TU Delft

TI3800 Bachelorproject

Bachelor Eindproject

Lentiz | Maasland college Staport v 2

Schrijvers:
David Festen,
Vincent Ghiëtte,
Thomas Valera

Begeleider:
Hans-Gerhard Gross

Opdrachtgever:
Rens Looij

11 Juli 2013

Inhoud

TU Delft

TI3800 Bachelorproject

Eindverslag

Lentiz | Maasland college Staport v 2

Schrijvers:
David Festen,
Vincent Ghiëtte,
Thomas Valera

Begeleider:
Hans-Gerhard Gross

Opdrachtgever:
Rens Looij

11 Juli 2013

Voorwoord

Dit is het eindverslag van bacheloreindproject aan de TU Delft. Dit project is uitgevoerd in opdracht van Lentiz | Maaslandcollege. Er is bij het schrijven van dit verslag uitgegaan van elementaire kennis op het gebied van Informatica; de non-triviale vaktermen worden wel uitgelegd. De hoofdstukken van dit rapport zijn op chronologische volgorde van het project ingedeeld. Graag bedanken wij iedereen die heeft meegewerkt aan dit project: De opdrachtgever de heer Looij, de begeleider de heer Gross en de coördinator mevrouw Larson. Dit project en bijbehorende verslag zijn in Delft, Nederland in de zomer van 2013 gemaakt door Thomas Valera, Vincent Ghiëtte, en David Festen.

Inhoudsopgave

Samenvatting

Er is in opdracht van de TU Delft faculteit Elektrotechniek, Wiskunde en Informatica (EWI) in de vorm van een Bachelor Eindproject (BEP) een stage registratiesysteem gemaakt voor het Lentiz | Maaslandcollege.
Bij de opdrachtgever is er geïnventariseerd wat de wensen waren waarna wij uitkwamen op een stage registratie webapplicatie waar de verschillende actoren op kunnen inloggen en de acties kunnen uitvoeren waarvoor ze geautoriseerd zijn. Er is een lichtgewicht ontwikkelmethode gebruikt die gebaseerd is op Scrum en als ontwikkel omgeving is ASP.NET MVC gekozen. De gebruikersinterface is opgebouwd uit een dashboard met aanklikbare tegels. De backend is op het MVC patroon gebaseerd en in combinatie met Entity Framework en een database geïmplementeerd. Verder zijn er Unit Tests en Integration Tests geschreven.
Hoewel het project goed en vlot verlopen is, is het niet gelukt om alle features te implementeren. Het programma is wel volledig functioneel, maar de extra features voegen een wat eenvoudigere werkwijze toe op sommige gebieden.
De projectgroep heeft veel geleerd van dit project en het project is in zijn opzet geslaagd.

1 Inleiding

Dit is het eindverslag van het bacheloreindproject bij het Lentiz | Maaslandcollege. Tijdens het project is er een stage registratie systeem gemaakt ter vervanging van het huidig systeem. Het huidig systeem bestaat uit drie losse systemen, die verder niet centraal bijgehouden worden. Er is een boekje waarin de voortgang van de leerling bijgehouden wordt. Verder is er een spreadsheet met bedrijven en een andere met de informatie over de leerlingen. Het voorgesteld systeem is een webapplicatie dat de drie lossen systemen met elkaar koppelt. In dit verslag komen aan bod de werkwijzen, gebruikte tools en het algemene ontwikkelproces van de applicatie. Allereerst zal er ingegaan worden op het plan van aanpak van het project. Vervolgens zal de ontwikkel methodologie geanalyseerd worden. Hierna zal het ontwerp en de implementatie van de software besproken worden om uiteindelijk het gebruikte testing proces te beschrijven. Daarna zal een korte reflectie op het project gegeven worden waarna de conclusie zal volgen. Dit rapport zal afgesloten worden met aanbevelingen omtrent verbeterpunten van de webapplicatie.

2 Plan van aanpak

In dit hoofdstuk is het plan van aanpak kort weergegeven. Het plan van aanpak bestaat ook nog als los document en voor meer informatie verwijzen we dan ook naar dat document. In dit hoofdstuk zullen we het document kort samenvatten.

Wij beschrijven in de volgende paar paragrafen hoe het project, een stage registratie systeem voor het Maaslandcollege, gemaakt zal worden. De belangrijkste eis die aan dit project gesteld word vanuit het Maaslandcollege is dat het stageverloop van de leerlingen gevolgd kan worden op een gecentraliseerde en geautomatiseerde manier. Dit project zal in elf weken gerealiseerd worden. Tijdens het project zal een systeem gemaakt worden dat aan de eisen van de opdrachtgever voldoet en conform is aan de uiteindelijke doelstelling. Om dit te waarborgen zal er nauw samengewerkt worden met de opdrachtgever. Daarbij zal er gebruik gemaakt worden van Agile ontwikkeltechnieken.
De opdracht van dit project komt van Lentiz | Maaslandcollege. Lentiz is een scholengemeenschap van vmbo en mbo scholen. Lentiz heeft ons gevraagd om een digitaal stage registratie systeem te maken. Op dit moment gebruiken ze daarvoor een papieren logboek en een verzameling van Excel spreadsheets. De huidige werkwijze is decentraal en dat zorgt voor de nodige problemen. Dit is tevens een aanleiding voor het uitvoeren van deze opdracht. De recente invoering van het competentie gericht onderwijs in het VMBO is de andere aanleiding.

2.1 Projectomgeving

De huidige werkwijze heeft enige beperkingen: Het bestaat uit drie losse systemen, die verder niet centraal worden bijgehouden.
Allereerst is er het logboek, dit wordt beheerd door de stagiair zelf. Deze neemt het logboek mee naar het stage bedrijf en naar school. Dit heeft twee doelen, namelijk het enerzijds mogelijk maken van communicatie tussen de school en het stagebedrijf, en het anderzijds bijhouden van de voortgang van de desbetreffende stagiair.
Ten tweede is er Excel spreadsheet waarin bijgehouden wordt welke stagiair welke stages doen. In deze spreadsheet worden de volgende gegevens bijgehouden: de gegevens van de stagiair, stagebedrijf, stagebezoeker (beoordelend docent), stagecoördinator en mentor. Het stagebedrijf bied een werkplek aan de stagiair om werkervaring op te doen. Tijdens de stageperiode controleert de stagebezoeker de voortgang van de stagiair. De stagecoördinator beheert de stages van alle stagiairs. De mentor heeft weinig met de stage te maken en wordt vooral ter referentie genoemd in de spreadsheet. In het huidig systeem is er één spreadsheet voor alle stagiairs.
Ten derde hangt er een lijst in de school waarop suggesties staan voor mogelijke stageadressen. De leerlingen kunnen daar een stagebedrijf uitzoeken en zich voor een stage opgeven bij de coördinator. Deze lijst bevat slechts suggesties voor stagebedrijven en wordt in een spreadsheet bijgehouden.

2.2 Probleemomschrijving

De huidige situatie levert problemen op. Door het gebruik van een logboek kan er gemakkelijk gefraudeerd worden. Ook kan dit kwijtgeraakt worden door de stagiair, stagebedrijf of stagecoördinator. Verder wordt het logboek aan het einde van de stage gearchiveerd en gedigitaliseerd wat voor overhead zorgt.
Het gebruik van een spreadsheet voor het bijhouden van welke stagiairs naar welke stage gaan levert ook wat problemen op. Zo is het mogelijk dat er meerdere versies van deze spreadsheet bestaan, er is namelijk geen centrale opslag. Dit zorgt ervoor dat deze spreadsheet snel onoverzichtelijk wordt.
De lijst met stageadressen op school biedt geen eenvoudige mogelijkheid voor de stagebedrijven om stages aan te bieden. Verder kunnen leerlingen op deze lijst geen gedetailleerde informatie vinden.
Tot slot is er nog een overkoepelend probleem: Er zijn drie aparte informatiesystemen nodig om stages te regelen. Hierdoor is er geen centrale plek waar alle informatie te vinden is, omdat de verschillende bronnen niet aan elkaar gekoppeld zijn. Als bijvoorbeeld het email-adres van de contactpersoon van de stagebieder gewijzigd wordt in de spreadsheet, dan is dit niet zichtbaar op de lijst die in de schoolgang hangt.

2.3 Doelstelling

De opdrachtgever heeft deze opdracht gegeven, omdat hij het huidige proces wil verbeteren. Het huidige proces is inefficiënt en bevat verbeteringsmogelijkheden. Door deze door te voeren wordt het proces goedkoper, toegankelijker, veiliger en groener.

De kernwoorden van de door te voeren verbeteringen zijn: centralisering & digitalisering, waarbij onze nadruk op digitalisering zal liggen. De centralisatie van het systeem zorgt ervoor dat het opzoeken van gegevens minder tijd in beslag neemt en zorgt dit ervoor dat de gegevens toegankelijker zijn. In plaats van drie losse systemen te raadplegen, is het na centralisatie mogelijk om in één systeem alle gegevens te vinden. De digitalisering heeft meerdere voordelen. Zo zorgt het voor gemakkelijke archiveringsmogelijkheden die ruimte efficiënt zijn; er hoeft immers alleen een digitaal archief aangelegd te worden. Ook zorgt digitalisering voor een milieubewuster imago omdat dit het gebruik van papier terug dringt. Door digitalisering wordt het proces efficiënter wat een tijd- en kostenbesparing met zich meebrengt. Het proces is veiliger, omdat de actoren na authentificatie een gelimiteerd aantal handelingen kunnen uitvoeren. Hierdoor is het bijvoorbeeld niet meer mogelijk dat een leerling een handtekening vervalst. Tot slot voegt digitalisering ook nieuwe mogelijkheden aan het proces toe. Zo is bijvoorbeeld informatie altijd en overal beschikbaar. Ook is de informatie redundant opgeslagen en daardoor minder gevoelig voor verlies.

2.4 Opdrachtformulering

De opdracht luidt als volgt: “Ontwikkel een stage-registratie applicatie”. Wij zullen zoals in het volgende hoofdstuk besproken wordt een Agile ontwikkelmethode toepassen genaamd ”Scrum”. Deze methode voorziet in de verantwoordelijkheden van het

ontwikkelteam en de opdrachtgever door gebruik te maken van “Backlogs”, dit wordt besproken in het volgende hoofdstuk. Er is afgesproken met de opdrachtgever om de voortgang tweewekelijks te bespreken en eventueel aanpassingen te maken.

2.5 Op te leveren

Een digitaal stage registratie systeem ter vervanging van het huidige systeem. De specificaties van dit systeem staan gedetailleerd omschreven in de “Product backlog” dat te vinden is in het losse plan van aanpak.

2.6 Aanpak

Wij gebruiken een Agile ontwikkelmethode genaamd Scrum. Daardoor is onze ontwikkeltijd opgedeeld in zogeheten “Sprints”. Deze Sprints laten wij een week duren, dit omdat wij dan een natuurlijk verloop in de week hebben en verwachten dat er geen of weinig Backlogs items zijn die langer dan een week aan tijd kosten. Omdat het project elf weken loopt zullen er dus elf sprints zijn.
Tijdens de ontwikkeling zal de opdrachtgever geen technische aspecten of beslissing voorgelegd krijgen. Deze zullen we besproken worden met de coördinator van de TU Delft en eventueel met de Software Improvement Group (SIG). De opdrachtgever wordt wel betrokken bij het maken van functionele beslissingen.
Wegens de gebruikte ontwikkelmethode is het noodzakelijk dat de opdrachtgever nauw betrokken is bij het ontwikkelen van de software. De opdrachtgever heeft met het ontwikkelteam afgesproken om een tweewekelijkse vergadering te houden. Daarnaast is het wenselijk om in de andere weken ook feedback te krijgen door middel van een email.
Deze feedback zal voor de nieuwe sprint gegeven moeten worden (dat betekent dus vrijdag of in het weekend, met eventueel donderdag bij verhindering).
Scrum, de gebruikte ontwikkelmethode, werkt met Backlogs. Een Backlog is een verzameling van geïsoleerde functionaliteiten. Hieronder zal de Product backlog worden opgesomd. De Product backlog bevat alle “would have”’s van het product. Uit deze Product backlog wordt later de Release backlog gemaakt. Deze “Release backlog” bevat alle “must have”’s; alles wat het programma moet hebben om uitgebracht te worden. In dit specifieke geval zijn de Product backlog en Release backlog gelijk aan elkaar. De release Backlog wordt verder onderverdeeld in “Sprint backlogs” (elf stuks voor deze release). Elke van deze Sprint backlogs moet af zijn in de vastgestelde tijd: namelijk een week. Lukt dit niet of houden we veel tijd over, dan moet het proces herzien worden.

2.7 Projectinrichting

2.7.1 Administratieve procedures

De voortgang van het project zal door vier partijen gecontroleerd worden. De project manager zal namens de projectgroep de tijdsbewaking doen door middel van zogeheten

“Burndown charts”. Hiermee kan worden bijgehouden of het project volgens planning verloopt. De opdrachtgever zal bijsturing geven bij de wekelijkse feedback ronde. De project coördinator zal ingrijpen als het project dreigt fout te lopen. De SIG controleert de kwaliteit van de code en geeft feedback in de vorm van een rapport en een code kwaliteitscijfer.

2.7.2 Financiering

In de begroting van het Maaslandcollege is er geen ruimte voor het vergoeden van de besteedde uren en secundaire kosten van de projectleden. De TU Delft kan eventueel in software voorzien. De groepsleden zullen zelf de nodige hardware voor de ontwikkeling moeten voorzien.

2.7.3 Rapportering

Onze communicatie met de opdrachtgever zal verlopen via de feedback momenten aan het einde van iedere sprint. Tweewekelijks is er bijeenkomst en de andere weken gaat het contact via de mail.

2.8 Kwaliteitsborging

De opdrachtgever is nauw betrokken bij het ontwikkelingstraject en kan de kwaliteit in een vroeg stadium al beoordelen, daardoor zijn er veel mogelijkheden tot bijsturing. Om bij te dragen aan de kwaliteit van het product kan de opdrachtgever duidelijk aangeven welke onderdelen belangrijk voor hem zijn en welke bij een overschrijding van de ontwikkeltijd kunnen vervallen.
De projectgroep zal de kwaliteit zelf echter ook in de gaten houden. Dit gebeurt door gebruik te maken van de Scrum ontwikkelmethode, welke veel ruimte laat voor verbetering en bijsturing. Dit in tegenstelling tot de waterval methode, waarbij het resultaat pas aan het einde zichtbaar is. Ook zullen er “test cases” en “use cases” opgesteld worden om de code zoveel mogelijk aan de eisen te laten voldoen. Verder adopteren wij een “MVC” patroon zodat de code makkelijk te onderhouden is en modulair uitgebreid kan worden. SIG zal de code op twee momenten beoordelen op kwaliteit.

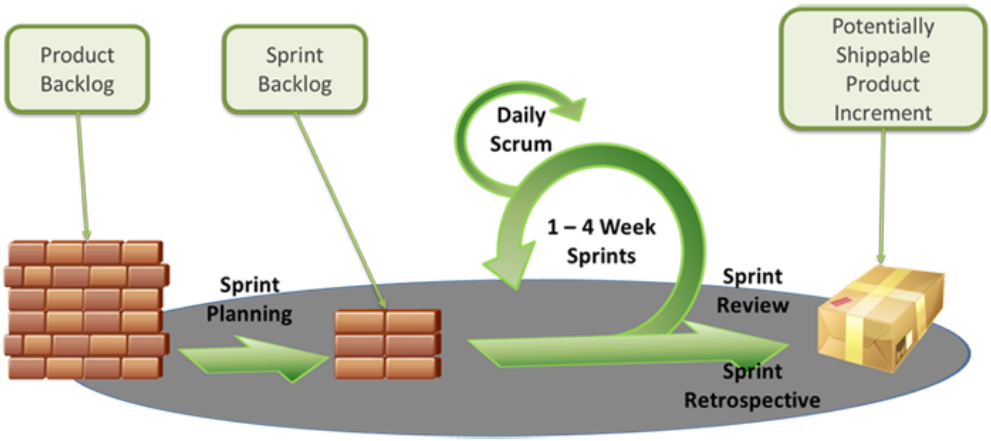
3 Methodologie

3.1 Process strategie na het onderzoek

Tijdens de oriëntatiefasen zijn er twee ontwikkel methoden naar voren gekomen, de Waterval ontwikkelmethode en de Scrum ontwikkelmethode. De ontwikkelmethoden zijn in die fase bestudeerd en met elkaar vergeleken. Na aanleiding van het onderzoek is er besloten om de Scrum ontwikkelmethode te gebruiken.

3.1.1 Gekozen ontwikkelstrategie

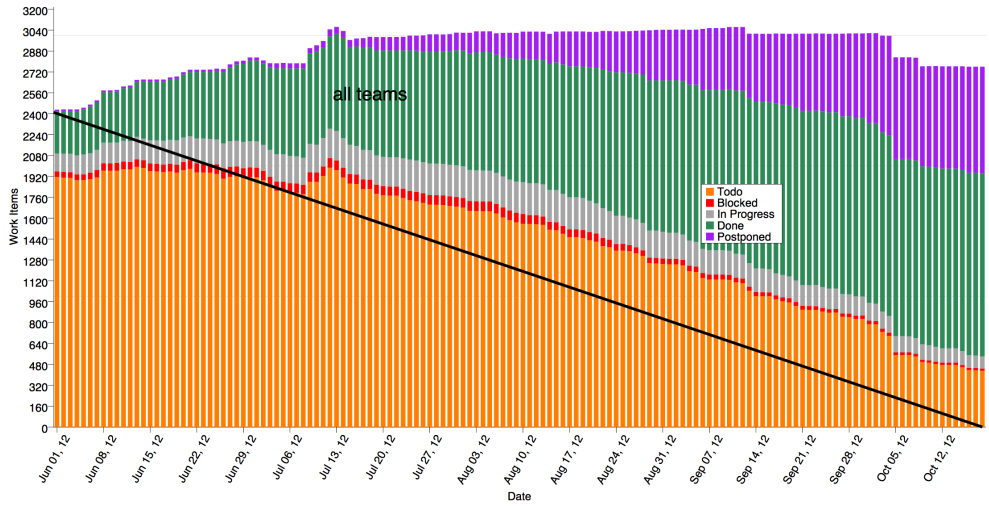
Er zijn twee redenen die ons aanspreken in de Scrum ontwikkelmethode en zo bijgedragen hebben aan het kiezen van deze ontwikkelmethode.
Vooraf aan de uitleg voor het kiezen van de Scrum ontwikkelmethode (zie Figuur 1) zal er in het kort uitgelegd worden wat de Scrum ontwikkelmethode inhoud. Bij Scrum word een lijst van features opgesteld (Product backlog). Hieruit worden een aantal features geselecteerd voor de huidige iteratie van het product (Sprint backlog). Per sprint worden er voor de gekozen features het ontwerp, implementatie, verificatie en integratie gedaan. Waarna er een werken product is. Na deze sprint voltooid is word er een andere sprint gedefinieerd. Dit proces herhaalt zich totdat alle features uit de Product backlog geïmplementeerd zijn. Bij een grote Product backlog kan ervoor gekozen worden om nog een Release backlog te maken met een subset aan features die af moeten zijn voor een bepaalde release. Bij een klein project zijn de Product en Release backlog gelijk aan elkaar. [1]



Figuur 1: Scrum ontwikkelmethode

Allereerst sprak de werkwijze van het scrum ontwikkel model ons aan. Door het gebruik van de backlog wordt het hele project gedefinieerd. Dit zorgt voor een duidelijk overzicht van alle taken die vervult moeten worden voor het voltooien van de applicatie.

Om beter grip te krijgen op de voortgang hebben we ook gebruik gemaakt van een burndown chart, zie Figuur 2. Deze chart maakt het mogelijk om op een overzichtelijke manier de voortgang van het project te monitoren. Ook bied het de mogelijkheid om eventuele vertraging van het project in een vroeg stadium op te merken. Zodoende bied het de mogelijkheid om tijdig in te grijpen om verder oponthoud te voorkomen.



Figuur 2: Burndown chart van Ubuntu 12.10

De burndown chart is niet de enige manier om de voortgang van het project te monitoren, zo is er ook elke dag een standing meeting voorzien. Deze korte meetings, vijf a tien minuten, worden staand gehouden aan het begin van elke dag. De korte duur van de meetings en het feit dat ze staande gehouden worden zorgen ervoor dat de besproken onderwerpen op een directe en duidelijke manier gebracht worden. Deze meetings zorgen er voor dat de projectleden feedback kunnen leveren om zo elkaar te updaten over de voortgang van het project.
Ten tweede is de Scrum methode interessant voor de opdrachtgever. Door het definiëren van sprints bied de Scrum ontwikkelmethode de mogelijkheid aan om de verschillende onderdelen van de applicatie direct te integreren in het eindproduct. Dit zorgt ervoor dat aan het eind van elke sprint de applicatie uitgebreid wordt met extra functionaliteit wat weer aan de opdrachtgever getoond kan worden in de vorm van een demo.
Door het hoge feedback niveau naar zowel de projectleden onderling als naar de opdrachtgever en het makkelijk kunnen traceren van de vooruitgang leek het scrum systeem een ideale ontwikkel methode.

