Procesboek programmeerproject:

WIKI

19 november:

College Django gekeken. Alle stappen kon ik goed volgen. Maar ik zou het zeker niet zelf na kunnen doen. Ik vind het ook nog niet echt duidelijk wat de functie van elke pagina die voor je wordt gemaakt. Dit zal ik dus nog even goed uit moeten zoeken.

23 november:

Het wiki-project aangemaakt via git. Het was eerst nog even zoeken hoe je nu op de goede manier de documenten van git naar je computer download zodat je ze in Atom kan gebruiken. Het aantal documenten vond ik nog wel echt overweldigend. Het lastigste vond ik dan ook om te kiezen waar je eigenlijk moet beginnen. Maar met het Django college ernaast ging dat ook wel beter. Ik heb deze dag vooral gebruikt om nog een keer rustig door het Django college te gaan en een aantal van die stappen te volgen. En dan vooral het gedeelte waarbij je in views.py een functie aanmaakt. Wat mij nog niet helemaal duidelijk is, is wat de variabele request is die door elke functie gaat. Dat vond ik niet zo goed uitgelegd in het college. Na het op internet opgezocht te hebben werd het wel al beetje duidelijker.

1 december

Vandaag heb ik de entry page en index page gemaakt. Een groot deel van deze opdrachten waren goed te doen met de lecture ernaast. Voor de pagina's was het even zoeken hoe je nou precies de markdown content goed kan weergeven. Ik had het namelijk wel naar html geconverteerd, maar de pagina weergaf ook de haakjes. Na even googelen werd duidelijk dat je |safe achter de content moet zetten zodat als HTML wordt weergeven.

5 december

Het maken van de search functie ging eigenlijk redelijk goed totdat ik de form moest weergeven. Ik ben namelijk heel lang bezig geweest met het invoegen van de form in de pagina zelf. De class voor de form aanmaken was me gelukt. Daarnaast had ik ook de form variabele toegevoegd aan de bijbehorende HTML-pagina. Maar het feit dat de form variabele ook doorgegeven moet worden aan de render functie was mij niet duidelijk. Het heeft mij ook wel veel tijd gekost om daarachter te komen. Achteraf was dit natuurlijk best heel logisch. Hoe kan de HTML-pagina anders weten wat de variabele {{form}} is? Maar als beginnend Django'er was het niet duidelijk hoe variabelen nou door worden gegeven tussen view.py, de HTML-pagina's en urls.py.

6 december:

Het was me nog niet eerder gelukt om mijn project weer naar git te pushen. Ik had namelijk toch de files op de verkeerde manier van git verwijderd. Bovendien werkte git nog niet op mijn computer door een nieuwe software-update. Dit moest ik dan ook even op Google opzoeken en daar kwam uit dat ik wat van de Apple-server moest downloaden. Toen ik dat het gedaan werkte het gelukkit wel weer. Bij deze minor heb ik ook zeker geleerd dat als je code of een programma niet werkt Google (en Stackoverflow) je beste vriend is.

Hierna heb ik nog de random page functie gemaakt. Deze opdracht stond als laatste tussen de opdrachten, maar ik had echt het gevoel dat deze opdracht met de kennis die ik tot nu toe had opgedaan zeer goed te doen moest zijn. Dit was ook het geval. Ik hoefde eigenlijk alleen de random functie te importen en vervolgens een random titel uit de lijst met titels te kiezen. Dit kostte mij dan ook maar 10 minuutjes om te implementeren.

7 december:

Waar ik lang op heb vastgezeten was het feit dat ik de textfield niet kleiner of groter kon krijgen. De textfield is namelijk op dit moment eigenlijk te lang voor de pagina waardoor het voorbij de linker layout balk komt. Ik vind dit niet echt mooi en wil wel graag dat het er goed uitziet. Maar na het internet voor ongeveer anderhalf uur afgestruind te hebben, bleek dit met een modelform te kunnen. Maar ik had nog nooit gehoord van modelforms en uitlegvideo's op YouTube hierover gingen of te snel voor mij, of ze gebruikte hele andere methodes om forms te creëren dan ik al had gedaan. Het is me op dit moment dus nog niet gelukt om de form kleiner te krijgen wat ik wel jammer vind.

