HANZEHOGESCHOOL

Informatica

Afstudeer scriptie

21 juli 2014, Hoogeveen

Auteur:
Marcel Horlings
351254
Opleiding: Informatica

Stagedocent:
Jacob Mulder
Hanzehogeschool Groningen

Stagebedrijf:
Nidaros
Pascal Hakkers
Stoekeplein 1a
7902 HM, Hoogeveen

Nidaros

Hanzehogeschool Groningen
University of Applied Sciences

Woord vooraf

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nidaros

Hanzehogeschool Groningen
University of Applied Sciences

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nidaros

Hanzehogeschool Groningen
University of Applied Sciences

Inhoud

1 Inleiding

Inleiding

1.1 Nidaros

Nidaros is een bedrijf dat adviseert en ondersteuning biedt bij IT-gerelateerde bedrijfsprocessen. Het doel van Nidaros is het stimuleren van bedrijfsprocessen, het informeren van procesverantwoordelijkheden en het bewust worden van wat IT kan toevoegen aan bedrijven en organisaties. Nidaros is een klein bedrijf met ongeveer twaalf werknemers, die op veel markten in de IT bezig is. Het biedt advies op het gebied van softwareontwikkeling en geven advies op basis van testmanagement en op basis van een informatieanalyse. Nidaros maakt zelf websites en software voor klanten en voeren optimalisaties door in bestaande websites. Daarnaast doet ze ook een stuk ICT-beheer binnen bedrijven, en voeren ze reparaties van computers en iPhones uit.

Consultancy

In de Consultancy tak van Nidaros worden werknemers gedetacheerd naar andere bedrijven om daar te helpen met het inbrengen van externe systemen, en voor het testen van systemen.

Solutions

Bij Solutions worden producten verkocht aan gebruikers. Deze producten kunnen ingekocht zijn, zelf gemaakt zijn of een combinatie hiervan. Daarnaast levert Solutions ook diensten op het gebied van ICT.

Organisatiestructuur

Nidaros is een klein bedrijf met 12 werknemers in dienst.

De primaire processen

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a,

molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

1.2 Probleemstelling

Bij elke aankoop die gedaan wordt in een winkel wordt er een bonnetje meegegeven. Met dit bonnetje kan de garantie van een product verhaald worden wanneer het product het begeeft. Voor veel mensen is het bijhouden van alle bonnetjes die bij elk apparaat verkregen wordt en lastige taak. Ze vergeten wanneer hoe lang het bonnetje nog geldig is of wanneer de garantie afloopt en veel bonnetjes raken ook kwijt. Voor de mensen die hier last van hebben wordt de ScanjeBon app ontwikkeld. Met deze app kunnen de bonnetjes gemakkelijk via de smartphone opgeslagen worden. Waardoor gebruikers de bonnetjes altijd op één centrale plek hebben. Gebruikers van de app zullen ook genotificeerd worden wanneer bonnen aflopen en ze kunnen de bonnetjes overzichtelijk uit elkaar houden.

1.3 Inhoud van dit rapport

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

2 Plan van aanpak

De looptijd van het project bedraagt twintig weken. Deze twintig weken zal verdeeld worden over vier fases zoals beschreven in RUP. De eerste fase is Inception, in deze fase wordt er voor gezorgd dat iedereen betreffende het project hetzelfde beeld krijgt over het project. De tweede fase, Elaboration genaamd, wordt gebruikt als de eerste iteratie.

Zo wordt er een werkend product opgeleverd, maar worden ook dingen gedaan als het opzetten van de ontwikkelomgeving en het versiebeheersysteem. De derde fase, Construction, is waar het ontwikkelen gebeurd. Hier zal het overgrote deel van de Scanjebon app gemaakt worden. Dit wordt gedaan in iteraties en na elke iteratie wordt de app getoond aan de deelnemende stakeholders.

