LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

Systemdokumentation

Författare

Arturas Aleksandrauskas
Joakim Johansson

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2016
Version 1.1

2016-10-12

Innehåll

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
2.2	Lagt till allt.	161013
2.1	Lagt till allt.	161013
2.0	Lagt till allt.	161013
1.5	Logging, Error handlingöch Toolsinlagt och första utkast inlämnat	161016
1.4	Description of the system components- Datalagretär första utkast inlämnat	161016
1.3	Description of the system components- Presentationssidanär klart	161015
1.2	User Functionsöch Folder Structuresectionerna är skrivna tillagda	161015
1.1	Skissat och lagt till sequensdiagram och skrivit klart Överview of the systemöch Front-page exemplet	161014
1.0	Dokumentet skapas och enkel information har lagt in som en start, A simple descriptio-nöch Installationsmanualär tillagt och klart	161012

2 A simple description

This project is a simple webpage that uses a python backend that servs the pages thought a module called Flask.

The purposes of this website i to present all list all projects the user has previously been involved in. A visitor of the site can scroll though all, or search for a specific project.

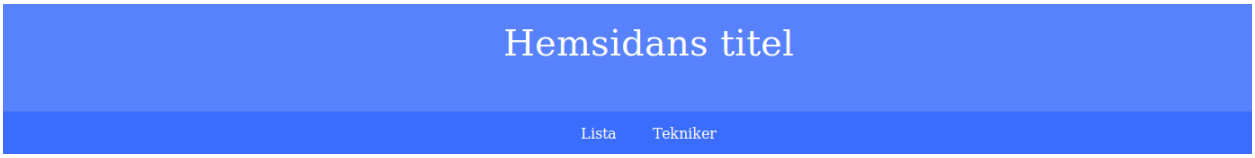
3 Installationsmanual

If you got this systemdocumentation file, you have probably also got a seperate pdf file that is the Installationmanual. Use that if you are going to install the system om a new computer.

In case you haven't got the PDF, here is the link to an online version:
`https://gitlab.ida.liu.se/antsu07/tdp003-installationsmanual-2016/blob/master/installationsmanual.pdf`

4 User Funktions

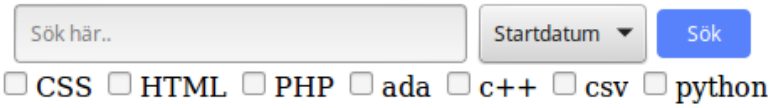
The user will we able to navigate around on four differant pages.



On the top of the webpage there is a header with the name of the website, wich the user always will be able to click to come back to the frontpage really fast.

Under the header, there is a navigation bar with two links. The List page and the Technologies page. This navigationbar with be accessable from all pages on the site.

The first page is a simple frontpage that displays userful about-information about the websites author. The page also displays all the three latest projects. Visitors can ofcource click on the projects to show more information.



The second page is the List page, which the user can get to by clicking in the navigationbar or searching from the 404 error page. The list page is made up of a long list of all projects in the database. There is also a box of tools filter the list. Visitors can search with free-text, choose requierd techniqes, and choose how the list should be ordered. And ofcause there is a search button to apply the filters to the list.

The third page is the Technologies page. This is more of a overview over all projects with the techniques in focus. This page is great for visitors to quickly see what typ of techniques the author users the most. But it is also really handy when a visitor looks att a project that uses ex. python and want to now other project the author have done where he also has used python.

The fifth page is a page that only shows a specific project. This page is hidden in the menu and you can only get to the page if you click on a project somewhere. This page shows a lot of information about the project, ex. the tecniques used, start and end date, peoples in the project and more. If the visitor clicks on a technique, he will be directed to the technique page and moved down to the right place for the clicked tech.

The last page we have are a error page that are displayed if the user tries to go to a page that don't exist. Simply as that. The page as a title, a short text and a search box so that the user can continue trying to find the right page.

5 Folder structur with description of the files

```
.
  Root/
      doc/
      static/
          images/
          style/
              style.css
      templates/
          index.html
          list.html
          project.html
          technologies.html
          error.html
      README.md
      data.json
      data.py
      data_test.py
      myFlaskProject.py
      logfile.log
.
```

5.0.1 Folder: doc

A folder with all documents related to this project but isn't actulay code, like this document for example.