3.1.2 Planning

Voor het project zijn er elf weken ingepland. Tijdens die elf weken wordt er verwacht dat elk project lid 8,5 uur per werkdag aan het project besteed. Omdat absenties niet vermeden kunnen worden is er besloten om de verloren tijd in te halen in het weekend of in de avond. Verder zijn we akkoord gegaan om zoveel mogelijk op dezelfde locatie te werken om zo de communicatie en de werksfeer te bevorderen.
Voor het opzetten van de planning is er gebruik gemaakt van de Team Foundation Service (TFS) [2]. De TFS is een tool aangeboden door Microsoft waarmee projecten die gebruik maken van een Agile ontwikkelmethode gepland kunnen worden. Zo bied het de mogelijkheid om een Product backlog samen te stellen om vervolgens de sprints te definiëren.
Door het gebruik van de Scrum ontwikkelmethode in combinatie met de TFS was de planning eenvoudig te maken. Allereerst werd de Product backlog gedefinieerd, dit werd gedaan door de eisen te verwoorden in backlog items. Vervolgens werden het aantal uren wat wij per dag zouden investeren in het project ingevuld, uiteindelijk werden er sprints gedefinieerd.
Het definiëren van de sprints van twee weken maakte het mogelijk om een tweewekelijkse meeting te houden met de opdrachtgever om de voortgang te tonen, in de vorm van een demo.

3.2 Het afstappen van de Scrum ontwikkelmethode

3.2.1 De aanleiding

Alhoewel de Scrum ontwikkelmethode een bewezen ontwikkelmethode is [3], hebben we drie weken na de start van het project ervaren dat het niet goed samenging met het ontwikkelteam en het project.
Drie weken in het project hadden we alle backlogs en sprints gedefinieerd. Tevens hadden we een sprint afgemaakt en een mockup van de globale user interface. Echter na de eerste meeting met de opdrachtgever bleken de gestelde eisen voor het eindproduct geen accurate weergave te zijn van de realiteit. Deze miscommunicatie, omtrent de eisen van de applicatie, zorgde ervoor dat de reeds opgezette backlog geen valide representatie meer was voor het gewenste product. Er waren in totaal elf weken gereserveerd voor het verwezenlijken van het project. Doordat er al drie weken verstreken waren en het een week duurt om de backlog aan te passen, hebben wij besloten om van de scrum ontwikkel methode af te stappen. Het afstappen van het voorgestelde Scrum model had als doel om meer tijd vrij te maken voor de implementatie van de webapplicatie. Door het niet opzetten van een nieuwe backlog hebben we een week werk bespaard en was het verwezenlijken van de opdracht in de gestelde tijd reëel.
Echter het aanpassen van de Scrum ontwikkelmethode impliceerde dat de planning aangepast moest worden. De aanpassing moest op zodanige wijzen gebeuren dat er minder tijd besteed zou worden aan het maken van backlogs en er meer tijd vrij kwam voor de implementatie.

3.2.2 Gevolgen voor de ontwikkel methode

Het wijzigen van de initiële Scrum ontwikkelmethode heeft ertoe geleid dat er nu een andere ontwikkel methode gekozen moest worden. Deze methode moest, zoals de Scrum ontwikkelmethode, ruimte bieden voor het tonen van voortgang aan de opdrachtgever op een tweewekelijkse basis. Ook moest de nieuwe ontwikkelmethode geen uitgebreid backlog systeem hebben zoals die van de scrum, wat teveel tijd zou kosten en het afronden van het project vertragen.
Uiteindelijk hebben wij gekozen voor het voortzetten van een uitgeklede versie van de Scrum ontwikkelmethode. In de volgende vier punten wordt deze uitgeklede Scrum ontwikkelmethode uitgelegd.
Ten eerste zijn de standing meetings behouden. Dit omdat wij merkte dat het meer overzicht gaf op de situatie, als er elke ochtend en avond besproken werd wat er die dag gedaan was of wat er gedaan zou worden.
Ten tweede hebben wij de baclock niet verder gespecificeerd dan de nieuwe eisen van het programma. Zodoende werd er minder tijd gestoken in het definiëren van de verschillende taken die elk backlog item bevat. Dit heeft aanzienlijk veel tijd bespaard aangezien de initiële opmaak van de eerste backlog meer dan een week in beslag nam.
Verder zijn de sprints behouden gebleven, maar werden deze gedefinieerd door element uit de opgestelde eisen. De bedoeling was dan nog steeds om om de week een sprint af te hebben zodat we nieuwe functionaliteit aan de opdrachtgever konden tonen.
Ook werd er na elke sprint een gehele exploratory UI test van de applicatie uitgevoerd. Bij deze manier van testen gaat de tester handmatig use casses simuleren, met het doel mogelijke defecten in de applicatie te vinden. Door deze manier van testen toe te passen hadden wij als doel om te controleren of de eisen goed geïmplementeerd waren. Om de controle zo goed mogelijk uit te voeren werd deze gezamenlijk door de projectleden uitgevoerd, omdat dit de kans verkleint dat fouten onopgemerkt blijven . Het resultaat van de controles werden dan opgeslagen in log bestanden. Deze bestanden gaven een goede indicatie van de status van de gerealiseerde eisen. Zie Appendix A voor een voorbeeld van een logfile zoals gebruikt in het project.
Verder werden de standing meetings gehouden om de de feedback momenten te behouden. Ook werd en er notulen bijgehouden om het verrichte werk te constateren en dus zo de vooruitgang te monitoren.

3.2.3 Gevolgen voor de planning

Deze uitgeklede versie van de scrum ontwikkel methode zorgde er wel voor dat er minder inzicht was in de vooruitgang van het project.
De kleine omvang van het ontwikkelteam (drie mensen), het feit dat we continue in dezelfde zaal waren en de standing meetings zorgde voor een goede communicatie tussen de projectleden. Deze goede communicatie leidde ertoe dat ieder groepslid continue op de hoogte waren van de activiteiten van de andere groepsleden. Door deze transparantie was het mogelijk om een simpele planning te maken. Dit gebeurde door middel van een dagelijkse eindvergadering waarin de vooruitgang en de toewijzingen besproken werden. De verslagen van de meetings werden geregistreerd in de notulen. Zie

Appendix B voor een voorbeeld van een notulen.
Naast het bijhouden van de notulen werd er een spreadsheet gemaakt die hoeveelheid gewerkten uren bijhoud. De spreadsheet houd ook statistieken bij zoals het aantal uren die nog aan het project besteed moesten worden. Deze uitgebreide spreadsheet is ook voorzien van een simpelere versie van de burndownchart zodat de voortgang van het project gevisualiseerd kon worden. Zie Figuur 3.

Figuur 3: Excel spreadsheet ter vervanging van de burndown chart

4 Ontwerp en implementatie

4.1 Gebruikersinterface

4.1.1 Interface Ontwerp

Nadat de gebruikte programma's, hulpmiddelen en ontwikkelmethode bekend waren, begonnen we aan het ontwerpen van het programma zelf. Nu vraagt de gebruikte ontwikkelmethode om het snel beschikbaar zijn van een demo. Dit was ook nodig omdat onze inschatting was dat de opdrachtgever met wat concretere ideeën zou kunnen komen als we snel een werkend prototype hadden.
Het belangrijkste design pattern voor deze applicatie, zijnde "Model, View, Controller" (MVC). MVC zullen we bespreken bij het ontwerp van de backend, lag al vast door de gekozen frameworks dus konden wij ons nu focussen op de Graphical User Interface (GUI). De reden dat we niet begonnen zijn met de UML, is omdat wij de opdrachtgever grafisch wilde weergeven hoe het programma er zal uitzien. Als we met de UML waren begonnen dan hadden we dit erg vaak aan moeten passen.
Over de ontwerpbeslissingen achter de GUI waren we het eigenlijk vrij snel eens. Een van onze uitdagingen was de bruikbaarheid (usability) van het programma. Een andere uitdaging was het afkrijgen van zeer rijke en uitgebreide functionaliteit in 11 weken. Verder was het gebruik van Scrum en onze feedback uit slechts demo's krijgen ook een uitdaging op zichzelf. Met dit in het achterhoofd hebben we besloten dat de GUI de volgende eigenschappen moest bezitten:

- De GUI moet voldoen aan het "DRY" principe (Don't repeat yourself) [4]
- De GUI moet modulair zijn
- De GUI moet voldoen aan het "KISS" principe (Keep it simple, stupid) [5]
- De GUI moet intuïtief in gebruik zijn.
- De GUI moet het gevoel geven van een rijke "webapp" in tegenstelling tot een website
- De GUI moet een duidelijke en simpele navigatie structuur hebben.

De eisen zijn gemakkelijk te verklaren aan de hand van de twee uitdaging waar deze op zijn gebaseerd. Zo zorgt het DRY principe ervoor dat veel functionaliteit herbruikbaar is. Zo wordt heel concreet bijvoorbeeld de personalia pagina die wij hebben gebruikt om zowel stagiairs als docenten als contactpersonen weer te geven.
De modulariteit is de eis die voortvloeit uit de gebruikte ontwikkelmethode. Door deze modulariteit kunnen wij gemakkelijk demo's laten zien waar een deel van de functionaliteit is ingebouwd en de rest niet aanwezig is. Hierdoor is het product in theorie klaar voor gebruik na iedere Iteratie. Ook kan makkelijk aan losse onderdelen gewerkt worden zonder hiermee functionaliteit in andere onderdelen te breken.
Het KISS principe en de eis dat de website intuïtief in gebruik moet zijn, vullen elkaar aan. Door de GUI zo simpel mogelijk te houden, is er minder ruimte om fouten te

maken en kan de nieuwe gebruiker de interface sneller onder de knie krijgen. Dit zorgt er tevens voor dat het intuïtief wordt om de website te gebruiken.
De laatste twee punten zijn nauw met elkaar verbonden. Het idee om een webapp te maken in tegenstelling tot een website en dat er een duidelijke en simpele navigatiestructuur moet komen heeft voornamelijk met de uitdaging in usability te maken. websites gebruiken vaak een boomstructuur voor navigatie: Menu's met submenu's en eventueel nog diepere niveaus. Om zo'n hiërarchische navigatie inzichtelijk te houden wordt vaak gebruikgemaakt van een "breadcrumb" (dat is bijvoorbeeld een tekst direct onder de navigatie in de vorm: Hoofdpagina > Subpagina > Subsubpagina) om aan te geven waar de gebruiker zich in deze hiërarchie bevindt. De opzet van deze webapp is echter zo dat er een dashboard is met slechts een niveau daaronder. Het idee daarachter is dat je altijd vanuit het dashboard naar alle pagina's kan komen en het altijd duidelijk is voor de gebruiker waar hij zich in de website bevindt.
De GUI die wij uiteindelijk hebben gebouwd is losjes gebaseerd op de "Modern UI" van Microsoft. Deze UI van Microsoft bestaat uit tegels die als snelkoppeling naar een nieuwe pagina dienen, maar tegelijkertijd ook alvast wat informatie op de tegel zelf kunnen zetten [6]. Deze GUI voldoet ook aan alle eerder gestelde eisen:

- Tegels met achterliggende functionaliteit zijn te hergebruiken
- Tegels zijn modulair: Ze kunnen worden verwijderd en toegevoegd
- Het dashboard oogt relatief eenvoudig, wat voldoet aan het KISS principe
- Het dashboard is intuïtief, het is meteen duidelijk wat met de verschillende blokken gedaan kan worden.
- Tegels geven een wat rijkere gebruikerservaring, omdat je daardoor niet het gevoel hebt dat je op een website zit.
- Door het gebruik van een dashboard met een achterliggende pagina is het altijd duidelijk waar de gebruiker zich bevindt.

Toen het duidelijk was welke kant de GUI op ging zijn we begonnen met het tekenen van mockups. Deze mockups hebben we met de proefversie van het programma "Balsamiq Mockup" gemaakt. Als een aantekening: Het was de bedoeling om deze mockup schetsen te laten zien bij de eerste meeting, omdat wij nog geen werkende demo zouden hebben na twee weken. Wij konden echter onze opdrachtgever niet bereiken wegens een twee weken durende schoolvakantie. Tegen de tijd dat de opdrachtgever wel bereikbaar was, hadden we wel een werkende demo te laten zien. De mockups zijn dus alleen intern gebruikt. zie de bijlage voor wat voorbeelden van mockups. Zie Apendix C voor de mockups.

4.1.2 Interface Implementatie

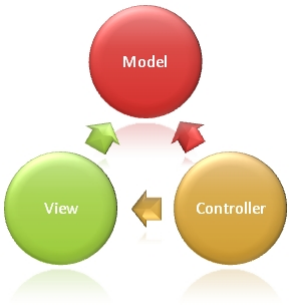
De hierboven genoemde design met mockups moest natuurlijk ook nog geimplementeerd worden. Daarvoor is er in eerste instantie gekozen voor HTML4.1/CSS2 en javascript/jQuery 1.9, zodat compatibel gebleven kon worden met Internet Explorer

(IE) versie 7 en 8. Echter is later besloten om alleen nieuwste versies van de Webkit browsers (Safari 6, Chrome 28, Opera 15), Firefox 22 en Internet Explorer 9 en 10 te ondersteunen. Door het laten vallen van ondersteuning voor IE8 en lager konden we nu gebruik maken van HTML5/CSS3 en javascript/JQuery 2.0. De keuze om IE8 en lager niet meer te ondersteunen kwam voor uit tijdsgebrek: Deze twee versie houden zich niet goed aan de standaarden van het Word Wide Web Consortium (W3C).
Verder biedt de ASP.NET nog een taal die sterk lijkt op C#, maar run-time gecompileerd wordt en daardoor ook iets weg heeft van PHP. Deze taal noemen we voor het gemak "Razor", maar het is eigenlijk de "Razor syntax for ASP.NET Razor HTML render engine". Dit zorgt ervoor dat er de HTML pagina's dynamische componenten kan hebben, waar wij gretig gebruik van maken. Zo is onze standaard manier van werken geweest om eerst statische HTML pagina's te maken en deze dynamisch te vullen met content via deze Razor taal. Door Razor is het gelukt om presentatie, logica en data compleet van elkaar te scheiden, wat de onderhoudbaarheid ten goede komt en het ultieme doel is van het gekozen MVC patroon. Dit zie je ook terug in de voorlopige SIG onderhoudbaarheidsindex: een 4- uit 5 (definitieve was op het moment van schrijven nog niet beschikbaar).
De implementatie van het dashboard was veel werk, er moest een hoop gesleuteld en afgestemd worden in de CSS, wat een erg arbeidsintensief proces was. Daarnaast werd ons duidelijk dat wij geen designers zijn. De initiële versie van het dashboard is dan ook niet bijzonder mooi en oogt alsof het uit het dos-tijdperk komt door het gebruik van veel grijs. Het idee erachter was overigens dat grijs een hele neutrale kleur was en op die manier de website rustig en simpel werd. De ontwikkeling die het dashboard heeft doorgemaakt van mockup tot definitief ontwerp is te vinden in Apendix D.
Nadat het dashboard af was, konden de losse modules gemaakt worden. Tot op dit punt waren de tegels nog statisch en zat er nog geen content achter. De uitdagingen die we tegen kwamen tijdens het implementeren van de interface lagen vooral in de GUI logica, zoals wanneer een foutmelding te laten zien. Verder moesten we leren omgaan met jQuery en het verslepen van vensters, maken van pop-ups en het sturen van AJAX aanroepen. Wij hadden daar allemaal niet zo veel ervaring mee dus kosten het wat tijd voordat dit vlot ging. Na de Initiële leercurve ging dit echter allemaal zeer vlot en door onze eerdere ontwerp keuzes was het eigenlijk altijd wel duidelijk welke GUI oplossingen er voor een gegeven probleem moest worden gemaakt.
Dan nog een kanttekening. We hebben, toen we redelijk aan het einde van het project waren, eens gekeken naar de GUI's van de concurrent (onStage). Na het nogmaals bekijken van deze GUI viel ons op dat onze GUI toch nog redelijk amateuristisch over komt. Wij zouden zeer veel profijt zouden kunnen hebben van een ontwerper, maar daar hebben we geen middelen voor.
Ondanks deze kanttekening is de uiteindelijke implementatie erg consistent en bruikbaar en voldoet deze prima aan de eisen die wij er aan stellen. We zijn dan ook zeer content met de definitieve versie van de GUI en positief gestemd over dit onderdeel van het project.

4.2 Backend

4.2.1 Backend ontwerp

Nadat we de ontwerpbeslissingen hadden gemaakt voor de GUI, was het tijd om de UML van de backend te gaan maken. Er is gekozen voor het ASP.NET MVC Framework, wat inhoud dat dus ook het al eerder genoemde MVC patroon zal worden aangehouden. Dit MVC design pattern bestaat uit drie onderdelen: Model, View, Controller. Zie ook Figuur 4. Het idee achter het patroon is dat het opdelen van het programma in deze drie onderdelen de onderhoudbaarheid ten goede komt. Dit gebeurd door te zorgen voor een scheiding van taken (seperation of concerns), voor zo min mogelijk verwijzingen tussen de onderdelen (low coupling) en door zoveel mogelijk taken te bundelen (high cohesion).
Alhoewel het MVC patroon in de loop van de jaren een iets andere betekenis heeft gekregen, is het in het ASP.NET MVC Framework wel duidelijk omschreven hoe elk onderdeel gebruikt dient te worden. Het verschil zit vooral in het feit dat MVC vroeger voor "stateful" aplicaties werd gebruikt (Dat zijn applicaties waar UML state machines voor te maken zijn). De MVC van ASP.NET MVC is echter bedoelt voor websites die gehost worden op het internet. Deze website zijn inherent "stateless" [7]. Concreet komt het er op neer dat de wat oudere literatuur over MVC controllers uitgebreide logica hebben en in de wat recentere, met name internet gerichte literatuur over MVC controllers, de controllers redelijk klein zijn. In de volgende drie paragrafen zullen we de losse onderdelen van MVC bespreken.



