Stageonderzoek Appsemble

David Diks

December 13, 2016

1 Inleiding

Appsemble als platform is een snel-groeiend platform voor het ontwikkelen van smartphones applicaties zonder enige kennis te hebben van programmeren. Via een simpele webapplicatie kunnen eindgebruikers een applicatie in elkaar knutselen, testen en doorsturen naar de relevante app-stores (Namelijk de Apple App Store en de Google Play Store).

Echter zijn componenten op het moment zeer gelimiteerd. Zij moeten namelijk ontwikkeld worden door het Appsemble team. Personen en bedrijven kunnen suggesties doen voor gewenste componenten, waarna d-centralize hiernaar gaat kijken.

Dit process is echter gecompliceerd en het kan een tijd duren voordat deze onderdelen geïmplementeerd zijn. Het bedrijf moet in dit geval dus wachten tot de aanvraag in behandeling word genomen.

Dit heeft geleid tot de volgende vraag; Kunnen wij niet zelf componenten ontwikkelen?

Op het moment is dit onmogelijk. Als eerste kunnen externe developers niet bij de bron van Appsemble. Dit betekent dat zij geen componenten kunnen maken en testen. Daarnaast is er geen enkele API beschikbaar om in Appsemble zelf externe componenten te faciliteren. Hier moet verandering in komen.

De beginstap van het ontwikkelen hiervan is dit onderzoek, waarin geprobeerd word een antwoord te geven op de volgende vraag: *Hoe wordt het process voor het ontwikkelen van plug-in componenten voor Appsemble zo simpel mogelijk gemaakt?*

2 Samenvatting

Het ontwerpen van een sdk vooral het anticiperen van de benodigdheden van aankomende gebruikers. Een developer kan veel willen zonder dat dit ook ooit echt gebruikt gaat worden. Hiervoor moeten de beschikbare keuzes eerst uitgewerkt worden.

Als eerst word er gekeken naar het bedrijf zelf, om te relativeren wie er gaat ontwikkelen en wat zij als ervaring hebben.

Daarna word er gekeken wat er standaard verwacht word van de appsemble sdk. Dit zijn onderdelen die nodig zijn.

Er word ook gekeken naar onderdelen die standaard in sdk's aanwezig horen te zijn, zoals documentatie, voorbeelden en meer.

Als laatste word er ook gekeken hoe een extension het gemakkelijkst gepubliceerd en verspreid kan worden. Er word naar meerdere manieren gekeken en uiteindelijk een aanrader gedaan.

Contents

3 Wie ontwikkelt met Appsemble

Als platform is Appsemble al live (Te vinden op https://www.appsemble.com/en/) en word al in de praktijk toegepast voor enkele apps.

3.1 Groen&Zo

Groen&zo is een app, ontwikkeld door Liesbeth Pilon, om mensen te helpen met hun groenwerk. De app is namelijk een woordenboek van planten, die de naam vertaalt van nederlandse volksmond naar de officieële naam en de engelse naam.

3.2 App-S

Op het terrein van Strijp-S in Eindhoven is veel gaande. App-S helpt ondernemers om te zien wat er aan de hand is in de omgeving van Strijp, zoals lectures, meetings en meer.

3.3 DOCFeed

DOCfeed is een plek in Strijp-S waar filmmakers en documentaire-liefhebbers samen komen. Hier worden films, workshops en tentoonstellingen weergegeven voor de geïnteresseerden.

4 Welke API's moeten beschikbaar zijn voor de eindgebruiker

Voor het maken van een SDK is het bijzonder belangrijk om te weten wie het publiek is. Het zomaar ontwikkelen van een SDK zorgt ervoor dat er een rare collectie van ongerelateerde functies voor gaan komen in het uiteindelijke product.

Het doel is dus om een enkel, cohesief pakket te maken. Een voorbeeld hiervan is de JDK[1]. Werk hieraan gebeurt in incrementele stappen. Waarbij iedere release duidelijk een aantal producten heeft, in plaats van een collectie halve API's [2].

Werk aan de JDK werkt dus in grote libraries, die besloten worden als de roadmap van het Java project. De community heef hier enige invloed op door middel van het java forum.

