TDP003 Projekt: Egna datormiljön

Systemdokumentation

Författare
Maximilian Bragazzi Ihrén, maxbr431@student.liu.se
Markus Lewin, marle943@student.liu.se

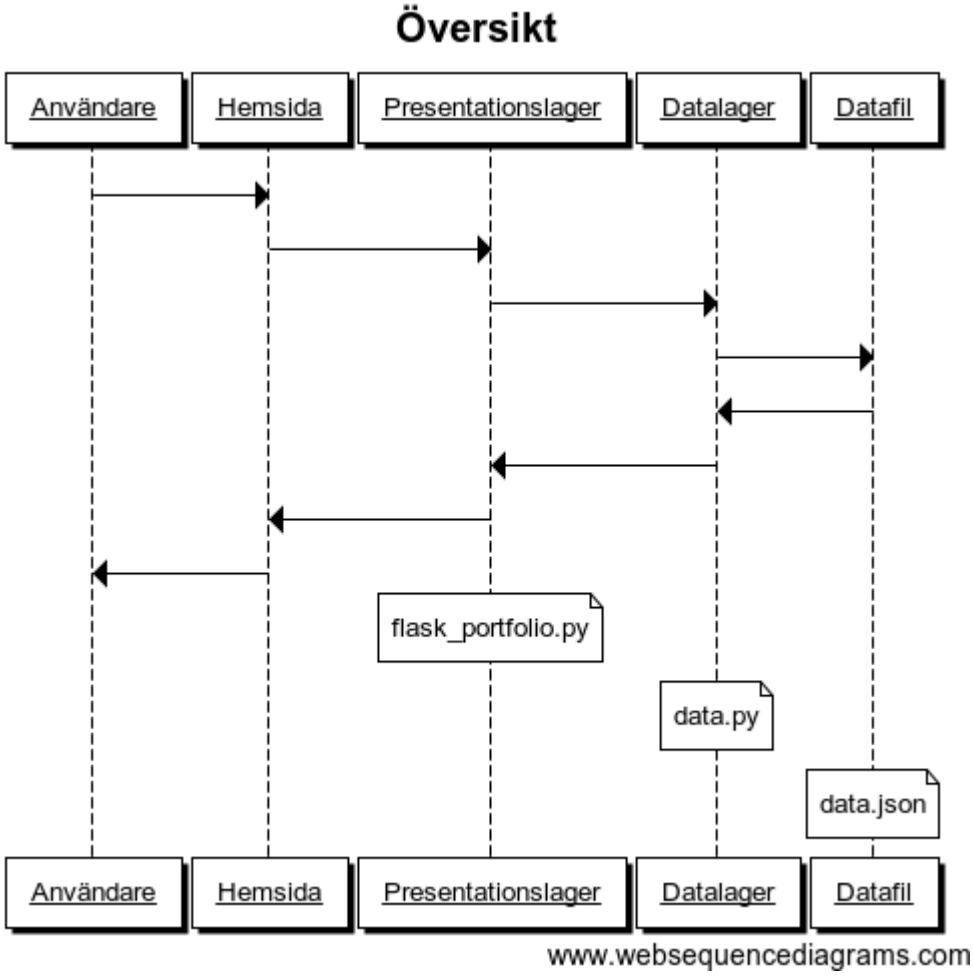
Höstterminen 2014
Version 1.0

1. Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Skriven för inlämning	141015

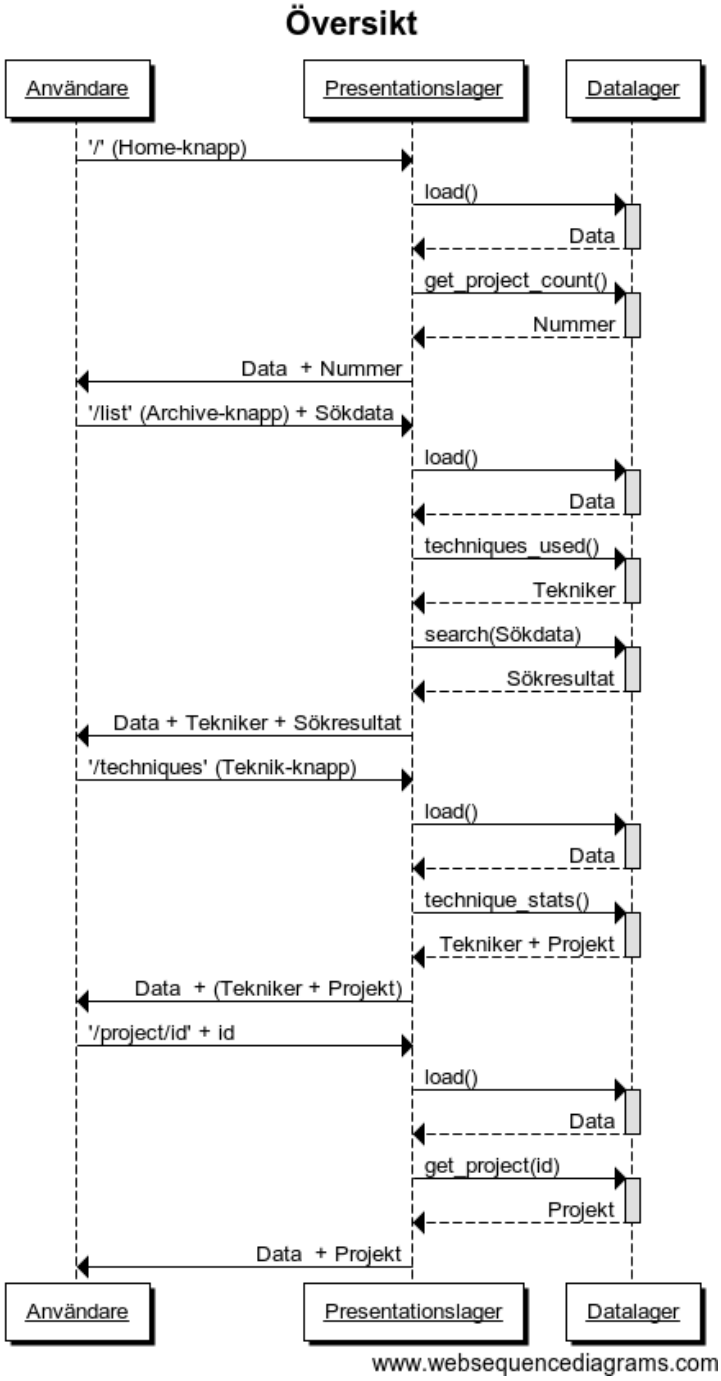
1. Översikt

1.1. Simpel översikt



Här visas en enkel förklaring för hur systemet fungerar. Användaren kommunicerar med hemsidan, som kommunicerar med presentationslagret, som i sin tur kommunicerar med datalagret, som tillslut kommunicerar med datafilen. Sedan skickas datan tillbaka genom hierarkin.

