Namen:

Producten

Requirements:

* React/TypeScript | .NetCore | Entity Framework / Linq
* Testomgeving: Jenkins?

Voorbeelden:

* Bol.com <- layout, opzet, aka structuur

Ideeën:

* Product = ?
* Testomgeving: Postman
* Diagrammen = Astah
* Database = ? → PostgreSQL
* Languages = Python / Java / C# / …

Deadlines:

* OP1
  + Wk4 feedback
  + Wk7 feedback
  + Wk9 Formative!
* OP2
  + Wk1 feedback
  + Wk4 feedback
  + Wk7 feedback
  + Wk8 Summative

**Meeting notes:**

* TypeScript front-end
* Back-end met REST-API (Restfull application)
* Diagrammen: **Use-Case Diagram / ERD / SRS / Activity / Sequence** / Deployment / Class Diagram
* Zie trello voor de to do’s en status updates

**Meeting met docent:**

Zijn idee: hij neemt twee rollen in van project: product owner (technische kennis) maar wij verkopen aan hem het product. Stemmen af wat we implementeren, wat haalbaar wordt, functionaliteiten e.d. & tutor: beoordeling, feedback & insights uit zijn ervaring en hij beantwoordt vragen wat betreft het onderwerp. Specifieke vragen of onderdelen waarover onduidelijkheid is etc… de meetings worden voorbereid.

Product door taal, diagrammen, wireframes, features etc….

P.O: dmv van Use Case D : welke rollen? Wat wordt bedacht op abstract niveau aan interactie

Beschrijving van taal in de use cases laten zien: in diagram vorm, een use case beschrijving erbij.

Webshop wat innovatief is (qua onderwerp / idee), niet wat er al is.

Niet in het geheel een product bouwen wat qua methodische kennis en weten, realiseren, waar geen dampf soort von herhalingen zijn.

Productoverzicht creëren, dan kan je ook een orderoverzicht creëeren.

Kijk naar wat overeenkomt met een uitdaging voor je; wat je motiveert en waar je van leert.

Begrijpen op het modelleren op een abstract niveau van het software project. Door het ombouwen van taal waar aan het eind het omzetten van een doordacht project en model naar een concrete source code minimaal tijd kost. Project in het geheel gemodelleerd, zou het omzetten ervan naar source code minimale tijd kosten.

**P.O:** qua tijdplanning: wat is de complexiteit van dit project en waar is de meeste tijd voor nodig wbt features. Excellence criteria’s en verdiepingsonderwerpen: security, scalability, tests, validation, order status handling, stockmanagement, product recommendations, selling strategies, web standard validity, responsive design, optimizing proces, database abstraction, design patterns, search optimization.

Innovatief idee = 1 feature ervan

Bouwen een project voor Excellence.

Zoektechnologie zou een onderwerp kunnen zijn

Fasic, inhoud etc.

Wireframes creëeren, welke delen van het product dat we bouwen worden complex of kunnen we minimaliseren.

Vraag: een specifieke vraag over testen: moeten dat unittests en integration tests zijn?

Schatting tijdsbesteding: 10 uur per persoon per week

Next times: vragen voorbereiden voor de meetings

Bijv. van: dít willen we gebruiken/toepassen is het dan interessant als koper/eigenaar van de webshop (bijv. veiliger / sneller etc).

Ideeën: verkoop van visualisatie, abstract idee wat breed ingezet kan worden:

Security: password encryption, user input validations, belastingstests,

***Sommige features bieden wat motiverend is, dat totaal voldoet aan de requirements en daarna na overleg met docent kiezen hoe we het gaan bouwen en met wat.***

Projecttijdsplanning opstellen.

ElasticSearch

Categorizeren van producten, fassets, kaliberheid? Wattes? Haha

**Meeting 23 oktober 2017**

* Max 2x afwezig totale projectperiode
* Woensdag team assessment inleveren
* Volgende week formative meeting (worst case volgende week 15 minuten feedback, wss meer)
* Documentatie:
  + Uml
  + Class diagram
  + Sequence
  + Use case beschrjiving
  + Activiteits -> beschrijving workflow interactie
  + Wireframes

*Aan de hand van de specificatie van het document kunnen we het project bouwen*

*“Car configurator, mobile design uitsluiten”*

Feedback modules-draw.io.png:

* Het onderste gedeelte is 1 API, met daarop 3 services die een functionaliteit
* Terminologie: middelste rij zijn services (modules zijn omschrijvingen)
* Back-end service is de middelste rij, daarbovenop nog een tussenlaag → de modules worden geimplementeerd op de serverkant
* De ordermanager laag etc draait in de achtergrond
* Wordt verbeterd -> inspiratie package diagram?

