Web API’s

Wat zijn de technische- en commerciële waarden van web API's voor producenten en gebruikers?

Frank Roor

Hilversum, 2007

Web API's

Wat zijn de technische- en commerciële waarden van web API's voor producenten en gebruikers?

Afstudeerscriptie Mediatechnologie

Hogeschool Utrecht

Faculteit Natuur en Techniek

Hilversum, 2007

Begeleiders
Erik Zoomers – Gather B.V. / Internetbureau Websight
Niels Floor – Hogeschool Utrecht

Auteur
Frank Roor
Student Mediatechnologie 2003 – 2007
1182700
frank@frankroor.nl

Voorwoord

Voor u ligt mijn afstudeerscriptie ter afsluiting van de opleiding Mediatechnologie aan de Hogeschool Utrecht. Mijn afstudeerstage heb ik uitgevoerd in de periode februari 2007 t/m juni 2007 bij het bedrijf Gather te Hilversum. Deze scriptie is het resultaat van vier jaar leren tijdens- en rondom mijn HBO studie.

De vier jaar die ik heb doorgebracht op de opleiding Mediatechnologie heb ik als zeer leuk ervaren. De eerste twee jaar waren erg theoretisch met afwisselend praktijk. Dit heeft een goede basis gelegd. Met de medestudenten die ik tijdens deze twee jaar heb leren kennen heb ik veel gezelligheid meegemaakt. Vervolgens heb ik in het derde jaar stage gelopen bij internetbureau Websight te Hilversum. Hier heb ik veel praktische kennis opgedaan en een leuke tijd gehad. Het semester hierna ben ik terechtgekomen bij het keuzesemester Broadcast & Crossmedia technology. Dit half jaar heb ik veel theoretische- en praktische kennis op het gebied van video, audio en crossmedia op kunnen doen. Dit kon ik vervolgens in het semester hierna toepassen bij het maken van de documentaire voor de nieuwe opleiding Security Technology. Na de uitstap naar audio en video wilde ik me tijdens mijn afstudeerstage weer wenden tot Internet. Een fantastisch medium. In mijn ogen hét toekomstige medium waar alles op gaat gebeuren. Ik kwam min of meer weer terug op het oude nest bij Websight, namelijk bij een dochteronderneming, Gather.

Bij deze scriptie zijn een aantal personen direct of indirect betrokken die ik graag wil bedanken. Allereerst Erik Zoomers van Gather: bedankt voor de leuke tijd, de goede leeromgeving en de kansen die je me geboden hebt. Daarnaast alle andere collega's van internetbureau Websight en Gather voor het delen van hun kennis en de gezelligheid (op vrijdagmiddag). Ook alle betrokkenen afkomstig van de opleiding wil ik graag bedanken (docenten en studenten): Ivar van der Burg, Robert van Geerenstein, Bertran Zwijnenburg, Peter van den Berg, Niels Floor en Bernard Weerdmeester. En niet te vergeten, mijn familieleden, allen bedankt!

Tot slot de opties voor na mijn studie. Werken of verder studeren? Verder studeren. Wat de vervolg studie gaat worden is op het moment van schrijven nog onduidelijk. Wat ik wél zeker weet, is dat ik een bewuste keuze kan maken met alle opgedane kennis uit mijn HBO studie.

Hilversum, 29 mei 2007

Frank Roor

Inhoudsopgave

Samenvatting

Web 2.0 wordt gezien als de grote verandering van het Internet. Dhr. O'Reilly is één van de grondleggers van de term. Web 2.0 geeft aan dat het Internet een platform wordt waarin losstaande webapplicaties gemakkelijk aan elkaar gekoppeld kunnen worden. Het koppelen wordt gerealiseerd door het beschikbaar stellen van zogenoemde web API's. In deze web API's staat beschreven hoe een webapplicatie door andere partijen benaderd kan worden.

Het specifieke onderdeel web API's wordt in deze scriptie onderzocht. Daarbij komt de volgende hoofdvraag aan bod: *Wat zijn de technische- en commerciële waarden van web API 's voor producenten en gebruikers?*

Het onderzoek wordt uitgevoerd bij het bedrijf Gather te Hilversum. In het kader van het onderzoek is het project Staatopdekaart.nl gestart. Dit project dient vanuit de praktijk inzicht te geven ten behoeve van het onderzoek. Een web API die voornamelijk behandeld wordt is de Google Maps API. Deze API maakt het mogelijk kaart- en satelliet beelden binnen een eigen applicatie te integreren.

Omdat web API's een belangrijke rol spelen binnen web 2.0 wordt allereerst deze term uitgelegd. De belangrijkste eigenschappen van web 2.0 zijn:

- Het Internet wordt één platform, waar verschillende diensten en applicaties aan elkaar gekoppeld kunnen worden;
- Het Internet krijgt een socialer karakter;
- De gebruiker levert een belangrijke bijdrage aan een web 2.0 dienst (User Generatred Content). De dienst moet de gebruikersgroep vertrouwen.
- De gebruikservaring van web 2.0 diensten is enorm, door inventieve interfaces en het gebruik van de techniek AJAX.
- Ontwikkelmethoden veranderen, de gebruiker wordt hierin volledig betrokken;
- Het vindbaar maken van informatie wordt mogelijk gemaakt doordat gebruikers trefwoorden (tags) kunnen toevoegen.

Web 2.0 heeft echter ook een aantal schaduwkanten. Critici zien veel overeenkomsten met de Internet hype rondom het jaar 2000. In dit jaar investeerden veel investeerders in startende Internetbedrijven. Deze bedrijven bleken achteraf geen goed verdien model te hebben en een groot deel ging uiteindelijk verloren. Tevens beweren critici, kijkende naar de gebruikte technieken, dat web 2.0 helemaal niet zo nieuw is als dat het lijkt.

Het belangrijke onderdeel van web 2.0 'het web als platform', wordt mogelijk gemaakt door API's. Web API's kunnen opgezet worden aan de hand van drie verschillende stijlen: Service Oriented Architecture, Remote Procedure Calls en Representational State Transfer. Allen maken gebruik van de opmaaktaal XML. Hierin kan data platform onafhankelijk uitgewisseld worden. Voor de uitwisseling tussen verschillende platformen via XML bestaan een aantal standaarden, waaronder: SOAP, WSDL, UDDI, XML-RPC en Javascript API's. Deze standaarden vormen een belangrijk onderdeel van veel web API's. Om een beeld te krijgen van de praktijk: de API's van Google Maps, Flickr en Youtube maken veelvuldig gebruik van Javascript, SOAP, REST en XML-RPC.

Verdere ervaringen uit de praktijk kunnen omschreven worden aan de hand van het project Staatopdekaart.nl. Deze applicatie maakt gebruik van de Google Maps API en toont verschillende locaties op een kaart. In deze scriptie komt aan bod hoe gebruik gemaakt wordt van de Google Maps API en hoe het systeem ingericht is.

Vervolgens wordt gekeken naar de commerciële aspecten. Voor API producenten of aanbieders zijn de volgende mogelijkheden: het aanbieden van een betaalde web API, de aangeboden API zorgt voor een goede merknaam, er kunnen inkomsten gegenereerd worden uit advertenties. Een voorbeeld uit de praktijk is Amazon, waarvan 28% van de omzet afkomstig is uit hun API. Voor de gebruikers van API's zijn ook commerciële mogelijkheden. Het gebruik van een API kan ingezet worden als marketingtool of een eigen dienst kan goedkoop verrijkt worden met complexe functionaliteiten. Daarnaast is er de mogelijkheid om te adverteren of kan de kennis van API's aangeboden worden als dienst, als API consultant.

Voor de producenten van API’s kan geconcludeerd worden dat het aanbieden van een web API technisch voordelen biedt door de platform onafhankelijkheid en de schaalbaarheid van een systeem. Ook biedt het ontwikkelaars meer gemak bij het koppelen van bepaalde componenten binnen een webapplicatie. Deze componenten kunnen ook gemakkelijk beschikbaar worden gesteld aan derden. Dit feit brengt ook commerciële voordelen met zich mee. Zo kan een eigen dienst populairder worden en kunnen inkomsten gegenereerd worden uit het aanbieden van commerciële API's. Op het gebied van bedrijfsprocessen kunnen API's meer efficiëntie en schaalbaarheid opleveren.

Voor de gebruikers van web API’s kan geconcludeerd worden dat het gebruik ervan vooral ‘leuk’ en gemakkelijk is. Technisch gezien kan er gemakkelijk, zonder veel ontwikkeltijd, gebruik gemaakt worden van ingewikkelde toepassingen zonder daar kennis van te hebben. Commercieel gezien kan op een goedkope manier data verkregen worden die voorheen niet toegankelijk was qua prijs. Daarnaast kan door middel van een advertentie model inkomsten gegenereerd worden of kunnen diensten verkocht worden als API consultant. De commerciële mogelijkheden blijken echter nog vrij beperkt.

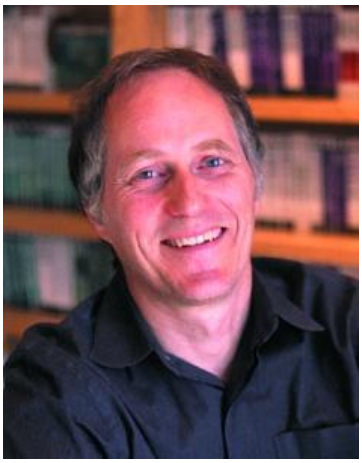
Inleiding

2 februari 2007, Yahoo! start een nieuwe service, genaamd Yahoo! Pipes. *"Yahoo!'s new Pipes service is a milestone in the history of the internet. It's a service that generalizes the idea of the mashup[1], providing a drag and drop editor that allows you to connect internet data sources, process them, and redirect the output.",* meldt dhr. O'Reilly op zijn weblog. Yahoo! Pipes maakt het mogelijk om diverse diensten van verschillende websites met elkaar te combineren, genaamd een 'Pipe'. Deze combinatie biedt de uiteindelijke informatie aan in een RSS feed. Volgens O'Reilly is deze dienst een mijlpaal in de geschiedenis van het Internet. Waar heeft deze man het over?

Toen omstreeks 2001 de Internet 'bubble' uiteenspatte gingen veel Internetbedrijven verloren. De jaren hiervoor kende het Internet een grote opmars en was er veel financieel gewin. Vele durfkapitalisten investeerden in Internet en de koersen explodeerden. Internet was 'hot'. Het was immers grandioos dat een relatief groot publiek op een zeer goedkope en interactieve manier benaderd kon worden. Achteraf gezien was het een hype, die historisch geplaatst kan worden in de rij van andere hypes rondom nieuwe technologieën. De beoogde winst werd niet behaald. De 'bubble' spatte uiteen.

Na deze periode groeide Internet weer gestaag. In 2004 werd de term web 2.0 geïntroduceerd. De eerder genoemde dhr. O'Reilly omschreef de toenmalige ontwikkelingen als versie 2 van het Internet, zoals dat ook bij software ontwikkeling wordt gedaan. Dit gebruikte hij bij conferenties en in artikelen. De term werd al snel opgepikt door technici en marketeers. De term web 2.0 impliceert dat er ook een eerste fase van Internet was. Het is lastig om harde periodes hieraan te verbinden, maar over het algemeen wordt de Internet bubble rondom 2000 gezien als 1.0 en de ontwikkelingen vanaf 2004 tot heden als 2.0. Bij de nieuwe ontwikkelingen vallen de termen 'social networking', 'wiki's', 'blogs', 'rich internet applications' en 'folksonomies' op. Centraal staan de sociale aspecten en het genereren van content door gebruikers binnen een 'desktop lijkende omgeving[2]'. O'Reilly wordt gezien als de grondlegger van de term web 2.0. Zelf omschrijft hij web 2.0 op de volgende manier: "*Web 2.0 is the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them.*" Binnen zijn benadering wordt onderscheid gemaakt tussen technische- en functionele aspecten. In deze scriptie zal hier in een apart hoofdstuk dieper op ingegaan worden.

Eén van de aspecten bij de benadering van O'Reilly is het toepassen of beschikbaar stellen van een web API[3]. Binnen de ontwikkelingen op Internet zie je steeds meer diensten ontstaan met web 2.0 kenmerken. Deze diensten worden vaak beschikbaar gesteld door middel van een API. Hierdoor kan een dienst door iemand anders gebruikt worden binnen een eigen dienst of website. De wijze waarop deze dienst aangesproken kan worden staat beschreven in een API. Veel diensten worden aan elkaar gekoppeld, wat beschreven kan worden als een 'mashup'. De populariteit van het gebruik van web 2.0 diensten stijgt enorm. Flickr, Hyves, Digg, Youtube, Facebook, Plazes en vele anderen, behoren tot het dagelijks gebruik van vele Internetters.



Afbeelding 1 – Tim O'Reilly, noemde voor het eerst de term web 2.0

Maar waarom was O'Reilly nou zo enthousiast over het gelanceerde Yahoo! Pipes? O'Reilly heeft het in zijn benadering van web 2.0 over een platform. Dit platform wordt gevormd door aan elkaar gekoppelde diensten via een API. Yahoo Pipes maakt het voor iedereen mogelijk om zonder kennis van programmeren diensten te koppelen. Dit betekend een enorme stimulans voor 'het platform' waar O'Reilly het over heeft. Consumenten kunnen op deze manier een eigen mashup maken binnen een inventieve applicatie. Kortom, een relevante en interessante dienst op het gebied van API's.

Naast de ontwikkelingen op het gebied van web 2.0 is het interessant om naar het specifieke onderdeel web API's te kijken. Deze scriptie zal aan de hand van de web 2.0 principes een toelichting geven op dit specifieke onderdeel. Er is onderzocht welke rol web API's spelen binnen de ontwikkelingen van het Internet, vanuit een commercieel- en technisch oogpunt, en welke waarden ze hebben. Daarnaast wordt de nadruk gelegd op één specifieke web API,

[1] Een mashup is een website of applicatie die verschillende diensten van anderen met elkaar combineert.
[2] Online omgeving wat lijkt op de traditionele desktop omgeving (Windows, Mac OSx of Linux).
[3] Application Programming Interface, beschrijft hoe software aangesproken kan worden.

namelijk die van Google Maps. De mogelijkheden hiervan waren opgepikt door Gather, waarbij ideeën zijn ontstaan om deze toepassing te gaan gebruiken. Dit onderzoek biedt ondersteuning in deze keuze.

Over Gather B.V.

De afstudeerstage is uitgevoerd bij het bedrijf Gather B.V. Gather is een dochteronderneming van Internbureau Websight. Websight is een internetbureau voor marketing- en communicatie mensen. Met een gevarieerd aanbod van diensten probeert Websight haar klanten te bedienen. Deze klanten variëren van het MKB tot marketing- en communicatie afdelingen van grotere ondernemingen. Gather is onderdeel van het dienstenpakket van Websight en richt zich vooral op nieuwe ontwikkelingen in de markt. Zo worden de mogelijkheden van nieuwe technieken onderzocht en geïmplementeerd, wordt er gekeken wat de mogelijkheden zijn van diverse webservices en wordt er naar kansen gezocht voor nieuwe initiatieven. De rol van de afstudeerstage is het onderzoeken van de mogelijkheden van web API's en in het bijzonder de mogelijkheden van de Google Maps API.

Probleemstelling

Het onderzoek

Dit onderzoek zal gericht zijn op *de technische- en commerciële waarden van web API's*, met als insteek twee doelgroepen: *de producenten of uitgevers van web API's* en *de gebruikers van web API's*[4]. Om dit onderzoek mogelijk te maken is een splitsing gemaakt in de vorm van deelvragen.

De deelvragen die aan bod zullen komen luiden als volgt:

1. Wat is Web 2.0?
2. Wat zijn web API's?
3. Wat zijn de technische mogelijkheden van web API's?
4. Wat zijn de commerciële mogelijkheden van het beschikbaar stellen van een web API?
5. Waarom moeten commerciële bedrijven web API's gebruiken?

Opbouw scriptie: de deelvragen toegelicht

Het gebruik van web API's, en de toepassing ervan, speelt een belangrijke rol binnen de huidige ontwikkelingen van het Internet. Het is daarom eerst van belang deze ontwikkelingen te omschrijven en aan te geven welke positie web API's hierin hebben *(deelvraag 1)*. Vervolgens kan een gedetailleerde definitie van web API's gemaakt worden *(deelvraag 2)*. Om aan te geven welke technische mogelijkheden web API's bieden, zal beschreven worden welke standaarden en protocollen er zijn, en hoe deze toegepast kunnen worden. Daarnaast zal technisch ingegaan worden op het zelf ontwerpen en beschikbaar stellen van een web API *(deelvraag 3)*.

Na de technische benadering, zal gekeken worden naar de commerciële waarden. Allereerst kijkend naar het beschikbaar stellen van een web API *(deelvraag 4)* en vervolgens het gebruik ervan *(deelvraag 5)*.

Voorgaande onderdelen worden beschreven aan de hand van bepaalde theoretische informatie met daarnaast informatie verkregen uit de praktijk. Afsluitend zullen er conclusies worden getrokken met daaruit volgend diverse aanbevelingen.

De hoofdvraag

Bovengenoemde deelvragen zullen uiteindelijk een antwoord geven op de hoofdvraag *'Wat zijn de technische- en commerciële waarden van web API's voor producenten en gebruikers?'*.

Doelstellingen

Het onderzoek heeft als doel inzicht te bieden in het, technisch- en commercieel, gebruik van web API's. Zoekend naar een antwoord op de hoofdvraag, heeft het onderzoek voor Gather als doel de mogelijkheden van Google Maps te belichten en het gebruik van de API te demonstreren. Daarnaast zal het onderzoek een aantal ontwikkelingen binnen de markt beschrijven, wat voor Gather tot nieuwe inzichten kan leiden.