1.2. Avancerad översikt

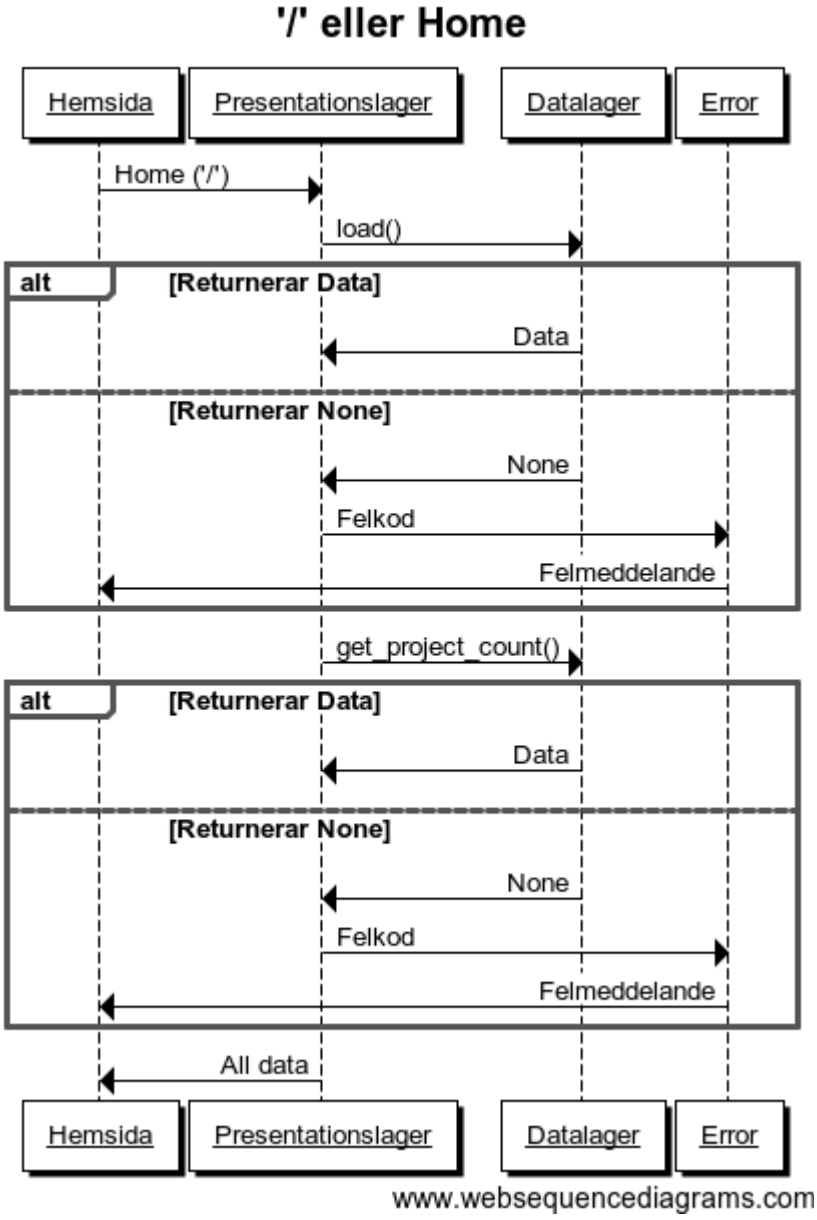


Här presenteras en översiktlig bild av systemet där användaren navigerar sidan. Till exempel: när användaren klickar på Home-knappen skickas '/' till presentationslagret, som i sin tur anropar funktionen load() från datalagret. Funktionen returnerar i sin tur data. Presentationslagret anropar även funktionen get_project_count() som returnerar ett nummer. Datan och numret skickas tillbaka från presentationslagret till hemsidan där användaren kan se det.

2. Presentationslager

Presentationslagret består av hemsidan och funktionerna bakom den. Det är till exempel den som håller reda på vad som ska skrivas ut och var det ska skrivas ut. Den ansvarar kort sagt för allt användaren kan se på hemsidan och att det hämtas.

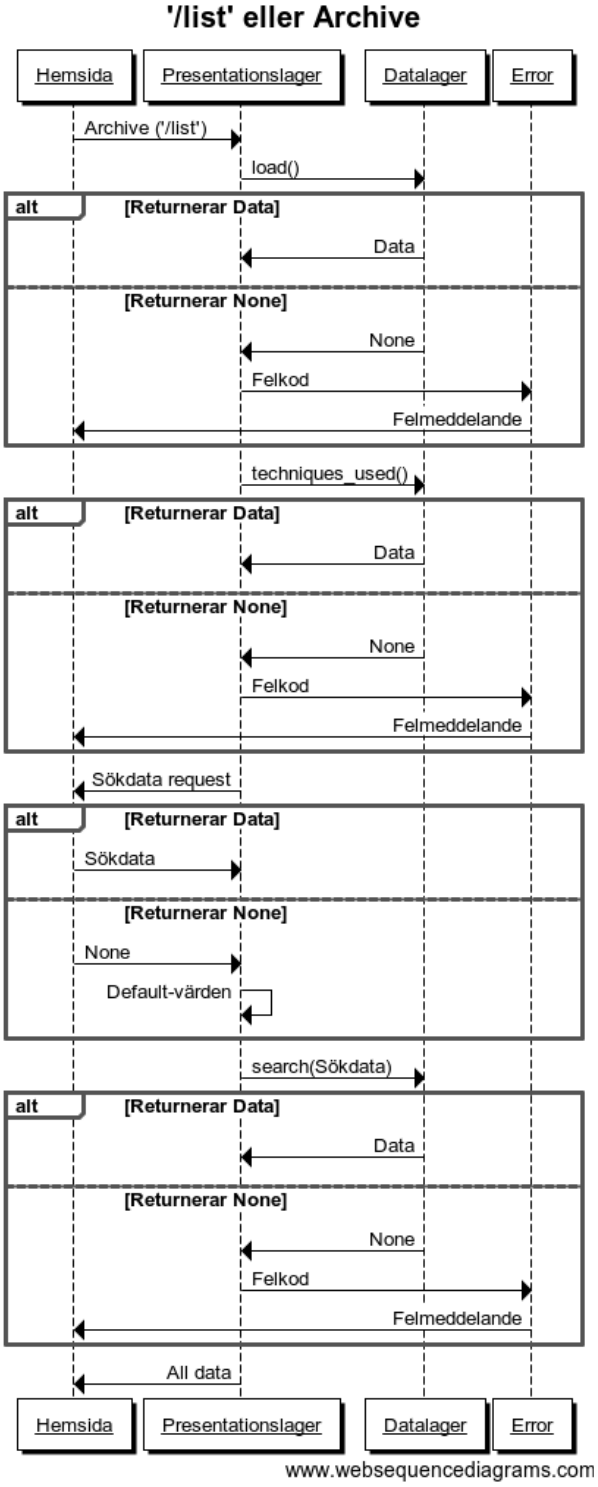
2.1. '/' (Home)



