**Native apps**

Om apps zo natuurlijk mogelijk te integreren in het besturingssysteem kunnen ontwikkelaars zogenoemde ‘native apps’ ontwikkelen. Dit houdt in dat de app volledig is toegespitst op het besturingssysteem waarin het zich bevindt, in tegenstelling tot HTML5-apps. De gebruikersomgeving is hetzelfde als die van de standaardapps in het besturingssysteem, en de gebruikersinteractie verloopt volgens het vaste patroon van het besturingssysteem. Echter, zoals bij elke techniek, zijn er voor- en nadelen aan het ontwerpen van een native app.

**Technieken**

Een groot nadeel van het ontwikkelen van een native app is dat de code slechts op één besturingssysteem is toegespitst, wat natuurlijk tot gevolg heeft dat deze op andere besturingssystemen niet goed werkt. Men kan de code herschrijven voor andere besturingssystemen, maar ook zogenoemde frameworks gebruiken. Met deze frameworks kan men voor meerdere platformen native apps ontwikkelen met één en dezelfde code. Voorbeelden hiervan zijn PhoneGap en Appcelerator Titanium. In deze paragraaf wordt dieper ingegaan op Titanium.

Titanium is, zoals hierboven genoemd, een framework om native apps te ontwikkelen. Het doel van Titanium is om zoveel mogelijk code te kunnen hergebruiken op verschillende platforms, waarna de code slechts ‘gepolijst’ hoeft te worden om net gepresenteerd te worden in specifieke besturingssystemen.

Titanium bestaat uit verschillende onderdelen:

Titanium SDK: de Software Development Kit zorgt ervoor dat de code van de applicatie juist geïnterpreteerd wordt. De Titanium SDK-tools werken met de SDK-tools van het besturingssysteem, waardoor de app als native app weergegeven wordt.

Titanium API: deze API voor JavaScript geeft toegang tot heel veel verschillende componenten waarmee een app ‘native’ ontwikkeld kan worden, zoals het maken van een knop in de stijl van het besturingssysteem of toegang tot de camera van de telefoon. In detail:

Stel dat we een knop willen toevoegen aan een scherm die zowel in Android als in iOS er ‘native’ uit moet zien. Deze knop moet een scherm openen waarin de gebruiker een tijdwaarde (voor bijvoorbeeld een wekker) kan selecteren en deze waarde kan opslaan met een ‘OK’-knop. Om dit te realiseren moeten we verschillende componenten uit de Titanium-API aanspreken:

-We beginnen met het declareren van een variabele ‘button’. Dit is een Titanium-button: hij ziet er ‘native’ uit. We voegen hem toe aan het scherm:

var window = Titanium.UI.createWindow();

var button = Titanium.UI.createButton();

window.add(knop);

createWindow en createButton zijn zogenoemde native methoden: deze methoden creëren een native object en een ‘proxy-object’ (window of button) waarmee we de eigenschappen en methoden van het native object kan aanroepen. Dit kunnen we dus ook bij button doen:

button.height = 10;

button.width = 200;

button.title = ‘Kies tijd’;

-Met een EventListener kunnen we aangeven wat er moet gebeuren als de gebruiker op ‘button’ drukt:

button.addEventListener(‘click’, (e) => {

this.createPickerWindow();

});

Wat er nu nog moet gebeuren is duidelijk: we moeten de methode createPickerWindow maken. Dit doen we als volgt:

-In de methode declareren we eerst de variabelen die we nodig hebben: een zogenoemde ‘picker’ om de tijd mee te selecteren, een ‘view’ om de picker in weer te geven, en weer een window om de view in weer te geven. De view is belangrijk omdat we hiermee de picker een positie en grootte kunnen geven. Als laatst declareren we nog een ok-knop:

createPickerWindow() {

var picker = Titanium.UI.createPicker({

type: Titanium.UI.PICKER\_TYPE\_TIME

});

var pickerWindow = Titanium.UI.createWindow({

backgroundColor: 'white',

layout: 'vertical',

height: Titanium.UI.SIZE,

width: Titanium.UI.SIZE

});

var pickerView = Titanium.UI.createView({

top: 0,

height: Titanium.UI.SIZE,

zIndex: 1

});

var okButton = Titanium.UI.createButton({

title: ‘OK’,

zIndex: 2

});

