Scriptie Luc van Donkersgoed
AppStats 2

Auteur:	Luc van Donkersgoed
Studentnummer:	1220049
Eerste examinator:	Jan Nijman
Datum:	December 14, 2010
Versie:	1.0

Voorwoord

Voor u ligt de scriptie van Luc van Donkersgoed. In deze scriptie wordt het ontstaan van een nieuw product beschreven: AppStats2 - een platform waarmee Coding Dutchmen, haar klanten en iedere andere ontwikkelaar inzicht kunnen krijgen in het gebruik van applicaties en achterliggende services.

Het ontstaan van AppStats2 heeft meerdere fases doorlopen: eerst als AppStats (zonder de 2), vervolgens als een behoefte naar meer informatie en een snellere, betrouwbaardere werking van het systeem. Daarna begon het vorm aan te nemen als een onderzoek naar de mogelijkheden, waar een theoretisch concept uit voortkwam. Uiteindelijk is uit het concept een robuust functioneel product ontstaan dat haar eerste vruchten reeds afgeworpen heeft.

Ik heb mijn afstudeerstage bij mijn eigen bedrijf kunnen lopen, wat een uitzonderlijke situatie is. Ik wil Leo van Moergestel en de rest van de examencommissie bedanken voor deze mogelijkheid - als zij geen vertrouwen in deze stage hadden gehad was ik niet zeker van mijn diploma geweest.

Ook wil ik Jan Nijman bedanken voor het licht dat hij op de processen, producten en documenten rond mijn afstuderen heeft laten schijnen. Een speciaal woord van aandacht gaat naar Petra Belgraver, zonder wie ik deze studie jaren geleden al opgegeven zou hebben.

Het meest wil ik echter Bob Vork en Jeroen Faijdherbe - mijn collega's, vennoten en beste vrienden - bedanken. Jullie houden het iedere dag met mij, mijn Duitse punk, buien van hyperactiviteit en sportongelukken vol, en hebben me nu ook nog een halfjaar kostbare tijd aan mijn afstuderen laten besteden.

Luc van Donkersgoed

Samenvatting

Deze scriptie beschrijft het ontwikkeltraject van het AppStats2-platform. Dit traject omvat een tweezijdig onderzoek, de ontwikkeling van het systeem en oplevering van het product.

De ene zijde van het onderzoek richt zich op de voorganger van AppStats2, waar geanalyseerd is welke tekortkomingen dit systeem had en welke onderdelen opnieuw gebruikt kunnen worden. De andere zijde is gericht op de verwachtingen en eisen van het nieuwe systeem en een analyse van toe te passen methoden, technologieën en *best practises.*

Over de ontwikkeling van het systeem wordt beschreven welke procesmanagement- en ontwikkeltechnieken gebruikt werden, welke problemen en uitdagingen tegengekomen zijn, welke keuzes gemaakt zijn en welke gevolgen dit heeft gehad voor het product.

De oplevering beschrijft hoe het systeem voldoet aan de gestelde eisen, hoe het gebruikt dient te worden en welke toekomstige mogelijkheden er zijn op het gebied van beveiliging, commerciële inzet en groei.

Inhoudsopgave

1. Inleiding

AppStats2 is een platform voor ontwikkelaars en hun klanten. Met het product kan het gebruik van (mobiele) applicaties gemeten en weergegeven worden. Daarnaast kan ook het gebruik en de belasting van de achterliggende services geregistreerd worden.

AppStats2 is de vervanger van het reeds bestaande AppStats-platform. Het oorspronkelijke systeem is ontstaan uit de simpele behoefte om te meten hoe vaak gebruikers een bepaalde app openen en hoe vaak ze op een banner drukken. Het platform bleek echter geschikt te zijn om veel meer informatie te registreren, waarmee gedetailleerde profielen van het gebruik van applicaties gemaakt konden worden.

Deze toename van gebruik bracht echter de ongestructureerde opzet van het eerste platform aan het licht. Uit de behoefte om structuur aan te brengen is het AppStats2-platform ontstaan: een volledig nieuwe versie van het registratieplatform, van de grond af opgebouwd en geschikt om de verwachte groei van de komende jaren te ondersteunen.

1.1. Uitgangssituatie

AppStats was nooit bedoeld om tot haar huidige grootte uit te groeien. Dit is terug te voeren naar haar oorsprong; in augustus 2009 heeft Coding Dutchmen voor de Rabobank de applicatie Rabo iSport ontwikkeld. Met deze app kon de Vuelta à España 2009 live gevolgd worden, konden de verschillende klassementen bekeken worden en kon algemene info van renners bekeken worden. De Rabobank wilde echter ook banners in de app verwerken en vanzelfsprekend moest het aantal clicks op deze banners geregistreerd worden. Uit deze wens is de eerste versie van de eerste versie van AppStats (toen nog CDStats genoemd) ontstaan: iedere keer dat een gebruiker op de banner drukte werd hiervan bericht naar onze servers gestuurd en daar in een database opgeslagen, zodat wij hier statistieken uit konden opbouwen.

Bij het ontwikkelen van de eerste applicatie na iSport bedachten we dat we hetzelfde systeem vrijwel zonder wijziging konden gebruiken voor het registreren van startups; het verzenden van een bericht vanuit onze apps op het moment dat ze door de gebruiker opgestart werden. Deze gegevens konden ons helpen een beeld te vormen van het gebruik van onze applicaties.

Later werden hier in verschillende iteraties van CDStats andere berichten aan toegevoegd, bijvoorbeeld de taal van de telefoon, het model van de telefoon, een bericht of de applicatie de laatste keer crashte of goed afgesloten was, enzovoorts.

Ook aan de andere kant van de database - het uitlezen van de gegevens - werd steeds meer gewijzigd, uitgebouwd en toegevoegd. In de eerste instantie hadden we enkel een website met een paar balkgrafieken waarmee het gebruik van de apps werd weergegeven, later kwamen hier taart- en lijndiagrammen van allerlei relevante data bij. Ook de aanlevering van data veranderde: oorspronkelijk werd de data direct uit de database gehaald, maar naarmate de database groter werd (we zitten nu op 4,7 miljoen records) werd dit te traag. Daarom zijn we later overgegaan naar het iedere minuut wegschrijven van gefilterde gegevens naar een kleinere, snellere tabel. Een volgende stap was de gegevens uit deze tabellen te cachen in files op het moment dat ze aangevraagd werden.

1.2. Probleemstelling

Het eerste AppStats-platform is een product met een aantal tekortkomingen. Hierdoor is het platform niet commercieel in te zetten, zijn er met regelmaat tijdrovende menselijke ingrepen nodig en is het systeem niet geschikt om de verwachte groei van het aantal inkomende berichten te verwerken.

Van het AppStats-platform dienen de beperkingen onderzocht en vastgelegd te worden. Daarnaast moet bepaald welke eisen aan het nieuw te bouwen AppStats2-platform gesteld worden, met het oog op huidige en toekomstige behoeften.

1.3. Doelstelling

Luc van Donkersgoed zal in het kader van zijn afstudeerproject een database en omringende softwarestructuur ontwikkelen waarmee het gebruik van de door Coding Dutchmen ontwikkelde applicaties geregistreerd, opgeslagen en geanalyseerd kan worden.

Het nieuw te ontwikkelen platform zal grotendeels gebaseerd zijn op het oorspronkelijke AppStats, maar zal ook een aantal ingrijpende wijzigingen bevatten.

Dit project wordt uitgevoerd met de volgende doelen:

- Hoofddoel: Het versterken van de marktpositie van het bedrijf door middel van het registreren en weergeven van het gebruik van haar softwareproducten.
- Subdoel 1: Het automatiseren van de analyse en rapportage van de gebruiksgegevens, waarmee de menselijke arbeidslast verminderd wordt.
- Subdoel 2: Het bieden van een onderscheidende strategische analysedienst op de gebruiksgegevens, waarmee de B2B-waarde van het bedrijf verhoogd wordt.
- Subdoel 3: Bovenstaande analysedienst zo inrichten dat deze intern gebruikt kan worden om het inzicht in de markt en doelgroepen te verbeteren.

2. Aanpak

2.1. Inleiding

AppStats2 ontwikkel ik als stagiair in mijn eigen bedrijf. Dit heeft een aantal gevolgen voor de begeleiding- en processtructuur, welke in dit hoofdstuk toegelicht zullen worden. Ook zal ik ingaan op de RUP-projectmethode en de gevolgen die deze methode op de werkwijze binnen het project heeft gehad.

De onderwerpen die in dit hoofdstuk beschreven worden komen voor een groot deel voort uit het Plan van Aanpak, dat eind oktober 2010 opgeleverd is.

2.2. Het bedrijf

Coding Dutchmen is het bedrijf dat ik in maart 2009 met twee vennoten gestart ben. Het vennootschap, dat als core business het ontwikkelen van applicaties voor Apple's iPhone, iPod Touch en iPad heeft, mag enkele grote namen uit de Nederlandse bedrijvencirkel tot haar klanten rekenen, waaronder de Rabobank, Omroep Friesland, de Dokkumer Vlaggencentrale en FMB Uitgevers.

Coding Dutchmen is een partner in ontwerp, ontwikkeling en productplaatsing voor middelgrote en grote bedrijven in Nederland. Daarnaast bedenken en creëren we zelf apps waarvan we denken dat ze een gat in de markt kunnen vullen of waar we onze eigen creativiteit in kwijt kunnen.

Coding Dutchmen onderscheidt zich door producten te ontwikkelen zoals de gebruiker ze wil zien: intuïtief, snel en stabiel. Door deze gedachte op alle onderdelen van onze iPhone-apps toe te passen leveren we gebruiksvriendelijke applicaties zonder overbodige functionaliteit.

2.3. De stage

Ik loop deze afstudeerstage in mijn eigen bedrijf, wat een aantal gevolgen voor de processtructuur heeft. Het belangrijkste effect is dat ik bepaalde verantwoordelijkheden moet naleven, ook als ik met mijn afstuderen bezig ben. De uren die ik hieraan kwijt ben compenseer ik echter door in avonden en weekenden aan mijn stage te werken.

Een ander gevolg van een stage in eigen bedrijf is dat ik geen meerdere heb en mijn eigen opdrachtgever ben. Dit zou in theorie kunnen leiden tot een afname van werkdruk en motivatie. Om dit probleem bij voorbaat te vermijden heb ik mijn stage en de op te leveren producten bij aanvang duidelijk beschreven, zodat ik een duidelijk doel had. Bovendien besteed ik bijna een halfjaar van mijn (betaalde) uren aan deze stage, waardoor ook mijn vennoten eisen stellen aan de producten die ik oplever.

Tenslotte is Jan Nijman bij de stage betrokken als docentbegeleider. Hij houdt de processen en (tussentijds) opgeleverde producten in het oog, wat de derde garantie is voor een goed ingevulde afstudeerstage.

2.4. Rational Unified Process

Tijdens dit project wordt de Rational Unified Process-projectmethode (RUP) toegepast. Deze methode is gericht op (software)processen waarvan de precieze invulling van het het eindresultaat gedeeltelijk bepaald wordt tijdens het ontwikkelen. Dit kan voortkomen uit wijziging van de eisen, beter begrip van het probleem of verandering van de technische mogelijkheden tijdens het project. Om deze wijzigingen tijdens het ontwikkeltraject te ondersteunen worden tijdens het project regelmatig iteraties van de deelproducten opgeleverd, welke geanalyseerd en getest worden voordat doorgegaan wordt met de rest van de ontwikkeling.

2.5. Fasen

In de RUP-methode zijn vier delen van de projectcyclus gedefinieerd: analyse en ontwerp, implementatie, testen en uitrol. De eerste drie fasen volgen elkaar circulair op. In de eerste ronde ontstaat een basisversie van het systeem, met een basale werking van ieder onderdeel. Nadat deze basisversie getest en compleet bevonden is wordt de ontwikkeling van alle componenten voortgezet, waaruit een nieuwe iteratie ontstaat. Deze cyclus wordt voorgezet tot het systeem als gereed voor uitrol beschouwd wordt.

In het AppStats2-project zijn de fasen van de RUP-ontwikkelmethode vertaald naar vier fasen met een iets afwijkende naamgeving, welke in onderstaande paragraven beschreven worden.

2.5.1. Analyse

Tijdens de analyse worden twee onderzoeken verricht; de eerste gaat in op de problemen van het bestaande AppStats platform, de tweede op de mogelijkheden van het nieuwe systeem. Uit deze onderzoeken komt een functioneel ontwerp voort. Bij latere iteraties wordt de analyse-fase gebruikt om het eerste functioneel ontwerp uit te breiden met nieuwe inzichten en eisen.

2.5.2. Toepassing

In de tweede fase wordt het functioneel ontwerp uitgebreid met implementeerbare oplossingen. In deze fase worden de keuzes voor toe te passen technieken en systemen gemaakt, wat uiteindelijk in de vorm van het technisch ontwerp opgeleverd wordt.

In latere iteraties wordt deze fase gebruikt om het technisch ontwerp bij te werken op basis van inzichten uit de uitvoering-fase.

2.5.3. Uitvoering

In deze fase wordt het technisch ontwerp geïmplementeerd. Het resultaat is een bruikbaar product dat getest en eventueel al ingezet kan worden.

Uit het testen van de opgeleverde producten komen problemen en verbeterpunten voort, welke meegenomen worden in de analysefase van de volgende iteratie.

2.5.4. Oplevering, conclusies en aanbevelingen

Uit de iteraties van de bovenstaande drie fasen komt een product voort dat in de praktijk gebruikt kan worden. Van dit product zal bekeken worden hoe het het best commercieel ingezet kan worden en waar het op langere termijn verbeterd en uitgebreid kan worden. Deze punten worden in hoofdstuk 6 opgenomen.

2.6. Op te leveren producten

Bij afronding van het project moet het platform geschikt zijn om berichten met gebruiksinformatie vanuit verschillende bronnen te ontvangen, deze op te slaan in een database en vervolgens in geoptimaliseerde vorm aan te kunnen leveren aan gebruikers van het AppStats2-platform. Bovendien moeten de problemen uit het oude systeem (welke vastgesteld worden in hoofdstuk 2: analyse) in het nieuwe platform verholpen zijn. De producten die hieruit voortkomen zijn:

- Een database, geschikt om alle te voorziene meldingen en berichten van verschillende bronnen op te slaan. Deze berichten moeten door gebruikers van het AppStats2-platform in verschillende rapportagevormen getoond kunnen worden. Niet iedere gebruiker heeft echter dezelfde rechten, waardoor het nodig om de database ook geschikt te maken voor gebruiker-management (registratie van gebruikers, rollen, enzovoorts).
- Een module waar data naartoe verzonden kan worden, *receiving* genoemd. Deze module moet geschikt zijn om berichten van meerdere bronnen te ontvangen en te verwerken. In deze module moet tijdens het verwerken van berichten bovendien nagegaan worden of alle ontvangen data geldig is en of het ontvangen bericht opgenomen mag worden in de database (validatie en filtering).
- Een module die op aanvraag van gebruikers van het AppStats2-platform gegevens uit de database kan aggregeren, filteren en sorteren. Deze module wordt *requesting* genoemd. Bij een aanvraag dient deze module te controleren of de aanvragende partij de juiste rechten heeft om de gegevens te ontvangen.
- Een module, geschikt om wijzigingen in de databases mee door te voeren. Deze module wordt *administration* genoemd. Met deze module kunnen bijvoorbeeld nieuwe gebruikers of applicaties toegevoegd worden, of kunnen bepaalde applicaties geblokkeerd worden.
- Een hardwareplatform om bovenstaande databases en modules te hosten.
- Documentatie van de API's, zodat deze door externe ontwikkelaars gebruikt kunnen worden. In de documentatie worden de mogelijke aanroepen op de API's vastgelegd, inclusief de verwachtte parameters en de mogelijke respons.
- Een beveiligingsanalyse van de mogelijke zwakheden van het systeem, inclusief registratie van de genomen stappen om lekken te voorkomen.

2.6.1. Te ontwerpen producten

De bovengenoemde producten zullen bij afronding van de afstudeerstage bruikbaar zijn of eventueel zelfs al in gebruik genomen zijn. Daarnaast zullen een aantal producten in het functioneel- en technisch ontwerp opgenomen worden die niet door Luc van Donkersgoed ontwikkeld zullen worden. De werking van deze producten heeft echter grote invloed op de rest van het systeem, wat geleid heeft tot de keuze om het ontwerp wel in de scriptie op te nemen.

2.6.2. API's

In de modules receiving, requesting en administration worden API's opgenomen waarmee de functionaliteit van de module aangesproken kan worden. Deze API's worden opgebouwd uit PHP-scripts, welke elk een eigen functie hebben. Per module wordt een eigen directory op de server ingericht, zodat de functie van de scripts voor een groot deel uit de URL opgemaakt kan worden.

Zo bestaat de URL https://api.codingdutchmen.com/Stats/request/getApplications.php bijvoorbeeld uit de volgende elementen:

- https:// - het protocol dat bij de aanroep gebruikt wordt.
- api.codingdutchmen.com/Stats/ - de basisdirectory waar alle scripts en klassen zich bevinden.
- request/ - de subfolder voor de request-module.
- getApplications.php - het aan te roepen script.

2.7. Ontwikkelomgeving

2.7.1. Werkplek

Bij de ontwikkeling van het AppStats2-platform zal gebruik gemaakt worden van de reeds bestaande werkplek bij Coding Dutchmen. Dit betekent dat de volgende systemen gebruikt zullen worden:

Ontwikkeling:

- Mac OSX
- SubEthaEdit voor het schrijven van PHP
- XCode IDE voor de ontwikkeling van iPhone-apps
- PHPMyAdmin voor het schrijven en testen van SQL-scripts

Hosting:

- Dell PowerEdge servers
- Ubuntu Linux
- MySQL 5.1
- Apache 2.2
- PHP 5.2

Administratie:

- ActiveCollab voor issuetracking
- Subversion voor versiebeheer

2.7.2. Projectteam

Het projectteam bestaat uit drie medewerkers:

1. Luc van Donkersgoed, ontwerper en hoofdontwikkelaar van het AppStats2-platform.
2. Jeroen Faijdherbe, projectbegeleider en ontwikkelaar van betrokken systemen.
3. Gerald Eersteling, stagiair en ontwikkelaar van een applicatie die gebruik maakt van het AppStats2-platform.

Jan Nijman is bij het project betrokken als docentbegeleider.

3. Analyse

3.1. Inleiding

De analyse bestaat uit twee onderzoeken:

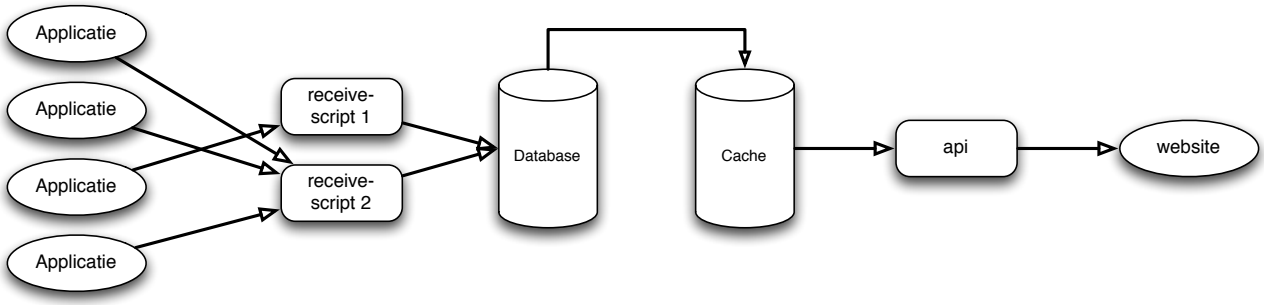
1. Voor aanvang van het project is onderzocht wat de tekortkomingen van het originele platform waren, maar ook welke delen herbruikbaar zijn.
2. Daarna is onderzocht welke verwachtingen we van het nieuwe systeem hebben, aan welke eisen het moet voldoen en welke methoden en technieken daar het best op aansluiten.

De resultaten van het eerste onderzoek worden beschreven in paragraaf 3.2. De resultaten van het tweede onderzoek zijn verzameld in het Functioneel Ontwerp (paragraaf 3.3) en de Lijst van Eisen (paragraaf 3.4).

3.2. Onderzoek naar bestaand product

Voor het bestaande product - AppStats - is nooit een ontwerp gemaakt. Het is begonnen als een simpele database, bestaande uit één tabel, waar een bericht geplaatst werd zodra een gebruiker op een banner drukte. Gegevens uit deze tabel werden direct uit de database gehaald door een SQL-script in te voeren en de resultaten te kopiëren.

Toen AppStats voor meer toepassingen gebruikt ging worden zijn de database en tabellen naar behoefte uitgebreid, wederom zonder eerst een ontwerp uit te denken. De globale opzet van dit systeem staat in figuur 1 beschreven:

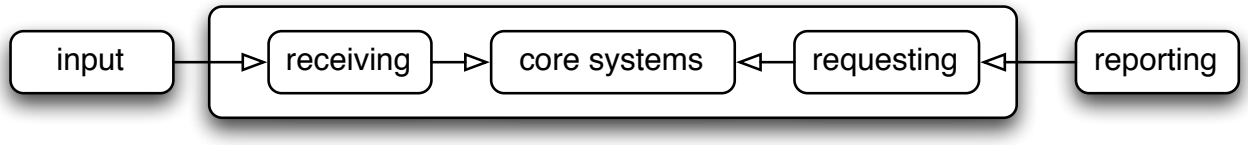


Figuur 1: Contextdiagram AppStats

AppStats is in grote lijnen een goed systeem, dat zich leent voor een breed scala aan toepassingen. Het probleem zit voornamelijk in de implementatie van de details en het gebrek aan vooruitziend ontwerp. In onderstaande paragraven worden de ervaringen met het oorspronkelijke ontwerp beschreven, inclusief de inzichten en conclusies die deze ervaringen met zich meebrachten.

3.2.1. *Vastgestelde kwaliteiten*

Het contextdiagram in figuur 1 biedt een goede basis voor een platform voor gebruikstatistieken. Het is echter teveel beperkt tot vaste systemen, bijvoorbeeld door enkel applicaties als input te nemen, enkel websites als output, en caching verplicht te maken. Daarom is het contextdiagram versimpeld tot het diagram in figuur 2:



Figuur 2: Versimpeling contextdiagram AppStats

De opdeling in segmenten in figuur 2 staat aan de basis van het ontwerp van AppStats2.

Een tweede kwaliteit van het oude systeem is de structuur van berichten. Vanaf de allereerste versie wordt gewerkt met een Subject en een Message. Deze combinatie blijkt erg flexibel te zijn, waardoor het mogelijk wordt om totaal verschillende berichten in dezelfde tabel op te slaan. Zie het voorbeeld in figuur 3:

UniqueID	CallerIP	Subject	Message	PhoneUID	HTTPappID
5427771	80.101.85.99	News	Van der Haar wint in Gieten	404c0a3242eac9ad14812e1c4e9852d3c6d4a039	RaboCycling/1.0.10
5427770	80.101.85.99	DeviceLanguage	nl	404c0a3242eac9ad14812e1c4e9852d3c6d4a039	RaboCycling/1.0.10
5427769	80.101.85.99	Startup		404c0a3242eac9ad14812e1c4e9852d3c6d4a039	RaboCycling/1.0.10
5427768	122.107.65.76	Startup	lang:en	c14cb919d027e760884131715f9494afbd8afc61	FML/2.2
5427767	122.107.65.76	DeviceLanguage	en	c14cb919d027e760884131715f9494afbd8afc61	FML/2.2
5427766	188.88.52.22	DeviceLanguage	nl	cf218d680ae0016dddf822b7dc9bbb158003665d	FML/2.2
5427765	188.88.52.22	Startup	lang:en	cf218d680ae0016dddf822b7dc9bbb158003665d	FML/2.2
5427764	118.209.202.163	FMLLanguage	en	ae2943b390b98d1d9ad1bee0b871a158229c2a66	FMLFree/1.0.1
5427763	118.209.202.163	Startup	Resume	ae2943b390b98d1d9ad1bee0b871a158229c2a66	FMLFree/1.0.1
5427762	118.209.202.163	DeviceLanguage	en	ae2943b390b98d1d9ad1bee0b871a158229c2a66	FMLFree/1.0.1
5427761	118.209.202.163	SessionDuration	656.564392	ae2943b390b98d1d9ad1bee0b871a158229c2a66	FMLFree/1.0.1

Figuur 3: Voorbeeld Subject-Message structuur

Vanwege de brede inzetbaarheid van deze structuur is ervoor gekozen deze ook toe te passen in het nieuwe systeem.

In het oude systeem is ook het principe van een AppKey ontstaan. Dit is een unieke waarde van 32 hexadecimale karakters waarmee een applicatie geïdentificeerd kan worden, ongeacht zijn versie of het platform waar het op draait. Door hier door het gehele systeem gebruik van te maken kunnen applicaties die op Android, iOS en Windows Phone 7 draaien als één app geregistreerd worden, waardoor de statistieken bijvoorbeeld het gebruik over de drie platformen cumulatief kunnen tonen.