Det här är en mer beskrivande bild av funktionerna som används när användaren visar startsidan på hemsidan. När användaren trycker på knappen Home så anropar presentationslagret funktionen load(). Datalagret returnerar antingen data eller None. Beroende av vad som returneras kommer presentationslagret att antingen skicka ett felmeddelande till errorsidan som i sin tur skickar ett

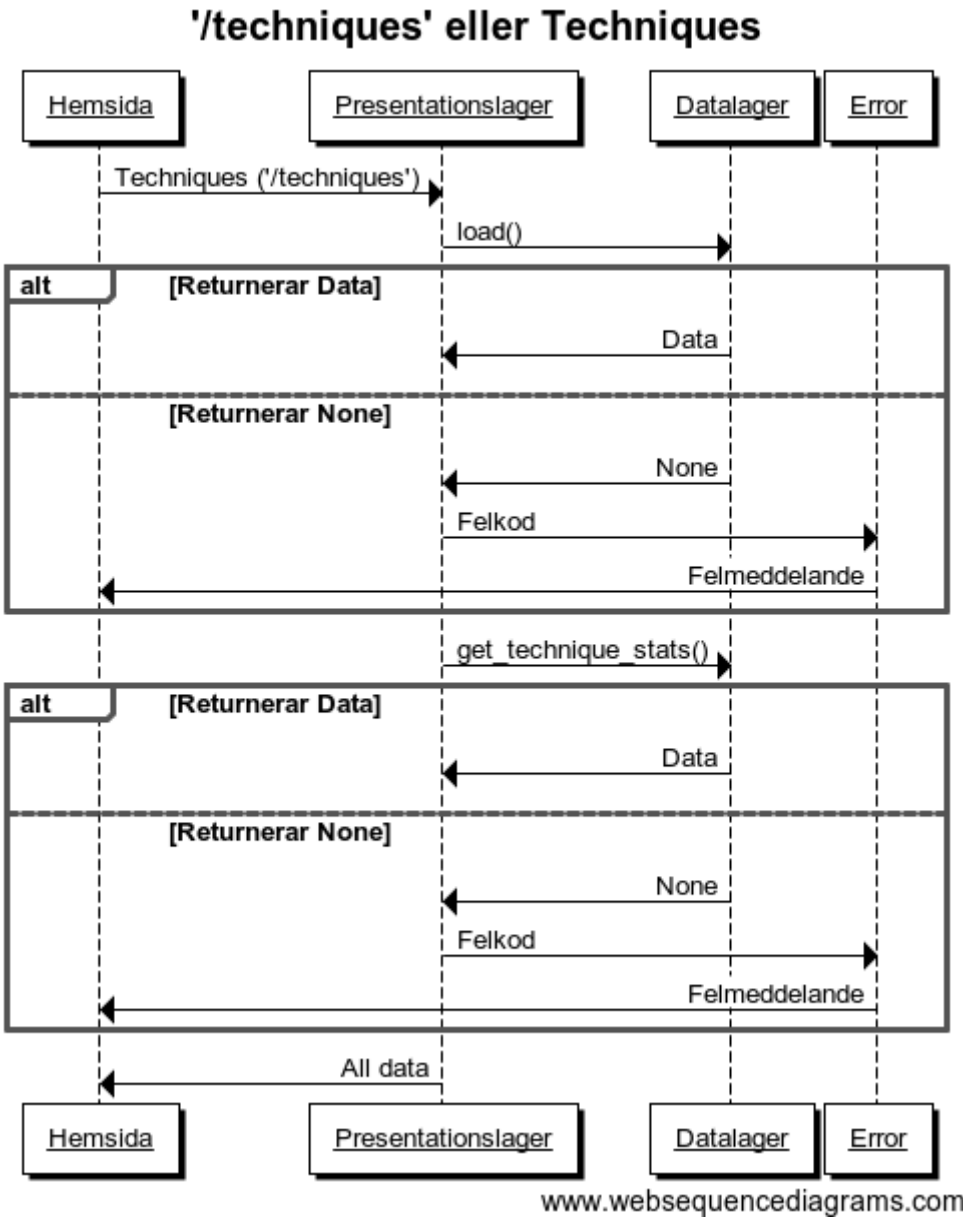
felmeddelande till hemsidan. Om load() däremot returnerar data så går programmet vidare till nästa punkt, vilket är att få get_project_count(). Där går den igenom samma process som innan. Om all data har returnerats till presentationslagret skickar presentationslagret vidare datan till hemsidan där det visas upp för användaren enligt mallen.

2.2. '/list'



Denna bild visar systemets flöde när användaren klickar på Archive-knappen eller söker efter någonting med hjälp av sökrutan. Presentationslagret anropar då load() som antingen returnerar data eller None. Om load() returnerar None så skickas ett felkod från presentationslagret till error, som då i sin tur visar ett felmeddelande på hemsidan. Annars går programmet vidare och hämtar använda tekniker via funktionen techniques_used(). Om None returneras går då systemet igenom samma felprocess som tidigare. Annars går systemet vidare och frågar hemsidan (URL:en) efter sökdata. Om den inte returnerar sökdata anger presentationslagret default-värden till de variabler som finns. Slutligen går den till sökfunktionen som går igenom felprocessen även den. Om allt har gått bra returnerar presentationslagret all data till hemsidan där den visas för användaren enligt mallen.

