bepalen met wie ze de data willen delen, is er voor dit project een OAuth authenticatie systeem geïmplementeerd met de intentie om dit systeem ook voor andere webservices binnen de Digitale Stamboom te gebruiken.

OAuth

Om de toegang tot webservices te reguleren is OAuth[17] ontwikkeld. OAuth is de open standaard voor communicatie tussen webservices namens een gebruiker. Het aanvragen van autorisatie namens een gebruiker wordt ook wel 3-legged OAuth genoemd. Een variant hierop is het 2-legged OAuth. Bij 2-legged OAuth is er alleen sprake van autorisatie tussen webservices. Omdat wij niet namens een gebruiker acties uitvoeren, de archiefbewaarplaatsen kennen in die zin geen geregistreerde gebruikers, gebruiken wij 2-legged OAuth.

Bij 2-legged OAuth worden aan de hand van het URL en de http-headers van een request samen met een geheime sleutel die de



Figuur 15 Schematische weergaven van het ondertekenen van een HTTP-bericht

[17] OAuth http://tools.ietf.org/html/rfc5849 [gezien 20 april 2011]

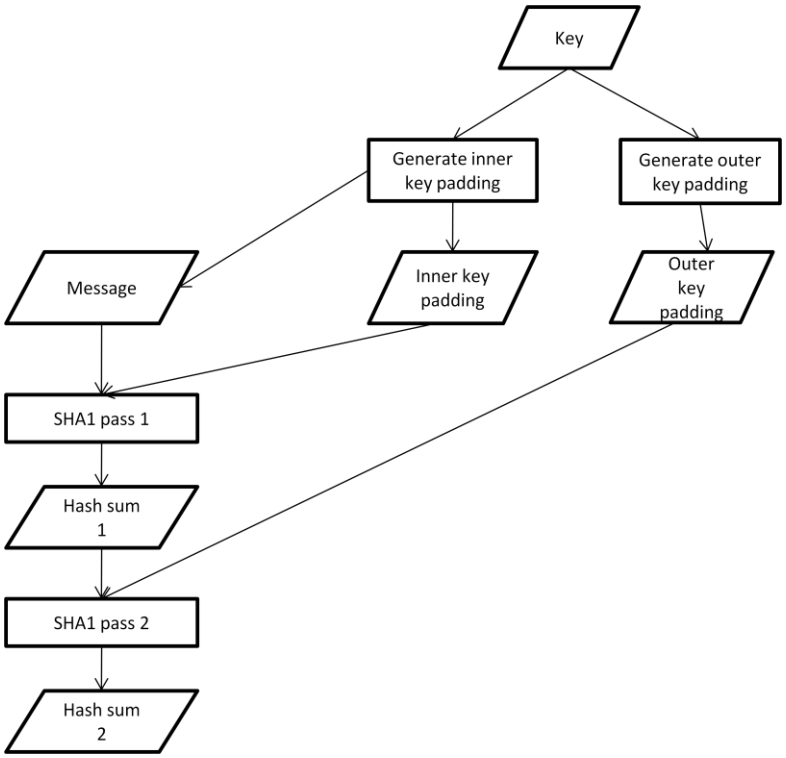
aanbieder en de afnemer delen een signatuur gegenereerd. Dat laatste met behulp van HMAC-SHA1. Deze signatuur wordt toegevoegd als header van hat request en gezamenlijk opgestuurd naar de aanbieder. Voordat de aanbieder het request in behandeling neemt controleert die eerst de handtekening. Het ondertekenen gebeurt met HMAC-SHA1[18]

Als de handtekening in orde is controleert de aanbieder ook nog eens aan de hand van een flag in de database of de gebruiker geautoriseerd is.

De geheime sleutel die de zender en ontvanger delen komt tot stand met behulp van het Diffie-Hellman key exchange protocol. Dit protocol laat de beide partijen op een veilige manier een sleutel afspreken over een onveilig kanaal.

