

HANZEHOGESCHOOL

Informatica

Afstudeer scriptie

Nidaros 

29 juli 2014, Hoogeveen



Auteur:

Marcel Horlings

351254

Opleiding: Informatica

Stagedocent:

Jacob Mulder

Hanzehogeschool Groningen

Stagebedrijf:

Nidaros

Pascal Hakkers

Stoekeplein 1a

7902 HM, Hoogeveen

Woord vooraf

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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.

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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.

Inhoud

1	Inleiding	1
1.1	Nidaros	1
1.2	Probleemstelling	2
1.3	Inhoud van dit rapport	2
2	Plan van aanpak	3
3	Versiebeheersysteem	5
3.1	CVCS of DVCS?	5
3.2	Tools	7
4	Ontwerp keuzes	8
4.1	Besturingsysteem	8
4.2	Communicatie	8
4.3	Frameworks	8
5	Implementatie	9
6	Conclusie	10
A	Persoonlijke ontwikkeling	11
B	Planning	12

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 software-ontwikkeling 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 consectetur.

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 gebeurt. 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 versiebeheersysteem 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 bekendste 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 gescreven wordt en het is bij allebei mogelijk om de code te delen tussen verschillende manieren, Maar er is wel een verschil hoe dat gebeurt 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 gebeurt 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 wijzigingen 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 ontstaat hier wel een verschil met de CVCS. Bij een DVCS staat op elke werkplek alle code en wijzigingen. Wijzigingen 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 wijziging er naast. Bij het delen van de code worden alleen wijzigingen 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 inhoud 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 wijziging 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 wijzigingen 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 gebeurt 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 repository verplaatst moet worden naar een nieuwe map. Dit kost tijd wat afhankelijk is van de grootte 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 enterprise

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

Keuze

4.2 Communicatie

4.3 Frameworks

Het gebruik van een framework voor de server kant maakt het werk gemakkelijker omdat in het framework al aan meerdere dingen is gedacht. Er zit een standaard authenticatie in, het framework kan al overweg met aanvragen en berichtgeving en het heeft al een structuur waar de code in gezet wordt. Het ontwikkelen met behulp van een Framework wordt een stuk gemakkelijker dan zonder. Doordat er voor veel onderdelen al standaarden zijn geïmplementeerd die daardoor niet meer zelf gemaakt hoeven te worden.

Doordat het framework ook al een structuur neerzet over het verdeelen van de code. Is het voor iedere ontwikkelaar gemakkelijk te vinden waar aanpassingen gedaan moeten worden. Wanneer een andere ontwikkelaar niet weet waar een bepaald stuk aangepast moet worden kan dit gemakkelijk opgezocht worden in de documentatie van het framework.

Vanuit Nidaros kwam de vraag of er aan de server kant PHP gebruikt kon worden, omdat er binnen nidaros alleen hier ervaring mee was. Als er problemen waren aan de server kant konden de mensen die bij Nidaros werken de code begrijpen en daaraan werken. Daarbij kwam ook de vraag of bij het gebruik van een framework er een klein framework gebruikt kon worden, zodat er niet teveel geleerd hoefde te worden van het

framework zelf en dat de werking duidelijk was te zien aan de code die zelf geschreven is.

Met de eisen die gesteld zijn vallen er veel frameworks af. Alle frameworks in andere talen zoals Rails (Ruby), Play!(Java) en Django(Python) vielen af, omdat daarmee niet in PHP geschreven wordt. Daarnaast vallen frameworks af die wel met PHP werken, omdat deze frameworks veel groter en ingewikkelder zijn dan nodig is. Voorbeelden hiervan zijn Zend, CodeIgniter en Yii. De frameworks die overblijven zijn Slim, Laravel en Recess.

Het Slim framework is een erg minimalistisch framework. Het is gemakkelijk om mee te beginnen, omdat er weinig nodig is om het eerste resultaat te boeken. Het aanmaken van REST endpoints is erg simpel om te doen, waardoor er snel een punt is om de requests binnen te laten komen. De documentatie hiervan is goed en uitgebreid. Er wordt per request uitgelegd wat er gedaan moet worden en hoe hoe de response verstuurd moet worden.

Slim heeft geen ORM voor de database. Het handige van een goede ORM is dat het maken van queries al gedaan is en om die queries te kunnen gebruiken moet daarvoor alleen de functies aangeroepen worden binnen het framework. Deze functies zijn generiek opgezet waardoor er bij het aanmaken van nieuwe velden en of tabllen er nauwelijks iets veranderd hoeft te worden. Bij het Slim framework is dit niet het geval en blijft het handwerk om de gegevens uit de database te halen. Een andere tegenvaller van Slim is dat de structuur van de applicatie zelf uitgedacht moet worden. Waar dit een nieuw framework is waar nog in geleerd moet worden hoe het werken ermee het handigste is, kost het extra veel moeite om direct de structuur goed te hebben of later nog eens aan te passen.

5 Implementatie

6 Conclusie

A Persoonlijke ontwikkeling

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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