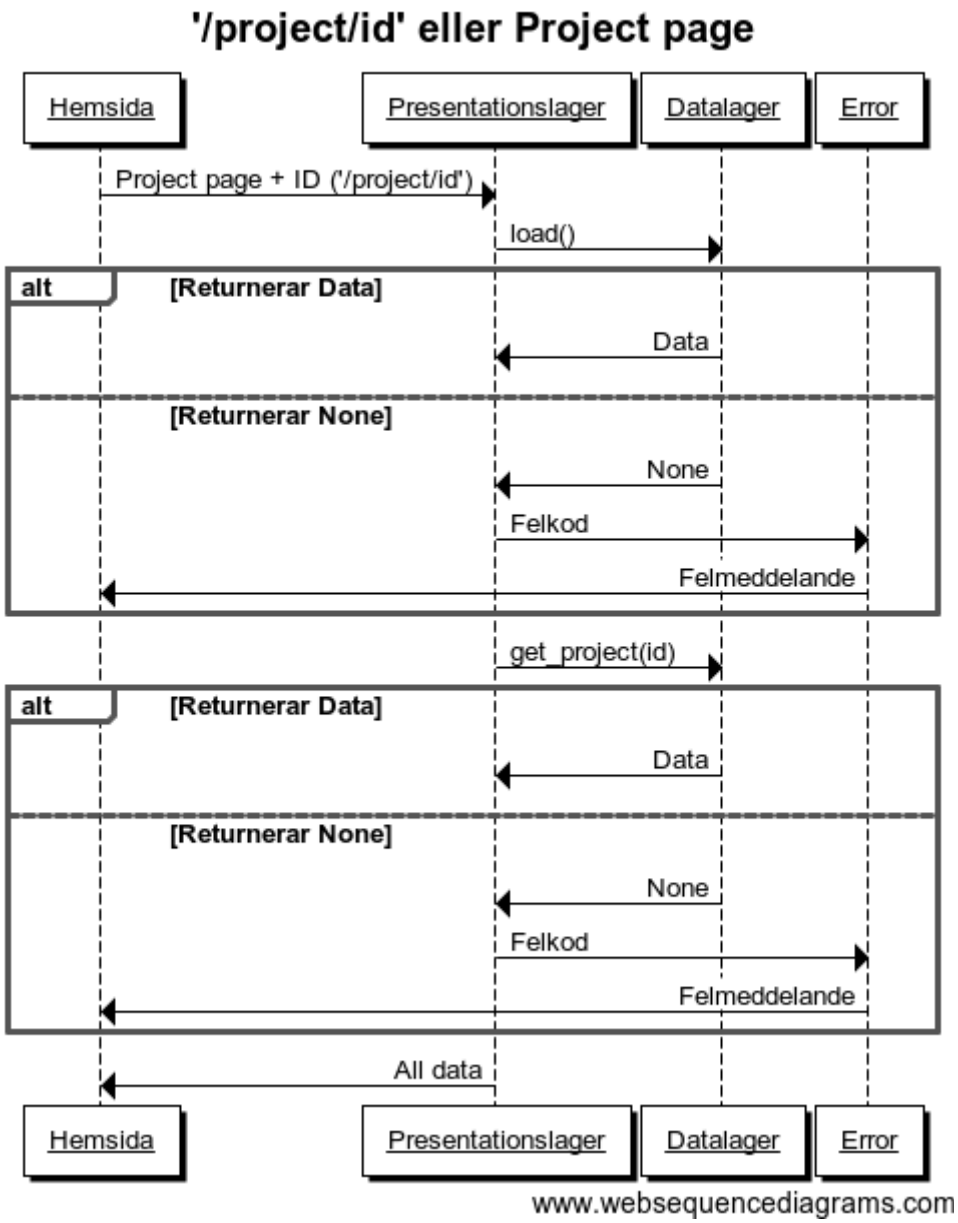
2.3. '/techniques'



Det här är sekvensschemat för när användaren använder sig av teknikfunktionen.

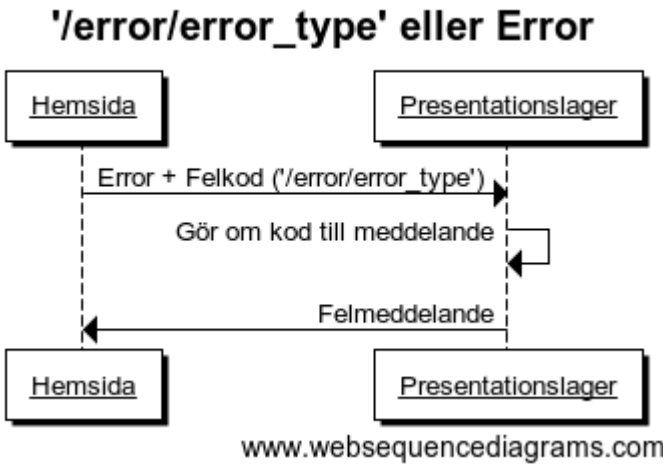
Presentationslagret anropar funktionen load(), om den returnerar None så skickas en felkod till error, som i sin tur skickar ett felmeddelande som ska visas på hemsidan. Annars går den vidare till att anropa funktionen get_techniques_stats(). Om get_techniques_stats() returnerar None så går systemet igenom liknande felprocess som tidigare. Annars, om allt går bra, skickas all data direkt till hemsidan där den visas upp enligt mallen.

2.4. '/project/id'



Detta är hur projektsidan fungerar. Det börjar med att användaren skickar in ett id till det projektet denna användare vill se. Funktionen load() anropas från presentationslagret. Beroende på vad den returnerar så kan en felkod skickas till error, som i sin tur skickar ett felmeddelande till hemsidan, där det visas upp. Om det annars går bra försöker den hämta projektet från databasen med hjälp av det ID som matades in tidigare. Om allt går bra skickas all data till hemsidan för att visas upp enligt mall.

2.5. '/error'

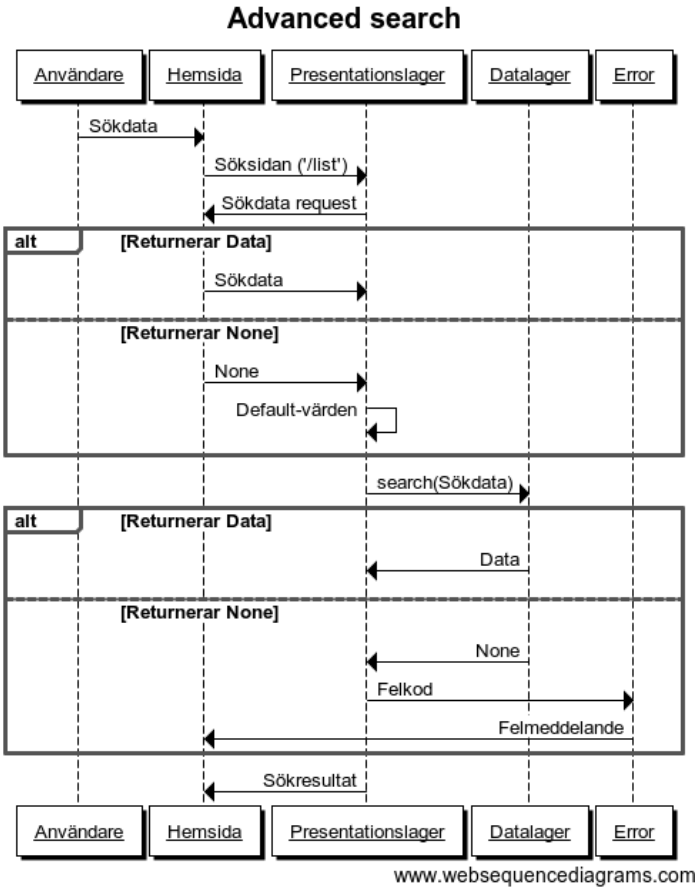


Det här är hur errorsidan fungerar. När den anropas matas en kod in, som i sin tur error gör om till ett felmeddelande. Detta felmeddelande skickas sedan till hemsidan där det visas upp enligt mall.

2.6. Mallar

Våra mallar är uppbyggda på så sätt att vi har en layout-fil som hanterar det generella som finns med på alla sidorna, som till exempel navigeringsknapparna. Varje mall, i sin tur, anropar då denna layout-fil och bygger ut det den behöver enligt layout-filens upplägg.

2.7. Avancerad sökning



Här ser du hur den avancerade sökningen på söksidan ('/list') fungerar. Om du vill ha en mer detaljerad förklaring på hur söksidan i sig fungerar, gå till punkt 2.2.