You can also find, the projectplan document, lofi prototype and the developers diary if you wanna follow the development, in this folder.

5.0.2 Folder: static

A folder with all the static files, like css, images and fonts.

We have no images yet, but we have added a folder incase we wanna do future graphical enhancements to the site. We also have a folder with the name style with a single css file inside it that is used to style all pages.

5.0.3 Folder: templates

A folder for all the HTML templates.

We user five templates. One for each page on the site. index.html for the front page, list.html for the list page, project.html for the page that shows one project, technologies.html for the page with all the projects sorted be tech and error.html for the error page.

5.0.4 File: README.md

A description file with info about the project and good information for outside people incase they open up this project and don't know what it is.

5.0.5 File: data.json

Our main databas name with all data from all the projects. Used be the API.

5.0.6 File: data.py

This is the API-side of the project. Gets imported by the flask file and is used on almost every HTTP request.

5.0.7 File: data test.py

This is a test file that is used to testa the data.py file so that it is ok on all the requirments made up be the univerisity.

5.0.8 File: myFlaskProject.py

This is the main flask file that is active running on the server and waiting for HTTP request to handle.

5.0.9 logfile.log

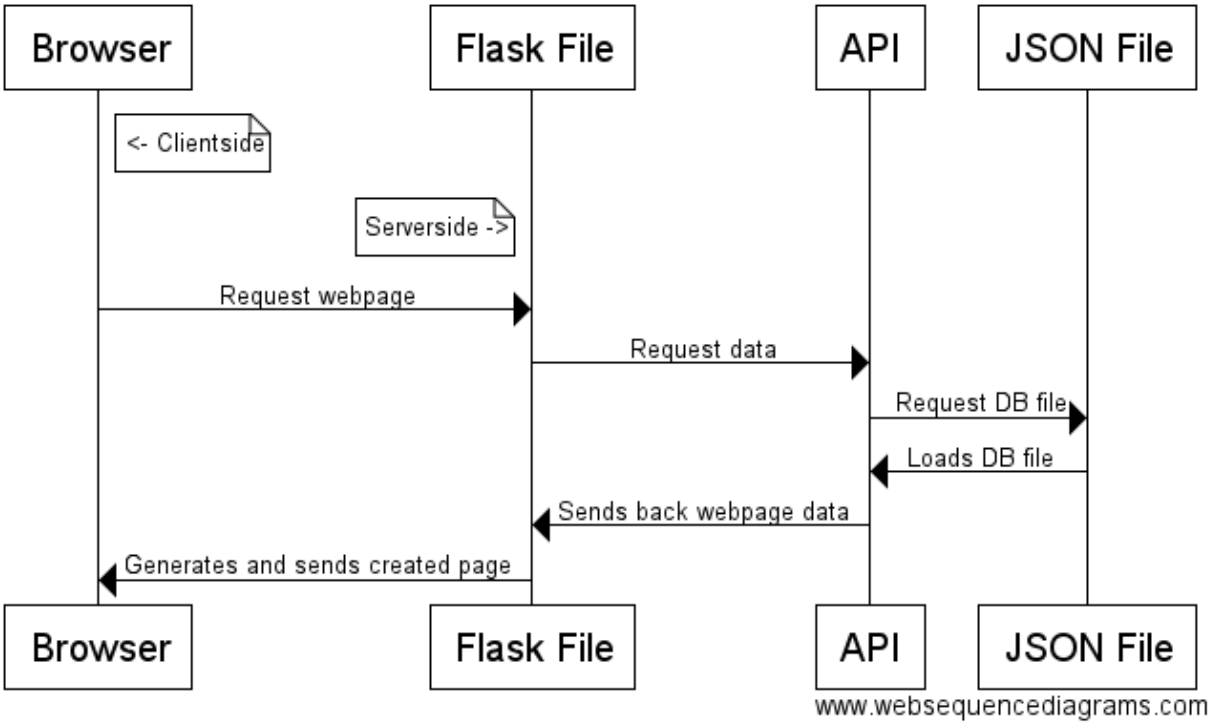
This is the log file that is created by flasks and contain alla logs and actions that flask have done.