De laatste fase is de Transition, de Scanjebon app is hier klaar voor gebruik en kan in de markt gezet worden. Daarnaast zal hier de overdracht plaats vinden van de broncode, de documentatie en de ontwikkelomgeving. De fases zullen er samen voor zorgen dat er een Scanjebon app voor minimaal één platform gemaakt zal worden met daarnaast een ontwikkelomgeving waar een volgende ontwikkelaar mee verder kan. De planning staat als gantt chart in B.

Overdracht app en broncode.Overdracht app en broncode.

Fase 1: Inception

- Projectgoedkeuring.
- Onderzoek talen / platform.
- Onderzoek ontwikkelomgeving.

Fase 2: Elaboration

- Opzet versiebeheersysteem..
- Wireframes maken
- Bouwen app:
- Foto's maken / laden uit galerij

Fase 3: Construction

Bouwen app:

- Gegevens invullen en meesturen
- Bonnen bekijken in een lijst en apart
- Gegevens automatisch uit de foto halen d.m.v. OCR
- Ontwikkeling REST API:
- Opzetten framework
- REST infrastructuur opzetten.

Fase 4: Transition

- Advies taal platform
- Ontwikkelomgeving / ontwikkelstraat
- Overdracht app en broncode.

3 Versiebeheersysteem

Tijdens het bouwen van de applicatie is er gebruikt van het versiebheersysteem Git. Voor de stageperiode begon werd er binnen Nidaros geen gebruik gemaakt van een dergelijk systeem, daarom moest tijdens de stage uitgezocht worden wat het beste zou werken voor de stage en voor de rest van het bedrijf.

3.1 CVCS of DVCS?

Voor de keuze van van versiebeheersystemen zijn er twee opties een Centralized Version Control System (CVCS) of een Distributed Version Control System (DVCS). Voor CVCS is de bekenste optie Subversion (SVN) en voor de DVCS zijn Git en Mercurial de grootste opties. Aan beide kanten worden er versies bijgehouden van de code die gescrheven wordt en het is bij allebei mogelijk om de code te delen tussen verschillende manieren, ... Maar er is wel een verschil hoe dat gebeurd en dat verschil is ook merkbaar in het gebruik en heeft zijn voor en nadelen.

Servers

Om de code te kunnen delen bij een CVCS is er een server nodig. Op deze server worden alle versies bijgehouden, wat erg handig is want zo kan iedereen bijhouden wat er met de code gebeurd en weet iedereen waar een ander mee bezig is. Het grote probleem met een CVCS server is dat als de server stuk gaat en de data op de server verloren gaat is alle code en vooral de geschiedenis van de code ook weg. Wat dan overblijft is alle code die de mensen op dat moment lokaal op de werkplek hebben staan en geen hele geschiedenis van alle code meer. Bij het werken met CVCS-en wordt er altijd maar een deel van de code naar de locale machine gekopieerd. Wanneer de server voor een tijd offline gaat kan niemand hun code meer delen, opslaan of verder gaan met een ander stuk. Als er een aftakking van de code gemaakt wordt zodat daar nieuwe functies aan toegevoegd kunnen worden heeft de gebruiker alleen toegang tot die code. De code van de nieuwe functies die een collega op dat moment maakt blijven alleen op de server staan.

Bij DVCS is een zelfde opzet mogelijk, er kan een server neergezet worden waar alle code beschikbaar op is en waar wijzegingen bekend blijven. Net als bij de CVCS kan hier de code ingezien worden door iedereen en kan iedereen zien waar een ander mee bezig is. Mocht er bij een DVCS de server stuk gaan of offline, dan onstaat hier wel een verschil met de CVCS. Bij een DVCS staat staat op elke werkplek alle code en wijzegingen. Wijzegingen kunnen opgeslagen worden zonder verbinding met het internet, dit betekent niet alleen dat er door gewerkt kan worden tijdens het offline gaan van de server maar ook wanneer er gewerkt wordt vanuit een trein zonder internet verbinding. Het stukgaan van de server betekent bij DVCS niet dat er geen code meer gedeeld kan worden onderling, want iedereen kan als server fungeren en van iedereen kan de code opgehaald worden. Elke werkplek is hierdoor een backup van de server en van elkaar. Dit kan ook gedaan worden omdat bij een DVCS alle code opgehaald wordt van de server en niet alleen het stukje waar de ontwikkelaar op dat moment mee bezig is. Dit betekent dat als de ontwikkelaar een stuk bij wil dragen aan een stuk code waar hij niet bij betrokken was hij dit in een handomdraai kan doen