Figuur 4: Het MVC patroon

Het Model bevat alle gegevens van de applicatie en alle mogelijkheden om nieuwe gegevens te verzamelen en te verwerken. Het Model wordt nooit direct voorgeschoteld aan de gebruiker maar alle data komt hier wel vandaan. Het Model heeft in principe geen weet van het bestaan van de andere twee onderdelen. Dit Model is een model van de werkelijkheid en bevat bijvoorbeeld objecten zoals personen en relaties. Verder bevat het Model alle "business logic", dat is de logica die op de achtergrond de data ergens vandaan haalt, opslaat, bewerkt et cetera. In ons specifieke geval heeft de meeste business logic betrekking op de database, authenticatie en onze import/export

functionaliteit. Deze onderdelen worden zo apart besproken, maar zitten dus in het Model. Bij ASP.NET kan het Model in twee talen geschreven worden: C# of Visual Basic (of een combinatie van beide). Wij hebben gekozen voor C# omdat dit dichter bij talen ligt waar wij ervaring mee hebben, namelijk C++ en Java.
De View wordt in ASP.NET MVC in HTML/CSS gemaakt. De View is de webpagina de gebruiker uiteindelijk te zien krijgt. In de View hoort geen logica te zitten, al is een kleine loop nog wel toegestaan om wat schrijfwerk te voorkomen (als concreet voorbeeld: als er een lijst van duizend man is hoeft dit met een simpele loop maar een keer opgeschreven te worden). Wat dus niet toegestaan is, is bijvoorbeeld het aanspreken van de database en daar een logische operatie uitvoeren. In ASP.NET MVC is het mogelijk om logica aan de HTML pagina's toe te voegen door gebruik te maken van Razor, zoals al eerder besproken. ASP.NET MVC dwingt echter niet af dat er geen business logic voor komt in de View, wij moesten er dus heel bewust voor zorgen dat er geen database calls e.d. gedaan werden in de View. Dit was nog behoorlijk ingewikkeld, maar meer daarover later in dit hoofdstuk. De View heeft geen weet van het bestaan van andere componenten, al spreekt de View wel met de Controller (wat een willekeurige Controller kan zijn) door middel van HTTP requests en HTTP responses. Verder kan Razor wel gegevens uit het Model halen maar is het de bedoeling dat er spaarzaam met deze functionaliteit omgegaan wordt, zodat het wisselen van gebruikersinterface makkelijker is.
De Controller in ASP.NET MVC is klein. Als een gebruiker een URL intypt in de adresbalk, dan zorgt het routing framework dat onderdeel is van ASP.NET dat deze terecht komt bij een functie in het Controller object. zo komt de pagina www.websitenaam.nl/standaard terecht bij de homecontroller en de index functie. De enige taak van de Controller is de juiste informatie van en naar het Model door te spelen en deze informatie ook aan de View te verstrekken. De index functie van het homecontroller object kijkt bijvoorbeeld wie er ingelogd is, en haalt de juiste gegevens uit het Model om de dashboard tegels in de View te kunnen weergeven. De Controller heeft dus weet van de View en het Model. Het is niet de bedoeling dat de Controller zelf de regels uitvoert, maar hier mag wel willekeurige data uit het Model gevraagd worden.
Een tweede framework dat gebruikt is, is het "Entity Framework" (EF). Wij hebben versie 5 van dit framework, dus EF5.0 gebruikt. Entity Framework maakt een mapping tussen relationele databases en object georiënteerde programmeren. Het is niet triviaal hoe relationeel in object georiënteerd moet worden veranderd. Zo ondersteunt het relationele model in principe geen "inheritance" en is in het object georiënteerde model het begrip "key/foreign key" (PK/FK) niet bekend. Dit komt omdat deze begrippen vreemd zijn in de andere context.
Normaal gesproken wordt dit probleem omzeilt door "search query" statements (SQL statements) te doen. Dit zijn methodes die in de object georiënteerde omgeving, van te voren gemaakte opdrachten uitvoeren in de relationele omgeving. Deze manier van werken brengt weer zijn eigen gevaren met zich mee. De belangrijkste is de zogeheten "SQL injection". Bij deze vorm worden de mee te geven parameters aan de opdracht zo gemanipuleerd dat de opdracht iets anders gaat uitvoeren. Dit is oplosbaar, maar er moet wel bij elke input bedacht worden dat deze mogelijkheid bestaat.
Entity Framework lost dit probleem op een andere manier op. EF maakt een mapping

tussen de ene denkwijze en de andere in de vorm van een object/relationele mapping. Hierdoor representeren een verzameling objecten in de object georiënteerde omgeving, een verzameling tabellen in de relationele omgeving. Vervolgens zijn deze objecten gewoon te gebruiken zoals deze altijd gebruikt werden. Het enige verschil is dat deze verzameling objecten nu eigenlijk een database voorstellen. Nadat de manipulatie van de objecten klaar is. Kan worden geverifieerd of de objecten in de juiste staat zijn (denk hierbij bijvoorbeeld aan een check of een bepaald veld in de objecten niet "null" is). Als de programmeur tevreden is met de staat van de objecten, kan hij EF de objecten op laten slaan in de database. Op de achtergrond voert EF dan wel een query uit, maar deze is verborgen voor de programmeur.
De mapping die EF gebruikt, moet normaal gesproken expliciet gemaakt worden. Dat wil zeggen: Eerst wordt een database en een object georiënteerde model gemaakt. Vervolgens wordt er een expliciete mapping grafisch of in XML gemaakt in een vorm die Entity Framework snapt. Een nieuw en veel aangeprezen feature in EF5.0 was het gebruik van zogeheten "code first" migrations. Het verschil met de tradionele "Database First" en "Model First" migrations – waarbij een explicite mapping gemaakt moet worden tussen het model en de database – is dat hier de mapping impliciet gebeurd. Het voordeel daarvan is dat er geen rekening meer gehouden moet worden met de database en alleen nog maar met het object-georiënteerde model. De database wordt aangemaakt door EF en bijgewerkt als het model aangepast wordt. Dit leek ons een ideale manier van werken. Hier zijn wij trouwens later op teruggekomen. Meer informatie daarover is te lezen in de sectie over backend implementatie.
Authenticatie is ook een belangrijk onderdeel van deze website. Het is bijvoorbeeld essentieel dat leerlingen niet alle beoordelingen zouden kunnen wijzigen. Het was vanaf het eerste ontwerp al duidelijk dat wij met meerdere rollen en permissies zouden moeten gaan werken. Hoewel de rollen goed gedefinieerd waren, waren de permissies redelijk variabel en was ook niet duidelijk of deze permissies in de toekomst hetzelfde zouden blijven.
Gelukkig heeft het framework voorzien in authenticatie door een statische class die "WebSecurity" heet en magische functies heeft inde vorm van bijvoorbeeld "Login", "Logout", "CurrentUser" en ondersteuning voor rollen en permissies. Het enige wat het moest weten is wat te gebruiken als username en wat te gebruiken als wachtwoord. Dus dit hadden we gekoppeld aan de database. Wat we nog niet wisten tijdens het design, is dat als je de standaard een beetje wilt wijzigen je een doos van Pandora opent. Ook meer hierover in de volgende paragraaf.
Verder was het duidelijk dat we een import en export class moesten schrijven. Dit omdat er leerlingen lijsten uit een Excel spreadsheet, wat het huidige systeem is, ingeladen moeten worden en omdat er een uitdraai van alle statistiek moest komen. Ook bleek na wat gesprekken met de opdrachtgever dat het heel veel tijd zou schelen als er een stagecertificaat uit het programma kan rollen, dat deden ze tot nu toen met de hand. Deze functionaliteit zouden we in eerste instantie bieden door gebruik te maken van de ingebouwde library, maar deze werd niet goed op servers ondersteund. Uiteindelijk is er gebruik gemaakt van een library genaamd "EPPlus" die de spreadsheet kan inlezen met header velden en gebruikte te maken van de "Reflection" library ingebouwd in C# om runtime de velden van een willekeurig object uit te lezen zodat we daarna een

koppeling tussen ons object en in het model en de headers van de spreadsheet konden maken.

4.2.2 Backend Implementatie

De implementatie van de backend was vooral een kwestie van doen. Het was veel werk omdat er een gigantische lijst aan features was. Wat misschien nog wel meer werk was, was het opzoek werk, omdat wij allen niet heel bekend waren met de gebruikte tooling. Hier was de website stackoverflow [8] een geweldige hulp in. Alle fouten die we hadden en alles wat we niet wisten waren hier al uitgebreid aan de community gevraagd en beantwoord. De enige vraag die je nooit op stackoverflow moet zoeken zijn vragen over een "Stack overflow exception", omdat die op elke pagina een hit geeft! Gelukkig hadden we er daar niet zoveel van.
De implementatie van het MVC patroon was eigenlijk niet zo heel moeilijk. Er was af en toe discussie over waar bepaalde logica ging, maar over het algemeen was het wel duidelijk. Het enige vervelende is dat alle objecten die we via EF hebben verkregen nooit daadwerkelijk de objecten waren. Aangezien iedereen wel met iedereen gelinkt was via via, betekent het daadwerkelijk verkrijgen van alle relaties dat de hele database gebruikt moet worden. EF lost dit op door alleen maar "Proxies" terug te geven van de objecten met de expliciet ingeladen objecten (dat zijn alle primitieven zoals strings en integers, maar ook de met "include" expliciet aangegeven objecten). Als er objecten aangeroepen werden die niet ingeladen waren, dan waren er twee mogelijkheden: Ofwel je kreeg een cryptische fout die er op neer kwam dat de database connectie dicht was (EF probeerde het object alsnog uit de database te halen en dat mislukte), ofwel EF haalde het object wel met succes uit de database, wat een violatie is van ons MVC model (dit mag niet in de View gebeuren). Toen we ons dit realiseerde hebben we de website nagelopen en hier verder extra op gelet.
Het Code First Entity Framework dat we gebruikte hebben was achteraf niet heel handig. We hadden veel te weinig controle over de database en we ware vaak aan kleine aanpassingen aan het maken aan het model om nog enige invloed op de vorm van de database te hebben. Eigenlijk hadden we ook wel kunnen bedenken dat dit niet de manier is. We hebben altijd in alle colleges die ook maar iets met dit onderwerp te maken hebben geleerd dat je beter eerst je database kan ontwerpen voordat je aan de rest begint. Dit wijze advies hebben wij compleet genegeerd omdat Microsoft zo vol was van hun code first methode. Tegen de tijd dat we het erover eens waren dat dit niet zo'n prettige werkwijze was, waren we te ver in het project om de complete database nog om te gooien. Reflecterend kunnen we wel stellen dat we volgende keer nog steeds EF zouden willen gebruiken, maar dan de Database First variant daarvan. Het was overigens wel heel leerzaam om het object naar relationele probleem eens anders aan te pakken.
De ingebouwde authenticatie die we zouden gebruiken voldeed niet helemaal aan onze eisen. Toen we deze wilde gaan aanpassen bleek dat gelijk aan het openen van een doos van Pandora. De ingebouwde WebSecurity class deed op te achtergrond ontzettend veel, inclusief het aanmaken van tabellen in de database. Uiteindelijk hebben

we de source van ASP.NET MVC [9] gekopieerd in ons project en deze gestript van alles wat we niet nodig gingen hebben (Dat was voornamelijk "OAuth" functionaliteit, waardoor het mogelijk is om in te loggen met je bestaande Facebook, Google, Microsoft account). Uiteindelijk hebben we dus de backend van Websecurity helemaal zelf geimplementeerd zodat we voledige controle hebben over wat er gebeurde als we login aanroepen. Ook hebben we een proxy voor Websecurity gezet die we Security noemde. Het voordeel van deze werkwijze is dat we nu onze eigen Security klasse kunnen gebruiken voor inloggen en alles eromheen, maar dat we ook nog gebruik kunnen maken van de ingebouwde functionaliteiten zoals "Annotations" Dit zijn sleutelworden die boven een methode gezet kunnen worden, zoals bijvoorbeeld: AuthorizeUser(Role=Coordinator). Dat scheelt een hoop werk. Het was overigens erg leerzaam om ASP.NET implementatie na te lezen en we hebben er ook voor gekozen om de "RFC 2898 Derived Bytes" te gebruiken die ASP.NET ook implementeerde. Er is veel werk gegaan in de cryptografie en password hashing, maar het uitleggen daarvan valt dat buiten de scope van dit eindverslag.
Toevalligerwijs is het systeem van import export, waarbij het meeste mis had kunnen gaan, gewoon goed gegaan en werkte alles in een keer zoals voorzien. Het was nog steeds wel veel werk omdat we de juiste libraries moesten vinden en daar de documentatie van lezen. We hebben helaas geen goede Word library gevonden, maar wel eentje voor Excel. Daarom zijn de certificaat exports niet in doc of pdf formaat maar in Open XML formaat. Dit formaat is overigens gewoon docx en dus te lezen in Word.

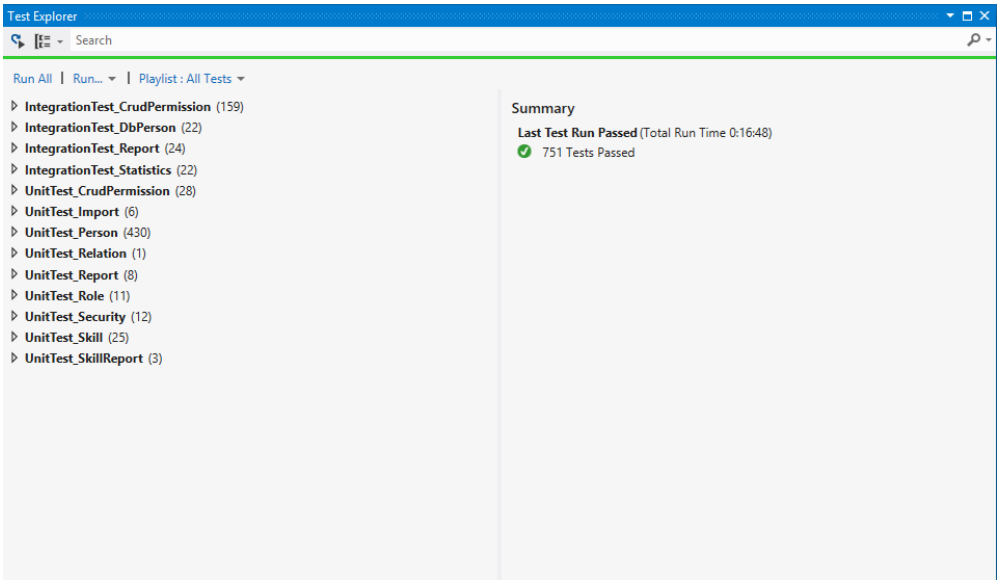
5 Testing

Het testen van software wordt vaak uitgesteld tot het einde van het project. Dit gebeurt omdat er dan externe frameworks geïnstalleerde moeten worden of omdat het ontwikkel model pas ruimte bied voor het testen na de volledige implementatie van de applicatie. Hierdoor word het testen een lang onderdeel van het project wat ertoe kan leiden tot een verdubbeling van de totale ontwikkeltijd.
In het volgend deel van het eindverslag zal het verband gelegd worden hoe het de keuzen van de ontwikkel omgeving en de ontwikkel methode geleid hebben tot de mogelijkheid om op een vroeg stadium van het project te beginnen met testen. Verder zullen de verschillende soorten testen aan bod komen en zal de SIG feedback en de daardoor doorgevoerde veranderingen geanalyseerd worden.

5.1 Tools

De gebruikte ontwikkelomgeving, Visual Studio, is voorzien van een uitgebreide test omgeving zoals het Microsoft unit testing framework en de Test Explorer.

De Test Explorer, afgebeeld in Figuur 5 integreert het Unit testen in het ontwikkelproces. Het bied de mogelijkheid om meerde test projecten te koppelen aan een project. Zo kunnen de verschillende soorten testen, voor hetzelfde project, gescheiden worden. De Integratie testen kunnen zodoende gescheiden worden van de Unit testen en dus onafhankelijk van elkaar gebruikt worden. Verder geeft de tool ook de mogelijkheid om de testen te groeperen op verschillende criteria. Zo kan er gefilterd worden op tijd, wederom de test geslaagd is en nog velen meer. Deze optie is zeer handig omdat het help om een duidelijk beeld te kunnen schetsen waar de code niet aan de verwachtingen voldoet en dus verbeterd moet worden.



Figuur 5: Visual Studio Test Explorer

Het Microsoft unit testing framework bied ook een breed scala aan hulpmiddelen. Een daarvan is dat het framework de mogelijkheid bied om, tijdens het testen, een stuk code te isoleren door het gebruik van de Fake Assemblies. Deze Assembly kan onderverdeeld worden in twee coterieën: “Stubs” en “Shims” met beiden voor en nadelen.
Stubs vervangen de klasse implementatie met een kleinere vervang klasse die dezelfde interface erop nahoud. Dit is interessant om te gebruiken, in gevallen waar er nog geen implementatie van de desbetreffende klasse is, wanneer er volledige controle over de klasse vereist is of om de stub te scheiden van gebruikte afhankelijkheden (dependencies).
Shims, in tegenstelling tot stubs die bij het compileren gegenereerd worden, worden tijdens de runtime geïnitialiseerd. Dit betekend dat in plaats van een methode aan te roepen dat de shim de gegeven shim code uitvoert. Dit is erg handig wanneer de gegeven functionaliteit van een klasse geïmplementeerd is in assemblies die niet gemodificeerd kunne worden of wanneer er maar controle nodig is over een klein onderdeel, een paar methodes, van een klasse.
Zo kan er geconcludeerd worden dat de keuzen van de ontwikkelomgeving een grote impact heeft gehad op de mogelijkheid om al in een vroeg stadium te beginnen met testen. Daarbij hebben de scrum ontwikkel methode en het MVC patroon hier ook aan bijgedragen.

5.2 Process methodology

Het scrum model en het MVC patroon hebben beiden een belangrijke rol gespeeld in het testen van de applicatie.
Het gebruik van de Scrum ontwikkelproces bood de mogelijkheid om in een vroeg stadium van de ontwikkeling al te beginnen met testen. Dit werd mogelijk gemaakt door de definities van de verschillende sprints aan het begin van het project. Aan het eind van iedere sprint werd de code, die geschreven was in de sprint, onderworpen worden aan testen. Zodoende is het functioneren van het programma gewaarborgd.
Door het gebruik van het MVC patroon, om de code te structureren, is het mogelijk om de verschillende testen gescheiden te houden. Hierdoor werd het testen van de verschillende onderdelen versimpelt. Unit en integratie testen werden opgesteld voor de modellen zodat de integriteit van de backend en de database gewaarborgd worden. Verder werden er ook UI testen gemaakt voor de View, zodoende werd de integriteit van de GUI gegarandeerd.

5.3 Types of tests

In dit project is er besloten om verschillende soorten testen te gebruiken om zo de kwaliteit van de software te verhogen. in het volgend stuk zullen de Unit test en de Integratie test aan bod komen, de reden waarom er afgestapt is van de automatische UI testen en er verder gegaan is met handmatig UI testen. Uiteindelijk zal er ingegaan worden op onze ervaring met SIG en de consequenties ervan.

5.3.1 Unit Testing

unit testen zijn kleine stukken code die getest worden. Deze testen onderzoeken of een gegeven methode de gewenste output geeft. Deze testen testen elk een methode en zijn onafhankelijk van elkaar. Dit leid tot kleinere, onafhankelijke en snellere testen. Omdat elke methode los van elkaar getest worden en dus geen verwijzingen naar elkaar mogen hebben werd er gebruik gemaakt van het stubben van de afhankelijke stukken code. Een aantal voorbeelden van Unit testen worden weergegeven in Listing 1, 2 en 3. Door het gebruik van deze vorm van testen werden er meer dan 500 testen geschreven waarvan een deel problemen met de implementatie aan het licht brachten.

```
/// <summary>
/// invalid email test,
/// this one misses the @-sign (should fail)
/// </summary>
[Test, UT]
[ExpectedException(typeof(ArgumentException))]
public void UnitTest_Person_Email_Invalid()
{
        p.Email = "abs.def_noatsign_ghi.com";
}
```

Listing 1: Unit test 1

```
/// <summary>
/// Adds a role class to the person
/// Please note that this not add it to the database
/// </summary>
[Test, UT]
public void UnitTest_Person_Email_Invalid()
{
        var roleClass = new Role();
        roleClass.Type = RoleType.CompanyContact;
        p.Roles.Add(roleClass);
        Assert.That(p.Roles, Contains.Item(roleClass));
}
```

Listing 2: Unit test 2

```
/// <summary>
/// Create new full skill score and check description
/// </summary>
[Test, UT]
public void UnitTest_Skill_CreateFullSkill_CheckName()
{
        SkillTrack track = new SkillTrack()
        {
                Id = 300,
                Value = "skillTrackValue"
        };
        SkillName name = new SkillName()
        {
                Id = 4100,
                Value = "skillNameValue"
        };
        SkillLevel level = new SkillLevel()
        {
                Id = 253,
                Value = "skillLevelValue"
        };
        Skill skill = new Skill()
        {
                Track = track,
                Name = name,
                Level = level,
                SkillTrackId = 300,
                SkillNameId = 4100,
                SkillLevelId = 253,
                Description = "descriptionValue"
        };
        Assert.AreEqual(4100, skill.Name.Id);
        Assert.AreEqual(4100, skill.SkillNameId);
        Assert.AreEqual("skillNameValue", skill.Name.Value);
}
```

Listing 3: Unit test 3

5.3.2 Integration testing

Met de term integratie testen wordt verwezen naar elke andere vorm van testen dan de Unit testen. In ons geval zijn dat de testen die afhankelijk zijn aan de database en die de database updaten. Omdat het systeem constant wijzigingen aan de database moet waarnemen is het van belang dat de database consistent blijft en dat de bijbehorende

modellen op een correcte wijze geüpdatet worden. Om deze criteria te waarborgen zijn er meer dan 200 Integratie testen geschreven.
De grootste uitdaging van de applicatie is het bijhouden van alle modellen. Dit doordat alle modellen ten alle tijden geüpdatet kunnen woorden en dat deze nauw met elkaar samenwerken voor het correct weergeven van de informatie aan de gebruiken. De implementatie van de applicatie is hieraan aangepast, er werd gebruik gemaakt van providers die de database updaten en consistent houden. Omdat de functionaliteit, in deze provider klassen geclusterd zijn, werd het testen van de database consistentie methode ook geclusterd en versimpeld. De meeste integratie testen testen dan ook de functionaliteit van de provider klasse.
Door deze opbouw van de applicatie en het schrijven van testen werd de dtatabase integriteit gewaarborgd. Een paar voorbeelden van integratie testen zijn afgebeeld in Listing 4 en 5.

```
/// <summary>
/// Tests is the coordinator has been added
/// successfully to the database
/// </summary>
[Test, IT]
public void IntegrationTest_DbPerson_AddMinimalCoordinator()
{
        // Get minimal Person
        Person minimalCoordinator = this.MinimalCoordinator;

        // Add person
        Person resultPerson =
                Security.AddOrUpdate(minimalCoordinator);

        // Check if person has been added
        Assert.IsNotNull(resultPerson);

        // Check if added person == minimal mentor
        Assert.AreEqual("coordinatorEmail@Value",
                resultPerson.Email);
        Assert.AreEqual("coordinatorFirstNameValue",
                resultPerson.FirstName);
        Assert.AreEqual("coordinatorLastNameValue",
                resultPerson.LastName);
}
```

Listing 4: Integration test 1

```
/// <summary>
/// SetUp method is called before each individual test case.
/// This SetUp deletes the database,
/// reseeds it and initializes a new database context
/// </summary>
[SetUp]
public virtual void SetUp()
{
        using (var context = new AuthenticatedUserDbContext())
        {
                context.Database.Delete();
                Database.SetInitializer
                (
                        new MigrateDatabaseToLatestVersion
                        <AuthenticatedUserDbContext,
                        Staport2.Migrations.Configuration>()
                );
                context.Database.Initialize(true);
                Database.SetInitializer
                (
                        new AuthenticatedUserDbContext
                                .Initializer()
                );
                context.Database.Initialize(true);
        }

        // Initialize shimContext
        shimContext = ShimsContext.Create();

        // Shim email methods
        ShimSecurity
                .SendEmailStringStringStringStringBoolean =
                    (str1, str2, str3, str4, bool1) => true;
        ShimSecurity
                .SendPasswordEmailPersonStringBoolean =
                    (person1, str1, bool1) => true;
}
```

Listing 5: Integration test setup

Zoals afgebeeld in Listing 5, maakt de integratie test voor de setup gebruik van een shim. De reden voor het gebruiken van een shim is dat de “Security” klasse een email verzend wanneer een nieuw persoon aangemaakt wordt. Omdat er elke keer een email verzonden word is dit niet handig tijdens het testen. De shim vervang de functionaliteit van de “sendEmail” methode waardoor er geen email verzonden word

en dat de test slaagt.

