

# TDP003 Projekt: Egna datormiljön

## Systemdokumentation

Författare

Daniel Huber, [danhu849@liu.se](mailto:danhu849@liu.se)  
Jens Öhnell, [jenoh242@liu.se](mailto:jenoh242@liu.se)

## 1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Systemdokumentation för Portfolio TDP003	151020

## 2 Översiktsbild

Hemsidans syfte är att presentera information om hemsidans ägare, och presentera project som denna har gjort på ett sökbart sätt. Den består av ett datalager samt ett presentationslager.

### 2.1 Datalagret

Datalagrets syfte är att gör information om olika projekt åtkommlig och sökbar. Det är skrivet i Python 3, och söker mot en json-fil.

### 2.2 Presentationslagret

Presentationslagrets syfte är att göra informationen tillgänglig till slutanvändaren på ett enkelt och intuitivt sätt. Det är skrivet i Python 3 med ramverket Flask samt templatemotorn Jinja2.

## 3 Specifikation - Presentationslagret

Skapandet av hemsidan har utgått från en kravspecifikation. Denna har varit vägledande i vilken funktionalitet sidan skall ha. Den fulla specifikationen kan hittas i dokumentet *Systemspecifikation av portfoliosystem*.

Sammanfattningsvis skall dock följande ingå i hemsidan:

- En förstasida med bilder.
- En söksida som gör det möjligt att söka och sortera på diverse fält för projekten.
- En tekniksida som gör det möjligt att sortera projekt på använda tekniker.
- Bilder, både thumbnails och fullstora, för all projekt.
- Begripliga felmedelanden.
- Korrekta statuskoder.

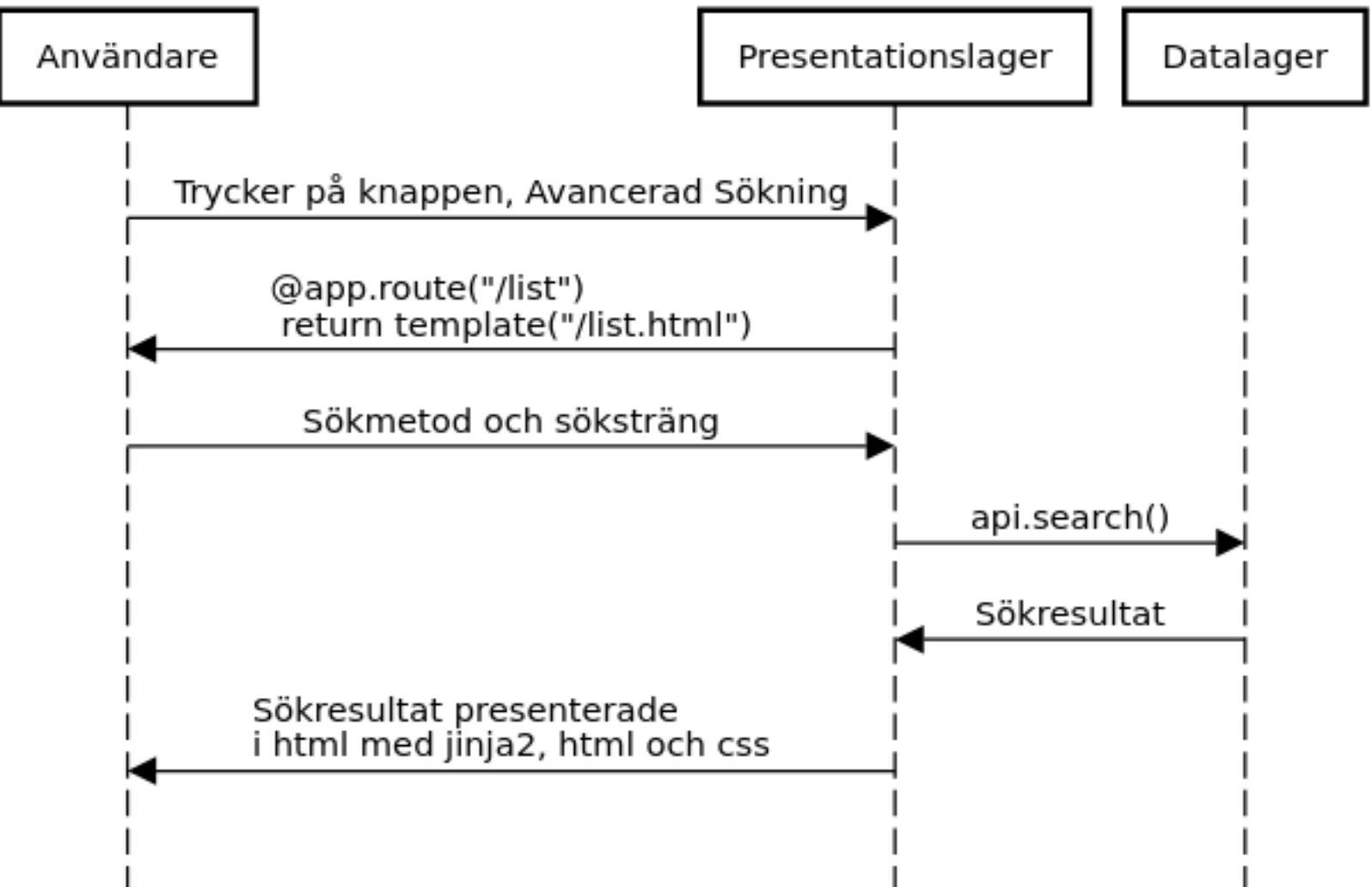
## 4 Sekvensdiagram - Sökning

??

