LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

Systemdokumentation

Författare

Jimmie Roos, jimro697@student.liu.se
Sebastian Grunditz, sebgr273@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2018
Version 1.0

2018-10-18

Innehåll

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första version	18/10-18
1.1	Revision efter kommentarer	25/10-18

1 Introduktion

1.1 Datalagret

Datalagret är utvecklat enligt kravspecifikationerna, som du kan hitta här. Vi har även skrivit tre hjälpfunktioner, `check_search()` och `search_project()`, för `search()`, och `get_latest_project()` för startsidan.

1.1.1 Check_search(search_word, item_)

Det funktionen gör är att kolla om det sökord man får in, finns i projektet man vill kolla i. Den kallas på varje projekt i listan, men hanterar ett projekt i taget. Funktionen har tre delar. Först kollar den om det man vill jämföra med sökordet är en integer. I sådant fall försöker den göra om sökordet till en integer, och om de stämmer överens skickar den tillbaka `True`. Annars kollar den om vi vill kolla i en lista, och kör en koll rekursivt på varje del föremål i listan, bryter ifall den hittar det den letar efter, och skickar tillbaka `True`.

Den sista delen antar att det vi söker i är en string. Då gör den om båda input så att de endast innehåller versaler, och kollar om sökordet finns i strängen vi söker i. Ifall den finns skickar den tillbaka `True`.

1.1.2 Search_project(project, search_word, searchfield)

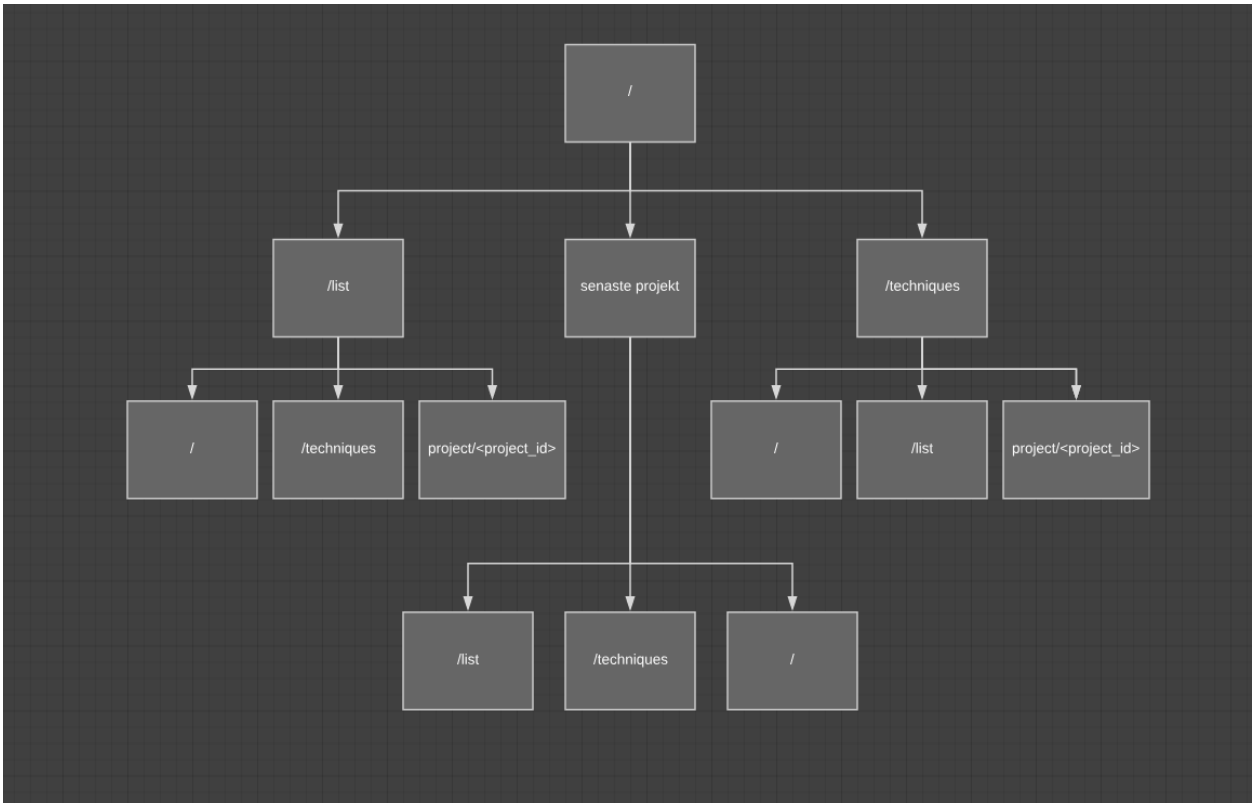
Kollar först om det man söker på är en lista med saker eller inte. Om det är en lista så körs `check_search()` på varje sökord och returnerar `True` om det stämmer. Om det inte är en lista så körs `check_search()` på det sökord man använde.

