Who: **Dan Christensen**

What: **Internship Report**

Where: **Danmarks Radio, TU Web Applikationer**

When: **8/8/2016 - 14/10/2016**

School: **Erhvervsakademi Sjælland, Datamatiker**

Supervisor: **Anders Børjesson**

Index

This report is also available at:

https://drive.google.com/open?id=1ZjHCtGH7GRLfT1_b_l73WGJZCz-4hERWzMttRtBhzDU

The google drive version is mobile friendly.

Introduction

I had my internship at the Web Applications department at Danmarks Radio. I was put in contact with them through one of my teachers who provided me with the email address of my contact person at DR. They were interested in having me, so I went to two meetings there to work out the details.

They had previously only had one intern there, a student who was doing his internship on his top-up bachelor, so he was 3 semesters ahead of me. We talked a bit about what that meant, and what they could expect from me, but fortunately they were still interested in having me there, so we went ahead with it.

I would be assigned to a development team, and would act as a full member of that team, participating in and working on the same projects as they were. I was initially assigned to Team Connect, a front-end development team. Due to a lack of things to do early in the internship, I was briefly moved to Team Drupal Core. This only lasted 4 days though, as Team Connect then got several large assignments that they wanted me to work on, and I was moved back again.

The Company

Danmarks Radio is a huge company with many different departments. Saying “I interned at DR” is about as precise as saying “I live in Scandinavia”. I need to get a bit more specific, so I will try and explain where in the organization I was placed, and what that department actually does.

DR in general

DR, or “Danmarks Radio” has existed as a media company since the 1st of April, 1925. They employ close to 3000 people across the country. DR spans 6 tv channels, 9 radio stations, various mobile apps and their website, dr.dk.

The company is publicly funded through a media license that all Danish households have to pay, with a budget set by the Danish government. As seen by the absolutely massive organizational chart, which you can find here[1], DR is divided into many different departments. This creates some interesting challenges in regards to workflow, but more on that later.

The Web Applications Department

The department I was assigned to, “TU Web Applikationer”, is part of the larger “DR Teknologi”. They are responsible for developing the dr.dk website and livestreaming, as well as various mobile applications. There are currently 50-60 employees in the department across several development teams ranging from 4-8 members. The teams are divided into general areas of expertise: Front-end, back-end, streaming and so on.

Team Connect

I was assigned to Team Connect, which is primarily a front-end team. They consist of 3 front-end developers, 2 .NET back-end developers and a Product Owner. They deal mainly with the functionality part of front-end applications, where the actual visual design is usually dictated by the design and UX teams. Other teams will have different compositions, depending on what area of expertise they are assigned.

Work and Workflow

Before I get into the assignments that I was actually working on, I feel it’s important to talk about how they do things at DR in regards to their web applications, as it’s a good deal different from what we have learned during our time at school.

Customers

The role of the customer at DR is actually other departments. Since everything is publicly funded, each department gets an allotment of these funds. If another department wants something made, then they have to make room for that in their budget, and “pay” the Web Applications department for their time. In most cases, this means paying for the salaries of the team, as well as any extra costs that might occur.

[1] DR organizational chart:
https://prod-public-files-cms-dr-dk.s3.amazonaws.com/static/documents/2016/09/22/organisation_20160922_543c88d4.pdf

The shaping process

“Shaping” is what they at DR call the initial period of a project. It’s basically the “requirements” phase of the Waterfall model, where the people doing the shaping define what needs to be delivered, and what the scope of the project is. Sometimes, it’s also during this phase that initial designs are drawn up and sent to the design team.

Most of the shaping process is often entirely disconnected from the development team. From what I’ve experienced, the development team isn’t brought on board until Shaping is mostly done, and the project is presented to the team during the sprint planning phase, where a rough count of hours is estimated, so a budget can be drawn up. This is also where they apply Scrum principles to the extent that it’s possible.