Det första som händer är att användaren matar in sitt sökdata in i den avancerade sökningsrutan. Detta uppföljs då av att hemsidan anropar söksidan igen, som i sin tur frågar efter sökdatat från hemsidan (URL:en). Om sökdatan inte hittas så sätter presentationslagret variablerna till sina default-värden. Sedan går den vidare till search-funktionen, där den matar in all sökdata som användaren matade in i systemet. Om denna funktion då returnerar None, så skickas en felkod till Error som sedan skickar ett felmeddelande till hemsidan för att visas upp.

Om search-funktionen faktiskt returnerar ett värde så skickas resultatet av sökningen till hemsidan där det skrivs ut enligt mall.

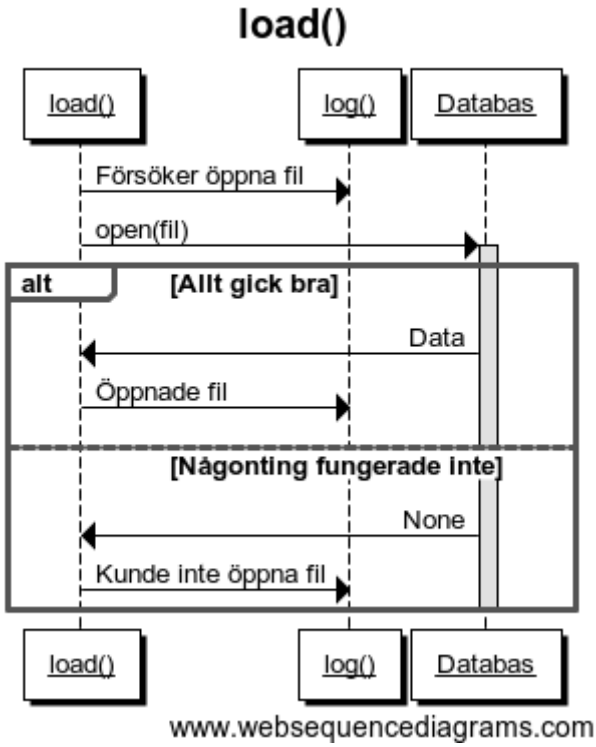
3. Datalager

Datalagret består av funktionerna som systemet hela tiden använder sig av. Det är främst presentationslagret som kommunicerar med datalagret och använder funktionerna där för att exempelvis ladda databasen, söka i den och hämta specifik information från den. Det mesta som sker i datalagret loggas i en separat fil för att lättare kunna felsöka problem i systemet.

Följande är en länk till datalagrets API:

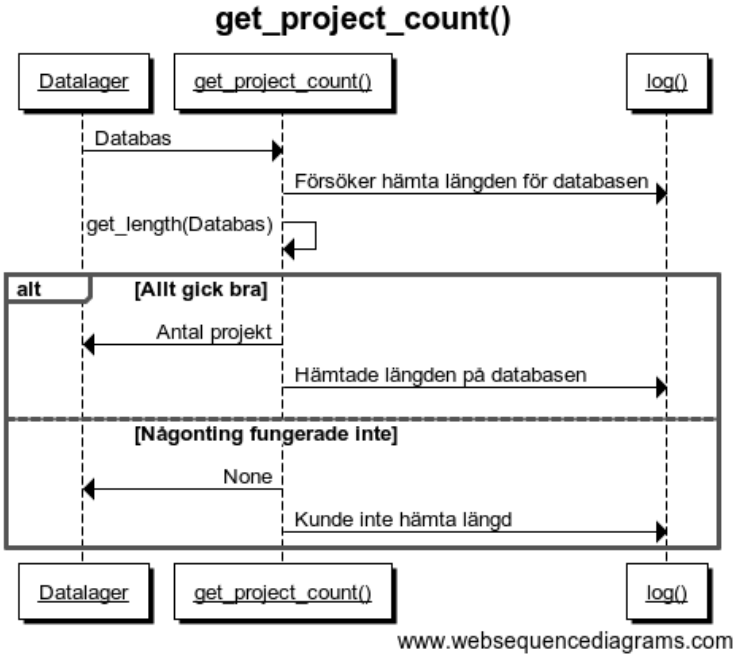
http://www.ida.liu.se/~TDP003/current/portfolio-api_python3/

3.1. load()



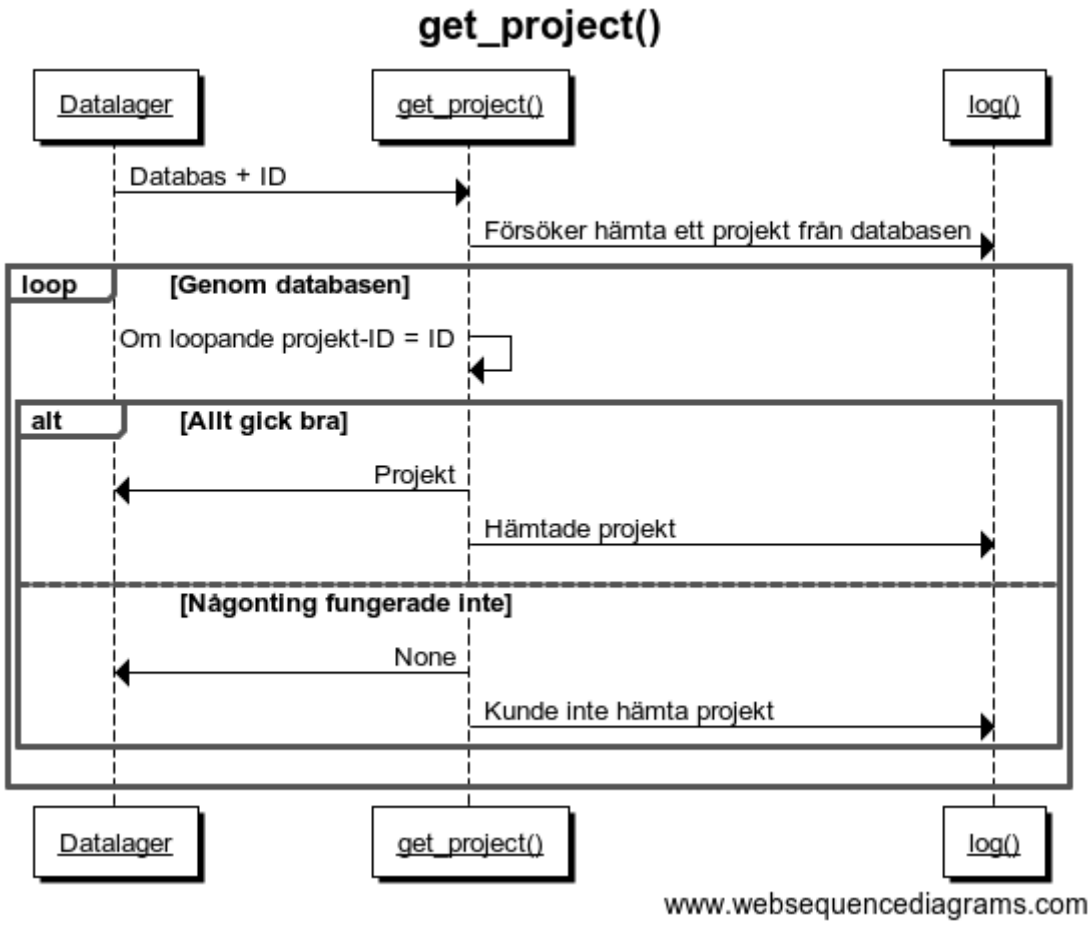
Det här är ett sekvensdiagram för hur load-funktionen fungerar i datalagret. Funktionen börjar med att försöka öppna en fil med hjälp av funktionen open(). Om det gick kommer data att returneras, och fungerade det inte kommer None returneras. Allt loggas genom att information om varje steg skickas till log-funktionen.