3.2.2. Vastgestelde problemen

Helaas is de lijst met vastgestelde problemen een stuk groter dan de vastgestelde kwaliteiten:

1. [Input] De applicaties die gebruik maken van het AppStats-platform moeten verbonden zijn met het internet om een bericht te kunnen versturen. Hierdoor kan het gebruik van onze applicaties bij het ontbreken van internet niet gemeten worden.
2. [Receiving] De eerste versie van de logger diende slechts om het aantal clicks op banners te meten. Deze werden destijds verstuurd naar count.php. Later werd count.php uitgebreid met extra functionaliteit, terwijl backwards-compatibilty behouden werd. In een latere versie was deze backwards-compatibilty niet meer te behouden en werd een nieuw script (appLog.php) toegevoegd. Als er nu wijzigingen doorgevoerd moeten worden, moeten beide scripts aangepast worden.
3. [Receiving] De geldigheid van de ontvangen gegevens wordt niet gecontroleerd, waardoor ook 'slechte' gegevens in de database terecht kunnen komen.
4. [Receiving] De API controleert niet of gegevens succesvol in de database opgeslagen zijn.
5. [Receiving / Input] De API stuurt geen bericht terug naar het device om te bevestigen dat het bericht aangekomen en verwerkt is.
6. [Database] De opslag van de berichten vindt plaats in één grote tabel. Dit datamodel wordt een platte database genoemd. Een platte database is erg inefficiënt wanneer gegevens herhaaldelijk terugkomen, zoals in AppStats het geval is. Door het gebruik van een platte database bevat ieder record bijvoorbeeld gegevens over een bepaalde telefoon, terwijl deze data voor alle berichten van een bepaald type telefoon gelijk zijn.
7. [Database] Veel gegevens worden opgeslagen in velden met een variabele lengte, wat de database zowel groter als trager maakt.
8. [Database / Requesting] In het huidige systeem is geen mogelijkheid om via een interface gebruikers / applicaties toe te voegen. Om dit te doen moesten wijzigingen direct in de database ingevoerd worden, wat omslachtig en onveilig is.
9. [Database] Iedere minuut wordt het aantal users en sessies in de hoofddatabase geteld en naar een cache-database opgeslagen. Deze methode mist flexibiliteit: andere gegevens dan gebruikers / sessies kunnen niet gecachet worden en moeten alsnog uit de hoofddatabase gehaald worden.
10. [Database] De bij issue 9 beschreven manier van cachen werkt niet samen met het offline verzenden van gegevens (issue 1). Bij het huidige cachen worden cumulatieve gegevens van de huidige en afgelopen minuut berekend en opgeslagen. Hierdoor hoeft niet de gehele database uitgelezen te worden, maar enkel de lijst van cumulatieven. Bij het ontvangen van berichten van dagen, weken of mogelijk zelfs maanden geleden zou deze manier van cachen niet werken, omdat deze laat bezorgde gegevens niet in de cache-database terecht zouden komen.
11. [Requesting] De huidige Request-API bevat enkel scripts en methoden die voorgedefinieerde gegevens opvragen. Als er nieuwe gegevens in de database opgenomen worden, moet een nieuw script dat deze records omvormt naar toonbare data aan de API toegevoegd worden. In de praktijk betekent dit dat de gegevens niet of pas na een periode van weken of maanden aan de statistiekenwebsite toegevoegd worden.

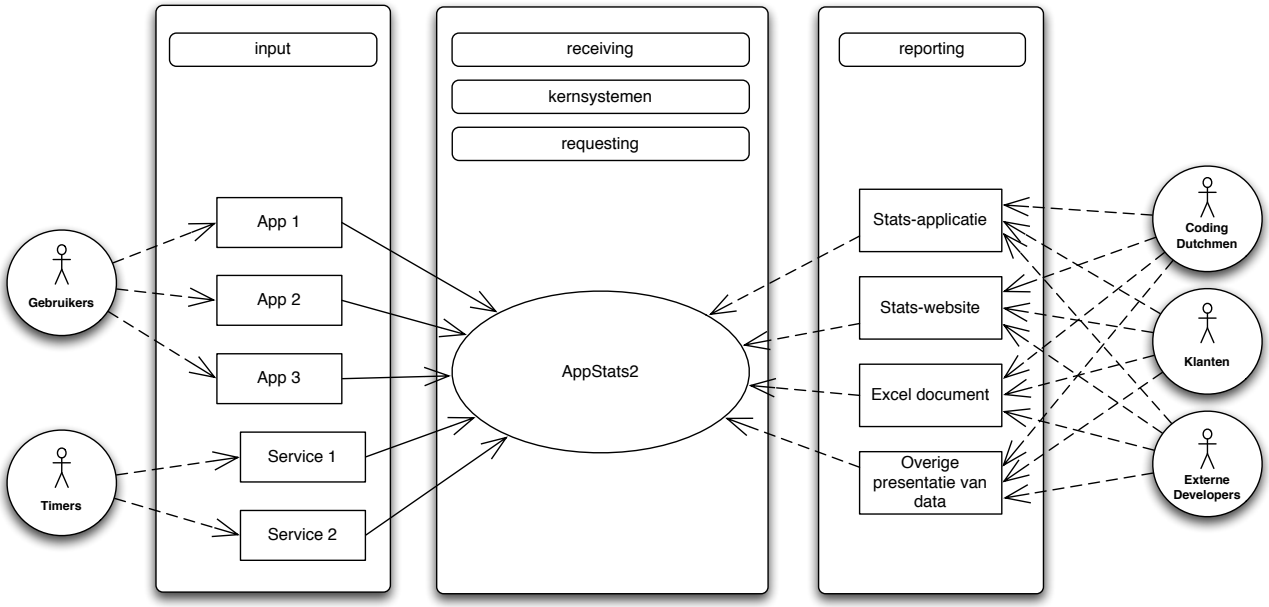
3.3. Functioneel ontwerp

Er zijn twee leidraden bij het ontwerp van het nieuwe systeem: ten eerste moet het systeem de hoofddoelstelling en de drie subdoelstellingen kunnen vervullen (zie paragraaf 1.3) en ten tweede moeten oplossingen voor alle tekortkomingen van het oorspronkelijke systeem (zie paragraaf 3.2.2) geïmplementeerd worden.

3.3.1. Context

Bij het ontwerp is eerst de context van het systeem vastgesteld. De context bepaalt de positie van het systeem tussen de externe factoren waar het interactie mee heeft. Deze factoren zijn de gebruikers die onze applicaties gebruiken, timers die de services aansturen en de gebruikers die met het systeem werken (bijvoorbeeld Coding Dutchmen en haar klanten).

In figuur 4 is de context van het systeem in de vorm van een simpel diagram weergegeven.



Figuur 4: Basis use cases AppStats2

In deze use case is de structuur van het oorspronkelijke AppStats te herkennen (zie paragraaf 3.2 en figuur 2), maar zijn reeds een aantal keuzes voor het nieuwe platform gemaakt. Ten eerste is het Input-segment vastgesteld als een verzameling van Apps en Services, ten tweede zijn de mogelijke gebruikers van het systeem in groepen verdeeld:

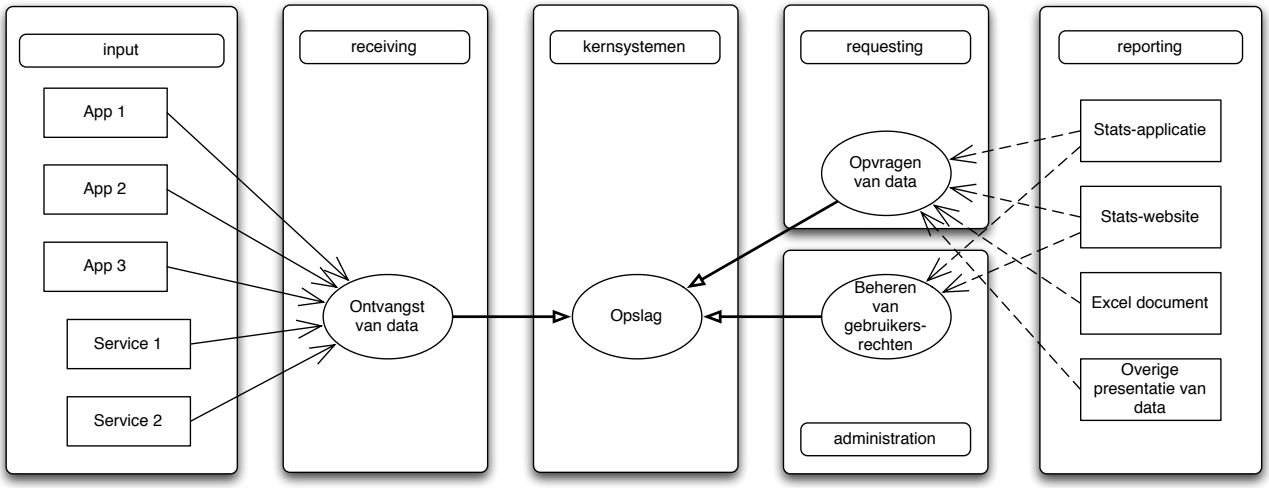
1. Coding Dutchmen: ontwikkelaar van iPhone- en iPad-applicaties. Het statistiekenplatform wordt gebruikt om inzicht te krijgen in gebruikerspatronen, waarmee in de toekomst betere applicaties gemaakt kunnen worden.
2. Klanten: de bedrijven die applicaties door Coding Dutchmen laten ontwikkelen. Deze klanten hebben graag inzicht in het aantal gebruikers, hun afkomst en interesses. Ook willen ze op de hoogte blijven van pieken en dalen in het gebruik van hun applicatie. Zij willen het systeem gebruiken op een manier die vergelijkbaar is met Google Analytics.
3. Externe developers: ontwikkelaars van iPhone-, Android- of Windows Phone 7-applicaties, anders dan Coding Dutchmen. Ook deze developers moeten gebruik kunnen maken van het systeem, primair vanuit een commerciële overweging vanuit Coding

Dutchmen; gebruik van het systeem kan via licenties verkocht worden. Het is echter ook van belang om een uitgebreide database van gegevens op te bouwen, welke door Coding Dutchmen gebruikt kan worden om verdere inzichten in de markt te verkrijgen.

Nu vaststaat welke *actors* aan de linker- en rechterkant van het systeem gebruik maken, kan invulling gegeven worden aan de werking van het AppStats2-platform. Hierbij komen de kerntaken van het systeem kijken:

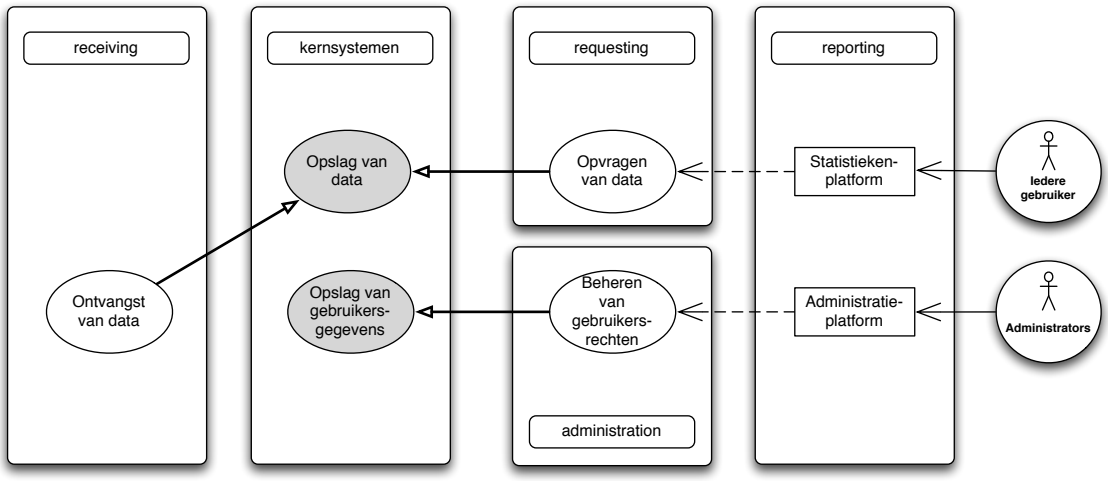
1. Het ontvangt berichten van applicaties en services.
2. Het slaat deze berichten op.
3. Het levert deze gegevens op aanvraag aan gebruikers.
4. Het verwerkt veranderingen in gebruikersrechten.

Deze functies staan in figuur 5 geïllustreerd.



Figuur 5: Gebruikersinteractie met AppStats2

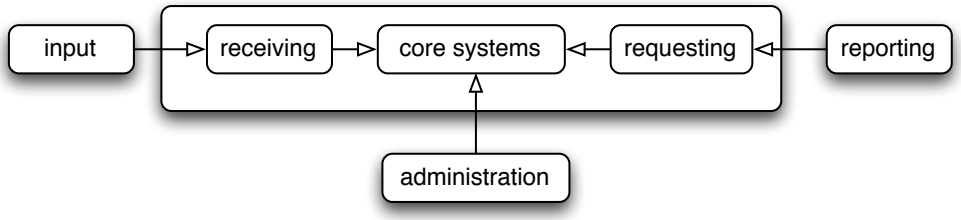
Het laatste punt uit de lijst van kerntaken is een belangrijk element in de volgende verdieping van het contextdiagram: gebruikers hebben specifieke rechten tot het lezen, bekijken en wijzigen van gegevens in de database, afhankelijk gebruikersrol. Deze rechten worden in aparte tabellen in de database opgeslagen, waardoor een scheiding in de opslag van gegevens ontstaat. Dit staat aangegeven in figuur 6:



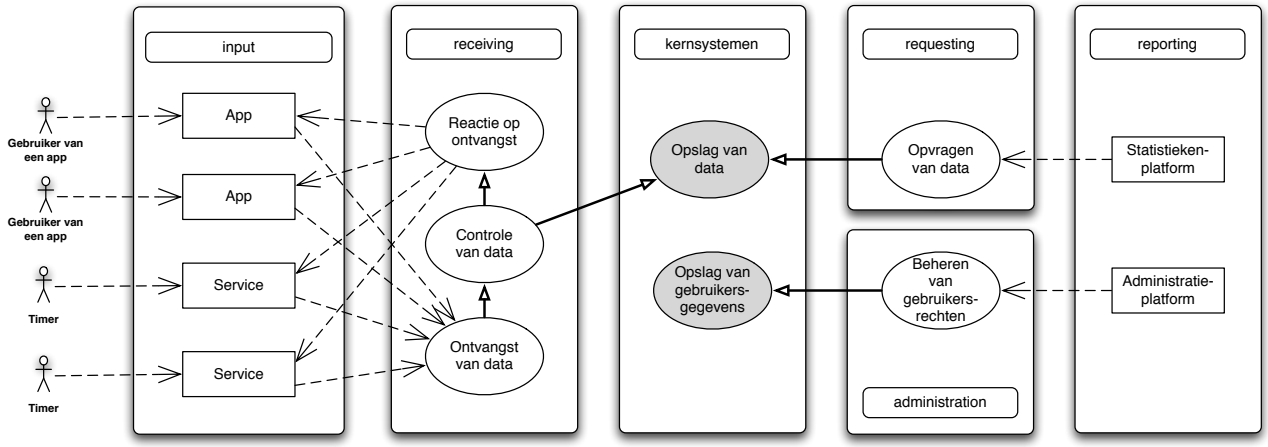
Figuur 6: Scheiding data en gebruikersgegevens

De twee onderdelen van het segment reporting in figuur 6 zijn een aggregatie van de eerder genoemde segmenten; onderdelen als een stats-applicatie, stats-website of excel-document kunnen zowel deel uitmaken van het statistiekenplatform als het administratieplatform. Dit kan bijvoorbeeld de vorm aannemen van een algemene website waar een gebruiker kan inloggen om statistieken te bekijken, maar waar een beheerder de mogelijkheid heeft om gebruikersrechten te wijzigen.

Door de use case 'Beheren van gebruikersrechten' toe te voegen verandert ook de opdeling in segmenten die in paragraaf 3.2.1 besproken werd. Er moet immers een methode zijn om gegevens in de database te wijzigen, wat leidt tot het diagram in figuur 7:

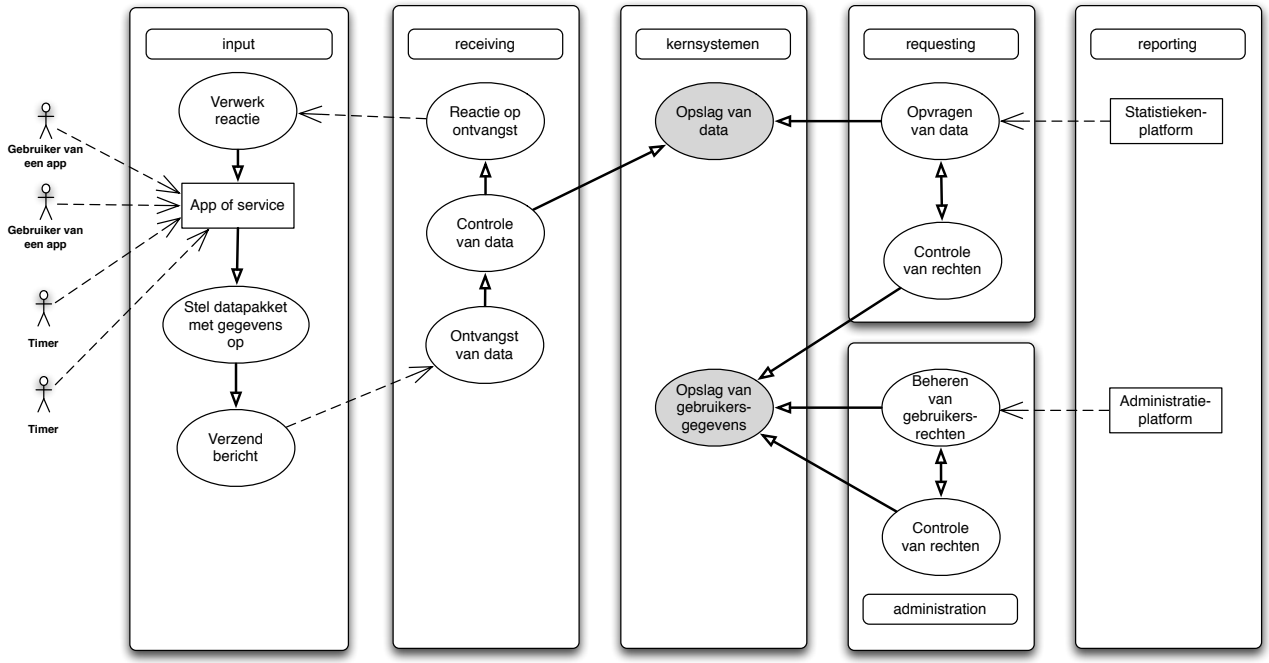


Figuur 7: Segmentatie inclusief administration-module



Figuur 8: Reactie op aankomst van data bij receiving-module

Tot en met figuur 6 was nog geen use case opgenomen waarin bevestigd wordt of verzonden data goed in het systeem opgenomen wordt. Hiertoe zijn in figuur 8 de onderdelen 'Controle van data' en 'Reactie op ontvangst' opgenomen. In het controle-segment wordt de ontvangen data geanalyseerd en gefilterd. Indien alle checks succesvol doorlopen worden zal het bericht in de database opgeslagen worden en zal een positief bericht naar de oorspronkelijke zender gestuurd worden. Indien er tijdens de controle fouten optreden zal een negatief bericht teruggestuurd worden.



Figuur 9: Controle van rechten en details input-segment

Het systeem bevatte al een methode om rechten te veranderen, maar voerde nog geen controle uit op het wel of niet bestaan van permissies bij een bepaalde aanvraag. Ook waren de verantwoordelijkheden van de input-module nog niet eerder getoond.

In figuur 9 zijn de verificatie van rechten en een verduidelijking van de werking van de input-module opgenomen.

3.4. Lijst van eisen

Uit het onderzoek van het bestaande systeem en een analyse van de gewenste functionaliteit is een lijst van eisen ontstaan. Deze lijst is verdeeld in een sectie van technische eisen, welke vastleggen wat het systeem moet kunnen en hoe bepaalde functies geïmplementeerd moeten worden, en een sectie van business-eisen, waarin genoemd wordt waar het systeem aan moet voldoen om commercieel inzetbaar te zijn.

3.4.1. Technische eisen

- Het systeem moet geschikt zijn voor groei. Deze groei kan in verschillende voorspelbare en onvoorspelbare richtingen plaatsvinden, waaronder toename van het aantal client-platformen, exponentiële groei van het aantal inkomende berichten, groei in de rapportage-wensen van klanten, enzovoorts.
- Het systeem moet snel zijn, zowel bij het inkomen, verwerken, opslaan als leveren van informatie.
- Het systeem moet veilig zijn. Gegevens in de database zullen voor een groot deel vertrouwelijk zijn. Indien een gedeelte of de totaliteit van de data verloren raakt of aan de verkeerde personen openbaar gemaakt wordt kan dit grote gevolgen hebben voor Coding Dutchmen en / of haar klanten.
- De (door-)ontwikkeling van de opgeleverde producten moet overdraagbaar zijn.

- De verschillende api's moeten een complete wrapper voor de database zijn; de ontwikkelaars van de grafische gebruikersinterfaces moeten in staat zijn via de api's alle benodigde gegevens op te vragen en bewerkingen te kunnen doen.
- De opgeleverde producten moeten schaalbaar zijn.
- Ontwikkelaars van iPhone-, Android-, Windows Mobile- of desktopapps die gebruik gaan maken van het AppStats2-platform moeten volledige vrijheid hebben over de te verzenden berichten en hun inhoud.
- Uit gebruik van het oude AppStats-platform is gebleken dat berichten met een 'Subject-Content'-structuur de beste balans tussen efficiëntie en flexibiliteit leveren. Het subject van een verzonden bericht is hierbij verplicht (bijvoorbeeld 'Startup'). Het content-gedeelte is optioneel en bevat specifiekere (eventueel meetbare) data, bijvoorbeeld de sessieduur of de taal van de telefoon. Bij het AppStats2-platform dient deze structuur ook gebruikt te worden.
- De implementaties van het input-segment moeten berichten direct naar de receiving-module verzenden als deze bereikbaar is. Als de receiving-module niet te bereiken is, bijvoorbeeld door het ontbreken van een internetverbinding, dienen berichten lokaal opgeslagen te worden. Vervolgens moet periodiek gekeken worden of een verbinding met de database gemaakt kan worden en moeten de berichten verzonden worden zodra de connectie beschikbaar is.

3.4.2. Business-eisen

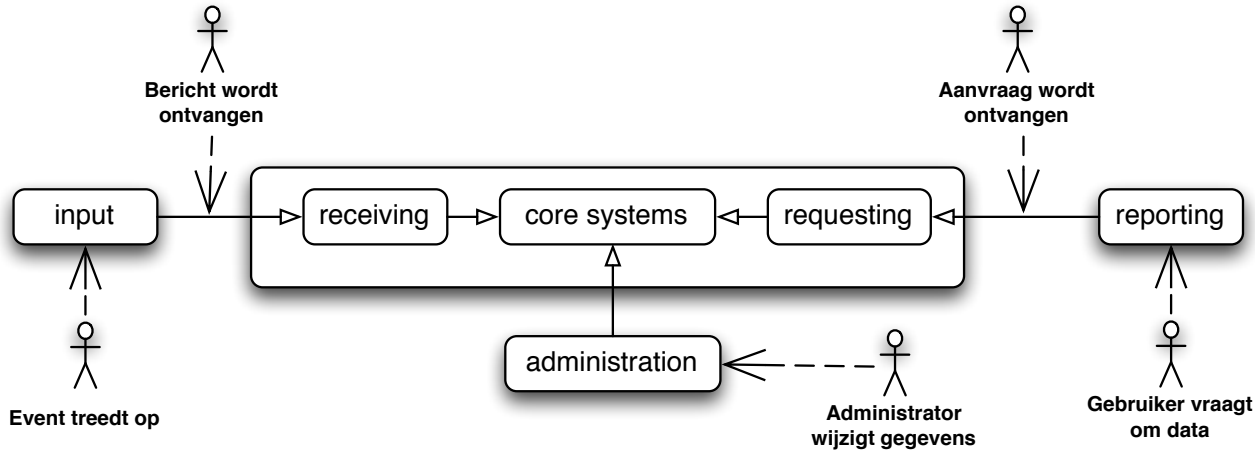
- De opgeleverde producten moeten zonder tussenkomst van ontwikkelaars en / of beheerders blijven functioneren.
- De eindgebruikers moeten op een gebruiksvriendelijke manier toegang krijgen tot de gewenste gegevens. De verantwoordelijkheid hiervoor ligt gedeeltelijk bij de ontwikkeling van intuïtieve gebruikersinterfaces (segment reporting), maar ook in het aanleveren van relevante informatie, bijvoorbeeld door het aggregeren, sorteren en filteren van gegevens, wat onder de verantwoordelijkheid van de requesting-module valt
- De implementaties van het input-segment dienen berichten zo snel mogelijk naar de database te schrijven, zodat de rapportage van de gegevens dichtbij real-time ligt.

4. Toepassing

4.1. Inleiding

De toepassing-fase omvat de implementatie van het functioneel ontwerp. Hierbij worden de verdeling in zes segmenten (input, receiving, core systems, requesting, administration, reporting) en de lijst van eisen (paragraaf 3.4) als belangrijkste leidraden gebruikt. De implementatie van het functioneel ontwerp wordt op basis van gemaakte keuzes verwerkt tot het technisch ontwerp in paragraaf 4.2.

Het technisch ontwerp wordt beschreven aan de hand van de vijf interacties die direct of indirect op het systeem uitgevoerd kunnen worden:



Figuur 10: Use cases interactie met het systeem

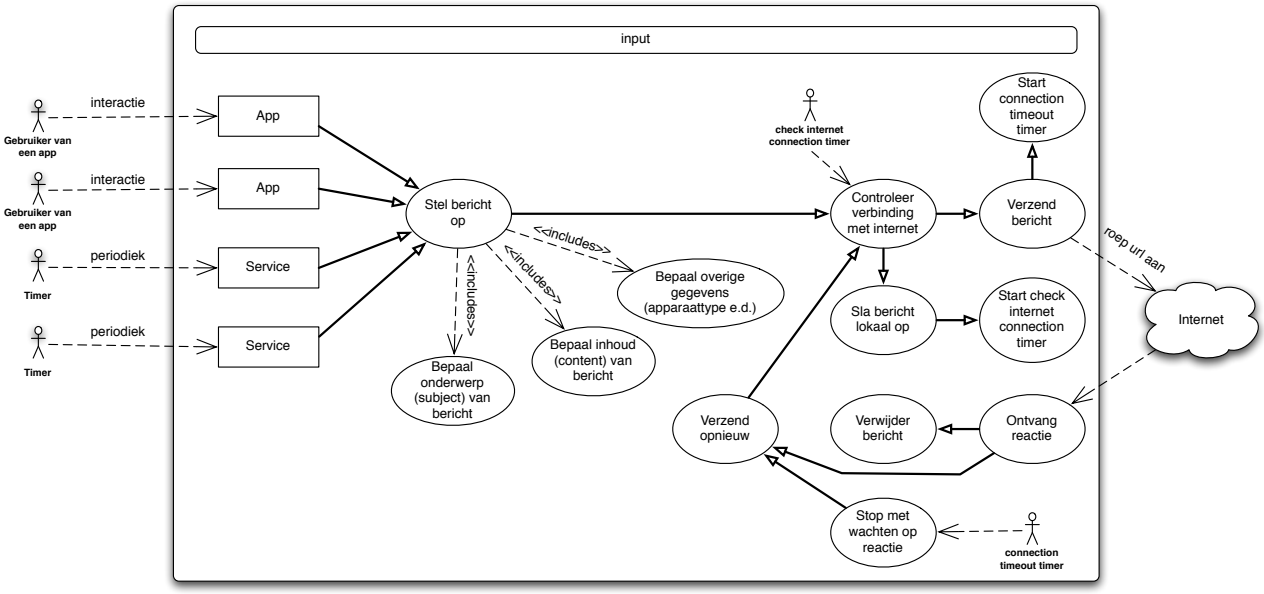
Onderstaand wordt het contextdiagram uit het functioneel ontwerp uitgebreid met usecases voor de vijf interacties in figuur 10. Deze use cases staan aan de basis van het technisch ontwerp.