ScrumBut

The development teams at the Web Applications department are organized into Scrum-like teams. In the Scrum framework, this variant is called ScrumBut[2]. Since the team usually doesn’t get involved until after the shaping and visual design / UX process is done (or well underway), they don’t really get to take part in that, and are instead left with a waterfall-type assignment with set requirements and design.

Despite this, the team is expected to be agile, and to be able to adjust to any changes that might come up - be it changes in requirements/design from outside, or limitations/difficulties discovered during the development process. They also perform a lot of the “rituals” associated with Scrum, such as the ones explained below. The teams being organized into Scrum teams is still a fairly new thing - approximately 6 months old by now - everyone is still adjusting to the methodology, and trying to find ways that work best for them.

Kanban Board

The Kanban Board is the center of the Scrum methodology, and is is in Team Connect as well. During my internship, this was a physical whiteboard with post-it notes of various colors stuck to it. However, there was discussion about maybe trying out other alternatives, such as Jira[3] or Pivotal[4].

The physical kanban board tends to only get updated during the Daily Scrum meetings. Also, when team members are working from home, they are unable to access and update the board. Moving to a digital implementation would solve these problems - but would also make the Daily Scrum more bothersome, as the team would have to huddle around a monitor instead of a big whiteboard.

Team Connect’s Kanban Board

[2] Scrumbut: https://www.scrum.org/scrumbut
[3] Jira: https://www.atlassian.com/software/jira
[4] Pivotal: http://www.pivotaltracker.com/

Daily Scrum

The team does a Daily Scrum meeting around the Kanban Board every day. Meeting times are flexible in Team Connect, but everyone has to be there for the Daily Scrum meeting, which starts at 9:30. Here, we discuss what we have done the day before, what we plan on doing today, and if there are any difficulties, or if we need help with anything.

The Daily Scrum is mostly relevant when the team is working on an assignment together, but it's still good to get an overview of what everyone is doing. Due to the nature of the assignments that the team had during my internship, we were almost always split between 2-3 different assignments. As a result of this, the Daily Scrum was sometimes cancelled, simply because it didn't make sense do it.

Sprint planning

At the start of every sprint, there is a Sprint Planning meeting, where the team lays out what they're going to do for the following sprint. User stories are broken down into tasks, and the team assesses how much they think they'll be able to manage during the upcoming sprint. Each sprint is 2 weeks, or 10 working days.

One thing to note here is the way Team Connect have chosen to timebox their assignments. For user stories, each "point" is equivalent to one day of work for one person - So a 10-point assignment is 10 work days worth of work for one person. For tasks, the "point" unit is roughly an hour of work, with 6 points being assumed as a full work day.

Sprint refinement

The refinement phase happens in the middle of the sprint. It works in much the same way as sprint planning, but is a re-evaluation of the tasks they are working on. The team adjusts velocity, and adds any rush jobs that might have come up to the backlog.

Sprint retrospective

The review & retrospective is a standard part of Scrum, and happens at the end of each sprint. Due to how things are organized at DR, the "review" part doesn't happen during this time, and instead the team focuses on the retrospective part. All in all, it's a fairly close-to-methodology approach - What went well, what could have gone better, etc. - but with some limitations due to how things are organized.

Friday Meetings

Every friday at 9:00, there is a special meeting in addition to the Daily Scrum. 3 times a month, this is a meeting for the entire Web Applications department, where everyone eats breakfast together, and the 3 department managers talk about news across the department. Sometimes, the friday meetings include demos of applications that other teams have worked on, and sometimes we get external speakers showcasing products or solutions that they are involved with.

Friday meeting in progress, with a presentation going on

General Meetings

Once a month, there is a general meeting for everyone in “DR Teknologi”, where the higher-up managers talk about things relevant to the department. I participated in two such meetings, and they were mainly scaled-up versions of the normal friday meetings, but with additional things such as budget talks and the like.

DR Teknologi Meeting in one of the bigger rooms at DR

Tools

Team Connect uses a wide range of tools to create their applications and projects. The programming languages and frameworks that they use vary from project to project, so I’m not going to cover them here. However, there are a couple of tools that I think it’s important to mention, because they are some that we have no, or very little experience with from our lessons at school.

