TDP003 Projekt: Egna datormiljön

systemdokumentation

Författare

Jesper Skoglund, Jessk378, Niklas Nilsson, Nikni292

Höstterminen 2015
Version 1.1

2015-10-06

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
0.1	skapade dokument för skrivning	151006
0.2	La in sekvensdiagram och översikts bild	151008
1.0	Färdigställde information om presentationslagret	151008
1.1	La in felhanteirng/logggning, html templates och interaktivt diagram	151009

Innehåll

2 Introduktion

I detta dokument kommer ni ta del av hur våran kod är tänkt att fugera. Ni kommer se sekvensdiagram över hur våra funktioner fungerar med varandra, samt bilder med förklaringar på specifika kodstycken.

3 Ffilstruktur

Filstrukturen vi har. - betyder att är en mapp och + betyder att det är en fil. |

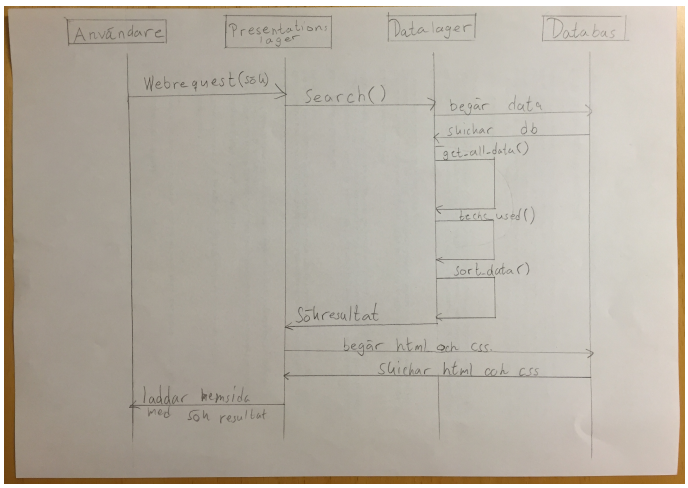
```
|-doc
|    +systemdokumentation.pdf
|
|-myproject
|         +README
|         +config.json
|         +data.json
|         +data.py
|         +server.py
|         -static
|             -images
|             -style
|                 +main.css
|         -templates
|             +404.hmtl
|             +_project.html
|             +application.html
|             +index.html
|             +project.html
|             +projects.html
|             +search.html
|             +techniques.html
|
```

4 Data.json

Formatet på data.json är en tabell med statiska nycklar och värden vi kan ändra. Alla nycklar kommer nedan.
start_date": < Start datum för projektet. >
short_description": < En kort beskrivning av projektet. >
"course_name": < Kursens namn.>
long_description": < En lång beskrivning av projektet. >
group_size": < Antal deltagare i projektet. >
ächademic_credits": < Hur många akademiska poäng projektet ger. >
lulz_had": < Hur kul vi hade. >
"external_link": < Länk till extern sida. >
small_image": < En liten bild relaterad till projektet. >
techniques_used": < En lista på tekniker som användes i projektet. >
project_name": < Titel/namn på projektet. >
"course_id": < Kursens ID. >
"end_date": < Slut datum för projektet. >
project_no": < Projektets nummer. Skall var unikt. >
big_image": < En stor bild relaterad till projektet. >

5 Interaktionsdiagram

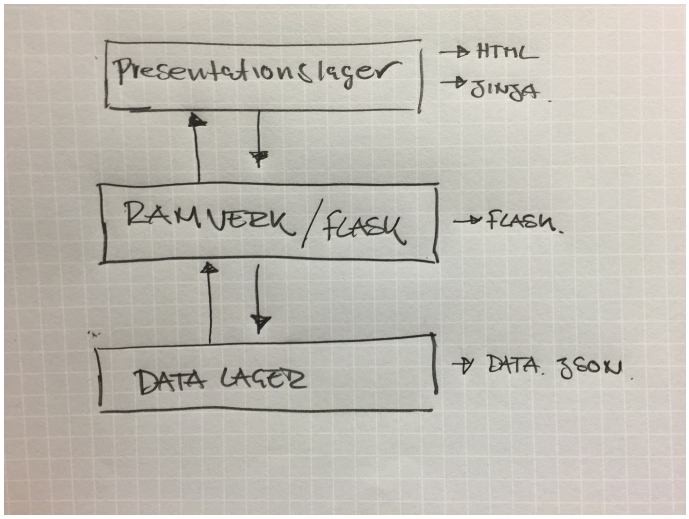
Användaren skickar ett web request som tas upp av flask. Flask anropar funktionen search() i databasen. Search() begär då databasen, databasen skickar tillbaka data.json. Search() anropar get_all_data() för att söka i data.json. Den får tillbaka ett sök resultat. Search() anropar sedan techs_used flr att filtrera bort alla projekt som inte har specifika tekniker. Search() anropar sist sort_data() för att sorta filtrerat sökresultat. Datalagret skickar sökresultatet till flask som begär html och css sidor. Den får tillbaka search sidan och visar den för användaren med givet sökresultat



