Portfolio

Åsmund Ødegård
@mandus
`https://github.com/mandus`

Most of what I do end up as server-side code not suitable for inclusion in a visual portfolio. Also, I shall not claim that front-end design is my speciality. Nevertheless, in several projects and in particular those where the staffing is limited, it is required to work as a full-stack developer—all they way from the database and server-side logic to the user experience.

The purpose of this document is thus to showcase some projects where I have been involved in front-end work (and where I legally am allowed to use images for this purpose). Only rarely will my work be on publicly available pages, so it is easier to showcase the sites in a document rather than pointing to the web-pages.

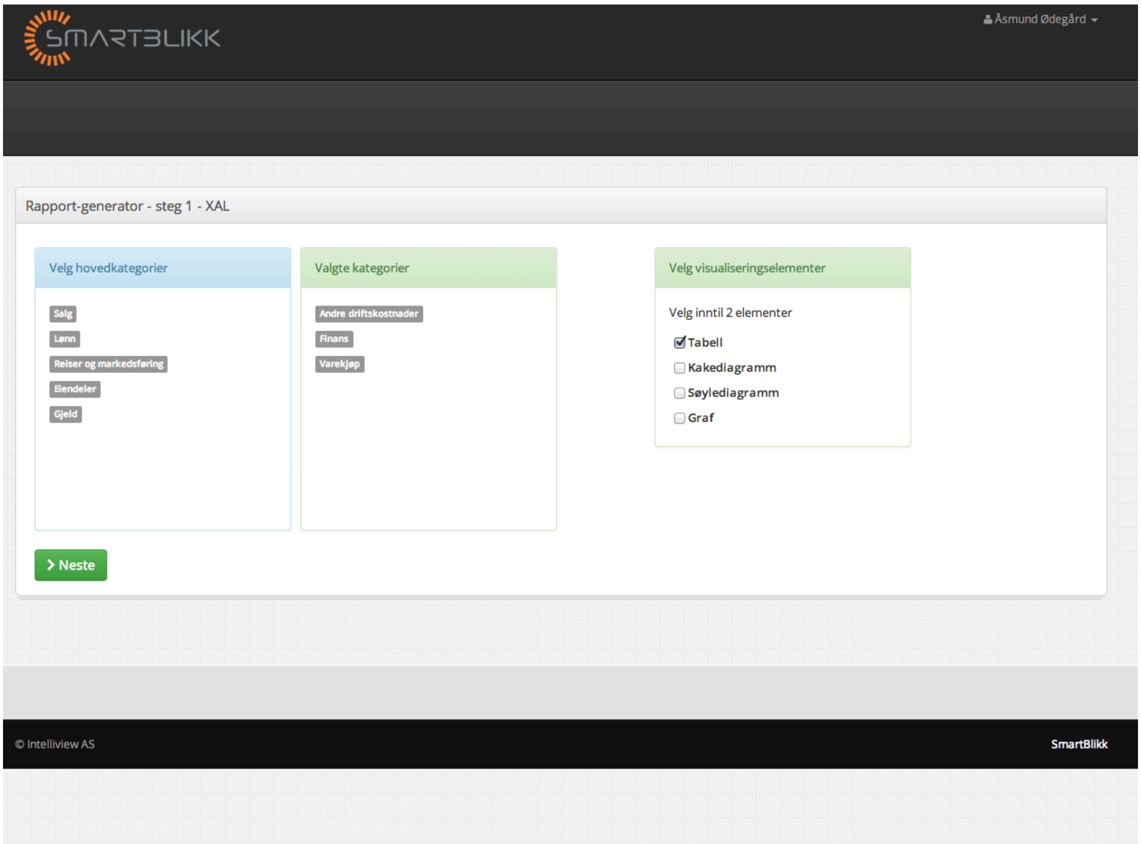
```
class updateValue:
    @utils.queuewrapmeth
    def POST(self):
        if not logged(session, dbmysql):
            retu
        i = web.
        web.head
        if debug
            prin
        if i.Chu
            Chur
        else:
            Chur
```

```
var attempt_login = function attempt_login(){
    var username = $('#login-username').val();
    var password = $('#login-password').val();

    var params = decode_params(window.location.search.substring(1));
    var refer = params['refer'];
    if (!refer) {
        refer = '/index';
    }

    $.couch.login({
        name: username,
        password: password,
        success: function(data) {
            setTimeout(function() { window.location = refer; }, 10);
        },
        error: function(status) {
            show_error("Unable to login user", status);
        }
    });
}

var attempt_logout = function attempt_logout() {
    $.couch.logout({
        success: function(data) {
            console.log("Logged out");
```

SmartBlikk

SmartBlikk is implemented in PHP using the Slim micro-framework and Twig template engine. On the front-end we utilise the Bootstrap and jQuery frameworks together with custom Javascript code. At the stage of these images the product is still unfinished, but I show a few images from the wizard for building custom reports, and a view of administrative settings for APIs.