[4] De term gebruiker is niet de eindgebruiker van de applicatie. De term gebruiker betekend de gebruiker (ontwikkelaar) van de API. In deze scriptie wordt de eindgebruiker van een applicatie die gebruik maakt van een API 'de consument' genoemd.

Naast de doelstellingen voor Gather, heeft het onderzoek als doel de bevindingen tijdens de afstudeerperiode kenbaar te maken aan geïnteresseerden, vakgenoten, begeleiders en de examencommissie. Er is verder nog weinig literatuur over web API's geschreven waarin koppelingen worden gelegd met web 2.0. Bestaande literatuur is vooral gericht op specifieke onderdelen, en met name alleen te vinden in losse artikelen op het Internet of in vakbladen. Een ander doel van dit onderzoek is daarom het beschikbaar maken van informatie over dit onderwerp, waarbij gekozen wordt voor een brede insteek.

Voor de auteur zijn er ook persoonlijke doelstellingen. Voornamelijk op het gebied van kennis van web API's en het leren beschrijven van interessante trends en ontwikkelingen. Ook het praktisch kunnen ontwikkelen en toepassen van web API's behoren tot de persoonlijke doelstellingen.

Definitieve opdrachtomschrijving

Het onderzoek

Onderzoek de technische- en commerciële mogelijkheden van web API's. Benader dit vanuit twee invalshoeken: vanuit de 'producent' of 'aanbieder' en vanuit de 'gebruiker'. Het is belangrijk dat het onderzoek de technische- en commerciële mogelijkheden samenvoegt en zal leiden tot verschillende aanbevelingen. Een belangrijke web API die aan bod moet komen is de Google Maps API. Deze API wil Gather gaan gebruiken binnen het dienstaanbod. Ten behoeve hiervan dient gedemonstreerd te worden wat de mogelijkheden van deze API zijn.

Staatopdekaart.nl

Voor de ondersteuning van de scriptie wordt een project gerealiseerd dat gebruik maakt van verschillende web API's. De belangrijkste hiervan is de Google Maps API, welke eventueel gecombineerd kan worden met andere API's. Het doel van het project is inzicht krijgen in de verschillende API's met daarnaast de mogelijkheid om klanten een demonstratie te geven van de mogelijkheden. Een extra doel is het experimenteren met dergelijke projecten op het gebied van commerciële aspecten. Het zal een leuke bijkomstigheid zijn als het project zal slagen en dus daadwerkelijk gebruikt gaat worden.

Staatopdekaart.nl moet aan de hand van een Google Maps kaart informatie over bedrijven, evenementen, vervoer, cultuur, media, architectuur en verblijfsplaatsen van bepaalde steden aan de hand van een geografische weergave vindbaar maken. Informatie van een bepaalde stad is op te vragen via het internet adres http://plaatsnaam.staatopdekaart.nl[5]. De bezoeker krijgt de mogelijkheid zijn interesses aan te geven in een lijst van categorieën. Vervolgens verschijnen er punten op de kaart, waarbij ieder punt een eigen icoon heeft, gebaseerd op de desbetreffende categorie. De bezoeker kan op deze manier op zoek gaan naar diverse informatie. Naast het kunnen zoeken op de kaart is er een alfabetische weergave van de gevonden punten.

[5] Hierbij is plaatsnaam de gewenste stad uit Nederland, bijvoorbeeld http://hilversum.staatopdekaart.nl

Plan van aanpak

In feite is de afstudeerstage verdeeld in twee hoofd producten, namelijk de scriptie en Staatopdekaart.nl. Binnen dit hoofdstuk wordt de te volgen aanpak per product omschreven. Vervolgens leidt dit tot een algemene planning voor de gehele stageperiode.

Staatopdekaart.nl

Doordat Staatopdekaart.nl zo snel mogelijk demonstreerbaar moet zijn voor klanten, vergt dit een snelle aanpak op het gebied van ontwikkelen. Iedere nieuw ontwikkelde functionaliteit wordt direct online gebracht, om deze direct te kunnen tonen aan klanten. Vooraf wordt een globaal functioneel ontwerp gemaakt met daaropvolgend een voorstel voor de vormgeving. Daarnaast worden alle ideeën die er zijn voor Staatopdekaart verwerkt in een document en op chronologische volgorde gezet. De planning voor al deze ideeën wordt zo globaal mogelijk gehouden, omdat zoals eerder al gezegd, de ontwikkeling in een rap tempo moet.

Scriptie

De eerste periode staat geheel in het teken van Staatopdekaart.nl. Langzamerhand wordt steeds meer de scriptie hierbij betrokken. De scriptie gaat gepaard met een literatuur onderzoek. Uiteindelijk moeten de bevindingen van het project Staatopdekaart.nl aanvulling bieden voor de scriptie.

Planning

In onderstaand overzicht de globale planning voor de afstudeerperiode. Deze planning biedt een richtlijn en kan op sommige momenten specifieker gemaakt worden.

Week 7	Planning, plan van aanpak, Google Maps API testen, Staatopdekaart systeem opzetten
Week 8	Staatopdekaart systeem opzetten, locaties importeren, **eerste versie Staatopdekaart.nl online**
Week 9	Kleine uitbreidingen Staatopdekaart, bugs oplossen
Week 10	Informatie vergaren voor scriptie
Week 11	Structuur scriptie opzetten, doorontwikkelen Staatopdekaart.nl
Week 12	doorontwikkelen Staatopdekaart.nl
Week 13	doorontwikkelen Staatopdekaart.nl
Week 14	**Routeplanner Staatopdekaart.nl online**
Week 15	Scriptie
Week 16	Scriptie
Week 17	Staatopdekaart.nl
Week 18	Scriptie
Week 19	Scriptie, Staatopdekaart.nl
Week 20	Scriptie
Week 21	Scriptie, **Scriptie af**
Week 22	Staatopdekaart.nl
Week 23	Staatopdekaart.nl
Week 24	Staatopdekaart.nl
Week 25	Staatopdekaart.nl, Presentatie voorbereiden
Week 26	Presentatie voorbereiden, **afstudeerbijeenkomst**
Week 27	**Afstudeerbijeenkomst**

Hoofdstuk 1:

Het nieuwe Internet

1.1 Web 2.0

De term web 2.0, een buzzwoord of een logische evolutie van Internet (technologie)? De meningen verschillen nogal. De opkomst van de term web 2.0 is beschreven in de inleiding. Deze paragraaf zal de term verder toelichten.

Web 2.0 is de term die staat voor de grote verandering van het Internet. Hierbij ontwikkelt het Internet zich steeds meer van een verzameling losstaande websites tot een totaal platform met interactieve webtoepassingen. Hierbij staan de consumenten met bijbehorende sociale aspecten centraal. Voorheen werd het Internet vooral gebruikt als eenrichtingsverkeer. De maker van een website bepaalde wat er gebeurde. Slechts op kleine schaal konden consumenten op de achtergrond met elkaar communiceren via forums en chat. Nu is dit geheel naar de voorgrond gekomen en staat bij web 2.0 de consument, de gebruiker, centraal. De gebruiker bepaald wat er gebeurd. Qua gebruikservaring komen Internet toepassingen steeds dichter in de buurt van de dekstop applicaties. Met een combinatie van bestaande technieken, worden applicaties voorzien van inventieve interfaces.

Om bovenstaande omschrijving duidelijker te maken, worden verschillende web 2.0 gedachtegangen afzonderlijk toegelicht.

1.1.1 Het Internet als een platform

Bij het ontstaan van de computer werd het begrip 'platform' geïntroduceerd. Een platform had als doel om computer instructies te vergemakkelijken. Bij het ontstaan van de computer werden in het begin alleen directe instructies aan de processor doorgegeven. Om dit gemakkelijker te maken ontstonden er programmeertalen en besturingssystemen. Dit werden platformen genoemd, die instructies, op een voor de mens leesbare manier, omzetten in processor code. Deze platformen bevatten een aantal standaard bibliotheken met voorgeprogrammeerde code, die hergebruikt kon worden door programmeurs. Op deze manier scheelde dit tijd en geld.

Bij web 2.0 is eigenlijk hetzelfde van toepassing. Waar eerst losstaande websites afzonderlijk opereerde, is de trend om op verschillende manieren data beschikbaar te stellen voor derden. Deze data wordt beschikbaar gesteld via API's. In een API staat beschreven hoe een bepaalde dienst van buitenaf aangesproken kan worden. Op deze manier kunnen verschillende diensten en datastromen gecombineerd worden (mashups) of is er de mogelijkheid tot 'hackability' (aanpassingen naar eigen smaak).

Kijkend naar het ontstaan van de computer met bijbehorende platformen, gaat het Internet ook richting één platform. Binnen dit platform kunnen diensten hergebruikt, aangepast of gecombineerd worden.

1.1.2 Vertrouwen in de creaties van de gebruikersgroep

Zoals gezegd staat de gebruiker centraal. Het vertrouwen in deze gebruiker is zeer belangrijk. Veel web 2.0 diensten bieden gebruikers de mogelijkheid om content toe te voegen en op deze manier een bijdrage te leveren aan de dienst. Dit kan omschreven worden met het begrip User Generated Content. Meestal gaat dit gepaard met de mogelijkheid om bijdragen van gebruikers te laten corrigeren door andere gebruikers. Gebruikers zijn bereid om energie hierin te steken. Daarom is het voor een web 2.0 dienst van belang te vertrouwen op deze bijdragen en er van uit gaan dat gebruikers elkaar corrigeren.

1.1.3 Sociaal

Sociale aspecten vormen een belangrijk onderdeel van web 2.0. Gebruikers willen alles met elkaar delen: evaringen, gebeurtenissen, producten en kennis. Via allerlei web 2.0 diensten wordt deze mogelijkheid geboden. Doordat bepaalde informatie gedeeld is, kunnen mensen met bepaalde interesses gemakkelijk met elkaar in contact komen. Daarnaast speelt (online) identiteit een belangrijke rol op sociaal vlak. Mensen willen ergens bij horen. Online wordt dat mogelijk, door specifieke informatie te delen, wat jouw identiteit bepaald. Er kan bijvoorbeeld gedeeld worden welke merken bij een persoon horen.

1.1.4 Gebruikservaring

De term "Rich user experiences" staat nauw in verband met web 2.0. De gebruikservaring van een web 2.0 service gaat gemoeid met een intuïtieve interface. Deze interface gaat steeds meer lijken op die van de traditionele desktop applicaties. Door het gebruik van AJAX wordt de gebruikservaring binnen een web pagina sterk verbeterd. AJAX is het asynchroon ophalen van data afkomstig van een server side taal. Later zal dit begrip verder toegelicht worden. De invloed van het gebruik van AJAX voor de interface en de gebruikservaring is enorm. Waar vroeger complete web pagina's herladen moesten worden, worden nu slechts enkele delen herladen aan de hand van acties door de gebruiker. Op deze manier lijkt het alsof een gebruiker binnen een online desktop applicatie werkt.

Naast het gebruik van AJAX, kenmerken web 2.0 applicaties zich door het toepassen van geavanceerde user interface componenten. Deze componenten zijn vergelijkbaar met traditionele desktop user interface componenten. Met behulp van Javascript behoren drag & drop, animaties, sliders, pull down menu's en interactieve componenten tot de mogelijkheden.

Het grootste voordeel is dat een interface compleet naar eigen smaak in te richten is. Bij desktop applicaties is een ontwikkelaar gebonden aan voorgedefinieerde componenten. Mede hierdoor, en de technische ontwikkelingen, wordt verwacht dat op een gegeven moment geen onderscheid meer zal zijn tussen 'desktop' en 'online'.

1.1.5 Ontwikkelen van web 2.0 applicaties

De ontwikkeling van web 2.0 applicaties is zeer tegenstrijdig met de traditionele software ontwikkeling. Bij traditionele ontwikkelmethoden worden nieuwe versies van de software pas gelanceerd na uitvoerig testen en volgens een vooraf gedefinieerde planning of roadmap. Bij web 2.0, waar de gebruiker centraal staat, wordt voor iedere kleine wijziging een update gelanceerd. Deze update kan direct door de gebruikersgroep, al dan niet een selecte groep, getest worden. Gebruikers kunnen direct feedback geven op een nieuwe functie. Op basis van de feedback kunnen bijvoorbeeld aanpassingen gedaan worden die de nieuwe functie verbeteren. Indien blijkt dat de nieuwe functie niet werkt of niet gewenst is door de gebruikersgroep, kan zelfs gekozen worden de functie volledig te schrappen. Deze eigenschap wordt vaak omschreven als 'de web 2.0 dienst die eeuwig beta is', vandaar is de term 'beta' (testversie) vaak terug te vinden in benamingen en logo's.

Deze methode heeft natuurlijk invloed op het ontwikkelproces van een applicatie en vergt een andere aanpak. Belangrijk is dat de gebruiker centraal staat, wat duidelijk naar voren komt binnen deze aanpak. Vaak worden web 2.0 applicaties voorzien van een weblog, waarin ontwikkelaars en gebruikers met elkaar kunnen discussiëren over nieuwe functies voor de applicatie.

1.1.6 Tags

Het organiseren van (persoonlijke) data kan een enorme kluif zijn. Een typische web 2.0 toepassing die dit probleem oplost is de mogelijkheid voor 'tagging'. Hierbij wordt geproduceerde content voorzien van trefwoorden, ook wel 'tags' genoemd. Het voorzien van deze tags gebeurd door de gebruikersgroep zelf. Als voorbeeld: er wordt een foto geplaatst op een site. Andere gebruikers zien de foto en voorzien de foto van trefwoorden die te zien zijn op de foto. Andere gebruikers zijn weer op zoek naar bepaalde trefwoorden. Na het zoeken op een trefwoord, worden alle foto's die voorzien zijn van dat trefwoord getoond.

Niet alleen het makkelijk kunnen vinden van content behoort tot de voordelen van tags. Zo kan door middel van tags content gepresenteerd worden aan de hand van persoonlijke voorkeuren. Een voorbeeld: op de muzieksite Last.fm kan je luisteren naar webradiostations met liedjes die bij één tag horen, zoals 'angry' of 'dutch'. Last.fm houdt nauwkeurig alle nummers bij die je op je computer draait. Je krijgt suggesties van andere bands die je waarschijnlijk ook goed vindt en ziet automatisch welke mensen dezelfde muzieksmaak hebben als jij. Typisch web 2.0: op basis van de verzamelde gegevens door de gebruikersgroep krijg je aanbevelingen die precies bij jou passen.

Het voorzien van tags hangt nauw samen met het eerder genoemde vertrouwen in de gebruikersgroep. Er moet op vertrouwd worden dat gebruikers op een serieuze wijze tags toevoegen.

1.1.7 Toepassingen

Om een nog duidelijker beeld te geven van web2.0 zullen een aantal kenmerkende toepassingen beschreven worden.

Weblogs
Een weblog, of ook wel blog, is een website die regelmatig, soms meerdere keren per dag, vernieuwd wordt en waarop de geboden informatie aflopend op datum wordt weergegeven. De auteurs van een weblog bieden in feite een logboek van informatie die ze willen delen met het publiek. Meestal gaat het hier om tekst, het kan ook om foto's (een fotoblog), video (vlog) of audio (podcast) gaan. Weblogs bieden lezers de mogelijkheid om reacties onder de berichten te plaatsen. Op deze manier ontstaat een hechte gebruikersgroep rondom een bepaald thema, waarin hevig gediscussieerd kan worden. Weblogs kunnen zakelijk gezien ook interessant zijn. Zo kan een bedrijf een corporate weblog starten, waarin informatie naar klanten gecommuniceerd kan worden, die daaropvolgend weer de mogelijkheid hebben om feedback daarop te geven.

Het is het persoonlijke of juist het gespecialiseerde karakter dat weblogs interessant maakt voor bezoekers.

Wiki's
Een wiki is een online applicatie waarin documenten gezamenlijk beheert kunnen worden. Doordat iedereen de mogelijkheid heeft aanpassingen in documenten te verrichten, biedt een wiki een open systeem aan voor het

produceren van informatie. Een goed voorbeeld is de online encyclopedie Wikipedia, waar iedere bezoeker de mogelijkheid heeft een bijdrage te leveren aan de encyclopedie. Het bewerken van documenten binnen een wiki gaat gepaard met tools voor discussie, historie van een document en misbruikmelding. Een wiki is typisch web 2.0, door het open karakter en het kunnen leveren van bijdragen door de gebruikersgroep.

Social Networking
Sociale netwerken brengen grote groepen met hetzelfde interessegebied met elkaar in contact. Zo zijn er sociale netwerken voor bedrijven, scholieren, sportverenigingen etc. Een sociaal netwerk biedt deze groepen de mogelijkheid om allerlei soorten informatie met elkaar te delen. Vaak bieden dergelijke diensten een totaal pakket van web 2.0 toepassingen aan onder één dienst. Het belangrijkste kenmerk van een social network is dat er connecties worden gelegd tussen personen (met dezelfde interesses). Hierdoor ontstaat een compleet netwerk aan connecties.

1.1.8 Schaduwkanten van web 2.0

Hoewel web 2.0 een complete beschrijving gekregen heeft, zijn er critici die anders denken over web 2.0. Zo wordt vaak het verwijt gemaakt dat alle eigenschappen van web 2.0 al eerder zijn toegepast en daarom helemaal niet nieuw zijn. Amazon liet het sinds de start in 1995 bijvoorbeeld al toe om door gebruikers product reviews te plaatsen. Een vorm van het betrekken van gebruikers. Daarnaast was het ook Amazon die in 2002 al een web API beschikbaar had voor andere ontwikkelaars. Andere critici vatten web 2.0 op als marketing buzzwoord. Voor hen heeft web 2.0 geen enkele betekenis en wordt het alleen gebruikt als verkooppraatje.

