

HANZEHOGESCHOOL

*Informatica*

---

## Afstudeer scriptie

---

**Nidaros** 

11 augustus 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

<b>1</b>	<b>Inleiding</b>	<b>1</b>
1.1	Nidaros . . . . .	1
1.2	Probleemstelling . . . . .	2
1.3	Inhoud van dit rapport . . . . .	2
<b>2</b>	<b>Plan van aanpak</b>	<b>3</b>
<b>3</b>	<b>Versiebeheersysteem</b>	<b>5</b>
3.1	CVCS of DVCS? . . . . .	5
3.2	Tools . . . . .	7
<b>4</b>	<b>Ontwerp keuzes</b>	<b>8</b>
4.1	Besturingsysteem . . . . .	8
4.2	Communicatie . . . . .	8
4.3	Frameworks . . . . .	10
<b>5</b>	<b>Implementatie</b>	<b>11</b>
5.1	Stash . . . . .	11
5.2	Het begin . . . . .	12
<b>6</b>	<b>Conclusie</b>	<b>14</b>
<b>A</b>	<b>Persoonlijke ontwikkeling</b>	<b>15</b>
<b>B</b>	<b>Planning</b>	<b>16</b>

# 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

De communicatie tussen de app en de server gebeurt over het internet, om de communicatie goed te laten verlopen moet er afgesproken worden wat de wijze van deze communicatie is en hoe die zal verlopen. Deze afspraken zitten gebonden in een protocol. Voor het versturen van gegevens tussen de client en de server zijn er twee bekende protocollen REST en SOAP en hieronder zal worden uitgelegd wat beiden inhouden, waarmee het verschil duidelijk wordt gemaakt tussen de protocollen. En welke keuze er uiteindelijk is gemaakt.

#### **SOAP**

Simple Object Access Protocol (SOAP) is een protocol waarmee berichten tussen systemen verstuurd worden. SOAP is bedacht voor Microsoft en wordt tegenwoordig door veel bedrijven en software oplossingen gebruikt. SOAP bestaat uit drie onderdelen, Het eerste deel beschrijft wat er in het bericht staat en hoe het verwerkt moet worden. Het tweede deel is een set met regels wat aangeeft

#### **REST**

Representational state transfer (REST) is een webarchitectuur die de basis vormt voor het world wide web (www). REST gebruikt voornamelijk JavaScript Object Notation(JSON) of XML voor de data overdracht.

REST werkt erg simpel. Het gebruikt create, read, update en delete (CRUD) voor zijn acties, via http wordt dit aangeroepen met POST, GET, PUT en DELETE. REST verstuurd zijn CRUD opdrachten naar een Uniform Resource Identifier (URI). Door de combinatie hiervan weet de server welke actie ondernomen moet worden en wat er vervolgens moet gebeuren.

Bij een GET op de URI `http://www.scanjebon.nl/receipts` worden alle bonnen verwacht waar de gebruiker bij mag. De data van deze bonnen worden dan in JSON of XML teruggestuurd. Bij een GET op de URI `http://www.scanjebon.nl/receipts/1` wordt de data van het bonnetje verwacht die als 'id' 1 heeft. Naast de data die teruggegeven wordt krijgt de client een HTTP code 200 terug

Wanneer er data opgehaald moet worden voor alle bonnetjes van een specifieke gebruiker kan dit door `http://www.scanjebon.nl/users/1/receipts`. Er wordt eerst aangegeven van welke gebruiker de data nodig is, met behulp van `/users/1` is het duidelijk dat er van de gebruiker met het id 1 data nodig is. Door `/receipts` erachter te plaatsen is het duidelijk dat van die gebruiker data van de bonnen gevraagd wordt.