Figur 1: Sekvensdiagram över searchfunctionen tillsammans med presentationslager

6 Övergripande bild

Bilden nedan visar simpelt de tre lager vi arbetar med. Presentationslagret är delen som visar html sidor. De laddas med hjälp av jinja. Jinja blir tilldelad sidorna av Flask som ligger i ramverket. Innehållet i htmlsidorna läses från en .json fil. Den filen läses in i datalagret. Datalagret tar även hand om sökningar i data.json filen. Flask kommer kalla på specifka delar av .json filen beroende på vad användaren skickar för förfrågan. Flask returnerar sedan html sidan som ska laddas tillsammans med informationen som ska fylla sidan.



Figur 2: Övergripande bild på hur datalager, ramverk och presentationlager kommunicerar med varandra

7 Datalagret

Här nedan kommer det finnas förklaring på funktioner som vi implemterat i vårat datalager. Här finns även sekvensdiagrammet. Det kommer till en början vara översiktligt och sedan gå mer in på djupet av varje funktion och hur de fungerar tillsammans med andra funktioner.

7.1 Kursen datalager specifikationer

I denna sektionen finns kursens krav på vad de olika funktionerna i datalagret ska ha. För varje funktiner kommer det finnas en koppling mellan funktionen vi skrivit och kursenskrav.

7.1.1 Load-funktionen

Kursenskrav: Ladda en .json fil och returnera en lista med alla projekt. Load ska ladda från en fil med UTF-8 teckenkodning. Vid error ska funktionen returnera “None”.
Parameterar: filnamn, en sträng som innehåller namnet på json filen vi vill ladda.
Returnerar: en lista, listan ska inne hålle alla project från json filen eller “None”
Vår funktion: 7.2.1 Load-funktionen.

7.1.2 Get-project-count-funktionen

Kursenskrav: Få antalet project i en lista av projekt.
Parametrar: En databaslista som man får ifrån Load-funktionen.
Returnerar: Antalet project i listan.
Vår funktion: 7.2.2 Get-project-count-funktionen.

7.1.3 Get-project-funktionen

Kursenskrav: Hämtar ett speciferat project från databaslistan utifrån angivet ID. Om project inte finns returneras “None”
Parametrar: En databaslista som man får ifrån Load-funktionen.
Projekt id:et, ett nummber som anger vilket projekt som ska hämtas
Returnerar: En tabell med all data från ett projekt eller “None”.
Vår funktion: 7.2.3 Get-project-funktionen.

7.1.4 Get-techiques-funktionen

Kursenskrav: Hämtar en lista med alla tekniker från våra projekt, de ska vara i alfabetisk ordning.
Parametrar: En databaslista som man får ifrån Load-funktionen.
Returnerar: En alfabetiskt sorterad lista med namnen på alla tekniker i databaslistan.
Vår funktion: 7.2.4 Get-techniques-funktionen.

7.1.5 Get-techique-stats-funktionen

Kursenskrav: hämta och returnerar statistik från alla tekniker speciferat i databaslistan. Nyckeln i tabellen är namnet på tekniken och nycklens värde är en lista med en tabell inuti. Den tabellen ska ha “id” som första nyckel med projektets “id” som värde. Den andra nyckeln är “name” den har projekets namn som värde. Id:ets och namnet värde ges av att tekniken ingår i det specifika projektet vi tittar på. I tabellen med id:et och namnen ska vara sorterade efter id:ets värde.
Parametrar: En databaslista som man får ifrån Load-funktionen.
Returnerar: En tabell med teknikernas statistik (se ovan).
Vår funktion: 7.2.5 Get-technique-stats-funktionen.