3.2. get_project_count()



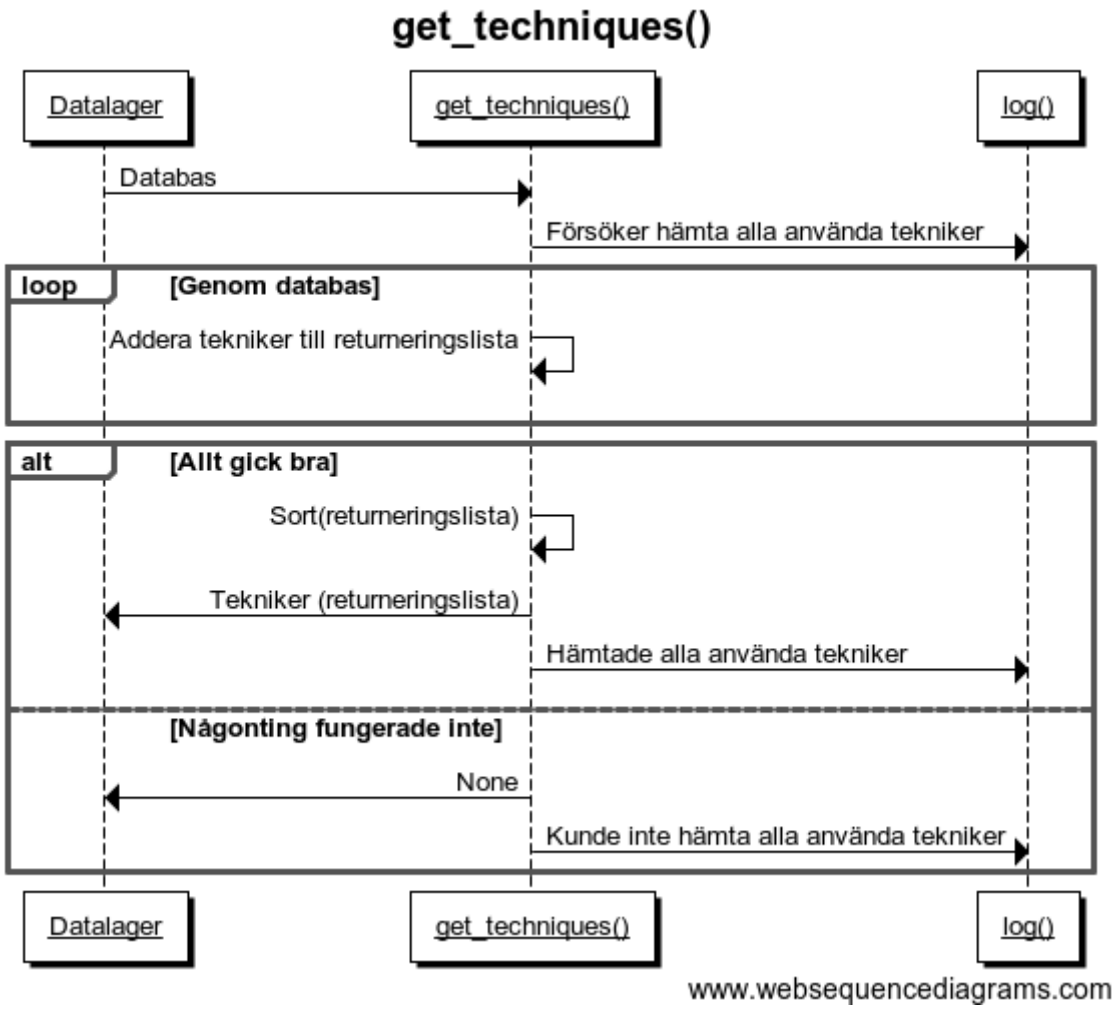
Här ser du ett sekvensdiagram för get_project_count(). Det första den gör är att den får en databas inskickad i funktionen. Därefter så försöker den hämta längden på databasen. Om allt gick bra skickar den tillbaka antalet projekt, annars skickar den tillbaka None. Allt loggas via log-funktionen.

3.3. get_project()



Funktionen get_project startar med att ta in en databas och ett ID. Den loopar igenom databasen och kollar om det loopande projekt-ID:t matchar det inmatade ID:t. Om det hittar en matchning skickas projektet tillbaka. Om den kommer ut ur loopen utan att ha skickat tillbaka någonting skickar den tillbaka None. Allt loggas via log().