Op technisch gebied menen velen ook dat er weinig vernieuwend is. AJAX bestond al een tijd en het is tevens geen verandering van het Internet. Het is slechts een extra laag boven op de huidige technieken en protocollen.

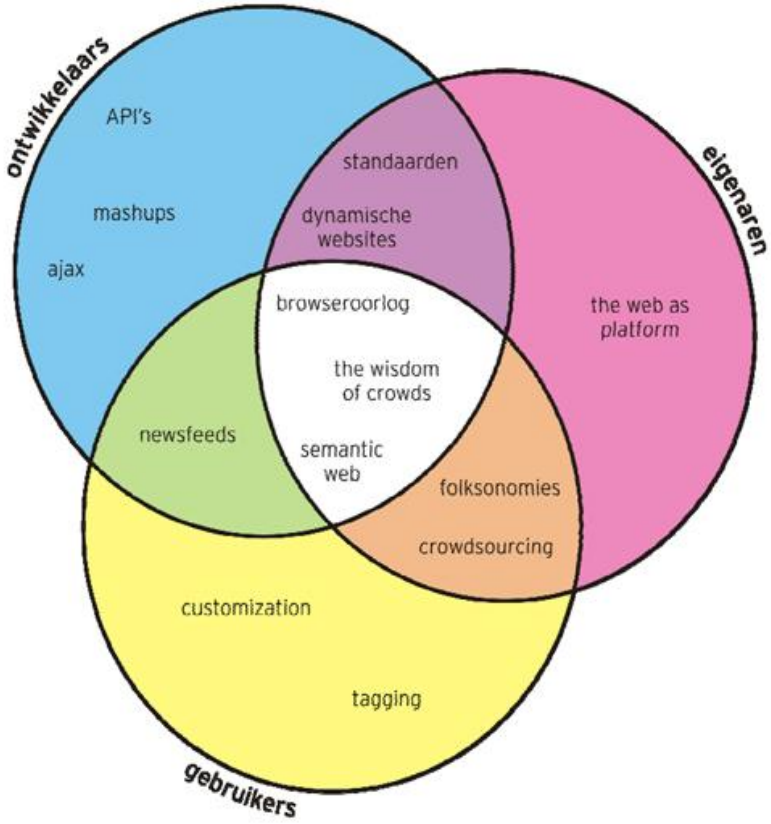
Een term die ook vaak genoemd wordt is bubble 2.0. Hierbij zien critici veel overeenkomsten met de Internet bubble rondom 2000. Investeerders laten weer hun neus zien en zijn bereid te inverteren in nieuwe startups die geen overtuigende verdien modellen hebben.

1.2 Web 2.0 en API's

Een belangrijke eigenschap van web 2.0 is 'het Internet als platform'. De eerdere beschrijving gaf aan dat het combineren en koppelen van online diensten nauw verbonden is met web 2.0. Door het combineren en koppelen van al deze diensten ontstaat één platform.

Om bepaalde onderdelen van een dienst beschikbaar te stellen aan derden moet beschreven worden hoe deze onderdelen te benaderen zijn. Deze beschrijving wordt gegeven middels een API. Een API geeft aan welke commando's (programmeercode) aangeroepen kunnen worden. Deze commando's hebben ieder bepaalde eigenschappen, gericht op bepaalde doeleinden. Als voorbeeld een kaart applicatie die voor externe ontwikkelaars te integreren is in een website. Om deze kaart daadwerkelijk te integreren worden in de API een aantal functies omschreven die het mogelijk maken de kaart op te halen. De ontwikkelaar wilt ook eigen gegevens in de kaart laden. Hiervoor staan ook weer losse functies omschreven in de API die het mogelijk maken eigen gegevens in de kaart te laden.

Tot slot kan de term web2.0 samengevat worden aan de hand van onderstaande afbeelding.



Zoals gezegd is een belangrijke eigenschap van web 2.0 het ontstaan van één platform. Hierbij is het gebruik van API's zeer van belang. Nu beschreven is hoe de web 2.0 filosofie in elkaar zit, zal in deze scriptie dieper op het onderdeel web API's in worden gegaan.

Hoofdstuk 2:

Web API standaarden en protocollen

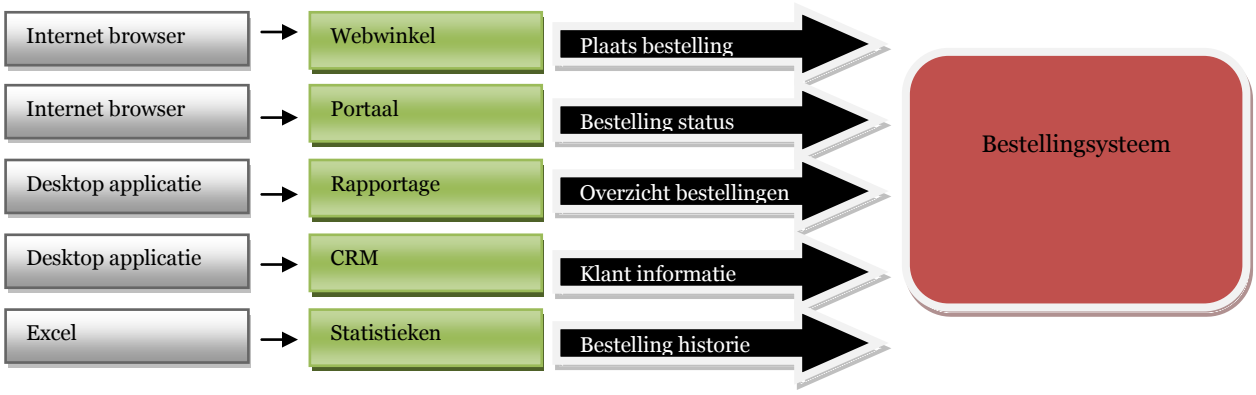
2.1 Wat is een API?

Een API (Application Programming Interface) definieert hoe computerprogramma's onderling kunnen communiceren of kunnen communiceren met een bibliotheek. Een bibliotheek bevat functies en objecten gericht op een bepaald onderdeel (bijvoorbeeld een bibliotheek voor de printer, USB poort, videokaart etc.). De manier van aanspreken hiervan staat beschreven in een API. API's vormen een scheiding tussen lagen van abstractie. Applicaties kunnen op deze manier op een hoog abstract niveau werken, en werk met een lagere abstractie af laten handelen door andere API's. Als voorbeeld een tekenprogramma, dat hoeft niet te weten hoe de videokaart een lijn op het scherm tekent. Het tekenprogramma hoeft alleen maar een eenvoudige functie uit de 'teken-API' aan te roepen, die er voor zorgt dat de videokaart een lijn op het scherm tekent. Een voordeel van een API is, dat ontwikkelaars geen kennis nodig hebben over de interne werking of implementatie van een bepaalde bibliotheek, maar door de beschrijving in de API wel gebruik van een bibliotheek kunnen maken.

API's zijn er op vele niveaus. Een besturingssysteem als Windows bevat de Microsoft Windows API (Win32 API). Deze API handelt de communicatie tussen software en hardware af. Daarboven kan bijvoorbeeld een API bestaan van de programmeertaal Java. Deze API beschrijft functies die naar alle waarschijnlijkheid gecombineerde Win32 API functies verwerken tot één functie. Java zorgt dus voor een snellere implementatie van bepaalde functionaliteiten, zonder te maken te krijgen met afhandeling van communicatie tussen hardware en software.

2.1.1 Web API's

Het principe van web API's is gebaseerd op de bovenstaande traditionele benadering. Web API's zorgen ervoor dat web applicaties onderling met elkaar kunnen communiceren. Deze onderlinge applicaties kunnen twee externe applicaties zijn, maar ook interne applicaties. Een voorbeeld van het laatste genoemde is bijvoorbeeld een bestellingsysteem. Onderstaande schema geeft aan welke acties er op het bestellingsysteem losgelaten kunnen worden. Het bestellingsysteem is één applicatie (rood) die deze acties intern afhandelt. Dit systeem kan aangesproken worden via een API (zwart). De ontwikkelaar die gebruik maakt van deze API, hoeft dus niet te weten hoe het bestellingsysteem intern werkt. In onderstaand schema is te zien dat het gebruik van het bestellingsysteem applicatie (groen) en platform (grijs) onafhankelijk is.



Een andere toepassing is het gebruik van de API door externe ontwikkelaars. Specifieke onderdelen van bovengenoemd bestellingsysteem kunnen ook beschikbaar worden gesteld aan derden, die bepaalde onderdelen in een eigen dienst kunnen integreren.

Een web API is in feiten een systeem dat XML berichten verstuurd over een netwerk via het HTTP protocol. XML is een opmaaktaal voor data, dat platformonafhankelijk toegankelijk is. Voor het versturen van XML berichten bestaan meerdere stijlen en standaarden. Allereerst worden verschillende stijlen web API's toegelicht. Vervolgens wordt dieper ingegaan op verschillende standaarden en protocollen.

2.2 Web architectuur stijlen

Voor het ontwikkelen van software bestaan verschillende architecturen. Dit is ook het geval voor web applicaties. Voor het intern koppelen van componenten bestaan verschillende web architectuur stijlen. Deze stijlen hebben invloed op het beschikbaar maken van een API.

Er zijn drie belangrijke manieren van communiceren binnen een systeem. Er kunnen *operations* gecommuniceerd worden, waarbij directe systeem functies direct aangeroepen worden. Er kunnen *messages* gecommuniceerd worden, waarbij functies van een systeem worden aangeroepen aan de hand van berichten. En tot slot kan er gecommuniceerd worden door middel van *resources*, waarbij een 'resource' centraal staat bij het communiceren. Deze drie manier zijn verbonden aan RPC, SOA en REST, welke nu afzonderlijk worden beschreven.

2.2.1 Remote Procedure Calls (RPC)

RPC is een software architectuur stijl gebaseerd op procedures die op afstand aangeroepen worden. Om de gedachtegang van RPC te beschrijven moet eerst gekeken worden naar het onderdeel 'procedure calls'. In veel software applicaties wordt gebruik gemaakt van procedures. Procedures bundelen bepaalde stappen die doorlopen worden na een bepaalde actie. Ter verduidelijking een voorbeeld uit het Windows besturingssysteem. Wanneer de gebruiker een muisklik actie uitvoert wordt een procedure uitgevoerd om te kijken waarop geklikt wordt. Om dit te controleren is het voor de procedure van belang de positie van de muis te weten (de x- en y coördinaten). Deze informatie wordt meegestuurd naar de procedure via de zogenoemde parameters. Aan de hand van de parameters bekijkt de procedure waarop geklikt is en geeft een antwoord terug. Samengevat is een 'procedure call' de omschrijving van een procedure, de procedure parameters en het resulterende antwoord van een procedure.

Een stap verder is een 'remote procedure call (RPC)'. RPC is een vrij simpele uitbreiding op het idee van procedure calls. RPC maakt het mogelijk om connecties te maken tussen verschillende applicaties door middel van het op afstand aanroepen van procedures. Belangrijk bij RPC is de manier van gegevens overdracht. De gegevens moeten in een formaat aangeboden wordt dat te begrijpen is door beide applicaties. Een belangrijk feit is dat bij RPC direct de applicatie logica van een externe applicatie wordt aangeroepen. Dit betekend dat de applicaties die onderling gegevens uitwisselen via de RPC gedachtegang sterk afhankelijk van elkaar zijn.

RPC kent een aantal oude gestandaardiseerde systemen, zoals Sun RPC, Distributed computing environment (DCE), Microsoft DCOM en COBRA. Tegenwoordig zie je systemen die op XML gebaseerd zijn en communiceren via het HTTP protocol, zoals XML-RPC en SOAP.

Kernwoord bij RPC is het communiceren van *operations*.

2.2.2 Service Oriented Architecture (SOA)

Vrij vertaald kan SOA (Service Oriented Architecture) omschreven worden als een architectuur gebaseerd op verschillende 'services'. Deze services zijn ieder afzonderlijke diensten binnen een architectuur die afzonderlijk kunnen opereren. Een service is een component binnen een architectuur dat te gebruiken of te koppelen is zonder de werking van de interne implementatie te weten. Door verschillende services aan elkaar te koppelen via API's, kunnen systemen ontwikkeld worden voor het uitvoeren van bedrijfsprocessen. Kijkend naar deze bedrijfsprocessen, is de volgorde van deze processen gemakkelijk te wijzigen binnen een service oriented architecture. Doordat iedere service een afzonderlijk component is, kan de volgorde van services binnen een systeem willekeurig zijn.

Bij de opkomst van web 2.0 is het populaire SOA ook van toepassing geworden op Internet applicaties. 'Het web als platform' ligt volledig in de lijn met service georiënteerde systemen. De services worden in dit geval 'web services' genoemd. Web services kunnen intern aan elkaar gekoppeld worden. Daarnaast kunnen afzonderlijke services ook beschikbaar gesteld worden voor derden, waarmee externe koppelingen gemaakt kunnen worden. Standaarden die bij web services komen kijken zijn vooral gebaseerd op XML. Standaarden die toegepast worden voor de service georiënteerde gedachtegang zijn bijvoorbeeld SOAP, WDSL en XACML.

Kernwoord bij SOA is het communiceren van *messages*.

2.2.3 Representational State Transfer (REST)

Bij REST is het uitgangspunt 'resources' en het toepassen van vier acties daarop: create, read, update, destroy (CRUD). Deze acties worden toegepast via de standaard HTTP request methoden, namelijk: GET (read), POST (create), PUT (update) en DELETE (destroy). Deze vier requests worden gezamenlijk verstuur met een URL, waarin beschreven staat op welke resource de actie uitgevoerd moet worden. Een resource is een afzonderlijk component binnen het systeem.

Het uitgangspunt van een systeem dat gebruik maakt van REST, ook wel een RESTful applicatie genoemd, is eigenlijk gebaseerd op een paar basis eisen. In onderstaand voorbeeld is een praktisch voorbeeld beschreven, ter verduidelijking van de REST theorie:

Een gebruikerssysteem bevat de resource 'gebruikers'. Door het URL http://voorbeeld.nl/users/ te benaderen via de methode GET verschijnt het overzicht van gebruikers. De gebruiker met het ID (identifier) '3' kan bekeken worden via http://voorbeeld.nl/users/3 via de methode GET. Deze gebruikers dient verwijderd te worden. Dat kan simpel gedaan worden via het URL http://voorbeeld.nl/users/3 via de methode DELETE. De laatste twee URL's waren dus identiek, met echter een andere aanroep methode.

http://voorbeeld.nl/users/	GET	Lijst van gebruikers tonen
http://voorbeeld.nl/users/3	GET	Gebruiker met ID '3' tonen
http://voorbeeld.nl/users/3	DELETE	Gebruiker met ID '3' verwijderen

Door REST te combineren met een XML output wordt het mogelijk om te communiceren tussen onderlinge componenten. Een aantal componenten kan ook beschikbaar worden gesteld aan derden als API. Door de eenvoud van REST, is het een veel toegepaste manier van beschikbaar stellen van API's.

Kernwoord bij REST is het communiceren van *stateful resources*.

2.3 Standaarden en protocollen voor web API's

2.3.1 Gebaseerd op XML

Veel web API's versturen en ontvangen XML berichten (Extensible Markup Language). XML is ontwikkeld met het doel om data te kunnen uitwisselen tussen verschillende soorten informatiesystemen, met name gericht op Internet. Het opmaken van een XML document geschiet op een semantische manier. Semantiek is de betekenis van woorden. In het geval van XML betekend semantiek de betekenis van aangegeven tags en attributen. Tags en attributen worden in onderstaand voorbeeld, een verzameling boeken, verduidelijkt:

```
<?xml version="1.0" encoding="UTF-8"?>
<verzameling>
  <boek nummer="1">
    <titel>Mijn eerste boek titel</titel>
    <auteur>Auteur 1</auteur>
  </boek>
  <boek nummer="2">
    <titel>Mijn tweede boek titel</titel>
    <auteur>Auteur 2</auteur>
  </boek>
</verzameling>
```

Opmerking voor code voorbeelden: **Aandachtspunten** *worden onderstreept en vetgedrukt aangegeven.*

Tags worden aangeduid met `<boek>` en attributen geven een extra omschrijving aan een tag, bijvoorbeeld `nummer="1"` (boeknummer 1). Tags kunnen ook een waarde bevatten, bijvoorbeeld `<titel>Boek titel</title>`. Hierin heeft de tag `<title>` de waarde 'Boek titel'. Door structuren aan te brengen met tags en attributen ontstaat de mogelijkheid deze data uit te wisselen via verschillende platformen. Deze platformen bevatten hun eigen functies voor het verwerken van XML data. De verdere mogelijkheden van XML zijn uitgebreider dan hierboven beschreven.

2.3.2 RSS

RSS (Really Simple Syndication) is niet zo zeer een web API of protocol, maar heeft wel kenmerken van een web API. RSS is een in XML opgemaakt document waarin de laatste toevoegingen van een site worden bewaard. RSS is vooral populair onder nieuws websites en weblogs, waarin de laatste nieuwsberichten in een RSS feed worden gepubliceerd. Gebruikers van een website kunnen een RSS lezer downloaden, die op een simpele manier alle laatste berichten van alle geabonneerde RSS feeds binnen haalt. Op deze manier is het eenvoudig om op de hoogte te blijven van al het laatste nieuws, zonder daadwerkelijk een bezoek te brengen aan al deze websites. Sinds RSS 2.0 is het mogelijk koppelingen aan te geven naar bestanden. Deze bestanden kunnen bijvoorbeeld geluidsfragmenten of video's zijn. Zo ontstonden er op basis van RSS, podcasts (een RSS feed met geluidsbestanden) en vodcasts (een RSS feed met video on demand video's).