HMAC

Om het bericht te ondertekenen word Hash-based Message Authenticatie[19] (HMAC) gebruikt. HMAC is een veel gebruikte manier om berichten aan de hand van een geheime sleutel te ondertekenen. De invoer voor HMAC is de sleutel en het bericht. Eerst genereert HMAC aan de hand van de sleutel twee 64 byte paddings, de inner en outer key padding. Vervolgens word het bericht geprefixt met de inner key padding en wordt dit geheel door de hash functie gehashed met als resultaat de Hash sum 1. In de volgende stap prefixen we Hash sum 1 met de outer key padding. Deze nieuwe string wordt ook door de hash functie verwerkt. Het resultaat van deze laatste bewerking, hash sum 2 is het resultaat van HMAC en geldt als de signatuur.



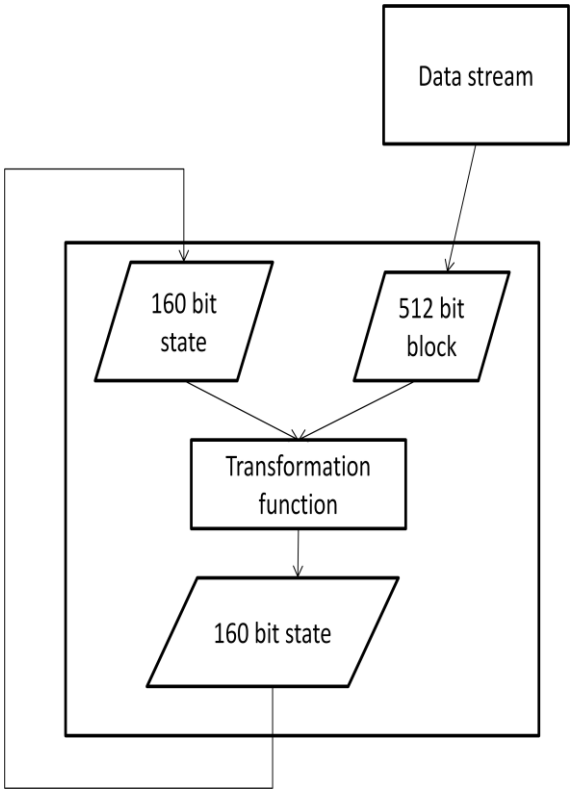
Figuur 16 Schematische weergave van HMAC

[18] Wikipedia SHA http://en.wikipedia.org/wiki/SHA-1 [gezien 16 februari 2010]
[19] HMAC: Keyed-Hashing for Message Authentication http://www.faqs.org/rfcs/rfc2104.html [gezien 7 september 2011]

SHA1

Secure Hash Algorithm of SHA[20] is een algoritme dat een datastring met een lengte tussen de 0 en $2^{64} - 1$ mapped naar een string van 160 bits. Belangrijke eigenschappen van een goede hash functie voor cryptologisch gebruik zijn, dat je aan de hand van een gegeven hash niet eenvoudig het originele bericht kan reproduceren. En dat je, gegeven een origineel bericht en de bijbehorende hash, niet eenvoudig het originele bericht zo kan aan passen zonder dat de hash zou veranderden. Het hash algoritme werkt op grote lijnen als volgt. Eerst wordt het bericht bewerkt zodat het een lengte heeft dat een meervoud is van 512 bits. Dit gebeurt door het bericht aan te vullen met een waar bit, en vervolgens met nul bits totdat de lengte van het bericht modulo 512 448 is. Vervolgens worden de laatste 64 bits gevuld met de lengte van het bericht.

SHA1 heeft een interne staat, die met een constante wordt geïnitialiseerd en een transformatie functie. De transformatie functie werkt op de interne staat en een Block van de data. Het resultaat van de transformatie functie is een nieuwe interne staat voor het algoritme. Met deze nieuwe interne staat wordt het volgende block data opgepakt en verwerkt door het algoritme. Aan het einde van de input stream geldt de laatste interne staat van het algoritme als de hash waarde.



Figuur 17 Schema van de werking van SHA1

De kwaliteit van HMAC-SHA1

In 2005 werden collision attacks op SHA1 bekend met een complexiteit van minder dan de 2^{80} voor een perfecte hash functie. Inmiddels lijkt het mogelijk om deze aanvallen met een complexiteit van 2^{51}uit te voeren[21]. HMAC-SHA1 heeft van deze aanvallen echter geen last omdat de hidden key en dus het eerste gedeelte van de string die gehashed is onbekend is. Dus zelfs als we een collision vinden met een fantasie key kunnen we er nog steeds niets mee.