4.1.1. Event treedt op (input)

Het input-segment bestaat uit de applicaties en services die berichten naar het AppStats2-platform verzenden. Applicaties verzenden hun berichten op basis van een interactie met de gebruiker, bijvoorbeeld als er op een knop gedrukt wordt. Services genereren met een vaste periode een bericht, bijvoorbeeld iedere minuut of ieder uur.

Het optreden van een event - door een gebruiker of een timer - wordt geïllustreerd in figuur 11. Hierin staat ook globaal uitgelegd hoe omgegaan moet worden met het ontbreken van een internetverbinding of een optreden van een timeout bij het verzenden van data over internet.

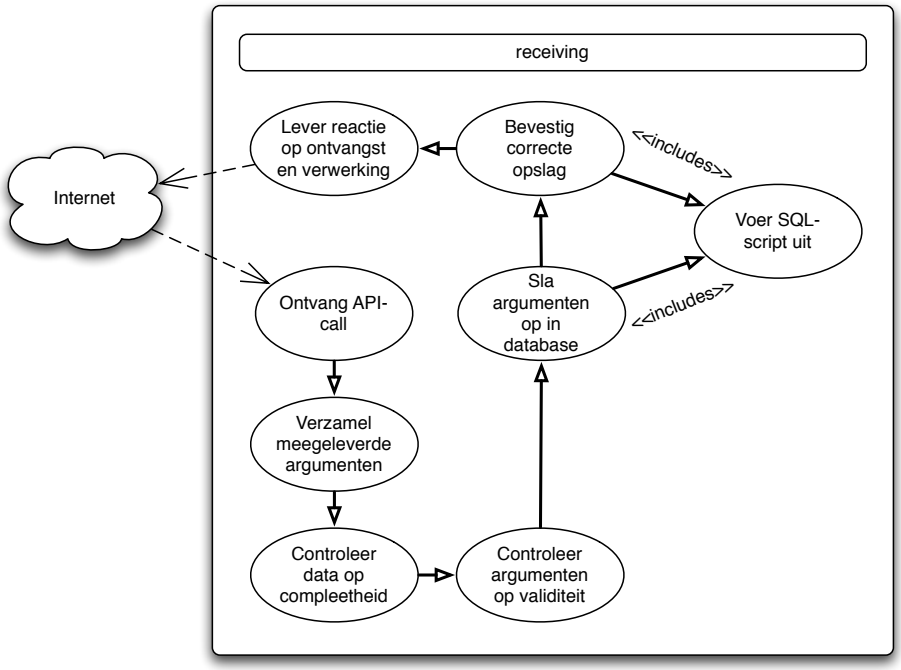
In de usecase wordt ook een duidelijke verwijzing naar de structuur van de te verzenden berichten gemaakt, zoals besproken in paragraaf 3.2.1: een bericht bestaat uit een subject, content en een verzameling andere gegevens. De precieze definitie van alle onderdelen van een te verzenden bericht is te vinden in bijlage C, waar vastgelegd staat welke argumenten de receiving-module verwacht bij het binnenkomen van een bericht.



Figuur 11: Use case input-module

4.1.2. Bericht wordt ontvangen (receiving)

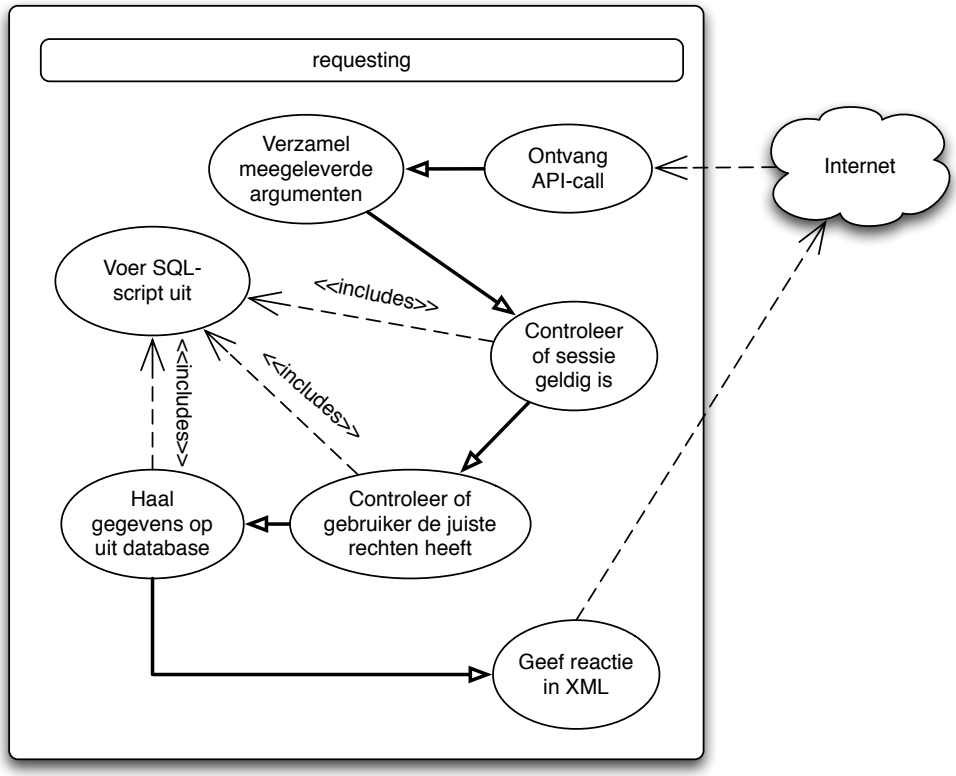
Na het verzenden van het bericht door het input-segment wordt de aanroep van de URL door het internet verwerkt. Van deze verwerking kan niet voorspeld worden hoe lang hij zal duren, maar als het goed is komt het bericht na een bepaalde periode aan bij de receive-api. Deze api verwacht bepaalde argumenten (zoals vastgelegd in bijlage C) en zal deze controleren en verwerken volgens de use case in figuur 12.



Figuur 12: Use case receiving-module

4.1.3. Aanvraag wordt verzonden (requesting)

Een aanvraag voor data wordt ontvangen door de request-api. Deze api is een onderdeel van de requesting-module, dat aanvragen voor data verwerkt en reactie geeft in de vorm van XML-files. In het segment requesting worden ook controles op gebruikersrechten uitgevoerd.

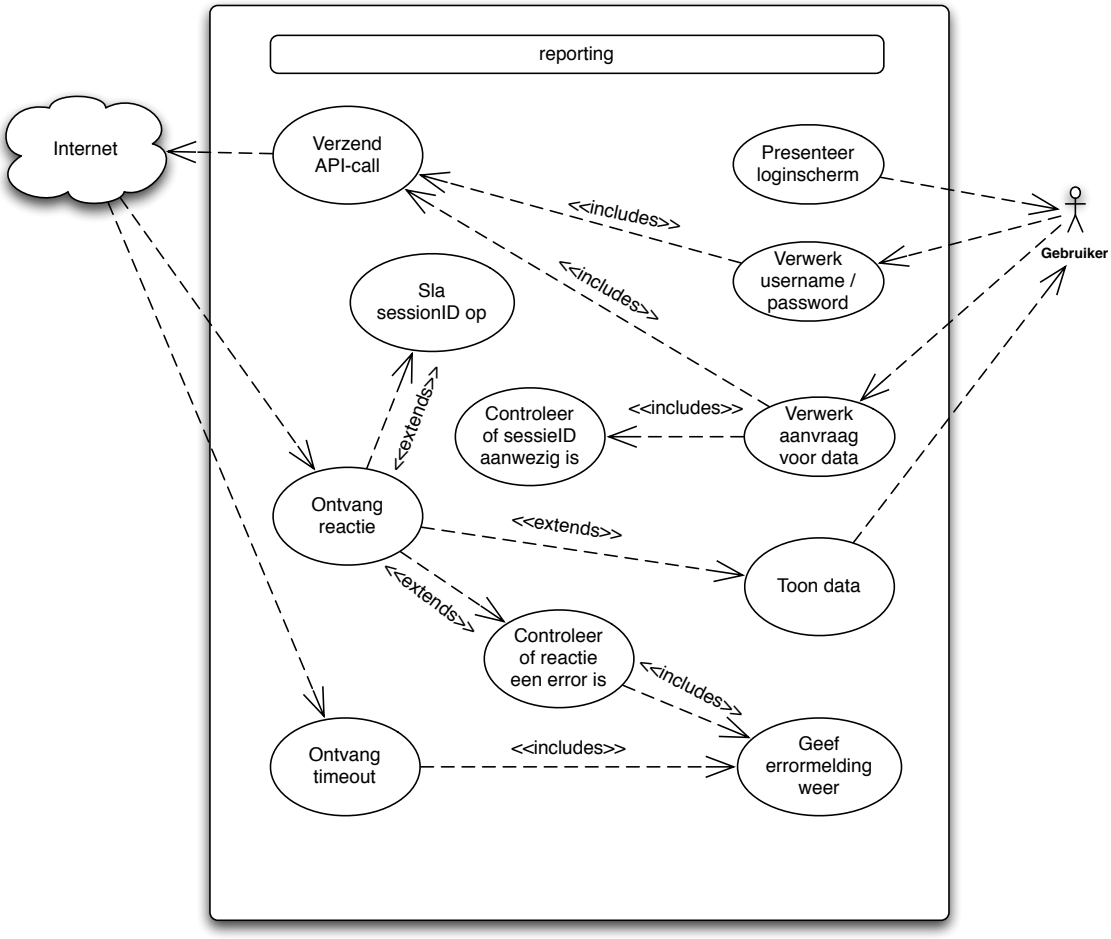


Figuur 13: Use case requesting-module

4.1.4. Gebruiker vraagt om data (reporting)

Gegevens uit de databases worden weergegeven in het segment reporting. Aan de basis van dit segment staan gebruikers die een aanvraag voor data doen, bijvoorbeeld door op een website een grafiek voor een bepaalde applicatie op te vragen. De verantwoordelijkheden van het reporting-segment zijn het verwerken van de aanvraag, het tonen van de data, het controleren van de gebruikersgegevens en het afhandelen van eventuele foutmeldingen.

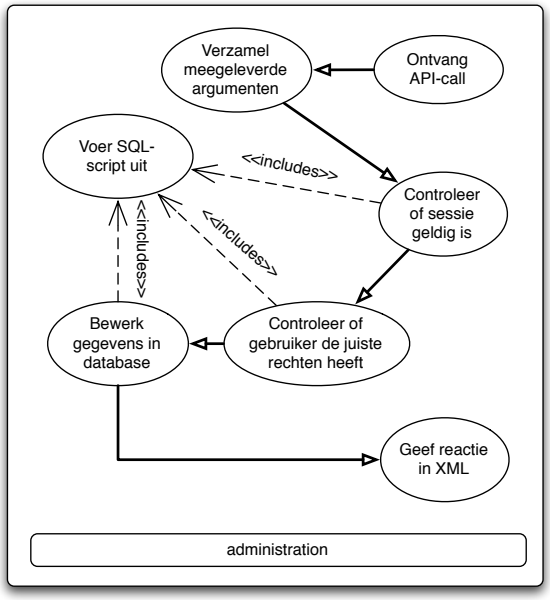
Zie voor een schematische weergave van deze verantwoordelijkheden figuur 14.



Figuur 14: Use case reporting-segment

4.1.5. Het beheren van gebruikersrechten (administration)

Het beheren van gebruikersrechten vindt op vrijwel dezelfde manier plaats als het opvragen van gegevens (zie paragraaf 4.1.3). Zie figuur 15 voor een illustratie van de verantwoordelijkheden van het administration-segment.



Figuur 15: Use case administration-module

4.2. Technisch ontwerp

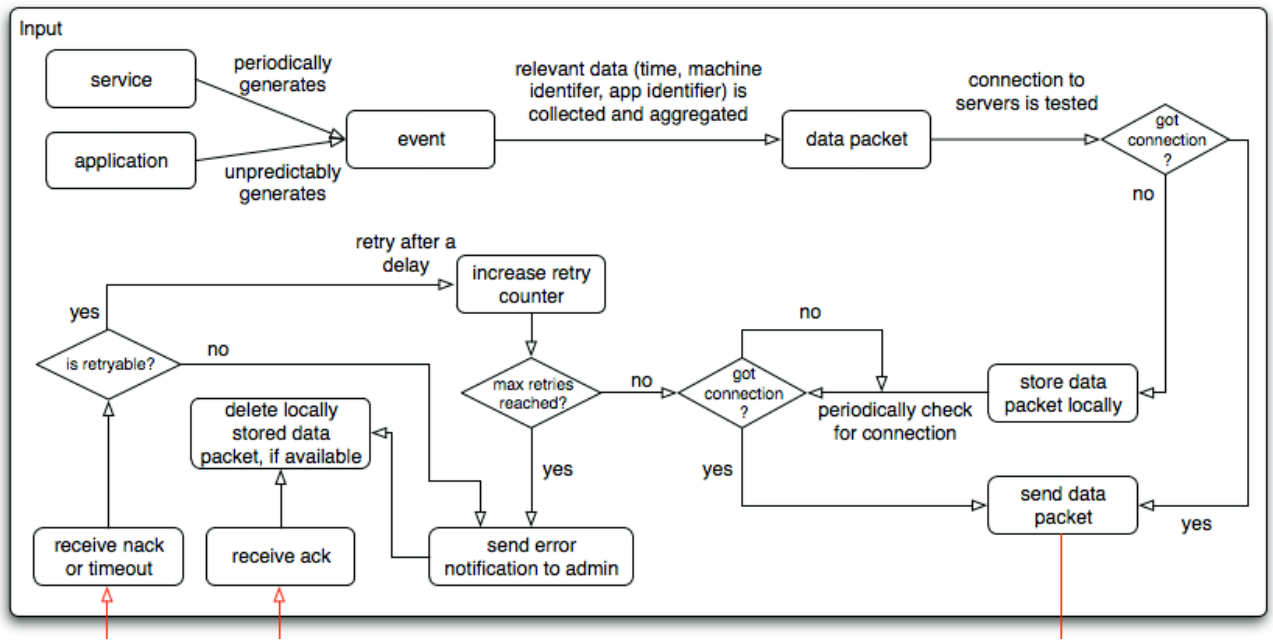
Dit hoofdstuk beschrijft de werking van de software in het AppStats2-platform. In het technisch ontwerp zullen dezelfde use cases aangehouden worden als in bovenstaande inleiding, waarbij voornamelijk dieper ingegaan zal worden op de werking van de verschillende modules.

4.2.1. Optreden van een event (input)

Beschrijving

Onder de registrerende systemen vallen de verschillende implementaties van het Input-segment. Hieronder vallen twee varianten: applicaties en services. Deze twee systemen hebben veel overeenkomsten, maar variëren in de periode waarmee ze berichten verzenden, waardoor ook de verwerking van de verzonden gegevens (later in het systeem) varieert.

Werking



Figuur 16: Flowchart input-module

[a1.0] Aan de kern van een applicatie staan gebeurtenissen waarvan niet op voorhand bepaald kan worden wanneer ze plaats zullen vinden. Deze gebeurtenissen ontstaan door een interactie met de gebruiker en worden asynchronous events genoemd. Bij services staat wel vast met welke periode ze gegevens verzenden, bijvoorbeeld eenmaal per minuut of viermaal per uur, maar is de inhoud van het bericht van belang. Deze events zullen gegenereerd worden door een timer en worden periodic of synchronous events genoemd.

[a1.1] Bij het optreden van een synchronous of asynchronous event wordt een datapakket opgesteld waarin het event en relevante data (bijvoorbeeld het tijdstip, een unieke identifier van het apparaat waar het event plaatsvindt, het versienummer van het OS) verzameld wordt.

[a2.0] Na het opstellen wordt gecontroleerd of communicatie met de servers mogelijk is. Dit gebeurt door middel van een aantal stappen, waaronder het controleren van de internetverbinding en het handshaken met de servers en databases. Als alle stappen succesvol doorlopen worden, gaat het systeem door naar [a5.0]. Indien er tijdens het testen van de verbinding problemen optraden wordt doorgegaan naar stap [a3.0].

[a3.0] Indien dit nog niet gebeurd is, wordt het datapakket opgeslagen in lokaal persistent geheugen. Ook wordt een vlag gezet waarmee aangegeven wordt dat er nog onverzonden pakketten aanwezig zijn.

[a4.0] Het systeem zal periodiek testen of de verbinding beschikbaar gekomen is. Als deze beschikbaar komt zal het systeem proberen het datapakket te verzenden (stap [a5.0]).

[a5.0] Het datapakket wordt verzonden naar de servers.

[a6.0] Het systeem ontvangt een reactie van de servers. Deze reactie kan zijn dat het datapakket succesvol verwerkt is, waarna doorgegaan wordt naar stap [a6.1]. Ook is het mogelijk dat een probleem opgetreden is, waarmee het systeem in stap [a7.0] terechtkomt.

[a6.1] Het bericht is aan de kant van de servers succesvol verwerkt en zal van het lokale systeem verwijderd worden.

[a7.0] Het systeem krijgt een negatieve reactie terug. Deze reactie bevat een melding of het bericht nogmaals verzonden kan worden (door naar stap [a8.0]), of dat het systeem het datapakket definitief niet kan verwerken stap [a9.0]. Ook is het mogelijk dat er helemaal geen reactie terugkomt, bijvoorbeeld door het wegvallen van het internet tijdens het verzenden. In dit geval gaat het systeem ook door naar stap [a8.0].

[a8.0] Na een vastgestelde periode probeert het systeem het datapakket opnieuw te versturen. In een counter wordt bijgehouden hoe vaak het pakket verzonden is.

[a8.1] Als het aantal pogingen om het pakket opnieuw te verzenden groter is dan een vastgestelde limiet gaat het systeem door naar stap [a9.0]. Indien deze limiet nog niet bereikt is, komt het systeem terug in stap [a2.0].

[a9.0] Het pakket kan definitief niet verzonden worden. Hiervan wordt melding gemaakt in een log of een bericht aan een administrator, zodat onderzocht kan worden waarom deze situatie plaatsvond. Na het maken van de melding wordt het datapakket verwijderd uit het lokale geheugen.

Samenstelling van datapakket

Bij punt [a1.1] worden de volgende gegevens verzameld:

1. Remote Timestamp - Het tijdstip waarop het event optrad.
2. Sender Type - Het type van het verzendende systeem. Dit kan een application of service zijn.
3. Device Identifier - Een unieke identifier voor het toestel of systeem waar het event plaatsvond. Voor een telefoon kan dit een serienummer zijn, voor een server een MAC-adres.
4. Online Status - Een getal dat aangeeft of het bericht direct bij het optreden verzonden werd, of eerst lokaal opgeslagen werd.
5. OS Version - Het versienummer van het besturingssysteem. Hiermee kan bepaald worden wat de verdeling van besturingssystemen voor een bepaalde applicatie is. Hiermee kan bijvoorbeeld geanalyseerd worden of een error, actie of ander event vaker voorkomt op specifieke besturingssystemen.
6. Application Key - Een unieke hexadecimale waarde voor een applicatie of service.
7. Subject - Het onderwerp van het opgetreden event.
8. Content - De inhoud van het onderwerp.
9. Application Version - De versie van de applicatie of service waar het event optrad. Dit kan gebruikt worden om fouten naar een bepaalde versie te traceren, maar ook om de analyseren hoeveel gebruikers de applicatie naar de nieuwste versie updaten.
10. Hardware Family - De serie waar de telefoon deel van uit maakt, bijvoorbeeld iPhone, iPod touch, Android of Windows Phone 7.
11. Machine Identifier - Een code die aanduidt van welk toestel of systeem het bericht verzonden is. Dit kan bijvoorbeeld iPhone2,1 zijn voor de iPhone 3G.

Bovenstaande gegevens worden samengevoegd en in één keer naar de AppStats2-servers verzonden. Daar zal gecontroleerd worden of de gegevens compleet zijn (zie paragraaf 4.2.2) en zullen ze verwerkt worden.

Relevante punten uit de probleemstelling

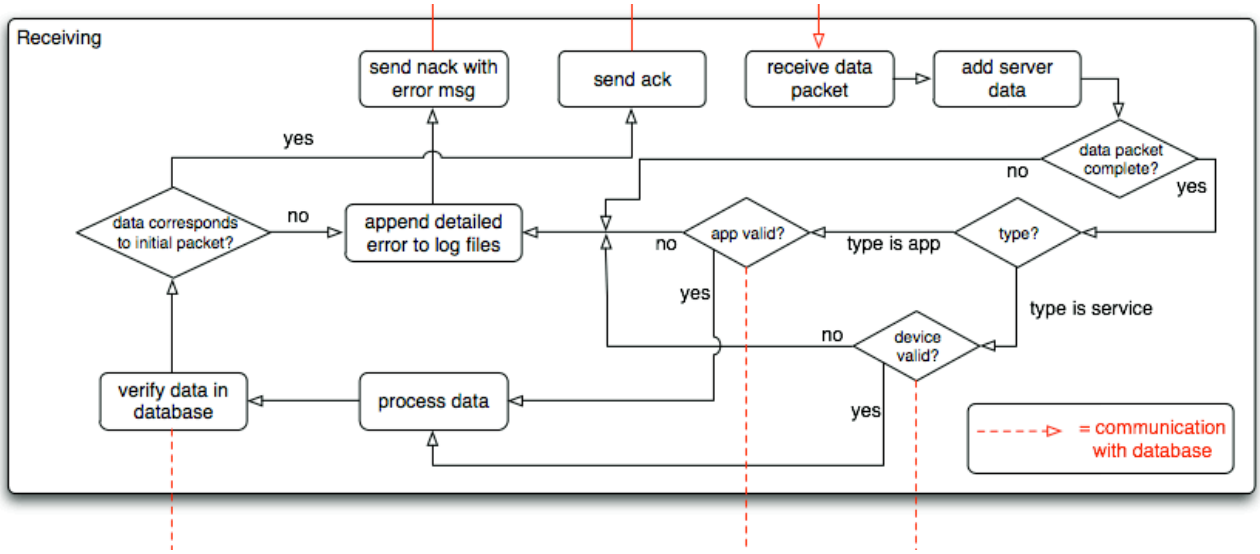
Met de beschreven werking van het Input-segment worden punten 1 en 5 uit de probleemstelling verholpen. Deze punten beschrijven respectievelijk het ontbreken van de optie om events te registreren als het systeem niet verbonden is met het internet en het ontbreken van een bevestiging dat een datapakket succesvol verwerkt is.

4.2.2. Het verzenden en ontvangen van een bericht (receiving)

Beschrijving

De data die door services en applicaties verzonden wordt komt aan bij de Receive-API. Deze API verwerkt en controleert de binnenkomende gegevens en zal een reactie naar de oorspronkelijke zender van de geven waarin gespecificeerd staat welke problemen eventueel opgetreden zijn.

Werking



Figuur 17: Flowchart receiving-module

[b1.0] Het datapakket komt via HTTPS binnen. Een ander kanaal voor communicatie zal in de eerste versies van het platform niet beschikbaar zijn, omdat we geen makkelijk te implementeren alternatieven voor het gecodeerd verzenden van data hebben. Codering van de verzonden gegevens is nodig zodat datapakketten niet door andere partijen dan Coding Dutchmen uitgelezen kunnen worden. Hiermee wordt misbruik van het systeem voorkomen.

[b2.0] Van de binnenkomende gegevens wordt gecontroleerd of ze compleet zijn. In deze controle kunnen ook filters toegepast worden, mocht hier op een later moment behoefte aan zijn. Als de gegevens compleet zijn gaat het systeem verder met de verwerking van de data [b3.0]. Indien er gegevens ontbreken gaat het systeem door naar stap [b6.0].

[b3.0] Uit de binnenkomende gegevens wordt eerst opgemaakt of de zender een applicatie [b3.1] of service [b3.2] is.

[b3.1] Van de applicatie wordt bekeken of hij geldig is. Het is mogelijk dat de applicatie geblokkeerd wordt, bijvoorbeeld vanwege het ontbreken van een licentie of het breken van de gebruiksvoorwaarden. Als het datapakket geen problemen oplevert gaat het systeem door naar stap [b4.0], anders naar [b6.0].

[b3.2] Ook een service kan geblokkeerd worden, maar de reden hiertoe zal waarschijnlijk bij het verzendende apparaat liggen. Als een server bijvoorbeeld HTTP-requests naar het AppStats2-platform verzendt zonder dat hij in het systeem bekend is, zal zijn bericht in deze fase afgekeurd worden. Het systeem komt dan in stap [b6.0] terecht. Als alle gegevens in orde zijn gaat het systeem door naar stap [b4.0].

[b4.0] De gegevens worden opgesplitst en in de juiste tabellen van de database opgeslagen.

[b5.0] De gegevens die zojuist weggeschreven zijn worden opgevraagd uit de database en vergeleken met de oorspronkelijke data in het datapakket. Als deze gegevens niet overeenkomen is er een fout opgetreden en komt het systeem in [b6.0] terecht.

[b5.1] De gegevens zijn weggeschreven en gevalideerd. Er wordt een bericht van geslaagde verwerking terug naar de oorspronkelijke zender gestuurd.

[b6.0] Er is een fout opgetreden. De details van deze fout worden in logfiles weggeschreven zodat later bekeken kan worden waar in het systeem eventuele problemen of bottlenecks zitten.

[b6.1] Na het schrijven van een logfile wordt een errormessage met beknopte beschrijving naar de oorspronkelijke zender verstuurd. In dit bericht wordt ook opgenomen of de zender het oorspronkelijke bericht opnieuw moet sturen (bijvoorbeeld bij corrupte data) of dat dit geen zin heeft (bijvoorbeeld bij blokkering van een server of applicatie).

Toevoegen en controle gegevens

Bij het ontvangen van data zullen eerst onderstaande gegevens aan het binnenkomende data packet toegevoegd worden:

Element	Beschrijving	Type	Lengte	Voorbeeld
senderIP	Het IPV4-adres van de zender van het bericht.	String	7-15	“94.124.138.194”
receivedTimestamp	De tijd waarop het event door het platform ontvangen werd.	Date (ISO 8601)		2010-02-12T15:19:21+01:00

Tabel 1: Toegevoegde elementen van het datapakket

Samen met de 11 gegevens die door de verzendende applicatie of service ingevuld waren, zijn nu in totaal 13 velden met data beschikbaar. Deze 13 velden worden aan een DataPacket-object toegevoegd, waarbij per veld gecontroleerd wordt of aan de gestelde eisen voldaan wordt.