RSS is op XML gebaseerd. Door van te voren bepaalde specificaties vast te stellen, kunnen meerdere platformen en meerdere applicaties gebruik maken van een RSS feed. In onderstaande code is een simpele opmaak van een RSS feed te zien. Dit geeft aan dat XML, en de specificaties van RSS, uiterst simpel zijn, maar zeer krachtig voor het uitwisselen van informatie tussen bepaalde systemen.

```
<?xml version="1.0" ?>
 <rss version="2.0">
 <channel>

 <title>Titel website</title>
 <link>http://www.locatie_website.nl/</link>
 <description>beschrijving website</description>

 <item>
   <title>titel item1</title>
   <link>http://www.locatie_website.nl/item1.html</link>
   <description>omschrijving item1</description>
 </item>

 <item>
   <title>titel item2</title>
   <link>http://www.locatie_website.nl/item2.html</link>
   <description>omschrijving item2</description>
 </item>

 </channel>
 </rss>
```

2.3.3 Simple Object Access Protocol (SOAP)

SOAP is een communicatieprotocol dat wordt gebruikt voor de communicatie tussen verschillende componenten binnen een systeem. SOAP verstuurt en ontvangt berichten op basis van XML, die verstuurd worden via het HTTP protocol. Naast HTTP kunnen ook berichten verstuurd worden via SMTP, HTTPS en FTP. Het SOAP protocol bestaat uit drie belangrijke onderdelen:

- Een envelop die een framework definieert voor het beschrijven van wat in een bericht staat en hoe het te verwerken.
- Een set van encodeerregels voor de expressie van 'instances' van applicatiegedefinieerde datatypen.
- Een conventie voor de representatie van 'remote procedure calls' en antwoorden. SOAP kan in potentie worden gebruikt in combinatie met een grote verscheidenheid aan andere protocollen.

SOAP is een open standaard en de meeste programmeertalen bieden functies voor het werken met SOAP. SOAP biedt ook ondersteuning voor beveiliging van berichten.

Ter voorbeeld een bericht dat verzonden wordt via SOAP:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>34</productID>
    </getProductDetails>
  </soap:Body>
 </soap:Envelope>
```

Dit bericht geeft aan dat de functie getProductDetails aangeroepen dient te worden en geeft daarbij de parameters productID (34) mee. De functie heeft als eigenschap dat product informatie opgevraagd kan worden aan de hand van een ID. Het systeem roept vervolgens deze functie aan en gebruikt de meegestuurde ID voor het tonen van het juiste product. Het volgende bericht wordt teruggestuurd als response.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
```

```
        <productName>Nike Air</productName>
        <productID>34</productID>
        <description>
          Mooie Nike Air schoenen in kleuren blauw, geel, rood.
        </description>
        <price currency="EURO">120.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

Het bericht dat als response teruggestuurd is bevat de product informatie van het product met ID 34. Deze gegevens kunnen verwerkt en gebruikt worden.

SOAP is zeer complex en bovenstaande is enkel beschreven in een notendop. Er is o.a. ondersteuning voor data types (string, integer, arrays etc.), foutafhandeling, beveiliging etc.

2.3.4 Web Services Description Language (WSDL)

WSDL is een beschrijving in XML formaat van een Web Service. In een document wordt een complete web service beschreven in de vorm van functies met verwachte parameters. Een client applicatie kan op deze manier een WSDL document opvragen aan de server en zien welke functies en parameters er beschikbaar zijn. Met deze informatie kunnen vervolgens aanvragen (bijvoorbeeld een SOAP call) gedaan worden.

Een voorbeeld van een WSDL document (bron: W3C):

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
              <complexType>
                  <all>
                      <element name="tickerSymbol" type="string"/>
                  </all>
              </complexType>
           </element>
           <element name="TradePrice">
              <complexType>
                  <all>
                      <element name="price" type="float"/>
                  </all>
              </complexType>
           </element>
       </schema>
    </types>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>
```

```
    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>

</definitions>
```

Te zien is dat WDSL de volgende eigenschappen bevat:

- Types: bevat de beschikbare datatypes en de naamgeving daarvan binnen verschillende messages (strings, floats, integers etc.);
- Messages: bevat de beschikbare berichten die gecommuniceerd kunnen worden;
- Operations: bevat de locatie (URL) van aan te roepen acties;
- Port types: geeft voor iedere operation aan wat de input- en output bericht is;
- Bindings: bundelt port types een geeft o.a. het transport protocol (bijvoorbeeld SOAP) aan;
- Ports: geeft de locatie (URL) van een specifieke binding aan;
- Services: koppelt verschillende ports tot een 'service'.

2.3.5 Universal Description, Discovery and Integration (UDDI)

UDDI is een soort bibliotheek met WDSL documenten. Bedrijven die web services (API's) beschikbaar willen stellen, kunnen een UDDI maken met daarin alle WDSL documenten. UDDI wordt net als WDSL opgemaakt in een XML document. Ontwikkelaars kunnen op deze manier zoeken naar bestaande web services in UDDI bibliotheken. Hierbij komen de termen 'het web als platform' en 'hergebruik' kijken. Voor een voorbeeld kan bijvoorbeeld gezocht worden in een UDDI zoekmachine, te vinden op http://soapclient.com/uddisearch.html. Gevonden wordt bijvoorbeeld een web service voor het omzetten van een Amerikaans postcode in een Amerikaans adres.

2.3.6 XML-RPC

XML-RPC is een protocol gebaseerd op XML dat verstuurd wordt via HTTP. Het protocol maakt gebruik van de gedachtegangen van de eerder beschreven RPG architectuur. XML-RPC is een uiterst simpel protocol en is zelfs in enkele A4 pagina's te beschrijven. Het protocol ondersteund het aanroepen van procedures (functies), meegegeven van parameters in verschillende data types en foutafhandeling.

Een simpel voorbeeld:

```
<?xml version="1.0"?>
<methodCall>
```

```
  <methodName>products.getProductTitle</methodName>
  <params>
    <param>
        <value><int>40</int></value>
    </param>
  </params>
</methodCall>
```

Bovenstaande aanroep roept de procedure getProductTitle op voor het ophalen van een product title met het ID 40 (waarvan het datatype een integer is). De reactie die teruggestuurd wordt luidt als volgt:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>Nike Air</string></value>
    </param>
  </params>
</methodResponse>
```

Het resultaat is een string met de titel van het product. Zoals te zien is in de voorbeelden, is de opmaak van het XML bericht uiterst simpel.

Na de ontwikkeling van het protocol in 1998 door Dave Winer is het protocol verder uitgebreid met functionaliteiten. Het protocol is uiteindelijk uitgegroeid tot wat nu SOAP is. De keuze voor XML-RPC is vaak dan ook gebaseerd op de simpelheid, in tegenstelling tot SOAP, dat veel complexer is.

2.3.7 Javascript, DOM

Een andere mogelijkheid om diensten beschikbaar te stellen via een API is via Javascript. Javascript is een client-side scripting taal, gebruikt op webpagina's. In combinatie met het Document Object Model (DOM) kunnen HTML, en in andere gevallen XML, elementen benaderd worden in de vorm van objecten. De manier van het beschikbaar stellen van een Javascript API gaat eigenlijk vrij eenvoudig. De producent (ontwikkelaar van de API) ontwikkeld verschillende Javascript classes met bijbehorende functies en beschrijft deze in een API reference. Ontwikkelaars die gebruik willen maken van de API kunnen vervolgens eenvoudig het Javascript bestand, gehost op de server van de aanbieder, 'includen' in hun HTML pagina via de `<script src="http://api-host.nl/link-naar-javascript-bibliotheek/classes.js"></script>` tag. Binnen de HTML pagina zijn dan alle beschikbare functies van de Javascript API beschikbaar.

2.4 Web API's toegepast in de praktijk

Nu de veel gebruikte technieken en protocollen beschreven zijn, worden een aantal voorbeelden uit de praktijk beschreven.

2.4.1 Google Maps

Google Maps, voorheen genaamd Google Local, maakt het vanaf 2005 mogelijk om online de kaart van de wereld te bekijken. Daarbij kan gezocht worden naar bedrijfs-, plaats- en verkeersinformatie. Deze informatie wordt op een geografische manier ontsloten. De gebruiker kan met een simpele inferface zijn weg vinden op de kaart. Er kan in- en uitgezoomd worden, de kaart positie kan verplaatst worden en er kan gekozen worden uit een satelliet- of kaartweergave. Wanneer gezocht wordt naar een plaats of een bedrijf, verschijnen de relevante resultaten op de kaart. Door het klikken hierop verschijnt meer informatie.

De API van Google Maps zit enorm goed in elkaar en is enorm populair. Van de eerder beschreven standaarden en protocollen valt deze onder de 'Javascript API's'. Om gebruik te kunnen maken van de API wordt een Javascript bestand ingeladen, dat gehost wordt op de Google server. Bij het aanspreken van het Javascript bestand wordt een API key meegestuurd. De key geeft per domein toegang om de API te gebruiken en is aan te vragen via de website van Google.

Om de kaart te laten tonen is een HTML container `<div>` nodig met een breedte en hoogte. De container div kan via Javascript vervangen worden door de Google Maps kaart. Een simpel voorbeeld hiervan:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abcdefg"
type="text/javascript"></script>
    <script type="text/javascript">
    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

Bron: Google Maps API documentation

Zoals te zien is wordt een Javascript bestand van de Google server ingeladen. Vervolgens wordt een zelfgemaakte functie load() aangemaakt met daarin een aanroep van de class GMap2, de hoofd class van de Google Maps API. Aan deze class wordt de HTML div met het ID 'map' meegegeven. De API zet deze div om in de kaart en het object 'map' is in Javascript beschikbaar voor verdere functionaliteiten. In bovenstaande voorbeeld wordt bijvoorbeeld de kaart gecentreerd op een specifieke lengte- en breedtegraad. Voorbeelden van andere functionaliteiten zijn: het toevoegen van punten (markers) op de kaart, het toevoegen van lijnen (polygons), control elementen toevoegen of verplaatsen, navigatie opties in- en uitschakelen (inzoomen met scrollwheel), info ballonnen toevoegen aan makers (GInfoWindow) etc.

Voor de officiële API reference kan gekeken worden op:
http://www.google.com/apis/maps/documentation/reference.html

Naast de Javascript API heeft Google voor Google Maps een API beschikbaar gesteld voor het verkrijgen van coördinaten aan de hand van adresgegevens. Door een request te sturen naar het URL http://maps.google.com/maps/geo?q=[ADRESGEGEVENS]output=xml&key=abcdefg worden coördinaten aangeboden van een bepaald adres in XML formaat.

2.4.2 Flickr

Flickr is een website voor het delen van foto's. Daarnaast is het een community en bieden ze verschillende web services aan. Gebruikers krijgen de mogelijkheid foto's te uploaden en deze beschikbaar te stellen aan anderen. Iedere foto kan gekoppeld worden aan een bepaalde categorie. Op deze manier kunnen foto's gegroepeerd worden aan de hand van een gerelateerd onderwerp. Naast de categorie indeling kunnen aan foto's 'tags' worden toegevoegd. Doordat iedere bezoeker de mogelijkheid heeft om deze tags toe te voegen, wordt het zoeken en navigeren gemakkelijker. Om de relevantie en populariteit van een foto te bepalen, is het mogelijk bij iedere foto een beoordeling te geven door bezoekers.

Foto's kunnen gevonden worden op verschillende manieren:

- Zoeken op onderwerp/zoekwoord (in feite wordt gezocht op tag);
- Foto's vinden via andere gebruikers/vrienden;
- Zoeken aan de hand van het merk camera;
- Populaire tags, ontsloten in een tag cloud (Afbeelding 2);
- Zoeken aan de hand van geografische gegevens of vanuit een kaart.

Afbeelding 2

Het community gehalte van Flickr is groot, bezoekers kunnen reageren op foto's, deelnemen aan gebruikersgroepen en op zoek gaan naar vrienden. Daarnaast biedt Flickr ook een API aan, waarmee vrijwel alle functionaliteiten extern te gebruiken zijn.

Flickr is uiterst flexibel in standaarden en protocollen voor het gebruik van de API. Ze bieden namelijk de API in drie verschillende standaarden aan: REST, XML-RPC en SOAP.

De werking via REST
Het aanroepen van het URL
http://www.flickr.com/services/rest/?method=flickr.test.echo&format=rest&foo=bar&api_key=d3abefe361b98305476ef394a46a9dad resulteert in een XML response:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
<method>flickr.test.echo</method>
<format>rest</format>
<foo>bar</foo>
<api_key>d3abefe361b98305476ef394a46a9dad</api_key>
</rsp>
```

De werking via XML-RPC
Het versturen van een request naar het URL http://api.flickr.com/services/xmlrpc/:

```
<methodCall>
  <methodName>flickr.test.echo</methodName>
    <params><param><value><struct>
      <member>
        <name>name</name>
        <value><string>value</string></value>
      </member>
    </struct></value></param></params>
</methodCall>
```

Resulteert in de volgende response:

```
<?xml version="1.0" encoding="utf-8" ?>
<methodResponse>
  <params><param>
    <value><string>
      [escaped-xml-payload]
    </string></value>
  </param></params>
</methodResponse>
```

Werking via SOAP
Versturen van een request naar het URL http://api.flickr.com/services/soap/:

```
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
        <s:Body>
                <x:FlickrRequest xmlns:x="urn:flickr">
                        <method>flickr.test.echo</method>
                        <name>value</name>
                </x:FlickrRequest>
        </s:Body>
</s:Envelope>
```

Resulteert in de volgende response:

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
        <s:Body>
                <x:FlickrResponse xmlns:x="urn:flickr">
                        [escaped-xml-payload]
                </x:FlickrResponse>
        </s:Body>
</s:Envelope>
```

In voorgaande voorbeelden werd de functie of methode flickr.test.echo steeds aangeroepen. Dit is echter een test functie. Een voorbeeld uit de praktijk is het toevoegen van een tag aan een foto via de functie flickr.photos.addTags. Met deze functie worden de parameters *api_key, photo_id en tags* meegezonden voor het toevoegen van *tags* aan de foto met het ID *photo_id*. Een handleiding voor het gebruik van de Flickr API is te vinden op http://www.flickr.com/services/api/.

2.4.3 Youtube

Bij Flickr zagen we al dat de gebruiker een belangrijke rol had. Bij Youtube kan je zeggen dat dit aandeel nog groter is. De zogenoemde term UGC (User Generated Content) wordt vaak in verband gebracht met web2.0. De content wordt geleverd door de gebruiker. Youtube biedt faciliteiten aan voor het beschikbaar stellen van video materiaal. Het gebruik van deze dienst heeft een ongekende groei doorgemaakt. Sinds de start in februari 2005 door drie oud Paypal medewerkers, is het gebruik alsmaar toegenomen. In Oktober 2006 is Youtube uiteindelijk overgenomen door Google Inc. voor een ongekend bedrag van $1,65 miljard (in Google aandelen).

Naast het beschikbaar stellen van video materiaal, kunnen gebruikers met elkaar in contact komen en eigen ‘channels’ maken. Ook het beoordelen van video’s kan gedaan worden. Hierdoor ontstaan functionaliteiten op het gebied van zoeken naar populaire video’s.

Ook Youtube heeft een API beschikbaar gesteld om de videodiensten op de website te integreren. Met de API kunnen gebruikers- en videogegevens opgevraagd worden. Daarnaast is er de mogelijkheid om ieder filmpje gemakkelijk in een eigen website te integreren d.m.v. een aantal HTML tags.

De API maakt gebruik van de standaarden REST en XML-RPC. Een voorbeeld van de REST methode om gebruikers informatie op te vragen gaat op de volgende manier: Request naar het URL http://www.youtube.com/api2_rest?method=youtube.users.get_profile&dev_id=YOUR_DEV_ID&user=YOUTUBE_USER_NAME, dat resulteerd in de volgende response:

```
<?xml version="1.0" encoding="utf-8"?>
<ut_response status="ok">
    <user_profile>
        <first_name>YouTube</first_name>
        <last_name>User</last_name>
        <about_me>YouTube rocks!!</about_me>
        <age>30</age>
        <video_upload_count>7</video_upload_count>
    </user_profile>
</ut_response>
```

Het integreren van een video op een eigen HTML pagina gaat ook eenvoudig. Dit gaat door middel van het plaatsen van de volgende HTML code:

