TDP003 Projekt: Egna datormiljön

Systemdokumentation

Författare
Erik Lindow, erili445@student.liu.se
Erik Edling, eried975@student.liu.se

Examinator
Klas Arvidsson, klas.arvidsson@liu.se

Version 1.0
Höstterminen 2014

Revisionshistorik

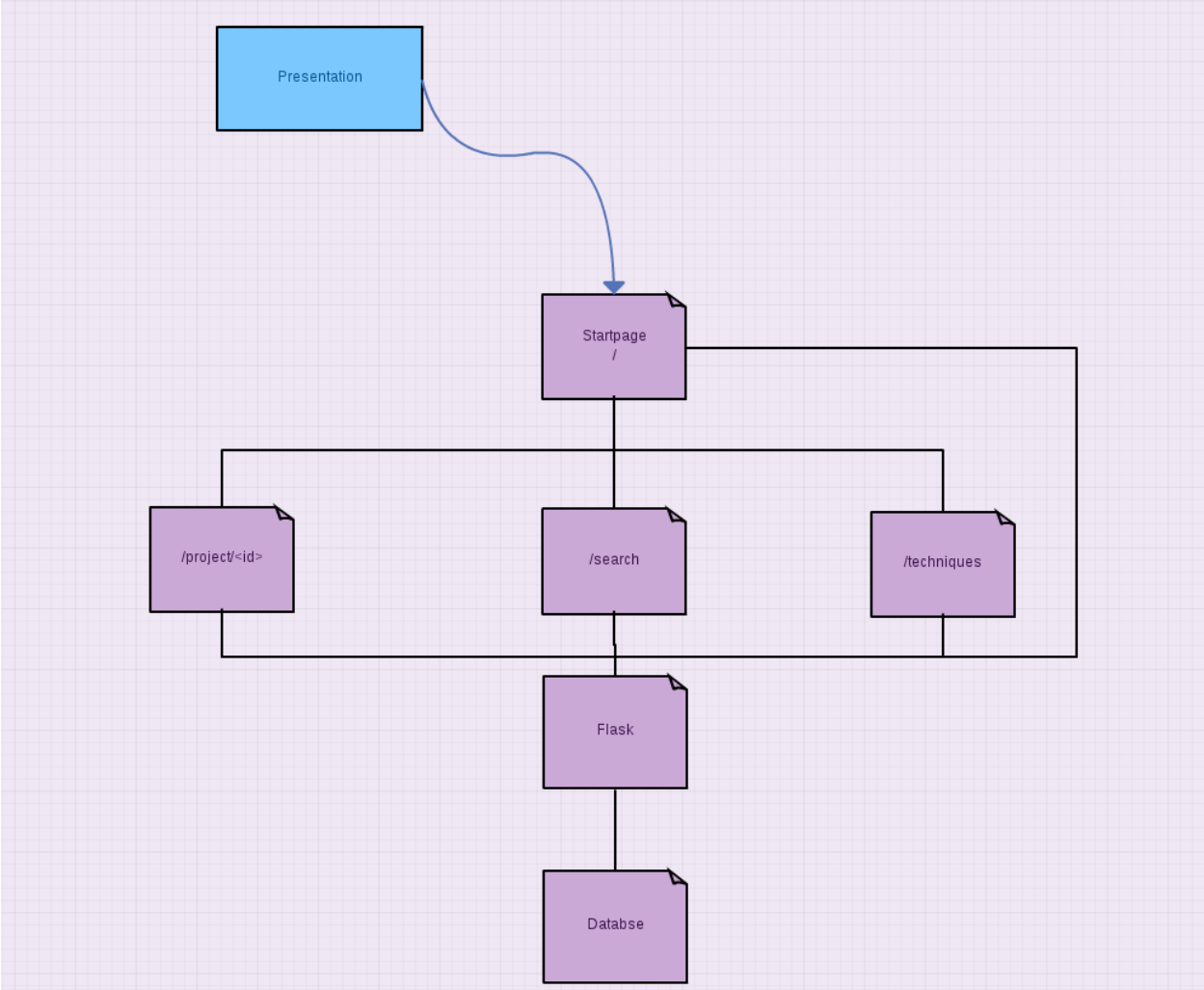
Ver.	Revisionsbeskrivning	Datum
1.0	Första versionen klar.	14/10-14

1. Översiktsbild av arkitekturen

Arkitekturen är uppbyggd kring 3 grundpelare.

Dessa är presentationslagret, datalagret, databasen med JSON filen.

En grafisk bild över sidans arkitektur, kort info om varje del finns under bilden med referens till den här bilden inom parentes.



1.1. Presentationslagret (/, /project/<id>, /search, /techniques)

Presentationslagret är den delen av sidan som visar sidan rent visuellt. Det använder sig av Flask och Jinja2 för att kunna producera HTML kod av python3 kod.
Jinja2 används inuti själva HTML sidorna för att kunna anropa kod från filen myFlaskProjekt som i sin tur använder sig av Flask för att kunna hämta data från datalagret och sedan rendera sidorna.