Als het DataPacket-object succesvol aangemaakt is, wordt gecontroleerd of de gegevens aan de database toegevoegd mogen worden. Onderdelen waarop gecontroleerd wordt zijn:

1. Als de zender een applicatie is, bestaat deze dan en is deze niet geblokkeerd?
2. Als de zender een service is, is bestaat deze dan en is de zendende machine niet geblokkeerd?
3. Is het verzendende IP-adres niet geblokkeerd?

Als deze stappen succesvol doorlopen worden, gaat het systeem door met het verwerken van de gegevens in het datapakket.

Verzendmethode
Voor het verzenden van een pakket wordt HTTPS POST gebruikt. Hiertoe is gekozen om de volgende redenen:

1. HTTP is het meestgebruikte protocol op het internet. Het protocol is betrouwbaar, breed inzetbaar en snel.
2. HTTPS voegt een aantal beveiligingslagen toe aan HTTP. Hiermee kan voorkomen worden dat een andere server dan die van Coding Dutchmen zich voordoet als de ontvangende partij (waardoor gegevens bij de verkeerde personen terecht zou komen). Ook wordt de verzonden data versleuteld, waardoor de inhoud van het bericht niet door onbevoegden uitgelezen kan worden.
3. POST is een van de standaardmethoden om data tussen een client en server te verzenden. Een andere methode om data te verzenden is GET. Er is echter voor gekozen deze niet te gebruiken, omdat alle parameters bij GET in de URL opgenomen worden, wat tot een lange URL kan leiden. Sommige browsers, webservers, firewalls en proxy's blijken problemen te hebben met URL's vanaf een bepaalde lengte (bij welke lengte de problemen optreden is variabel). Bij de POST-methode worden de parameters in de *body* verzonden. De grootte hiervan is ongelimiteerd, waardoor het geschikter is voor gebruik bij het AppStats2-platform.

Meer informatie over de inhoud van het request wordt gegeven in Bijlage C: Request API

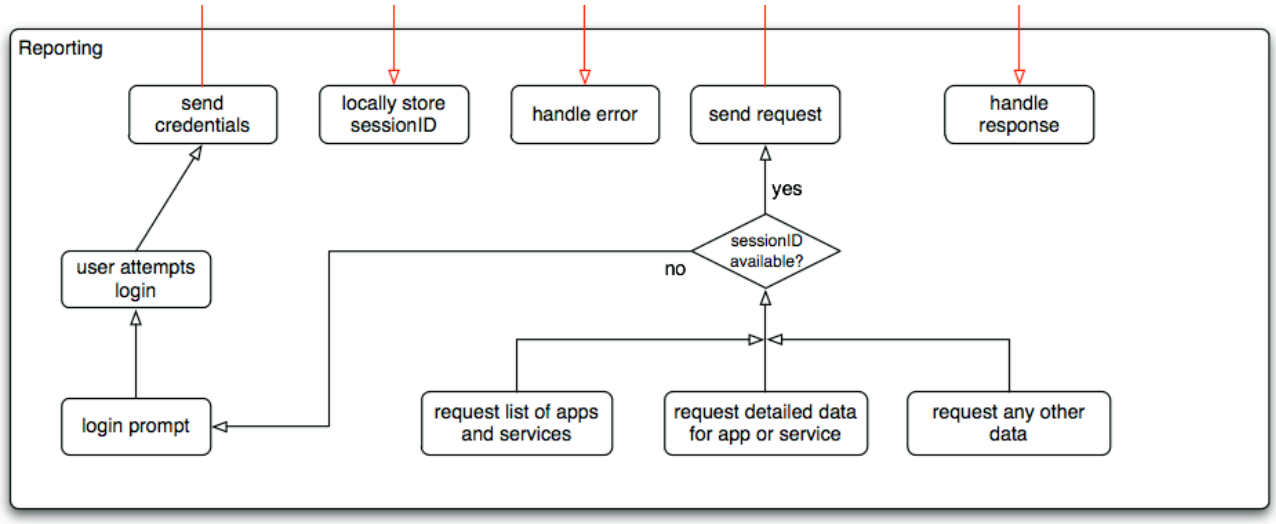
Relevante punten uit de probleemstelling
Met de beschreven werking van de receiving-module worden de volgende punten uit de probleemstelling (paragraaf 3.2.2) aangepakt:

- Punt 2: verschillende clients hoeven nog maar met één script op de server te communiceren.
- Punt 3: de correctheid en compleetheid van de binnenkomende gegevens wordt gecontroleerd.
- Punt 4: de opslag van binnenkomende gegevens wordt gevalideerd.
- Punt 5: de API stuurt een bericht terug naar de oorspronkelijke zender, waarin gespecificeerd staat of het datapakket succesvol verwerkt is, en zo niet, welke problemen optraden.

4.2.3. Het tonen van data (reporting)

Beschrijving
De geregistreerde gegevens moeten door Coding Dutchmen en haar klanten op een overzichtelijke manier uitgelezen kunnen worden. Hiertoe bestaan de verschillende implementaties van het segment Reporting. Het is aan de ontwikkelaars van deze onderdelen om zelf te bepalen welke gegevens uit de database ze beschikbaar maken, waardoor onderstaande werking generiek is opgezet; alle gewenste data moet volgens deze werking opgevraagd kunnen worden.

Werking



Figuur 18: Flowchart reporting-module

[c1.0] Het opvragen van gegevens ban de server begint altijd met het invullen van de gebruikersnaam en wachtwoord. Dit kan door de gebruiker zelf of geautomatiseerd door het reporting-systeem verzorgd worden (bijvoorbeeld bij een inlog op basis van een cookie in een webbrowser).

[c1.1] De ingevulde gegevens worden naar de server verzonden. Hier komen twee mogelijkheden uit voort: de server reageert met een error, bijvoorbeeld als de gebruikersnaam en wachtwoord incorrect zijn [c2.0] of de server geeft een sessionID terug, welke in toekomstige communicatie met de server gebruikt kan worden [c3.0].

[c2.0] De server geeft een errorcode en een beschrijving van de opgetreden fout. De ontwikkelaar van het reporting-segment kan nu kiezen om deze error in een melding te tonen of er op een andere manier mee om te gaan (bijvoorbeeld een nieuwe request doen met dezelfde gegevens als de keer daarvoor).

[c3.0] De server heeft een sessionID teruggegeven. Deze wordt lokaal opgeslagen en gebruikt bij alle volgende requests.

[c4.0] De gebruiker doet een aanvraag, bijvoorbeeld voor een lijst van applicaties of detailinfo voor een bepaalde service. Hierbij wordt gecontroleerd of de sessionID beschikbaar is. Als deze er is gaat het systeem verder met de request [c5.0], als hij niet beschikbaar is komt het systeem weer op zijn beginpunt [c1.0] uit.

[c5.0] Een request wordt naar de server gestuurd. Deze request kan wederom een error bevatten [c2.0] of het verwachtte antwoord teruggeven [c5.1]. Ook kan er een timeout optreden, bijvoorbeeld als het opvragen van gegevens te lang duurt [c5.2].

[c5.1] De server geeft het antwoord op een request. Dit antwoord kan door het systeem gebruikt worden om grafieken te tonen, lijsten te vullen, nieuwe requests te doen, etcetera.

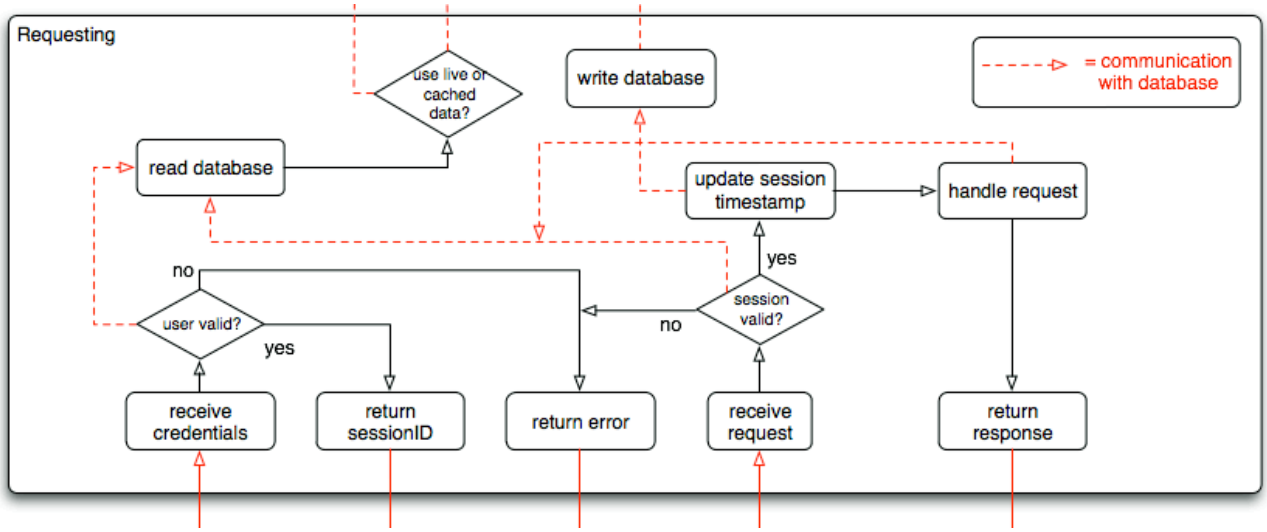
[c5.2] Een timeout is opgetreden. Het is aan de ontwikkelaar van het systeem om te kiezen hoe hij hier mee om wil gaan.

Security
Om te voorkomen dat ongeautoriseerde personen via het Reporting-segment toegang krijgen tot de gegevens in de database worden HTTPS en sessies toegepast. HTTPS wordt gebruikt om gegevens tussen de clients en servers te versleutelen. Sessies zorgen ervoor dat de gebruikersnaam en het wachtwoord maar één keer verzonden hoeven te worden en dat gebruikers maar op één plek tegelijk ingelogd kunnen zijn.

4.2.4. Het verwerken van een aanvraag van data (requesting)

Beschrijving
De gegevens die in 4.2.3 aan de gebruiker getoond worden, moeten uit de database gehaald worden, waarbij per aanvraag gekeken moet worden of de gebruiker toestemming heeft om de data te bekijken, en of de data uit de live database of een cache gehaald moet worden.

Werking



Figuur 19: Flowchart requesting-module

[d1.0] Een aanvraag voor data gebeurt altijd op basis van een sessionID, welke gekoppeld is aan een gebruiker, IP-adres en een aantal andere gegevens. Om een sessionID te krijgen moet de gebruiker zijn username en wachtwoord aan de Request-API aanbieden, waarna in de database gekeken wordt of deze gegevens kloppen [d1.1]. Indien de gebruiker een verkeerde username-password-combinatie heeft opgegeven of door het systeem geblokkeerd is, wordt een error teruggegeven [d2.0].

[d1.1] De gebruikersgegevens blijken in orde te zijn. In de database wordt gekeken of de gebruiker al een geldige sessie heeft lopen, en als dit niet zo is, wordt een nieuwe sessie aangemaakt. Uit beide mogelijkheden komt een sessieID voort, welke samen met wat begeleidende informatie door de API teruggegeven wordt aan het inloggende systeem.

[d2.0] Een error wordt in XML-formaat teruggegeven. Deze XML bestaat uit een numerieke code en een beschrijving, waardoor het systeem dat de error ontvangt deze op een nette manier kan afhandelen.

[d3.0] De Request-API ontvangt een aanvraag voor gegevens. Deze aanvraag kan alles zijn, bijvoorbeeld:

- Geef de lijst van applicaties die deze gebruiker mag inzien.
- Geef de lijst van bedrijven waar deze gebruiker toe behoort.
- Geef de applicaties voor een specifiek bedrijf.
- Geef de gebruikscijfers over de afgelopen 24 uur voor applicatie X.
- Geef de gemiddelde belasting voor service Y over de afgelopen week.
- Geef de verdeling van gebruikte operating systems voor applicatie X en applicatie Y.

Ongeacht welke aanvraag gedaan wordt, wordt altijd eerst gekeken of het meegegeven sessionID in orde is, wat gebeurt in stap [d3.1].

[d3.1] In de database wordt de sessionID opgezocht. Deze sessionID is gekoppeld aan een userID en alle bij een gebruiker horende gegevens, waaronder de bedrijven en groepen waar hij of zij bij hoort. Indien de sessie verlopen is wordt een error teruggegeven en komt het systeem terug in stap [d2.0]. Als de gegevens in orde zijn wordt de sessie vernieuwd (de sessie-timeout wordt gereset op zijn standaardwaarde) en komt het systeem in stap [d3.2] terecht.

[d3.2] Op dit moment is bevestigd dat de sessie valide is en zijn de gebruikersdata beschikbaar. Hiermee wordt gekeken of de gebruiker permissie heeft om antwoord te krijgen op zijn aanvraag. Als een gebruiker bijvoorbeeld een aanvraag heeft gedaan voor een applicatie die niet binnen zijn bedrijf valt, dan zal dat hier gedetecteerd worden en zal een error teruggegeven worden (stap [d2.0]). Als de rechten van de gebruiker in orde zijn komt het systeem in stap [d4.0] terecht.

[d4.0] Een XML met de gevraagde gegevens wordt teruggegeven aan de aanvragende gebruiker. Deze kan de gegevens gebruiken om te tonen op een website, in een applicatie, in een Excel-document, enzovoorts.

Verbinding met de database

In paragraaf 4.2.5 wordt beschreven hoe de database uit een live- en een cache-deel bestaat. Als de requesting-module een aanvraag voor data verwerkt, zal gekeken worden of deze data gecached beschikbaar is, of dat deze uit de live-database gehaald moet worden.

Als data weggeschreven wordt gebeurt dit altijd naar de live-database. In paragraaf 4.2.5 wordt beschreven hoe de gegevens later naar de cache-database gekopieerd worden.

Variatie in subjects

De data die in de database opgeslagen wordt kan op verschillende manieren gepresenteerd worden, bijvoorbeeld als taartdiagram, lijndiagram of tabel. De beste manier om gegevens te tonen is afhankelijk van het soort data. Een aantal voorbeelden van mogelijke data staat in tabel 2 beschreven, inclusief de manier waarop ze gepresenteerd kunnen worden:

Subject	Content	Presentatie
Startup	“FirstRun”, ”Resume”, ”AppDidCrash”, ””	Lijndiagram van de aantallen, taartdiagram van de percentages
News	Naam van het artikel	Lijndiagram van aantallen, tabel van meest gelezen nieuws
SessionDuration	Float	Lijndiagram van de gemiddelden

Subject	Content	Presentatie
ShowDownload	Naam van de show	Lijndiagram van aantallen, tabel van meest gedownloadde shows
Video	Naam van de video	Lijndiagram van aantallen, tabel van meest bekeken video's
Score	Integer	Lijndiagram van aantallen, tabel van meest gelezen nieuws
Language	De geselecteerde taal in de applicatie	Lijndiagram met verdeling per periode, taartdiagram van de percentages
LastMinuteMysqlQueries	Het aantal MySQL queries in de afgelopen minuut	Lijndiagrammen met minimale, gemiddelde en maximale waarden, tabel met pieken.
AdPressed	"iAd" (voor Apple-advertenties), "AdMob" (voor Google-advertenties)	Lijndiagram van aantallen, taartdiagram met verdeling, lijndiagram met verhouding tussen iAds en AdMob-ads.

Tabel 2: Voorbeelden van subjects en hun presentatie

Om de Subjects aan verschillende presentaties te koppelen is in de database een integer aangemaakt waar verschillende bits overeenkomen met de te gebruiken diagrammen. Dit is mogelijk door de binaire representatie van getallen; het decimale getal 1 ziet er binair uit als 00000001, het decimale getal 2 is 00000010, het getal 4 is 00000100, enzovoorts. Door een betekenis aan de individuele bits toe te kennen (de mogelijke betekenissen staan beschreven in tabel 3) kan met een decimale waarde aangegeven worden welke representatie gebruikt moet worden.

Een voorbeeld: een applicatie verzendt iedere keer dat hij opgestart wordt een bericht met als Subject "Startup" en als Content de waarde "AppDidCrash" (als de applicatie bij het vorige gebruik niet goed afsloot), de waarde "Resume" (als de applicatie uit de achtergrond gehaald werd), de waarde "FirstRun" (als de applicatie voor het eerst gestart werd) of een lege string als geen van de bovenste punten van toepassing is. Voor dit bericht gelden volgens tabel 3 parameters 3, 4, 6, 7, 8 en 10, wat overeenkomt met decimale waarden 4, 8, 32, 64, 128 en 512. Het decimale getal 748 (de som van de losse decimale getallen) is in binaire vorm 111101100. In dit getal (gelezen van rechts naar links) betekent iedere 1 dat een veld van toepassing is. Door het getal 748 in de database op te nemen staat dus vastgelegd staat hoe met het Startup-subject omgegaan moet worden.

Binair	Decimaal	Betekenis	Voorbeeld	Opmerkingen
1	1	In het content-veld staat een integer.	Het subject is "Score", in het content-veld staat een integer.	
2	2	In het content-veld staat een float.	Het subject is "SessionDuration", in het content-veld staat een float die de sessieduur in seconden weergeeft.	

Binair	Decimaal	Betekenis	Voorbeeld	Opmerkingen
3	4	In het content-veld staat een string.	Het subject is “News”, in het content-veld staat de titel van het geopende nieuwsartikel.	
4	8	Het aantal keer dat deze waarde in het Subject-veld voorkomt kan geteld worden.	Het subject is “Startup”, door het aantal berichten met dit subject te tellen kan gemeten worden hoe vaak een applicatie opgestart is.	
5	16	De waarden in het Content-veld kunnen geteld worden.	Het subject is “News”, door het aantal specifieke nieuwsartikelen te tellen, welke in het content-veld gedefinieerd staan, kunnen de meest populaire berichten bepaald worden.	
6	32	Toon de waarden van het Subject-veld in een lijndiagram.	Het subject is “Startup”, in een lijndiagram kan het verloop van het aantal gebruikers in een bepaalde periode getoond worden.	
7	64	Toon de waarden van het Content-veld in een lijndiagram.	Het subject is “AdPressed”, in een lijndiagram kan het aantal geopende Google-advertenties tegenover Apple-advertenties gezet worden.	Deze waarde moet enkel gebruikt worden bij beperkte variatie in het Content-veld.
8	128	Toon de waarden in een taartdiagram.	Het subject is “Language”, in een taartdiagram kan de verdeling van gebruikte talen getoond worden.	Een taartdiagram wordt altijd bepaald op basis van waarden in het content-veld. Bit 128 kan niet aan staan als bit 16 uit staat.
9	256	Toon de top 10 in een tabel.	Het subject is “News”, in een top 10 kunnen de meest bekeken berichten getoond worden.	De top 10 wordt altijd bepaald op basis van waarden in het content-veld. Bit 256 kan niet aan staan als bit 16 uit staat.
10	512	Dit bericht wordt verzonden bij het opstarten van een applicatie	Het subject is “Startup”	Bepaalde grafieken worden enkel getekend op basis van Startups, waardoor ze hun eigen code krijgen.
11	1024	Negeer deze berichten		Wordt gebruikt bij Subjects die niet meer getoond hoeven worden.

Tabel 3: Betekenis van binaire waarden bij subjects

4.2.5. De databases

Beschrijving

In paragraven 4.2.2 en 4.2.4 wordt respectievelijk beschreven hoe gegevens door de receiving-module in de database geschreven worden en hoe ze door de requesting-module uit de database gelezen worden. In paragraaf 4.2.6 komt ook de administration-module aan de orde, waarmee gegevens in de database bewerkt kunnen worden. De drie modules dienen dus als interfaces tussen de buitenwereld en de database. In dit hoofdstuk wordt beschreven hoe de databases zelf functioneren: hoe de snelheid van de levering van data hoog gehouden wordt, hoe toegang tot de database beperkt wordt tot enkel de personen die daar rechten toe hebben en hoe de gegevens zelf beschermd worden tegen verwijdering (per ongeluk of opzettelijk).

Performance

Er zijn op alle niveaus maatregelen genomen om de database snel te houden. De maatregelen op het laagste niveau bevinden zich in het ontwerp van de database: terugkerende gegevens (bijvoorbeeld de versie van een applicatie) worden in aparte tabellen opgenomen en volgens het principe van relationele databases via numerieke ID's aan andere tabellen verbonden.

Deze numerieke ID's staan ook een ontwerp toe waar variabele veldlengtes zoveel mogelijk vermeden worden. Variabele lengtes kunnen zorgen voor overhead, wat het zoeken in een tabel traag kan maken.

Om het zoeken in tabellen verder te versnellen wordt veelvuldig gebruik gemaakt van indices. Indices verhogen het schijfgebruik van een database, maar maken het sorteren en groeperen van data veel sneller.

Na bovenstaande optimalisaties, welke allemaal op de live-database toegepast zijn, blijft het zoeken en verzamelen van veel data nog steeds traag. Daarom worden secties van de database periodiek gekopieerd naar kleinere, snellere databases (caching). Deze databases bevatten dezelfde optimalisaties als de live-database (velden van vaste lengte en indices), maar bevatten slechts een fractie van de records, door bijvoorbeeld enkel de laatste 24 uur of de laatste week te kopiëren.

Security

Om te garanderen dat gegevens niet in de verkeerde handen vallen of data verwijderd wordt waar dit niet de bedoeling is wordt uitgebreid gebruik gemaakt van de user-structuur die MySQL biedt. Hiermee kunnen per SQL-gebruiker permissies toegekend worden voor het lezen, toevoegen, wijzigen en verwijderen van gegevens. Door deze rechten op het principe van least privilege toe te kennen wordt ervoor gezorgd dat een systeem dat enkel hoeft te lezen niet kan schrijven, een systeem dat enkel hoeft te schrijven niet kan lezen, en dat geen van beiden gegevens kunnen verwijderen.

De database wordt gehost op een fysiek van de webserver gescheiden machine. Toegang tot de database is enkel toegestaan vanaf één specifieke webserver, alle andere pogingen tot toegang worden door de firewall geblokkeerd.

De servers zelf staan in het datacenter van DCG in Amsterdam. Zij leveren 24-uurs bemande bewaking en toegang op basis van biometrische identificatie, waardoor fysieke toegang tot de servers voor onbevoegden vrijwel onmogelijk gemaakt wordt.

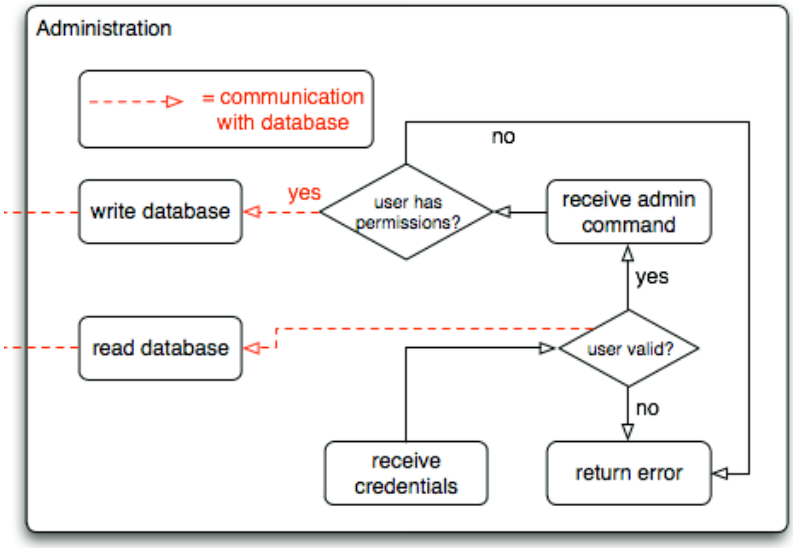
Backups
Ieder uur wordt de volledige database uitgelezen en opgeslagen als gecomprimeerd sql-script. Dit script wordt weggeschreven naar een fysiek andere machine. Op deze backup-machine worden backups van de afgelopen 24 uur (een per uur), backups van de afgelopen week (een per dag) en backups van het afgelopen jaar (een per week en een per maand) bewaard.

Deze backups kunnen in hun volledigheid of gedeeltelijk teruggezet worden, waarmee de kans op verlies van data geminimaliseerd wordt.

4.2.6. Het bewerken van gegevens in de database (administration)

Beschrijving
Na oplevering van het systeem moet het niet meer nodig zijn om de database met een SQL-editor te benaderen. Hiertoe dienen alle te voorziene bewerkingen op de database vastgelegd worden in een API die door gebruikers met de juiste permissies (administrators) gebruikt kan worden.

Ontwerp



Figuur 20: Flowchart administration-module

Rechten
In het systeem zijn verschillende niveaus van administrator-rechten te onderscheiden. Helemaal bovenaan de hiërarchie staan de administrators die iedere applicatie, service, user, bedrijf en groep kunnen wijzigen. Deze rol zal normaliter voorbehouden zijn aan enkele medewerkers van Coding Dutchmen.

Een stap lager bevinden zich de company admins, welke nieuwe gebruikers, applicaties en services aan hun eigen bedrijf kunnen toevoegen maar geen mogelijkheid hebben om gegevens van andere bedrijven te bekijken of wijzigen.

Het laagste niveau van administrators zijn de application- en servicemanagers van een bedrijf. Deze gebruikers kunnen simpele gegevens als de titels en labels van applicaties en services wijzigen, zonder dat dit invloed heeft op de werking van het systeem.

4.3. Gemaakte keuzes

4.3.1. Opslag van gegevens

In dit document wordt regelmatig gerefereerd aan data of gegevens. Er wordt echter vermeden te beschrijven om welke gegevens het precies gaat, of zelfs waar de limieten van de hoeveelheid gegevens ligt. De reden hiertoe is simpel: ten tijde van de ontwikkeling van het AppStats2-platform - en waarschijnlijk ook niet tijdens de eerste periode van gebruik - is niet bekend welke gegevens de gebruikers willen zien of hoe we de gegevens gaan presenteren.

Toen T-Mobile de iPhone in Nederland introduceerde werd hen tijdens een presentatie gevraagd of hun datanetwerk geschikt was voor de enorme verwachte toename in mobiel dataverkeer bij het ontbreken van datalimieten. T-Mobile's antwoord was: "Onze prioriteit is het leveren van geweldige service. Eventuele problemen lossen we op als we ermee geconfronteerd worden". De filosofie achter dataopslag bij het AppStats2-platform is vergelijkbaar: we stellen nauwelijks limieten aan de frequentie of de inhoud van berichten - hoe meer data we binnenkrijgen, hoe meer informatie we hier op een later tijdstip uit kunnen filteren.

Grofweg zijn er twee manieren om data op te slaan: geaggregeerd en onveranderd. De eerste methode past filters, sortering en andere berekeningen toe op de oorspronkelijke data, waarna het resultaat in de database opgeslagen wordt. De tweede methode slaat de data onveranderd op in de database, wat meer ruimte inneemt.