```
<object width="425" height="350"><param name="movie" value="http://www.youtube.com/v/-prfAENSh2k"></param><param name="wmode" value="transparent"></param><embed src="http://www.youtube.com/v/-prfAENSh2k" type="application/x-shockwave-flash" wmode="transparent" width="425" height="350"></embed></object>
```

Hierbij wordt via de <object> en <embed> tag[6] een Flash video aangesproken die gehost wordt op de Youtube server. Deze methode is niet daadwerkelijk een API te noemen, maar is wel een toepassing van web 2.0 (delen van informatie).

[6] Simpel gezegd wordt <object> gebruikt door Internet Explorer en <embed> door Mozilla Firefox.

Hoofdstuk 3:

API’s in de praktijk: Staatopdekaart.nl

3.1 Inleiding

Nu verschillende protocollen en technieken naar voren zijn gekomen en beschreven is wat web2.0 is, kan de stap naar de praktijk gemaakt worden. Voor de ondersteuning van deze scriptie is het project Staatopdekaart.nl gerealiseerd. Er is eerder al omschreven wat Staatopdekaart.nl precies is. In dit hoofdstuk wordt het project in detail besproken. Aan bod komen de uitgangspunten, het functioneel ontwerp, vormgeving, implementatie, beschrijving van het systeem en de evaluatie.

Wanneer deze scriptie gelezen wordt kunnen er uitbreidingen gedaan zijn aan de site. De website is zoals in de web2.0 gedachte constant in ontwikkeling. Nieuwe functionaliteiten worden wekelijks toegevoegd en online (live) getest.

3.2 Uitgangspunten

Voorafgaand aan het project zijn een aantal uitgangspunten vastgesteld, welke beschreven worden in onderstaand overzicht.

3.2.1 Locaties

Locaties worden ontsloten op een Google Maps kaart. Om deze locaties navigeerbaar te maken komt er een categorie structuur die als volgt is ingedeeld:

- **Hoofd thema's**
  Hoofd thema's waren in eerste instantie steden. Later werden dit ook andere thema's (zoals attractieparken en trouwlocaties). Het systeem moest daarom de mogelijkheid bieden om thema's te maken. Voorbeelden van thema's: Hilversum, Amsterdam, Attractieparken, Trouwlocaties. Per thema kunnen kleuren, advertenties en afbeeldingen bepaald worden.
- **Hoofd categorieën**
  De hoofd categorieën zorgen voor de navigatie binnen de kaart applicatie. Een bepaald thema kan op deze manier ingedeeld worden in verschillende categorieën. Iedere categorie heeft een eigen kleur. Voorbeeld categorieën zijn: Vervoer, cultuur, vermaak.
- **Sub categorieën**
  Hoofd categorieën kunnen onderliggende 'sub' categorieën bevatten. Op deze manier wordt de ontsloten informatie specifieker. Iedere sub categorie krijgt een icoon in combinatie met de kleur van de hoofd categorie. Als voorbeeld de onderliggende sub categorieën van de categorie 'vervoer': parkeergelegenheden, fietsverhuur, stations.
- **Locaties**
  Sub categorieën bevatten de daadwerkelijke locaties. De locaties worden vertoond in de categorie lijst en op de kaart zelf met punten. Voor de punten op de kaart wordt het icoon van de sub categorie overgenomen met daarbij de kleur van de hoofd categorie.

Door de locaties in categorieën in te delen wordt het navigeren naar de juiste interesse vergemakkelijkt. Tevens komt er de mogelijkheid om meerdere categorieën tegelijk te selecteren door middel van een checkbox. Hierdoor wordt het mogelijk meerdere categorieën tegelijk te zien op de kaart. Als voorbeeld iemand die een dag naar Amsterdam wilt om cultuur te proeven. Deze persoon gaat met de auto. Hiervoor is het handig de categorieën 'cultuur' en 'parkeergelegenheden' gelijkertijd te selecteren.

De locaties zelf kunnen worden voorzien van adresgegevens, foto's, korte omschrijving, lange omschrijving, video's en een website adres.

3.2.2 Techniek en comptabiliteit

De applicatie wordt opgemaakt met behulp van XHTML en CSS. Daarnaast wordt Javascript gebruikt voor interactieve en functionele elementen. Om het werken met Javascript gemakkelijker te maken wordt gebruik gemaakt van het Javascript framework 'Prototype JS'. Voor het gemakkelijk kunnen toepassen van effecten wordt de library script.aculo.us gebruikt. Daarnaast wordt content asynchroon verwerkt in Javascript doormiddel van het AJAX principe. Voor de uitwisseling van content wordt gebruik gemaakt van PHP, dat XML aanbiedt aan Javascript.

De applicatie dient te kunnen werken op PC in Internet Explorer 6 en 7, Mozilla Firefox 1.4 en hoger, Opera 9 en op Apple Mac in Safari, Camino en Firefox.

3.2.3 Content productie via Excel
Het creëren van content (locaties) dient te geschieden via Microsoft Excel. Dit is een vertrouwd software pakket voor de makers van de content. Om deze manier van content productie mogelijk te maken is het van belang een template te maken van een Excel document waarin de locaties aangegeven kunnen worden. Dit bestand kan vervolgens via een importeer script geïmporteerd worden in de database middels CSV.

3.2.4 Verschillende doelen
De applicatie heeft verschillende doelen. Allereerst moet het een leuke omgeving worden voor site bezoekers die op zoek zijn naar bepaalde locaties. Daarnaast heeft de applicatie als doel om een voorbeeld te zijn voor de klanten van Gather BV. De applicatie geeft op deze manier een indruk van de mogelijkheden van Google Maps en prikkelt (potentiële) klanten wellicht om ook Google Maps toepassingen in te gaan zetten. Het laatste doel van de applicatie is om een testomgeving te creëren waarin de mogelijkheden van de Google Maps API geëxperimenteerd kunnen worden.

3.2.5 Advertentie mogelijkheden
De website moet mogelijkheden bieden voor advertenties. Advertenties zijn afkomstig van Google Adwords en kunnen opgenomen worden in verschillende formaten. Er is geen doel gesteld voor het genereren van directe advertentie inkomsten. Echter zou het een leuke bijkomstigheid zijn wat inkomsten te generen en te kunnen analyseren via welke kanalen de beste conversie behaald wordt (bijvoorbeeld kunnen zien dat het zoekwoord 'plattegrond' uit Google de meeste conversie, in dit geval banners clicks, opgeleverd heeft).

3.2.6 Vindbaarheid
Door het veelvuldig gebruik van Javascript en het inladen van content via AJAX wordt de website niet goed toegankelijk voor zoekmachines. Daarom is het van belang een aantal optimalisaties toe te voegen die de website beter vindbaar maken voor zoekmachines. Tot de oplossingen behoren onder andere:

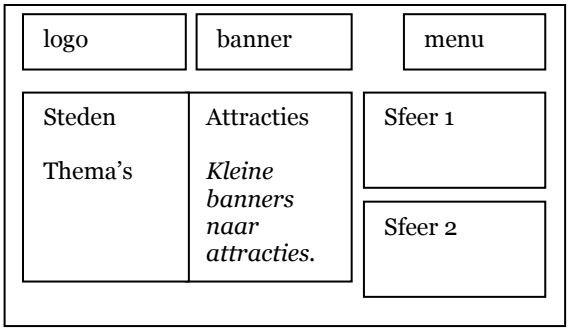
- toevoegen van duidelijke pagina titels en descriptions per thema;
- toevoegen van HTML elementen met daarin informatie speciaal voor zoekmachines;
- speciale landingspagina's met tekst die vindbaar is voor zoekmachines.

3.2.7 Web 2.0 gedachtes toepassen
De eerdere beschreven web 2.0 gedachtes dienen doorgevoerd te worden in de applicatie. Concreet betekend dit dat de gebruiker de mogelijkheid heeft iets te delen. Dit gebeurd met de route planner. Bezoekers kunnen zelf hun route intekenen en deze doorsturen naar vrienden via e-mail. Daarnaast ligt de manier van ontwikkelen ook in de lijn van web 2.0. In stapjes wordt de applicatie verder ontwikkeld. Iedere update wordt direct online gezet en wordt gekeken of de nieuwe functionaliteit aanslaat. Tot slot heeft de applicatie een enorme gebruikservaring, doordat gebruik wordt gemaakt van AJAX en verschillende Javascript effecten.

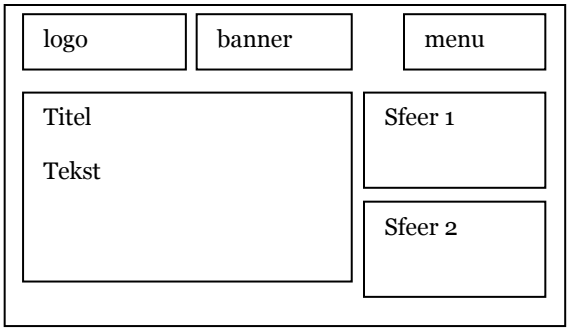
3.3 Functioneel ontwerp
Voorafgaand zijn een aantal schermen uitgewerkt in een functioneel ontwerp. Het belangrijkste functionele idee was dat de Google Maps kaart schermvullend was, met daaroverheen de inklapbare menu's met navigatie- en interactie functionaliteiten.

3.3.1 Homepage
Hier wordt de keuze gemaakt van een bepaald thema. De homepage bevat veel teksten en is opgebouwd uit semantische HTML. Dit zorgt voor een goede vindbaarheid in zoekmachines.

Homepage

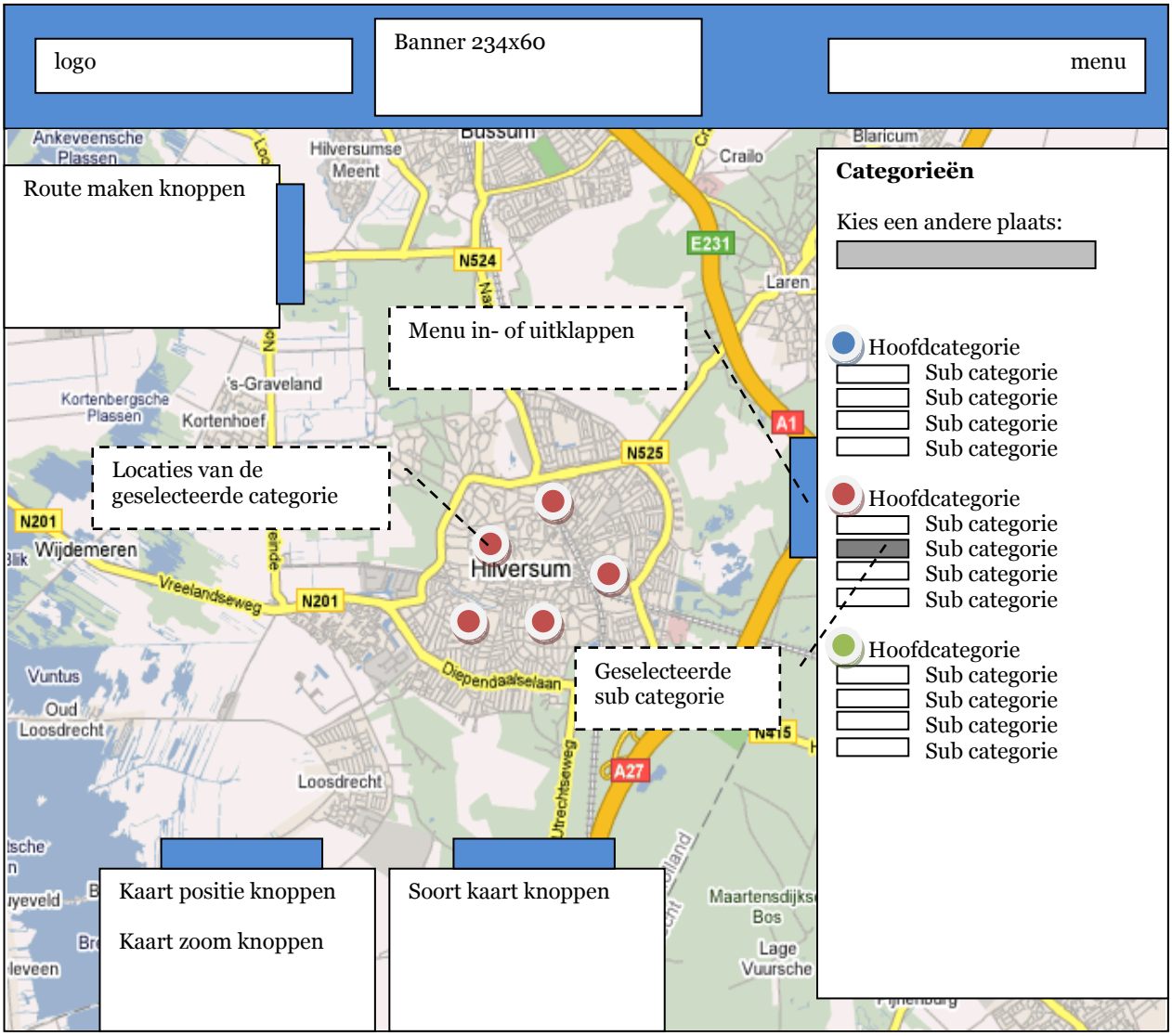


Informatie pagina



3.3.2 Kaart applicatie

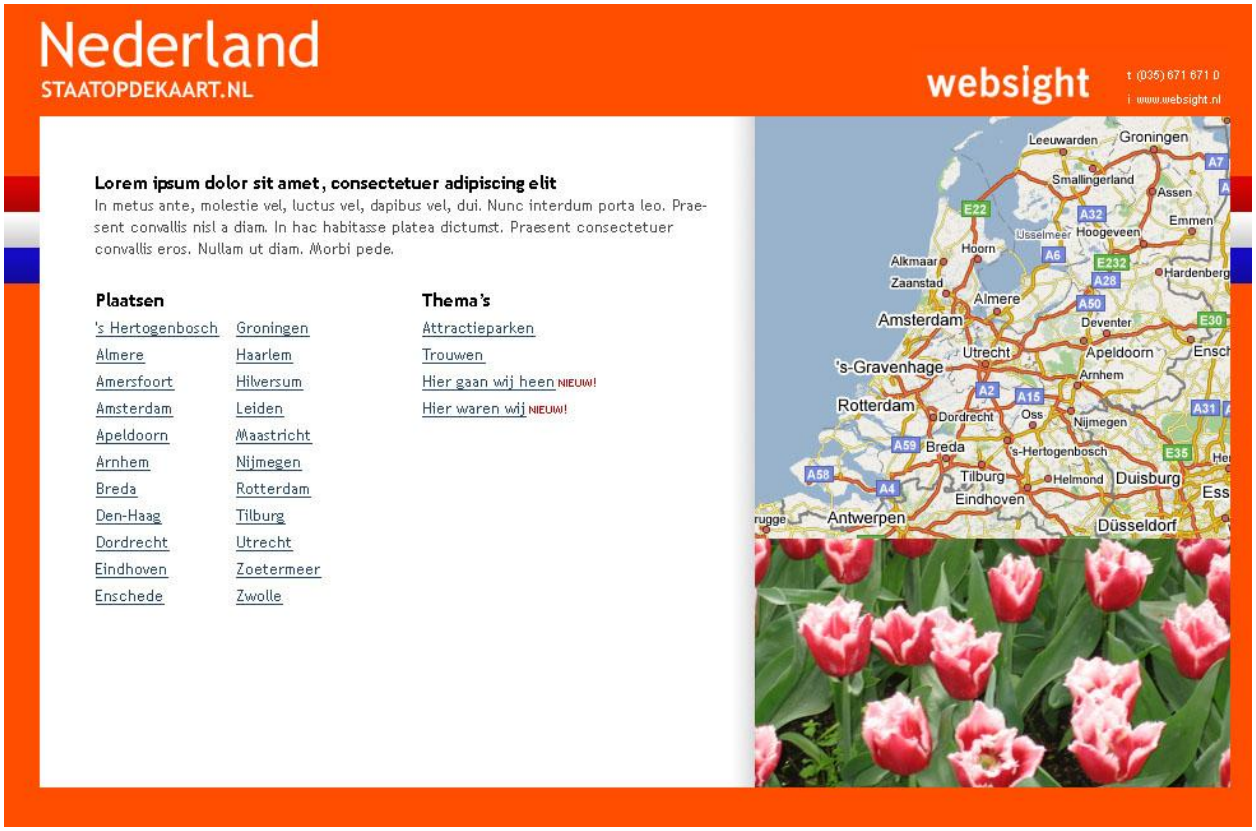
Nadat gekozen is voor een bepaald thema (of een plaats) verschijnt de kaart applicatie. Voor ieder thema is een sub domein beschikbaar ([thema].staatopdekaart.nl).



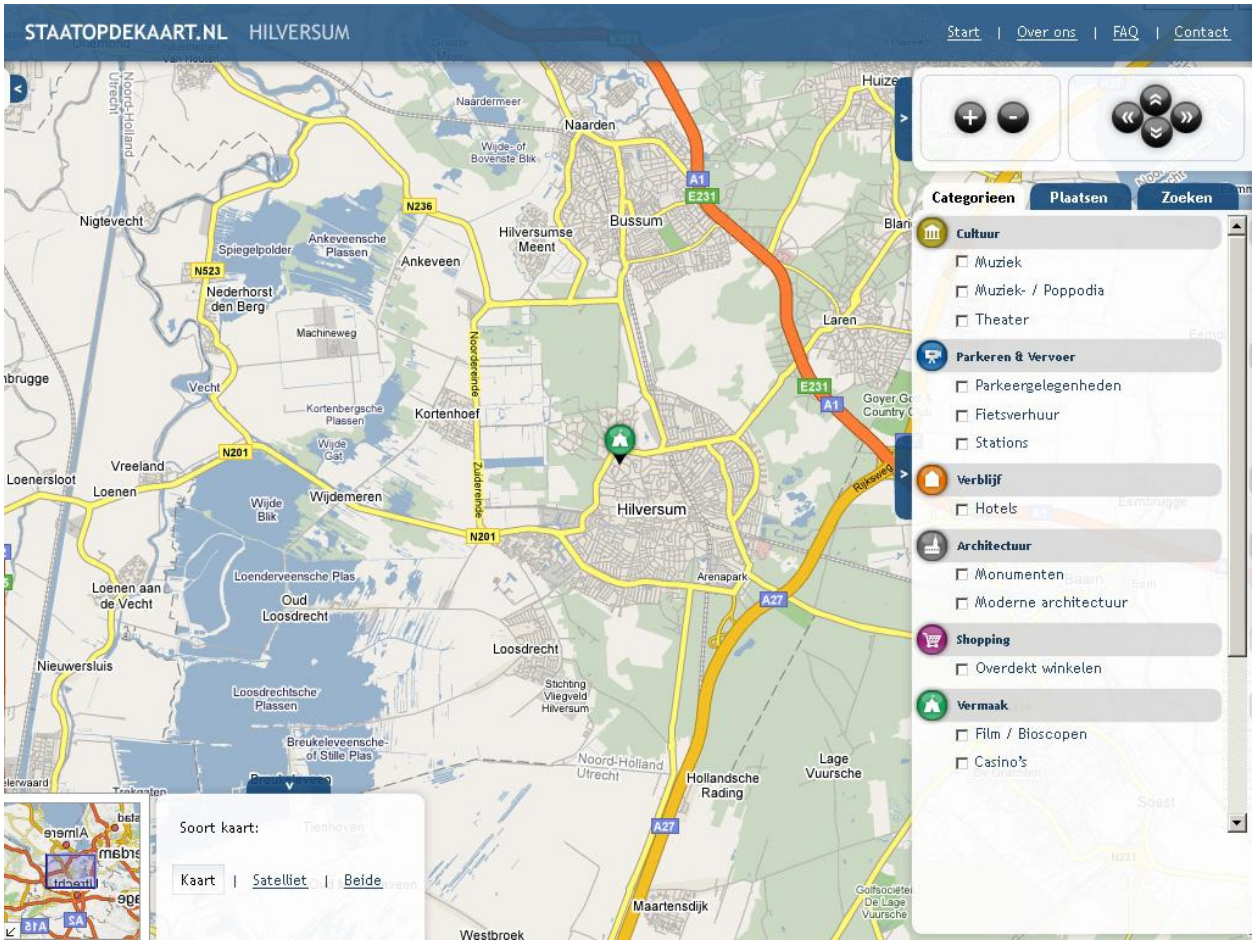
3.4 Vormgeving

Op basis van het functioneel ontwerp is een vormgeving gemaakt in Photoshop. De schermen die uitgewerkt waren, zijn:

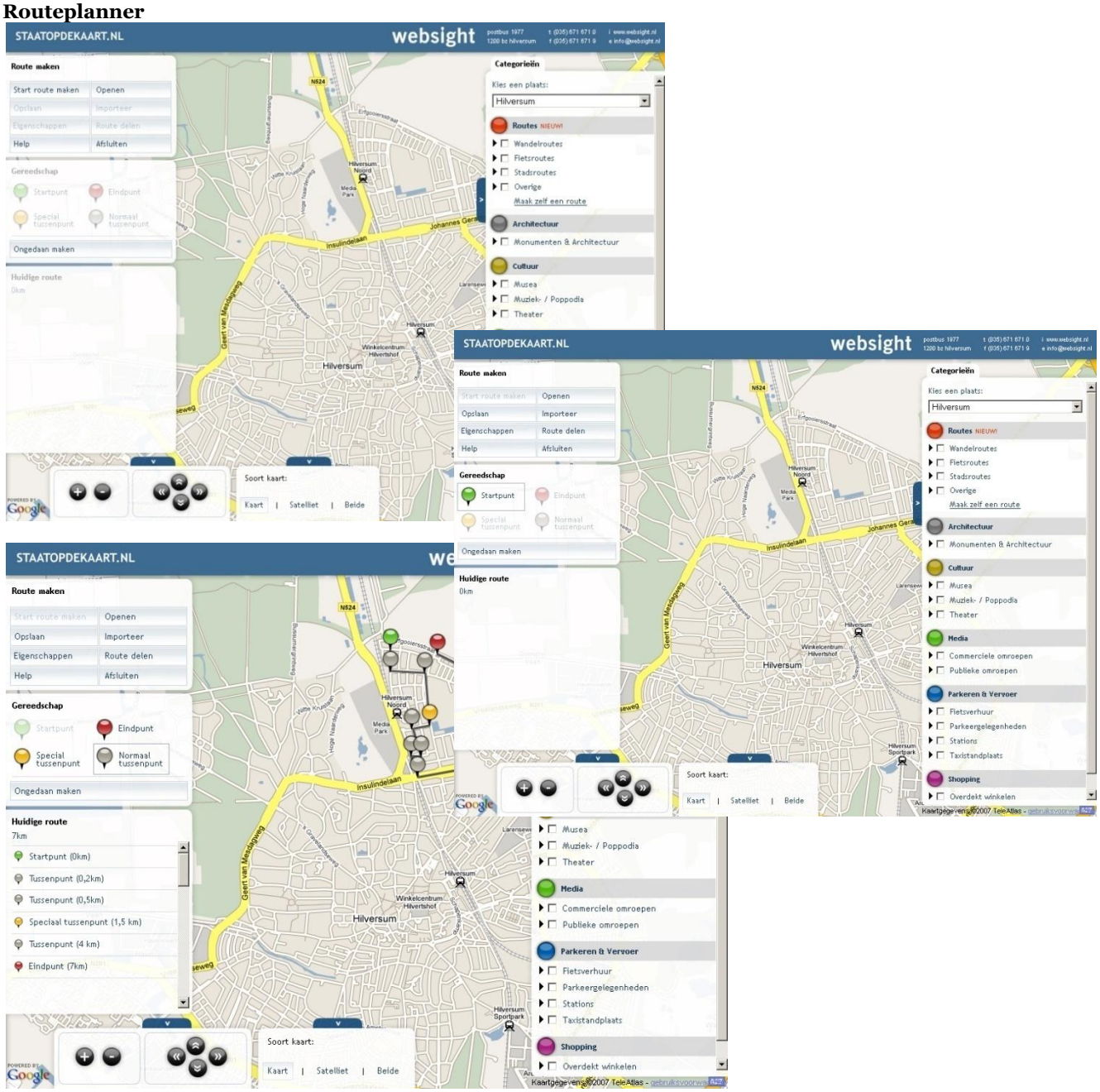
Homepage



Kaart (eerste ontwerp)



Routeplanner



3.5 Usability

De belangrijkste eigenschap op het gebied van usability is het gebruik van 'windows' (layers). Dit zijn de lagen die over de schermvullende kaart heen liggen. Met dit als uitgangspunt worden functionaliteiten aan de kaart toegevoegd. Door de beperkte ruimte in lagere scherm resoluties kunnen windows ingeklapt worden en is het mogelijk dat meerdere windows over elkaar heen te zien zijn. Op deze manier wordt optimaal gebruik gemaakt van de beschikbare ruimte, zonder dat het belangrijkste deel, de kaart, verwaarloosd wordt. Verder is er voor gekozen de categorisering aan de rechterkant te laten plaatsvinden. Doordat meerdere categorieën gelijktijdig geselecteerd kunnen worden via een checkbox, heeft dit window het idee van 'filter opties'. Dit is aan de rechterkant geplaatst, omdat filter opties qua logica rechts horen.

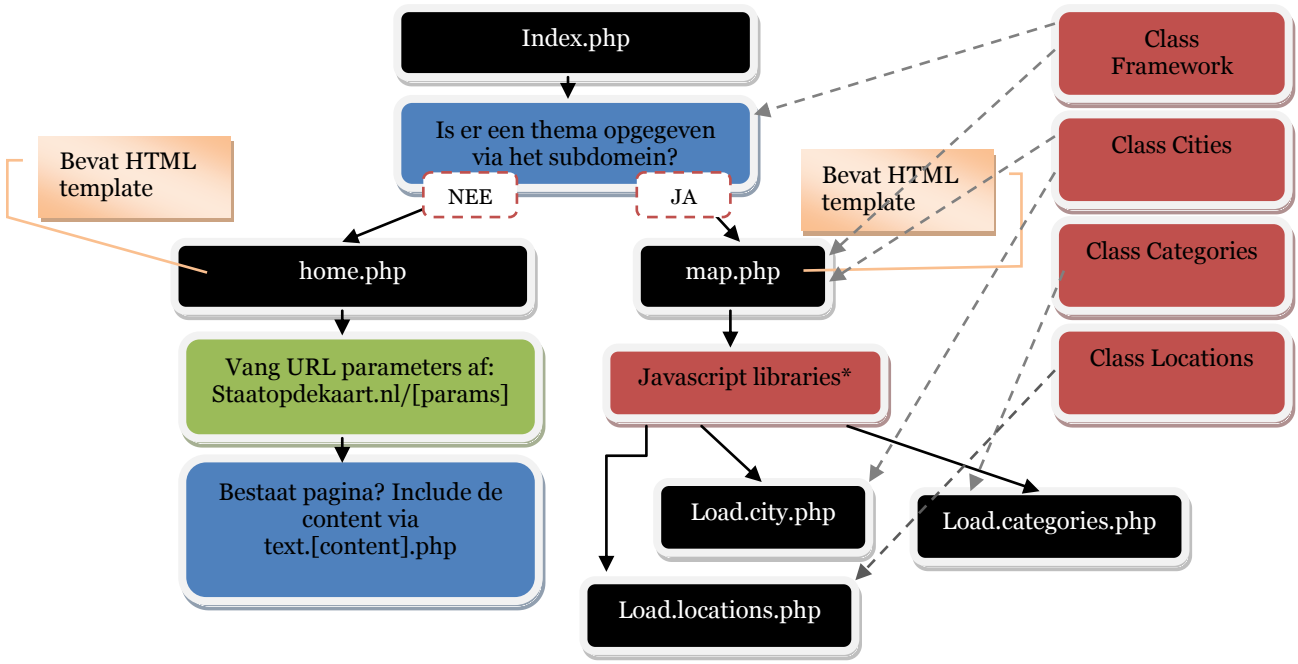
Onder de categorieën vallen de locaties. Deze zijn te zien door een categorie 'uit te klappen'. Om dit te doen is in de eerste versie gekozen voor een icoontje met een pijl. Er bleek echter dat veel mensen dit niet snapte. Daarom is later dit icoontje aangepast naar een plus- en min icoon. Dit bleek wel duidelijk te zijn, omdat het refereerde aan een map uitklappen in de windows verkenner.

3.6 Beschrijving van het systeem

In deze paragraaf worden een aantal belangrijke onderdelen van het systeem toegelicht. Hierbij wordt de werking aangegeven en worden bepaalde keuzes verantwoord. Allereerst zal een schematische weergave van het systeem worden gegeven.

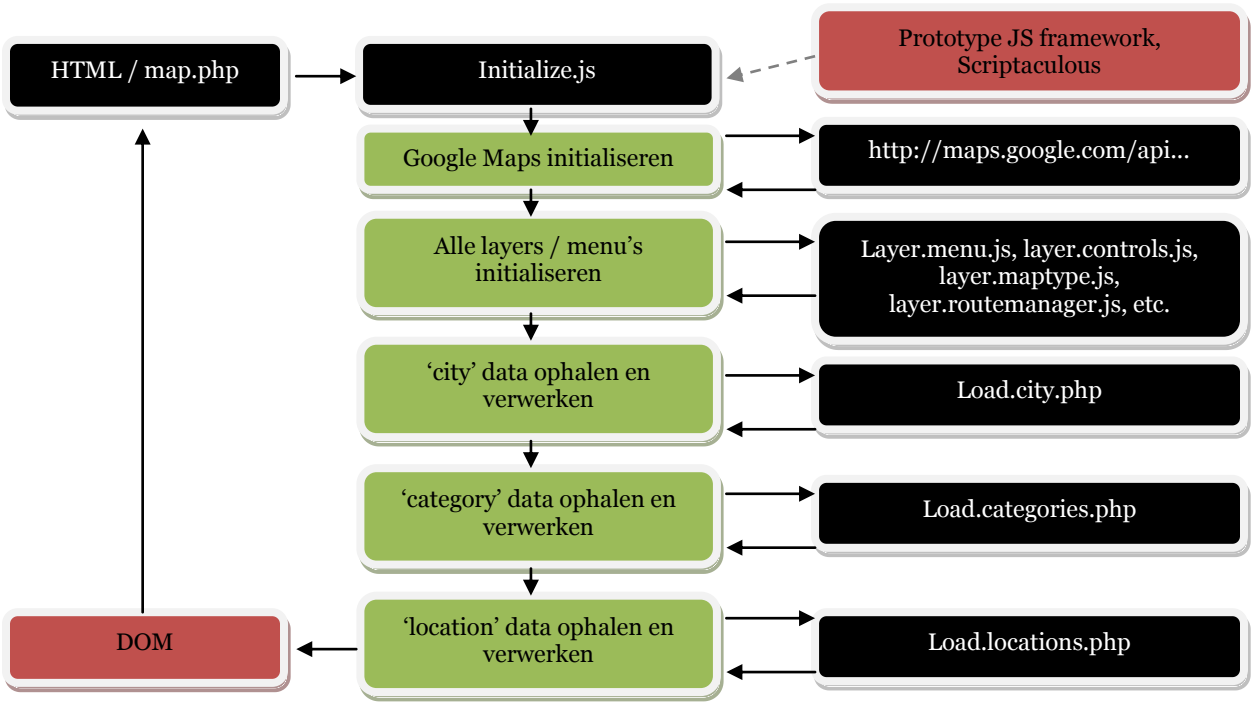
3.6.1 Schematische weergave PHP

De serverside afhandeling verloopt als volgt via PHP:



3.6.2 Schematische weergave Javascript libraries

Het specifieke onderdeel 'Javascript libraries' uit bovenstaand schema (aangegeven met *) wordt in onderstaand schema beschreven.



3.6.3 Toelichting classes / onderdelen

Load.inc.php

- Wordt overal geinclude;
- Alle classes laden;
- Database connectie maken.

Framework class

- Hoofdclass;
- Bevat de belangrijkste data, data uit andere classes wordt naar deze class gecommuniceerd;
- Vang het subdomein af en controleert of dit een bestaande 'city' is;
- Indien een geldig subdomein is opgegeven `boolean show_homepage = false` en anders `boolean show_homepage = true`;
- Vang URL parameters af (http://[subdomein].staatopdekaart.nl/[parameters]);
- Afhankelijk van meegegeven parameters kan een individuele categorie, locatie of route actief worden. Indien de homepage zichtbaar is, kunnen via de parameters andere tekst pagina's geladen worden.

Cities class

- Haal huidige city op en communiceer data naar Framework;
- Haal overzicht van alle cities uit database en communiceer naar Framework;
- Haal de coördinaten van een bepaalde city_id op.

Categories class

- Haal 1 specifieke categorie op;
- Haal alle categorieën en sub categorieën op.

Locations class

- Haal alle locaties van 1 category op.

Layer.*.js

- Bevat een init functie;
- Bevat een onresize functie (wordt getriggerd bij body onresize);
- Maakt een layer in- en uitklapbaar;
- Voegt events toe aan de buttons binnen een layer.

Initialize.js

- Initialiseert de Google Maps API;
- Roept alle init functies aan;
- Roept data functies aan voor het ophalen van benodigde XML data uit PHP;
- Initialiseert alle events;
- Laadt alle standaard waardes (kaart positie, zoom level) in en verwerkt deze op de kaart.

3.6.4 Database

Zie bijlage A

3.6.5 Belangrijke keuzes toegelicht en interessante details

Waarom wordt de benaming 'cities' (tabel en class) gebruikt?
De eerste ideeën waren om alleen uit te gaan van plaatsnamen. In de eerste versie was dit ook het geval en werd de benaming 'cities' in de database en classes gehanteerd. Later kwamen er ook onverwachts thema's bij. Door tijdsgebrek is de benaming intern hetzelfde gelaten.

Eigen marker manager
Markers zijn de punten die op de kaart aangegeven worden (locations). Binnen de Google Maps API bestaat een speciale class genaamd MarkerManager. Hiermee kunnen grote hoeveelheden markers op de kaart getoond worden. Echter ondersteund de MarkerManager niet het onzichtbaar maken van markers en de verdeling in categorieën. Daarom is er voor gekozen een eigen MarkerManager te bouwen die dit wel ondersteund.

De keuze voor scriptacolous
Om gemakkelijk effecten te kunnen realiseren binnen Javascript is gekozen voor de library Scriptacolous. Een effect (move) kan gemakkelijk gedaan worden op een Block element via:

```
// verplaats block element naar positie 100, 100
Effect.Move('element_id', {x: 100, y: 100});
```

Op deze manier kunnen effecten zeer gemakkelijk gerealiseerd worden. Met het oog op tijd was het vrijwel niet interessant om zelf deze functionaliteiten te ontwikkelen.

Keuze voor PHP / MySQL
Bij Gather is PHP / MySQL de gebruikelijke ontwikkel omgeving. Voor andere platformen en omgevingen zijn geen faciliteiten.

Waarom gebruik gemaakt wordt van AJAX
Door de enorme hoeveelheid data is het enorm interessant om losse delen data pas in te laden wanneer ze nodig zijn. Tevens is het voor een aantal doeleinden goed voor de gebruikservaring. Wanneer iemand een route gemaakt heeft wordt deze via javascript 'gepost' naar PHP (AJAX), op deze manier hoeft niet een complete pagina herladen te worden.

Tabel 'types'
Om records uit de tabel 'cities' te kunnen bundelen is er een tabel types. Op deze manier kunnen 'cities' gebundeld worden met een zelfde style en naamgeving.

Route maken: achtervolgende transparante lijn
Bij het maken van een route is er de mogelijkheid om punten te zetten. Bij het bewegen van de muis wordt het punt achtervolgd door een transparante lijn en marker. Voor iedere beweging op de 'map' wordt een event mapMove getriggerd. Vervolgens gebeurd het volgende:

```
function mapMove (latlng)
{
	if (routeEdit && !routeEnd)
	{
		if (currentRoute[0])
		{
			// line
			if($('showPolygon').checked)
			{
				map.removeOverlay(moveLine);
				var lastpoint = currentRoute[currentRoute.length-1];
				var arr = Array();
				arr.push(lastpoint);
				arr.push(latlng);
				moveLine = new GPolyline(arr, '#0E3152',3,0.4);
				map.addOverlay(moveLine);
			}

			// marker
			map.removeOverlay(moveMarker);
			var active_trans = routeMarkerActive+"_trans";
			moveMarker = new GMarker(latlng, routeMarkersIcons[active_trans],
{clickable: false});

			map.addOverlay(moveMarker);
		}
	}
}
```

Cookies in Javascript
Om in Javascript gebruik te maken van cookies worden een drietal handige functies gebruikt:

```
function createCookie (name,value,days)
{
	if (days)
	{
		var date = new Date();
		date.setTime(date.getTime()+(days*24*60*60*1000));
		var expires = "; expires="+date.toGMTString();
	}
	else var expires = "";

document.cookie = name+"="+value+expires+"; path=/; domain=staatopdekaart.nl";

}

function readCookie (name)
{
	var nameEQ = name + "=";
	var ca = document.cookie.split(';');
	for(var i=0;i < ca.length;i++)
	{
		var c = ca[i];
		while (c.charAt(0)==' ') c = c.substring(1,c.length);
		if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
	}
	return null;
}

function eraseCookie (name)
{
	createCookie(name,"",-1);
}
```

3.6.6 Problemen en oplossingen

Transparante PNG's in Internet Explorer 5.5 / 6.0

Internet Explorer 6.0 en lager ondersteunen geen transparante PNG afbeeldingen. Omdat dit wel gebruikt wordt in de inklapbare menu's moest dit probleem opgelost worden. In Internet Explorer 6.0 en lager bestaat de mogelijkheid voor 'filters'. Hiermee gebruik makende is het probleem als volgt opgelost:

```
function iePng (obj)
{
	var image = obj.style.background.substr(4, (obj.style.background.length-5));


	if (BrowserDetect.browser == 'Explorer' && BrowserDetect.version == 6)
	{
		if (obj.filters['DXImageTransform.Microsoft.AlphaImageLoader'])
		{
			obj.style.background = 'none';
			obj.style.filter =
'progid:DXImageTransform.Microsoft.AlphaImageLoader(src="../'+image+'")';
		}
	}
}
```

Friendly URL namen

Voor betere indexatie door zoekmachines wordt gebruik gemaakt van 'friendly URL's'. Vaak moet de titel van een item hierin voorkomen. Deze titel mag geen speciale tekens bevatten. Om de titel op te schonen t.b.v. het URL wordt gebruik gemaakt van de volgende PHP functie:

```
function safe_string ($input, $dont_use_strtolower = false)
{
	$output = null;
	$input = str_replace (" ", "_", $input);

	for($i = 0; $i < strlen ($input); $i++)
	{
		if ((ord ($input{$i}) >= 65 && ord ($input{$i}) <= 90) OR (ord
($input{$i}) >= 97 && ord ($input{$i}) <= 122) OR (ord ($input{$i}) >= 48 && ord
($input{$i}) <= 57) OR    (ord ($input{$i}) == 95))
		{
			$output .= $input{$i};
			$output = str_replace ("__", "_", $output);
		}
	}