Diffie-Hellman

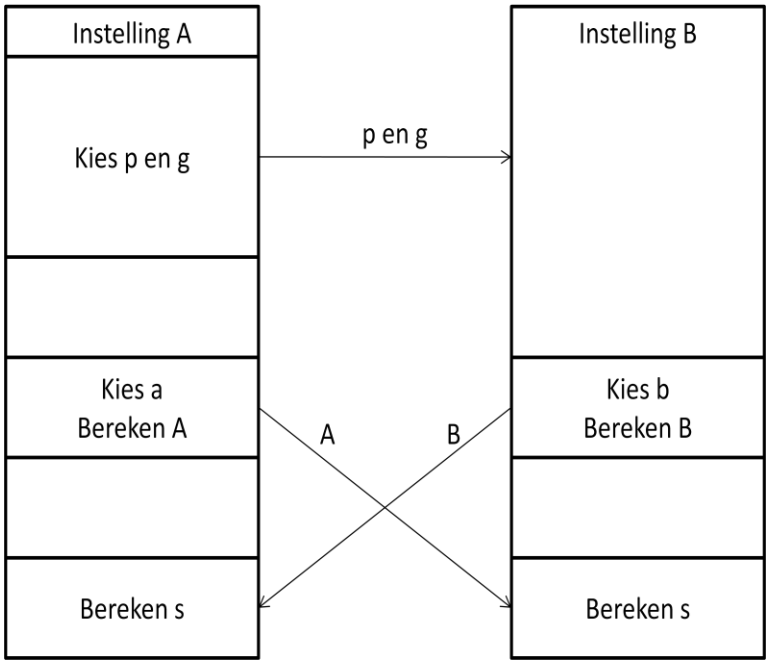
Om aan de harvester kant te ondertekenen en aan de repository kant de handtekening te controleren moeten de Harvester en het repository een sleutel hebben die alleen zij kennen. Met behulp van het Diffie-Hellman key exchange protocol[22] kan deze sleutel tot stand komen over een onbeveiligd

[20] US Secure Hash Algorithm 1 (SHA1) http://tools.ietf.org/html/rfc3174 [gezien 7 september 2011]

[21] Stéphane Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1 http://eprint.iacr.org/2008/469.pdf [gezien 30 augustus 2011]

[22] Diffie–Hellman key exchange

communicatiekanaal. Het Diffie-Hellman key exchange protocol werkt als volgt: instelling A kiest een priem getal p en een generator g kleiner dan p met de eigenschap dat p een primitieve wortel modulo p is. Instelling A maakt deze nummers openbaar door ze naar instelling B te sturen. Vervolgens kiezen instelling A en instelling B elk een geheim nummer respectievelijk genaamd a en b en vervolgens berekenen ze A en B waar $A = g^a \, mod \, p$ en $B = g^b mod \, p$. De instellingen communiceren A en B met elkaar. Vervolgens berekenen de beide instellingen de geheime sleutel door en door $s = A^b mod \, p$. Hoewel de gegevens p, g, $g^a \, mod \, p$ en $g^b mod \, p$ openbaar zijn is de waarde van s niet te achterhalen.