}

Zoals te zien is, halen we alle componenten uit de Titanium-API. Deze componenten kunnen dan speciale waarden meekrijgen:

-‘picker’ krijgt het tijd-type mee: hiermee geeft Titanium aan dat met deze picker een tijd (en niet een datum) geselecteerd kan worden

-van ‘pickerWindow’ wordt de hoogte en breedte automatisch bepaald door Titanium: de window wordt even groot als de ruimte die de picker en de knop innemen. Hiermee kan de ‘native’ look and feel behouden blijven.

Nu we alle componenten tot onze beschikking hebben is het maken van de popup-window een fluitje van een cent:

pickerView.add(picker);

pickerWindow.add(pickerView);

pickerWindow.adD(okButton);

De ok-knop komt netjes onder de picker te staan: we hebben de picker een hogere ‘zIndex’ meegegeven dan de ok-knop. Aangezien we de popup-window een verticale structuur meegegeven hebben (layout: 'vertical'), weet Titanium dat de ok-knop verticaal onder de picker geplaatst moet worden.

De laatste stap in het proces is het maken van de EventListener voor de ok-knop, maar dit spreekt tot de verbeelding: men slaat de geselecteerde tijd op in een lokale variabelen en sluit de pickerWindow.

Bovenstaand voorbeeld is slechts een voorproefje van wat er met Titanium mogelijk is. Het voorbeeld maakt echter wel duidelijk dat Titanium uit één stuk code verschillende interfaces kan opbouwen die passen bij het besturingssysteem van de gebruiker.

Als een Titaniumapplicatie gaat runnen, wordt de Titanium-API in de code in native programmeertaal omgezet. Een JavaScript-interpreter wordt ingezet om de code uit te voeren.

Titanium Studio: met deze software kan men Titanium-apps programmeren en testen, alsmede de Titanium SDK updaten.

Titanium Modules: talloze ‘modules’ zijn ontwikkeld voor Titanium. Dit zijn uitbreidingen op de API van Titanium. Aangezien Titanium een open-sourceproject is, is iedereen vrij om zelf modules voor Titanium te ontwikkelen.

Clouddiensten: Appcelerator, de ontwikkelaar van Titanium, biedt via de cloud aan ontwikkelaars bepaalde statistieken over hun app(s) aan, beschikbaar via de API.

**Voordelen**

Het ontwikkelen van een native app brengt duidelijke voordelen met zich mee. Men kan allereerst ten volste gebruik maken van de mogelijkheden die het besturingssysteem biedt. De app heeft toegang tot native elementen van de telefoon zoals de camera en de microfoon. Ook kunnen native apps via push notifications communiceren met de gebruiker. Native apps presteren vaak sneller dan HTML5-apps en zijn voorspelbaarder in hun gedrag, wat vooral het gevolg is van dat de app gebruik maakt van de UI van het besturingssysteem. Dit maakt de drempel om een app te gebruiken vaak lager.

**Nadelen**

Een van de grootste nadelen van native apps is dat ze gebonden zijn aan één besturingssysteem. Met bepaalde methoden, zoals Titanium, kan men dit probleem omzeilen, maar deze frameworks kunnen nooit 100% van alle functionaliteiten dekken. Redelijk geavanceerde applicaties hebben dan ook platformspecifieke code nodig, wat de ontwikkeling van de app moeilijker en langduriger maakt. Als een ontwikkelaar ervoor kiest om voor één besturingssysteem te ontwikkelen heeft hij bovendien twee nadelige opties: accepteren dat hij slechts een deel van de mobiele gebruikers bereikt of besluiten de app helemaal opnieuw te schrijven voor een ander systeem.

Ook is het updaten van een native app gecompliceerder dan bij een HTML5-app: de gebruiker moet de update zelf downloaden en installeren, terwijl HTML5-apps centraal geupdate kunnen worden, zonder dat de gebruiker hier iets voor hoeft te doen. Bovendien moet de ontwikkelaar voor elk besturingssysteem aparte updates uitbrengen.