Some of the surrounding graphics in these images (mainly the header/footer in the first image) were already set up when I started working on the project. However, the main content on these pages, with the accompanying back-end functionality, has been implemented by me.



Rapport-generator - steg 1 - XAL

Velg hovedkategorier

- Salg
- Lønn
- Reiser og markedsføring
- Eiendeler
- Gjeld

Valgte kategorier

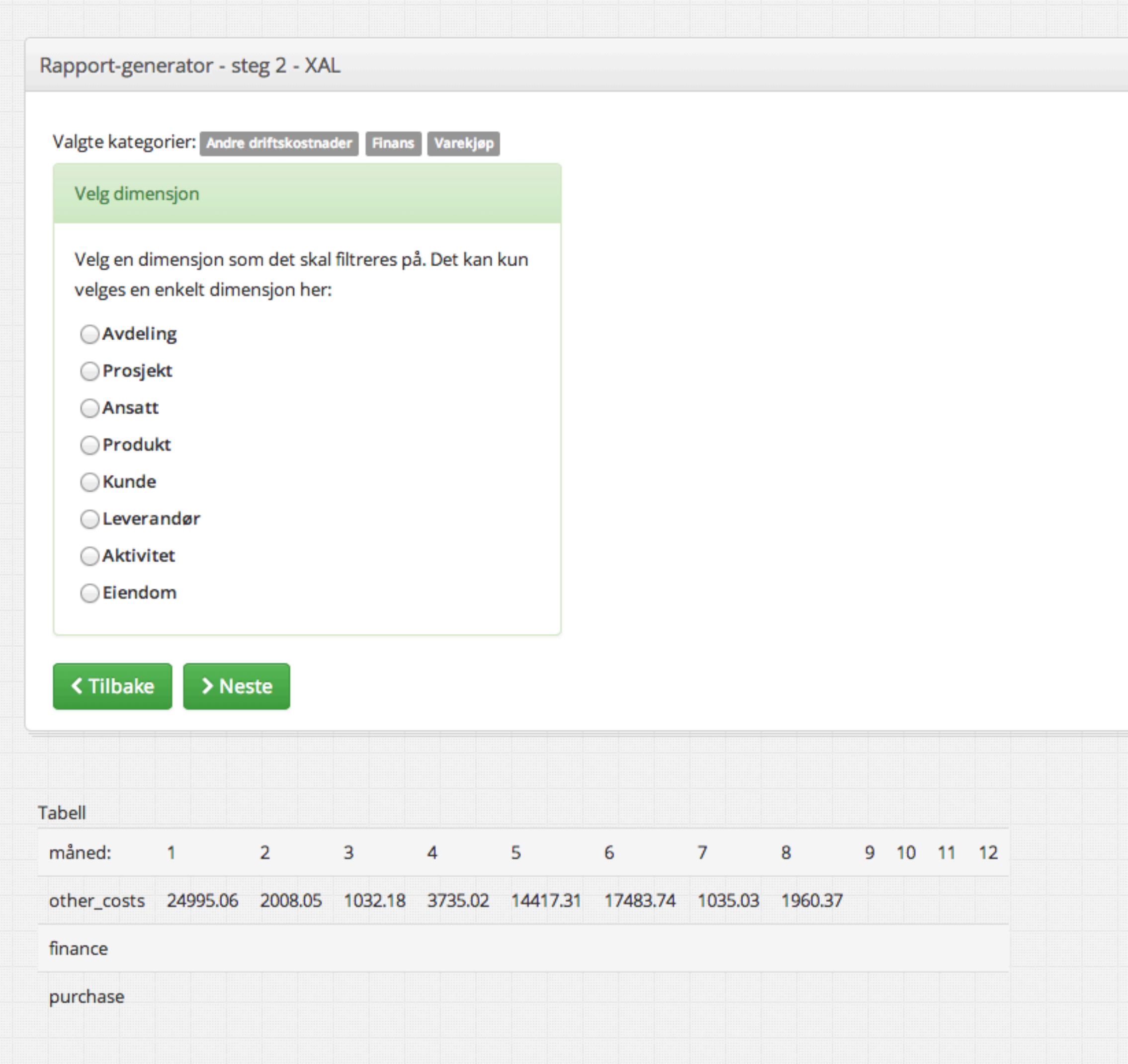
- Andre driftskostnader
- Finans
- Varekjøp