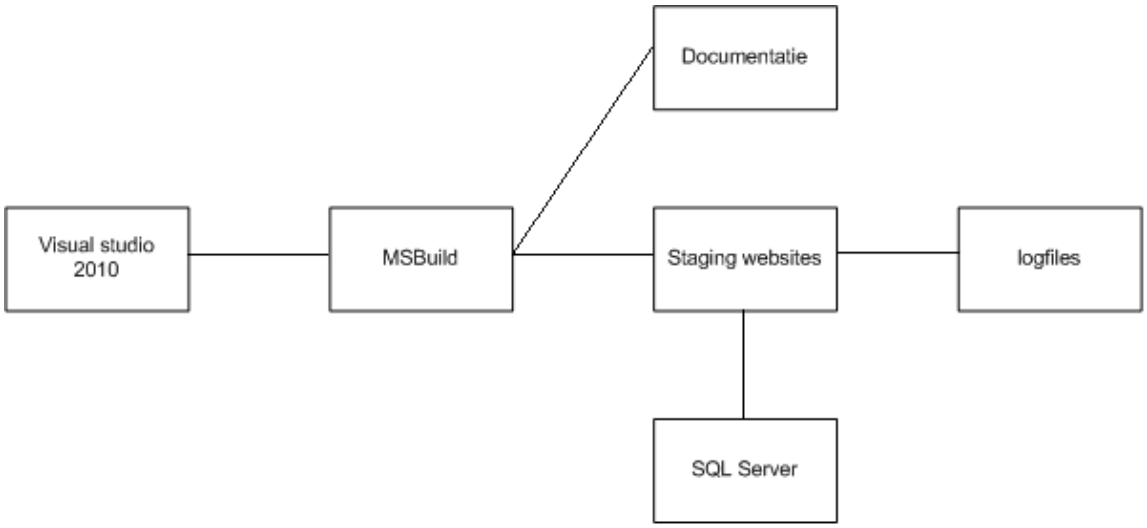


Figuur 18 Schema van Diffie-Hellman key exchange protocol

http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange [gezien 16 februari 2010]

8. Tools

Ontwikkeltools



Figuur 19 Ontwikkel straat

Het project is geschreven in C#. Het gebruikte .Net framework heeft versie 3.5. Voor de database is SQL Server 2005 standard edition gebruikt. Het schrijven van de code is gedaan in Visual Studio. De code is opgedeeld in 3 projecten(repository, harvester en administratie). Met behulp van een MSBuild[23] script kan deze code gecompileerd worden en gedeployed naar een staging server(een test omgeving). Het MSBuild script zorgt er ook voor dat aan de hand van de code met behulp van sandcastle technische documentatie wordt gegenereerd. De staging websites maken gebruik van een SQL Server database en zijn daarmee verbonden met NHibernate[24]. Om te debuggen worden er logfiles weggeschreven met behulp van log4net. De SQL code wordt niet door het MSBuild script aangepast.

Gebruikte libraries

In het project worden een aantal externe libraries gebruikt. Deze worden hier kort besproken.

nHibernate

Hibernate is een service die de communicatie tussen de business logica en de datalayer ondersteund. Het doet dit door het ondersteunen van persistente objecten en door meerder query languages te ondersteunen. Hibernate kan met diverse typen databases overweg. Zo kan men met een minimale inspanning de applicatie geschikt maken voor bijvoorbeeld MySQL of Access. Voor .Net is speciaal nHibernate ontwikkeld.

---

[23] MSBuild Reference http://msdn.microsoft.com/en-us/library/0k6kkbsd(v=VS.90).aspx [gezien 3 mei 2011]
[24] nHibernate versie 2.1.0 is gebruikt en deze is te vinden op https://www.hibernate.org/343.html [gezien 3 mei 2011]

OAuth

Voor het tekenen en controleren van de tekening met OAuth[25] is de OAuth library van Apache gebruikt.

In OauthBase.cs zijn enkele kleine wijzigingen gemaakt.

Diffie-Hellman

Voor het Diffie-Hellman[26] key exchange protocol wordt een library van mentalis gebruikt.

Loggen

Voor het loggen van de gebeurtenissen wordt gebruik gemaakt van log4net[27]. Log4net is een tool van apache die de log berichten in een tekst bestand op de webserver schrijft. En het is hetzelfde pakket dat ook door nHibernate wordt gebruikt.

---

[25] OAuth http://code.google.com/p/oauth/ [gezien 3 mei 2011]

[26] Diffie-Hellman http://www.mentalis.org/soft/class.qpx?id=15 [gezien 3 mei 2011]

[27] log4net http://logging.apache.org/log4net/ [gezien 3 mei 2011]

9. Slot

Samenvatting

In deze scriptie hebben we uitgebreid gekeken naar het ontwerp en de implementatie van een systeem voor het uitwisselen van historische persoonsgegevens. Eerst is er gekeken naar de opbouw van het systeem uit een repository, een harvester en een administratie module. Verder is er beschreven welke functies deze onderdelen vervullen. In hoofdstuk 3 hebben we gekeken hoe we tot deze opbouw zijn gekomen door te beschouwen hoe het systeem in de bestaande infrastructuur in de erfgoedwereld past. Het OAI-PMH protocol voor uitwisseling en hoe het toegepast kan worden is besproken. De opbouw van het systeem in een datalaag, een data access laag, een logica laag en een interface laag is uitgelegd.