Github

Github[5] is an important part of the development process at the Web Applications department. Previously, they had been running with a self-hosted Github Enterprise[6] solution, but are in the process of moving everything to a cloud-based github.com solution instead. They have integrated Github into the larger development pipeline.

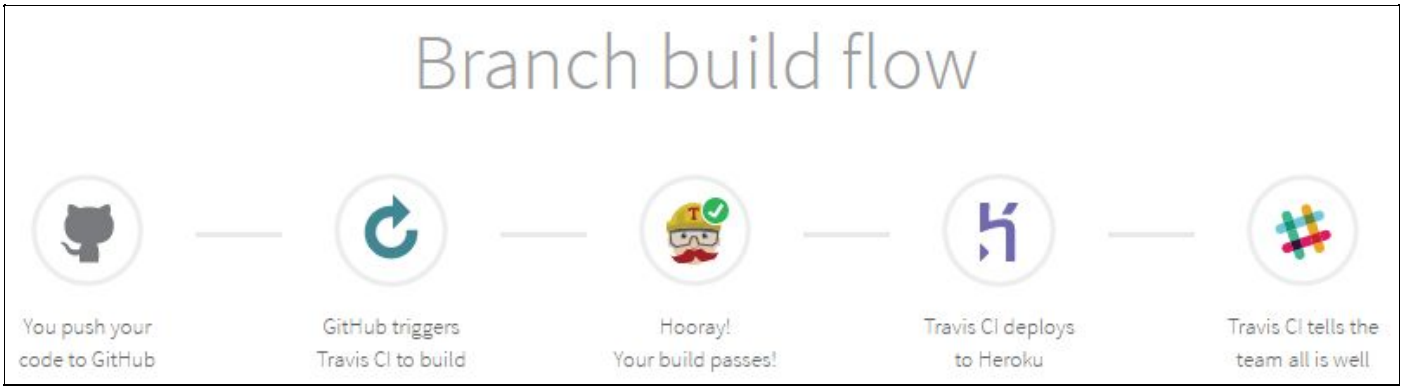
Heroku

DR used to use Azure[7] as their cloud hosting of choice, but are starting to move over to Heroku[8] instead. Heroku is the final two steps of the development pipeline. Once a project on Github is ready to go live, it is pushed to Heroku, to a “staging” build, so that the application can run in a live environment. It then goes through final approval and is pushed to the “live” build.

Automated Testing

As part of the development pipeline, Team Connect have started to implement automated testing, along with a continuous integration framework. Apart from unit tests, they also use Selenium Webdriver[9] to run automated browser tests. This means that they can automate most aspects of testing, and along with the Travis CI[10] framework, they can ensure that things are tested every time new code is pushed to GitHub.

[5] Github: https://github.com/
[6] Github Enterprise: https://enterprise.github.com/home
[7] Azure: https://azure.microsoft.com/en-us/
[8] Heroku: https://www.heroku.com/
[9] Selenium: http://www.seleniumhq.org/projects/webdriver/
[10] Travis CI: https://travis-ci.org/



Example build flow with Travis CI

Slack

Finally, pretty much everyone at the Web Applications department uses Slack[11], an instant messenger service aimed towards professional teams. It has integration with a lot of popular tools (such as Travis CI). Some teams use Slack more than others. Team Connect doesn't use it all that much, but Team Drupal Core use it for pretty much all their communication.

Work assignments

The idea for the internship was that I would be placed on a team, and then counted as a full member of said team. Unfortunately, when I started, Team Connect didn't have any major assignments going on - mostly maintenance and smaller tasks - so I couldn't pair up with a particular person and work, but rather, I was given a bunch of minor assignments to work out mostly on my own.

However, around week 5, the team managed to get two major assignments: Making things for the US election coming up on November 8th, and working more long-term on a generic candidate database for danish elections. Alongside the latter assignment, the team were also assigned to come up with a candidate test (based on data from the database), and the associated applications for this.