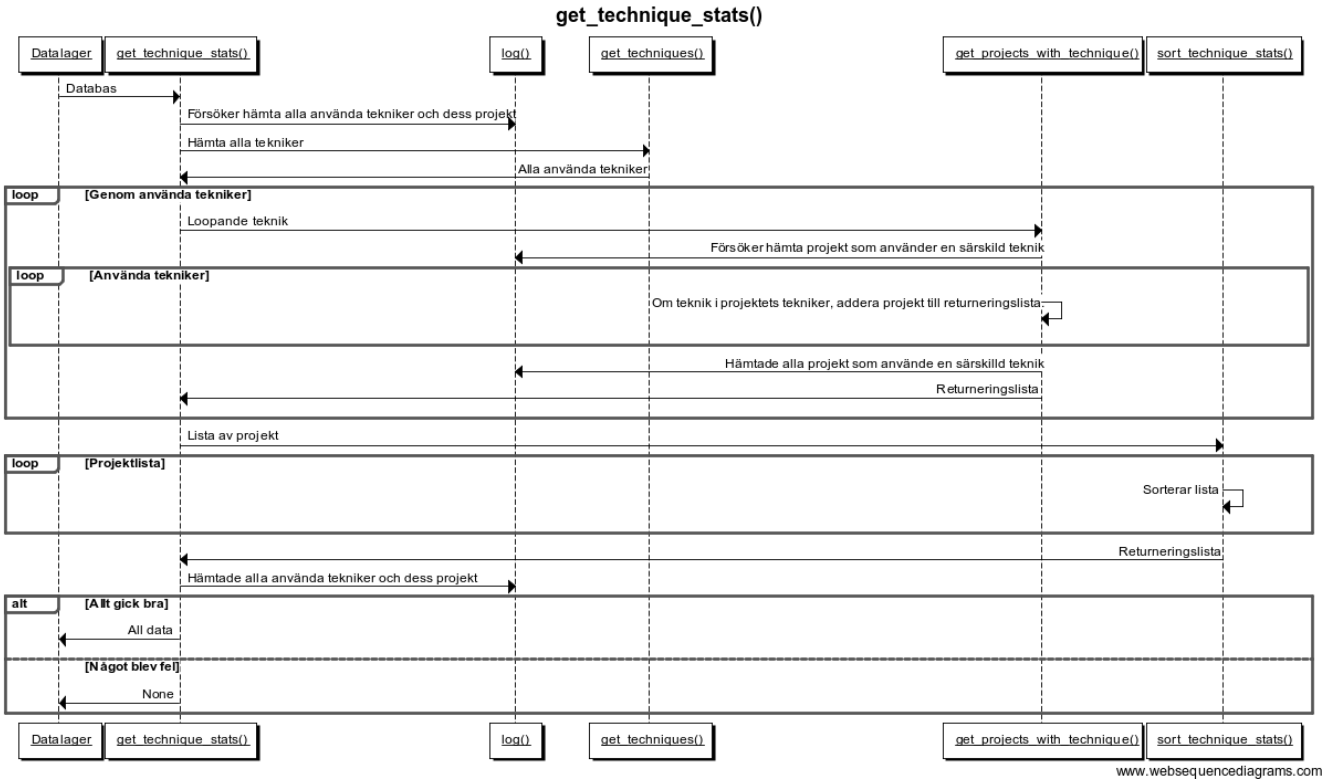
3.4. get_techniques()



Funktionen get_techniques() börjar med att ta in en databas. Den fortsätter med att addera alla tekniker den hittar till en lista som den ska returnera (den lägger inte in något i listan om det redan är däri). Om allt går bra sorterar den sedan listan innan den skickar tillbaka den. Annars returnerar den None.

Allt som görs i funktionen loggas via log().

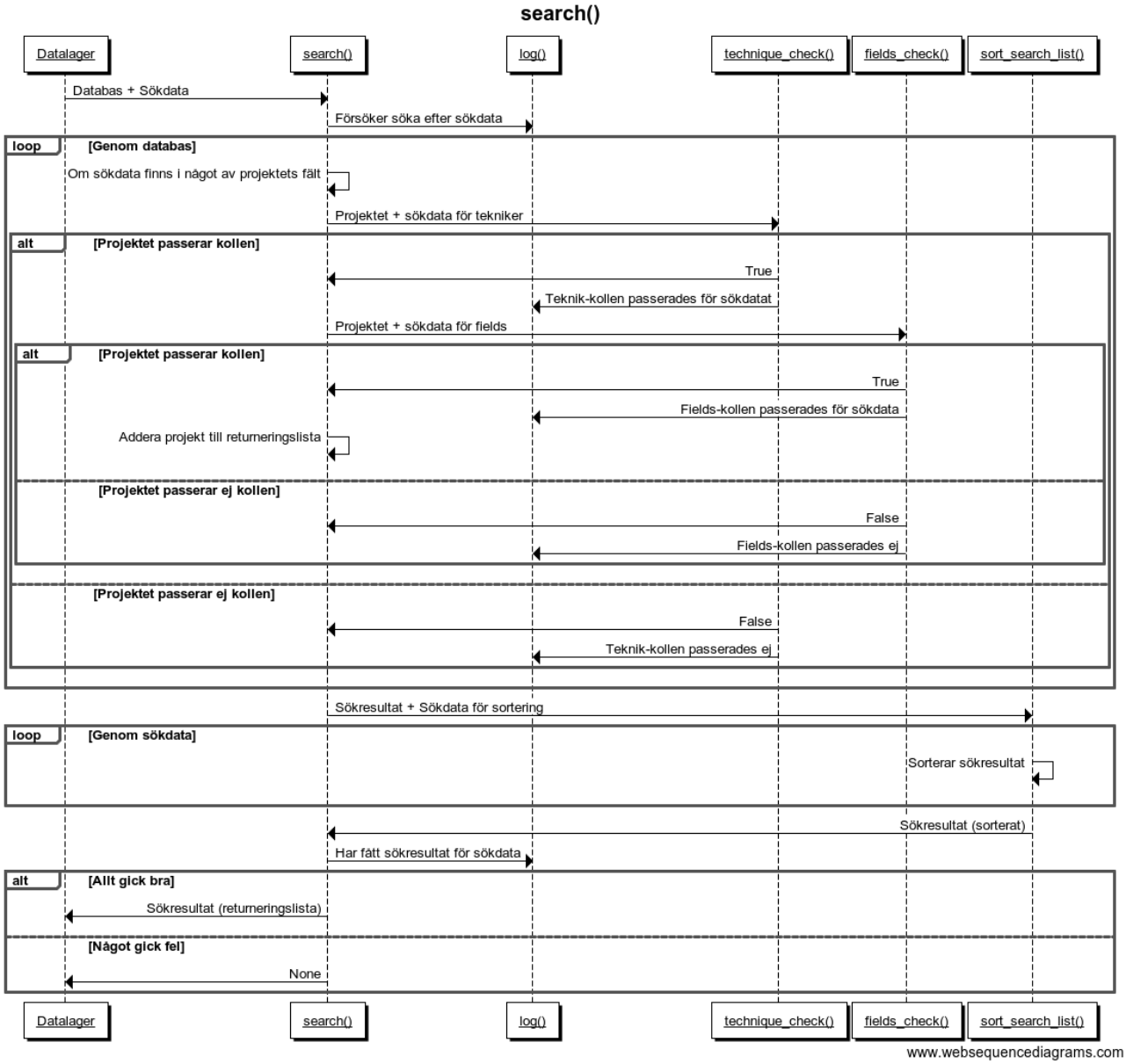
3.5. get_technique_stats()



get_technique_stats() tar först in en databas. Den anropar sedan funktionen get_techniques() (se punkt 3.4) som returnerar alla projekt använda. Funktionen loopar sedan igenom alla teknikerna och skickar dem var för sig till funktionen get_projects_with_technique(), som loopar igenom alla projekt i databasen och kollar om tekniken du skickade in finns med i teknikerna som projektet använde. Om den finns med där sparas den till en lista.

Denna lista returneras då till get_technique_stats(). När funktionen har loopat klart, skickar den listan av projekt till sort_technique_stats(), vilket sorterar listan enligt projektens namn. sort_technique_stats() returnerar sedan tillbaka listan av projekt. Om allt då har gått bra, skickar get_technique_stats() all data tillbaka, annars returnerar den None. Allt som händer loggas via log().