Velg visualiseringselementer

Velg inntil 2 elementer

☑ Tabell
□ Kakediagramm
□ Søylediagramm
□ Graf

The user can move categories between the boxes on the left side in order to enable or disable main categories in the report. In addition a few visuals to be used in the report can be selected already at the first stage. These visuals will then be live updated in subsequent steps as further choices is done for the report.



Rapport-generator - steg 2 - XAL

Valgte kategorier: Andre driftskostnader Finans Varekjøp

Velg dimensjon

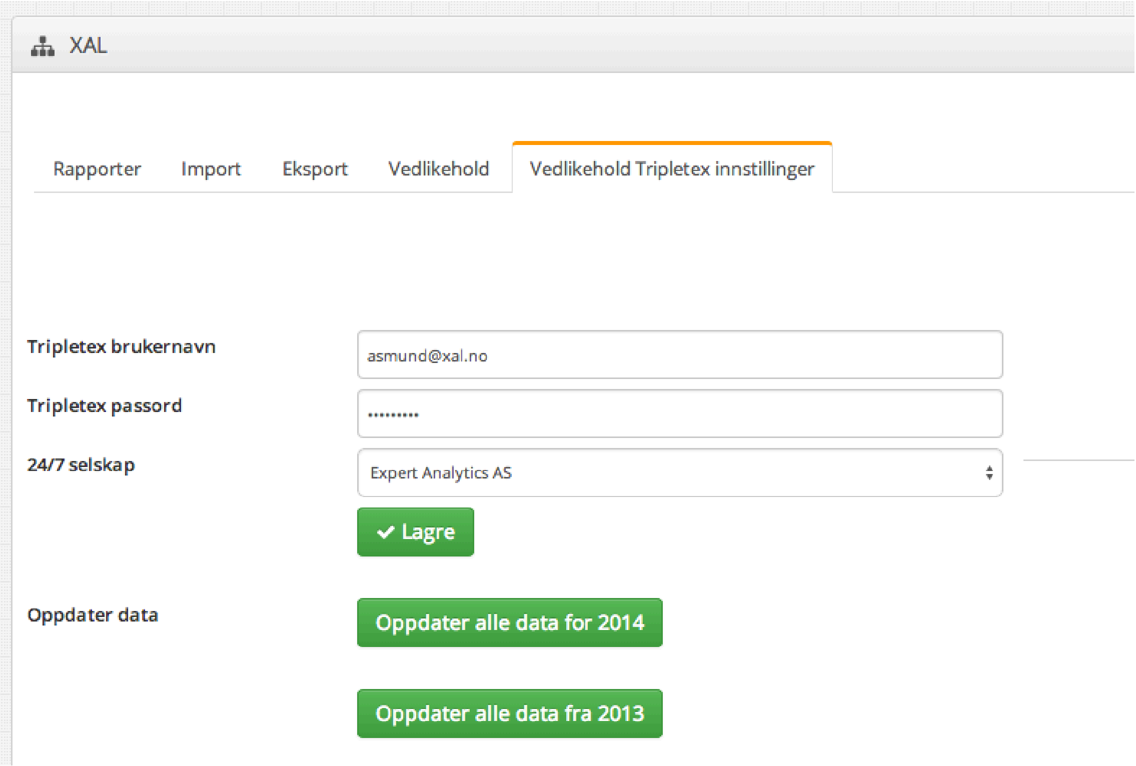
Velg en dimensjon som det skal filtreres på. Det kan kun velges en enkelt dimensjon her:

○ Avdeling
○ Prosjekt
○ Ansatt
○ Produkt
○ Kunde
○ Leverandør
○ Aktivitet
○ Eiendom

Tabell

måned:	1	2	3	4	5	6	7	8	9	10	11	12
other_costs	24995.06	2008.05	1032.18	3735.02	14417.31	17483.74	1035.03	1960.37				
finance												
purchase												

On the next stage in the wizard, the used can further customise the report by selecting dimensions. Ajax is used to fill the selected visual (in this case table) such that the user-interface remains responsive, and the data is updated as choices is done in the list of radio-buttons.