Feedback wireframe:

* Laag 4: wordt scrolling bij A B C etc
* Laag 4: waar zie je de prijs? In de progress → waar ben je? Kan je terug? Als je terug gaat en het toepast dat je dan de aangepaste prijs ziet
* Laag 4: van het zoekveld filters (is een uitbreidmogelijkheid)
* Laag 5: overzicht configuratie samen met een prijskaart
* Laag 5: is voor de docent in beeldvorming nog onduidelijk omdat daar nog de complexiteit in zit
* Wishlist zouden willen afbeelden in het product
* Er zijn nog x aantal tussenstappen, in 5 jaar een auto waarbij je behalve wielen ook nog sport of limited kan aanbieden. Dat dit toegepast zou kunnen worden in de toekomst, met oog dus op het dynamische.
* Order bestelling en wishlist kan nog ingewikkeld worden in de backend

Dit zou complexiteit kunnen bevatten maar hij heeft het idee dat we nog een excellent feature moeten toevoegen aan het eind. Geen suggestions / recommendations, complexiteit, wielen die erbij passen etc.

Van te voren geconfigureerde auto’s maken waarbij ze kunnen zoeken, pre configured auto’s; daarin wordt de wishlist aangeboden.

Home page -> model kiezen / pre configured

Indicaties van modelleren, hebben het bedacht laten het zo zien...kunnen in volgende meeting in detail laten zien etc…

Complexiteit modelleren orderproces / overzicht

Orderproces zijn ook nog 3-4 stappen

Proefrit: car dealer, bespreking zus en zo, aanbieden

Type pagina’s als wireframe, niet een wireframe van elke losse stap

Denk vanuit het oogpunt; hoe kan je aan de PO een product verkopengit

Voeg toe model:

* menu

Class diagram:

* Interactie qua datamodelen, veranderingen van parameters zichtbaar wordt
* taalgebruik/taxonomie: geen invoice nog, maar bestelling?
* Een product/auto wordt geconfigureerd uit delen
* produtImages = technisch detail
* Customers moet adres hebben
* Adres = straat, huisnummer, postcode
* Beschrijven: typen aan de variabelen toevoegen
* Abstractclass, daarvan inherit configuration, invoice en bestelling
* Invoices taakbeschrijving → is eigenlijk geen invoice, invoice is aan het eind van de daadwerkelijke bestelling

Dit opsturen woensdagmiddag (de diagrammen etc, feedback krijgen we maandag)

Twee weken nog nawerk qua hetzelfde level als de andere groepen zijn

***In twee weken model compleet klaar en dan over drie weken gaan bouwen***

***Tijdplan creëeren***

**Feedback 30-10-2017**

Eerste feedback aan de hand van de ingeleverde documentatie

* De documentatie mag in het geheel meer uitgebreid zijn
  + Specificaties
  + Use cases → pre en post conditions
  + Helemaal in het Nederlands → **documentatie in het Engels schrijven**
  + Laat niet zoeken in de bestanden, 2 documenten met alle referenties erin
  + Diagrammen → wireframes, uml,
  + Ontwikkelen en documentatie neerzetten
    - Een visie dat op papier neergezet wordt wat overeenstemd met de broncode en het uiteindelijke product.
  + De excellent feature ontbreekt nog qua documentatie

***Nieuw idee:***

Basic webshop met onderdelen voor Tesla auto’s. Categorieën, zoekfunctie, filter. Excellent feature: livechat (met email van de conversatie).

***Feedback na nieuw idee:***

* Geen complexiteit qua algoritmen
* Misschien threading
* Uitdaging qua user interface e.d.
* Een doorheen navigerende databestanden, soortvan facets, combineert zoeken en criteria’s, is complexiteit inhoud. Geen algoritmes (dmv frameworks zijn er al oplossingen), qua statistieken. Backend statistiek uitgebreid / specifieker kunnen maken. Complexiteit door mathematische berekeningen e.d.
* Feedback chat implementeren: datamodel, gehele timing, threading, service test, mailfunctie etc.
* Frameworks met javascript grafische weergave
* Een week eraan besteden en dan na 1 week overstappen.
* Goed: scoping, deel ervan, soms iets ingewikkeld etc. dat is goed. Scope beperkingen.