5.3.3 Coded UI Tests

Coded UI testing (CUIT) is voor het testen van de user interface en Visual Studio heeft daar zijn eigen framework voor. Deze manier van testen is een goede manier voor het waarborgen van consistentie van de user interface waarmee de eindgebruiker geconfronteerd zal worden.
Door de definitie van sprints is tijdens de ontwikkeling van het project elke week een onderdeel af. We zijn begonnen om aan het eind van elke sprint een par user test te doen om de kwaliteit van de UI te verifiëren. Tijdelijk werd er overgestap naar het CUIT om het testen te automatiseren. Echter na een par weken bleek het implementeren van zulke testen veel tijd in beslag te nemen en werdt er geconcludeerd dat het handmatig testen voor ons project efficiënter was.

5.3.4 Gebruikerstests

Zoals hierboven vermeld werden gebruikerstests gedaan aan het eind van elke sprint. Om de gebruikerstests te kunnen uitvoeren was er een gevulde database nodig. Het vullen van de database met simulatie data werd gedaan door een template op te zetten en deze over te brengen naar de database. In deze template zijn er verschillende instantie van elk onderdeel in de database gecreëerd. Het vullen van de database maakte het mogelijk om realistische gebruikers testen te maken om te verifiëren of alles naar wens werkte.
Het gebruik van deze testen brachten velen inconsistenties aan het licht die veroorzaakt werden door Javascript of CSS fouten, die niet getest konden worden. Die fouten werden veroorzaakt door foute implementatie van functies maar ook door de implementatie van events, zoals het koppelen van de events aan de “drag and drop” functionaliteit in de webapplicatie.
Tijden het gebruikerstesten kwamen er ook fouten aan het licht die niet veroorzaakt werden door de door ons geschreven code. Zo werd er een fout ontdekt in de JQuery code. The drag and drop functionaliteit werkte niet naar wens in het bijzonder in Internet Explorer. Door de open source eigenschap van JQuery hadden wij toegang gekregen tot de broncode en hebben wij het probleem verholpen.

5.4 Software Improvement Group

Tijdens dit project moest de code tweemaal ter revisie naar de Software Improvement Group (SIG) gestuurd worden. SIG beoordeelde dan vervolgens de onderhoudbaarheid en integriteit van code. Ten tijde van het schrijven van dit document was de tweede revisie nog niet terug.

5.4.1 Eerste feedback

De SIG was positief over onze eerste inzending. Ze hebben ons dan ook een 4- op 5 gegeven tijdens het eerste feedback moment. Daarbij kwam de opmerking dat als we

de punten zouden verbeteren en dezelfde fouten niet meer in volgende code zouden maken, de gegeven index waarschijnlijk een 4 op 5 zou worden. De verkregen feedback is na te lezen in Apendix E van dit document.

We hebben na de feedback van SIG ervoor gekozen om de person class uit te splitsen in een person en een useraccount class. Verder waren onze permissie checks eerst in person (bijvoorbeeld person.IsA(Administrator), dat hebben we verplaatst naar de security class.
De import functie hebben we opgesplitst en werkt nu door middel van pipelining. Elke methode heeft nu een afgeschermd stukje van de verantwoordelijkheid om uiteindelijk te kunnen importeren.
Verder hebben we de onderdelen die SIG genoemt heeft ook voor de rest van de code nog even nagelopen en is het advies in acht genomen om zo door te gaan.

5.4.2 Tweede feedback

De SIG was wederom positief over onze inzending. SIG merkte op dat onze code gegroeid was, maar desondanks haalde deze hoger op de onderhoudsindex. Tevens waren ze positief over het doorvoeren van de aanbevelingen van de eerste evaluatieronde. De verkregen feedback is na te lezen in Appendix F van dit document.

6 Reflectie

In dit hoofdstuk wordt gereflecteerd op het verloop van het project. Eerst zal gereflecteerd worden op de eerder geïdentificeerde uitdagingen (Main challenges) van de gebruikte werkwijze, het testen en het ontwerpen van de applicatie. Daarna wordt besproken hoe de samenwerking tussen de projectleden is verlopen en hoe de samenwerking tussen de projectgroep en andere partijen is verlopen.

6.1 Uitdagingen

We hebben veel geleerd van dit project. Zo zijn de hieronder beschreven uitdagingen allemaal succesvol afgerond aan het einde van het project en hebben wij een hoop ervaring opgedaan. Als algemene opmerking is nog wel te stellen dat de uitdagingen in de praktijk niet altijd voorzien zijn in de theorie.

6.1.1 Testen

De uitdaging bij het testen zat niet zo zeer in het testen zelf, maar in het ontwerpen van testbare code. Zo moesten we zorgen dat alle afhankelijkheden van de te testen classes geen directe implementaties, maar interfaces waren. Op deze manier is het makkelijk om de testen te isoleren en de afhankelijkheden te beperken. Wij hebben ook vaak gebruik gemaakt van het "Factory design pattern". Dit patroon zorgt ervoor dat niet direct de te gebruiken class zelf wordt aangemaakt, maar een factory class die de class aanmaakt. Het voordeel hiervan is dat deze factory class vervangen kan worden door een andere factory voor het testen. Een voorbeeld van gebruikte factory class is die voor de databasecontext. De normale factory maakt een database context aan, maar tijdens het testen kon dit vervangen worden door een andere factory, waardoor de database niet getest hoeft te worden om andere functionaliteit te testen.

6.1.2 Ontwerpen

De belangrijkste uitdaging bij het ontwerpen was dat we vanaf nul moesten beginnen. Hierdoor moesten het ontwerp van de grond af aan gemaakt worden. De andere uitdaging was dat er een gebruiksvriendelijke applicatie moet komen. Het blijkt dat "gewoon simpel" voor de eindgebruiker betekend dat het niet "gewoon simpel" voor de programmeur is, integendeel zelfs. Bij elke ontwerp beslissing moesten we er rekening mee houden dat het simpel in gebruik moest zijn. Wij vinden dat we uiteindelijk goed geslaagd in onze ontwerp uitdagingen: Het eindproduct voldoet aan onze ontwerpeisen.

6.1.3 Werkwijze

Bij de werkwijze was de belangrijkste uitdaging om alles zelf te ontwerpen. Er was weinig ondersteuning vanuit het bedrijf en ook niet vanuit de Universiteit. Er moest dus een goed en gedisciplineerde manier van werken gebruikt worden. Door altijd fysiek in de zelfde ruimte aanwezig te zijn waren er makkelijk afspraken te maken.

Daarnaast was het op die manier gemakkelijk om overeen te stemmen over de ontwikkelbeslissingen.

6.2 Samenwerking

6.2.1 Interne samenwerking

De samenwerking binnen de projectgroep ging goed. De werksfeer was plezierig en de projectleden zaten op een lijn wat betreft beslissingen over interface en code. Ook hebben we allemaal zeer significant bijgedragen en elkaar geholpen waar nodig. Er was niemand die mee heeft gelift met de rest, iets wat normaal gesproken toch vaak gebeurd. Elke dag van half negen tot half zes werken was erg zwaar maar is toch goed volgehouden door iedereen. Vrijdag mochten we een half uur eerder stoppen, erg fijn tegen het weekend aan. Door elkaar veel te helpen een de positieve arbeidsethos werd door ons alle drie dit project niet als een verplichting maar als een uitdaging ervaren. Al met al is dit laatste project van onze bachelor tijd op de TU Delft, het leukste project om aan gewerkt te hebben.

6.2.2 Externe samenwerking

De samenwerking met de opdrachtgever ging goed. We hebben in totaal vier meetings gehad. De eerste meeting had vooral als doel om de lijst met eisen samen te stellen. De meetings daarna om de verschillende demo's te tonen. Miscommunicatie zorgde er wel voor dat we meerdere weken aan vertraging opliepen. Dit had door meer ervaring van onze kant wel voorkomen kunnen worden. Verder had er eerder ingegrepen kunnen worden als de opdracht gever niet de eerste vier weken onbereikbaar was.
We hebben een enkele meeting gehad met de begeleider vanuit de TU Delft waarin eigenlijk vooral besproken is wat we op moesten leveren. Wij kregen de indruk dat dit ook de bedoeling was dat we op een onafhankelijke manier te werk moesten gaan en hebben daarom erg veel van onze problemen zelf uit kunnen vinden.

7 Conclusie

Wij hebben erg veel geleerd van dit project. Communicatie is essentieel, zowel tussen de projectleden als tussen de projectgroep en opdrachtgever. Door een goede samenwerking tussen de projectleden was het een leuk project om aan te werken. Aan het groepsproces hoeft concluderend niet zoveel aangepast te worden. Wel zouden we de volgende keer, zoals al eerder gesuggereerd, misschien beter een andere ontwikkelmethode kunnen gebruiken.
Het belang van communicatie met de opdrachtgever is tijdens dit project aan het licht gekomen. Hier zouden we nog zeker veel beter in kunnen worden. Zo hadden we bijvoorbeeld de opdrachtgever beter kunnen betrekken bij het project. Verder hadden we misschien toch UML aan de opdrachtgever voor moeten leggen, zoals gesuggereerd door onze begeleider van de TU Delft. Onze inschatting was dat het voorleggen van UML niet zo veel op zou schieten. Wij zouden voor toekomstige projecten meer nadruk kunnen leggen op het communiceren met opdrachtgevers. Onze verwachting is echter dat dit met wat meer praktijkervaring vanzelf beter zal gaan.
Ook hebben wij onderschat hoeveel werk een project als dit is. In eerste instantie verwachtte we dat dit project ongeveer twee van de drie maanden zou duren. Achteraf kunnen wij concluderen dat dit zeker niet het geval is en dat om aan alle features en alle omgevingen te voldoen wij eerder richting de zes maanden nodig zouden hebben. Zo hebben wij bijvoorbeeld de uitrol (deployment) van de applicatie nog niet gedaan.
Als conclusie stellen wij dat het project voor ons een goed leerproces was waarbij we velen onderdelen uit de studie Informatica terug hebben zien komen. Tot slot concluderen wij dat het opdoen van praktijkervaring ook essentieel is voor dit soort projecten.

8 Aanbevelingen

In deze paragraaf zullen er nog enkele aanbevelingen gedaan worden voor de toekomstige verbetering van het project. Zo zullen de extra functionaliteiten die nog niet geïmplementeerd zjin aan bod komen en zullen verschillende mogelijke code optimalisaties besproken worden. Daarna zal er nog wat over extra abstractie en testen gezegd worden.

8.1 Extra functionaliteiten

Er kunnen natuurlijk enorm veel functionaliteiten toegevoegd worden aan ieder willekeurig programma. De functionaliteiten die wij in dit hoofdstuk bespreken hebben zijn echter functionaliteiten die wij graag nog hadden willen inbouwen als we iets meer tijd hadden. Ze vergemakkelijken handelingen die nu ook al mogelijk zijn, of voegen wat extra's toe dat niet noodzakelijk is.
Een eerste functionaliteit die we graag zouden willen toevoegen is het importeren van lerarenlijsten. Op dit moment is het wel mogelijk om nieuwe leraren in te voeren in het systeem, maar het zou makkelijk zijn als dit automatisch uit een lijst geïmporteerd kan worden.
In navolging van de vorige extra functionaliteit, zou het ook prettig zijn als dit systeem direct gekoppeld zou kunnen worden aan het leerlingen basisregister, de Dienst Uitvoering Onderwijs (DUO) en andere externe systemen.
Het zou ook prettig zijn als overal in het systeem gewerkt kan worden met klassen en leerjaren in plaats van losse leerlingen. Dit kan op dit moment wel op sommige plaatsen, zoals bijvoorbeeld bij het toewijzen van certificaten, maar nog niet consistent door de hele webapplicatie.
Een gevraagde functionaliteit die we hebben moeten laten vallen, is het toevoegen van uitgebreide bedrijfsgegevens en de mogelijkheid voor bedrijven om zichzelf te promoten bij de studenten. Dit is functionaliteit dat we graag nog zouden bouwen.
Graag hadden we ook nog een uitgebreid notificatiesysteem ingebouwd. Zo zou het bijvoorbeeld mooi zijn als de leerlingen een "warning tile" krijgen als ze nog een verslag in moeten leveren of als ze hun competentie rapport niet afgemaakt hebben. Ook kan het handig zijn om voor de deadlines automatisch gegenereerde herinneringen per email en sms te sturen. Het is op dit moment wel mogelijk om emails naar de leerlingen te sturen, maar dat moet handmatig met een extern mailprogramma gebeuren.

8.2 Optimalisaties

Verder zijn er nog wat vergaande optimalisaties die doorgevoerd zouden kunnen worden. Er zijn twee reden waarom deze optimalisaties niet doorgevoerd zijn. De hoofdreden was gebrek aan tijd om deze optimalisaties nog door te voeren. De andere reden is dat het programma altijd binnen honderd milliseconden reageerde in onze testomgeving. In een project als deze is tijd een schaars goed en in zo'n geval moeten er altijd keuzes gemaakt worden in wat er wel en niet gedaan wordt. Doordat het programma

al zo rap reageert vonden wij de optimalisaties van gebruikte processortijd niet zo belangrijk. Veel significanter wordt bijvoorbeeld een server te hebben op de Nederlandse backbone in plaats van in de VS. Het verschil van locatie kan makkelijk oplopen tot 2000 milliseconden. Met deze overwegingen in het achterhoofd zouden we de volgende optimalisaties nog wel door willen voeren.
De eerste optimalisatie die we willen doorvoeren, is het optimaliseren van de database queries. Op dit moment manipuleren wij de objecten van Entity Framework (EF) en laten we EF daarna zelf de queries genereren. Op zich is dat prima, maar er moet nog wel even nagekeken worden of de objectmanipulatie niet efficiënter kan op andere manieren en of er niet teveel naar en van de database gelezen wordt.
Verder heeft ASP.NET MVC een ingebouwde caching feature. Hiermee worden bepaalde pagina's tijdelijk als statische pagina in de cache opgeslagen. Als veel mensen op de zelfde pagina komen, hoeft op deze manier niet voor iedere persoon een nieuwe dynamische pagina aangemaakt te worden. Dat scheelt een hoop processortijd maar voornamelijk een hoop database queries. Om deze optimalisatie door te voeren, moet wel van elke pagina nagedacht worden over de "cachebaarheid" en de levensduur van die cache.
Ook hebben we een statische code analyse gedaan op onze code. Deze gaf nog een aantal suggesties die we niet hebben doorgevoerd. De belangrijkste hiervan was om de "assembly" van het model los te koppelen van de rest en deze een andere onderverdeling te geven. Omdat ons model echter niet zo groot was, en voornamelijk omdat de tijd er niet voor was, hebben we ervoor gekozen om deze optimalisatie te laten zitten. Het voordeel van deze optimalisatie is dat het makkelijker is om andere frontends te koppelen aan de backend, bijvoorbeeld een mobiele versie van de website.
Tot slot zouden er nog meer abstracties gebruikt kunnen worden. Tijdens het testen zijn we erachter gekomen dat hoe minder je eist van de aanroepende functie, hoe makkelijker het te testen is. Zo is bijvoorbeeld het gebruik van een generieke collectie interface die geen volgorde garandeert, makkelijker te implementeren dan het gebruik van een lijst die behalve volgorde ook nog verlangt dat alle objecten vergelijkbaar zijn en dus een vergelijkingsinterface implementeren.

8.3 Abstracties

Doorgaand op het laatste optimalisatie punt, zou het achteraf ook handig zijn geweest om gebruik te maken van Inversion of Control (IoC). IoC is een redelijk recente design pattern, bij IoC is het de bedoeling dat de programmeur eenmalig IoC container maakt en deze op alle plaatsen in het project gebruikt. In deze container kan dan voor elke interface een klas geregistreerd worden. Dit gebeurt met Register en Deregister functies. Als deze interface dan aangeroepen wordt, dan gebruikt de container een factory pattern om de bijbehorende class aan te maken. Bij wat uitgebreidere IoC containers is het mogelijk om alleen de bij de eerste aanroep een class aan te maken, of om deze al op voorhand aan te maken. Het voordeel hiervan is dat "lazy loading" verweven zit in de het IOC patroon. Verder is het mogelijk om voor het testen speciale "mocks" te registeren in de IoC container, deze worden dan automatisch in de hele applicatie gebruikt.

8.4 Testmogelijkheden

Zoals net genoemd zijn de IoC een en andere abstracties goede toevoegingen om de testbaarheid van de code te vergroten. Dat komt omdat hierdoor betere testisolatie kan worden gegarandeerd en daardoor preciezer kan worden nagegaan welk onderdeel eventueel niet goed werkt.
Een vorm van testen die we door het gebrek aan tijd hebben moeten laten vallen is Coded UI Testing. Deze vrij recente vorm van testen automatiseert het klikken op de gebruikersinterface. Hierdoor hoeft niet meer handmatig de website nagelopen worden en kunnen deze testen na elke wijziging gebruikt worden om te kijken of er niet per ongelijk functionaliteit gebroken wordt. Het nadeel van deze testen is dat ze allemaal eenmalig handmatig opgenomen moeten worden, inclusief de checks (asserts) die tijdens het gebruiken van de gebruikersinterface gedaan moeten worden. Op de langer termijn moet deze optimalisatie echter tijd op gaan leveren.

9 Nawoord

Met de hierboven genoemde conclusies en aanbevelingen kan aan de slag gegaan worden om verder te ontwikkelen aan dit programma. We willen dit nawoord gebruiken om iedereen te bedanken die heeft meegewerkt aan dit project: De opdrachtgever de heer Looij, de begeleider de heer Gross en de coordinator mevrouw Larson.