XAL
Rapporter
Import
Eksport
Vedlikehold
Vedlikehold Tripletex innstillinger
✓ Lagre
Oppdater alle data for 2014
Oppdater alle data fra 2013

Tripletex brukernavn: asmund@xal.no

Tripletex passord: •••••••••

24/7 selskap: Expert Analytics AS

Oppdater data

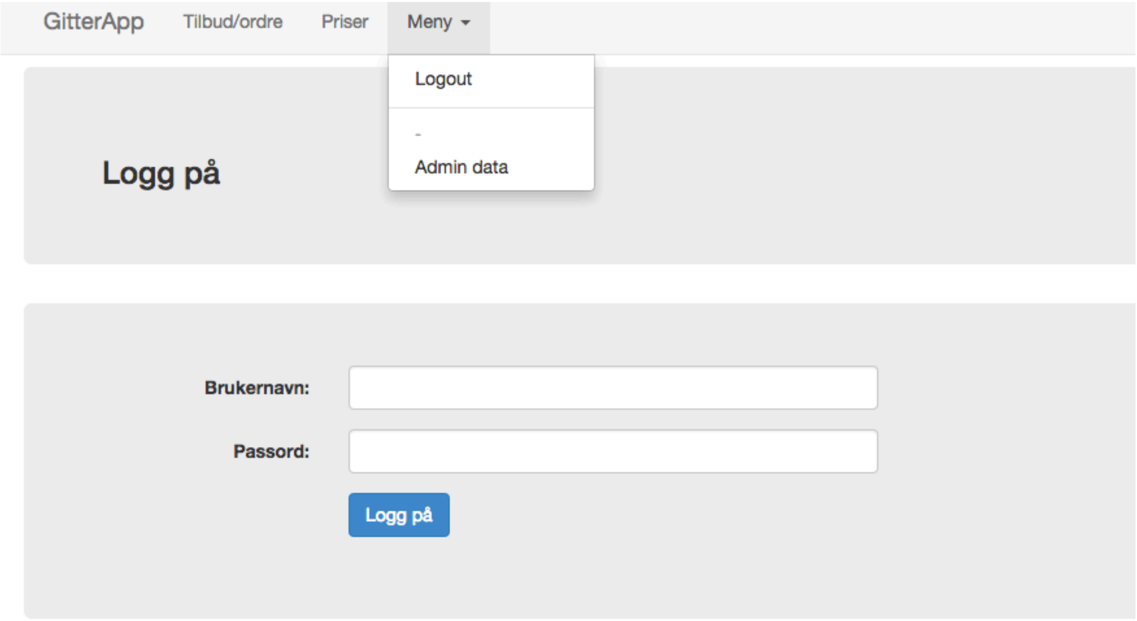
On this screen, the user will be able to maintain settings for one of the implemented API integrations. The buttons at the bottom will trigger calls to the backend server that rebuilds the internal dataset. Again we are using ajax-calls for this such that the user interface remains responsive. In case the user stays on this page, a message will appear when the task is finished, mentioning some statistics for the imported data. As a side-note, the user credentials dialog work together with the security functions on the backend. We do not want to store password in cleartext in the database, so the cleartext password will only be available in memory. As SmartBlikk needs to know the password when making calls to the API on the user's behalf, we utilise a symmetric AES algorithm for the encryption.