6 Overview of the system

The system is a basic system for handeling some webpage requests and some data.

It's made up of a client side and a server side. The clientside is simply the users browser, with the functions of requesting a page using the GET method.

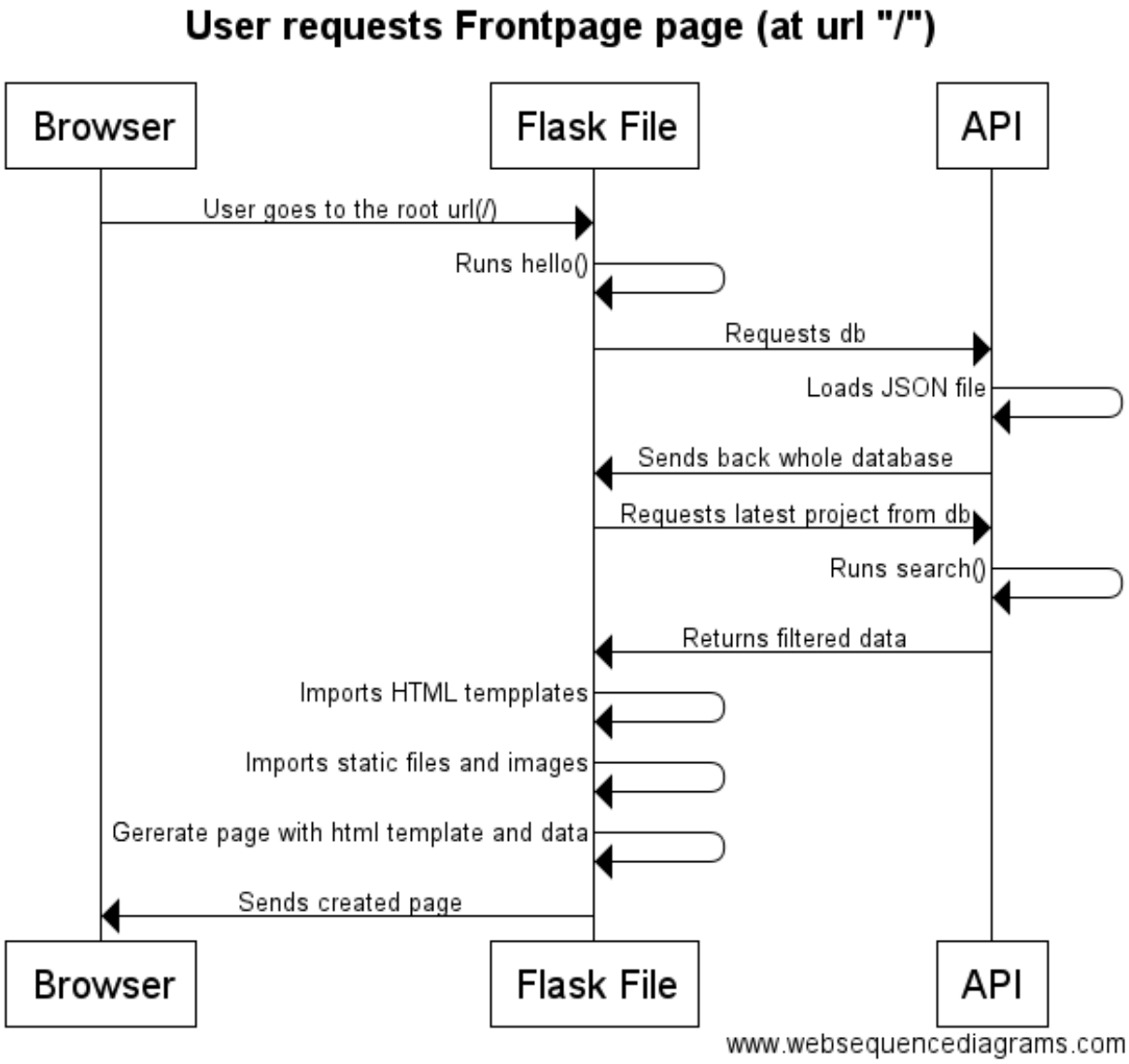
The server side is made up of two main processes, the python flask webserver file and the API file that handles alla the data. The API file uses a JSON file to store all the informationen.