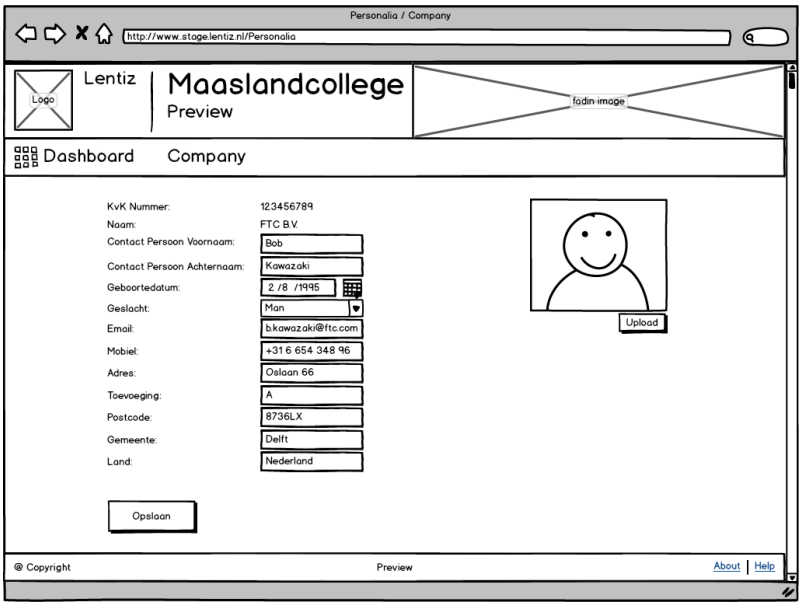
A Voorbeeld van een logfile

```
[X] Error messages moeten er overal uit
[ ] Debug ding moet weg
[X] De role van een staff moet gewijzigd worden
[X] Skill reports moeten nog helemaal gefixed worden
[X] Admin create

[X] title moet in nl
[X] onderaan back to list moet in het nl
[X] geslacht moet in het nederlands
[X] creer nieuw bedrijf, bij het opslaan moet je terug naar het bedrijven overzicht
[X] bij ongeldig email adres komt er een fout

[X] Admin kan zichzelf niet updaten de company contacts en personeelsleden ook niet
[X] Geavanceerd zoeken werkt niet, de substrings worden neit goed
benut zie search controller person regel 436
[X] Update weerk rapporten heeft een engelse popeup en hours worked moe tin het nederlands
[X] Skill rapport moet de popup ook in het nederlands
[X] Skill rapport "not all fields filled" popup moet in het nederlands
[X] skill rapport moet de conform knop onderaa de popup in het nederlands gezet worden
[X] Coordinator kan geen bedrijven verwijderen, voeg knop toe
aan personalia pagina van bedrijf om het bedrijf te verwijderen
[X] Faalt bij het uploaden van een vervang foto
[X] Debuggen verstuur wachtwoord om te kijken of er wel een mail verstuurd wordt

[X] Roles update
[X] Person update password
[X] Coordinator self change email bug with CurrentUser
[X] Adv search genders are not in dutch
[X] Update intern not working because email is undefined in JS
[X] Rapport hours worked not saved
[X] Export certificates zip name is "test.zip"
[-] Export certificates no popup when zip downloaded
[X] CompanyContact views company => dubble contacts
[X] For every type user, when report/skillreport deadline has passed,
it is still possible to fill and save them
[-] Uitloggen button visible on error page even when not logged in
[X] CompanyCotnact and Intern can see all info about staff
[X] Intern can see all info about CompanyContact
[X] Crash when user with no deadline opening report of Intern
[X] Lock lookuptable to fix bug when reloading a page fast and numerous times
```

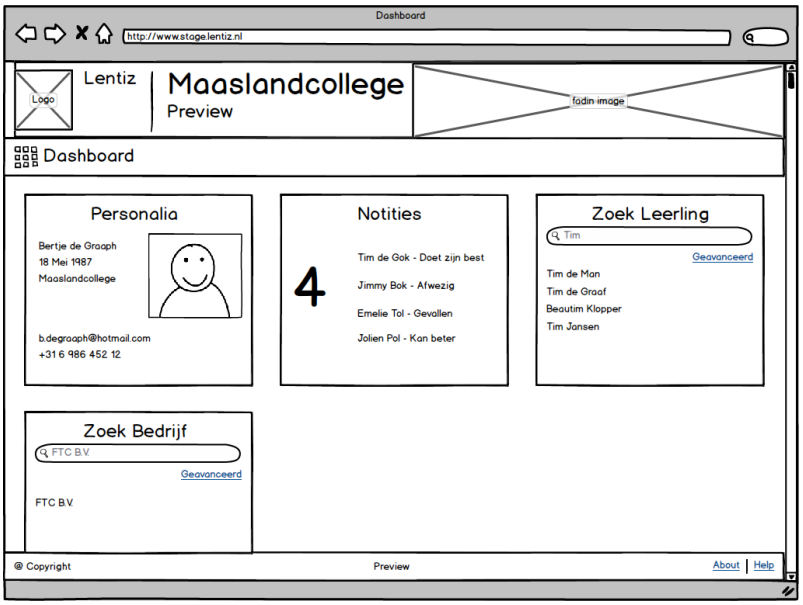
B Voorbeeld van een notulen

```
NOTULEN 24 MEI 2013

*--------------------
* START MEETING 8:30
*--------------------

WHAT WILL WE DO?

 - make css relation view (Thomas)
 - make relation view for other relations (Thomas)
 - Report Detail pane (David)
 - Report Detail routing (David)
 - Finish Seedmethod (Vincent)
 - Link model and view of skillscore (Vincent)

*------------------
* END MEETING 8:45
*------------------

*---------------------
* START MEETING 17:15
*---------------------

WHAT HAS BEEN DONE?
 - jquery draggable has been fixed
 - report detail pane has been routed
 - still working on detail pane (now detail popup)
 - Database has been designed and reviewed
  skillscoreRecord seeds have been inserted into the db

WHAT WILL WE DO NEXT TIME?
 - make css relation view (Thomas)
 - make relation view for other relations (Thomas)
 - Report Detail pane (David)
 - Finish Seedmethod (Vincent)
 - Link model and view of skillscore (Vincent)

*-----------
* END 17:30
*-----------
```

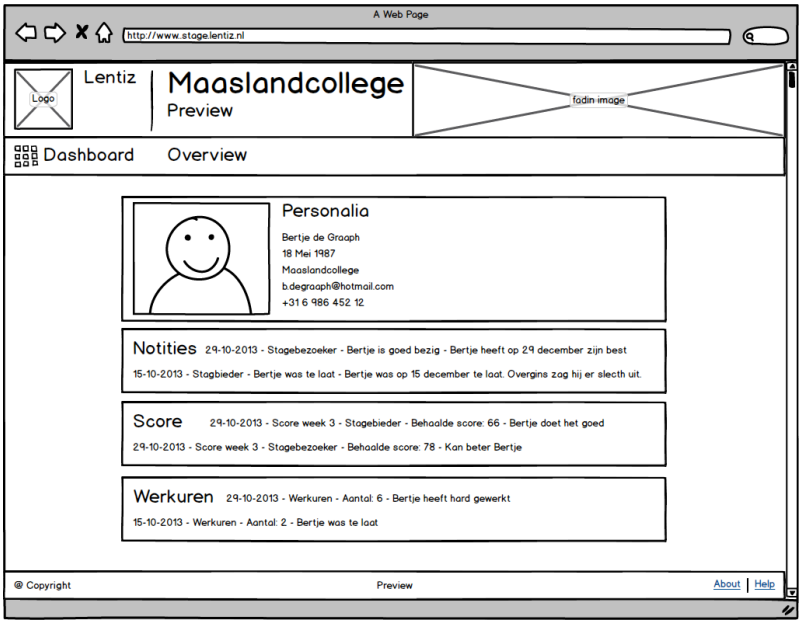
C Mockups



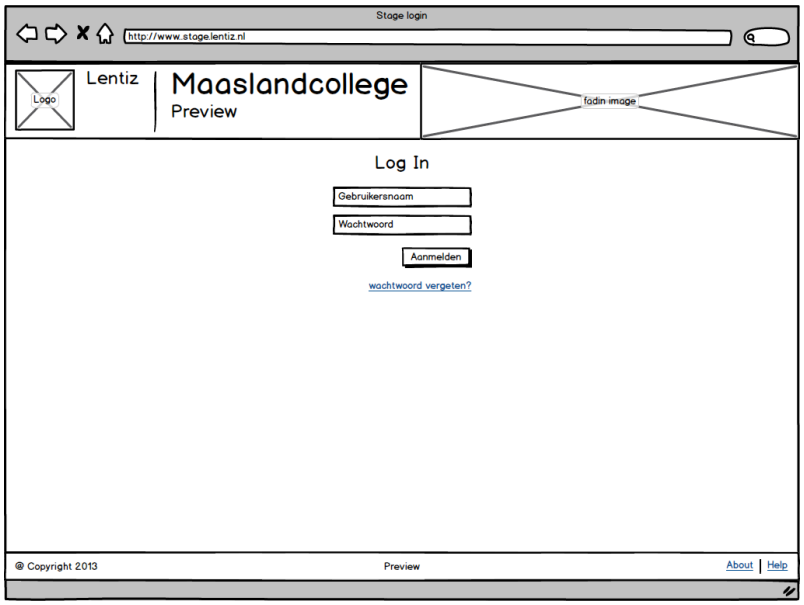
Figuur 6: Mockup van een bedrijfspagina



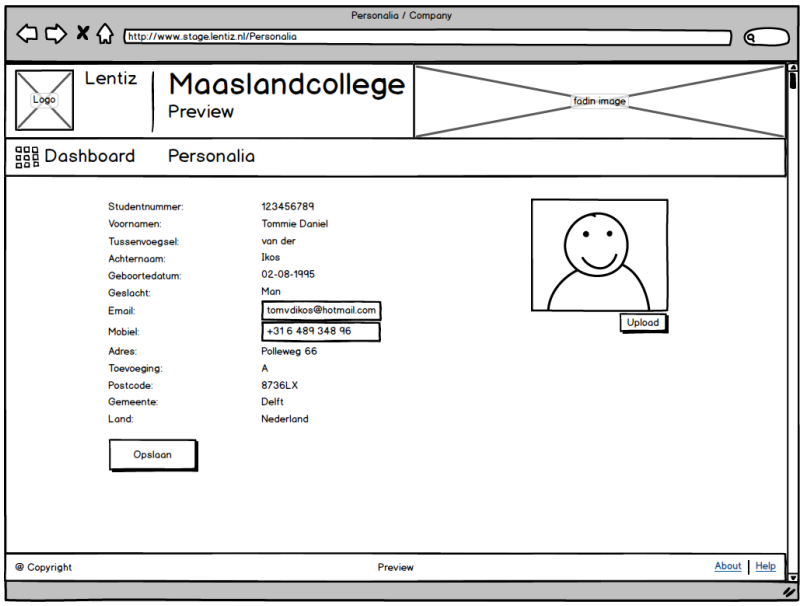
Figuur 7: Mockup van een dashboard



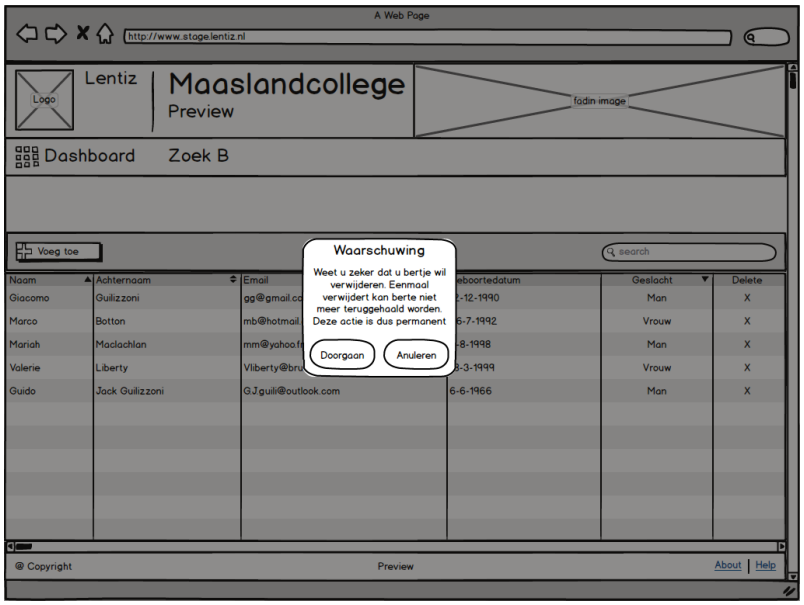
Figuur 8: Mockup van een algemeen overzicht