GitterApp

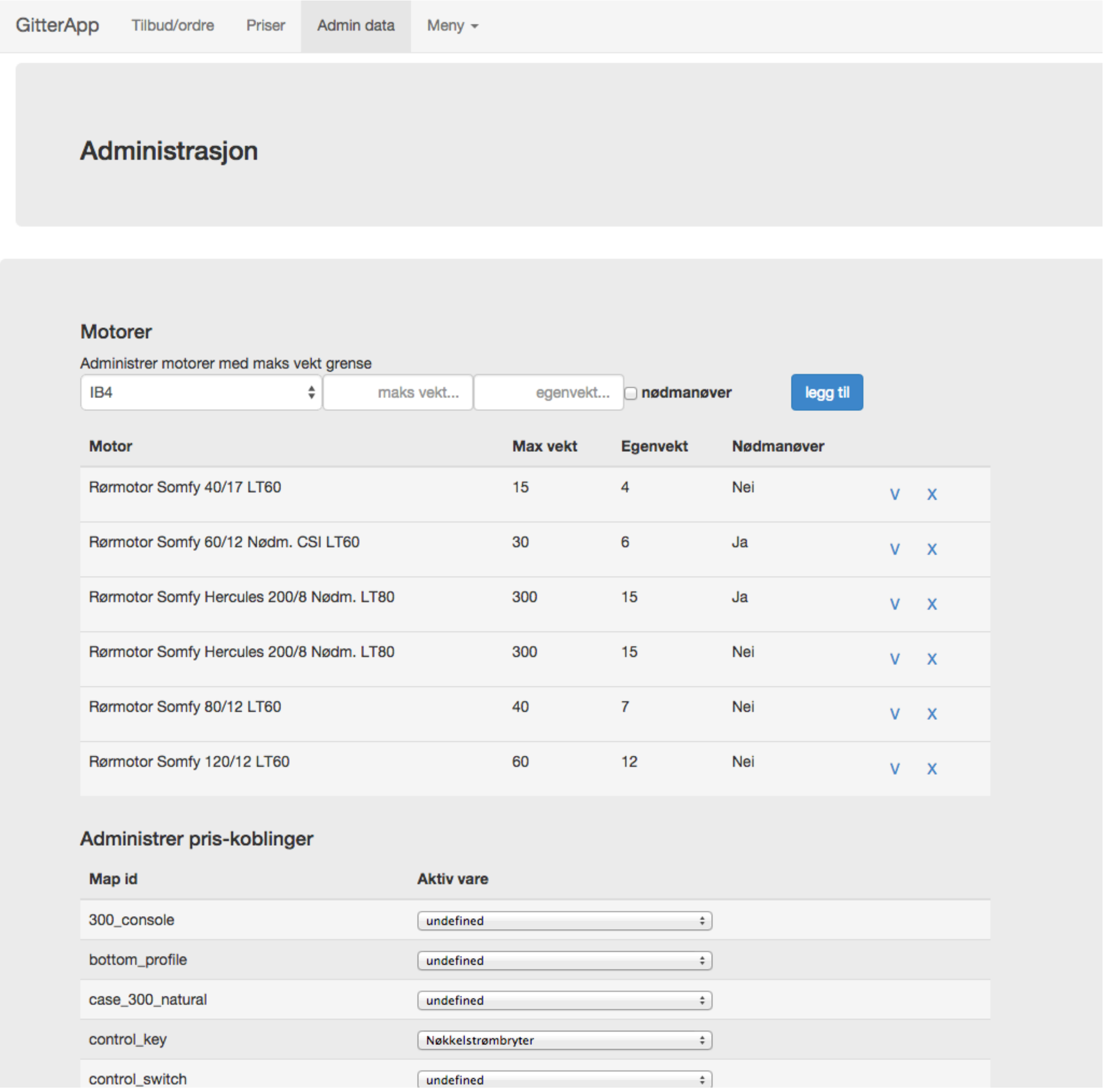
For a Norwegian SME producing physical security items for the business sector, we have created a web-based system for configuration of security grilles. The application is implemented using CouchDB as back-end and HTML5 front-end utilising jQuery and custom Javascript code. The visual appearance is based on the Bootstrap framework. This application is somewhat unique in that there is no application server involved except the database server; the sources are served directly from the CouchDB server and all processing are done on the front-end. The system is running in the "cloud," on an Amazon EC2 instance.

I display three pages from this system; the log-on screen presented when you enter the site, and excerpts from two different administrative pages. I am not able to showcase the main part of the system where our customer configure the grilles, but the main features of that system is live calculation of the final price and validation of completeness; i.e. there are certain features that must be selected in order to ensure a complete security grille. There are about 20 different settings, many of them with several different options, which yields a fairly complex system.

GitterApp was fully designed and implemented by Expert Analytics. I have implemented everything on display here, while Ola Skavhaug implemented some other parts of the system.



The system is implemented in a lean and simplistic style, featuring only three main pages, and with some administrative features only accessible by super-users.