It doesn't make much sense to list my work on a week-by-week basis, as I would often jump back and forth between assignments. Instead, I have chosen to describe my work on a by-assignment basis.

For a day-to-day breakdown of my work, please refer to the work logs in the appendix.

Automated Testing / Unit testing

DR's website has a feature called "Tæt på" that they use to make Q&A sessions with various people of interest. The application can be run as text or video based, where users of the site can ask questions, and the VIP will then answer these questions. Once the session is over, everything is saved, so that it can be accessed later on. An example of a "Tæt på" session that has ended can be seen here[12].

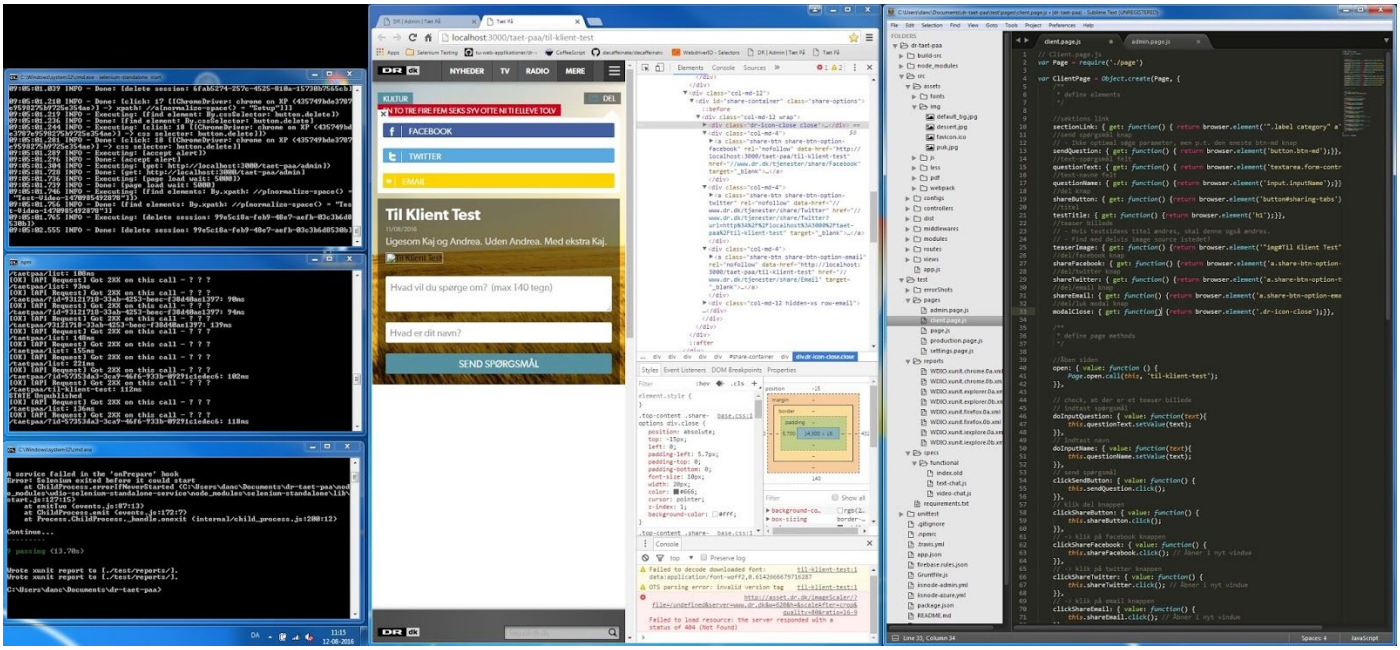
The task

I was assigned to work on writing both browser tests and unit tests for the application. There were some written already, but not nearly enough.

The tools

This involved learning Selenium Web Driver[13] (for the browser testing automation), mocha[14] (for testing), chai[15] (for assert libraries), and Travis CI[16] (for continuous integration).