1.1.3 Get_latest_project(db)

Den här funktionen söker igenom listan, med hjälp av `search()` så att databasen kommer tillbaka, sorterad efter start datum. Sedan returnerar den det första elementet i listan, vilket motsvarar senast påbörjade projekt.

2 Översikt

2.1 Översiktsbild



Figur 1: Översiktsbild över hemsidan

Figur 1 visar ett enkelt flödesschema för navigering på hemsidan. “Senaste projektet” är isolerat, då det ligger på startsidan. Varje steg neråt i flödesschemat visar alla möjliga steg man kan ta från just den sidan.

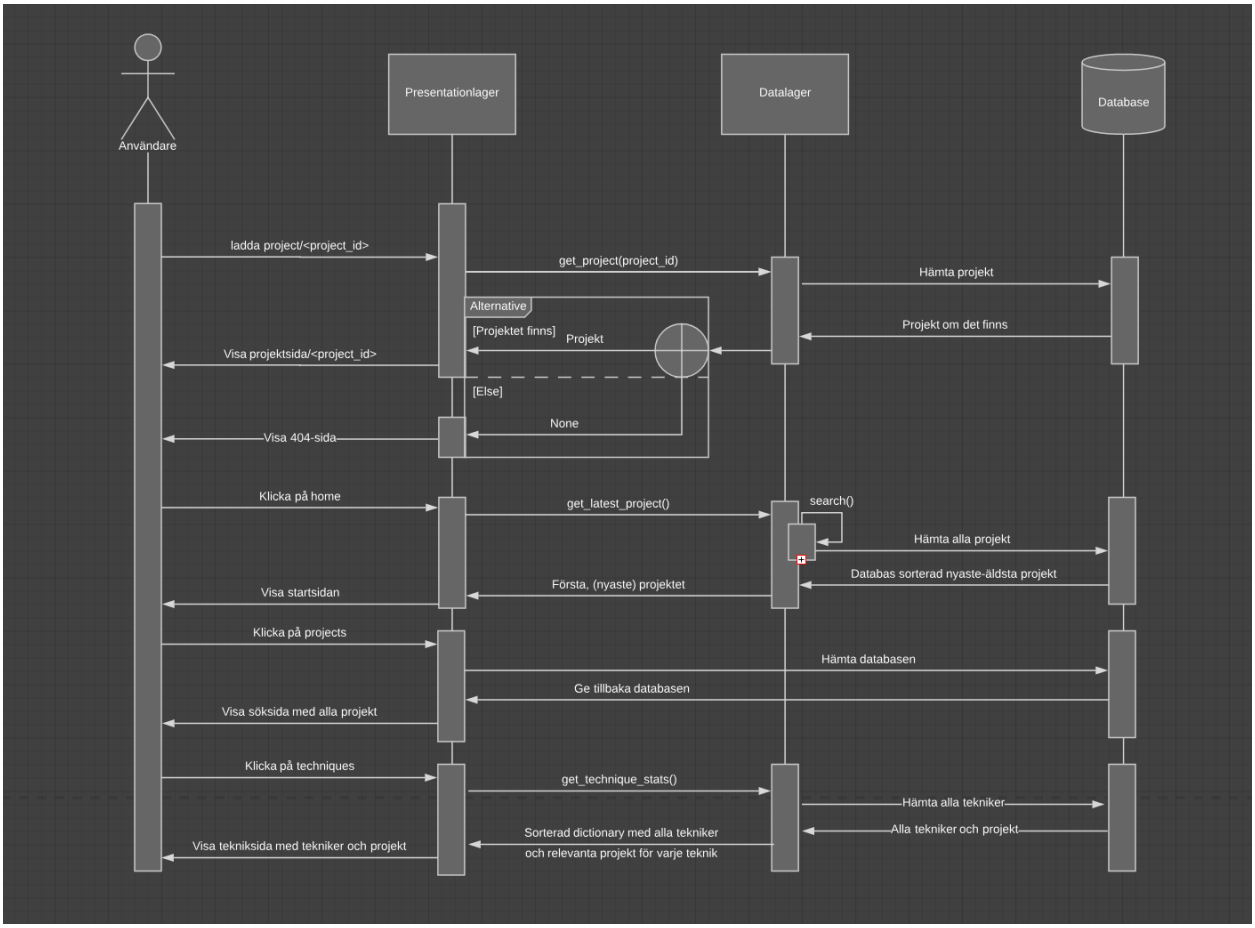
2.2 startsidan

På indexsidan laddas det senaste projektet och visas på skärmen för användaren, tillsammans med information om den som äger hemsidan. Därifrån får man själv välja vad man vill ladda, t.ex projektsidan eller tekniksidan.