Er zijn een aantal argumenten aan te voeren om gegevens aan de client-kant (op de iPhone, iPad of ander mobiel device) te filteren en te aggregeren, waarna de bewerkte gegevens in de database opgeslagen worden. De belangrijkste hiervan zijn:

1. Ieder mobiel device gebruikt zijn eigen processorkracht voor het filteren en aggregeren, waardoor deze niet meer door de server besteed hoeft te worden.
2. Geaggregeerde informatie neemt minder ruimte in dan raw-data, waardoor de database sneller blijft en de server minder snel vol raakt.

Deze argumenten zijn in overweging genomen, maar met het oog op de volgende argumenten is besloten om voor de opslag van raw-data te gaan:

1. De variatie van de binnenkomende berichten is erg groot en veel berichten kunnen op verschillende manieren gefilterd en geaggregeerd worden. Bij voorbaat is niet bekend welke informatie klanten belangrijk vinden, en met raw-data hebben we ook over 10 jaar nog de mogelijkheid om de berichten te analyseren op patronen die we nu niet konden voorzien.
2. De opslag van veel berichten kost relatief weinig. 5 miljoen records nemen nog geen halve gigabyte in, en schijfruimte wordt ieder jaar goedkoper.
3. Dataopslag kan geschaald worden, in de eerste instantie door harde schijven bij te plaatsen, later door het aantal fysieke servers uit te breiden.
4. Hetzelfde geldt voor processorkracht: indien het nodig blijkt om zware berekeningen uit te voeren kan dit door een aparte machine verzorgd worden.

4.3.2. Databasemodel

De eerste keuze op het gebied van databases was het *database model*. Er zijn verschillende geaccepteerde modellen, waarvan de meestgebruikte zijn:

1. Platte database
2. Hiërarchische database
3. Netwerkdatabase
4. Relationele database

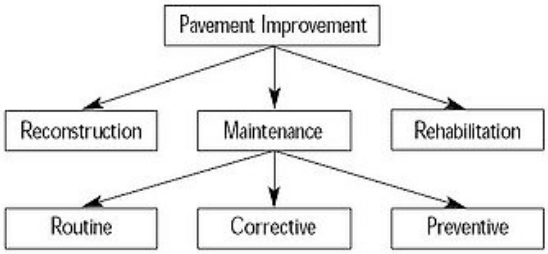
De platte database is een lijst van gegevens (records) waar geen verwijzingen tussen verschillende records bestaan. Deze manier van opslag is inefficiënt wanneer veel records dezelfde data bevatten; bij ieder record moet alle relevante informatie opgeslagen worden, omdat niet verwezen kan worden naar een andere tabel (waar veelvoorkomende informatie eenmalig is opgeslagen).

Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

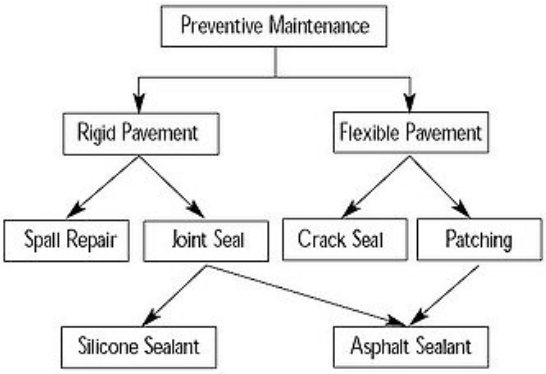
In een hiërarchische database worden gegevens opgeslagen in een boomstructuur. Bij deze structuur wordt een zogenaamde *parent-child* verhouding tussen gegevens aangelegd, waarbij iedere ouder meerdere kinderen kan hebben, maar het kind slechts één ouder heeft. Een hiërarchische database is daarom enkel van toepassing bij datastructuren waar gegevens altijd een bovenliggend niveau hebben.

Hierarchical Model



Het netwerkmodel is een variatie op het hiërarchische model, met als voornaamste verschil dat een 'kind' meerdere ouders kan hebben. Het voornaamste argument voor het toepassen van het netwerkmodel is dat het 'natuurlijkere' datastructuren toelaat, zoals in de afbeelding rechts geïllustreerd; in een hiërarchisch model had 'Asphalt Sealant' niet een kind kunnen zijn van zowel 'Joint Seal' als 'Patching', waardoor het twee keer in de database had moeten bestaan.

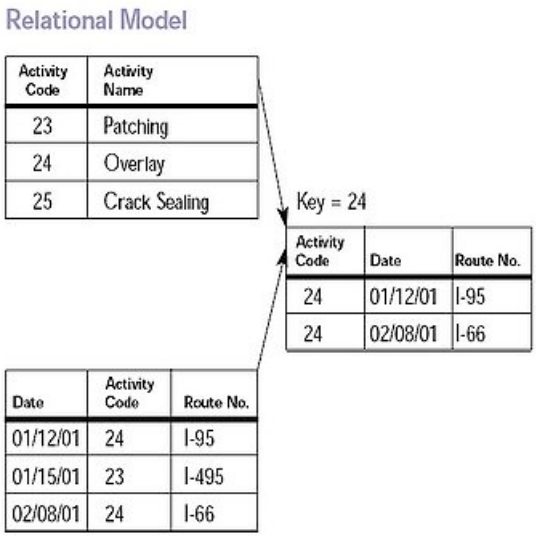
Network Model



Tabellen in het relationele databasemodel bieden verwijzingen naar andere tabellen door middel van *keys* - unieke waarden die aan een specifiek record gekoppeld zijn. Door hier gebruik van te maken kunnen tabellen beperkt worden tot een optimaal opgeslagen, beperkte set van gegevens.

Indien een andere tabel gegevens heeft die *relateren* aan de gegevens in de eerste tabel wordt enkel verwezen naar de key uit de eerste tabel, zoals in het voorbeeld rechts geïllustreerd is.

Hierdoor hoeven niet alle gegevens uit de eerste tabel opgeslagen te worden in de tweede tabel, wat resulteert in een snellere, kleinere database.



NoSQL is, naast de vier types hierboven, een databasemodel dat het vernoemen waard is . NoSQL is deze recent door een aantal grote partijen, waaronder Facebook, LastFM en eBay, toegepast om grote hoeveelheden data op te slaan en toch snel doorzoekbaar te blijven. In het geval van Facebook wordt bijvoorbeeld 50 Terabyte aan inbox-data in NoSQL opgeslagen, welke met hoge frequentie door gebruikers gelezen worden. NoSQL is ontworpen met het oog op het vermijden van relationele verbindingen in een database en het gedistribueerd opslaan van de gegevens over meerdere servers.

De uiteindelijke keuze is op het relationele databasemodel gevallen. De primaire reden hiertoe is de structuur van de data, waarin een groot aantal relaties tussen verschillende datatypes bestaan. Zie ook het databaseontwerp (inclusief relaties) in bijlage B. NoSQL was ook een goede optie, omdat het zeer schaalbaar is - dit is ook de reden is dat het door genoemde grote partijen gebruikt wordt - maar het biedt geen ACID-garanties (atomicity, consistency, isolation, durability), wat kort gezegd inhoudt dat niet vertrouwd kan worden op de volledigheid en correctheid van de gegevens in de database.

4.3.3. Database management system

Veel RDBMS leveren vergelijkbare specificaties en variëren slechts op detailniveau. Voor AppStats2 geldt dat geen gebruik gemaakt wordt van zeldzame datatypen of transacties. Ook wordt niet verwacht dat de database in de nabije toekomst groter dan 100GB zal groeien. Wel belangrijk is dat het systeem horizontaal schaalbaar moet zijn, wat betekent dat het systeem sneller en groter gemaakt kan worden door meer servers bij te schakelen. Een laatste overweging is het benodigde besturingssysteem, maar omdat de database beschikbaar komt op speciaal daarvoor ingerichte server, is ook de keuze voor het besturingssysteem vrij.

Na bovenstaande punten te overwegen is de keuze op MySQL gevallen. De voornaamste afwegingen hierin waren kosten en schaalbaarheid: MySQL zelf is gratis en kan op vrijwel ieder platform geïnstalleerd worden, waaronder Linux, dat ook gratis is. Het systeem ondersteunt bovendien horizontale schaalbaarheid in de vorm van DRBD (*Distributed Replicated Block Device*).

Enkele andere eigenschappen van MySQL, welke alle binnen acceptabele waarden vallen:

- Maximale grootte van database: oneindig
- Maximale grootte van een tabel: 256TB
- Maximale breedte van een kolom: 64KB
- Maximaal aantal kolommen per rij: 4096
- Maximale grootte van kolomnaam: 64 karakters

4.3.4. Webservice

De verschillende API's zijn aan te spreken via een webservice. Deze service wordt gehost op een server met Ubuntu, Apache en PHP. In combinatie met de eerder besproken keuze voor MySQL wordt deze groep van applicaties een LAMP-systeem genoemd (Linux, Apache, MySQL, PHP). Deze samenvoeging van softwarepakketten heeft grote bekendheid vergaard door de bijna naadloze onderlinge aansluiting.

Omdat het LAMP-systeem door een grote groep systeembeheerders gebruikt wordt, is bijna iedere denkbare toepassing en uitbreiding reeds beschikbaar. Door aan te sluiten op deze bestaande groep gebruikers kunnen de oplossingen voor eventuele problemen snel online gevonden worden. Bovendien is een LAMP-systeem erg schaalbaar, wat de keuze ook geschikt maakt voor gebruik in de toekomst.

4.3.5. Databaseontwerp

Om tot een logisch ontwerp van de relationele database te komen is geprobeerd zo veel mogelijk overeenkomst met de werkelijkheid te behouden. Dit betekent dat tabellen vaak overeenkomen met fysieke elementen of specifieke onderdelen van een applicatie of service. Vanuit deze optiek zijn de volgende tabellen ontstaan:

- Application - bevat algemene informatie over de applicaties die met het systeem communiceren.
- AppGroup - bevat informatie over groepen van gebruikers die rechten hebben tot bepaalde applicaties. Zo kan een groep 'sport' aangemaakt worden, waarbij gebruikers recht hebben om alle sportapps te bekijken.
- ClientCompany - bevat informatie over een bedrijf dat rechten heeft tot toegang tot het systeem.
- DeviceType - bevat informatie over verschillende toestellen en systemen die berichten naar het platform verzenden.
- Host - bevat informatie over de systemen die services draaien.
- HostGroup - bevat informatie over groepen van gebruikers die rechten tot bepaalde hosts hebben. In een hostgroup kunnen bijvoorbeeld de servers in een specifiek datacenter of rack opgenomen worden.
- Persona - bevat informatie over de gebruikers van de applicaties die met het platform communiceren.
- Service - bevat informatie over de services die met het systeem communiceren.
- ServiceGroup - bevat informatie over groepen van gebruikers die rechten tot bepaalde services hebben. Zo kan een databasebeheerder toegevoegd worden aan een groep van MySQL-servers.
- ServiceVersion - bevat informatie over de verschillende versies van de services.
- Session - bevat informatie over de gebruikerssessies.
- Subject - bevat informatie over de onderwerpen van de binnenkomende berichten.

- User - bevat gegevens over de gebruikers die op het platform kunnen inloggen.

Naast de bovenstaande tabellen bevat de database ook een aantal tabellen die dienen om de gegevens aan elkaar te koppelen:

- AppsInGroup - bevat een lijst van appID's die bij een bepaalde AppGroup horen.
- AppsForCompany - bevat een lijst van appID's die bij een bepaalde ClientCompany horen.
- HostAndServicesForUser - koppelt hosts en services aan een bepaalde gebruiker.
- HostsForCompany - bevat een lijst van hostID's die bij een bepaalde ClientCompany horen.
- MessageLog - bevat de binnenkomende berichten van applicaties.
- ServiceLog - bevat de binnenkomende berichten van services.
- UsersForCompany - bevat een lijst van userID's die bij een bepaalde ClientCompany horen.

Bij de inrichting van de tabellen is geprobeerd zoveel mogelijk gebruik te maken van referenties. Hiermee hoeven gegevens die regelmatig voorkomen slechts één keer opgeslagen te worden, waarna verwezen kan worden naar de unieke identifier van de gegevens. Zo bestaat de messageLog-tabel naar verwijzingen naar een persona, appVersion en subject, waardoor de gegevens van een bepaalde persoon, applicatieversie of onderwerp niet in de MessageLog opgenomen hoeven te worden.

Alle hierboven genoemde informatie is overzichtelijk opgenomen in het database-ontwerp in bijlage B.

5. Uitvoering

5.1. Inleiding

In dit hoofdstuk wordt beschreven hoe het functioneel ontwerp in de praktijk toegepast is. Zoals in de paragraaf 'op te leveren producten' beschreven is, is de ontwikkeling van de modules Input en Reporting geen onderdeel van het afstudeerproject, waardoor hier niet ingegaan kan worden op de praktische implementatie van deze onderdelen. Van de overige modules (receiving, core systems, requesting, administration) wordt stapsgewijs beschreven hoe ze functioneren.

5.2. Receiving

De Receive-API bestaat uit één PHP-script dat de gegevens van de Input-module ontvangt. De mee te leveren parameters en de te verwachten respons staat beschreven in Bijlage C: Receive API.

5.2.1. Errors

Het script is opgebouwd uit een aantal functies die in serie doorlopen worden. Bij vrijwel iedere functie wordt een error-argument meegegeven, zoals in onderstaand voorbeeld:

```
function processSubject($packet, &$error)
```

Dit error-argument, welk als referentie meegegeven wordt, zal het getal 0 bevatten tenzij er ergens in het script een fout optreedt. Door telkens na het uitvoeren van een functie te controleren of de errorcode nog op 0 staat of een foutcode bevat kan bepaald worden of het script verder moet gaan of een foutcode moet tonen.

In een aparte, statische klasse (_Globals.php) worden alle mogelijke foutmeldingen en hun beschrijving vastgelegd. Een error kan gezet worden door een regel code als:

```
$error = Globals::$applicationBlocked;
```

Later kan deze errorcode gebruikt worden om te analyseren waar een fout zich in een script bevindt, of om een XML met foutcode uit te printen, bijvoorbeeld met de volgende code:

```
error_log(Globals::descriptionForErrorCode($errorCode));
```

5.2.2. Afhankelijkheden

Omdat de database relationeel is opgezet dienen gegevens in een bepaalde volgorde in de database gezet te worden. Een verwijzing naar een tabel kan immers pas opgenomen worden op het moment dat de gegevens in de betreffende tabel beschikbaar zijn.

Om de compleetheid van de binnenkomende data te garanderen worden alle gegevens eerst in een aparte PHP-klasse (_DataPacket.php) opgeslagen. Dit datapakket 'verwacht' dat de gegevens die in hem opgeslagen worden aan bepaalde eisen volden (zoals in Bijlage C gedefinieerd) en zal een foutcode genereren als de data van de eisen afwijkt.

Als het datapakket succesvol gevuld is, wordt dit pakket aan de functies aangeboden die de gegevens in de database opslaan. Door deze functies in de juiste volgorde te doorlopen wordt gegarandeerd dat aan de afhankelijkheden van data voldaan kan worden. Indien er in één van de functies een fout optreedt wordt de rest van het script afgebroken en wordt een foutmelding teruggegeven.

Berichten van applicaties en services worden gescheiden in de database opgeslagen, omdat ze een licht verschillende structuur hebben (zie paragraaf 4.3.5). Voor applicaties geldt de volgende procedure:

1. Verwerk het apparaattype (bijvoorbeeld iPhone 4). Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor het apparaattype (deviceTypeID). Deze verwerking vindt plaats in de functie processDeviceType().
2. Verwerk de persona. Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor de telefoon / het systeem dat het bericht verzond (personaID). Deze verwerking vindt plaats in de functie processPersona().
3. Verwerk de applicatie. Indien het niet bestaat, wordt een error teruggegeven. Applicaties moeten voor ze berichten kunnen registreren in de database aangemaakt zijn. Het resultaat is een uniek nummer voor de applicatie (appID). Deze verwerking vindt plaats in de functie processApplication().
4. Verwerk de applicatieversie. Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor de applicatieversie (appVersionID). Deze verwerking vindt plaats in de functie processAppVersion().
5. Verwerk het onderwerp. Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor het onderwerp (subjectID). Deze verwerking vindt plaats in de functie processSubject().
6. Verwerk de totale bericht, gebruikmakend van bovenstaande vijf ID's en de overige gegevens in het datapakket. Het resultaat is een uniek nummer voor het bericht (messageLogID). Deze verwerking vindt plaats in de functie processMessage().

Services doorlopen een vergelijkbaar traject, maar met kleine verschillen in de opgeslagen data:

1. Verwerk de host (een specifieke server). Indien de host niet bestaat, wordt een error teruggegeven. Hosts moeten voor ze berichten kunnen registreren in de database aangemaakt zijn. Het resultaat is een uniek nummer voor de host (hostID). Deze verwerking vindt plaats in de functie processHost().
2. Verwerk de service. Indien het niet bestaat, wordt een error teruggegeven. Services moeten voor ze berichten kunnen registreren in de database aangemaakt zijn. Het resultaat is een uniek nummer voor de service (serviceID). Deze verwerking vindt plaats in de functie processServiceID(). Hier is gekozen voor een licht afwijkende naam omdat de functie processService() al bestond.
3. Verwerk de serviceversie. Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor de serviceversie (serviceVersionID).
4. Verwerk het onderwerp. Indien het niet bestaat, wordt het aan de database toegevoegd. Het resultaat is een uniek nummer voor het onderwerp (subjectID). Deze verwerking vindt plaats in de functie processSubject().
5. Verwerk de totale bericht, gebruikmakend van bovenstaande vier ID's en de overige gegevens in het datapakket. Het resultaat is een uniek nummer voor het bericht (serviceLogID). Deze verwerking vindt plaats in de functie processServiceMessage().

5.2.3. Response bij succes

Als alle bovenstaande functies succesvol doorlopen zijn wordt een XML gegenereerd waarin de unieke ID voor het afgelopen bericht opgenomen wordt. Dit ID kan later eventueel gebruikt worden voor het opzoeken van het bericht in de database.

5.2.4. Debug

Bij het invoeren van gegevens kan optioneel de parameter “debug”=”true” meegegeven worden. Als dit het geval is wordt na iedere succesvolle stap, bijvoorbeeld het toevoegen van de deviceTypeID aan de database, een bericht opgenomen in de serverlogs.

In debugmode wordt het volledige proces van het toevoegen van een bericht aan de database doorlopen, maar na succes wordt het bericht weer verwijderd uit de database. Hierdoor kunnen eventuele fouten in het proces opgespoord worden, zonder dat er corrupte data in de database opgenomen wordt.

5.3. Requesting

De request-api bestaat uit 19 scripts waarmee alle benodigde details van applicaties en services opgevraagd kunnen worden. Deze scripts zijn losse PHP-files in een voor de Request-API gereserveerde directory op de AppStats2-webserver. In bijlage E wordt gedetailleerd beschreven welke parameters de 19 script vereisen, en welke response de aanvrager kan verwachten.

Alle scripts accepteren variabelen via POST en GET, zolang alle variabelen maar via hetzelfde systeem binnenkomen.

In bijlage D wordt geïllustreerd welke afhankelijkheden de scripts van elkaar hebben. Zo is uit de bijlage op te maken dat de appVersionSubjects bijvoorbeeld niet opgevraagd kunnen worden zonder eerst bekendheid van de beschikbare appVersions te hebben.

Onderstaand wordt de werking van de 19 scripts beschreven, in de volgorde van de Request Flow in bijlage D. Voor de benodigde argumenten en te verwachten response, zie bijlage E: Request API.

De vijf laatst beschreven scripts (getPlatformStats, getApplicationSubjectDetails, getAppVersionSubjectDetails, getServiceSubjectDetails en getServiceVersionSubjectDetails) zijn de scripts waar het grootste deel van de data opgevraagd kan worden. De overige scripts dienen voornamelijk om te bepalen welke argumenten (applicaties, services, hosts, subjects) gebruikt kunnen worden bij het aanroepen van de laatste scripts.

getSession

Het getSession-script accepteert een username en een wachtwoord en controleert of deze correct zijn. Als dit het geval is, wordt gekeken of er al een sessie bestaat voor de inloggende gebruiker en wordt eventueel het bestaande sessieID teruggegeven. Ook wordt de timestamp bijgewerkt, zodat de sessie vanaf het moment van opnieuw inloggen weer een uur geldig is.

Als de sessie nog niet bestond wordt een nieuwe sessie aangemaakt met het huidige IP en platform.

De sessieID en andere algemene informatie over de gebruiker zullen in een XML teruggegeven worden.

getApplications
Met getApplications wordt een lijst van beschikbare appID's voor de ingelogde gebruiker opgevraagd. Hiertoe wordt eerst gekeken of het meegezonden sessieID geldig is. Daarna wordt een lijst van applicaties opgevraagd, welke via XML opgeleverd wordt.

Bij het aanvragen van dit script wordt de timestamp van de sessie bijgewerkt, waardoor de gebruiker weer een uur lang ingelogd blijft.

Dit script dient gebruikt te worden in situaties waar enkel applicaties relevant zijn en er bijvoorbeeld geen gegevens over services benodigd zijn. Op het moment dat informatie over zowel applicaties, services als hosts opgevraagd wordt, kan beter het getPermissions-script gebruikt worden.

getServices
Het getServices-script werkt vrijwel exact hetzelfde als getApplications, maar vanzelfsprekend worden geen applicaties, maar een lijst met Services teruggegeven. Ook Bij het aanvragen van dit script wordt de timestamp van de sessie bijgewerkt.

Dit script dient gebruikt te worden in situaties waar enkel services relevant zijn. Op het moment dat informatie over zowel applicaties, services als hosts opgevraagd wordt, kan beter het getPermissions-script gebruikt worden.

getHosts
Het getHosts-script werkt ook als getApplications en getServices: op basis van het sessionID wordt een lijst van hosts opgevraagd en in een XML teruggegeven.

Doordat een gebruiker over het algemeen rechten zal hebben om een beperkt aantal services per host te bekijken - en zelden precies dezelfde services over alle hosts - heeft het opvragen van een lijst van hosts weinig waarde. Wel kunnen de hostID's die uit het getHosts-script voortkomen gebruikt worden in het getServicesForHost-script.

Het getHosts-script dient gebruikt te worden in situaties waar enkel hosts relevant zijn en er bijvoorbeeld geen gegevens over services benodigd zijn. Op het moment dat informatie over zowel applicaties, services als hosts opgevraagd wordt, kan beter het getPermissions-script gebruikt worden.

getServicesForHost
Het getServicesForHost-script accepteert een hostID en toont de services op deze host waarvan de gebruiker recht heeft om ze te bekijken. Het script zal de services in een XML teruggeven.

getPermissions
Het getPermissions-script is in feite een samenvoeging van de vorige vier scripts (getApplications, getServices, getHosts, getServicesForHosts). Het neemt het meegeleverde sessieID, bevestigt dat de gekoppelde sessie geldig is en levert vervolgens een lijst van applicaties en host-service-combinaties in XML.

De toegevoegde waarde van dit script is niet alleen dat alle gegevens met één aanvraag binnengehaald kunnen worden, maar ook de opgenomen extra informatie, waarmee alle applicaties, services en hosts makkelijk in een lijst getoond kunnen worden.

De extra informatie omvat het volgende:

- Voor applicaties:
    - *usersToday*, waarin het aantal gebruikers sinds 00:00 opgenomen zijn.
    - sessionsToday, waarin het aantal sessies sinds 00:00 opgenomen zijn.
    - weightParameter, wat een suggestie voor het sorteren van applicaties is, gebaseerd op het aantal gebruikers.
    - Totals: een optelling van alle beschikbare applicaties voor een gebruiker, waarmee een overkoepeld beeld van het gebruik van de applicaties gegenereerd kan worden.
- Voor services:
    - weightParameter, wat ook hier een suggestie voor de sortering is.
    - averageToday, waarin de gemiddelde waarde van de service sinds 00:00 opgenomen is.

getApplicationSummary

Met een appID, welke via getApplications of getPermissions verkregen kan zijn, kan met het getApplicationSummary-script een verzameling gegevens over een applicatie opgevraagd worden. Bij het aanroepen van het script dient tevens een periodID meegeleverd te worden, welke aanduidt over welke tijdspanne de gegevens opgevraagd worden. Deze tijdspanne kan bijvoorbeeld de laatste 24 uur, de laatste week of het laatste halfjaar zijn.

Het script levert de volgende gegevens:

- Het minimale, gemiddelde en maximale aantal gebruikers en sessies over de gekozen periode.
- Gegevens waarmee een grafiek getekend kan worden
- Een lijst van de vijftien meest recente binnenkomende berichten.

De XML die door het getApplicationSummary-script geleverd worden, kunnen bijvoorbeeld gebruikt worden om gegevens te tonen zoals in figuur 21.

Figuur 21: Tonen van application summary

getApplicationVersions

Een applicatie bestaat uit minimaal één versie (normaliter versie 1.0), maar een applicatie die regelmatig bijgewerkt wordt kan vele versies hebben. Deze versies verzenden allemaal hun eigen berichten en nemen een eigen percentage van het totale gebruik in. Iedere versie van een applicatie krijgt een eigen ID, waarmee statistieken specifiek voor die versie opgevraagd kunnen worden.

Bij het aanroepen van getApplicationVersions hoeven enkel de sessionID en appID meegegeven te worden. Als alle gegevens correct zijn zal een lijst met versieID's voor de geselecteerde app teruggegeven worden, inclusief een string waarin het versienummer uitgeschreven staat.

getApplicationSubjects

Een applicatie verzendt een verzameling aan berichten, welke geïdentificeerd worden aan de hand van hun *subject*. Het subject wordt aangeduid met een unieke subjectID. Met het getApplicationSubjects-script wordt een lijst van subjectID's voor een specifieke applicatie aangeleverd.

Het script heeft, net als getApplicationVersions, enkel een geldig sessionID en appID nodig. Als deze twee argumenten goed zijn, zal in XML een lijst van subjectID's en hun beschrijving teruggegeven worden.

getAppVersionSubjects

Het is ook mogelijk om enkel de subjects van een specifieke versie van een applicatie op te vragen. Hiertoe dient het getAppVersionSubjects-script, dat hetzelfde functioneert als getApplicationSubjects, maar een appVersionID in plaats van een appID verwacht. Dit appVersionID kan verkregen worden met het getApplicationVersions-script.

getServiceSummary

Het getServiceSummary-script levert een samenvatting van het gebruik van een service op een specifieke host. Hiertoe worden een periodID, serviceID, hostID en sessionID aangeleverd.