Administrasjon

Motorer

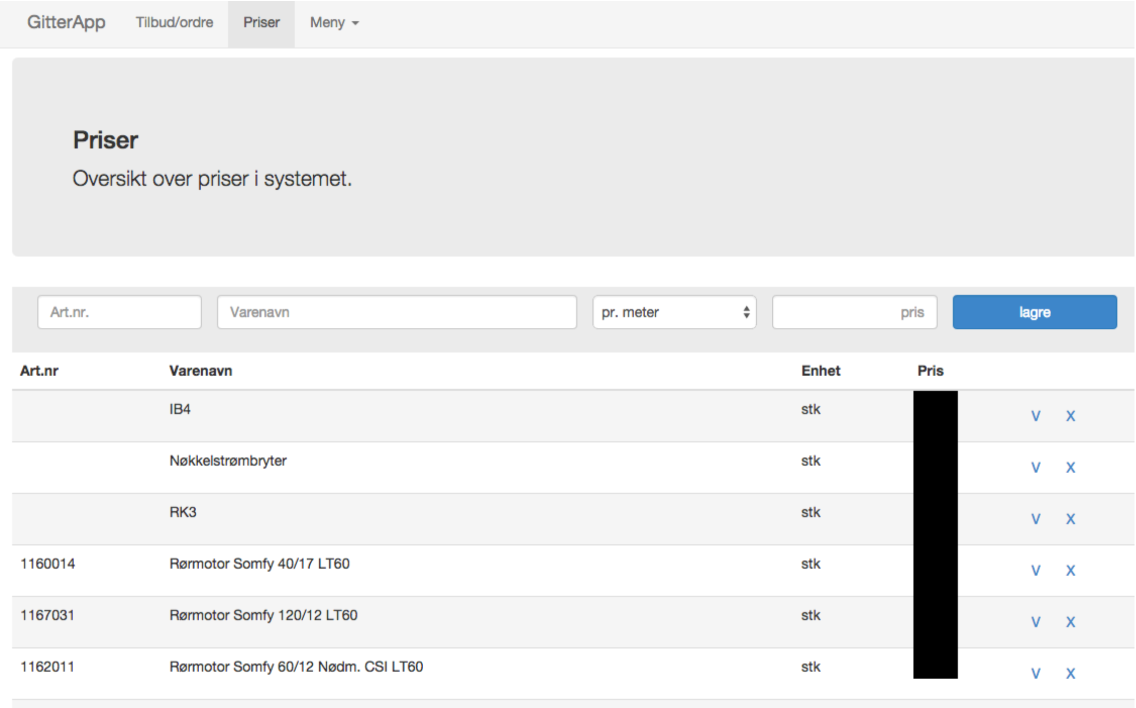
Administrer motorer med maks vekt grense

IB4 | maks vekt... | egenvekt... | □ nødmanøver | legg til

Motor	Max vekt	Egenvekt	Nødmanøver		
Rørmotor Somfy 40/17 LT60	15	4	Nei	V	X
Rørmotor Somfy 60/12 Nødm. CSI LT60	30	6	Ja	V	X
Rørmotor Somfy Hercules 200/8 Nødm. LT80	300	15	Ja	V	X
Rørmotor Somfy Hercules 200/8 Nødm. LT80	300	15	Nei	V	X
Rørmotor Somfy 80/12 LT60	40	7	Nei	V	X
Rørmotor Somfy 120/12 LT60	60	12	Nei	V	X

Administrer pris-koblinger

Map id	Aktiv vare
300_console	undefined
bottom_profile	undefined
case_300_natural	undefined
control_key	Nøkkelstrømbryter
control_switch	undefined



Priser

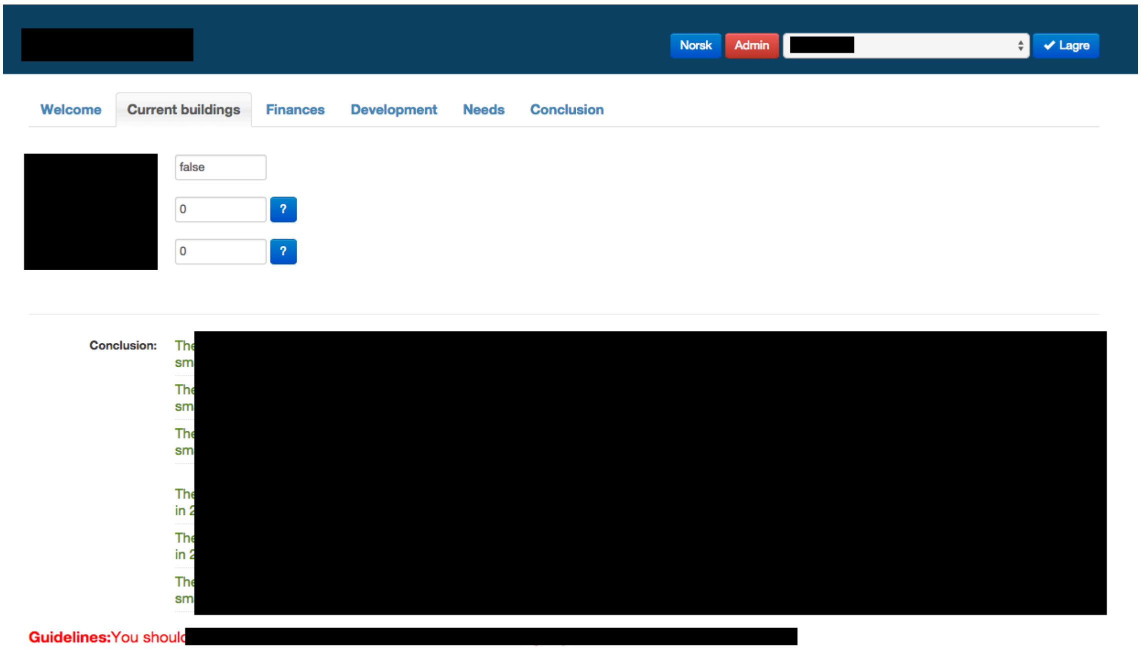
Oversikt over priser i systemet.