VCS -> Git

Een groot verschil tussen de eerder versiebeheersystemen en Git is hoe ze code en vooral veranderingen daarin opslaan. Bij de eerder VCS-en wordt elke keer dat er een stuk code toegevoegd wordt wordt er een nieuwe versie van dat bestand opgeslagen en de wijzeging er naast. Bij het delen van de code worden alleen wijzegingen en bestanden opgeslagen van de bestanden die dan aangepast zijn. Bij Git gaat het opslaan anders, Git kijkt bij elke toevoeging van code naar alle bestanden en onthoud van alle bestanden wat de status op dat moment was. Dit doet Git niet door de bestanden opnieuw op te slaan maar door te verwijzen naar de laatste aanpassing van dat bestand.

Branches

Elke wijzeging die toegevoegd wordt bij een VCS is een commit. Alle commits zijn stukjes code die op elkaar volgen, bij meerdere commits komt er als het ware een lijn van commits. Branches zijn aftakkingen van de reguliere lijn (master branch) aan commits. Een branch wordt gemaakt

met het geven van een naam, deze naam is een beschrijving van wat de wijzegingen gaan doen, een bug-fix of de naam van de functie die in deze branch wordt toegevoegd. De commits in deze branch zijn ook alleen hier zichtbaar en te gebruiken totdat deze branch samengevoegd wordt met de reguliere lijn aan commits, mergen genoemd.

Het maken van branches is een goede manier om mee te werken, omdat de master branch blijft werken en pas wanneer een branch voltooid is die wijzigingen erbij komen. Zo raakt de master branch niet vervuild met versies die maar half werken of nog getest moeten worden maar gebeurd dat al in de branch. Met branches kan er ook tegelijk aan een nieuwe functie van de applicatie gewerkt worden en aan een bug in de code. Wanneer er een bug gemeld wordt in de applicatie kan de ontwikkelaar naar de master branch gaan en vanaf daar een nieuwe branch aanmaken om de bug te maken. Wanneer deze bug gemaakt is kan het getest worden en daarna op de master branch gezet zodat het direct opgeleverd kan worden.

Tussen Git en oudere versiebeheersystemen is er een groot verschil in het maken en gebruiken van branches. In oudere versiebeheersystemen kost het veel tijd en moeite om een branch aan te maken, aangezien het hier vaak betekend dat alle code in de repsoitry verplaatst moet worden naar een nieuwe map. Dit kost tijd wat afhankelijk is van de grote van een repository.

In git is het maken van een branch niet meer dan het wegschrijven van 41 karakters. Doordat het maken van een branch in Git niet meer is dan een verwijzing naar een vorige commit. Dit wordt gedaan in een SHA-1 checksum van 40 karakters die bij de laatste commit hoorden en een enter. Hier hoeven dus verder geen bestanden voor gekopieerd of verplaatst te worden en hoeft er bij Git niet gewacht te worden om een branch aan te maken.

Mergen

3.2 Tools

Stash - github - bitbucket - gitlab - github enteprise

4 Ontwerp keuzes

4.1 Besturingsysteem

Voordat er begonnen kon worden aan het project moest eerst duidelijk zijn op welk platform de applicatie zou draaien. Door de grote overmacht van Android en iOS op het gebied van besturingssystemen voor de smartphone is er in het onderzoek voornamelijk hierop de nadruk gelegd. Hieronder zal uitgelegd worden waarom er voor een iOS applicatie is gekozen en wat de voor en nadelen van beide besturingssystemen zijn

Native of multiplatform

Naast alleen een applicatie op Android of op iOS is het ook mogelijk om in een keer een applicatie te maken die op beide en zelfs op meerder systemen kan werken. Hier zijn een aantal frameworks voor die dit mogelijk maken zoals Titanium, Xamarin of Phonegap.