Bij het aanroepen van het script wordt eerst bevestigd dat de gebruiker recht heeft om de gevraagde service-host-combinatie te bekijken. Als de rechten in orde zijn worden de gegevens op basis van de eerste drie argumenten uit de database gehaald. Met deze gegevens worden de punten voor een average-, minimum- en maximum-grafiek bepaald en via XML aan de gebruiker teruggegeven.

Met deze XML kunnen grafieken als in figuur 22 getekend worden.

Figuur 22: Tonen van service summary

getServiceVersions

Services hebben, gelijk aan applicaties, één of meerdere versies. Door het opgeven van een sessionID en een serviceID geeft het getServiceVersions-script een lijst van gebruikte versies terug, inclusief een uitgeschreven versienummer.

getServiceSubjects

Een service bevat één of meerdere subjects. Zo kan de *Apache Logger Service* subjects bevatten als Requests per Minute, Requests per Day, of 404's per Day. Een lijst van deze subjects kan opgevraagd worden met het het getServiceSubjects-script.

Het script verwacht enkel een geldig sessionID en serviceID. Als deze in orde zijn zal de lijst met subjectID's en de bijbehorende beschrijvingen in XML teruggegeven worden.

getServiceVersionSubjects
Van een bepaalde versie van een service kunnen ook de berichten opgevraagd worden. Dit functioneert vrijwel hetzelfde als getServiceSubjects, alleen wordt in plaats van een serviceID een serviceVersionID meegeleverd.

getPlatformStats
Het getPlatformStats-script levert gegevens over de verdeling van een applicatie, verspreid over meerdere typen telefoons / apparaten. Deze gegevens beslaan een specifieke periode, aangeduid door middel van de het periodID. Het script accepteert naast het sessionID en periodID een appID òf een appVersionID.

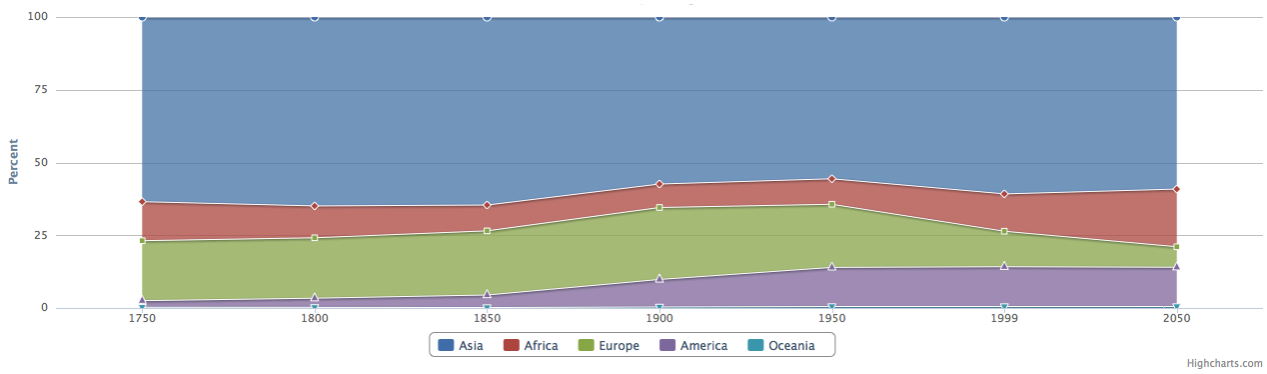
Indien een appVersionID meegeleverd wordt, zal de volgende data geleverd worden:

- Verdeling van hardwareplatformen bij het gebruik van de applicatie, bijvoorbeeld de verdeling tussen iPhone 4, iPhone 3GS en iPod Touch.
- Verdeling van operating systems bij het gebruik van de applicatie.

Als in plaats van een appVersionID een appID meegeleverd wordt, zal ook de volgende informatie geleverd worden:

- Verdeling van versies van de applicatie, bijvoorbeeld dat 90% van de gebruikers versie 1.0 heeft en versie 1.1 slechts door 10% gebruikt wordt.

Alle bovenstaande informatie wordt in percentages geleverd, waardoor de informatie in een zogenaamde *stacked area* getoond kan worden. Zie ook het voorbeeld in figuur 23.



Figuur 23: Stacked area grafiek

De informatie uit het getPlatformStats-script geeft met name een beeld over het marktgebied van een applicatie. Dit kan gebruikt worden om te bepalen waar de doelgroep zich bevindt, of waar nog open kansen liggen.

getApplicationSubjectDetails
Nadat door middel van de getApplications- en getSubject-scripts een subject geselecteerd is, kan met het getApplicationSubjectDetails-script de relevante informatie verkregen worden. Deze gegevens zullen geleverd worden in de vorm van één of meerdere grafieken.

Bij het aanroepen van getApplicationSubjectDetails worden een sessieID, een appID, een subjectID en periodID meegegeven. De laatste bepaalt over welke tijdspanne de grafiek geleverd wordt, bijvoorbeeld de afgelopen dag, week of maand.

Ieder subject heeft zijn eigen typische eigenschappen, welke bepalen hoe de gegevens van het subject getoond moeten worden. Dit kan bijvoorbeeld in de vorm van een lijngrafiek,

taartdiagram of top 10 zijn. Een aanduiding van de manier van tonen wordt in de XML opgenomen, waardoor de weergevende applicatie of website hierop af kan stemmen.

getAppVersionSubjectDetails
Het getAppVersionSubjectDetails-script functioneert op vrijwel dezelfde manier als getApplicationSubjectDetails, maar waar laatstgenoemd script data levert over alle versies van een applicatie, levert dit script enkel gegevens over een specifieke versie.

Om aan te duiden van welke applicatieversie en subject de gegevens opgevraagd worden, dienen het appVersionID en het subjectID meegegeven te worden. Net als bij getApplicationSubjectDetails kunnen gegevens relateren aan verschillende tijdspannen. Om aan te geven over welke periode de gegevens opgevraagd worden dient ook een periodID meegegeven te worden.

getHostServiceSubjectDetails
Met getHostServiceSubjectDetails kunnen de gegevens van een een specifiek bericht binnen een service, op een specifieke host, over een bepaalde periode opgevraagd worden. Hiertoe dienen de sessionID, hostID, serviceID, subjectID en periodID aangeleverd te worden.

Ter illustratie:

- De service is **Apache Logger (serviceID: 4)**
- Het bericht is **Requests per minute (subjectID: 23)**
- De host is **lucas.codingdutchmen.com (hostID: 5)**
- De periode is **last month (periodID: 4)**

Het script zal op basis van de bovenstaande variabelen gegevens teruggeven waarmee bijvoorbeeld een lijngrafiek getekend kan worden.

getHostServiceVersionSubjectDetails
Het getHostServiceVersionSubjectDetails-script levert vrijwel dezelfde als data als het hierboven beschreven getHostServiceSubjectDetails-script, maar waar het vorige script de gegevens over alle versies van een applicatie zal aanleveren, worden deze met dit script beperkt tot één specifieke versie.

De argumenten bij het aanroepen van het script zijn ook vrijwel hetzelfde als bij het vorige script; er dienen een sessionID, hostID, subjectID en periodID opgegeven te worden. In plaats van een serviceID wordt echter een serviceVersionID verwacht.

De response van het script zal dezelfde structuur als getHostServiceSubjectDetails aanhouden.

5.3.1. Sessies
Sessies zijn een verzameling gegevens, gekoppeld aan een getal. De gegevens omvatten het ID van de gebruiker, het adres en platform van waar hij met platform communiceert en een timestamp. Door het sessieID te gebruiken bij het opvragen van gegevens hoeft een aanvraag niet iedere keer de gebruikersnaam en wachtwoord van de gebruiker te bevatten. Omdat de sessieID gekoppeld is aan het IP-adres en het platform van de gebruiker kan een ‘man in the middle’ geen gegevens opvragen met hetzelfde sessieID. Dit platform wordt opgeslagen als MD5-waarde van de *user agent* om geen onnodige ruimte in de database op te nemen. Zo wordt de user agent “Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0b7) Gecko/20100101 Firefox/4.0b7” opgeslagen als “41c30f7d3d25d1ec1ff5b84c835fa928”. Als

een gebruiker inlogt vanaf een ander platform zal de MD5-waarde van zijn user agent veranderen en is de oorspronkelijke sessie niet meer geldig.

5.3.2. Klassen

De in dit hoofdstuk beschreven scripts zijn over het algemeen niet meer dan een laag die de aanroep en het leveren van de XML verzorgt. Achter de scripts liggen echter PHP-klassen die de gegevens uit de databases halen en omvormen tot bruikbare arrays. De namen van deze klassen komen over het algemeen overeen met de tabellen in de database waar ze mee communiceren. Er zijn echter een paar uitzonderingen doorgevoerd, waar een verzameling functionaliteit nodig was die niet in een andere klasse past.

Alle klassen en hun verantwoordelijkheden zijn opgenomen in bijlage F.

5.3.3. Perioden

Bij een groot aantal scripts wordt een periodID als argument meegegeven. Zoals bij de beschrijving van de scripts werd vermeld wordt dit ID gebruikt om aan te duiden aan welke tijdspanne de opgevraagde gegevens relateren. Iedere tijdspanne wordt door drie variabelen vastgesteld:

1. Het aantal dagen in de spanne.
2. De tijdeenheid waar de periode in opgedeeld wordt.
3. Het aantal subperioden in de tijdspanne, wat in feite voortkomt uit variabelen 1 en 2.

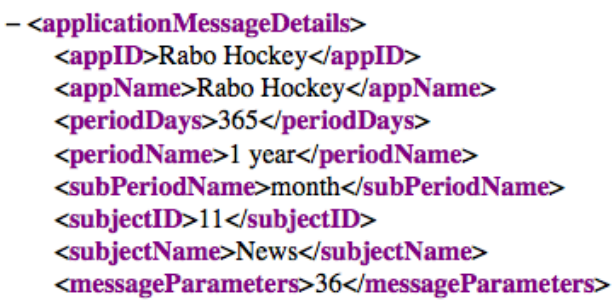
Met deze variabelen zijn de volgende perioden vastgelegd:

PeriodID	Dagen	Tijdeenheid	Aantal subperioden
1	1	Uur	24
2	7	Dag	7
3	14	Dag	14
4	31	Dag	31
5	90	Week	12
6	180	Week	26
7	365	Maand	12

Tabel 4: Perioden en hun variabelen

Uit tabel 4 valt op te maken dat het opvragen van gegevens met periodID 3 zal resulteren in gegevens over een periode van 14 dagen, opgedeeld in 14 delen, waarbij ieder deel data bevat die relateert aan één dag. Bij het gebruik van periodID 6 worden gegevens over een periode van 180 dagen teruggegeven, opgedeeld in 26 delen, waarbij ieder deel data bevat die relateert aan één week.

Bij het opvragen van gegevens met een periodID worden de variabelen uit tabel 4 altijd in de XML teruggegeven, zoals hiernaast in figuur 24 afgebeeld. Als vervolgens met de gegevens uit de XML een grafiek getekend wordt, kan met de data in de header een begeleidende opgesteld worden. Zo zou met de gegevens uit figuur 24 de tekst “Rabo Hockey - News. Values represent data per month, over a period of 1 year.” opgesteld kunnen worden.



Figuur 24: Header getApplicationSubjectDetails

5.4. Administration

Bij het schrijven van deze scriptie is de admin-api nog niet ontwikkeld. De werking van het systeem staat echter wel gedefinieerd in het functioneel- en technisch ontwerp.

Bij de ontwikkeling zal een lijst van benodigde bewerkingen (bijvoorbeeld het toevoegen van een gebruiker, het wijzigen van de displayname van een applicatie) opgesteld worden. Deze bewerkingen worden vervolgens opgenomen in scripts, die net als bij de receive- en request-api’s voornamelijk dienst zullen doen als front voor de achterliggende klassen.

De meeste functionaliteit van de nieuwe scripts zal opgenomen kunnen worden in de huidige, bestaande klassen. Zo kunnen bewerkingen op gebruikers opgenomen worden in de UserUtils-klasse en kunnen wijzigingen op applicaties opgenomen worden in de ApplicationUtils-klasse.

6. Oplevering, conclusies en aanbevelingen

6.1. Periode na de scriptie

Na het inleveren van deze scriptie is er nog een kleine maand beschikbaar voor het opleveren van het afstudeer project. Deze paragraaf beschrijft de werkzaamheden die in deze laatste maand verricht zullen worden.

6.1.1. Oplevering laatste onderdelen

In paragraaf 2.3 werden de volgende punten op te leveren producten genoemd:

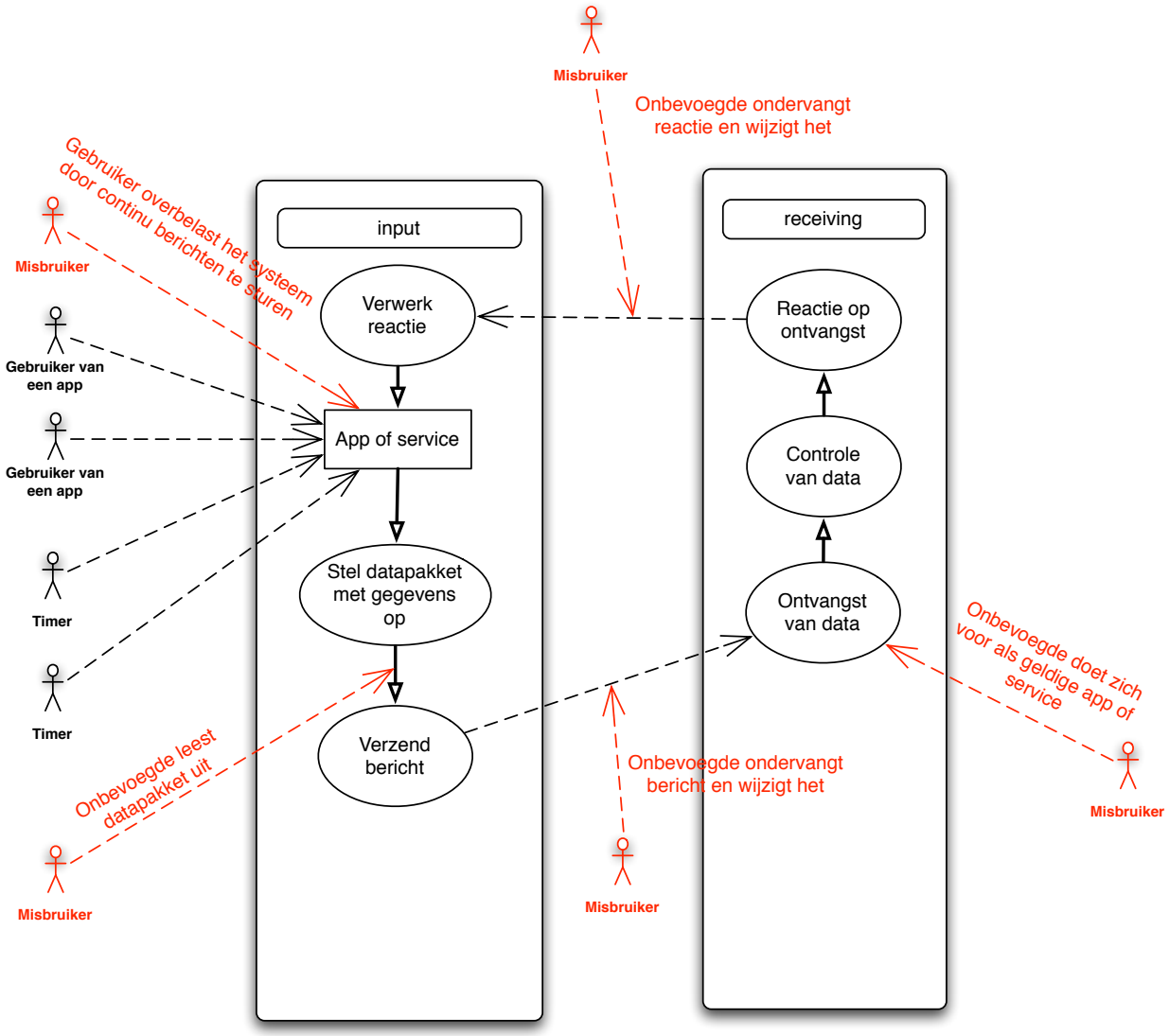
1. Een database, geschikt om alle te voorziene meldingen en berichten van verschillende bronnen op te slaan.
2. Een module waar data naartoe verzonden kan worden, receiving genoemd.
3. Een module die op aanvraag van gebruikers van het AppStats2-platform gegevens uit de database kan aggregeren, filteren en sorteren.
4. Een module, geschikt om wijzigingen in de databases mee door te voeren.
5. Een hardwareplatform om bovenstaande databases en modules te hosten.
6. Documentatie van de API's, zodat deze door externe ontwikkelaars gebruikt kunnen worden.

Van de bovenstaande punten is enkel de Administration-module nog niet gereed. De periode van 14 december tot 12 januari zal gebruikt worden voor het ontwikkelen van deze module en de uitrol van alle andere modules.

6.1.2. Security-onderzoek

Indien er na de uitrol nog tijd beschikbaar is, zal deze besteed worden aan een security onderzoek. Dit onderzoek gaat uit van de use cases zoals beschreven in het functioneel- en technisch ontwerp. Bij alle lijnen en objecten in deze diagrammen zal worden bekeken of er mogelijkheden tot inbraak of verlies van data mogelijk zijn.

Bij de elementen van use cases waar misbruik kan optreden wordt een aantekening gemaakt, wat lijdt tot een zogenaamde anti-use case. Een voorbeeld hiervan wordt getoond in figuur 25.



Figuur 25: Anti-use case AppStats2

6.2. Privacy

Het AppStats2-platform registreert gegevens van gebruikers, zonder dat zij hier kennis van hebben of goedkeuring voor gegeven hebben. Dit is in potentie een gevoelig punt voor privacy-kwesties, maar wij voelen ons gesterkt door drie punten:

1. Namen of andere te traceren gegevens van gebruikers worden niet geregistreerd. De waarden die wel aan een gebruiker gekoppeld worden, zijn zijn IP-adres en unieke identifier voor het toestel of apparaat dat hij gebruikt. Aangezien dit niets zegt over de persoon die het toestel vasthoudt (het zouden zelfs meerdere personen kunnen zijn) kan de data als anoniem beschouwd worden.
2. De gegevens die aan de gebruikers van het AppStats2-platform getoond worden zijn geaggregeerd en bevatten geen persoonlijke informatie. Zou kan van een applicatie vermeld worden dat hij in een bepaald uur 200 gebruikers had, en dat 30 hiervan de app nog niet eerder opgestart hadden. Er zal echter geen optie opgenomen worden om een specifieke gebruiker te traceren.

3. Er zijn andere partijen op de markt die vergelijkbare software aanbieden. Flurry Analytics is een van de grotere partijen in dit segment. Zij leveren het volgende statement: “Flurry may obtain information as a result of data being sent to our servers from our software "agent" that may be embedded in an end user's mobile application. We aggregate that data, which tracks information such as use sessions, an end user's handset, or an end user's country. Once the data is aggregated, we make it available for analysis by the developer of the application.”. Flurry heeft in de drie jaar dat ze bestaan geen publieke juridische problemen gehad, terwijl ze op veel grotere schaal dan wij functioneren.

6.3. Uitbreidingen en aanbevelingen

6.3.1. IPv6

Het nieuwe ontwikkelde systeem leunt sterk op IPv4. Zo worden de IP-adressen van gebruikers van het Input-segment als IPv4 verzonden en opgeslagen, en worden ook bij het identificeren van sessies IPv4 adressen gebruikt. In de nabije toekomst zullen echter steeds meer systemen, waaronder handsets en servers, zowel het 32-bits lange IPv4 als het 128-bit lange IPv6 gebruiken. Naar alle verwachting zullen dezelfde systemen op langere termijn helemaal geen IPv4-adres meer gaan gebruiken. Aangezien de opslag van een IPv4-adres slechts 32 bits in de database kost, moeten database en scripts op gegeven moment aangepast worden om de nieuw adressen te kunnen accommoderen.

6.3.2. Optimalisatie

Een groot deel van het AppStats2-platform is ontworpen met het oog op de optimale opslag en overdracht van gegevens. Hieronder vallen bijvoorbeeld gecomprimeerde kolommen in de database, het gebruiken van *unsigned* integers waar mogelijk en het gebruiken van veel indexen in de databases. Er is echter nog op verschillende gebieden winst te maken, bijvoorbeeld in het verzenden van gegevens tussen de input- en receiving-modules. In die communicatie worden gegevens op dit moment als *plain text* verzonden. Van de gegevens is echter bij voorbaat bekend hoe lang ze maximaal kunnen zijn, waardoor ze opgenomen kunnen worden in een byte-array van vaste lengte. Deze array kan vervolgens gecomprimeerd worden, waardoor een veel kleiner - en dus sneller - datapakket verzonden kan worden.

6.3.3. Servers

Bij het ontwerpen van het huidige systeem is rekening gehouden met *horizontal scaling*. Dit houdt in dat de belasting van het systeem verspreid kan worden over meerdere database- en webservers, al naar gelang de behoefte. Op het moment van schrijven is één webserver en één databaseserver voldoende om de toestroom aan berichten af te kunnen handelen, maar als het systeem door meer ontwikkelaars gebruikt gaat worden, kan dit snel wijzigen. Daarom moet na oplevering van het AppStats2-platform een implementatieplan voor de uitbreiding van servers gemaakt worden.

Naast horizontal scaling om een hogere capaciteit te leveren, dient ook rekening gehouden te worden met redundantie en *high availability*, wat inhoudt dat een of meerdere servers waar het platform op gehost wordt moeten kunnen uitvallen, zonder dat het systeem hier merkbare hinder van ondervindt. Dit brengt echter een vrij grote investering met zich mee, zonder

directe positieve gevolgen voor het functioneren, waardoor een afweging voor het moment van implementatie lastig is.

6.3.4. Uitbreiden van Request API

De huidige API in de requesting-module bevat scripts waar de belangrijkste gegevens in de database opgevraagd kunnen worden. Er zijn echter altijd meer gegevens beschikbaar, welke voor sommige klanten / gebruikers interessant kunnen zijn. Er zal eerst een lijst opgesteld moeten worden van mogelijke uitbreidingen, waarna een keuze gemaakt kan worden welke van deze mogelijkheden geïmplementeerd zullen worden.

7. Bronvermelding

- Alistair Cockburn, *Writing Effective Use Cases*. ISBN 978-0-201-70225-5
- Ken Coar & Rich Bowen, *Apache Cookbook*. ISBN 978-0-596-52994-9
- Andrew Cumming & Gordon Russell, *SQL Hacks*. ISBN 978-0-596-52799-0
- C.J. Date, *The Relational Database Dictionary*. ISBN 978-1-4302-1041-2
- Charles Bell, Mats Kindahl & Lars Thalmann, *MySQL High Availability*. ISBN 978-0-596-80730-6
- Jon Stephens & Chad Russell, *Beginning MySQL Database Design And Optimization*. ISBN 978-1-590-59332-5
- Toby J. Theorey, Stephen Buton & Lowell Fryman, *Database Design*. ISBN 978-0-123-74630-6

8. Begrippenlijst

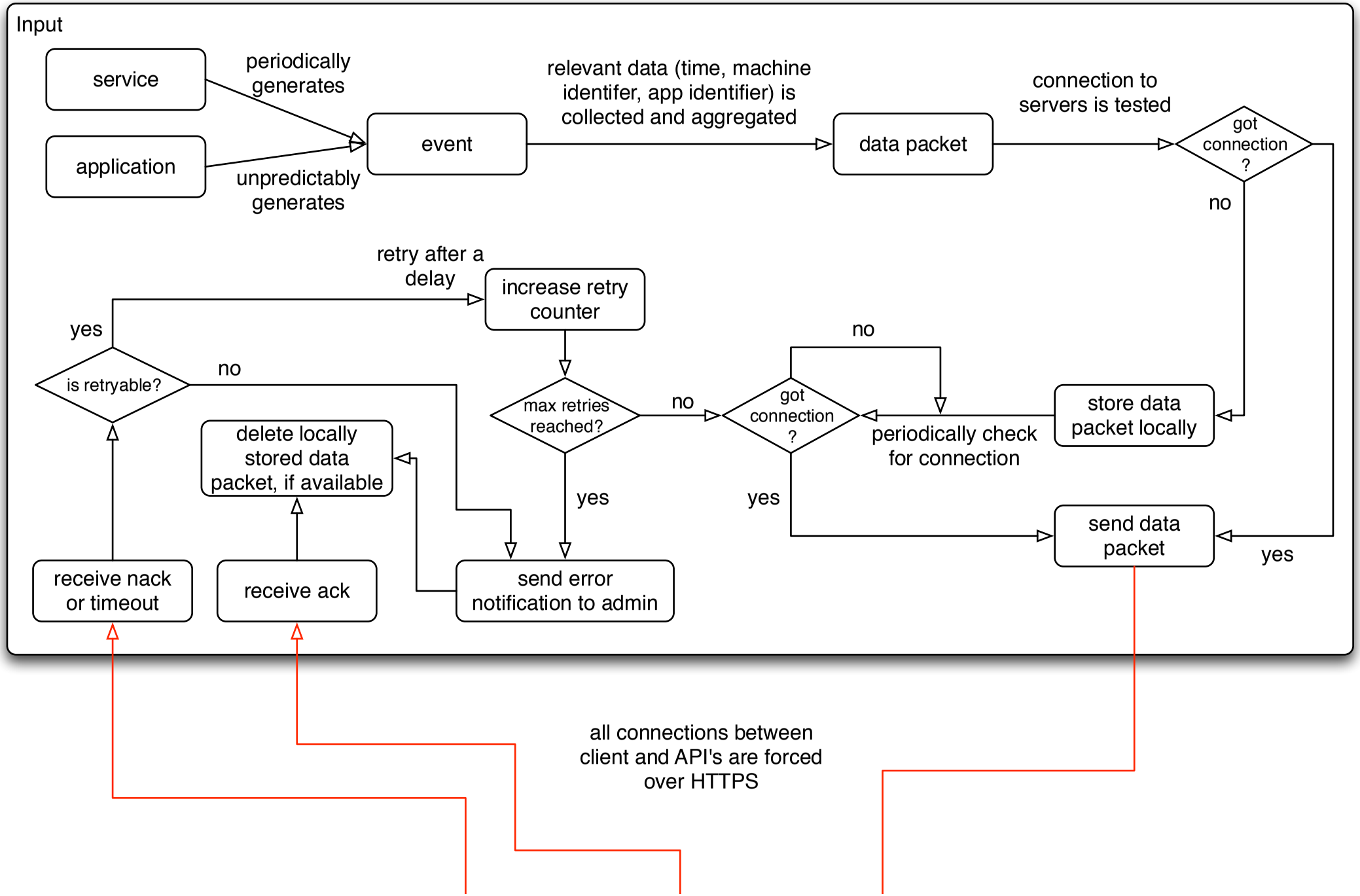
- *Actor* - een persoon of systeem dat interactie met het platform heeft.
- *API* - een Application Programming Interface - een schil om een syteem, dat dient als communicatielaag met de buitenwereld.
- *Applicatie* - een softwareproduct.
- *B2B* - Business to Business, het leveren van diensten aan bedrijven.
- *Event* - Een gebeurtenis (fysiek of in software) welke plaatsvindt buiten de grenzen van het systeem maar binnen het systeem afgehandeld wordt.
- *IDE* - een Integrated Development Environment is een ontwikkelomgeving voor software, meestal bestaande uit een teksteditor, compiler en debugger.
- *Module* - Een component van een systeem met eigen, geïsoleerde werking. De buitengrenzen van een module veranderen over het algemeen weinig, terwijl de werking van een module ook later aangepast kan worden.
- *PHP* - PHP: Hypertext Preprocessor is een scripttaal die op veel webservers gebruikt wordt om dynamische webpagina's te genereren op basis van meegeleverde variabelen.
- *Platform* - Een softwaresysteem dat voor meerdere toepassingen gebruikt kan worden.
- *RUP* - Rational Unified Process, een processmethode ontwikkeld door IBM.
- *Use Case* - Een beschrijving van de werking van een systeem op basis van input of een aanroep.
- *Wrapper* - Een laag om een systeem dat dient als vertaalslag en beveiliging voor de kernsystemen.