[11] Slack: https://slack.com/
[12] Tæt På example: http://www.dr.dk/taet-paa/taet-paa-astrofysiker-tina-ibsen-faa-svar-paa-det-
[13] Selenium: http://docs.seleniumhq.org/
[14] Mocha: https://mochajs.org/
[15] Chai: http://chaijs.com/
[16] Travis CI: https://travis-ci.org/



Test writing in progress!

The result

The testing assignment was tough, because it was my first assignment there, and because it required me to learn a bunch of new frameworks all at once. However, I did end up writing a number of tests, and getting good code coverage for testing purposes.

Remove coffeescript from application

There is a small web application called “Attention Overlay”. It places an overlay on top of every page on the dr.dk domain (which consists of many smaller sites), containing text and a link to whatever they want to draw attention to.

The task

My assignment for this part was fairly simple: Parts of the application had been written in coffeescript[17]. The person who wrote the application no longer worked for DR, and the use of coffeescript has fallen out of favor. So I had to convert the coffeescript parts into plain javascript.

The tools

In order to do this, I had to familiarize myself with coffeescript first, as well as grunt[18], which is the task runner they use to compile the site. All in all, a fairly small assignment.

The result

Once I got a handle on how coffeescript and grunt worked, it was an easy assignment. Because I was again faced with entirely new things, it took time to get familiar with everything, but once I was, it was a fairly small matter to remove the coffeescript.

A/B testing research

As part of their ongoing digital strategy, it has been decided that DR in general, and of course the Web Applications department in particular, will start using A/B testing[19] to optimize and test their site. The principle of AB testing is to provide two different versions of the same site, and to then split your incoming traffic between the two sites. You set specific parameters to test, and then analyze the metrics from the two different sites to determine which version is better.

The task

I was assigned to do research on the different AB testing frameworks available, and take some notes on them, so we could decide which would be better for our team.

[17] Coffeescript: http://coffeescript.org/
[18] Grunt: http://gruntjs.com/
[19] A/B Testing principles: https://en.wikipedia.org/wiki/A/B_testing

The tools

Internet research, and nothing else.

The result

As an assignment, this was pretty straight-forward. Since it was a documentation/research assignment, I didn't run into any problems with this one. I also ended up providing the document to the Product Owner of another team, who had also been assigned to look at AB solutions (for a different purpose), and he found this very helpful[20].

Drupal Development Environment Setup

I was only with team Drupal Core for less than a week, and they were unfortunately incredibly busy when I was there, so I wasn't able to get much help with setting up their development environment.

The task

Set up the required programs for the development environment. Drupal 7, Virtual boxes, etc.

The tools

DR uses a heavily modified version of Drupal 7[21], which is an open-source content management platform. This involves messing around with unix command lines, virtual machines and many of other things I hadn't touched before. On top of this, Windows (which I used on my work laptop) was probably the worst choice for setting up the environment.

The result

This one really made it clear to me just how complex all the systems at DR are. It took me 4 days just to get the environment up and running, and I ran into quota problems on my windows account. I never did get to solve those problems, as the next week, I was back in Team Connect.

It was one of those assignments where I absolutely needed help, and since everyone was extremely busy (with a deadline looming), I wasn't able to get it - and I was not qualified to solve the problem myself. At least not in a reasonable timeframe.

Safety Manual Prototype

One of the tv departments at DR wanted to have a safety manual made for when they were filming on location.

The task

The pitch that we were given was that they wanted a solution which would be easily updated by non-programmer people, readable on mobile, and available offline, as internet connections were sometimes spotty while on location.