Huidige model wat het inhoud, daarmee integreren.

Teamoverleg:

* Admin op domein → domein.iets/admin/
  + Admin gedeelte beveiligen
  + Admin controller die het afhandelt
* Twee documenten:
  + Technisch document
    - Use case descriptions + srs
  + Visueel document
    - Uml diagrammen + wireframes

**Teammeeting 13 november 2017**

* Zie lijst in Trello met alle to do: geel is front-end,
* Angular 4
* Als je een view maakt: schrijf ook op wat je nodig hebt / verwacht

**Docentmeeting 13 november 2017**

* Presentaties zijn nu geupload op Google Classroom
* Tools voorbeelden zijn daarin geupload
* Chapter 5/6 van het boek <- wat implementeren en waarom

**=== assignment volgende week voor de bespreking**

* Software architecture
* (nieuwe) Planning
* Initieële ideeën
* Bedenken welke features geïmplementeerd worden en wat de keuzes zijn
* ***== Planning | testmethodieken | tooling***
* De docent heeft de rol als Product Owner komende weken
* Aan hem presenteren de status en vragen stellen en discussieren qua features.
* Features wat binnen zichzelf presenteerbaar is als units, aangepakt worden als testomgeving, indeling van teammembers qua werkdruk
* Afstellen van unit tests, plannen etc...
* 🡪 *team* *testbespreking?*
* Je mag testen op de manier die je fijn vind:
  + Bijv basis feature
  + Basis feature testen
  + Dan de codes
  + Dan verder ontwikkelen
  + Kies een use case, stuksgewijs werken bijv.
  + Stel voor of het handig is om te bouwen

**Docentmeeting 20 november 2017**

* Status: documentatie is geupdate (SRS en SRD), deze week begonnen met code schrijven, opgesplitst in twee back-end en twee front-ends, repository, Play Framework, zit een testmodule in verwerkt, voor front-end AngularJS
* Testingtool: Managementtools 🡪 GitHub issues aan toevoegen, prioriseren etc...managementtooling 🡪 We maken gebruik van Trello
* Definition of done: definiëeren binnen een sprint van 1-2 weken, dat iedere iteratie een product levert
* Back-end: REST service 🡨 unit test --
* HTTP-request 🡨 Integration test
* Front-end: AngularJS 🡨 UI getest worden 🡪 gebruik maken van *Selendium*
* Iteration test en unit test op features, op de business logic
* Features implementeren binnen 1-2 weken, aan de hand van requirementstesting / integration testing / unitttesting 🡪 functionaliteit als done geevalueerd kan worden.
* Volgende week Product Owner rol
  + Presenteren testing en waarom welke keuzes
  + Implementaties
  + Huidige features die we implementeren / bouwen:
    - Registratie / Authentication
    - Login gedeelte
    - Home Page
    - Product view / overzicht / zoekfunctie
    - Order plaatsen
    - Admin paneel
  + Doel is een shippable product
  + Prioriteiten:
    - Iteration gedeelte werkt
    - Zie de backend interactie met bijv. Postman
    - Daarop wordt de UI gebouwd
      * Met welke feature aan de slag gaan, wat is haalbaar qua timemanagement.
  + Volgende week Dockerpipeline
  + Material Design 🡪 Google stijl
  + Aan het eind van een shippable product bij iedere iteratie, een backlog over hebben.
* Idee: de issues in Trello koppelen aan een commit

**Meeting notes 27 november 2017**

* Shippable product: website laten zien (Bram)
* Good idea to present a design, gives a precion
* Means that we implement 1-week sprints
* Trello is the scrumboard
* Unittest met Junit
* Idee voor testen: features met de meeste complexiteit zoals het orderproces, dat daar eerst tests voor geschreven worden. Zijn de meest ingewikkelde delen van implementatie.
* Integration test toevoegen waar HTTPrequest toe te voegen (voor de HTTP implementatie, geen styling, maar smoke achtige testimplementatie.
* Businesslogica ontstaat pas in order gedeelte. Domein gedeelte.
* Je kan nu een webrequest sturen naar de api: api.hrwebshop.tk
* Website: [www.hrwebshop.tk](http://www.hrwebshop.tk) 🡪 in de backend straks api calls (nu aparte link maar is al in één)
* Volgende meeting: product gedeelte compleet af, admin en start tests
* De bonus is order gedeelte