Art.nr	Varenavn	Enhet	Pris
	IB4	stk	[redacted]
	Nøkkelstrømbryter	stk	[redacted]
	RK3	stk	[redacted]
1160014	Rørmotor Somfy 40/17 LT60	stk	[redacted]
1167031	Rørmotor Somfy 120/12 LT60	stk	[redacted]
1162011	Rørmotor Somfy 60/12 Nødm. CSI LT60	stk	[redacted]

Web.py Project

Finally, I will show some images from a project implemented using the Python Web.py framework and a WSGI application server. Due to the nature of this project, I must edit out the details. The system was implemented for an volunteer-based non-profit international organisation. As for the other projects, Bootstrap, jQuery and custom Javascript is used on the front-end. There are a lot of calculations going on in the front-end when users utilise this tool for their analysis, with ajax-based interaction with the python application server when necessary. The Highchart Javascript library is used to generate live dynamic graphs in the browser.

I was the sole programmer on this project, but worked together with a designer/UX-specialist who organised the information into tabs and worked on placement of the elements. This was crucial for the project since it was to be used by several persons with quite different prior competences. We both need to implement a natural path through the wizard, as well as providing some flexibility for more advanced users.



[redacted]

[redacted]

[redacted]

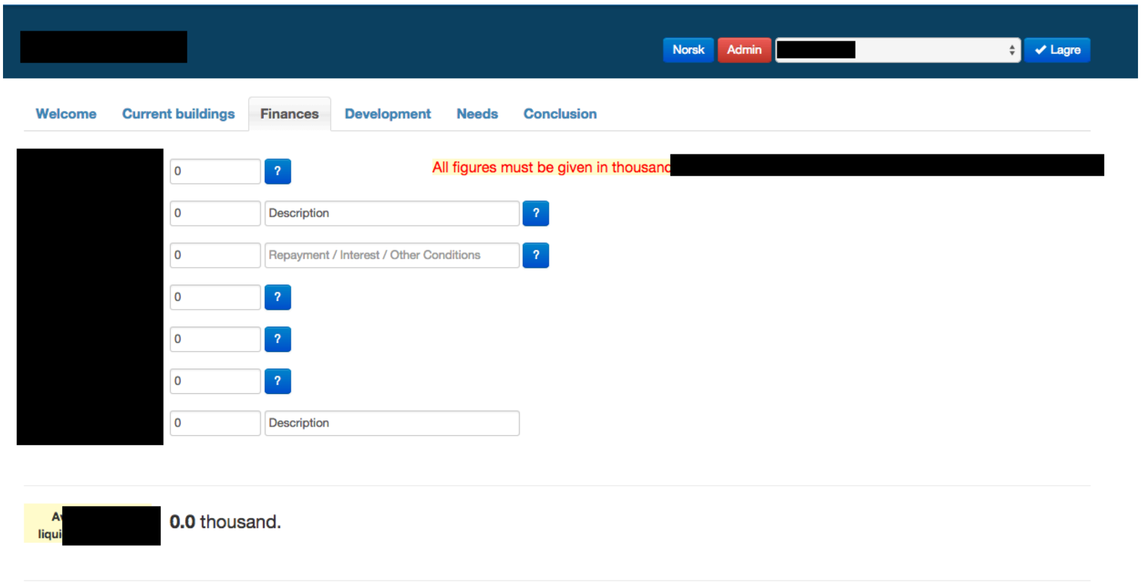
false

0 ?