6.1 Example Senario - Frontpage

When the user goes to the root of the webpage, the browser sends a request to the the server where flask is waiting and have a function bind to requests for the root directory. In this case, the hello() funktion is run. Hello() does two things, requests the database file from the API and then sends it back to the API layer with the request that it should search and filter the file. In this case we only the sends the database parameter to get all the latest projects with the newest projects first, but we could specifiy multiply parameters to tell the search function how it should filter the projects.

When the Flask layer get the filtered data, it imports the HTML templates, css files and other static files like images. Then it generates the HTML-files from the templates and the filtrered data, and finally sends it back to the users browser that displays the webpage to the user.



7 Description of the system components

7.1 Client Browser

Client Browser is stylede with CSS3 and built with HTML5 HTML files are stord in /templates and CSS files are stord in static/style

7.2 Flask Webserver

7.2.1 hello()

Handling requests for the index page. Returns a html page with some css. This function loads the all data thoguht the API and uses the search-function of the API to filter out the latest projects. Uses the ninja function render_template to combine the filtered project with the html-page

7.2.2 techniques()

Handling requests for the technologies page. Returns a html page with some css. This function loads the all data thoguht the API and uses the get_technique_stats-function of the API that returns all project stats for the tech page. Uses the ninja function render_template to combine the tech-data with the html-page

7.2.3 list()

Handling requests for the list page. Returns a html page with some css. Uses the methods GET. This function loads the all data thoguht the API and uses the get_techniques-function of the API that returns a list of all techniques for the checkboxes on the page. Then it pushes the data the API:s search-function. Also have some selfexplanitory code for autofilling the page on load. Uses the ninja function render_template to combine the data with the html-page

7.2.4 project(xid)

Handling requests for the project page. Returns a html page with some css. This function uses the id-number form the URL request. This function loads the all data thoguht the API and uses the get_project-function of the API that returns the required project. Uses the ninja function render_template to combine the data with the html-page Also has some error handling for is the types a unexpected id in the URL, the it renders the error page instead of the regular page.

7.2.5 error(e)

Handling error requests. Uses the ninja function render_template to combine the data with the html-page

7.3 Datalager API

7.3.1 load(file_json)

Loads the specified file and returns it if found else returns None [1]:2016

7.3.2 get_project_count(db)

Takes in the database and returns amout of project in the database

7.3.3 get_project(db, id)

Returns the project with the selected ID.

[1] `http://www.ida.liu.se/~TDP003/current/portfolio-api_python3/`

7.3.4 search(db, sort_by='start$_d$ate', sort_order =' desc', techniques = None, search = None, search_fields = None)

Handeling a search request. Creates a empty templist. Filters out all projects that has the right seach text in the right searchfields. Filters out all projects with the right techniquies. Then comes some code for selecting order. Sortes the projects left. Returns the projects that passed thought the filters.

7.3.5 get_techniques(db)

Returns a sorted list of all tecniques

7.3.6 get_technique_stats(db)

Returns a custom list with techniques and its projects.

7.3.7 search_in_search(projects_left, search_fields, search, item, project)

Checks if the search text is in the selected fields. Used the search_search function for help.

7.3.8 search_search(value, search)

Returns true of false if a specific spot in the search text matches any text in the search field.

7.3.9 search_search_fields(db, projects_left, search_fields, search)

Checks if the searchs something in the search fields. Used the search_in_search function.

7.3.10 search_techniques(projects_left, tech)

Filters out the project that uses a specific techquies

8 Others

8.1 Logging

Flask is setup in a way that it should log every action that happends in a logfile in the root directory named logfile.log.

8.2 Error handling

The website redirects the user to the error page incase they tries to go to a url that is not valid.

If you want to enable debug mode in flask, simply add debug=true as a parameter in the app.run(). You will then be able to see every action in the terminal and you will also be able to edit the files and see the change on the page without restarting the server.

8.3 Tools and programs

To compile all the documents in the doc folder, you have to use a latex compiler to export a PDF from the code. On linux, we recomend using the command latexmk -pvc -xelatex and then the filename of the .tex file of the wanted document.