Alle drie claimen ze dat er door maar een taal te gebruiken een applicatie gebowd kan worden die op alle grote mobiele platformen gedraaid kan worden. Bij Xamarin claimen ze dat door de applicatie te compileren de code net zo snel gaat als wanneer het native gebouwd zou worden. Bij Titanium doen ze hier nog een schepje bovenop en zeggen ze dat de code sneller uitgevoerd wordt dan bij een native app.

Phonegap is een heel ander verhaal. Ze claimen niet dat het sneller of net zo snel gaat als een native app. Dit komt doordat ze applicaties laten maken door middel van HTML5, CSS3 en JavaScript. JavaScript is de grote boosdoener voor de snelheid op mobiele applicaties. Dit komt door de hardware die gebruikt wordt op telefoons en de garbage collector van JavaScript.

Door het gebruik van HTML5 en CSS3 in Phonegap heeft de app niet de style die een native app zal hebben, waar de terug knop of de menu knop van android zit ten opzichte van iOS is een heel verschil en bij Phonegap wordt ook dat in een keer gebouwd. Dit is wel aan te passen voor beide, maar dan zijn er toch weer twee verschillende projecten waar aan gewerkt wordt.

Met de snelheid van JavaScript op mobiele apparaten blijft naast de uitstraling op de app ook het gevoel achter. Bij het wisselen tussen pagina's in de app zelf merkt de gebruiker dat het net allemaal even langzamer gaat en bij het drukken op een knop is er niet het gevoel dat er direct wat gebeurd. Voor een app is dit funest voor de gebruikerservaring, zeker wanneer de gebruiker het gevoel heeft dat er alleen maar simpele dingen gebeuren in de app. Het is ook niet voor niks dat grote bedrijven die erg willen inzetten op een mobiele app, zoals facebook, misschien wel zijn begonnen aan JavaScript, maar er toch vanaf zijn gestapt.

Xamarin en Titanium claimen dat ze de uitstraling geven van native apps en net zo snel werkt, omdat het gecompileerd is. Daarnaast geven ze aan dat tijdens het maken van de app de API van de het platform aangesproken kan worden, daardoor is het mogelijk om hetzelfde te kunnen doen wat native apps ook kunnen. In Xamarin wordt ontwikkeld met C# en wordt naar elk platform apart gecompileerd. Terwijl bij Titanium JavaScript code gebruikt wordt. Wat titanium met deze code doet voordat het als een app op de smartphone terecht komt is niet duidelijk, mocht het JavaScript blijven zal er altijd snelheidsverlies blijven zoals ook het geval is bij phonegap.

In alle gevallen moeten de ontwikkelaar rekening houden met elk platform waar de app op moet komen. Van elk device moet de taal bekend zijn en hoe de look en feel van elke app moet zijn. Voor elk platform is zijn hier aparte richtlijnen opgesteld die er voor zorgen dat de app intuitief aanvoelen op het desbetreffende platform. Hierdoor moeten er met multiplatform frameworks altijd n+1 talen en systemen geleerd worden. Waar 'n' voor het aantal platforms staat en de '+1' het framework betekend. Door het werken op een ander framework betekent het ook dat er nog een onderdeel bij komt waar ook bugs zit. Deze frameworks Bedienen ook minder mensen dan de Android SDK of de iOS SDK en bij problemen binnen het framework zijn er daarom een stuk minder problemen uitgelegd en bij een vraag zijn er minder mensen die daarmee kunnen helpen.