2.3 söksidan

Söksidan skriver ut alla projekt som stämmer in på sökkriterierna som skickats in från användaren. Ifall man blivit omdirigerad från en annan sida visar den alla projekt, sorterad i fallande ordning på start datum.

2.4 Projektsida



Figur 2: Sekvensdiagram för en projektsida. O med kryss i skickar vidare en av två möjliga output.

Projektsidan laddar automatiskt projektet som användaren valde på förra sidan och skickar med bilder och beskrivningar, med mera, som hör till korresponderande projekt. Som syns i figur 2 är det ett väldigt simpelt arbete för servern att ladda "/list", då den inte behöver interagera med datalagret. Detta beror på att vi skickar med hela databasen, som laddas under uppstart av servern, när man blir omdirigerad dit.

2.5 Tekniksidan

Tekniksidan skriver ut alla tekniker som använts i projekten. Alla projekt delas sedan in i menyer för de tekniker som de har använt.

2.6 Filstruktur

- static
  - images
    - *.jpg
    - *.png
    - *.tiff
  - style
    - *.css
- templates
  - *.html
  - *.json
  - *.xml
- *.py
- *.md
- database.json
- app.py
- datalayer.py
- debug.log

3 Presentationslagret

I det här avsnittet beskrivs funktionerna i presentationslagret `app.py` mer ingående.

3.1 page_not_found(error)

Metoder

- GET - Begär data från resurs.

Parametrar

- error(error code) - Ett felmeddelande.

Returns: render

Redirect till 404-sidan.

Returnerar en 404-sida när vi får en 404 error.

3.2 index()

Metoder

- GET - Begär data från resurs.

Parametrar

- Inga.

Returns: render

Redirect till index.html.

Returnerar hemsidan, med senaste projektet.

3.3 search()

Metoder

- GET - Begär data från resurs.
- POST - Skicka data till servern för att skapa eller uppdatera en resurs.

Parametrar

- Inga.

Returns: render

Returnar en lista med sökresultat.

Om vi får en GET så returnerar vi hela databasen. Om vi får en POST så filtrerar vi databasen baserat på input av användaren.

3.4 techniques()

Metoder

- GET - Begär data från resurs.

Parametrar

- Inga.

Returns: render

Redirect till techniques.

3.5 project(project_id)

Metoder

- GET - Begär data från resurs.

Parametrar

- project_id(integer) - För att beskriva vilket projekt sidan handlar om.

Returns: render

Redirect till projektsida eller 404.

Returnerar en sida med specifikt projekt eller 404 om inte projektet hittas.

3.6 list()

Metoder

- GET - Begär data från resurs.

Parametrar

- Inga.

Returns: render

Redirect till söksida med resultat.

Får in databasen, representerad av en dictionary. Skickar tillbaka en söksida med alla projekt om inget söks efter, eller om man kommer till "/list" från en annan sida. Returnerar söksidan med filtrerat resultat om användaren har sökt på något specifikt.

4 Felsökning

Vi använde oss av en funktion som heter RotatingFileHandler för att kunna skriva ut errormeddelanden till fil. Vid en error så skickas användaren till en 404-sida och felmeddelandena skrivs ut till en fil `debug.log` i projektmappen. Den skriver felmeddelande på en *DEBUG*-nivå, vilket innebär att den skriver ut information som är användbar vid debugging av hemsidan. För att inte `debug.log` ska bli allt för stor, har den en max byte-gräns, innan den stänger och sparar loggen, för att öppna en ny. Den sparar de fem senaste dokumenten, vilket innebär att totalt sparar den 60000 bytes med log från hemsidan. Om man vill ändra, gör man det i app.py på följande ställe:

```
if __name__ == "__main__":
    handler = RotatingFileHandler('debug.log', maxBytes=10000, backupCount=5)
    handler.setLevel(logging.DEBUG)
    app.logger.addHandler(handler)
    app.run(debug=True)
```

I `RotatingFileHandler()` ändrar man på *maxBytes* för att ändra storleken på `.log`-filen och *backupCount* för att ändra antalet `.log`-filer som sparas.

För att prova om datalagret fungerar så har vi en `data_test.py` som vi kör på `data.json` och ser om vi får några fel. Båda dessa finns att hämta på gitlab.

5 Appendix

Figurer