Hierna ben ik begonnen met het maken van de Edit functie. Hier ging helaas ook een heleboel mis. Mijn ervaring is nu dat Django heel leuk kan zijn als het lukt. Maar het kan ook echt heel frustrerend zijn als je telkens errors krijgt en je niet meer weet waar je moet beginnen om je probleem op te lossen. De error messages zijn voor mij namelijk bijna altijd heel vaag. Dit maakt het ook lastiger om te lokaliseren waar je probleem precies zit. Na een hele tijd gestruggled te hebben met waar ik nou de titel moet invullen van de pagina's. Heb ik uiteindelijk besloten dat het mooi was geweest en het was tijd om even afstand van de problemen te nemen.

8 december:

Vandaag ben ik met verdergegaan met het maken van de Edit page. Ik had er nu te veel tijd ingestoken om op te geven. Wat mij heel erg hielp was alles wat ik gister voor deze functie had geschreven gewoonweg te verwijderen. Na een goed nachtje slapen was het tijd voor een frisse start. Dit keer begon ik met het maken van de HTML-pagina en het aanpassen van de andere HTML-pagina's zodat er een edit-page knop bij elke entry zou komen. Dit was mij vrij snel gelukt. Maar elke edit-page knop verwees natuurlijk naar dezelfde pagina. En dit is natuurlijk niet de bedoeling. We moeten dus net als bij de page path een soortgelijke path aanmaken waar we <str:title> gebruiken om de titel van de pagina die we op willen zoeken te krijgen. Dit is waar het eerder misging. Maar na rustig te bedenken welke functies deze titel nodig hadden bleek het goed te doen. En is dan ook gelukt.

Daarna kwam ik erachter dat mijn search functie nog niet helemaal volgens de opdracht was. Als je namelijk CS intikte in de zoekbalk kwam je automatisch bij Page Not Found. Terwijl de bedoeling is dat je dan naar een pagina gaat waar je de CSS-pagina kan aanklikken. Met de kennis die ik nu over Django heb opgedaan was dit voor mij niet een hele moeilijke fix. Ik hoefde namelijk alleen maar in de search_form functie toe te voegen dat wanneer een deel van de zoekterm in de titel van een pagina zit we de goede lijst met pagina's laten zien. Het lastigste hiervan was wel het bedenken hoe je weet of "CS" in "CSS" zit. Maar na even te Googelen bleek dat we gewoon if "CS" in "CSS" kunnen gebruiken (best logisch natuurlijk). Verder hoefde ik alleen nog een nieuwe HTML-pagina aan te maken die de lijst van pagina's weergeeft. Maar deze lijkt heel erg op de index pagina. Dit was dus ook niet al te lastig.

Nu heb ik eigenlijk aan alle vereisten voldaan. Het enige wat ik nog wel graag wil veranderen is de grootte van de edit en page forms. Die zijn namelijk nog wel wat te groot. Ik weet alleen nog steeds niet hoe je deze aanpast. Ik zal dit nog raadplegen bij de Django docenten.

COMMERCE

9 december:

SQL-college gekeken. Het grootste gedeelte van dit college was goed te begrijpen. Vooral met het eerste gedeelte waar SQL wordt uitgelegd was al bekend bij mij. Dit door het datarepresentatie vak vorig blok. Wat natuurlijk wel nieuw voor mij was, was hoe je in python SQL-querries kan uitvoeren.

10 december:

Git aangemaakt en de opdracht goed doorgenomen. Op het eerste gezicht lijkt me de opdracht goed te doen.

12 december:

De index (active listing) pagina aangemaakt. Om überhaupt active listings te hebben heb je 2 modellen nodig. Dit zijn de User en Auction_Item models. Deze heb ik dan ook als eerste geïmplementeerd.

Het user model was al grotendeels gegeven. Om dit model te begrijpen heb wel nog de migrations 0001 goed door moeten lezen. Het is namelijk handig om te weten hoe de variabelen in deze model gedocumenteerd zijn zodat je ze later weer aan kan roepen. Ik heb zelf nog de __init__ functie aan User toegevoegd. Dit omdat ik de gebruikers graag als “Voornaam Achternaam” wilde weergeven. Toen ik dit had gedaan waren de gebruikers op de admin pagina alleen niet meer beschikbaar. Hier heb ik een hele tijd op vastgezeten. Uiteindelijk zat het probleem in de register functie. De gebruiker moest namelijk wel zijn voor- en achternaam in vullen in een form-veld, maar dit werd vervolgens niet opgeslagen. Na deze fixt werkte het weer goed.