In hoofdstuk 4 hebben we stil gestaan bij diverse bestaande dataformaten en waarom deze niet geschikt zijn voor onze applicatie. In hoofdstuk 5 hebben we beschreven wat we precies willen van een dataformaat. Dat is gedaan door onder andere te kijken naar wie het dataformaat gaan gebruiken en aan welke eisen het data formaat van uit dat oogpunt moet voldoen. Ook hebben we een aantal principes vastgesteld die als leidraad dienen voor de uitwerking van het datamodel. Er is een korte beschrijving gegeven van het dataformaat zelf.

In hoofdstuk 6 hebben we het gehad over hoe het XML-formaat in een database wordt opgeslagen en kort besproken hoe daar ook andere afwegingen kunnen worden gemaakt.

In hoofdstuk 7 zijn we dieper ingegaan op de beveiliging van het systeem. Onder andere OAuth, HMAC, SHA1 en Diffie-Hellman komen aan bod. En ook hoe deze het systeem, ondanks de zwaktes in SHA1, veilig maken.

In het laatste hoofdstuk hebben we het gehad over de infrastructuur die gebruikt is voor de ontwikkeling van dit systeem. We zijn we met name ingegaan op de diverse tools en libraries die zijn gebruikt.

Conclusie

Dit project was een leerzame ervaring. Het maken van een xsd definitie is een tegenvallend groot project. Temeer omdat ik tijd heb moeten investeren om de xsd taal onder de knie te krijgen. Op het dataformaat heb ik de feedback gehad van Martijn Spruit van het CBG en van Bennie Blom van het gemeente archief Delft. Door het dataformaat te testen met de voorbeelden (bijlage Voorbeelden Dataformaat) heb ik laten zien dat het bruikbaar is voor veel verschillende bronnen. In aangepaste vorm wordt het datamodel toegepast in WieWasWie en in de open source versie van de Digitale Stamboom. Het uitwisselsysteem wordt zonder autorisatie en met een ander datamodel gebruikt voor de huidige Digitale Stamboom en zal met autorisatie en het A2A model worden gebruikt voor de open source versie.

Het datamodel voldoet aan de gestelde eisen en is gezien het bovenstaande goed toepasbaar in applicaties buiten het uitwisselsysteem.

Het uitwisselsysteem zelf werkt in de basis. Vooral de administratie functionaliteit kan nog verder ontwikkeld worden. Het is jammer dat het autorisatie systeem nog niet in gebruik is. Ik ben erg benieuwd naar hoe dat ervaren wordt.

De complexiteit van de constructie van de software was onverwacht groot. Onder andere doordat er tijd geïnvesteerd is in het leren van het gebruik van diverse tools en technieken zoals nHibernate, xsd en MSBuild. Het hele project heeft ongeveer 8 maanden tijd gekost.

Discussie

OAI-PMH

Het systeem zelf heeft als basis het OAI-PMH protocol. Dit voldoet aan de wensen die we in dit project hadden. Allicht worden er in de toekomst uitwisselingsmodellen ontwikkeld die beter zijn. Wat mij betreft zou een uitwisselingsmodel zonder de eis om de Dublin-Core standaard te ondersteunen de voorkeur genieten. Het uitwisselingsmodel in haar eenvoud zou zo meer waarde krijgen doordat het algemener kan worden ingezet.

Het datamodel

Het datamodel zoals in deze scriptie beschreven is zeker in de toepassing van dit datamodel binnen WieWasWie, op een aantal punten bekritiseert en aangepast. Het probleem met het huidige datamodel waarin we personen en gebeurtenissen kennen en ook de link tussen deze personen en gebeurtenissen, zijn de personen die genoemd worden op een akte maar geen relatie hebben tot de gebeurtenissen. Het gaat hierbij om bijvoorbeeld een gebeurtenis waarbij één van de partijen weduwe is van iemand anders. Deze melding heeft een grote betekenis in een genealogisch onderzoek maar dient op een akte feitelijk alleen ter identificatie van een partij. We geven drie voorbeelden van hoe een persoon zonder relatie op een akte kan voorkomen

Voorbeeld 1

Het huwelijk van Marie Jansen, weduwe van Piet Smit, is anders dan het huwelijk van Marie Jansen, weduwe van Harm de Boer. Terwijl Piet Smit bij de gebeurtenis, het nieuwe huwelijk van Marie, geen functie vervult.

