Evaluatie Live Performance

Roy v.Oldenbeek

2016

Inhoud

[Planning 2](#_Toc454463903)

[Functionaliteit 4](#_Toc454463904)

[Must haves: 4](#_Toc454463905)

[Could haves: 5](#_Toc454463906)

[Bronnen 6](#_Toc454463907)

[Problemen 7](#_Toc454463908)

[Oplossingen en dingen om trots op te zijn 8](#_Toc454463909)

# Planning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Taak | Begin geplant | Eind geplant | Begin echt | Eind  echt |
| Analysedocument | 09:00 | 09:15 | 09:00 | 09:10 |
| User requirements | 09:15 | 09:45 | 09:10 | 09:35 |
| Scenario | 09:45 | 10:15 | 09:35 | 09:56 |
| ERD | 10:30 | 11:00 | 10:00 | 10:43 |
| Use-case diagram | 11:00 | 12:00 | 10:45 | 10:56 |
| Use cases | 12:45 | 13:30 | 10:56 | 11:33 |
| UI schetsen | 13:30 | 14:00 | 11:33 | 12:54 |
| Ontwerpdocument | 14:00 | 14:15 | 12:55 | 13:04 |
| Datamodel | 14:30 | 15:15 | 13:05 | 13:43 |
| Klassendiagram | 15:15 | 15:45 | 13:45 | 14:46 |
| Testplan | 15:45 | 16:30 | 14:46 | 15:40 |
| SQL create scripts | 16:30 | 09:15 | 15:40 | 16:06 |
| DAL | 09:15 | 10:15 | 16:06 | 16:50 |
| Klassendiagram implementeren | 10:30 | 11:45 | 16:50 | 09:30 |
| SQL insert scripts maken | 11:45 | 13:00 | 09:30 | 09:57 |
| Overige functionaliteiten programmeren en gelijktijdig unit tests aanmaken | 13:00 | 15:30 | 10:00 | 15:30 |
| Documentatie in orde maken | 15:30 | 17:00 | 15:30 | 16:15 |
| Evaluatie schrijven | 16:30 | 17:00 | 16:15 | 16:50 |

Bovenstaand is de planing die ik voor mijzelf gemaakt heb plus de echte tijden dat ik er aan begon en mee klaar was. Alle groene velden zijn helemaal goed, en de gele velden zijn kleine opmerkingen bij. Voor het analysedocument geld dat het de start was dus dan begin je precies op tijd, daarom dat het geel gemarkeerd is want naar mijn mening moet ik in ieder geval iets voorlopen qua documentatie en dergelijke om in het OK gebied te zitten.

Verder is het bij het UI schetsen geel als eindtijd, Ik was hier eerder klaar als geplant, maar heb er wel langer over gedaan, daarom dat het geel is. Ik had voor de UI schetsen maken een half uur staan, maar uit eindelijk heb ik er bijna anderhalf uur aan gewerkt, er zat wel een pauze van drie kwartier tussen dus per saldo heb ik er een kwartier langer als geplant aan gewerkt. Bij een volgend project zal ik rekening houden met dat het langer duurt dan dat ik in eerste instantie denk bij het schetsen van de UI’s en hier dan ook meer tijd voor inplannen, verdere actie is hier gelukkig niet voor nodig geweest.

Het klassendiagram is ook geel, ik heb hier een uur over gedaan terwijl er een half uur voor ingeplant stond. Dit is gekomen omdat ik het had onderschat hoe ingewikkeld het klassendiagram zou zijn voor een redelijk kleine applicatie. Ik ben hier niet in problemen door gekomen qua planning, want ik liep nog steeds voor na het afronden maar ik zal zeker de volgende keer een uur standaard inplannen voor een klassendiagram in plaats van een half uur.

Het testplan is ook over de gestelde tijd gegaan, dit kwam echter omdat ik heel veel requirements had. Daardoor moesten er veel testcases geschreven worden en hier ging dan ook meer tijd in zitten als gedacht. De volgende keer zal ik de tijd voor een testplan maken pas goed inplannen als ik de requirements op heb geschreven.

Het laatste vakje is natuurlijk nog leeg omdat ik nu nog bezig ben met het schrijven van de evaluatie, maar te zien aan hoe het nu verloopt krijg ik de evaluatie ook goed op tijd af, hier zal dan ook niks op aan te merken zijn.

# Functionaliteit

Qua functionaliteit heb ik alle must haves kunnen implementeren, dit is goed echter had ik zelf liever gehad dat ik nog een paar meer should en/of could haves had kunnen implementeren. Ik weet dat ik er in ieder geval nog een paar had kunnen doen als ik deze gelijk had geïmplementeerd vanaf het begin, onder andere het meerdere boten toevoegen was eenvoudig geweest als ik dit gelijk had geprogrammeerd in plaats van één boot eerst en daarna nog kijken. Hierdoor zou ik klassendiagrammen en extra dingen aan moeten passen en nog een keer testen en daar heb ik helaas geen tijd meer voor gehad. Het toevoegen van meerdere boten zou kunnen door in de huur klasse een List<boot> van Boot te maken en in de huurdatabase klasse een foreach loop te doen net zoals met materiaal nu, aangezien de database hier ook al goed voor ingericht is. Het voorbereidende werk is er dus wel alleen heb ik door niet slim genoeg te programmeren deze helaas niet kunnen implementeren. Op dit moment heb ik nog twee could haves kunnen implementeren, het kunnen exporteren van het bestand als een HTML bestand en het kunnen kiezen tussen tekst en HTML. Deze waren eenvoudig te realiseren dus dat heb ik ook gedaan.