## 5 Felhantering och Loggar

Samtliga portfoliehändelser loggas med hjälp av Pythons logging.config modul. Konfigurationen för loggningen definieras i filen <logging.cfg>. Loggar skrevs till filen <portfolio\_log.log> i kronologisk ordning. Senast log längst ner. Båda filerna finns i root katalogen för portfolio-appen. Loggar sparas med datumstämpel, viktighet, namn, tråd och meddelande. Alla loggar skrivs in i logfilen oavsett viktighet, men kan ändras vid behov genom att ändra <level=DEBUG> under <[logger\_root]> i konfigurationsfilen <logging.cfg>. En nivå som filtrerar bort mer än DEBUG rekommenderas ej då händelser som lett upp till kraschen kan

## Sekvensdiagram - Sökning



Figur 1: Sekvensdiagram för Sökning

ha filterats bort. Logfilen finns tillgänglig även när flask inte körs. Vid ny körning av flask skrivs inte `portfolio_log.log` över utan nya loggar skrivs in längst ner i filen.

Terminalkommandot `tail` med flaggan `-f` används vid flask run för att i realtid övervaka de senast tillagda loggarna i `<portfolio_log.log>`. Vid behov används flaggan `-nN` där `N` definieras av antalet rader som visas. Utan flaggan `-n` visas de sista 10 raderna från filen i terminalen. Att alltid de 10 sista raderna visas från filen möjliggörs av flaggan `-f`.

```
<code>tail -n15 -f /path/to/portfolio_app/portfolio_log.log</code>
```

Utöver logfilen hanteras fel med hjälp av python's `print()` funktion som kan skrivas in i varje `@app.route()`

funktion, men också egendefinierade funktioner. Informationen skrivs ut i terminalfönstret när funktioner aktiveras under en flask run körning. Med `print()` kan variabler från bland annat databasen komma åt. Flasks egen debug mode aktiveras genom att `FLASK_DEBUG` variabeln sätts till 1 innan flask run:

```
<code>export FLASK_DEBUG=1</code>
```

## 5.1 Testning/felsökning - Databas

Testning av datalagrets funktioner tillhandahålls av kursledningen och benämns enhetstester. Utöver de initiella enhetstesterna har fler med tiden lagts till. Enhetstesterna skrevs i python3 där pythons modul unittest (Unit testing framework) används. Testerna skrevs med kravspecifikationen i åtanke. Körning av testerna utan fel bevisar att funktionens krav uppfylls.

## 5.2 Docker Testning Lokalt

”Om ni vill får ni gärna ha med hur man startar en lokal Docker-avbild och testar den” - Filip Strömbäck

Vecka	Deadline Datum	Uppskattad	Beskrivning	Kunskap
37	Torsdag 10/9	6h	Planeringsdokument	God
	Lördag 12/9	3h	Sätta emacs i python-mode	Vag
	Söndag 13/9	2h	Inkorporera Magit i Emacs	Vag
38	Tisdag 15/9	3h	Bekantskap med Seleniumhq.org	Vag
	Torsdag 17/9	6h	Lofi-prototyp	God
	Torsdag 17/9	4h	Grundläggande Installations manual	God
39	Torsdag 24/9	24h	Projektplan, Första utkast	Vag
	Torsdag 24/9	24h	1:a Versionen gemensam installationsmanual	God
	Fredag 25/9	2h	Flask o Jinja2 Föreläsning	Vag
	Söndag 27/9	6h	Ha deploy:at första testhemsidan med flask	Vag
	Söndag 27/9	12h	Fatta Jinja2	Vag
40	Torsdag 1/10	1h	Bidra med icke-trivial förbättring git eller tester	God o inget
	Torsdag 1/10	1-2h	Korrigerar eventuella brister installationsmanualen	okänt
	Torsdag 1/10	1-2h	Korrigerar Brister, Projektplanen	Beror på
	Torsdag 1/10	24h	Datalagret Godkänt	Vag
42	Torsdag 15/10	-	Portfolion Publicerad	Vag
	Torsdag 15/10	3h	Systemdemonstration	Vag
	Torsdag 15/10	8h	1:a Versionen Systemdokumentation	Vag
43	Torsdag 22/10	8h	Testdokumentation inlämnad	Vag
	Torsdag 22/10	6h	Individuellt Reflektionsblad	Vag
	Torsdag 22/10	2-3h	Korrigerat event. brister i systemdokumentationen	Vag
Alla		20min/dag	Dokumentera Dagbok	God

## 5.3 Tidsåtgång - Presentativ Del

5h	Statisk eller dynamisk sida med bilder på /
8h	Dynamisk sida som sorterar och listar projekten på /list
2h	Visar fullständig infosida för specifikt proj med id på /project/id
3h	Sammanställning av alla project baserat på använda tekniker på /techniques

## 5.4 Tidsåtgång - Funktioner i datalagret

Nedan listas den specifika tidsåtgången för implementeringar av respektive funktion i datalagret.

Tid	Funktion
2h	load
1h	get_project_count
1h	get_project
3h	search
2h	get_techniques
2h	get_technique_stats