Voorbeeld 2

Op de akte van het burgerlijk huwelijk van Marie, zal haar vader altijd genoemd worden. Dit laatste ongeacht dat hij aanwezig was bij het huwelijk of niet. Als de vader echter aanwezig was bij het huwelijk zou hij normaal gesproken ook als getuige optreden en dus een rol spelen bij de gebeurtenis. Als de vader echter niet aanwezig is bij het huwelijk wordt deze enkel genoemd als "eigenschap" van Marie.

Voorbeeld 3

De aangifte van het overlijden van Jan wordt door Piet gedaan. Piet heeft daarom de rol van aangever op de akte. Op een akte wordt vaak ook geregistreerd wat de relatie tussen Jan en Piet was. Het is voor de genealogische onderzoeker veel interessanter te weten wat de relatie is tussen Jan en Piet dan het gegeven dat Piet aangever is. Is Piet de vader van Jan of is Piet de veldwachter die Jan vond.

De relaties uit deze voorbeelden worden in het datamodel op verschillende manieren weergegeven. De gegevens in het eerste voorbeeld worden opgeslagen als een eigenschap van Marie. Het verschil tussen of de vader in voorbeeld twee aanwezig was of niet wordt genegeerd in het datamodel. Het uitgangspunt is dat de vader altijd de rol vader heeft bij de gebeurtenis. Ongeacht of deze aanwezig was. De relatie tussen de aangever en de overledene in voorbeeld 3 wordt opgeslagen als een extra veld in de relatie van de aangever met de gebeurtenis.

Deze oplossing in het datamodel is niet echt fraai te noemen. Met betrekking tot de toepassing van het datamodel in WieWasWie was er twijfel of deze drie manieren genoeg zijn om alle voorkomende gevallen af te dekken. Hoewel een voorbeeld van een geval dat niet afgedekt wordt op de bovenstaande manieren is besloten om het datamodel uit te breiden met een vierde manier om dit te ondervangen: namelijk een persoon-persoon relatie. Het toepassen van deze constructie betekent dat niet alle personen(in de zin van XML-objecten) een relatie hoeven te hebben met een gebeurtenis. Dat betekent dat een gebeurtenis via een of meer tussenpersonen gerelateerd kan zijn aan een persoon. Daardoor wordt de structuur van de data minder eenvoudig. Een tweede bezwaar is dat het geschetste probleem niet is opgelost, we hebben nu feitelijk 4 manieren om personen die geen rol spelen bij een gebeurtenis te beschrijven.

Ik zie meer in een wat minder scherpe definitie van een rol hebben bij een gebeurtenis. Zo zou Piet Smit in ons eerste voorbeeld bij het huwelijk van Marie de rol van vorige partner kunnen spelen, ondanks dat hij geen feitelijk rol heeft bij de gebeurtenis zelf. Het beschrijven van een vorige partner als eigenschap van een persoon kan dan verdwijnen. Het probleem van voorbeeld twee bestaat binnen een ruimere definitie van rol ook niet meer. Deze aanpassing zou het datamodel eenvoudiger en eenduidiger maken.

Naast de persoon-persoon relatie zijn er een groot aantal andere toevoegingen aan het datamodel gedaan. Deze toevoegingen zijn een object-object, een relatie tussen objecten, tussen objecten en gebeurtenissen, tussen objecten en personen en tussen objecten en de bron. Deze toevoegingen, vijf in totaal, wanneer de toevoeging van persoon-persoon relaties worden meegerekend, representeren het halve dataformaat. Deze objecten beslaan echter een zeer klein gedeelte van de totale verzameling aan historische persoonsgegevens. Het idee achter een object is om bijvoorbeeld bij een akte van verkoop ook het verkochte object te registreren. Dit maakt echter geen impact op de data die getranscribeerd moet worden. De objecten kunnen immers ook als een eigenschap van een gebeurtenis dan wel een persoon worden beschreven.

10 Bijlagen

Xsd schema
Documentatie xsd schema
Voorbeelden Dataformaat
Admin code
Harvester code
Aanbieder code
Admin documentatie
Harvester documentatie
Aanbieder documentatie
Database generation script
Database stored procedures
Handleiding A2A