0 ?

Conclusion: The sm The sm The sm The in The in The sm [redacted]

Guidelines: You shoul [redacted]



[redacted]

[redacted]

[redacted]

0 ?

All figures must be given in thousand [redacted]

0 Description ?

0 Repayment / Interest / Other Conditions ?

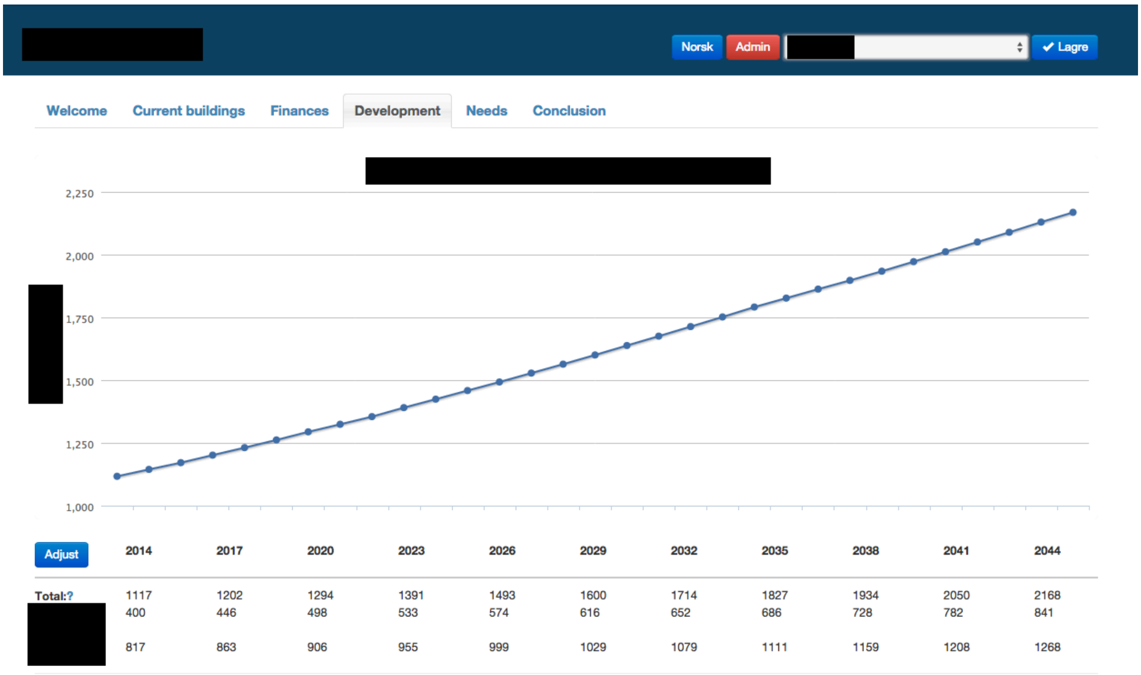
0 ?

0 ?

0 ?

0 Description

A liqui [redacted] **0.0** thousand.



Adjust	2014	2017	2020	2023	2026	2029	2032	2035	2038	2041	2044
Total:?	1117	1202	1294	1391	1493	1600	1714	1827	1934	2050	2168
[redacted]	400	446	498	533	574	616	652	686	728	782	841
[redacted]	817	863	906	955	999	1029	1079	1111	1159	1208	1268

This is actually the most important and advanced part of this system. Based on some data made available by the organisation, algorithms are used to compute a forecast graph for the next 30 years, as well as key accompanying numbers included in the table below. Using the "Adjust" button, the user may modify some parameters for the algorithm which will trigger recalculation of the forecast. When the updated forecast is finished, it will be loaded back in the graph. There is a lot of stuff going on both in Javascript on the client side and on the back-end application server, which is all hidden for the user. The user may even visit some of the other sub-pages while the calculations are carried out, if they want.

Conclusion

Hopefully these images provide some ideas on what Expert Analytics will be able to deliver in terms of front-end work, given that this is not our main area of expertise. Unfortunately I do not have a lot of recent code publicly available, but the PySE code that was an important part of my PhD work is available at `http://sourceforge.net/projects/pyfdm/`. Recently this code was cloned by another researcher and made available at `https://bitbucket.org/sinayoko/pyse`; the original code may be easier to browser there but note that the researcher who uploaded the code has made a few updates. When looking at PySE as a reference, please keep in mind that this code about ten years old now, so both my coding-style and the community at large have evolved quite a bit since then.