Het Auction_item model kon ik aan de hand van het Django college goed maken. Na deze aan de admin.py programma toe te hebben gevoegd kon ik in de admin pagina van de site zelf items toevoegen. Dit werkte dan ook goed en de namen van de actieve items worden nu keurig in een lijstje weergeven. Op dit moment worden alleen nog de namen van de items weergeven de prijs, beschrijven etc. voeg ik later hier nog aan toe.

13 december:

Na gister de eerste modellen geïmplementeerd te hebben was het nu tijd om op de site zelf nieuwe listings te kunnen maken. Eerst heb ik hiervoor de create_listing.html aangemaakt. Met de kennis van het Wiki project kon ik redelijk snel de forms toevoegen aan de html pagina. Er zit natuurlijk wel een verschil in het plaatsen van een form of het plaatsen van een modelform. Je moet bij het maken van een modelform namelijk de class Meta toevoegen. Dit was niet duidelijk van het de Django colleges. Maar het werd mij wel al snel duidelijk na

op internet op te zoeken hoe je modelsforms gebruikt. Met hetzelfde filmpje werd mij ook duidelijk hoe je informatie uit een modelform opslaat. Dit werkt ook bijna hetzelfde als bij gewone forms, alleen gebruik je nu request.POST[“naam van variabele”] om de informatie van een form op te slaan.
Na de informatie van de form opgeslagen te hebben moesten we alleen nog een Auction_item object aanmaken met de juiste waardes. Eerst had ik in mijn Auction_item de owner als Many-To-Many field gemaakt. Maar na een tijdje kwam ik erachter dat een item natuurlijk helemaal niet meerdere eigenaren kan hebben. Toen heb ik er een key-field van gemaakt. Het toevoegen van een key-field-item gaat wel wat anders dan een many-to-many item toevoegen. Een key-field kun je namelijk gelijk meegeven in bij het aanmaken van een object. Voor een many-to-many field moet je eerst het object hebben aangemaakt voordat je er items aan kan toevoegen. Het kostte mij ook redelijk wat tijd op het internet om achter dit feitje te komen.

Daarnaast heb ik nog de watchlist functionaliteit toegevoegd. Hiervoor heb ik een watchlist variabele als ManyToManyField aan de Auction_item class toegevoegd. Vervolgens heb ik de add_to_watchlist en remove_from_watchlist functies aan views.py toegevoegd die een item als input nemen en dat item dan toevoegen aan de watchlist van de gebruiker. Elke gebruiker kan meerdere item in een watchlist hebben en elk item kan door meerdere mensen aan zijn/haar watchlist toe worden gevoegd. Hierdoor is een ManytoManyfield de juiste optie.

14 december

Vandaag het ik de bid functionaliteit toegevoegd aan de site. Heel veel van deze functionaliteiten komen overeen met de eerder gemaakte classes. Een bid-object hoeft maar 3 variabelen te hebben namelijk: een de prijs van de bieding, degene die de bieding doet en het item van de bieding. Alleen voor de prijs hebben we een form nodig. Een bieding is natuurlijk een prijs, dus moet het in een decimalfield worden ingevuld. Het was nog even uitzoeken hoe je zorgt dat alleen getallen met 2 decimalen als input kunnen worden gegeven. Maar een decimalfield heeft als attribute decimal_places. Door deze op 2 te zetten neemt hij alleen getallen van 2 decimalen als geldige input. Wat ik deze keer alleen was vergeten te doen, was het opslaan van het Bid_object na het aanmaken. Dit had ik wel eerder gedaan bij de Auction_item, maar was ik hier vergeten. Dit zorgde ervoor dat ik een prijs kon invullenen en submitten op de site zonder crashes, maar dat hij niet updatete op de admin pagina. Na een tijdje lang gezocht te hebben zag ik de fout.

15 december

Categorie functie en model toegevoegd
Listing pagina mooi gemaakt. Ik merk nog wel dat ik het stijlen van pagina’s lastiger vind dan de back-end van een pagina maken. Met CSS en de bootstrap containerclass is het uiteindelijk wel goed gelukt naar mijn idee. Maar ik merk wel dat er veel tijd in gaat zitten. Bij de back-end werkt iets, of werkt iets niet. De front-end van een website kun je oneindig blijven verbeteren en mooi maken. Ten eerste moet je voor de front-end creatief zijn, maar daarnaast ook nog in CSS/HTML/Bootstrap precies weten hoe je de ideeën ook echt implementeert. Nu heb ik namelijk de bootstrap container classes gebruiker waarbij je een aantal rijen en kollommen kan maken.