	if ($dont_use_strtolower) return $output;
	else return strtolower ($output);
}
```

Hoofdstuk 4:

Commerciële waarden van web API’s

Veel succesverhalen van web 2.0 verdien modellen zijn er nog niet. De één ziet de toekomst somber in, de ander is razend enthousiast. De complete discussie rondom verdien modellen en web 2.0 is een discussie op zich en te omvangrijk om in deze scriptie aan bod te laten komen. Echter kijkend naar het specifieke onderdeel web API's, kunnen wél een aantal ideeën van verdien modellen en commerciële aspecten geschetst worden. In dit hoofdstuk zullen er een aantal omschreven worden met daarbij voorbeelden uit de praktijk. Gekeken wordt vanuit het oogpunt van de API producenten en de API gebruikers.

4.1 Commerciële waarden voor API producenten

Het aanbieden van een web API kan tot een aantal commerciële aspecten leiden, welke beschreven worden in deze paragraaf.

4.1.1 Betaalde web API's

Door een gratis API beschikbaar te stellen, worden andere ontwikkelaars enthousiast om deze API te gebruiken binnen eigen diensten. Vaak zie je bij gratis API's een limiet van het aantal aanvragen per dag. Wanneer dit limiet overschreden wordt kan gebruik gemaakt worden van een commerciële API. Een voorbeeld hier van uit de praktijk is de Google Maps API. Deze API staat het toe om maximaal 50.000 aanvragen per dag te doen. Wanneer meer aanvragen gewenst zijn kan overgestapt worden naar een commerciële, betaalde, API. Ook zijn er modellen waarbij, naast een gratis API met beperkte functionaliteiten, een commerciële API beschikbaar wordt gesteld met daarin extra functionaliteiten. Een ander voorbeeld van een betaalde API is de S3 (simple storage service) van Amazon. Met deze API kan gebruik worden gemaakt van de opslag capaciteit van Amazon via een API. Betaald worden de daadwerkelijk gebruikte Gigabytes, voor een zeer interessante prijs ($0,15 per GB opslag).

4.1.2 Brandgagement

'Het opzettelijk betrekken van je klanten, bij je innovatie, merkbouw en/of marketing'. Moderne bedrijven nodigen klanten nadrukkelijk uit om hun wensen en ideeën kenbaar te maken. Het uitnodigen van deze klanten gaat uiteraard verder dan 'het ophangen van een ideeënbus'. Bedrijven laten hun klanten écht deelnemen aan hun bedrijfsprocessen innovatie, merkbouw en marketing. Brandgagement heeft niet alleen betrekking op consumenten. Zakelijke afnemers, groothandels en detailhandels (Business 2 business) zijn ook zeer interessant. Een simpel voorbeeld hier van is het laten insturen van zelf gemaakte commercials van een bedrijf. Koppel hier een wedstrijd aan, waarbij de winnende commercial daadwerkelijk op televisie vertoond zal worden.

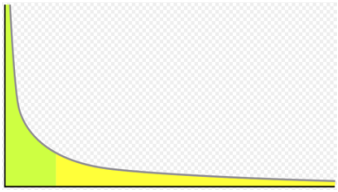
Het beschikbaar stellen van een API is geheel in de lijn van brandgagement. De API zorgt ervoor dat klanten betrokken worden bij de innovatie en merkbouw. Door het beschikbaar stellen van een API worden anderen aangespoord hier innovatieve dingen mee te doen. Nieuwe functionaliteiten worden op deze manier door anderen verzonnen en toegepast in mashups. De producent van de API kan deze ideeën gebruiken voor een eigen dienst. Bijvoorbeeld de Google Maps API. Veel ontwikkelaars maakten al snel routeplanners in combinatie met de Google Maps API. Google pikte dit op en heeft vervolgens deze functionaliteiten doorgevoerd in haar eigen Google Maps applicatie met het onderdeel 'my maps'.

Bij het beschikbaar stellen van een API kan ook gedacht worden aan het plaatsen van de merknaam binnen de applicatie (bijvoorbeeld het Google logo tonen op de kaart). Dit zorgt er voor dat de merknaam steeds herkenbaar terug komt.

4.1.3 Advertenties in API

Er kan gekozen worden om inkomsten te generen met advertenties binnen een API. Ontwikkelaars die gebruik maken van een API krijgen dan advertenties binnen hun eigen applicatie te zien. In ruil voor het gebruik van de API worden advertenties getoond. Een extra mogelijkheid is om advertentie inkomsten binnen een API te delen. Alle inkomsten uit bepaalde advertenties kunnen dan verdeeld worden over de producent van de API en de gebruiker van de API. Bijvoorbeeld Microsoft heeft deze plannen voor hun kaart API: "Tom Bailey (Virtual Earth) said Microsoft also is planning to put ads on maps and share revenue with mashups".

4.1.4 Inkomsten van eigen diensten vergroten via API's



Grote spelers als Ebay en Amazon verdienen geld met het verkopen of verruilen van producten. The Long Tail theorie van Chris Anderson, hoofdredacteur Wired, is volledig van toepassing op deze twee partijen. The Long Tail beschrijft 'waarom we

in de toekomst minder verkopen van meer'. Door Internet zijn opslag capaciteit en distributiekosten verwaarloosbaar. Waar voorheen grote loodsen van toepassing waren met kostbare distributiekanalen, praten we in het Internet tijdperk over goedkope bandbreedte en goedkope virtuele opslagcapaciteit. Ieder product dat online aangeboden wordt staat nu immers in een database, in tegenstelling tot in een groot pakhuis. Hierdoor wordt het mogelijk om grote hoeveelheden producten aan te bieden, zelfs de kleinere niches kunnen aangeboden worden. Laat nu net deze minder populaire producten (de tail), bij elkaar, de helft of meer van de totale omzet bedragen, naast de omzet van de populaire (head) producten.

Het beschikbaar stellen van API's voor het verkopen of verruilen van producten kan daarom erg interessant zijn. Ontwikkelaars kunnen deze API bijvoorbeeld gaan gebruiken in een specifieke niche (CD's van de onbekende Klaas X uit 1963, met nog een kleine fanatieke fangroep). Een ontwikkelaar maakt hiervoor een mashup waarmee deze CD's op een gemakkelijk manier gekocht kunnen worden via Amazon. Doordat de fangroep erg fanatiek en hecht is bestellen relatief veel mensen uit deze groep de CD's. Ditzelfde gebeurd bij andere niches. Bedenk dat dit bij elkaar een enorm deel van de omzet kan betekenen.

Omzet kan dus ook via API's verlopen, wat met bovengenoemde constateringen erg interessant kan zijn. Ter illustratie: 28% van de omzet van Amazon komt via derde partijen. Een groot deel hiervan verloopt via de API's van Amazon (helaas wil Amazon niet kenbaar maken hoe groot dit percentage exact is). Een ander voorbeeld is Salesforce.com, leverancier van CRM systemen. 40% van het netwerk verkeer verloopt via hun API's.

4.1.5 Verbeteren van bedrijfsprocessen en productiviteit

Complete bedrijfsprocessen kunnen afzonderlijk beschikbaar worden gesteld via een API. Doordat deze bedrijfsprocessen gemakkelijk te integreren zijn voor derden kan dit geld besparen. Bijvoorbeeld een leverancier kan het assortiment via een API beschikbaar stellen. Doordat gebruik gemaakt wordt van bepaalde standaarden, kan het assortiment van de leverancier gemakkelijk geïntegreerd worden in de applicaties van de afnemers. Tevens kan het bestellen via een API gebeuren. Hierdoor hoeven deze processen niet handmatig gedaan te worden, maar kunnen ze volledig geautomatiseerd worden. Het integreren van een API gaat gemakkelijk door het gebruik van standaarden, dit scheelt tijd en uiteindelijk dus geld.

4.1.6 Beschikbaar stellen van data

"Content is king". Partijen met grote hoeveelheden content en data zijn machtig. Door de komst van web API's hebben bepaalde partijen hun voordeel hiermee kunnen doen. Bijvoorbeeld Navteq, een bedrijf dat kaart informatie en satellietbeelden beschikbaar heeft. Sinds de komst van Google Maps, die gebruik maakt van deze data, is de omzet (in 2006) van Navteq met 26% gestegen ten opzichte van dat jaar ervoor. Er kan gesteld worden dat bedrijven met grote hoeveelheden data, met eventueel gekoppeld daaraan een commerciële API, winstgevender kunnen worden.

4.2 Web API's gebruiken

Naast de commerciële aspecten voor API producenten zijn er voor de gebruikers ervan ook een aantal te noemen. In deze paragraaf komen ze aan bod.

4.2.1 Eigen diensten verrijken met een API

Een bestaande dienst kan verrijkt worden met een externe API. Op deze manier kan op een goedkope en eenvoudige manier de dienst uitgebreid worden met extra functionaliteiten. Teven kan in een aantal gevallen de bestaande dienst populairder en/of winstgevender worden. Als voorbeeld Marktplaats. Hierin is een koppeling gemaakt van aangeboden producten met Google Maps. Op de kaart worden aangeboden producten getoond. Op deze manier is de huidige manier van zoeken naar producten, via categorieën, verrijkt met een geografische manier van zoeken. Door de eenvoudige API van Google kon dit op een snelle manier geïmplementeerd worden. Het was voor Markplaats namelijk niet nodig om zelf een kaart applicatie te ontwikkelen.

4.2.2 Advertenties

Goedkoop verschillende API's gebruiken en hiervan een leuke mashup maken. De mashup wordt positief ontvangen en het gebruik ervan neemt toe. Geld kan verdiend worden door in de mashup advertenties te verwerken. Wel moet hierbij op de voorwaarden van de API's gelet worden. Sommige API's verbieden het gebruik ervan te combineren met advertenties of andere commerciële doeleinden.

4.2.3 Marketingtool

De populariteit van de Google Maps API was enorm toen deze gelanceerd werd. De toepassingen met deze API schoten als paddenstoelen uit de grond. Tim Hibbard maakte wel een hele special toepassing. Op de website Where's Tim Hibbard kon live gezien worden, in een Google Maps kaart, waar Tim was. Dit was mogelijk doordat hij altijd een GPS apparaat bij zich had, dat constant zijn positie doorgaf. Deze speciale toepassing was enorm populair en de website werd vele malen bezocht en stond op veel websites tussen de populairste Google Maps Mashups. Plotseling verschenen er advertenties op de website, van een bedrijf dat zich bezig houdt met GPS software. Achteraf bleek dat Tim Hibbard bij dit bedrijf werkte en dat het bedrijf dit als ludieke marketing actie had bedacht. Hieruit blijkt dat het gebruik van web API's ook ingezet kan worden als marketingtool.

4.2.4 API / mashup consultants

Het ontwikkelen van mashups zou aangeboden kunnen worden als dienst. Toepassingen kunnen dan gebouwd worden voor anderen die geen inhoudelijke kennis hebben van het ontwikkelen met verschillende API's.

Hoofdstuk 5:

Conclusie en aanbevelingen

5.1 Conclusie

5.1.1 Wat zijn de technische- en commerciële waarden voor API producenten?

Web 2.0 en web API's staan nauw met elkaar in verband. Bij web 2.0 wordt getracht een platform te creëren en consumenten volledig bij de dienst te betrekken. Web API's liggen volledig in deze lijn. Het betrekken van de consument wordt gedaan door andere ontwikkelaars de mogelijkheid te bieden eigen diensten te maken op basis van de API. Doordat verschillende API's met elkaar gekoppeld kunnen worden ontstaat één groot platform.

Bij het aanbieden van een web API is de achterliggende architectuur van de applicatie ook van belang. Er zijn verschillende stijlen van architectuur: SOA, RPC en REST. Belangrijk uitgangspunt hierbij is de schaalbaarheid en onafhankelijkheid. Verschillende componenten binnen een architectuur zijn onafhankelijk van elkaar. Op deze manier kunnen deze componenten afzonderlijk gebruikt worden door verschillende ontwikkelaars. Belangrijk voordeel hierbij is dat een ontwikkelaar geen kennis hoeft te hebben van de interne werking van een component, maar enkel de bijbehorende API toepast. Hierdoor wordt een architectuur uiterst flexibel en schaalbaar. Ook het documenteren van een systeem wordt gemakkelijker. Doordat ieder component een API aanbiedt, is gemakkelijk in kaart te brengen hoe een bepaalde applicatie werkt. De API is op deze manier een goed punt om een systeem te documenteren. Door alle losstaande componenten beschikbaar te maken via een API, kunnen deze componenten later gemakkelijk publiekelijk gemaakt worden.

Een API maakt een dienst platform onafhankelijk. De dienst kan gemakkelijk aangesproken worden op verschillende platformen. Een belangrijk feit hierbij is de standaarden. Door op XML gebaseerde standaarden te gebruiken wordt de onafhankelijkheid gewaarborgd. Een nadeel hierbij is echter performance. Doordat XML overdracht gepaard gaat met een grote hoeveelheid gegevensoverdracht, kan dit nadelige gevolgen hebben voor de snelheid.

Commercieel gezien kan het beschikbaar stellen van een API de populariteit van de eigen dienst sterk vergroten. Andere ontwikkelaars die enthousiast zijn kunnen de dienst gemakkelijk koppelen en daarmee handige tools ontwikkelen. Deze tools kunnen vervolgens door consumenten gebruikt worden, wat de bekendheid van de eigen dienst vergroot. Ook scheelt dit tijd, de handige tool die ontwikkeld is door een derde, hoeft immers niet zelf gemaakt te worden. Tevens kunnen er commerciële API's beschikbaar worden gesteld met daarin extra functionaliteiten. Indien een bepaalde API erg populair is, kan er behoefte zijn aan meer functionaliteiten. Ontwikkelaars die deze extra functionaliteiten wensen, zijn vaak bereid hiervoor te betalen.

Een belangrijk commercieel voordeel van een op SOA gebaseerde architectuur is de efficiëntie en schaalbaarheid van bedrijfsprocessen. Doordat ieder bedrijfsproces een losstaand component is kunnen deze processen in willekeurige volgorde een systeem vormen. Indien binnen een organisatie besloten wordt deze processen aan te passen, kan dit gemakkelijk gedaan worden in een SOA systeem. Tevens bevorderen API's de datastroom binnen een organisatie. Verschillende data is gemakkelijk beschikbaar en kunnen met elkaar gekoppeld worden.

Concluderend biedt het aanbieden van een web API technisch voordelen door de platform onafhankelijkheid en schaalbaarheid van een systeem. Ook biedt het ontwikkelaars meer gemak bij het koppelen van bepaalde componenten binnen een architectuur. Deze componenten kunnen ook gemakkelijk beschikbaar worden gesteld aan derden. Dit feit brengt ook commerciële voordelen met zich mee. Zo kan een eigen dienst populairder worden en kunnen inkomsten gegenereerd worden uit het aanbieden van commerciële API's. Op het gebied van bedrijfsprocessen kunnen API's meer efficiëntie en schaalbaarheid opleveren.

5.1.2 Wat zijn de technische- en commerciële waarden voor API gebruikers?

Door gebruik te maken van een web API worden ingewikkelde functionaliteiten gemakkelijk toegankelijk. De gebruiker van een API hoeft geen kennis te hebben van ingewikkelde technieken en toepassingen. Naast dit gemak levert het tijdwinst op. Bestaande eigen diensten kunnen op eenvoudige wijze met weinig tijd- en energie inspanningen verrijkt worden met extra functionaliteiten. Nadelen aan het gebruik van andermans web API's zijn er ook. De producent kan bepalen dat onderdelen uit de API veranderd worden die niet compatibel zijn met voorgaande versies van de API. Gebruikers van de API moeten dan hun eigen applicatie aanpassen. Ook kan de producent ineens bepalen advertenties te plaatsen binnen een API of zelfs complete onderdelen van de API betaald te maken. Hier zitten API gebruikers uiteraard niet op te wachten. En wat als een API producent zelfs bepaald de API op te heven?

Het beschikbaar zijn van bepaalde data via een API heeft voordelen voor ontwikkelaars. Bepaalde gegevens konden voorheen zeer kostbaar zijn en het gebruik van die data was niet weggelegd voor kleinere partijen. Tegenwoordig is veel data beschikbaar via API's en er is dan ook te zien dat kleinere partijen interessante toepassingen met deze data kunnen doen. Bijvoorbeeld de kaart- en satelliet gegevens. Vroeger zeer kostbaar, maar nu gewoon gratis te gebruiken via de Google Maps API.

Commerciële waarden voor het gebruik van andermans web API's zijn nog pril. Vaak worden inkomsten gegenereerd uit advertenties. Doordat grote diensten erg populair zijn worden combinaties hiervan, de mashups, ook vaak handig bevonden. Mashups hebben daarom potentie op het gebied van gebruik er van. Daaropvolgend biedt dit mogelijkheden op het gebied van advertentie inkomsten. Een andere commerciële waarde van het gebruik van een web API is het verzinnen van nieuwe functionaliteiten. Door nieuwe en innovatieve functionaliteiten te verzinnen op basis van een bestaande dienst, kan deze bestaande dienst de verzonnen functionaliteiten kopen van de 'mashup maker'.

De laatste commerciële waarde voor API gebruikers is de kennis op het gebied van web API's kunnen aanbieden aan derden. De eerder benoemde 'mashup consultants' kunnen andere partijen van dienst zijn met het ontwikkelen van applicaties die gebruik maken van web API's.

Concluderend is het gebruik van een web API vooral 'leuk' en gemakkelijk. Technisch gezien kan er gemakkelijk, zonder veel ontwikkeltijd, gebruik gemaakt worden van ingewikkelde toepassingen zonder daar kennis van te hebben. Commercieel zijn de mogelijkheden nog zeer beperkt qua omvang.

5.2 Aanbevelingen

Op basis van alle bevindingen en conclusies kunnen er een aantal aanbevelingen gedaan worden. Deze aanbevelingen zijn gericht aan API producenten, ontwikkelaars van diensten, gebruikers van API's en tot slot specifiek voor Gather.

5.2.1 API producenten en ontwikkelaars van diensten

1. Stel bij het ontwikkelen van een web 2.0 dienst een API beschikbaar. Gebruikers van de API kunnen nieuwe functionaliteiten verzinnen en handige tools ontwikkelen. Dit laatste kan de populariteit van de eigen dienst vergroten.
2. Biedt een API aan op basis van standaarden. Ontwikkelaars zijn bekend met de standaarden en kunnen op deze manier gemakkelijk de API implementeren. Wel hebben ontwikkelaars een voorkeur, het is daarom verstandig de API in verschillende standaarden aan te bieden.
3. Biedt als API producent in eerste instantie zoveel mogelijk gratis aan. Extra functionaliteiten kunnen vervolgens commercieel aangeboden worden.
4. Bij het ontwikkelen van een nieuwe dienst dient goed nagedacht te worden over de architectuur. In het web 2.0 tijdperk is de ontwikkel methode vooral gericht op het snel online brengen van nieuwe functionaliteiten, zodat deze direct getest kunnen worden door de gebruikers. Om de dienst succesvol uit te breiden met functionaliteiten dient vooraf een goed fundament gelegd te worden.

5.2.2 API gebruikers

1. Ontwikkel als API consultant eerst zelf een mashup en toon daarmee je kwaliteiten aan.
2. Let op dat een mashup later gemakkelijk uitgebreid kan worden met extra API's.
3. Bekijk regelmatig de community of blog van de API, hierin zijn vaak interessante voorbeelden en ontwikkelingen te vinden.
4. Maak gebruik van standaarden.
5. Maak eerst een waardevolle toepassing van één of meerdere API's en kijk vervolgens of er commerciële mogelijkheden zijn.

5.2.3 Aanbevelingen voor Gather

Onderstaande aanbevelingen zijn wellicht niet te begrijpen door andere partijen dan Gather.

1. Maak de huidige CMS module voor Google Maps dynamischer. Nu verloopt het weergeven van data via select elementen, maar wellicht zijn andere manieren van presentatie gewenst.
2. Kijk wat de mogelijkheden zijn op het gebied van API's binnen het CMS. Op dit moment is geen data uit het CMS te koppelen met andere systemen. Door standaard een API beschikbaar te stellen die gebruikt maakt van standaarden wordt dit wel mogelijk. Deze API kan dan, indien gewenst, per klant geactiveerd worden.
3. Maak naast Staatopdekaart.nl een aantal kleinere demonstraties waarin diverse toepassingen gedemonstreerd worden (vacatures, dealer locations).

Evaluatie

Het afstuderen zit erop. Achteraf heb ik een goed gevoel overgehouden aan de stage. Ik heb kunnen leren hoe bepaalde technieken daadwerkelijk in elkaar zitten en heb daarmee kunnen experimenteren. Daarnaast heb ik kunnen leren hoe het is om bepaalde technieken en ontwikkelingen te beschrijven in de vorm van een scriptie.

Wat ging goed? Vooral op het gebied van Javascript is mijn kennis gestegen. Voor de afstudeerstage was mijn kennis nog vrij minimaal. Doordat er tijdens de stage veelvuldig gebruik van gemaakt is, zijn mijn Javascript skills sterk verbeterd. Daarnaast heb ik op het gebied van HTML, CSS en PHP enkele details bijgeleerd. Mijn kennis hiervan was goed en het was daarom ook niet van belang dit te verbeteren tijdens deze stage. Wel was de toepassing van deze technieken in de praktijk waardevol.

Op het gebied van web API's heb ik veel kunnen experimenteren en leren. Doordat ik onderzocht heb welke technieken hierbij van toepassing waren kon ik gemakkelijk de koppeling naar de praktijk maken. Nadat ik API's in de praktijk bekeken had (o.a. Google Maps, Youtube, Flick etc.), kon ik zien hoe zij de technieken toepasten. Deze combinatie van theorie en praktijk was een goed leermoment.

Het toepassen van alle technieken op Staatopdekaart.nl ging goed. Er is zeer snel een eerste versie online gekomen en de uitbreiding daarvan ding goed. De manier van ontwikkelen sprak me erg aan: iedere update direct online brengen. Door de beheersing van de meeste technieken ging dit vrij goed.

Het vinden en verwerken van informatie t.b.v. de scriptie ging ook goed. Hoewel er vrij weinig literatuur aanwezig was dat nauw in verband stond met mijn onderwerp, heb ik voor mijn gevoel de juiste informatie in mijn scriptie kunnen verwerken.

Wat kan beter? Vooral het uitdenken van een systeem/architectuur kan beter. Doordat Staatopdekaart.nl zo snel mogelijk online moest was er weinig aandacht aan dit punt besteed. Toen er vervolgens steeds nieuwe functionaliteiten bedacht en toegevoegd werden, had dit nadelige gevolgen voor de structuur. Sommige delen van de structuur moesten tussentijds aangepast worden en sommige delen moesten op vreemde manieren opgelost worden i.v.m. tijd. Kort gezegd was dit punt een ondergeschoven kindje. Ik heb hierdoor geleerd dat dit punt niet te verwaarlozen is.

De manier van ontwikkelen leverde af en toe ook wat problemen op. Tijdens het online brengen van tussentijdse versies ontstonden af en toe kleine bugs in de online versie. Voor mijn gevoel was dit vervelend. Maar achteraf gezien is het ontstaan van bugs altijd mogelijk. Helemaal wanneer bij iedere update iets online gebracht wordt (volgens de web 2.0 gedachte).

Het schrijven van de scriptie ging over het algemeen goed. Echter heb ik naar mijn mening hierin één grote misstap gemaakt. Dat is het niet toepassen van de ik-vorm in de inleiding. Vooraf is hier bewust voor gekozen, maar achteraf blijkt het slecht uit te pakken. Door in de inleiding in de ik-vorm te schrijven, bouw je een persoonlijke band op met de lezer. Dat is nu dus niet het geval. Verder vond ik het laten aansluiten van de verschillende onderdelen een lastig punt. Vooraf had ik wel een goede structuur opgezet, maar tijdens het schrijven heb ik te weinig op de aansluiting gelet. Dit was voor mij ook een leermoment.

Afsluitend kan ik zeggen dat ik een goed gevoel heb overgehouden aan mijn afstudeerstage. Ik heb voldoende leermomenten gehad, ook buiten mijn afstudeerproject om van collega's. Ook het toepassen van de kennis die geleerd is tijdens de studie, is voldoende aanwezig geweest tijdens de stage.

Slotwoord

Ik hoop met deze scriptie een duidelijk beeld te hebben gegeven van mijn afstudeerstage bij Gather. Daarnaast hoop ik een goede weergave te hebben gegeven van de rol van web API's in het web 2.0 tijdperk voor producenten- en gebruikers van web API's. Tijdens mijn afstudeerzitting in week 26 of 27 wil ik graag een toevoeging doen aan deze scriptie door een demonstratie te geven van een eigen API en het project Staatopdekaart.nl. Daarnaast wil ik kort terugkomen op mijn scriptie en eventuele vragen hierover beantwoorden. In de maand juni (2007) breid ik Staatopdekaart.nl op kleine punten uit in de vorm van extra functionaliteiten. Deze zullen ook teruggekoppeld worden tijdens de afstudeerzitting.

Bibliografie

Anderson, C. (2006). *The Long Tail.* Amsterdam: Nieuw Amsterdam.

Beek, J.-W. v. (2007, maart 12). *Yahoo Pipes: wat kun je ermee?* Opgeroepen op maart 13, 2007, van Vue Royale: http://www.vue-royale.nl/index.php/site/comments/yahoo_pipes_uitleg_aan_de_hand_van_2_voorbeelden/

Bosma, Y. (sd). *Web 2.0 archief.* Opgeroepen op april 24, 2007, van Ymerce: http://www.yme.nl/ymerce/category/web-20/

Cone, E. (2006). Inside eBay's Innovation Machine. *CIO Insight* .

Costello, R. (sd). *Building Web Services the REST Way.* Opgeroepen op april 13, 2007, van xFront: http://www.xfront.com/REST-Web-Services.html

Dijkstra, B. (2005, maart). *Webservices via REST.* Opgehaald van Whitehorses: http://www.whitehorses.nl/webservices_via_rest_889.1195.html

Evans, C. (2007, april 9). *Integrating PHP with System i using Web Services.* Opgeroepen op april 9, 2007, van Zend Developer Zone: http://devzone.zend.com/node/view/id/1912

Fielding, R., & Taylor, R. (2002, mei). *Principled Design of the Modern Web Architecture.* Opgehaald van http://www.ics.uci.edu/%7Etaylor/documents/2002-REST-TOIT.pdf

Google Maps. (2007, maart 14). Opgeroepen op maart 15, 2007, van Wikipedia: http://en.wikipedia.org/wiki/Google_maps

Janssen, F. (2007, januari 16). *Web 2.0 businessmodellen*. Opgeroepen op maart 6, 2007, van Frankwatching.com: http://www.frankwatching.com/archive/2007/01/16/web-20-businessmodellen/

Mangold, C. (2007, januari 2). *Cursus Brandgagement, deel 1: 'Wat is Brandgagement?'.* Opgeroepen op mei 9, 2007, van Marketingfacts: http://www.marketingfacts.nl/berichten/20070102_cursus_brandgagement_deel_1_wat_is_brandgagement/

Mangold, C. (2007, januari 4). *Cursus Brandgagement, deel 3: Wat wil en doet je klant online?* Opgeroepen op mei 2007, 10, van Marketingfacts: http://www.marketingfacts.nl/berichten/20070104_cursus_brandgagement_deel_3_wat_wil_en_doet_uw_klant_online/

O'Reilly, T. (2005, september 9). *What Is Web 2.0*. Opgeroepen op februari 2007, van http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Programmableweb Blog. (sd). Opgeroepen op mei 22, 2007, van Programmableweb: http://www.programmableweb.com/

Rijksuniversiteit Groningen. (2005, april 13). *Schriftelijke vaardigheden voor studenten.* Opgeroepen op maart 10, 2007, van http://www.rug.nl/noordster/schriftelijkevaardigheden/voorstudenten/index

Stronks, J. (2006, maart 6). *De mashup: bekijk 'm, (en aardiger nog) gebruik 'm.* Opgeroepen op maart 16, 2007, van De Nieuwe Reporter: http://www.denieuwereporter.nl/?p=311

W3C. (sd). *Web API Working Group*. Opgehaald van W3C: http://www.w3.org/2006/webapi/

W3C World Wide Web Consortium. (2000, mei 8). *Simple Object Access Protocol (SOAP) 1.1.* Opgeroepen op mei 9, 2007, van http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Web. (sd). Opgehaald van ZDNet: http://updates.zdnet.com/tags/Web.html

Web 2.0. (sd). Opgeroepen op februari 2007, van Wikipedia: http://en.wikipedia.org/wiki/Web_2

Web service. (sd). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/Web_services

Web Services Basics. (sd). Opgehaald van Addision-Wesley: http://www.awprofessional.com/content/images/0321185773/samplechapter/manesch02.pdf

Bijlagen

Bijlage A: Database Staatopdekaart.nl

Table: categories

Veld	Type	Null	Standaardwaarde	Commentaar
category_id	int(11)	Nee		
title	varchar(255)	Nee		
alias	varchar(255)	Nee		

Table: categories_sub

Veld	Type	Null	Standaardwaarde	Commentaar
category_sub_id	int(11)	Nee		
category_id	int(11)	Nee	0	
title	varchar(255)	Nee		

Table: cities

Veld	Type	Null	Standaardwaarde	Commentaar
city_id	int(11)	Nee		
type	varchar(255)	Nee		
title	varchar(255)	Nee		
alias	varchar(255)	Nee		
long	varchar(255)	Nee		
lat	varchar(255)	Nee		
zoomlevel	int(10)	Nee	0	
browser_title	varchar(255)	Nee		
browser_description	varchar(255)	Nee		
google	text	Nee		

Table: locations

Veld	Type	Null	Standaardwaarde	Commentaar
location_id	int(11)	Nee		
city_id	int(11)	Nee	0	
category_id	int(11)	Nee	0	
category_sub_id	int(11)	Nee	0	
long	varchar(255)	Nee		
lat	varchar(255)	Nee		
title	varchar(255)	Nee		
street	varchar(255)	Nee		
number	varchar(255)	Nee		
zip	varchar(255)	Nee		
city	varchar(255)	Nee		
country	varchar(255)	Nee		
url	varchar(255)	Nee		
description	text	Nee		
description_abstract	text	Nee		
image	varchar(255)	Nee		
video	text	Nee		
image_intern	varchar(255)	Nee		

Table: replies

Veld	Type	Null	Standaardwaarde	Commentaar
reply_id	int(11)	Nee		
location_id	int(11)	Nee		
date	int(30)	Nee		
author	varchar(255)	Nee		
author_email	varchar(255)	Nee		
ip	varchar(255)	Nee		
message	text	Nee		

Table: routes

Veld	Type	Null	Standaardwaarde	Commentaar
route_id	int(11)	Nee		
route_code	varchar(255)	Nee		
title	varchar(255)	Nee		
description	text	Nee		
ip	varchar(255)	Nee		
city_id	int(11)	Nee	0	
author	varchar(255)	Nee		
show_line	enum('true', 'false')	Nee	false	
zoomlevel	int(10)	Nee		
preview	enum('true', 'false')	Nee	false	

Table: routes_locations

Veld	Type	Null	Standaardwaarde	Commentaar
location_id	int(11)	Nee		
route_id	int(11)	Nee	0	
long	varchar(255)	Nee		
lat	varchar(255)	Nee		
title	varchar(255)	Nee		
type	varchar(255)	Nee		
specialtype	varchar(255)	Nee		

Table: types

Veld	Type	Null	Standaardwaarde	Commentaar
type_id	int(11)	Nee		
type	varchar(255)	Nee		
opt_route	enum('true', 'false')	Nee	false	
opt_select_text	varchar(255)	Nee		
style	varchar(255)	Nee		