Echter betekend dit niet dat ik ontevreden ben met het resultaat, aangezien ik in ieder geval alle musthaves heb kunnen implementeren; Hieronder staat de tabel met de requirements die geïmplementeerd zijn:

## Must haves:

|  |  |
| --- | --- |
| Code | Omschrijving user requirement |
| M1 | Van een huurder moeten naam en e-mail adres opgeslagen worden. |
| m2 | Elke boot moet een unieke naam hebben. |
| m3 | Als een boot een motorboot is dan moet de tankinhoud en de actieradius van de boot opgevraagd kunnen worden. De actieradius is vastgesteld op de literinhoud vermenigvuldigd met 15. |
| m4 | Er moeten bijkomende artikelen gehuurd kunnen worden. |
| m5 | Alle gegevens in een huurcontract moeten in de database op worden geslagen. |
| m6 | Er moeten nieuwe huurcontracten aan kunnen worden gemaakt. |
| m7 | Een lijst met alle huurcontracten kunnen worden weergegeven op het scherm. |
| m8 | In het systeem moet het type boot, het aantal bijkomende artikelen, het budget en of een klant ook op de Noordzee of het IJsselmeer wilt varen ingevoerd worden waarna de applicatie terug geeft op hoe veel Friese meren er gevaren mag worden. |
| m9 | Het gemaakte huurcontract moet kunnen worden geëxporteerd naar een leesbaar tekst (txt) bestand. |
| m10 | Er moeten verschillende type boten gehuurd kunnen worden. |
| m11 | Er moeten twee categorieën zijn waar deze boten onder kunnen vallen, motorboot en spierkracht aangedreven boot. |
| M12 | De applicatie moet eenvoudig uitgebreid kunnen worden met extra vaarwateren. |
| M13 | De applicatie moet door een programmeur eenvoudig uitgebreid kunnen worden met extra type boten. |
| M14 | Met een spierkracht aangedreven boot mag alleen maar op de Friese meren gevaren worden. |

## Could haves:

|  |  |
| --- | --- |
| Code | omschrijving user requirement |
| C6 | Het huurcontract moet ook als een html bestand kunnen worden geëxporteerd. |
| C7 | Er moet gekozen kunnen worden tussen een HTML export en de tekst file export. |

# Bronnen

Ik heb niet heel veel bronnen geraadpleegd, ik heb voornamelijk op MSDN de documentatie van C# gebruikt als ik eventjes iets moest opzoeken qua functionaliteit en ik heb natuurlijk ook gebruik gemaakt van de documentatie die Intellisense en Resharper mij bieden in Visual Studio zelf.

# Problemen

De grootste problemen waar ik tegen aan ben gelopen is dat ik de requirements te uitgebreid heb gemaakt en dat ik nog niet héél vaardig ben met het goed maken van de documentatie. Hierdoor heb ik, alhoewel ik voor op de planning liep, toch minder tijd voor het programmeren gehad dan dat ik dacht. Gelukkig heb ik wel genoeg tijd gehad om alles wat af moet af te krijgen maar ik had liever nog meer dingen kunnen implementeren. Verder ben ik niet echt heel erg tegen problemen aangelopen.

# Oplossingen en dingen om trots op te zijn

Ik ben er zelf echter wel trots op dat ik alles geprogrammeerd en geschreven heb in alleen de tijd die er voor de Live Performance zelf staat, ik heb dus thuis nergens meer aan gewerkt en ik heb toch alle must haves af gekregen. Hier ben ik trots op aangezien ik van meerdere mensen heb gehoord dat zij nog best veel thuis hebben gedaan, en dat ze dan een paar should haves en could haves erin hebben maar ze hebben dan ook veel meer tijd eraan besteed als ik.

Verder ben ik trots op de LINQ queries die ik heb kunnen gebruiken voor het berekenen van het aantal meren dat bevaren kunnen worden; hierdoor is dit gedeelte van de code een stuk compacter geworden als met een foreach voor deze drie statements.

Verder ben ik er trots op dat ik het exporteren naar HTML in ongeveer 5 minuten heb kunnen programmeren, door de tijdsdruk wilde ik nog wel iets extra’s in de applicatie zetten en daardoor ben ik heel snel gaan werken en is dit toch nog goed gelukt.

Ook ben ik er trots op hoe eenvoudig de applicatie uit te breiden is met extra boten en/of vaarlocaties en/of materialen en zelfs boottypes, de combobox voor het selecteren van boottype wordt opgehaald uit de database, er is een koppeling tussen de (mogelijk extra) verschillende types zodat er daar ook nog onderling verschil kan worden gemaakt tussen wat waar op mag varen mochten er extra types bij komen, stel bijvoorbeeld een type “Jacht” dat alleen op zee mag varen, de applicatie kan hier gelijk al mee werken want de vaarwateren worden opgehaald uit de database rekening houdend met welk type boot je wilt selecteren en waar deze volgens de database mogen varen. Materialen worden ook in de database opgeslagen en het systeem om paketten toe te voegen is ook al aanwezig, alleen moet hier nog ondersteuning voor worden ingebouwd in de applicatie zelf alleen is dit ook eenvoudig te doen. Al met al is er bijna niks hardcoded in de applicatie en alles wat hardcoded is kan nog steeds eenvoudig worden overruled.