4.1 Interfacen met app

Meta-data over de app is belangrijk. Een developer moet toegang hebben tot runtime constantes (Zijnde App ID, naam en gegevens over de huidige status.). Hiervoor is er een simpele "Read-Only"[2] interface die om deze gegevens heen wrapt.[3]

Toegang zou bestaan uit de volgende onderdelen:

- Versie-nummer
- App settings (Naam, kleuren en meer)
- Runtime variabelen (Wifi/LTE, batterijstatus, etc)

4.2 Instellingen

Instellingen in een App zijn altijd op een plek te bereiken. Hiervoor is al een scherm gedefinieërd. Er is hier enkel geen toegang toe.

Met de Settings-Api zou er een geunificeerde toegang zijn tot de instellingen, waarmee een gebruiker zijn eigen instellingen kan aanmaken, en de status hiervan kan inlezen. Dit zorgt ervoor dat een developer snel instelling kan maken en hierop kan reageren

[1] Java Development Kit, de tools nodig om java te ontwikkelen
[2] Enkel leesbaar, niet schrijfbaar
[3] Hij omvat alle gegevens en geeft hier simpele toegang toe

Deze API bestaat uit de volgende onderdelen:

- Uitlezen van settings
- Registreren van instellingen
- Intellingen programmatisch veranderen

4.3 Interfacen met Appsemble datastores

Opslag van data is essentieël voor het ontwikkelen van een applicatie. Op het moment slaan de componenten van appsemble al data op in de Appsenble-datastores.

Deze interface is echter niet bereikbaar op het moment, aangezien het een intern component is om mee te ontwikkelen en dus enkel beschikbaar is voor de d-centralize developers.

Het publiekelijk maken van deze API voor bepaalde doeleinden maakt het simpeler om componenten te ontwikkelen, aangezien de component developer zelf geen backend zal hoeven te hosten.

Er zullen echter bepaalde stops moeten worden gelegd om te zorgen dat developers niet te ver gaan met het gebruiken. Hiervoor zijn een paar oplossingen;

- Gebruikslimieten per dag
- Vastgestelde gratis opslaglimieten
- Enkel bepaalde types opslag regelen. (Text, plaatjes of andere)

De API zou dan bestaan uit de volgende onderdelen;

- Opslaan van data
- Ophalen van data
- Eigen structuren aan brengen (JSON classes)
- Filtering van resultaten (User-focused queries)

4.4 Interfacen met externe API's

Soms is het gebruik van een Appsemble-gehoste API niet genoeg. Een eindgebruiker kan aan limieten komen die vanuit Appsemble zelf komen of heeft simpelweg complexere functies nodig.

Hiervoor moet het mogelijk zijn om gemakkelijk met externe api's te communiceren. Hiermee kan een developer het weer ophalen van het internet of een interface maken met een bestaand network.[4]

4.4.1 Risico's

Het maken van externe calls naar het internet brengt op zich wel een inherent risico met zich mee. Een kwaadwillende developer kan hiermee payloads downloaden en installeren via bestaande beveiligingslekken. Hiervoor zijn enkele oplossingen;

Screening	Apps die deze API gebruiken moeten gescreened worden op exploits.
Restricties op data	Limieten zetten op de data die geladen kan worden (Enkel JSON)

Uiteindelijk zou het volgende in de API zitten

- GET, HEAD, PUT, POST en DELETE http calls op arbitraire endpoints.
- JSON data teruggeven

4.5 Event Hooks

Inhaken op events is een goede manier om te reageren op de app. Een developer kan inhaken op het installeren van een component, het opstarten of het afsluiten van zijn component en meer.

Daarnaast kan een developer inhaken op systeem-events, zoals wifi-verbinding of een impending reboot.

- Inhaken op OS events
- Zelf events kunnen versturen
- Meewerken met events uit de app zelf

4.6 Questionnaire

Er moet dus een plan opgesteld worden voor het ontwikkelen van de Appsemble SDK. Hierbij is er een kleine enquete gedaan om uit te vinden welke onderdelen van een SDK prioriteit hebben volgens developers.

In de enquete worden vragen gesteld over de verschillende API's die beschikbaar kunnen worden gemaakt in de aankomende versie van de SDK. Er word eerst gevraagd welke componenten zij graag zouden zien in de appsemble SDK. Vervolgens word er aangegeven welk onderdeel zij het liefste zouden zien in de SDK.

Voor het onderzoek, zie de appendix.