Bij een POST wordt er geen data opgehaald maar opgeslagen in het framework. Vervolgens wordt de data wel teruggestuurd, omdat bij het aanmaken op de server nog het een en ander is veranderd. Zo krijgt een resource pas een id op het moment dat deze wordt aangemaakt op de server en de client heeft die id nodig om later die resource te kunnen ophalen, aanpassen of verwijderen. Het aanmaken van een resource wordt gedaan door een POST te versturen naar een URI van een resource zoals `www.scanjebon.nl/receipts`. Bij de post worden de gegevens in de vorm van JSON of XML meegestuurd. De server controleert dit en slaat het vervolgens op en stuurt diezelfde data terug met extra informatie die op de server is aangemaakt. Bij het slagen van de POST wordt de HTTP 200 code meegestuurd.

Om de data aan te passen die is opgeslagen kan een PUT uitgevoerd worden op dat object. Deze put wordt gedaan op een resource met id als `www.scanjebon.nl/receipts/1`. Met deze actie wordt via JSON of XML meegestuurd wat het huidige object moet worden. Bij een PUT wordt

dezelfde data via JSON of XML meegestuurd als bij een POST maar in tegenstelling tot een POST krijg de PUT geen data meer terug. De client zal weten dat de opdracht gelukt is wanneer er de HTTP 204 code teruggestuurd wordt.

Als laatste van de CRUD opdrachten blijft de DELETE over. De DELETE wordt uitgevoerd op een recourse met een id `www.scanjebon.nl/receipts/1` en verwijderd het object met dat id. Bij de uitvoering van een DELETE worden geen gegevens meegestuurd en krijgt dat doorgaans niet teruggestuurd. Wanneer het gelukt is krijgt de client een HTTP 200 code terug wanneer het gelukt is en er data wordt meegestuurd. Wanneer er geen data wordt teruggestuurd maar de opdracht wel gelukt is krijgt de response de 204 HTTP code terug.

## 4.3 Frameworks

## 5 Implementatie

Voor de implementatie was een planning gemaakt, deze planning staat in B omschreven. Hierin was duidelijk gemaakt welke features er wel en niet in zouden komen van te voren, maar er werd wel vanuit een agile omgeving begonnen met bouwen. De grote features van het systeem stonden van te voren al beschreven met MoSCoW rating. Deze features worden stap voor stap ingevoerd en na elkaar. Wanneer de ene feature af is kan er pas begonnen worden aan de volgende. Hierdoor blijft er ruimte open voor aanpassingen en wanneer er wijzigingen doorgevoerd moeten worden heeft het minder invloed op het geheel.

### 5.1 Stash

Als Git server is er voor Stash gekozen. Om de code bij te houden en de kunnen delen is de Git server geïnstalleerd voor het daadwerkelijk bouwen van de applicatie. Voor de stash applicatie was er ruimte op een server van Nidaros, waartoe alleen toegang toe was via ssh. Stash werd geïnstalleerd aan de hand van een handleiding die Atlassian op de website heeft staan.

Het eerste wat op de server geïnstalleerd moest worden was Git zelf en Java omdat Stash stash gemaakt is met Java en Git nodig heeft om te kunnen werken. Vervolgens kan de installer met een wget opgehaald worden. Na het ophalen van de installer kan daarin nog aangegeven worden in welke map stash geplaatst moet worden. Vervolgens wordt de installer gestart door het uit te voeren in de shell. De installer gaat nu verder op een webpagina die gehost wordt op poort 7990. Deze poort moest eerst opengezet worden in de firewall voordat de pagina bereikt kon worden.

Op de webpagina volgen een aantal instructies met welke database er gekozen moet worden en het maken van een admin account. Vervolgens wordt Stash geïnstalleerd aan de hand van de gekozen instellingen. Daarna blijft Stash bereikbaar op de poort 7990. Na het installeren van stash is het direct verbonden met Jira. Jira is een software pakket dat



net als Stash gemaakt wordt door Atlassian. Met Jira kunnen er issues en user stories bijgehouden worden. door de koppeling tussen Jira en Stash zijn de gebruikersnamen van stash dezelfde gebruikersnamen die in Jira gebruikt worden. Deze issues en user stories die in Jira worden aangemaakt kunnen ook gekoppeld worden aan de branches en commits die gemaakt worden in de stash repositories. Hierdoor kan er in Jira niet alleen bijgehouden worden welke issues en user stories er aangemaakt zijn, maar kan er daarnaast gezien worden in Jira hoeveel er momenteel gedaan is aan die issues en user stories.