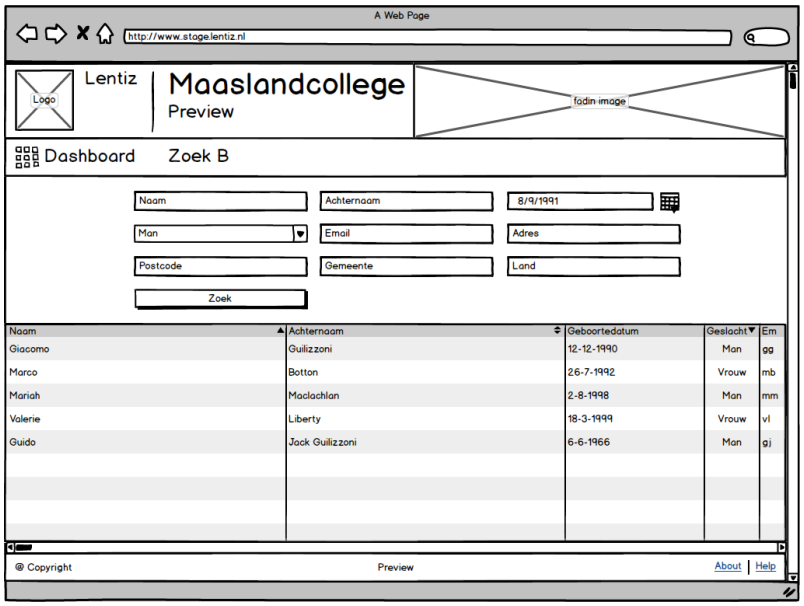
Figuur 9: Mockup van de login pagina



Figuur 10: Mockup van de personalia pagina

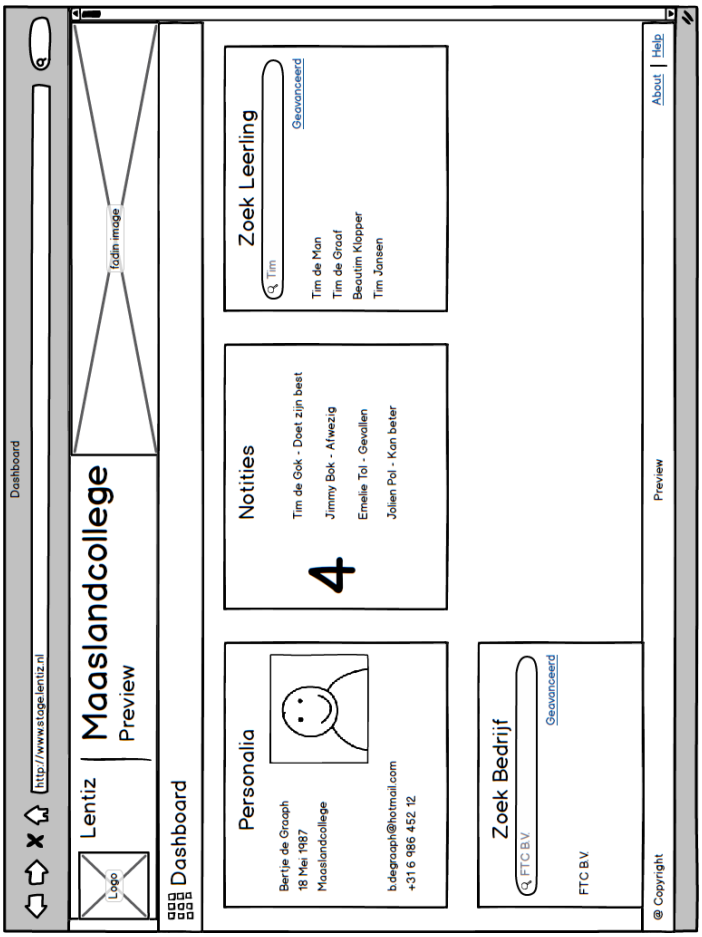


Figuur 11: Mockup van een popup

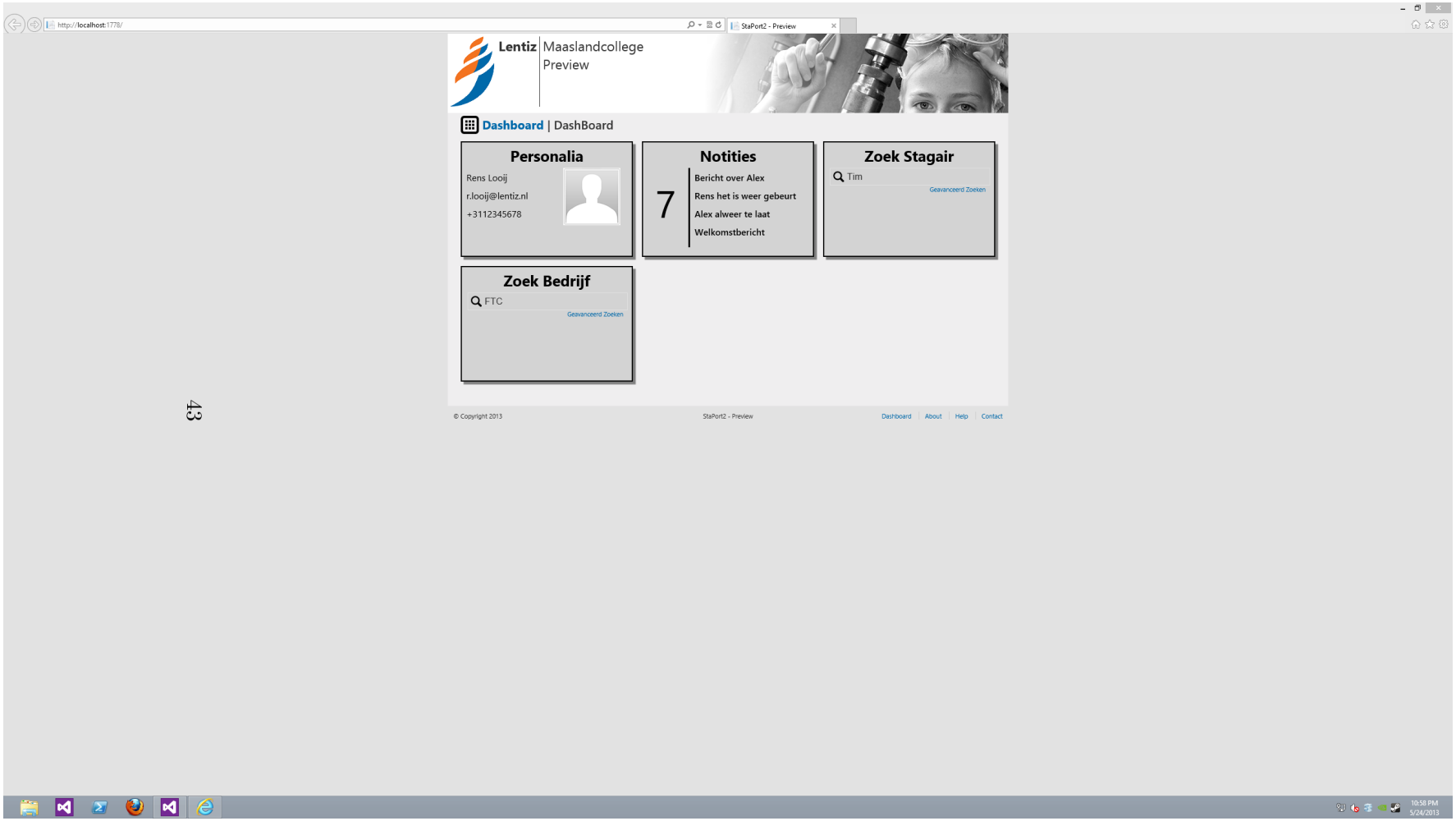


Figuur 12: Mockup van een zoekpagina

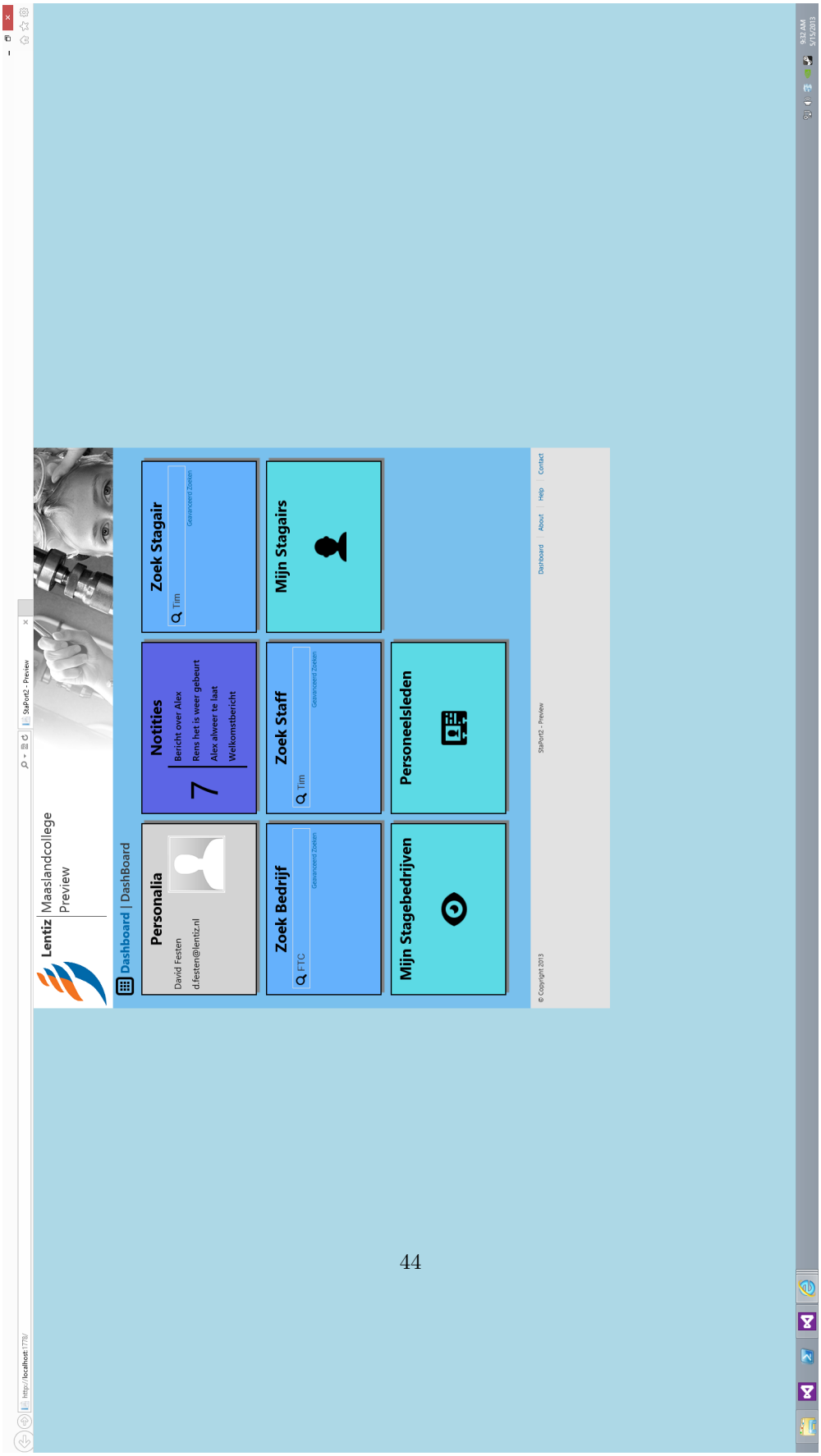
D GUI Progressie



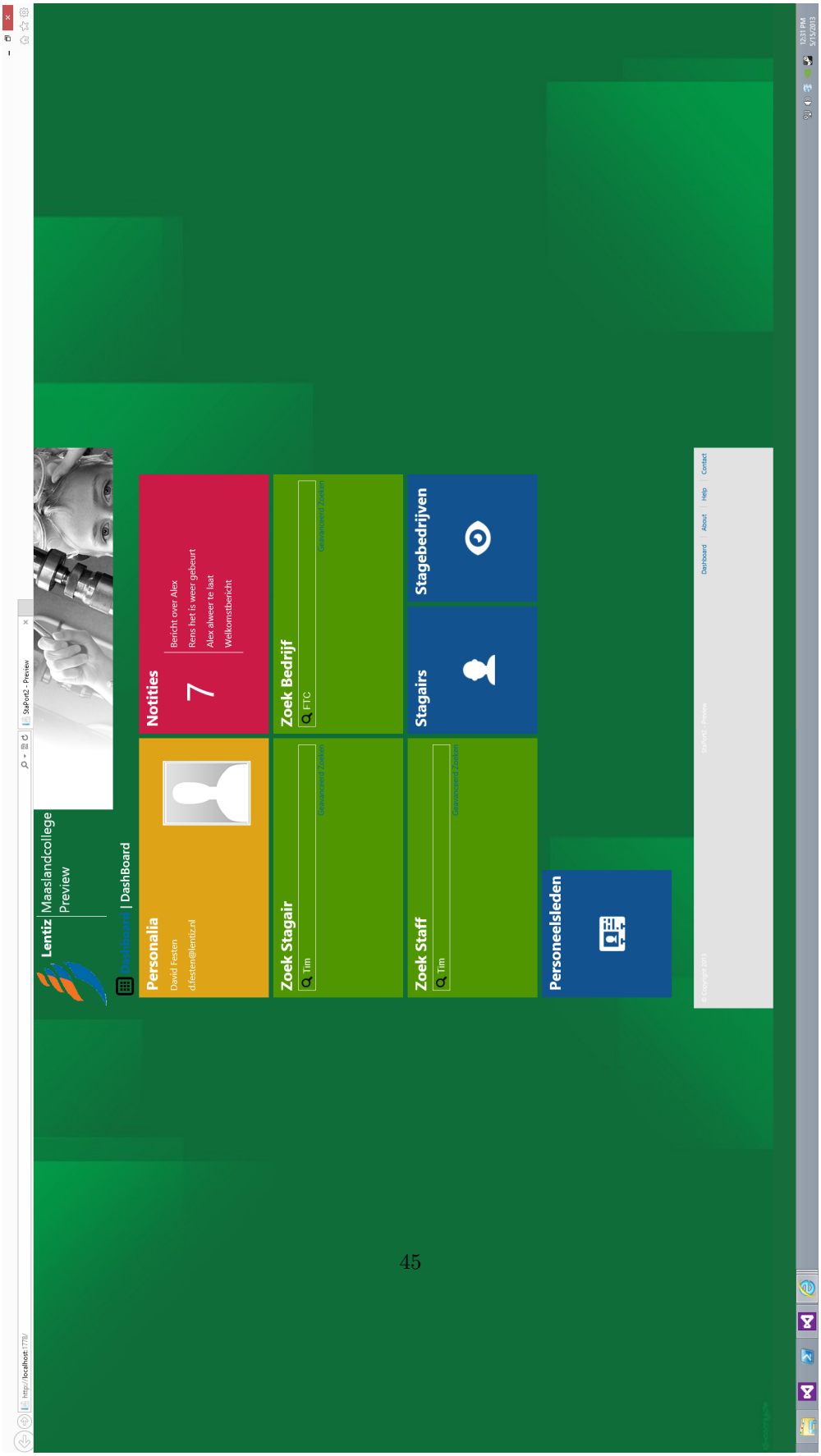
Figuur 13: GUI zoals in de mockup



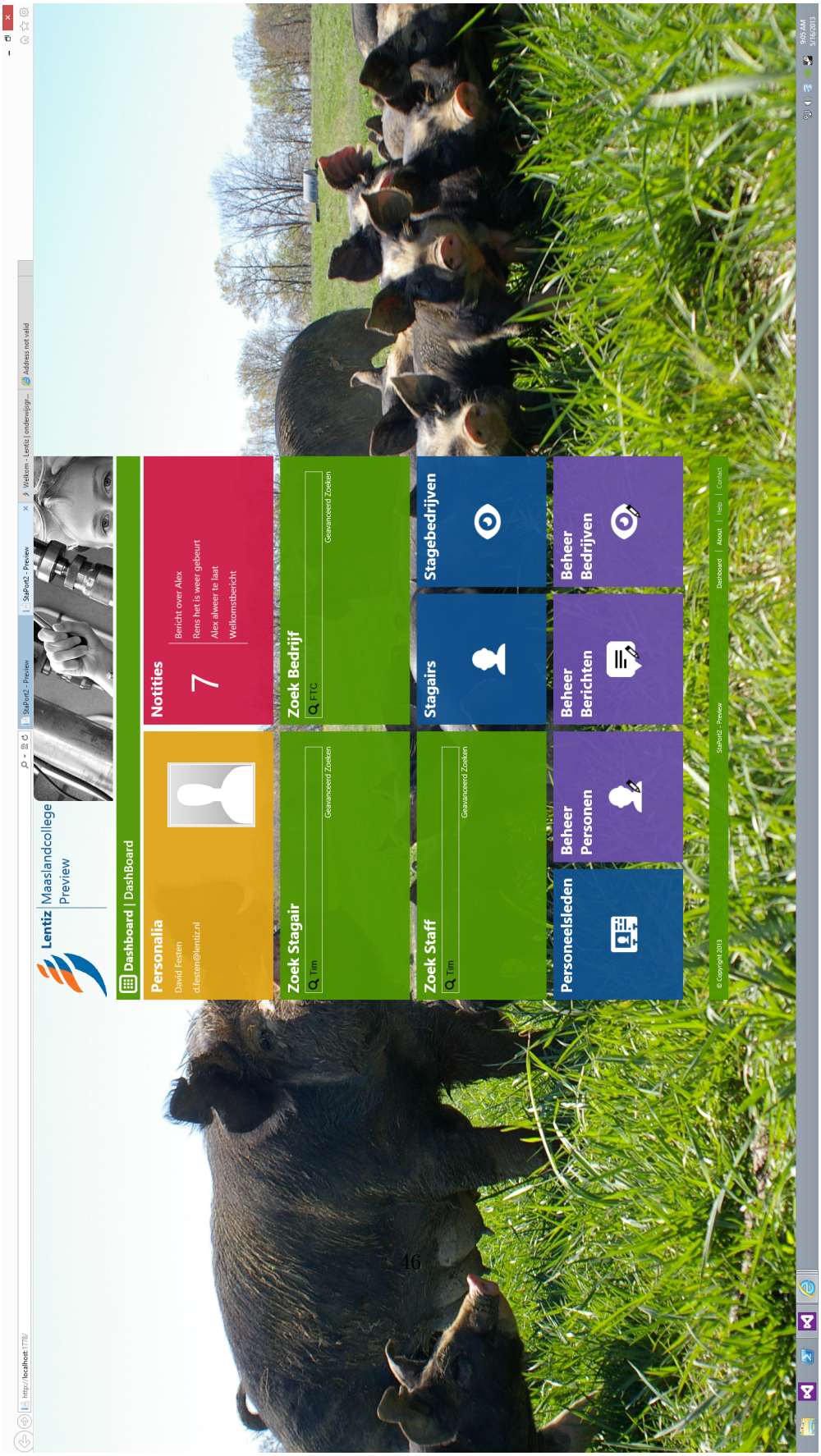
Figuur 14: GUI versie 1



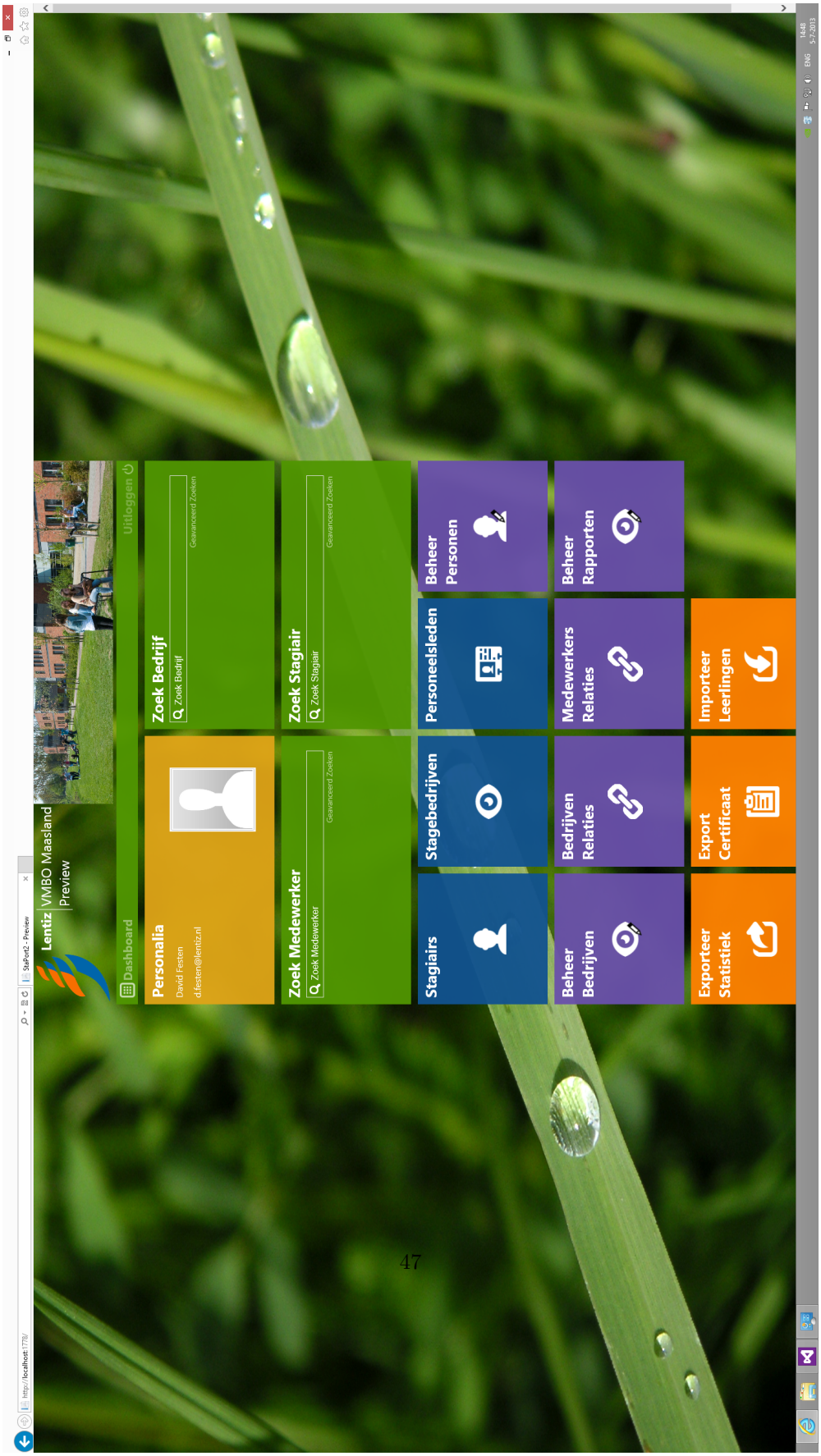
Figuur 15: GUI versie 2



Figuur 16: GUI versie 3



Figuur 17: GUI versie 4



Figuur 18: Definitieve GUI

E Eerste revisie van SIG

Beste David Festen,

Hierbij ontvang je mijn evaluatie van de door jou opgestuurde code. De evaluatie bevat een aantal aanbevelingen die meegenomen kunnen worden in de laatste fase van het project.
Deze evaluatie heeft als doel om studenten bewuster te maken van de onderhoudbaarheid van hun code en dient niet gebruikt te worden voor andere doeleinden.
Mochten er nog vragen of opmerkingen zijn dan hoor ik dat graag.
Met vriendelijke groet,
Eric Bouwers

Aanbevelingen
De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code gemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Unit Size, Unit Complexity en Module Coupling.
Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld de 'ImportPersonsFromExcel'-methode in de 'Import'-class, zijn aparte stukken functionaliteit te vinden welke gerefactored kunnen worden naar aparte methodes. Commentaarregels zoals bijvoorbeeld '// Convert the cell text to the corresponding Type' zijn een goede indicatie dat er een autonoom stuk functionaliteit te ontdekken is. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.
Voor Unit Complexity wordt er gekeken naar het percentage code dat bovengemiddeld complex is. Ook hier geldt dat het opsplitsen van dit soort methodes in kleinere stukken ervoor zorgt dat elk onderdeel makkelijker te begrijpen, makkelijker te testen en daardoor eenvoudiger te onderhouden wordt. In dit geval komen de meest complexe methoden ook naar voren als de langste methoden, waardoor het oplossen van het eerste probleem ook dit probleem zal verhelpen.
Voor Module Coupling wordt er gekeken naar het percentage van de code wat relatief vaak wordt aangeroepen. Normaal gesproken zorgt code die vaak aangeroepen wordt voor een minder stabiel systeem omdat veranderingen binnen dit type code kan leiden tot aanpassingen op veel verschillende plaatsen. In dit systeem wordt de class 'Person' op ruim 50 verschillende plaatsen aangeroepen. Daarnaast is deze class vrij fors. Het lijkt erop alsof deze class twee verschillende type functionaliteit bevat, enerzijds is het een representatie van een 'persoon' (en van een 'account'), maar de class bevat ook functionaliteit met betrekking tot de rol-gebaseerde toegang ('IsUserInAllRoles'). Om zowel de grootte als het aantal aanroepen te verminderen zouden deze functionaliteiten gescheiden kunnen worden, wat er ook toe zou leiden dat de afzonderlijke functionaliteiten makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden

worden.
Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase. De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

F Tweede revisie van SIG

In de tweede upload zien we dat het codevolume is gegroeid, terwijl de score voor onderhoudbaarheid licht is gestegen. Deze stijging wordt met name veroorzaakt door een (lichte) toename in de deelscores die in de vorige analye werden besproken: Unit Size, Unit Complexity en Module Coupling.
Het is goed om te zien dat naast een verbetering in de onderhoudbaarheid er ook nog steeds een stijging in het volume van de test-code te zien is.
Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject.

Referenties

[1] Ming Huo et al. Software quality and agile methods. 2004.

[2] Microsoft, 2013. URL `http://tfs.visualstudio.com`.

[3] Laurie Williams et al. Scrum + engineering practices: Experiences of three microsoft teams. 2011.

[4] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Proffesional, 1999.

[5] Apache. The kiss principle. URL `http://people.apache.org/~fhanik/kiss.html`.

[6] Microsoft. Design principles, . URL `http://msdn.microsoft.com/en-us/library/windows/apps/hh779072.aspx`.

[7] Fielding et al. Rfc 2616 protocol specification. URL `http://www.w3.org/Protocols/rfc2616/rfc2616.html`.

[8] Stackoverflow. Stackoverflow website. URL `http://stackoverflow.com`.

[9] Microsoft. Asp.net mvc 4 source, . URL `http://aspnetwebstack.codeplex.com/SourceControl/latest`.

TU Delft

TI3800 Bachelorproject

Orientatieverslag

Lentiz | Maasland college Staport v 2

Schrijvers:
David Festen,
Vincent Ghiëtte,
Thomas Valera

Begeleider:
Hans-Gerhard Gross

Opdrachtgever:
Rens Looij

11 Juli 2013

Abstract

In dit verslag wordt het vooronderzoek van het stage registratie systeem, dat als bachelorproject bij het Maaslandcollege uitgevoerd wordt, beschreven. Dit oriëntatie verslag is geschreven nadat de opdracht duidelijk was. In dit oriëntatie verslag vergelijken we verschillende frameworks van verschillende talen, onder andere ASP.NET MVC, Zend Framework, Django. Daarna maken wij hieruit een onderbouwde keuze.

Inhoudsopgave

1 Inleiding

Dit is het vooronderzoek voor het bacheloreindproject bij het Lentiz | Maaslandcollege. In dit project gaan we een stage registratie systeem maken. Dit is het oriëntatie verslag waarin uitgezocht wordt wat de setting is waarin het programma moet gaan draaien en welke technieken bruikbaar zijn. Ook omschrijven we het huidige systeem en wat de tekortkomingen zijn die de aanleiding zijn tot het bouwen van een nieuw systeem.

2 Lentiz | Maaslandcollege

De Lentiz onderwijsgroep verzorgt voortgezet onderwijs (vmbo, havo, vwo) en middelbaar beroepsonderwijs (mbo). Het onderwijs wordt aangeboden zowel in voltijd- als in deeltijdopleidingen en ook in de vorm van contractactiviteiten. Het ontwikkelen van de mogelijkheden en het benutten van capaciteiten van de leerlingen en studenten staan hierbij hoog in het vaandel.
De Lentiz onderwijsgroep verzorgt voortgezet onderwijs op interconfessionele grondslag met algemene toegankelijkheid en onderwijs op algemeen bijzondere grondslag waarbij ruimte wordt geboden voor het karakter van het openbaar onderwijs.
De Lentiz onderwijsgroep bestaat uit totaal 11 scholen. Het onderwijs binnen de Lentiz onderwijsgroep is divers van aard, waarbij ieder onderwijssoort zijn specifieke kenmerken binnen de eigen omgeving heeft:

- Vmbo en mbo in onder andere de groene sector worden verzorgd in het Westland en Schiedam.
- Vmbo (Zorg & Welzijn, Techniek en Economie), havo en vwo worden verzorgd in de regio Nieuwe Waterweg Noord.

Maasland college is een groene vmbo onderwijsinstelling onderdeel van de Lentiz groep. De opdrachtgever is de heer R. Looij. Op het Maaslandcollege in klas 4 wordt gedurende het schooljaar een dag per week stage gelopen. Deze stage, waardoor de leerling een beter inzicht krijgt in een beroep of sector, is een verplicht onderdeel van het lesprogramma.[1][2]

2.1 Huidig systeem

De huidige werkwijze heeft enige beperkingen: Het bestaat uit drie losse systemen, die verder niet centraal worden bijgehouden.
Allereerst is er het logboek, dit wordt beheerd door de stagiair zelf. Deze neemt het logboek mee naar het stage bedrijf en naar school. Dit heeft twee doelen, namelijk het enerzijds mogelijk maken van communicatie tussen de school en het stagebedrijf, en het anderzijds bijhouden van de voortgang van de desbetreffende stagiair (Appendix A Figuur 3).
Ten tweede is er Excel spreadsheet waarin bijgehouden wordt welke stagiair welke stages doen. In deze spreadsheet worden de volgende gegevens bijgehouden: de gegevens van de stagiair, stagebedrijf, stagebezoeker (beoordelend docent), stagecoördinator en mentor. Het stagebedrijf bied een werkplek aan de stagiair om werkervaring op te doen. Tijdens de stageperiode controleert de stagebezoeker de voortgang van de stagiair. De stagecoördinator beheert de stages van alle stagiairs. De mentor heeft weinig met de stage te maken en wordt vooral ter referentie genoemd in de spreadsheet. In het huidig systeem is er één spreadsheet voor alle stagiairs (Appendix A Figuur 4).
Ten derde hangt er een lijst in de school waarop suggesties staan voor mogelijke stageadressen. De leerlingen kunnen daar een stagebedrijf uitzoeken en zich voor een stage opgeven bij de coördinator. Deze lijst bevat slechts suggesties voor stagebedrijven en

wordt in een spreadsheet bijgehouden.

2.2 Waarom het verandert moet worden

De huidige situatie levert problemen op. Door het gebruik van een logboek kan er gemakkelijk gefraudeerd worden. Ook kan dit kwijtgeraakt worden door de stagiair, stagebedrijf of stagecoördinator. Verder wordt het logboek aan het einde van de stage gearchiveerd en gedigitaliseerd wat voor overhead zorgt.
Het gebruik van een spreadsheet voor het bijhouden van welke stagiairs naar welke stage gaan levert ook wat problemen op. Zo is het mogelijk dat er meerdere versies van deze spreadsheet bestaan, er is namelijk geen centrale opslag. Dit zorgt ervoor dat deze spreadsheet snel onoverzichtelijk wordt.
De lijst met stageadressen op school biedt geen eenvoudige mogelijkheid voor de stagebedrijven om stages aan te bieden. Verder kunnen leerlingen op deze lijst geen gedetailleerde informatie vinden.
Tot slot is er nog een overkoepelend probleem: Er zijn drie aparte informatiesystemen nodig om stages te regelen. Hierdoor is er geen centrale plek waar alle informatie te vinden is, omdat de verschillende bronnen niet aan elkaar gekoppeld zijn. Als bijvoorbeeld het email-adres van de contactpersoon van de stagebieder gewijzigd wordt in de spreadsheet, dan is dit niet zichtbaar op de lijst die in de schoolgang hangt.