[4] Bijvoorbeeld interne API's, Google services en meer

4.7 Prioriteiten

5 Wat zit er in een standaard SDK

Een groot gedeelte van de bruikbaarheid van een SDK is afhankelijk van hoe makkelijk het is om hiermee te ontwikkelen. Dit betekent dat er genoeg tutorials beschikbaar moeten zijn, dat vragen snel beantwoord worden en dat er met de community gepraat word.

5.1 Tutorials

Het voorbereiden van tutorials is essentieël voor het bevorderen van het gebruik van een library of een SDK. Tutorials zorgen voor een gemakkelijk en geleid instappunt voor het gebruiken van de SDK, door middel van een klein projectje dat laat zien wat de SDK wel niet te bieden heeft.

Tutorials zijn het effectiefst als zij een vooropgezet project incrememnteel updaten. Hierbij is het een slim idee om een van de basiscomponenten te behandelen.

5.2 Voorbeelden

Voorbeelden zijn gemakkelijk in gebruik te nemen door gebruikers die al enkele kennis hebben van het platform. Deze developers kunnen dan zonder tutorial aan de slag gaan.

Daarnaast zijn voorbeelden goed om aan te geven welke manieren om iets te doen de voorkeur hebben. In feite wordt er een best-practice cursus gegeven over het platform.

5.3 Documentatie

Bovenstaande is op zich al een documentatie. Voor iedere beschikbare functie is echter gedetailleerde documentatie nodig. Dit heeft vooral te amekn met return-types, parameter informatie en exceptions die voor kunnen komen.

Als deze samen in een grote, doorzoekbare wiki zitten word navigatie gemakkelijk gemaakt en hoeft er minder support geleverd te worden.

6 Hoe worden componenten gepubliceerd

Om deze componenten bruikbaar te maken moeten het voor de eindgebruiker makkelijk en veilig worden gemaakt om een custom component te gebruiken in het eigen project. Als hier iets fout aan is zullen gebruikers minder geneigd zijn om externe componenten te gebruiken.

Om hier overheen te komen zal er dus een algemene oplossing moeten komen voor het verifieëren en verspreiden van componenten. door middel van de volgende punten kan dit een stuk simpeler worden gemaakt.

6.1 Veiligheid

6.1.1 Pre-release checks

Voor het waarborgen van kwaliteit kan er, voordat een pakketje toe word gestaan, een pre-release audit uitgevoerd. Hiermee kan er gekeken worden dat een component voldoet aan bepaalde eisen[1] en dat het geen malicious payloads probeert te downloaden.

Na het passeren van deze checks zou het component gepubliceerd worden. Dit betekent dat er een quality control is over de componenten die online worden gezet.

6.1.2 Signing

Signing is een manier om de integriteit van een pakketje te waarborgen. Door middel van een een digitale handtekeningen, die veriefieërbaar is door anderen.

De gemakkelijkste manier om te kijken of een pakketje afwijkt van de bron is met een *hash* van de file contents. De contents van een plugin worden door een hashing-algoritme getrokken[2]. Hier komt vervolgens een hash uit, die de contents van de plugin beschrijft.

Als een gebruiker deze hash zelf uitvoert zal hij op hetzelfde getal uitkomen als de originele hash. De orginele hash kan dus vervolgens online gezet worden, waarna een gebruiker zijn hash kan vergelijken met de online beschikbare versie. Bij een mismatch

[1] Geen crashes, material UI en meer
[2] One-way hash met SHA

zijn er dus[3] andere bestanden geleverd dan origineel bedoeld was.

Een aanvaller kan echter de online hash aanpassen om deze overeen te laten komen met de nu aangepaste componenten. Dit betekent dat de zekerheid niet absoluut is.

Een andere vorm kan komen door middel van *Digital Signing*[4]. Hiermee met de private key een handtekening gezet en gepubliceerd. Een gebruiker kan vervolgens met de public key verifieëren dat de huidige handtekening en hash klopt.[1]

6.1.3 Hosting

Componenten moeten ergens opgeslagen en geladen worden. De vraag hierbij is waar de verantwoordelijkheid moet liggen.