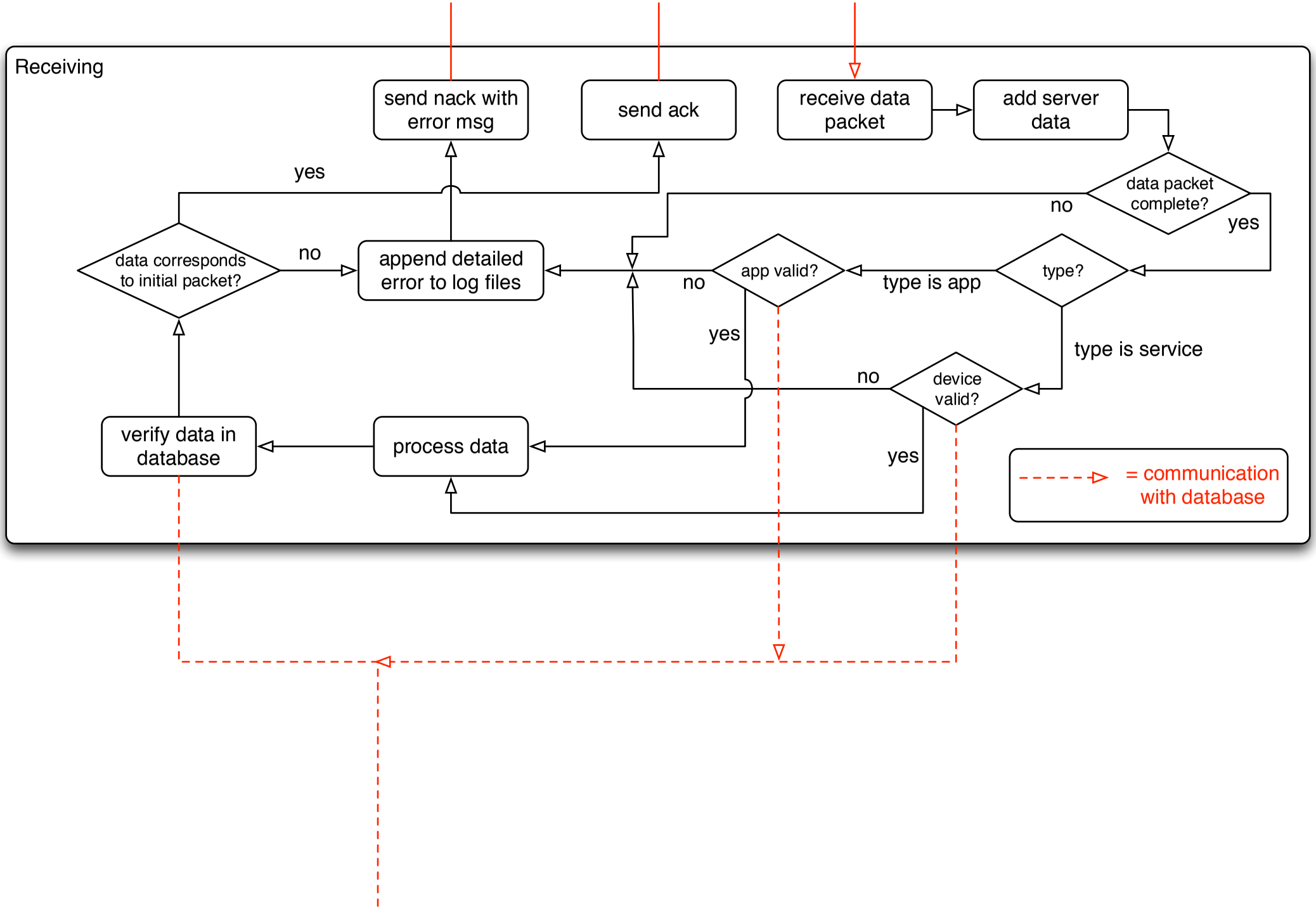
9. Lijst van figuren

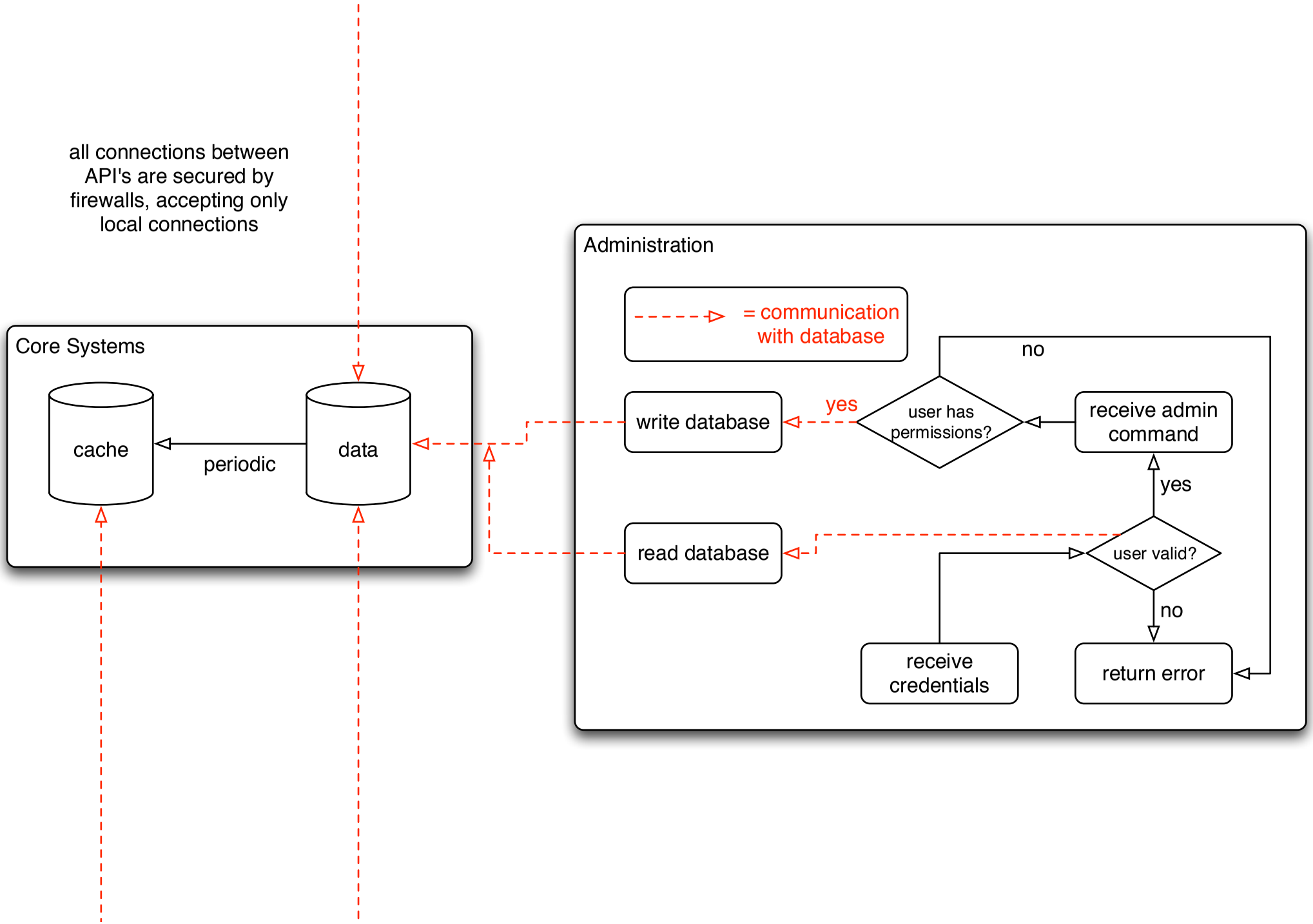
10. Lijst van tabellen

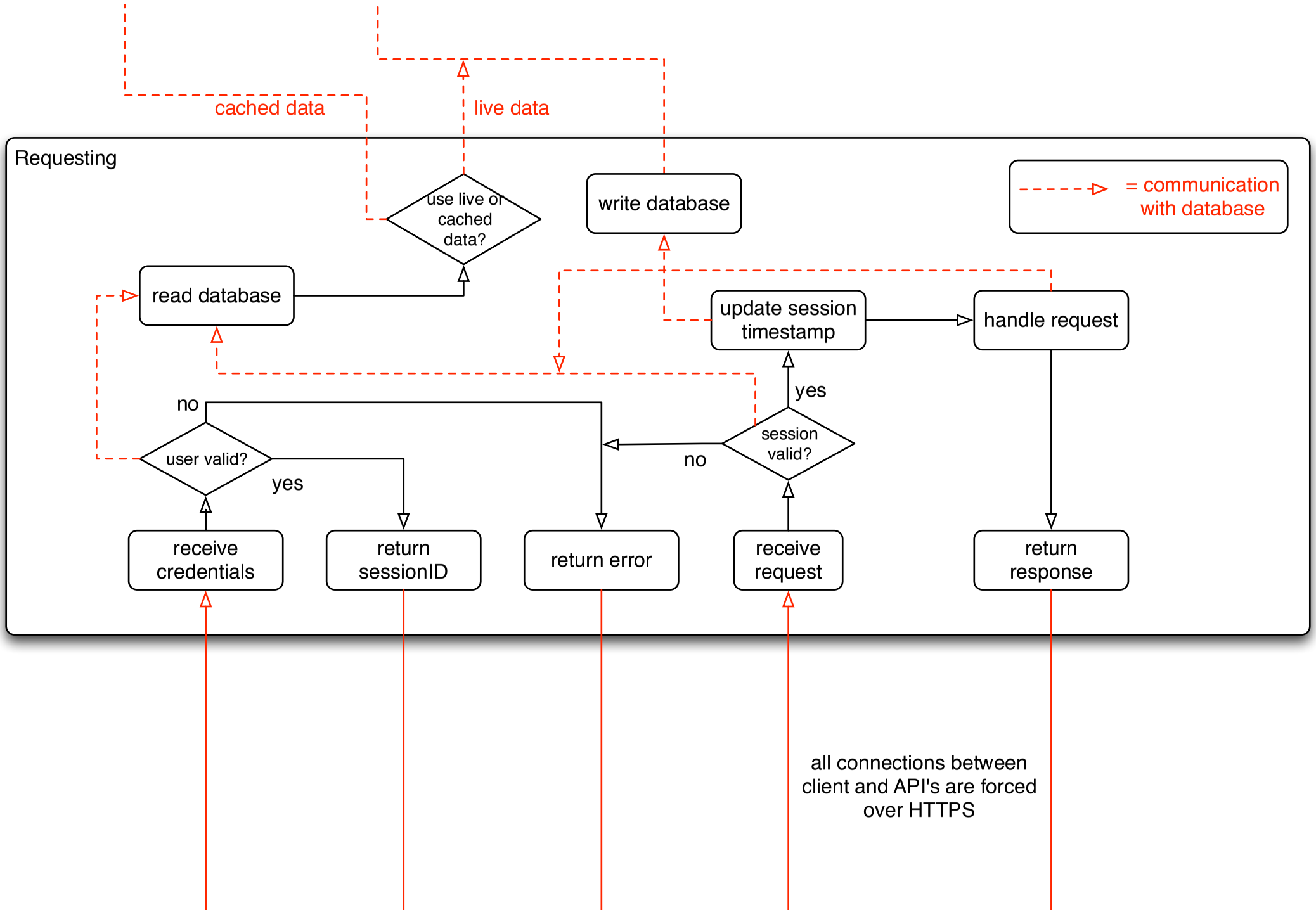
11. Lijst van bijlagen

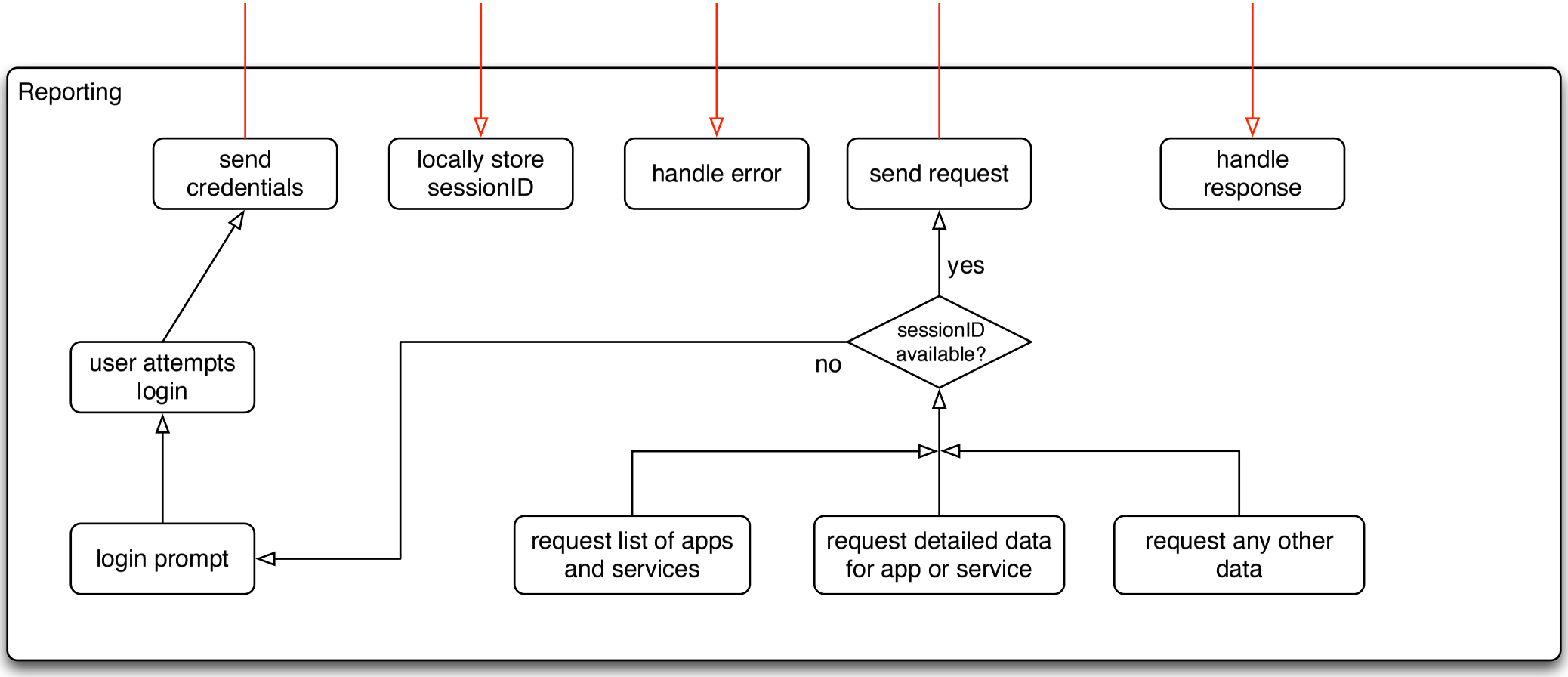
Bijlage A: Flowchart functioneel ontwerp


Input
service
application
periodically generates
unpredictably generates
event
relevant data (time, machine identifer, app identifier) is collected and aggregated
data packet
connection to servers is tested
got connection ?
no
yes
store data packet locally
periodically check for connection
no
yes
send data packet
yes
retry after a delay
increase retry counter
yes
is retryable?
no
max retries reached?
no
yes
delete locally stored data packet, if available
receive nack or timeout
receive ack
send error notification to admin
all connections between client and API's are forced over HTTPS

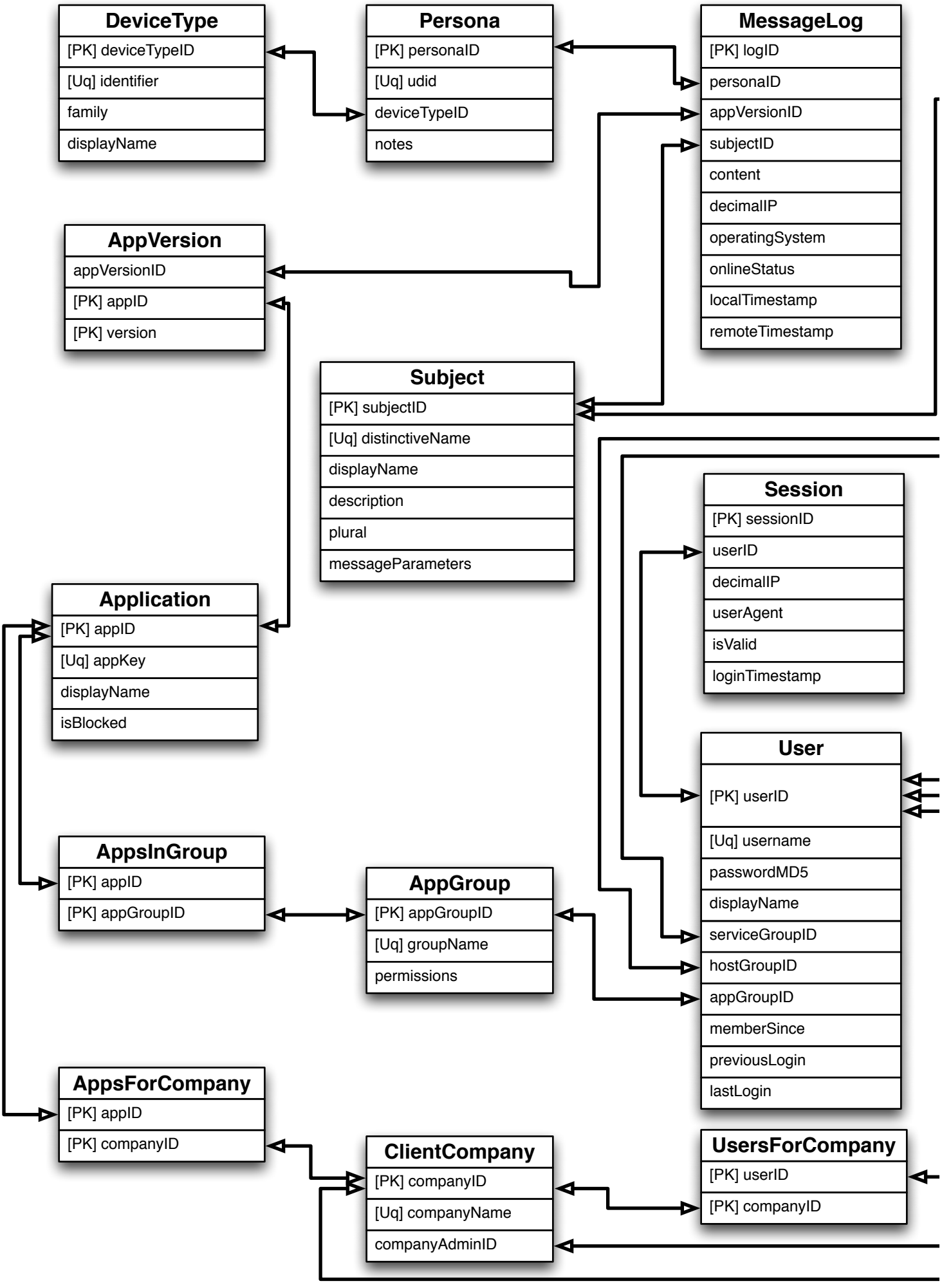

Receiving
send nack with error msg
send ack
receive data packet
add server data
data packet complete?
yes
no
data corresponds to initial packet?
no
append detailed error to log files
app valid?
no
type?
type is app
yes
type is service
no
device valid?
yes
verify data in database
process data
= communication with database

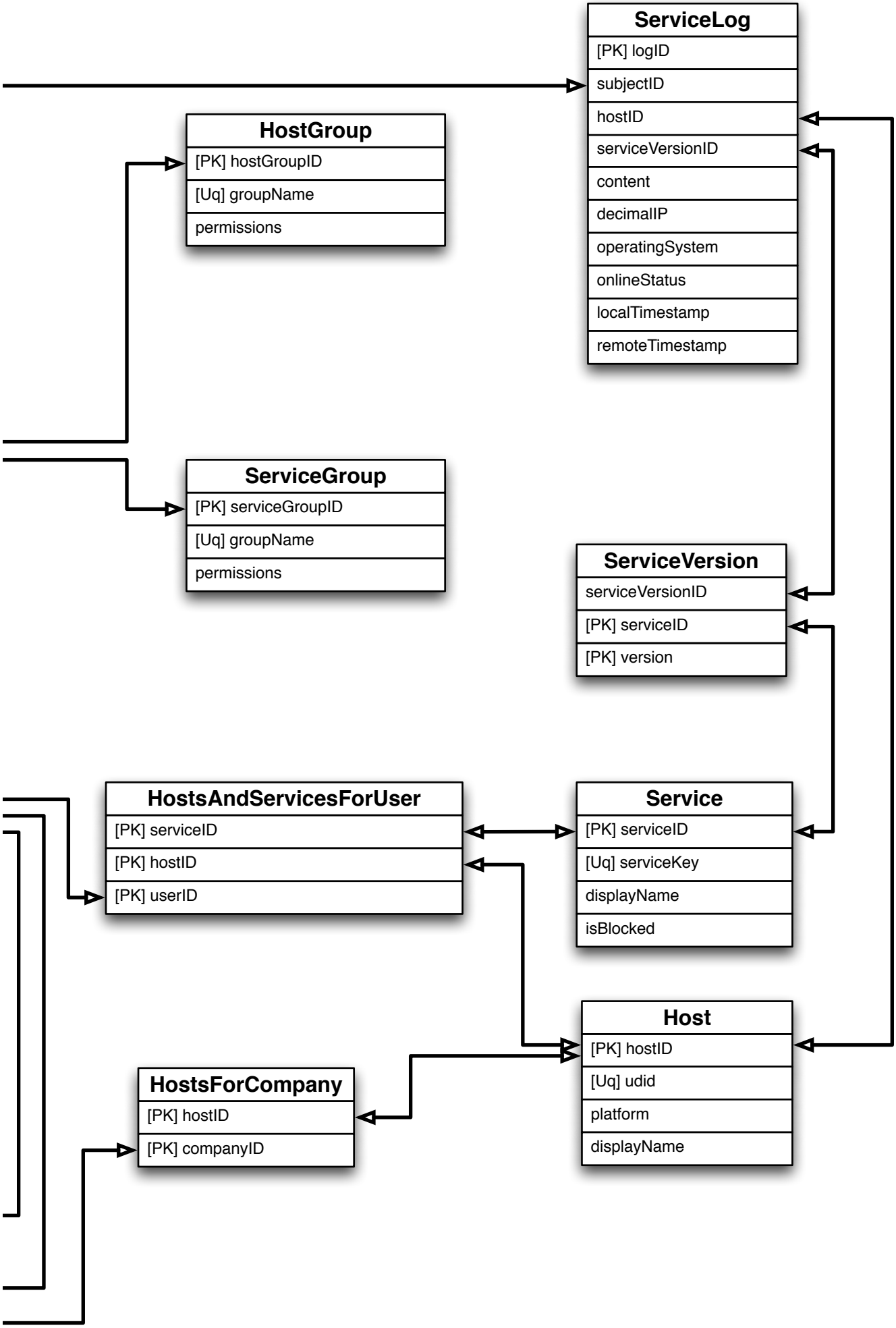

all connections between API's are secured by firewalls, accepting only local connections
Core Systems
cache
periodic
data
Administration
= communication with database
write database
yes
user has permissions?
no
receive admin command
yes
read database
user valid?
no
receive credentials
return error


cached data
live data
Requesting
use live or cached data?
write database
= communication with database
read database
update session timestamp
handle request
yes
no
user valid?
session valid?
no
yes
receive credentials
return sessionID
return error
receive request
return response
all connections between client and API's are forced over HTTPS


Reporting
send credentials
locally store sessionID
handle error
send request
handle response
yes
sessionID available?
no
user attempts login
login prompt
request list of apps and services
request detailed data for app or service
request any other data

Bijlage B: Database design


DeviceType
[PK] deviceTypeID
[Uq] identifier
family
displayName
Persona
[PK] personaID
[Uq] udid
deviceTypeID
notes
MessageLog
[PK] logID
personaID
appVersionID
subjectID
content
decimalIP
operatingSystem
onlineStatus
localTimestamp
remoteTimestamp
AppVersion
appVersionID
[PK] appID
[PK] version
Subject
[PK] subjectID
[Uq] distinctiveName
displayName
description
plural
messageParameters
Session
[PK] sessionID
userID
decimalIP
userAgent
isValid
loginTimestamp
Application
[PK] appID
[Uq] appKey
displayName
isBlocked
User
[PK] userID
[Uq] username
passwordMD5
displayName
serviceGroupID
hostGroupID
appGroupID
memberSince
previousLogin
lastLogin
AppsInGroup
[PK] appID
[PK] appGroupID
AppGroup
[PK] appGroupID
[Uq] groupName
permissions
AppsForCompany
[PK] appID
[PK] companyID
ClientCompany
[PK] companyID
[Uq] companyName
companyAdminID
UsersForCompany
[PK] userID
[PK] companyID


ServiceLog
[PK] logID
subjectID
hostID
serviceVersionID
content
decimalIP
operatingSystem
onlineStatus
localTimestamp
remoteTimestamp
HostGroup
[PK] hostGroupID
[Uq] groupName
permissions
ServiceGroup
[PK] serviceGroupID
[Uq] groupName
permissions
ServiceVersion
serviceVersionID
[PK] serviceID
[PK] version
HostsAndServicesForUser
[PK] serviceID
[PK] hostID
[PK] userID
Service
[PK] serviceID
[Uq] serviceKey
displayName
isBlocked
Host
[PK] hostID
[Uq] udid
platform
displayName
HostsForCompany
[PK] hostID
[PK] companyID

Bijlage C: Receive API

post

URL

https://api.codingdutchmen.com/Stats/receive/post.php

Argumenten

Element	Type	Lengte	Opmerkingen
remoteTimestamp	date (ISO 8601)		De datum en tijd waarop het event plaatsvond
senderType	integer		1 voor app, 2 voor service
deviceIdentifier	string	40 (app) 10-40 (service)	Een unieke identifier voor het verzendende apparaat. Dit kan bijvoorbeeld een MAC-adres of een serienummer van een telefoon zijn.
onlineStatus	integer		0 voor offline, 1 voor online
osVersion	string	1-20	Vb.: 4.0.2
appKey	string	32	Een appKey wordt gegenereerd bij het aanmaken van een applicatie of service.
subject	string	1-30	
content	string	0-100	Content is optioneel
appVersion	string	2-10	De versie van de verzendende applicatie
hardwareFamily	string	2-20	iPhone, iPod Touch, Android, enzovoorts.
machineIdentifier	string	2-20	iPhone2,1, Samsung Galaxy S, Google Nexus One, enzovoorts.
debug	string (optional)		Als deze parameter de waarde “true” bevat worden debugberichten in de serverlogs opgenomen en zullen opgenomen gegevens aan het eind van het proces uit de database verwijderd worden.