7.1.6 Search-funktionen

Kursenskrav: Hämtar och sorterar projekten som matchar sökningen som angetts.
Parametrar: En databaslista som man får ifrån Load-funktionen.
En variabel “sort_by”, namnet på fältet som funktionen ska sortera efter. En variabel
En variabel “sort_order”, anger om sorteringen ska ske fallande eller stigande.
En lista av tekniker som det returenerade projektet måste ha med. Em tom lista betyder att fältet ska ignoreras.
En sträng variabel “search”, en fri text sökning.
En lista, “search_fields”, fältet som vi vill att fri text sökning ska ske i. Om den är tom returneras inget. Om den är “None” ska alla fält sökas.
Returnerar: En lista med tabeller som innehåller alla projekt som matchade sökningen.
Vår funktion: 7.2.6 Search-funktionen.

7.2 övergripande beskrivning av funktionerna

En djupare nivå av hur funktionerna fungerar och hur de fungerar med varandra.

7.2.1 Load-funktionen

Vi använder angivet namn som vi fick som invärde i ett “with” block för att öppna filen. Eftersom filen vi läser in ifrån är en .json fil kan vi lätt läsa in filens innehåll med den inbyggda funktionen “json.load”. Vi returnerar sedan listan med databasen. Om det av någon anledning inte gått att läsa filen returnerar vi “None” som databas.

7.2.2 Get-project-count-funktionen

Funktionen kollar längen av listan, altså antalet element som finns i listan. Funktionen returnerar sedan det numret vi fick fram.

7.2.3 Get-project-funktionen

Vi itererar igenom databasen och kollar specifikt på varje projekt. Vi jämför sedan projektets id med id:et som använder angett. Om id:et finns i något projekt returnerar vi projektet som det stämde överens med. Om angivet id inte finns i något projekt retuernerar funktionen “None”.

7.2.4 Get-techniques-funktionen

Funktionen iterar över hela inladdade databasen, den letar sedan upp fältet “techniques-used”. För varje teknik i “techniques-used” lägger vi till den teknik i en lista om listan inte redan innehåller den tekniken. Vi sorterar sedan teknikerna i listan i bokstavsordning. Funktionen returnerar sedan den sorterade listan.

7.2.5 Get-techniques-stats-funktionen

Funktionen börjar med att anropa get-techniques och sparar undan alla använda tekniker som använts i databasen. Vi iterar sendan igenom sparade tekniker för att undersöka varje teknik, en i taget. Vi skapar en dict med tekniken som nyckel. Vi iterar nu över varje projekt. Om tekniken är med i projektet lägger vi till projekts namn och id som värde för nyckeln. Innan vi börjar med nästa teknik sorterar vi värdena som tillhör varje nyckel efter id:ets stortlek. Vi börjar sedan om med nästa teknik och köra samma process. När vi gått igenom alla tekniker returnerar vi listan med tabellen.

7.2.6 Search-funktionen

Alla invärden i våran search funktion har fördefinerade värden ifall inget värde skulle skickas med. Deras start värde är: sort_by=”start_date”, sort_order=”desc”, techniques=”None”, search=”None”, search_fields=”None”. Search anropar sedan get-all-datafunktionen som ger tillbaka en lista. Den lista skickar search in techs-used-funktionen för att se om om sökresultaten stämmer överens med teknikerna som angetts vid anropet. Detta sker bara så länge någon teknik har angets annars hoppas detta steg över. Sista funktionaliteten som search har är att den skickar in den sökta och filtrerade databasen i sort-data-funktionen för att sortera sökesultatet enligt användarens specifikationer. Den sökta, filtrerade och sorterade listan retuneras sedan.

7.2.7 Get-all-data-funktionen

Vi börjar funktionen med att titta ifall search är “None” och om search_fields är “None”. Om detta stämmer skickar vi tillbaka hela databasen som sökresultat. Nästa koll är om search_fields är “None”, om detta stämmer söker vi i alla fält genom att iterera över varje projekt för sig och om search får några träffar i projektet. Om search får en träff läggs det träffade projektet till i en lista. Detta sker tills det inte finns fler projekt. När alla är genomsökta skickar vi tillbaka alla projekt som fick en sökträff. Härnäst udnersöker vi om search_fields är tom, om detta stämmer returnerar vi “None” som sökresultat. Om ingen tidigare “if” sats stämmer vet vi att search och search_fields har ett värde. Så då itererar vi över insickad databas. Vi undersöker sedan om sökresultat är en lista, en int eller en sträng. Om det är en sträng gör vi sökning och sökresultat till gemener. Nästa steg undersöker om fältet matchar search_fields och att search matchar något i databasen. Om allt detta stämmer lägger vi till projektet vi hittade alla saker i en lista. När vi gått igenom alla projekt och undersökt alla fält och sökvärden returnerar vi en lista av alla projekt som matchade sökningen.