Als eerste kan alles gehost worden vanaf Appsemble zelf. Dit betekent dat het laden altijd vanuit het bedrijf komt en er extra checks uitgevoerd kunnen worden. Daarnaast is het een gateway, waardoor kwaliteit gewaarborgd kan worden. Het nadeel is echter dat een hoop plugins veel plek in beslag zal nemen en appsenble deze dus moet hosten.
De bestanden kunnen echter ook direct van externe bronnen geladen worden. Hierdoor worden component-developers zelf verantwoordelijk voor het hosten van een plugin. Hierbij kunnen dus echter geen checks worden uitgevoerd. Daarnaast vergrooot het de "barrier to entry".[5]

Een manier om dit te verspreiden is door JSON configs in een git repository of het inleveren van een kopie om up te loaden.

6.2 Verspreiding

6.2.1 Marktplaats

Keer en keer opnieuw is bewezen dat het hebben van een online marktplaats de integratie bevorderd[6]. Developers kunnen hiermee op een redelijk makkelijke manier gevonden worden, terwijl gebruikers met veel gemak kunnen vinden wat zij nodig hebben.

Een marktplaats is simpelweg een index van de gepubliceerde pakketjes. Een gebruiker kan hiermee simpelweg een component toevoegen. Om ervoor te zorgen dat de marktplaats makkelijk te gebruiken is zijn de volgende componenten aangeraden:

- Voorpagina met geselecteerde componenten

[3] in transit
[4] Het digitaal ondertekenen van bestanden door middel van een PGP private-public-keypair
[5] Drempel om zelf een component te ontwikkelen en te verspreiden
[6] Zie Google Play, Apple App Store, Wordpress themes en meer

- Tags voor componenten
- Een rating-systeem
- Sorteren van elementen (Op basis van rating, downloads, datum, etc)
- Zoekfunctie (Doorzoeken via titel en tags)

6.2.2 Laden tijdens runtime

Deze componenten moeten geladen worden voordat zij gebruikt kunnen worden. Er zijn twee grote manieren om componenten te laden op het web.

Als eerste is het mogelijk om ieder beschikbaar component te laden tijdens runtime. Dit betekent echter dat de paginagrootte snel in de vele megabytes zou komen. Het zou er wel voor zorgen dat, zodra een pagina geladen is, de applicatie harstikke snel is.

De andere mogelijkheid is het gebruiken van lazy loading, waarbij de componenten pas geladen worden als zij nodig zijn. Dit betekent dat iedere app een lijst van eigen componenten moet gaan laden. bij het opstarten van een app zal deze vervolgens zijn eigen resources injecteren, om zo ervoor te zorgen dat de aanwezige source files aanwezig zijn.

De app moet hiervoor wel zelf weten welke componenten geladen moeten worden. Dit kan via document.write('source')

7 Conclusies

Uiteindelijk zal er toch functionaliteit geimplementeerd moeten worden. Aan de hand van dit onderzoek kunnen we de te implementeren onderdelen in 2 categorieen onderverdelen.

7.1 Implementatie

Om te zorgen dat developers nette applicaties kunnen ontwikkelen zal de api bepaalde functies uit moeten kunnen voeren, waaronder het opslaan van data, dorogeven van requests en interfacing aan te bieden met appsemble zelf.

Hierbij is het het belangrijkste dat de api data op kan slaan en deze later op kan halen. Dit zorgt ervoor dat de basis van de implementatie er ligt.

Daarna is het belangrijk dat er een interface bestaat tussen appsemble en de extensions. Dit betekent dat appsemble hooks moet kunnen versturen die een extension kan interpeteren.

Het is ook belangrijk dat de extension bepaalde metadata over de applicatie kan opvragen. Hieronder vallen simpele dingen, zoals de naam van de app, welk kleurenschema gebruikt word en wat voor versie het is.

De rest van de functionaliteit is fijn om te hebben, maar is niet essentieel voor het maken van een functionerende sdk binnen appsemble.

7.2 Documentatie

De andere helft van een sdk is de documentatie die nodig is om deze te begrijpen. Dit kan bestaan uit verschillende onderdelen waar een gebruiker mee aan de slag kan.

Hieronder vallen vooral voorbeelden. Zij geven aan hoe een functie aangeroepen moet worden en wat ervan verwacht kan worden.

Hierbovenop is het handig om documentatie te hebben voor de individuele onderdelen van de code. Dit betekent dat een functie zichzelf moet documenteren. De beste manier om dit aan te pakken is via javadocs en zelf-beschrijvende functies.

Bibliography

[1] Abhi Shelat Rafael Pass. *A Course in Cryptography.*

[2] Wikipedia. Java version history.