2.3 Een stage registratie applicatie

Zoals blijkt uit de hiervoor geschreven paragrafen, is het uitrollen van een digitaal stage registratie systeem een onvermijdbare stap voor het beter begeleiden van de leerlingen tijdens de stage perioden. Een dergelijke applicatie moet ervoor zorgen dat zowel de leerlingen, docenten en stagebedrijven een beter overzicht krijgen van het stageverloop van de leerling. Dit willen wij realiseren door een systeem te ontwikkelen waar de data centraal opgeslagen wordt en ten alle tijden bereikbaar is. Verder is het ook wenselijk dat de applicatie op een breed spectrum van hardware kan draaien. Er moet dus rekening gehouden worden met een scala aan operatie-systemen en de hardware waarop deze draaien. Daarbij is het van belang dat de applicatie veilig is. Met name dat het niet fraude gevoelig is, dat is immers een van de redenen voor het overstappen van het huidig systeem naar een digitaal systeem.

3 Technische Oriëntaties

3.1 Type applicatie

Tijdens het onderzoek zijn er drie type applicaties naar voren gekomen. Dit zijn de desktop applicatie, web applicatie en de mobile applicatie. Deze drie type applicatie hebben allen hun voor- en nadelen.

3.1.1 Desktop applicatie

Pro:

- UI makkelijker te realiseren
- Kan native de hardware aanspreken
- Computers zijn vaak beschikbaar op de werkplekken van de gebruikers
- Kan geïntegreerd worden in andere desktop applicaties

Con:

- De gebruiker moet rechten hebben op de computer
- Er moet rekening gehouden worden met een scala aan hardware en os-en
- Bij het update van de applicatie moet de gebruiker zelf de software updaten

3.1.2 Mobile applicatie

Pro:

- Er kan overal mee gewerkt worden

Con:

- Een smartphone of tablet is vereist
- Er moet rekening gehouden worden met een scala aan hardware en os-en
- Bij het update van de applicatie moet de gebruiker zelf de software updaten

3.1.3 Web applicatie

Pro:

- Geen installatie is vereist
- De gebruiker heeft geen andere rechten nodig dan het browsen op het internet
- Niet hardware of os afhankelijk

- Makkelijk te updaten

Con:

- De weergave is browser afhankelijk
- Hardware is niet direct aan te spreken

3.1.4 De keuze

De voorgaande analyse suggereert dat het maken van een mobile applicatie niet wenselijk is voor de opdrachtgever. Dit omdat de gebruikers dan in het bezit moeten zijn van een smartphone of een tablet. Die is een restrictie niet niet opgelegd kan worden aan de gebruikers van het systeem. Door deze restrictie valt de optie mobile applicatie al af. Er blijven dus twee kandidaten over voor het type applicatie.
Om een beter beeld te krijgen van de daadwerkelijk behoeften van de gebruikers te krijgen is er gebruik gemaakt van een enquête, zie Appendix B. De resultaten van deze enquête uit Appendix B Tabel 3, wijzen erop dat de gebruiker meer baat bij een webapplicatie zou hebben dan bij een desktop applicatie.[3]
Los van de enquête is er ook een tweede zwaar weegpunt in het voordeel van de webapplicatie. Dit is namelijk het feit dat de gebruikers van het systeem vaak op computers werken die afgeschermd zijn tegen de gebruikers. Dit betekend dat het installeren of zelfs starten van een applicatie moeizaam kan verlopen.
In overweging van al het voorgaande is er gekozen om een webapplicatie aan te bieden.

3.2 Soortgelijke applicaties

Er bestaan al verscheidene applicatie op het gebied van stage registratie systemen, het merendeel daarvan zijn webapplicaties. Een van de grote spelers op de markt is OnStage [4], een uitgebreid stage begeleidingssysteem. Een andere applicatie is Parsys [5]. Deze applicaties zijn echter allemaal gericht op het hoger onderwijs en niet op het voortgezet onderwijs. Verder zijn deze applicatie erg uitgebreid wat tot een steile leercurve leidt.
Hierop spelen wij in door een applicatie die speciaal gericht is op het voortgezet onderwijs te ontwerpen die functionaliteit bevat dat speciaal op het Maaslandcollege is gericht.

3.3 Mogelijke programmeertalen

Er is gekozen voor het maken van de applicatie in de vorm van een web applicatie. Dit brengt met zich mee dat er verschillende talen gebruikt kunnen worden om het programma te schrijven. Omdat het om een web applicatie gaat worden de gebruikte talen in twee categorieën gesplitst, client- en serverside.

3.3.1 Clientside

Om het aantal talen waaruit geselecteerd kan worden te minimaliseren zijn de meest populaire talen in de vergelijking meegenomen. Het gaat dus om de volgende talen:

- Actionscript voor Flash [6]
- Javascript [7]
- HTML & CSS [8]
- C# of Visual Basic voor Silverlight [9]

Flash is afhankelijk van een meestal niet in de browser geïntegreerd stuk software. Daarom is besloten, gezien de geslotenheid van de computers, om hier van af te zien. Dat is ook het geval voor Silverlight. Dus rest on nog met de keuzen voor HTML & CSS en Javascript. Aangezien we een webapplicatie willen maken is het cruciaal om HTML te gebruiken met CSS voor de styling. Het gebruik van Javascript is niet noodzakelijk maar wel wenselijk omdat de website dan dynamischer en gebruiksvriendelijker gemaakt kan worden.
Er is dus gekozen om HTML & CSS en Javascript te gebruiken om de clientside van de webapplicatie te maken.

3.3.2 Serverside

Voor de server implementatie van de webapplicatie kan er ook uit verschillende talen gekozen worden. Wederom zijn de meest populaire talen gekozen om de keuze te verfijnen.

- PHP [10]
- Java [11]
- C++ [12]
- C# [13]
- Python [14]
- Ruby [15]

De genoemde talen zijn allemaal geschikt voor het implementeren van de logica achter de webapplicatie. Er moet dus gekeken worden naar de frameworks en de libraries die gebruikt kunnen worden om de ontwikkeltijd te kunnen inperken.

3.4 Bestaande frameworks

Wederom zijn er een scala aan frameworks aanwezig voor de eerdergenoemde talen. Daardoor zullen we een selectie maken van de meest populaire.

- ASP.NET MVC [16]
- Zend [17]
- CodeIgniter [18]
- Cake PHP [19]
- CppCMS [20]
- Ruby on Rails [21]
- Spring [22]
- Django [23]

Deze frameworks zijn vergelijk op verschillende punten zoals weergegeven in Appendix C in de Tabel 4. uit deze vergelijking zien we dat de meest complete frameworks ASP.NET, Zend, CakePHP, RubyOnRails en Django zijn. Hierdoor komen de overige frameworks te vervallen. Omdat er nog steeds een ruim aanbod aan talen overblijft, met name PHP, Python, Ruby en C# moeten we verder kijken naar de ontwikkel omgevingen die deze talen ondersteunen.

3.5 Ontwikkel omgeving

Wederom zijn er veel kandidaten voor de ontwikkel omgevingen. Weer zijn de meest populaire gekozen om de vergelijking aan te gaan.

- Visual Studio [24]
- Eclipse [25]
- Netbeans [26]

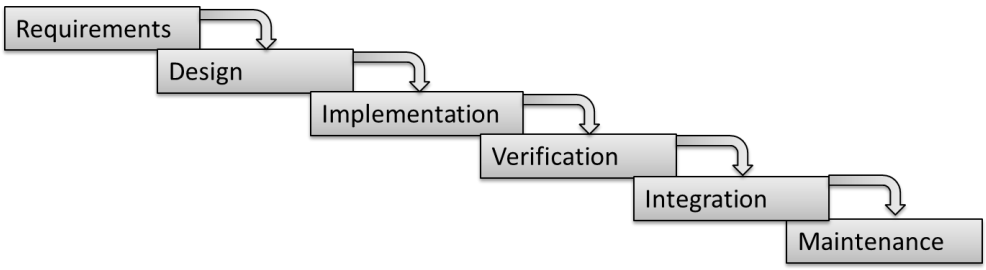
Alhoewel al deze ontwikkelomgevingen voldoen aan onze eisen, prefereren wij toch Visual Studio. Dit omdat in Visual Studio Source control, Test suites, Mocking, Code analysis, Performance analysis, et cetera en de frameworks zelf geïntegreerd zijn. Bij Eclipse en Netbeans moeten er veel plugins gezocht en ingeladen worden. Denk hierbij aan EclEmma, Jenkins, Codestyle et cetera.

3.6 Ontwikkel methode

Voor de ontwikkel methode hebben we twee kandidaten onderzocht. Het waterval model, Figuur 1, en het agile model, Figuur 2.

3.6.1 Waterval

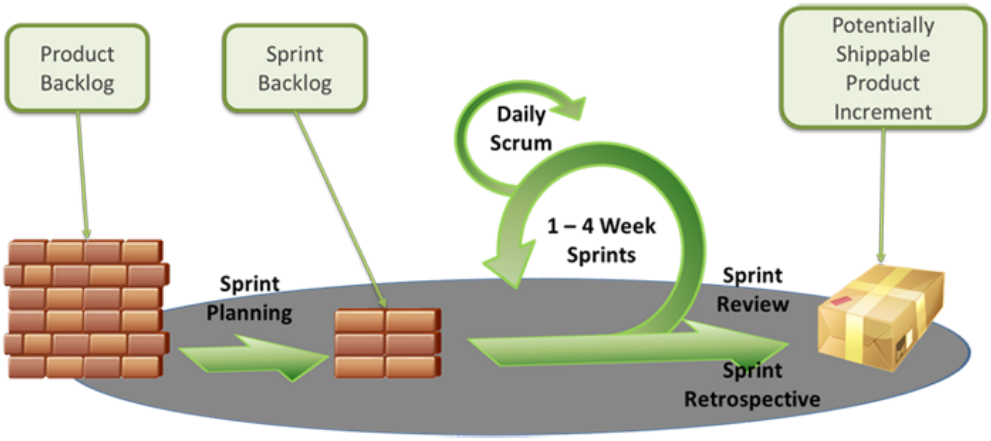
Het waterval model kent in het algemeen zes stappen. Elke volgende stap wordt pas begonnen nadat de vorige stap voltooid is. Dit model begint met de het opstellen van alle eisen. Zodra dit gebeurt is worden de technische diagrammen van de applicatie ontworpen (UML). Als deze technische ontwerpen geverifieerd zijn word er begonnen aan het implementeren van losse modules van het programma, dit is het daadwerkelijk coderen. Nadat de implementatie voltooid is, worden de losse onderdelen geverifieerd door middel van testen. Zodra deze testen allemaal slagen word gekeken naar de integratie van de losse onderdelen. Als integratie voltooid is kan het programma uitgerold en onderhouden worden. Bij grote onderhoud beurten moet een nieuw waterval model opgezet worden.



Figuur 1: Schema van het waterval model

3.6.2 Scrum

Scrum verschilt van het waterval model. Bij Scrum word een lijst van features opgesteld (Product backlog). Hieruit worden een aantal features geselecteerd voor de huidige iteratie van het product (Sprint backlog). Per sprint worden er voor de gekozen features het ontwerp, implementatie, verificatie en integratie gedaan. Waarna er een werken product is. Na deze sprint voltooid is word er een andere sprint gedefinieerd. Dit proces herhaalt zich totdat alle features uit de Product backlog geïmplementeerd zijn. Bij een grote Product backlog kan ervoor gekozen worden om nog een Release backlog te maken met een subset aan features die af moeten zijn voor een bepaalde release. Bij een klein project zijn de Product en Release backlog gelijk. [27]



Figuur 2: Schema van het scrum model

3.6.3 Gekozen methode

We hebben besloten om het scrum model te gebruiken omdat we dan demo's kunnen laten zien aan de opdrachtgever en met de feedback daarvan het ontwikkelproces kunnen bijsturen. Scrum is een moderne manier van ontwikkelen dat in opkomst is en wij willen dit graag leren in het project.

4 Conclusie

Er is besloten om een webapplicatie te maken zodat deze overal beschikbaar is voor de gebruikers. Verder hebben wij gekozen voor ASP.NET MVC in combinatie met HTML & CSS en Javascript omdat de bijbehorende ontwikkelomgeving, Visual Studio, de meest complete is. Tot slot hebben wij gekozen voor de scrum ontwikkelmethode om hier ervaring mee op te doen en tijdig feedback te krijgen.

A Huidig systeem

Woensdag 4 september (*dag 1*)

Ik heb de volgende werkzaamheden gedaan:

Wat me is opgevallen:

Ik heb de volgende dingen geleerd:

Ik heb samen met mijn begeleider de volgende dingen afgesproken, die ik anders ga aanpakken, beter ga doen, etc.:

De beoordeling voor vandaag was:

Leerling was vandaag op tijd? o ja o nee

gewerkt van .....uur tot.....uur = ...uur (excl. pauze)

Handtekening bedrijfscoach	Handtekening begeleidende docent

Eventuele opmerkingen docent:

Figuur 3: Voorbeeld van een weekverslag

Gegevens stagiair								Stagebieder								Stagebezoeker					Stagecoordinator					Mentor				
Klas	Voornaam	Tussenvoegsel	Achternaam	Geboortedatum	Geboorteplaats	E-mailadres	Telefoonnummer	Naam bedrijf	Straat	Nummer	Postcode	Plaats	Contactpersoon	Telefoonummer	E-mailadres	Naam	Tussenvoegsel	Achternaam	Telefoonnummer	E-mailadres	Naam	Tussenvoegsel	Achternaam	Telefoonnummer	E-mailadres	Naam	Tussenvoegsel	Achternaam	Telefoonnummer	e-mailadres

B Web versus desktop development factors

Tabel 1: Primary questions

Questions	Score (1 = yes, 0 = No)
Are your users comfortable using a Web browser?	1
Are your users located in remote sites?	1
Do your users in remote sites have access to the Internet?	1
Are you creating a Business to Business (B2B) application?	0
Are you creating a Business to Consumer (B2C) application?	1
Is the amount of data entered minimal?	1
Is the amount of data to display on the screen minimal?	1
Is the number of fields on the screen fairly small?	1
Does each user öwn"their data?	0
Are the same rows of data rarely updated by multiple users at the same time?	0
Is this application mainly for light data entry, where speed of data entry isn't critical?	1
Is there a lot of data review that requires "tall"pages?	0
Do your users like to scroll through the data, as opposed to tabbing through data?	0
Are there minimal data items on a screen that cause other data to change on that same screen?	0
Can your users minimize the need to exchange data dynamically with other products running on the same desktop?	1
Is performance a secondary consideration?	1
Do your developers (or you) want to develop for the Web?	1
Do your developers (or you) have the skills to develop for the Web (or can they quickly learn how)?	1
Do you want a very graphically appealing look and feel?	1
Do you have a lot of large screens that would warrant scrolling windows?	0
Is it important to keep deployment costs to a minimum?	1
Is it important to keep upgrade costs to a minimum?	1
Will there be frequent updates to software?	0
Can you hire/train Web programmers more cheaply than desktop programmers?	0
Do investors and/or shareholders want a Web application?	1
Would your users prefer a browser interface to a desktop application?	0

Continued on next page

Tabel 1 – *Continued from previous page*

Questions	Score (1 = yes, 0 = No)
Do users in remote sites have a high-speed connection to your internal network?	1
Is it fairly easy to install Internet access in remote sites?	1
When your users travel, do they usually have access to the Internet?	1
Is this application only for one department?	1
Subtotal for Primary Questions	20

Tabel 2: Secondary questions

Questions	Score (1 = yes, 0 = No)
Is there a need to connect to special hardware?	0
Do you need Drag-and-Drop support in this application?	0
Are you designing a game, CAD, or CAM application?	0
Do you need a lot of special controls for limiting data input (such as input masks)?	0
Can deployment of this system be done through a network, by distributing CDs, or using push servers?	1
Can upgrades of this system be done through a network, by distributing CDs, or using push servers?	1
Subtotal for Secondary Questions	2

Tabel 3: Now Do the Math!

Question serie	Score
Subtotal for Primary Questions	20
Deduct Score for Secondary Questions	2
Total Score	18 / 30

C Vergelijking frameworks

Tabel 4: Comparison of Frameworks

Framework	Ajax	MVC	i18n and L10n	ORM	Testing	DB migration	Security	Template	Form validation
ASP.NET MVC	yes	yes	yes	yes	yes	yes	yes	yes	yes
Zend	yes	yes	yes	yes	yes	yes	yes	yes	yes
CodeIgniter	yes	yes	mostly	no	yes	yes	yes	yes	yes
CakePHP	yes	yes	yes	yes	yes	yes	yes	yes	yes
CppCMS	yes	yes	yes	yes	no	no	yes	yes	yes
Ruby On Rails	yes	yes	yes	yes	yes	yes	yes	yes	yes
Spring	yes	yes	yes	yes	yes	no	yes	yes	yes
Django	yes	yes	yes	yes	yes	yes	yes	yes	yes

MVC: Model view controle
i18n and L10n: Internationalization and localization support
ORM: Object-relational mapping

Referenties

[1] Lentiz onderwijsgroep. Lentiz, January 2013. URL http://www.lentiz.nl/.

[2] Lentiz onderwijsgroep. Maaslandcollege, January 2013. URL http://www.lentiz.nl/maaslandcollege.

[3] Paul D. Sheriff. Designing for web or desktop?, January 2002. URL http://msdn.microsoft.com/en-us/library/ms973831.aspx.

[4] Xebic Onderwijs B.V. Onstage, 2013. URL http://www.bpvsoftware.nl/.

[5] Parview Interim-Management en Advies. Parsys, 2013. URL http://www.parview.nl/.

[6] Adobe. Actionscript, 2013. URL http://www.adobe.com/devnet/actionscript.html.

[7] Mozilla Developer Network. Javascript, 2013. URL https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[8] World Wide Web Consortium. W3c html, 2013. URL http://www.w3.org/html/.

[9] Microsoft. Silverlight, 2013. URL http://msdn.microsoft.com/en-us/library/cc838158%28v=vs.95%29.aspx.

[10] The PHP Group. Php, 2013. URL http://php.net/.

[11] Oracle Corporation. Java, 2013. URL http://www.oracle.com/us/technologies/java/overview/index.html.

[12] The C++ Standars Committee. C++, 2013. URL http://www.open-std.org/jtc1/sc22/wg21/.

[13] Microsoft. C#, 2013. URL http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx.

[14] Python Software Foundation. Python, 2013. URL http://www.python.org/.

[15] Ruby community. Ruby, 2013. URL http://www.ruby-lang.org/en/.

[16] Microsoft. Getting started with asp.net mvc, 2013. URL http://www.asp.net/mvc.

[17] Zend Technologies Ltd. Why use zend framework, 2013. URL http://framework.zend.com/.

[18] EllisLab Inc. Codeigniter, 2013. URL http://ellislab.com/codeigniter.

[19] Cake Software Foundation Inc. Why use cakephp, 2013. URL http://cakephp.org/.

[20] CppCMS Comunity. Cppcms, 2013. URL http://cppcms.com/wikipp/en/page/main.

[21] David Heinemeier Hansson. Ruby on rails, 2013. URL http://rubyonrails.org/.

[22] GoPivotal Inc. What is spring?, 2013. URL http://www.springsource.org/.

[23] Django Software Foundation. Meet django, 2013. URL https://www.djangoproject.com/.

[24] Microsoft. Visual studio, 2013. URL http://www.microsoft.com/visualstudio/eng.

[25] The Eclipse Foundation. Eclipse, 2013. URL http://www.eclipse.org/.

[26] Oracle Corporation. Netbeans, 2013. URL https://netbeans.org/.

[27] Ming Huo et al. Software quality and agile methods. 2004.

TU Delft

TI3800 Bachelorproject

Plan van Aanpak

Lentiz | Maasland college Staport v 2

Schrijvers:
David Festen,
Vincent Ghiëtte,
Thomas Valera

Begeleider:
Hans-Gerhard Gross

Opdrachtgever:
Rens Looij

11 Juli 2013

Voorwoord

In dit document is het plan van aanpak weergegeven. Dit document kan gezien worden als een 'contract' waarin de probleemstelling wordt vastgelegd en hoe dit opgelost gaat worden. Er wordt vastgelegd waaraan de opdrachtgever en de projectleden moeten voldoen. Bovendien wordt beschreven hoe het project gefaseerd zal worden en worden overige afspraken en beslissingen hier ook vastgelegd.