1.2. Datalagret (Flask)

Datalagret är den del av arkitekturen som laddar in en JSON fil och arbetar med den för att i sin tur skicka rätt information tillbaka till presentationslagret. Som ett exempel på detta används search funktionen i datalagret för att hitta matchande projekt som sedan ska visas i presentationslagret till användaren. Datalagret kräver att den är byggt utifrån detta givna API “http://www.ida.liu.se/~TDP003/current/portfolio-api_python3/”. Därför måste returvärden och parametrar stämma överens med det API som är givet.

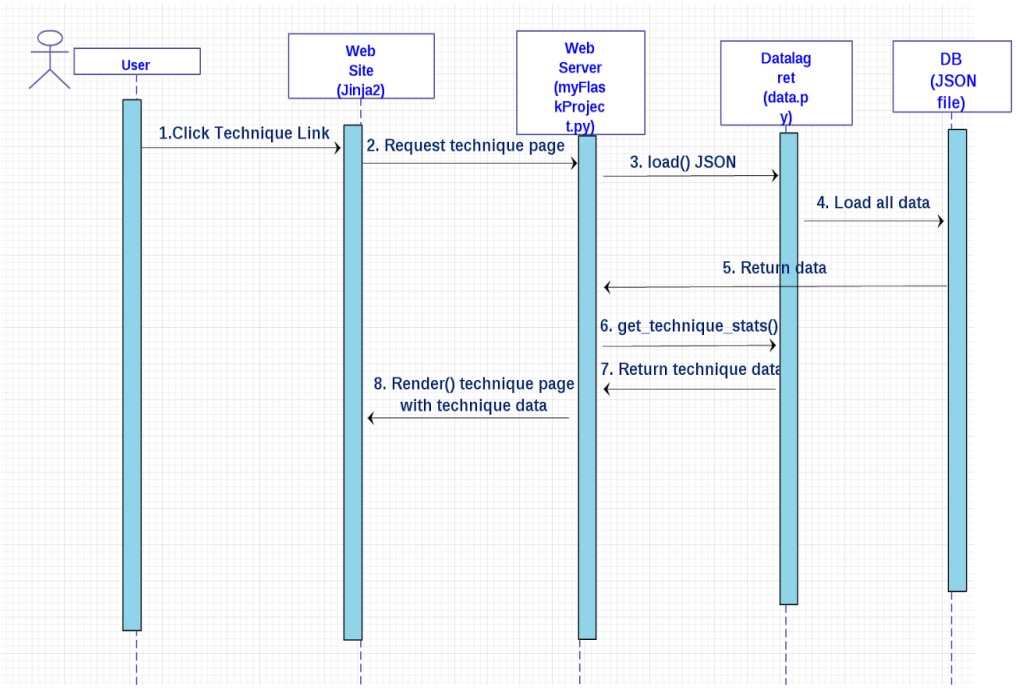
1.3. Databasen (Database)

Databasen(JSON filen) används för att spara respektive projekts information. Informationen hämtas av datalagret som i sin tur kommer arbetas med för att sedan visas i presentationslagret.

2. Detaljerad genomgång av arkitekturen

Här nedan finns en bild på hur systemet jobbar när man trycker på Teknik sidan. För mer detaljerade beskrivningar av systemet kan du gå till följande sida:
http://www-und.ida.liu.se/~erili445/html/namespacedata.html

Nedan finns en bild som grafiskt visar hur arkitekturen fungerar, hur de olika delarna skickar information och ger varandra information när användaren i det här fallet trycker på länken ‘Technique’/’Teknik’ på hemsidan.



3. Felhantering

Den primära felhanteringen sker av tries och excepts. Blir något fel under “Try” kommer den gå vidare till Except och exekvera den kod som finns där. I vårt fall loggas vad som gick fel i en logfil med information om vilken funktion som misslyckades och vilka parametrar den kallades med. Ett bra ställe att påbörja felsökning är därför logfilen. Logfilen ligger i rotmappen i systemet, alltså på samma ställe som JSON filen och datalagret.

Felsökning kan göras på två olika sätt. Ett bra felsökningsverktyg är “Python Debugger” även kallat “PDB”. Där kan man gå igenom en så kallad “traceback”. Alltså steg för steg se vad som faktiskt har hänt i exekveringen , vilka värden olika har.

Korta instruktioner om PDB
För att köra PDB måste du först i terminalen gå till mappen “MyPortfolio”. Skriv sedan:
“python3 -m pdb data.py”

Olika användbara kommandon:

“continue”: Kör filen tills nästa stoppunkt påträffas eller ett fel inträffar. Påträffas ett fel kommer en lista upp med en traceback. Man kan då röra sig i tracebacken för att se vad som hänt. Detta kan göras med nästkommande två kommandon

“up”: Rör sig uppåt en frame i tracebacken

“down”: Rör sig nedåt en frame i tracebacken.

Vill man kolla vad ett värde i en variabel har vid ett specifikt tillfälle kan man skriva variabelns namn för att se värdet.

Mer avancerad användning av PDB finner du i pythons dokumentation
“https://docs.python.org/2/library/pdb.html”.