7.2.8 Techs-used-funktionen

Vi börjar genom att itererar över sökvärdet som search-funktionen skickade in. Vi iterar sedan över alla tekniker i fall att vi vill sortera efter flera tekniker. Vi undersöker sedan om teknikerna finns med med i projekten. Om teknikerna är med lägger vi till projektet i en lista. Vi returnerar sedan den listan med filtrerade projekt.

7.2.9 Sort-data-funktionen

sorterar den sökda data mängden som search skickade genom att köra “.sort” fukntionen. Med nycklen som vi sorterar efter som värdet sort_by hade. Vi sorterar i ordning efter reverse eller inte beroende värdet som sort-order-funktionen returnerar. Om den returnerar True sorterar vi i fallande ordning om den returnerar False sorterar vi i fallande ordning.

7.2.10 Sort-order-funktionen

kollar om sort_order är “desc”(descending) om sort_order är “desc” returnerar funktionen True. Annars returnerar funktionen False.

8 Ramverk / Flask

Nedan finns funktionerna som vi skrivit i flask för att kunna ladda html sidorna med inbäddade python funktioner.

8.1 Funktioner

Detta är funktionerna som kallar på jinja och laddar in html sidor så de visas i vår portfolio.

8.1.1 Data och Config - load funktionerna

I dessa funktioner har vi hårdkodat namnet på .json filerna. Config-load: Laddar en config fil med information som är i till exempel header och footer.
Data-load: Laddar databasen från en fil med information som vi sedan använder på våra hemsidor.

8.1.2 Index-funktionen

Denna funktion läser in senaste projektet samt hämtar information via config-load funktionen, för att generera statisk information som finns på alla sidor. Detta kan vara till exempel information av skapare, kontakt information etc.

8.1.3 Projects-funktionen

Html sidan som den här funktionen skapar är till för att visa alla projekt i form av en lista. Denna funktion läser in alla projekt från databasen och skickar de vidare till presentationslagret där en html sida skapas. Mallen som används för att visa projekten är den samma som används i search-funktionen.

8.1.4 Project-funktionen

Html sidan som den här funktionen skapar är till för att visa ett specifikt projekt beroende på vilket id. Funktionen kommer åt id:et via urlen. Denna funktion hämtar information om ett specefikt projekt ur databasen och skickar det till presentationslagret.

8.1.5 Techniques-funktionen

Html sidan som den här funktionen skapar är till för att visa alla tekniker och projekt som använder den specifika tekniken. Denna funktion hämtar alla tekniker och de projekt som använder en viss teknik och skickar detta till presentationslagret. Användarna kan klicka på ett specefikt projekt och komma till project-funktionen.

8.1.6 Search-funktionen

Html sidan som den här funktionen skapar är till för att visa en sökruta och klickbara check boxar. Man kan skriva en söksträng i rutan och filtrera projekt som har specifika tekniker. Sökresultat visas sedan under checkboxarna. Denna funktion skiljer sig från de andra då den hanterar både GET och POST, i det första skedet (GET) så visas ett formulär för användaren där man kan skriva in ett sökord och klicka i tekniker. När användaren söker på ett ord så anropas funktionen med POST och då hämtas träffande projekt ur databasen. Dessa presenteras sedan för användaren med samma mall som i projects-funktionen.

9 Html templates

Sidorna som renderas genom funktioner.

9.1 project.html

Html sidan som används av projet-funktionen.

9.2 projects.html

Html sidan som används av projects-funktionen.

9.3 _project.html

Mall som används för att generera projekt listor.

9.4 search.html

Html sidan som används av sök-funktionen.

9.5 techniques.html

Html sidan som används av techniques-funktionen.

9.6 404.html

Html sidan som används när 404 errorn uppstår.

9.7 application.html

Mall sidan för alla html sidor. I applicatoin.html

10 Felhantering och loggning

Felhanteringen använder vi vid två olika tillfällen. Ena gången när vi laddar config och load. Då får användaren ett 400 error som säger att filen inte kunde laddas. Den andra felhanteringen vi gör är användaren försöker hitta ett projekt med ett id som inte finns. Då får användaren ett 404 error. Vi båda fallen loggar vi när det skedde, att det var ett error, vilken funktion felet uppstod i och ett medelande som innehåller sidan de letade efter. Vi loggar även alla sökningar som görs. Då loggar vi när det skedde, att det var en sökning, att det skedde i sök-funktionen och ett medelande som innehåller vad de säkte på.