Samenvatting

In dit plan van aanpak zullen wij beschrijven hoe het project, een stage registratie systeem voor het Maaslandcollege, gemaakt zal worden. De belangrijkste eis die aan dit project gesteld word vanuit het Maaslandcollege is dat het stageverloop van de leerlingen gevolgd kan worden op een gecentraliseerde en geautomatiseerde manier.
Dit project zal in elf weken gerealiseerd worden. Tijdens het project zal een systeem gemaakt worden dat aan de eisen van de opdrachtgever voldoet en conform is aan de uiteindelijke doelstelling. Om dit te waarborgen zal er nauw samengewerkt worden met de opdrachtgever. Daarbij zal er gebruik gemaakt worden van Agile ontwikkeltechnieken.

Inhoudsopgave

1 Introductie

De opdracht van dit project komt van Lentiz | Maaslandcollege. Lentiz is een scholengemeenschap van vmbo en mbo scholen. Lentiz heeft ons gevraagd om een digitaal stage registratie systeem te maken. Op dit moment gebruiken ze daarvoor een papieren logboek en een verzameling van Excel spreadsheets. De huidige werkwijze is decentraal en dat zorgt voor de nodige problemen. Dit is tevens een aanleiding voor het uitvoeren van deze opdracht. De recente invoering van het competentie gericht onderwijs in het VMBO is de andere aanleiding.
In dit document zetten wij uiteen wat ons plan van aanpak is. We geven een afgebakende formulering van de opdracht en brengen de eisen vanuit het bedrijf in kaart. Verder wordt er een tijdsplanning gemaakt en wordt de projectinrichting besproken. Tot slot wordt uitgelegd op welke wijze de kwaliteit gewaarborgd wordt en hoe de actoren – de opdrachtgever, coördinator en projectleden – hierbij betrokken zijn.

2 Projectopdracht

2.1 Projectomgeving

De huidige werkwijze heeft enige beperkingen: Het bestaat uit drie losse systemen, die verder niet centraal worden bijgehouden.
Allereerst is er het logboek, dit wordt beheerd door de stagiair zelf. Deze neemt het logboek mee naar het stage bedrijf en naar school. Dit heeft twee doelen, namelijk het enerzijds mogelijk maken van communicatie tussen de school en het stagebedrijf, en het anderzijds bijhouden van de voortgang van de desbetreffende stagiair.
Ten tweede is er Excel spreadsheet waarin bijgehouden wordt welke stagiair welke stages doen. In deze spreadsheet worden de volgende gegevens bijgehouden: de gegevens van de stagiair, stagebedrijf, stagebezoeker (beoordelend docent), stagecoördinator en mentor. Het stagebedrijf bied een werkplek aan de stagiair om werkervaring op te doen. Tijdens de stageperiode controleert de stagebezoeker de voortgang van de stagiair. De stagecoördinator beheert de stages van alle stagiairs. De mentor heeft weinig met de stage te maken en wordt vooral ter referentie genoemd in de spreadsheet. In het huidig systeem is er één spreadsheet voor alle stagiairs.
Ten derde hangt er een lijst in de school waarop suggesties staan voor mogelijke stageadressen. De leerlingen kunnen daar een stagebedrijf uitzoeken en zich voor een stage opgeven bij de coördinator. Deze lijst bevat slechts suggesties voor stagebedrijven en wordt in een spreadsheet bijgehouden.

2.2 Probleemomschrijving

De huidige situatie levert problemen op. Door het gebruik van een logboek kan er gemakkelijk gefraudeerd worden. Ook kan dit kwijtgeraakt worden door de stagiair, stagebedrijf of stagecoördinator. Verder wordt het logboek aan het einde van de stage gearchiveerd en gedigitaliseerd wat voor overhead zorgt.
Het gebruik van een spreadsheet voor het bijhouden van welke stagiairs naar welke stage gaan levert ook wat problemen op. Zo is het mogelijk dat er meerdere versies van deze spreadsheet bestaan, er is namelijk geen centrale opslag. Dit zorgt ervoor dat deze spreadsheet snel onoverzichtelijk wordt.
De lijst met stageadressen op school biedt geen eenvoudige mogelijkheid voor de stagebedrijven om stages aan te bieden. Verder kunnen leerlingen op deze lijst geen gedetailleerde informatie vinden.
Tot slot is er nog een overkoepelend probleem: Er zijn drie aparte informatiesystemen nodig om stages te regelen. Hierdoor is er geen centrale plek waar alle informatie te vinden is, omdat de verschillende bronnen niet aan elkaar gekoppeld zijn. Als bijvoorbeeld het email-adres van de contactpersoon van de stagebieder gewijzigd wordt in de spreadsheet, dan is dit niet zichtbaar op de lijst die in de schoolgang hangt.

2.3 Doelstelling

De opdrachtgever heeft deze opdracht gegeven, omdat hij het huidige proces wil verbeteren. Het huidige proces is inefficiënt en bevat verbeteringsmogelijkheden. Door deze door te voeren wordt het proces goedkoper, toegankelijker, veiliger en groener.

De kernwoorden van de door te voeren verbeteringen zijn: centralisering & digitalisering, waarbij onze nadruk op digitalisering zal liggen. De centralisatie van het systeem zorgt ervoor dat het opzoeken van gegevens minder tijd in beslag neemt en zorgt dit ervoor dat de gegevens toegankelijker zijn. In plaats van drie losse systemen te raadplegen, is het na centralisatie mogelijk om in één systeem alle gegevens te vinden. De digitalisering heeft meerdere voordelen. Zo zorgt het voor gemakkelijke archiveringsmogelijkheden die ruimte efficiënt zijn; er hoeft immers alleen een digitaal archief aangelegd te worden. Ook zorgt digitalisering voor een milieubewuster imago omdat dit het gebruik van papier terug dringt. Door digitalisering wordt het proces efficiënter wat een tijd- en kostenbesparing met zich meebrengt. Het proces is veiliger, omdat de actoren na authentificatie een gelimiteerd aantal handelingen kunnen uitvoeren. Hierdoor is het bijvoorbeeld niet meer mogelijk dat een leerling een handtekening vervalst. Tot slot voegt digitalisering ook nieuwe mogelijkheden aan het proces toe. Zo is bijvoorbeeld informatie altijd en overal beschikbaar. Ook is de informatie redundant opgeslagen en daardoor minder gevoelig voor verlies.

2.4 Opdrachtformulering

De opdracht luidt als volgt: “Ontwikkel een stage-registratie applicatie”. Wij zullen zoals in het volgende hoofdstuk besproken wordt een Agile ontwikkelmethode toepassen genaamd ”Scrum”. Deze methode voorziet in de verantwoordelijkheden van het ontwikkelteam en de opdrachtgever door gebruik te maken van “Backlogs”, dit wordt besproken in het volgende hoofdstuk. Er is afgesproken met de opdrachtgever om de voortgang tweewekelijks te bespreken en eventueel aanpassingen te maken.

2.5 Op te leveren

Een digitaal stage registratie systeem ter vervanging van het huidige systeem. De specificaties van dit systeem staan gedetailleerd omschreven in de “Product backlog” Appendix B.

2.6 Eisen en beperkingen

De explicite eisen zijn in Appendix A beschreven. Verder werd er in de interview, door de opdrachtgever, gesuggereerd om een webapplicatie te maken die off-site gehost zal worden en om het niet te koppelen met bestaande systemen.
Ook zijn er impliciete eisen die voortkomen uit de gebruikers omgeving. De applicatie moet toegankelijk zijn vanaf de schoolcomputers. Deze computers zijn uitgerust met Windows 7 en Internet Explorer 8. Verder blijkt, uit Appendix A, dat alle gebruikers van het systeem email moeten ontvangen en derhalve moet er van iedere gebruiker een email-adres bekend zijn.

2.7 Voorwaarden

De gestelde eisen zijn genoemd in Appendix B). De opdrachtgever is verantwoordelijk voor het stellen van de eisen. De projectgroep is verantwoordelijk voor het behalen van de gestelde eisen.

3 Aanpak

Wij gebruiken een Agile ontwikkelmethode genaamd Scrum. Daardoor is onze ontwikkeltijd opgedeeld in zogeheten “Sprints”. Deze Sprints laten wij een week duren, dit omdat wij dan een natuurlijk verloop in de week hebben en verwachten dat er geen of weinig Backlogs items zijn die langer dan een week aan tijd kosten. Omdat het project elf weken loopt zullen er dus elf sprints zijn. Deze Sprints zullen verderop in deze sectie ook als mijlpalen genoemd worden, inclusief bijbehorende eisen.
Tijdens de ontwikkeling zal de opdrachtgever geen technische aspecten of beslissing voorgelegd krijgen. Deze zullen we besproken worden met de coördinator van de TU Delft en eventueel met de Software Improvement Group (SIG). De opdrachtgever wordt wel betrokken bij het maken van functionele beslissingen.
Wegens de gebruikte ontwikkelmethode is het noodzakelijk dat de opdrachtgever nauw betrokken is bij het ontwikkelen van de software. De opdrachtgever heeft met het ontwikkelteam afgesproken om een tweewekelijkse vergadering te houden. Daarnaast is het wenselijk om in de andere weken ook feedback te krijgen door middel van een email.
Deze feedback zal voor de nieuwe sprint gegeven moeten worden (dat betekent dus vrijdag of in het weekend, met eventueel donderdag bij verhindering).
Scrum, de gebruikte ontwikkelmethode, werkt met Backlogs. Een Backlog is een verzameling van geïsoleerde functionaliteiten. Hieronder zal de Product backlog worden opgesomd. De Product backlog bevat alle “would have”’s van het product. Uit deze Product backlog wordt later de Release backlog gemaakt. Deze “Release backlog” bevat alle “must have”’s; alles wat het programma moet hebben om uitgebracht te worden. In dit specifieke geval zijn de Product backlog en Release backlog gelijk aan elkaar. De release Backlog wordt verder onderverdeeld in “Sprint backlogs” (elf stuks voor deze release). Elke van deze Sprint backlogs moet af zijn in de vastgestelde tijd: namelijk een week. Lukt dit niet of houden we veel tijd over, dan moet het proces herzien worden.

4 Projectinrichting

4.1 Organisatie

Wij hebben de verantwoordelijkheden op de volgende manier over de teamleden verdeeld:

Manager/Scrum Master David Festen
: Zorgt voor taakverdeling, tijdsbewaking en communicatie met de andere partijen

Lead Programmer Thomas Valera
: Zorgt voor consistentie en structuur in de code

Lead Tester Vincent Ghiëtte
: Zorgt voor code kwaliteitswaarborging en voor de het behalen van de eisen

Opdrachtgever Rens Looij
: Geeft wekelijkse feedback en zorgt op die manier voor bijsturing

Project Coördinator Hans-Gerhard Gross
: Coördineert het project, geeft tips en begeleiding

Quality Assurance SIG
: Controleert de kwaliteit van de code, zorgt voor kwaliteitswaarborging

4.2 Projectgroep

Van de drie projectleden hierboven genoemd wordt verwacht dat ze 42 uur per week besteden aan het project. Tevens wordt verondersteld dat ze aan de ingangseisen van het vak voldoen.

4.3 Administratieve procedures

De voortgang van het project zal door vier partijen gecontroleerd worden. De project manager zal namens de projectgroep de tijdsbewaking doen door middel van zogeheten "Burndown charts". Hiermee kan worden bijgehouden of het project volgens planning verloopt. De opdrachtgever zal bijsturing geven bij de wekelijkse feedback ronde. De project coördinator zal ingrijpen als het project dreigt fout te lopen. De SIG controleert de kwaliteit van de code en geeft feedback in de vorm van een rapport en een code kwaliteitscijfer.

4.4 Financiering

In de begroting van het Maaslandcollege is er geen ruimte voor het vergoeden van de besteedde uren en secundaire kosten van de projectleden. De TU Delft kan eventueel in software voorzien. De groepsleden zullen zelf de nodige hardware voor de ontwikkeling moeten voorzien.

4.5 Rapportering

Onze communicatie met de opdrachtgever zal verlopen via de feedback momenten aan het einde van iedere sprint. Tweewekelijks is er bijeenkomst en de andere weken gaat het contact via de mail.

4.6 Hulpmiddelen

Hier volgt een opsomming van de gebruikte hulpmiddelen:

- Computers
- Visual Studio 2012
- Team Foundation Server
- Azure
- Open source software libraries
- Koffie

5 Kwaliteitsborging

De opdrachtgever is nauw betrokken bij het ontwikkelingstraject en kan de kwaliteit in een vroeg stadium al beoordelen, daardoor zijn er veel mogelijkheden tot bijsturing. Om bij te dragen aan de kwaliteit van het product kan de opdrachtgever duidelijk aangeven welke onderdelen belangrijk voor hem zijn en welke bij een overschrijding van de ontwikkeltijd kunnen vervallen.
De projectgroep zal de kwaliteit zelf echter ook in de gaten houden. Dit gebeurt door gebruik te maken van de Scrum ontwikkelmethode, welke veel ruimte laat voor verbetering en bijsturing. Dit in tegenstelling tot de waterval methode, waarbij het resultaat pas aan het einde zichtbaar is. Ook zullen er “test cases” en “use cases” opgesteld worden om de code zoveel mogelijk aan de eisen te laten voldoen. Verder adopteren wij een “MVC” patroon zodat de code makkelijk te onderhouden is en modulair uitgebreid kan worden. SIG zal de code op twee momenten beoordelen op kwaliteit.

6 Nawoord

Dit document is het plan van aanpak voor het project in opdracht van het Maaslandcollege. Hierin staan de eisen, de manier van werken en de globale planning die bijdragen aan de verwezenlijking van het project. Eventuele afwijkingen van dit plan van aanpak zullen verklaart worden in het volgende document: Het eindverslag.

A Eisen van de opdrachtgever

Wat kan de stagecoördinator doen?

Op de openingspagina van de stagecoördinator zijn dit de mogelijkheden:

INVOER

1. Invoer leerling-namen, stage bieders, etc. [zie Excel]
2. Invoer veranderingen per leerling
3. Invoer standaard e-mails (als totale batch)
4. Invoer lay-out certificaat

UITVOER

5. Uitvoer per leerling gerealiseerde stage-uren
6. Uitvoer per week gerealiseerde stage-uren (alle leerlingen)
7. Uitvoer per leerling notities
8. Uitvoer per stage bieder notities
9. Uitvoer alle leerlingen notities
10. Uitvoer per e-mail toekenning gebruikersnaam en wachtwoord leerlingen (in totale batch)
11. Uitvoer per e-mail toekenning gebruikersnaam en wachtwoord stage bieders (in totale batch)
12. Uitvoer per e-mail toekenning gebruikersnaam en wachtwoord stage bezoekers (in totale batch)
13. Uitvoer per e-mail toekenning gebruikersnaam en wachtwoord mentoren (in totale batch)
14. Uitvoer voor alle leerlingen weergave scores competenties
15. Uitvoer per leerling weergave scores competenties
16. Uitvoer van op- en aanmerkingen over school, organisatie en begeleiding
17. Uitvoer van jaarverslag stage met totale hoeveelheid gelopen stages, bedrijven, etc

Wat kan de stage bieder doen?

Op de openingspagina van de stage bieder zijn dit de mogelijkheden:

INVOER

18. Invoeren gerealiseerde stage-uren per week
19. Invoeren op- en aanmerkingen over leerling
20. Invoeren van zijn score van leerling op competenties
21. Invoeren van op- en aanmerkingen over school, organisatie en begeleiding

LEZEN

22. Lezen van uren, op- en aanmerkingen, scores
23. Lezen contactmogelijkheden met de school

Wat kan de stagiair doen?

Op de openingspagina van de stagiair zijn dit de mogelijkheden:

INVOER

24. Invoeren van werkzaamheden per week
25. Invoeren van eigen score competenties

LEZEN

26. Lezen score op competentie door stage bieder
27. Lezen score op competentie door stage bezoeker

Wat kan de stage bezoeker doen?

Op de openingspagina van de stage bezoeker zijn dit de mogelijkheden:

LEZEN

28. Lezen van bedrijfsgegevens van stage bieder (naam stage bieder, naam bedrijf, adres, telefoonnummer, e-mailadres)
29. Lezen gerealiseerde stage-uren per week
30. Lezen op- en aanmerkingen over leerling
31. Lezen score van stagebieder leerling op competenties

32. Lezen op- en aanmerkingen over school, organisatie en begeleiding
33. Lezen score door leerling op competenties
34. Lezen van werkzaamheden per week door leerling ingevuld

INVOEREN

35. Invoeren van zijn score van leerling op competenties
36. Invoeren op- en aanmerkingen over leerling
37. Invoeren op- en aanmerkingen over school, organisatie en begeleiding

Wat kan de mentor doen?

Op de openingspagina van de mentor zijn dit de mogelijkheden:

LEZEN

38. Lezen van bedrijfsgegevens van stage bieder (naam stage bieder, naam bedrijf, adres, telefoonnummer, e-mailadres)
39. Lezen gerealiseerde stage-uren per week
40. Lezen op- en aanmerkingen over leerling
41. Lezen score van stagebieder leerling op competenties

B Product backlog

Tabel 1: Product backlog

B#	E#	Omschrijving	Niveau	Uren
2	1	Coördinator maakt een stagiair aan	5	4
3	2	Coördinator update een stagiair	5	4
4	24	Stagiair maakt gewerkte uren aan	5	4
6	1	Coördinator leest de stagiaires uit	5	4
7	25	Stagiair maakt zijn score aan	5	32
9	27	Stagiair leest scoren gegeven door de bezoeker	5	4
10	3	Coördinator verwijderd een stagiair	5	4
11	18	Stagebieder bewerkt de voortgang van de stagiair	5	4
12	1	Coördinator maakt stagebezoeker aan	5	4
13	28	Stagebezoeker leest contact informatie van de stagebieder	5	4
14	29	Stagebezoeker leest de werk uren van de stagiair	5	4
15	2	Coördinator bewerkt de informatie van de stagebieder	5	4
16	8	Coördinator leest de details van de stagebieder	5	4
17	32	Stagebezoeker leest op- en aanmerkingen over de stagiair	5	4
18	5	Coördinator leest de details van de stagiair	5	4
19	31	Stagebezoeker leest de score die hij aan de stagiair gegeven heeft	5	4
20	2	Coördinator verwijdert stagebieder	5	4
21	21	Stagebieder maakt een op- aanmerking aan over school, organisatie en begeleiding	4	4
22	32	Stagebezoeker leest op- en aanmerkingen over school, organisatie en begeleiding van de stagebieder	5	4
23	1	Coördinator leest de stagebezoekers uit	5	4
24	33	Stagebezoeker leest de scores van de stagiair uit die door de stagiair zelf ingevuld zijn	5	4
25	1	Coördinator maakt een stagebieder aan	5	4
26	1	Coördinator verwijderd een stagebezoeker	5	4
27	34	Stagebezoeker leest de samenvatting van de wekelijkse prestaties die ingevuld zijn door de stagiair	5	4

Continued on next page

Tabel 1 – *Continued from previous page*

B#	E#	Omschrijving	Niveau	Uren
28	1	Coördinator leest de stagebieders	5	4
29	1	Coördinator leest de details van een stagebezoeker	5	4
30	1	Coördinator bewerkt een stagebezoeker	5	4
31	35	Stagebezoeker maakt score aan voor de stagiair	5	4
32		Administrator maakt een coördinator aan	5	8
33	36	Stagebezoeker maakt op- en aanmerkingen over de stagiair aan	5	4
34	22	Stagebieder leest uren/op- en aanmerkingen/score van een stagiair	4	4
35	1	Coördinator maakt een mentor aan	5	4
36	1	Coördinator verwijdert een mentor	5	4
37	1	Coördinator leest alle mentors in	5	4
38	1	Coördinator leest een mentor in	5	4
39	1	Coördinator bewerkt een mentor	5	4
40	23	Stagebieder leest de contact informatie van de school	4	4
41	3	Coördinator importeert data vanuit een Excel sheet	2	24
43	38	Mentor leest een bedrijf in	5	4
45	40	Mentor leest op- en aanmerkingen over een stagiair in	5	4
46	39-41	Mentor leest de voortgang van de stagiair in	5	4
47	4	Coördinator importeert de certificaat layout	3	24
48	25	Stagiair leest zijn eigen gegeven scores in	5	4
49	6	Coördinator leest de gewerkte uren in van de stagiairs	4	4
50	14	Coördinator leest alle scores van de stagiaires in	4	4
51	15	Coördinator leest de score in van een stagiair	5	4
52	8	Coördinator leest de op- en aanmerkingen van een bedrijf over een stagiair	4	4
54	9	Coördinator leest op- en aanmerkingen over alle stagiairs	4	4
55	17	Coördinator leest de jaarlijkse statistieken in	5	4
56	22	Stagebieder leest eigen op- en aanmerkingen over school, organisatie en begeleiding in	4	4
57	/	Maak het initiële framework	5	4
58	/	Plan van aanpak maken/bijwerken	?	?

Continued on next page

Tabel 1 – *Continued from previous page*

B#	E#	Omschrijving	Niveau	Uren
60	/	TFS bijwerken	5	8
63	/	User Interface maken	5	42

B#: Backlog nummer
E#: Eis nummer van de opdrachtgever (zie Appendix A)
Omschrijving: Omschrijving van de eis
Niveau: Belang van de eis
Uren: Aantal uren die aan het realiseren van de eis besteed zullen worden