Waarden in de respons (success)

Element	Type	Lengte	Opmerkingen
success	string		Een string die het succes van het verwerken beschrijft.
@id	integer		De unieke identifier van het in de database geplaatste bericht.

Waarden in de respons (error)

Element	Type	Lengte	Opmerkingen
error	string		In <error> zijn een error-integer en retry-boolean opgenomen. Op basis hiervan kan bepaald worden hoe met de error omgegaan moet worden.
@code	integer		De numerieke errorcode
@retry	integer		1 als het bericht opnieuw verzonden kan worden, 0 als dit niet moet gebeuren

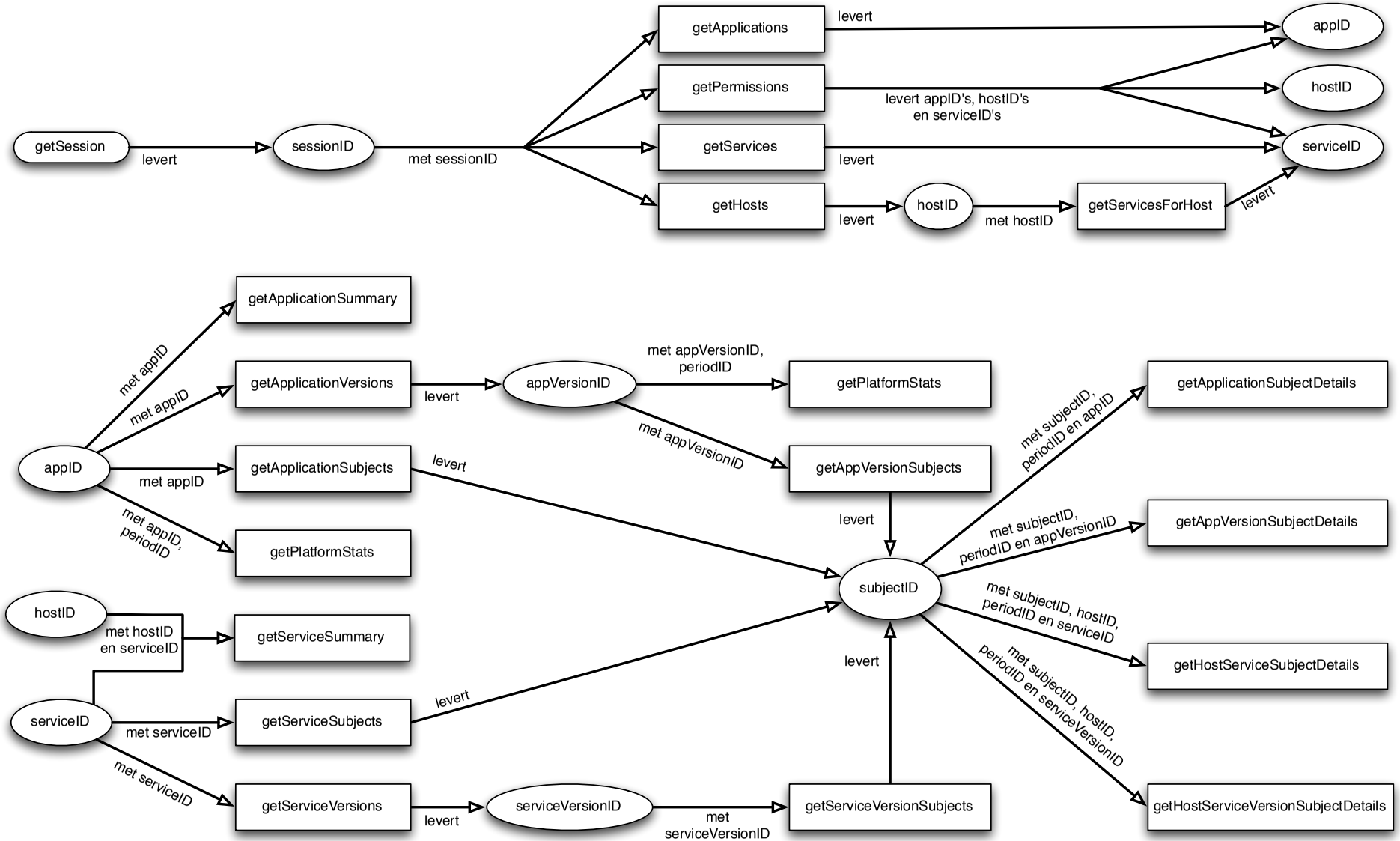
Voorbeeld response (success)


```
– <success id="347529">
    Data packet was succesfully inserted and then removed from the database.
  </success>
```


Voorbeeld response (error)


```
– <error code="-4" retry="0">
    The argument provided does not meet the requirements. See server error log for more detail.
  </error>
```

Bijlage D: Request flow


getSession
levert
sessionID
met sessionID
getApplications
getPermissions
getServices
getHosts
levert
appID
levert appID's, hostID's
en serviceID's
hostID
levert
serviceID
levert
hostID
met hostID
getServicesForHost
levert
getApplicationSummary
met appID
getApplicationVersions
levert
appVersionID
met appVersionID,
periodID
getPlatformStats
met appVersionID
appID
met appID
getApplicationSubjects
levert
getAppVersionSubjects
met appID,
periodID
getPlatformStats
levert
met subjectID,
periodID en appID
getApplicationSubjectDetails
met subjectID,
periodID en appVersionID
getAppVersionSubjectDetails
subjectID
met subjectID, hostID,
periodID en serviceID
getHostServiceSubjectDetails
hostID
met hostID
en serviceID
getServiceSummary
levert
met subjectID, hostID,
periodID en serviceVersionID
getHostServiceVersionSubjectDetails
serviceID
met serviceID
getServiceSubjects
levert
met serviceID
getServiceVersions
levert
serviceVersionID
met
serviceVersionID
getServiceVersionSubjects

Bijlage E: Request API

Inleiding

Deze bijlage bevat de definitie van de Request-API. De definitie legt vast welke argumenten meegeleverd moeten worden als ze aangeroepen worden en welke response verwacht kan worden.

Inhoudsopgave

getSession

Beschrijving

GetSession levert een sessionID, welke gebruikt wordt voor alle opvolgende requests. Als een sessionID reeds bestaat voor de inloggende gebruiker wordt deze teruggeven. Indien hij nog niet bestaat wordt een nieuwe gegenereerd.

URL

https://api.codingdutchmen.com/Stats/request/getSession.php

Argumenten

Element	Type	Lengte	Opmerkingen
username	string	1-20	
password	string (MD5)	32	Een MD5-waarde bestaat altijd uit 32 hexadecimale karakters

Waarden in de respons

Element	Type	Lengte	Opmerkingen
login	node		
<login>			
sessionID	integer		
userID	integer		
displayName	string	0-40	
appGroupID	integer		
hostGroupID	integer		
serviceGroupID	integer		
memberSince	date (ISO 8601)		Vb.: 2004-02-12T15:19:21+00:00
previousLogin	date (ISO 8601)		Vb.: 2010-11-28T13:05:20+01:00

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/getSession.php?
username=luc&password=411fb54cf88adb73e8aae2a826a9d9ef
```

Voorbeeld response

```
– <login>
    <sessionID>922</sessionID>
    <userID>1</userID>
    <displayName>Luc van Donkersgoed</displayName>
    <appGroupID>1</appGroupID>
    <hostGroupID>3</hostGroupID>
    <serviceGroupID>2</serviceGroupID>
    <memberSince>2010-10-29T16:13:06+02:00</memberSince>
    <previousLogin>2010-11-28T13:05:20+01:00</previousLogin>
  </login>
```

getApplications

Beschrijving

GetApplications levert een lijst van beschikbare applicaties voor een gebruiker.

URL

https://api.codingdutchmen.com/Stats/request/getApplications.php

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		SessionID is verkregen met getSession.php

Waarden in de respons

Element	Type	Lengte	Opmerkingen
applications	node		
<applications>			
app	node		
<app>			
appID	integer		
appKey	string (MD5)	32	Een unieke identifier voor de app
displayName	string	0-50	De naam zoals deze aan de gebruiker getoond kan worden
isBlocked	integer		0 of 1

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/getApplications.php?
sessionID=922
```

Voorbeeld response


```
- <applications>
  - <app>
      <appID>37</appID>
      <appKey>9D0883FB1333AB24F92115900E7447C8</appKey>
      <bundleName>EpisodesLite</bundleName>
      <displayName>Episodes</displayName>
      <isBlocked>0</isBlocked>
    </app>
  - <app>
      <appID>28</appID>
      <appKey>DB0174D74AF5E3626E2A2F7B2E8930DF</appKey>
      <bundleName>FMLFree</bundleName>
      <displayName>F*** My Life Free</displayName>
      <isBlocked>0</isBlocked>
    </app>
  - <app>
      <appID>36</appID>
      <appKey>D94AD7CCFD829792F6706D3B76E3769A</appKey>
      <bundleName>EyeDart</bundleName>
      <displayName>EyeDart</displayName>
      <isBlocked>0</isBlocked>
    </app>
  </applications>
```

getServices

Beschrijving

GetServices levert een lijst van beschikbare services voor een gebruiker.

URL

`https://api.codingdutchmen.com/Stats/request/getServices.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		SessionID is verkregen met getSession.php

Waarden in de respons

Element	Type	Lengte	Opmerkingen
services	node		
<services>			
service	node		
<service>			
serviceID	integer		
serviceKey	string (MD5)	32	Een unieke identifier voor de service
displayName	string	0-50	De naam zoals deze aan de gebruiker getoond kan worden
isBlocked	integer		0 of 1

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getServices.php?sessionID=922`

Voorbeeld response


```
– <services>
  – <service>
      <serviceID>3</serviceID>
      <appKey>5BD7350C3E6C47EEE0A4777BD9F78BF0</appKey>
      <displayName>Apache Logger</displayName>
      <isBlocked>0</isBlocked>
    </service>
  – <service>
      <serviceID>1</serviceID>
      <appKey>AC647732ADA3AAA792206E8C17E47033</appKey>
      <displayName>Core Server Logger</displayName>
      <isBlocked>0</isBlocked>
    </service>
  – <service>
      <serviceID>4</serviceID>
      <appKey>8BA69E6A3AAD06E2DB4636BBB028D1F5</appKey>
      <displayName>MySQL Logger</displayName>
      <isBlocked>0</isBlocked>
    </service>
  </services>
```

getHosts

Beschrijving

GetHosts levert een lijst van beschikbare hosts voor een ingelogde gebruiker.

URL

`https://api.codingdutchmen.com/Stats/request/getHosts.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		SessionID is verkregen met getSession.php

Waarden in de respons

Element	Type	Lengte	Opmerkingen
hosts	node		
<hosts>			
host	node		
<host>			
hostID	integer		
udid	string	0-40	Een unieke identifier voor de host, bijvoorbeeld een MAC-adres
platform	string	0-100	Een identifier voor het hardwareplatform, bijvoorbeeld ‘x86-64’ voor 64-bits systemen.
displayName	integer	0-100	Een herkenbare naam voor een server

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getHosts.php?sessionID=922`

getServicesForHost

Beschrijving

GetServicesForHost neemt een hostID en levert de services die voor deze host bekeken mogen worden.

URL

`https://api.codingdutchmen.com/Stats/request/getServicesForHost.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
hostID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
services	node		
<services>			
service	node		
<service>			
serviceID	integer		
serviceKey	string (MD5)	32	Een unieke identifier voor de service
displayName	string	0-50	De naam zoals deze aan de gebruiker getoond kan worden
isBlocked	integer		0 of 1

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getServicesForHost.php?sessionID=922&hostID=5`

getPermissions

Beschrijving

GetPermissions combineert de gegevens van getApplications, getServices en getHosts en voegt deze samen tot een lijst van toonbare applicaties en services.

URL

`https://api.codingdutchmen.com/Stats/request/getPermissions.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		SessionID is verkregen met getSession.php

Waarden in de respons

Element	Type	Lengte	Opmerkingen
permissions	node		
<permissions>			
applications	node		
services	node		
<applications>			
app	node		
<app>			
appID	integer	32	Een unieke identifier voor de app
displayName	string	0-50	De naam zoals deze aan de gebruiker getoond kan worden
usersToday	integer		Het aantal gebruikers van de app sinds 00:00 vandaag.
sessionsToday	integer		Het aantal sessies van de app sinds 00:00 vandaag.
weightParameter	integer		Een suggestie voor de sortering van de apps. Hogere waarden zijn ‘zwaarder’ en komen lager in de lijst.
<services>			
service	node		
<service>			
serviceID	integer		Een unieke identifier voor de service
hostID	integer		Een unieke identifier voor de host
displayName	string	0-100	De naam van de service/host-combinatie, bijvoorbeeld “Miles - MySQL Queries / Minute”
averageToday	double		De gemiddelde waarde van deze service/host-combinatie sinds 00:00
weightParameter	integer		Een suggestie voor de sortering van de service/host-combinaties. Hogere waarden zijn ‘zwaarder’ en komen lager in de lijst.

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/getPermissions.php?sessionID=922
```

Voorbeeld response

```
- <permissions>
  - <applications>
    - <app>
        <appID>-1</appID>
        <displayName>Totals</displayName>
        <usersToday>196</usersToday>
        <sessionsToday>329</sessionsToday>
        <weightParameter>-1</weightParameter>
      </app>
    - <app>
        <appID>37</appID>
        <displayName>Episodes</displayName>
        <usersToday>136</usersToday>
        <sessionsToday>214</sessionsToday>
        <weightParameter>1</weightParameter>
      </app>
    - <app>
        <appID>28</appID>
        <displayName>F*** My Life Free</displayName>
        <usersToday>60</usersToday>
        <sessionsToday>115</sessionsToday>
        <weightParameter>2</weightParameter>
      </app>
    </applications>
  - <services>
    - <service>
        <serviceID>1</serviceID>
        <hostID>4</hostID>
        <displayName>Miles - Server Load / Minute</displayName>
        <averageToday>0.10</averageToday>
        <weightParameter>1</weightParameter>
      </service>
    </services>
  </permissions>
```

getApplicationSummary

Beschrijving

Dit script levert een samenvatting van gegevens over een bepaalde applicatie.

URL

`https://api.codingdutchmen.com/Stats/request/getApplicationSummary.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
appID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
summary	node		
<permissions>			
appName	String		
periodDays	Integer		
periodName	String		
subPeriodName	String		
userAvg	Float		
userMax	Integer		
userMin	Integer		
sessionAvg	Float		
sessionMax	Integer		
sessionMin	Integer		
graphData	node		
messages	node		
<graphData>			
elements	node		
<elements>			
element	node		
<element>			

Element	Type	Lengte	Opmerkingen
id	Date (ISO 8601)		
value	Integer		
<messages>			
message	node		
<message>			
date	Date (ISO 8601)		
subject	String		
content	String		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getApplicationSummary.php?sessionID=1169&periodID=1&appID=23
```

getApplicationVersions

Beschrijving

Dit script levert de versies van een applicatie.

URL

`https://api.codingdutchmen.com/Stats/request/getApplicationVersions.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	Integer		
appID	Integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
versions	node		
<versions>			
appID	Integer		
displayName	String		
appVersion	Node		
<appVersion>			
appVersionID	Integer		
version	String		

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getApplicationVersions.php?sessionID=939&appID=37`

getApplicationSubjects

Beschrijving

Dit script levert de subjects die door een applicatie verzonden worden.

URL

```
https://api.codingdutchmen.com/Stats/request/
getApplicationSubjects.php
```

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	Integer		
appID	Integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
subjects	node		
<subjects>			
appID	Integer		
displayName	String		
subject	Node		
<subject>			
subjectID	Integer		
distinctiveName	String		
displayName	String		
plural	String		
messageParameters	Binary		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getApplicationSubjects.php?sessionID=959&appID=37
```

getServiceSummary

Beschrijving

Dit script levert een samenvatting van gegevens van een bepaalde service.

URL

`https://api.codingdutchmen.com/Stats/request/getServiceSummary.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
serviceID	integer		
hostID	Integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
summary	Node		
<summary>			
serviceName	String		
hostName	String		
subject	String		
graphData	Node		
<graphData>			
elements	Node		
<elements>			
element	Node		
<element>			
id	Integer		
avg	Float		
min	Float		
max	Float		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getServiceSummary.php?
sessionID=1169&hostID=5&serviceID=1&periodID=2
```

getServiceSubjects

Beschrijving

Dit script levert de subjects van een service.

URL

`https://api.codingdutchmen.com/Stats/request/getServiceSubjects.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
serviceID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
subjects	Node		
<subjects>			
serviceID	Integer		
displayName	String		
subject	Node		
<subject>			
subjectID	Integer		
distinctiveName	String		
displayName	String		
plural	String		
messageParameters	Binary		

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getServiceSubjects.php?sessionID=966&serviceID=2`

getServiceVersions

Beschrijving

Dit script levert de versions van een service.

URL

`https://api.codingdutchmen.com/Stats/request/getServiceVersions.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
serviceID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
versions	Node		
<versions>			
serviceID	Integer		
displayName	String		
serviceVersion	Node		
<serviceVersion>			
serviceVersionID	Integer		
version	String		

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getServiceVersions.php?sessionID=939&serviceID=1`

getAppVersionSubjects

Beschrijving

Dit script levert de subjects die door een bepaalde appVersion verzonden worden.

URL

`https://api.codingdutchmen.com/Stats/request/getAppVersionSubjects.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
appVersionID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
subjects	node		
<subjects>			
appID	Integer		
displayName	String		
appVersionID	Integer		
version	String		
subject	Node		
<subject>			
subjectID	Integer		
distinctiveName	String		
displayName	String		
plural	String		
messageParameters	Binary		

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getAppVersionSubjects.php?sessionID=959&appVersionID=27`

getServiceVersionSubjects

Beschrijving

Dit script levert de subjects die door een bepaalde serviceVersion verzonden worden.

URL

`https://api.codingdutchmen.com/Stats/request/getServiceVersionSubjects.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
serviceVersionID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
subjects	Node		
<subjects>			
serviceVersionID	Integer		
displayName	String		
subject	Node		
<subject>			
subjectID	Integer		
distinctiveName	String		
displayName	String		
plural	String		
messageParameters	Binary		

Voorbeeld aanroep

`https://api.codingdutchmen.com/Stats/request/getServiceVersionSubjects.php?sessionID=966&serviceVersionID=3`

getPlatformStats

Beschrijving

Dit script levert de grafiekdata voor de verdeling van hardware, applicatieversies en operating systems.

URL

`https://api.codingdutchmen.com/Stats/request/getPlatformStats.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
appID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
platformStats	Node		
<platformStats>			
graphData	Node		
<graphData>			
type	String		
title	String		
numberofelements	Integer		
names	Node		
elements	Node		
<names>			
name	String		
<name>			
@id	Integer		
<elements>			
element	Node		
<element>			
id	Date (ISO 8601)		
point	Float		
<point>			

Element	Type	Lengte	Opmerkingen
id	Integer		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/getPlatformStats.php?
sessionID=1169&appID=73&periodID=1
```

getApplicationSubjectDetails

Beschrijving

Dit script levert de grafiekdata voor een bepaald subject op een applicatie.

URL

`https://api.codingdutchmen.com/Stats/request/getApplicationSubjectDetails.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
subjectID	Integer		
appID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
applicationMessageDetails	node		
<applicationMessageDetails>			
appID			
appName			
periodDays			
periodName			
subPeriodName			
subjectID			
subjectName			
messageParameters	Binary		
graphs	Node		
<graphs>			
graphData	Node		
<graphData>			
type	String		
title	String		
numberofelements	Integer		

Element	Type	Lengte	Opmerkingen
names	Node		
elements	Node		
<names>			
name	String		
<name>			
@id	Integer		
<elements>			
element	Node		
<element>			
id	Date (ISO 8601)		
point	Float		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getApplicationSubjectDetails.php?
sessionID=1169&appID=23&subjectID=11&periodID=1
```

getAppVersionSubjectDetails

Beschrijving

Dit script levert de grafiekdata voor een bepaald subject op een applicatieversie.

URL

`https://api.codingdutchmen.com/Stats/request/getAppVersionSubjectDetails.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
subjectID	Integer		
appID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
appVersionMessageDetails	node		
<appVersionMessageDetails>			
appID	Integer		
appName	String		
version	String		
periodDays	Integer		
periodName	String		
subPeriodName	String		
subjectID	Integer		
subjectName	String		
messageParameters	Binary		
graphs	Node		
<graphs>			
graphData	Node		
<graphData>			
type	String		
title	String		

Element	Type	Lengte	Opmerkingen
numberofelements	Integer		
names	Node		
elements	Node		
<names>			
name	String		
<name>			
@id	Integer		
<elements>			
element	Node		
<element>			
id	Date (ISO 8601)		
point	Float		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getAppVersionSubjectDetails.php?
sessionID=1169&appVersionID=45&subjectID=4&periodID=2
```

getHostServiceSubjectDetails

Beschrijving

Dit script levert de grafiekdata voor een bepaald subject op een host-service-combinatie.

URL

`https://api.codingdutchmen.com/Stats/request/getHostServiceSubjectDetails.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
subjectID	Integer		
hostID	integer		
serviceID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
hostServiceMessage Details	node		
<hostServiceMessageDetails>			
serviceID			
serviceName			
hostID			
hostName			
subjectID			
subjectName			
messageParameters	Binary		
graphs	Node		
<graphs>			
graphData	Node		
<graphData>			
type	String		
title	String		
numberofelements	Integer		
names	Node		

Element	Type	Lengte	Opmerkingen
elements	Node		
<names>			
name	String		
<name>			
@id	Integer		
<elements>			
element	Node		
<element>			
id	Date (ISO 8601)		
point	Float		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getHostServiceSubjectDetails.php?
sessionID=1169&hostID=23&serviceID=4&subjectID=11&periodID=1
```

getHostServiceVersionSubjectDetails

Beschrijving

Dit script levert de grafiekdata voor een bepaald subject op een host-service-combinatie, beperkt tot een specifieke versie van een service.

URL

`https://api.codingdutchmen.com/Stats/request/getHostServiceVersionSubjectDetails.php`

Argumenten

Element	Type	Lengte	Opmerkingen
sessionID	integer		
periodID	integer		
subjectID	Integer		
hostID	integer		
serviceVersionID	integer		

Waarden in de respons

Element	Type	Lengte	Opmerkingen
hostServiceMessage Details	node		
<hostServiceMessageDetails>			
serviceVersionID	Integer		
serviceName	String		
version	String		
hostID	Integer		
hostName	String		
subjectID	Integer		
subjectName	String		
messageParameters	Binary		
graphs	Node		
<graphs>			
graphData	Node		
<graphData>			
type	String		
title	String		

Element	Type	Lengte	Opmerkingen
numberofelements	Integer		
names	Node		
elements	Node		
<names>			
name	String		
<name>			
@id	Integer		
<elements>			
element	Node		
<element>			
id	Date (ISO 8601)		
point	Float		

Voorbeeld aanroep

```
https://api.codingdutchmen.com/Stats/request/
getHostServiceVersionSubjectDetails.php?
sessionID=1169&hostID=23&serviceVersionID=23&subjectID=11&periodID
=1
```

Bijlage F: Klassen

Filename	Klassenaam	Verantwoordelijkheid
_AppGroup.php	AppGroupUtils	Communicatie met de tabel AppGroup.
_Application.php	ApplicationUtils	Communicatie met de tabel Application. Haalt ook gegevens uit de tabel MessageLog en de gecachete vormen daarvan op. Een aanvraag op basis van een appID of appVersionID zal over het algemeen door deze klasse afgehandeld worden.
_Caching.php	CachingUtils	Bevat functies en SQL-scripts om gegevens uit de live-database naar de cache-tabellen te kopiëren.
_ClientCompany.php	ClientCompanyUtils	Communicatie met de tabel ClientCompany.
_DataPacket.php	DataPacket	In een DataPacket worden berichten die bij de receive-api binnenkomen gevalideerd en opgeslagen.
_Globals.php	Globals	In de Globals worden waarden opgeslagen die door alle andere klassen gebruikt kunnen worden. Hieronder vallen bijvoorbeeld errorcodes en vast tijdswaarden.
_Host.php	HostUtils	Communicatie met de tabel Host.
_HostGroup.php	HostGroupUtils	Communicatie met de tabel HostGroup. Werkt nauw samen met de klasse UserUtils, aangezien veel informatie in de HostGroups gekoppeld wordt aan de rechten van een gebruiker.
_HostsAndServices.php	HostsAndServicesUtils	Communicatie met de tabel HostsAndServices.
_Service.php	ServiceUtils	Communicatie met de tabel Service.
_ServiceGroup.php	ServiceGroupUtils	Communicatie met de tabel ServiceGroup. Werkt nauw samen met de klasse UserUtils, aangezien veel informatie in de ServiceGroups gekoppeld wordt aan de rechten van een gebruiker.
_Subject.php	SubjectUtils	Communicatie met de tabel Subject.
_User.php	UserUtils	Communicatie met de tabel User. Deze klasse verzorgt ook het ophalen en bewerken van sessies, aangezien deze altijd aan een gebruiker gekoppeld zijn.