3.6. search()

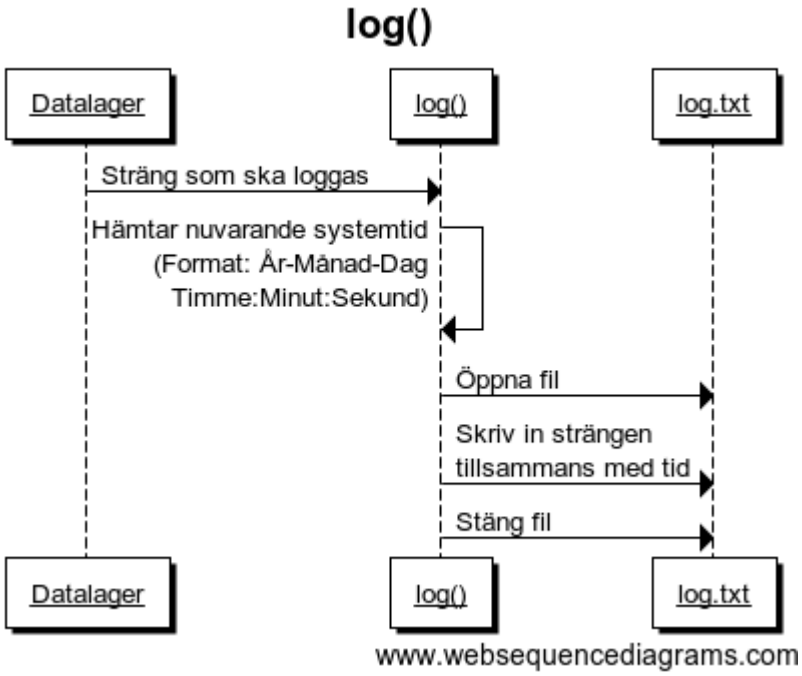


I sökfunktionen (search()) börjar vi med att skicka in databasen och sökdata som den ska söka efter (samt kriterier, så som vilka teknik-fält man får söka i, etc.). Den loopar då igenom databasen och kollar om sökvärdet finns med i något av projektets fält. Om det stämmer, skickar den projektet och relevanta sökkriterier till technique_check(), vilket kollar att det passerar kriterierna. Om den returnerar True så skickas projektet vidare till fields_check() tillsammans med relevanta sökkriterier för denna funktion. Om den returnerar True läggs projektet till i sökresultatet.

När search() har loopat igenom klart, så skickar den sökresultatet och de relevanta sökkriterierna till sort_search_list(), som sorterar resultatet.

Om allt sedan gick bra returneras sökresultatet, annars returneras None. Allt loggas då via funktionen log().

3.7. log()



Funktionen log() är nog en av de enklaste funktionerna, men fortfarande den viktigaste.

Den börjar med att ta in en sträng som ska loggas till log-filen. Den hämtar nuvarande tiden från systemet, och skriver sedan ut tiden tillsammans med strängen till log-filen (log.txt).

4. Felhantering

Några exempel av systemets felhantering har visats i tidigare sekvensdiagram, exempelvis processen där värdet None har returnerats och felmeddelanden loggas. När en funktion som anropas från presentationslagret returnerar None kommer lagret skicka vidare användaren till en av error-sidorna beroende på vilken funktion som använts. Om load-funktionen returnerar ett None värde kommer alltså en error-sida presenteras med texten

> The database couldn't be read

4.1. Logg-fil

Det kanske mest effektiva sättet att felsöka problem i systemet är att använda sig av logg-filen som finns sparad som ”log.txt” i samma bibliotek som datalagret. I logg-filen skrivs allt som datalagrets funktioner försöker att göra tillsammans med datum och tid för händelsen.

Raden

> 2014-10-16 10:54:45 | LOAD: Trying to load datafile 'data.json'

i logg-filen visar till exempel att en funktion i datalagret försökte att ladda datafilen 'data.json' klockan 10:54:45 2014-10-16. Ordet i versaler visar i vilken funktion i datalagret som händelsen har skett. Detta kan vara viktig information i fall meddelandet skulle visa att något gått fel.

4.2. Error-sidor

Systemet tillhandahåller även error-sidor för större problem, dessa visas om någon utav databasens huvudkomponenter saknas, exempelvis om själva databasen inte kan läsas, eller om tekniker och projekt inte kan hittas. När en error-sida visas är det stor risk att databasen saknar komponenten som nämns i meddelandet.

När error-meddelandet

> A project with that ID does not exist

visas är risken alltså stor att projektet saknas eller är felformaterat i databasen.

4.3. Felsökning

Vid felsökning av systemet rekommenderas att först se om ett errormeddelande visas på sidan. Errormeddelandet beskriver vad som går fel, vilket kan ge en fingervisning om vad en senare bör leta efter i loggfilen.

Nästa steg är att leta i loggen efter felmeddelandet. Eftersom loggen snabbt fylls med en mängd olika meddelanden är det rekommenderat att först rensa loggen, men endast om man är säker på att man kan återskapa problemet. Detta kan göras genom att radera all text i själva filen, men det lättaste är att radera hela filen. Loggfunktionen i datalagret kommer då skapa en ny fil vid nästa log-anrop.

När loggen är rensad kan användaren navigera till problemet via hemsidan. Med en rensad loggfil kan man följa vad datalagret gör för varje steg i navigeringen. När användaren har nått sidan där problemet uppkommer bör ett meddelande i loggfilen innehålla information om i vilken funktion felet inträffar.

Som exempel visar felmeddelandet

> LOAD: Failed to get datafile 'data.json'

att problem uppstår i funktionen ”LOAD”.

För att åtgärda felet kan användaren öppna datalagrets fil ”data.py” för att se vad funktionen gör när felmeddelandet loggas.

4.4. Gjorda feltester

4.4.1. Felformaterad sökväg

Portfolion kan hitta ett projekt med felformaterad sökväg. Portfolion visar då ingen bild. Det enda som skulle kunna räknas som felformaterat är ett projekt som börjar med ”/” eller ”images”.

4.4.2. Hitta projekt-ID som inte finns

Testet genomfördes genom att gå in på projektsidan och skriva in ett ID som inte existerar. Resultatet blir att användaren skickas till en felsida med meddelandet att projektet inte finns.

4.4.3. Att söka på '"'

För att försöka stänga strängen i sökvariabeln testade vi att mata in citattecken i sökrutan. Sökningen skedde dock som vanligt.

4.4.4. Att söka på symboler

Vi har även gjort försök att söka på symboler som exempelvis åäö/*+?. Inget fel upptäcktes dock och sökningen skedde som vanligt.

4.4.5. Ändra variabel i URL

En variabel i URL:en har ändrats manuellt på archive-sidan, inget hände dock.

4.4.6. Ändra namn i URL

Ett namn i URL:en har ändrats manuellt på archive-sidan, den fick då ett default-värde.

4.4.7. Sökning med kod

Test har också gjorts på att söka med kod som skulle kunna skada systemet. Sökfunktionen sökte endast på koden utan att exekvera den.