Android

Android heeft het grootste marktaandeel op het gebied van smartphones. Maarliefst 80% van de mobiele apparaten werkt op Android. Hierdoor

kan bij het maken van een native app op Android kan erdirect een groot gedeelte van de mensen bereikt worden. Wel moet er rekening gehouden worden met de verschillende devices. Er zijn veel verschillende soorten devices waar Android het besturingsysteem van is en omdat het op het ene aparaat goed werkt betekent nog niet dat het op een ander apparaat met Android ook goed zal werken. Op alle mobiele apparaten werkt alles net even anders en moet voor veel dingen die gebeuren in de app aparte code geschreven worden.

Het werken aan een Android project is relatief goedkoop, er moet 25 euro betaald worden bij het inschrijven van een app in de playstore. Het ontwikkelen kan in principe op alle computers. Er hoeft dus geen extra hardware aangeschaft te worden, tenzij het op een specefiek apparaat getest moet worden. De taal om applicaties te schrijven voor Android is Java. Eeen voordeel hiervan is dat veel mensen bekend zijn met de taal en veel mensen met weinig moeite en scholing kunnen beginnen aan het ontwikkelen van een taal in Java.

iOS

iOS is na Android de grootste speler op de markt, maar heeft slechts 13% van de markt in handen. iOS draait alleen op de apparaten die van Apple zelf zijn waardoor er een naadloze integratie is tussen de hardware en software op alle apparaten. Hierdoor zijn applicaties die geschreven worden voor het ene apparaat ook goed voor het andere apparaat. Er zijn specefieke gevallen waar dit minder is, maar met de simulator die Apple biedt is de code te testen voor alle apparaten, en kan er veel gedaan worden zonder een echt apparaat in bezit te hebben.

Voor het ontwikkelen in iOS is er een Mac systeem nodig waar Xcode op draait. Dit maakt de onwtikkeling duurder voor iOS wanneer er nog geen Mac aanwezig is. De applicaties worden geschreven in Objective-C een taal die op weinig andere plekken wordt gebruikt dan het maken van applicaties voor iOS en OS X. Het aantal mensen dat Objective-C kan is aanzienlijk lager dan Java, maar er zijn veel mensen die apps voor iOS en OS X maken.

Keuze

Uiteindelijk is de keuze gevallen op iOS. De oplossing van multiplatform leek weinig te beloven, zeker wanneer er in de toekomst nog functies bij moesten komen als OCR is het niet zeker of dit zou kunnen met Xamarin of Titanium. Bij Phonegap zou het al lastig worden om een aantal bonnetjes in het geheugd op te slaan door de garbage collector en het geheugen wat per applicatie gebruikt mag worden.

Voor deze applicatie moest er een goede look en feel komen waar de gebruikers makkelijk in konden stappen en snel hun weg in konden vinden. Dit zou teniet worden gedaan door het gebruik van JavaScript maar zelfs met de oplossing van Xamarin is het moeilijk om een juiste manier te vinden in waar de knoppen terecht komen en hoe de bewegingen zouden moeten gaan.

De keuze voor iOS werd gemaakt, omdat het maken van applicaties heirvoor zekerder aanvoelde dan voor Android. Bij Android mag dan wel een grotere markt zijn, maar het aantal devices waar dit goed op werkt zou net zo klein of kleiner kunnen zijn dan die van iOS. Door de verschillende grotes en hardware specefieke oplossingen van alle fabrikanten binnen Android is het lastig om app te maken, die goed gebruikt kan worden door alle Android gebruikers.

Voor Nidaros is de ScanjeBon de eerste mobiele applicatie die gebouwd wordt binnen het bedrijf. Nidaros kan hiermee zijn portfolio uitbreiden en laten zien aan de huidige klanten en toekomstige klanten. De meeste klanten en voornamelijk de leidinggevenden daarvan zijn in het bezit van een iPhone of iPad waar iOS op draait. Door de applicatie op dit besturingssysteem te bouwen denk Nidaros er uiteindelijk meer aan over te houden dan deze app. Ze kunnen de app laten zien als iets wat ze kunnen maken en daardoor meer opdrachten binnen halen.

4.2 Communicatie

4.3 Frameworks

5 Implementatie

6 Conclusie

A Persoonlijke ontwikkeling

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

B Planning