Na het installeren van Stash, het verbinden met Jira en vervolgens het inloggen kan er een project aangemaakt worden. Dit project heet ScanjeBon en in het project staan twee repositories iPhone en Laravel. Voor zowel het Laravel als het iPhone project werd vervolgens een verwijzing gemaakt naar de betreffende repository als origin. Hiermee wordt elke keer als code gepushed wordt naar de server de juiste server bereikt en de juiste repository in die server. Dit wordt gedaan met `git remote add origin https://username@stash/scm/ScanjeBon/iPhone.git` waarbij username staat voor de gebruikersnaam en stash voor het ip en de poort waar stash zich bevind.

Na het maken van deze instellingen is Git geconfigureerd, kan iedereen bij Stash, omdat Jira al geïnstalleerd was binnen Nidaros en iedereen die dat nodig had daar al een account had.

## 5.2 Het begin

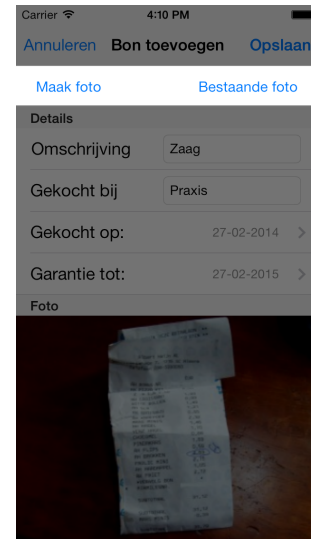
Het maken van de app was in het begin gedaan aan de hand colleges gegeven door Paul Hegarty deze colleges staan op videos online en werden gegeven op de universiteit van Stanford in de eerste colleges werd een basis gelegd om de infrastructuur van iOS te begrijpen. Xcode de IDE waarmee ontwikkeld wordt voor iOS werd uitgelegd en de taal Objective-C werd uitgelegd. In de colleges die daarop volgden werd Xcode er opnieuw bij gepakt om te beginnen met het ontwikkelen. Hoe nieuwe schermen aangemaakt kunnen worden en hoe vervolgens knoppen daarop werken en hoe aan die knoppen acties gehangen kunnen worden.

Door het volgen van deze colleges werd snel een beeld geschept over hoe dingen werken in iOS en hoe dingen gemaakt kunnen worden. De

basis was snel gelegd en er werd uitgelegd hoe met de iPhone foto's gemaakt en ingeladen kunnen worden. Er werden twee knoppen op het scherm geplaatst één met `Maak foto` en de andere met `Bestaande foto` zoals ook te zien is in figuur 5.1. Na het klikken op een van de knoppen voert de applicatie een code uit die de API van iOS aanspreekt. Hierna worden schermen getoond van iOS zelf die een foto laat maken als er op `Maak foto` is geklikt of een foto uit de mappenstructuur van de iPhone haalt als er op `Bestaande foto` is geklikt.

Zodra een foto is gemaakt of opgehaald moet het getoont worden. Als eerst wordt dit gedaan op dit scherm zelf, omdat het daarvoor nog niet opgeslagen hoeft te worden. Om dit te doen moet er een `ImageView` naar het huidige scherm gesleept worden. Deze `ImageView` wordt vervolgens verbonden aan de code waardoor er een methode voor gecreeerd wordt daarna hoeft de image alleen nog maar naar die methode gestuurd te worden en is die image zichtbaar.

De gegevens die bij de bon horen zijn: het plaatje van de bon zelf, de datum waarop het gekocht is, de datum waarop de garantie afloopt, bij welke winkel het gekocht is en de foto van de bon zelf.



Figuur 5.1: Toevoegen van een foto

## 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**