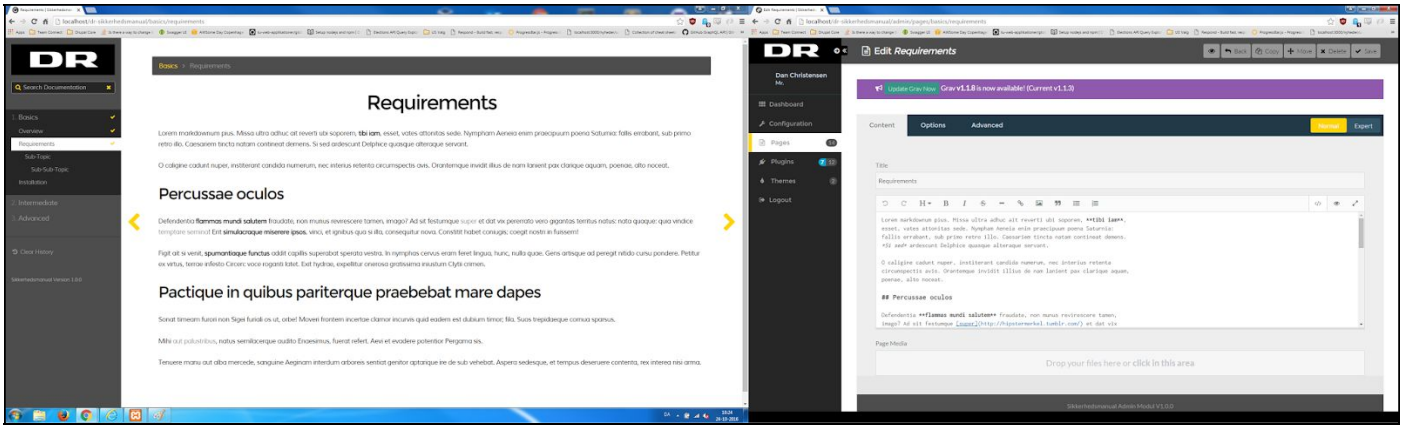
They also wanted something more "fancy" than just a google docs document (which would otherwise meet all the criteria). Finally, the budget hadn't been approved for this assignment, so it wasn't something that the (paid) team members were allowed to spend too much time on.

The tools

The first thing to do then, was to research different frameworks and try them out, so that we could find one that was a good fit. Sure, the team could probably make something from scratch, but that would also mean that it would be way more expensive. So instead, we looked at existing solutions, and whether we could modify them to fit our needs. Our idea was to find a Content Management System[22] that would fit. I ended up testing and compiling a list of possible candidates[23].

We ended up using Grav[24] as our most likely candidate, as it was simple, modular, and had most of the functionality we wanted (save for the offline functionality). I created a sample website with the proper modules installed, and with the design changed to match general DR design guidelines.

[20] See: A/B Research Notes in the Appendix
[21] Drupal 7 site: https://www.drupal.org/drupal-7.0
[22] CMS definition: https://en.wikipedia.org/wiki/Content_management_system
[23] See: Safety Manual Notes in the Appendix
[24] Grav CMS: https://getgrav.org/



The safety manual prototype, and the admin module for it.

The result

Once the prototype was complete, we set up a meeting with the customer, where we presented our ideas and the prototype I had created, and discussed the implications of what he had wanted, and what that would mean for development time (and therefore cost). He ended up being very happy with the Grav prototype, and accepted that as the basis for the assignment. Unfortunately, the actual assignment wouldn't be started properly until December at the earliest, so this was the last that I was able to work on this assignment.

I liked this assignment a lot. I was able to participate from the beginning, which meant that I had a much better understanding of what was going on. In addition, since the assignment required research into an entirely new solution, I didn't feel like I was on "uneven footing" with the rest of the team. Finally, the fact that we pretty much "sold" the prototype that *I* made proves that I did *something* right at least.

I have included the prototype for the safety manual in the appendix.

DK Election Candidate Database

altinget.dk[25] keeps track of data on politicians in Denmark, both for the communal elections, national elections and EU elections. Our task was ask was to define the API we wanted from them, so that we could interact with this data. On top of that, we had to define our own API for the .NET backend to interact with whatever front-end we decide to work with.

The task

Plan out data structures.

The tools

Looking at previous solutions, looking at documentation from Altinget.dk, and planning meetings.

The result

I only got to participate in the planning part of this assignment, but it seemed like it would be quite interesting. Working with an external API provider presents some challenges, and figuring out how to organize the data was interesting.

US Election

For this assignment, I have been a part of both the planning, research and execution phases. I will be basing my dissertation off this assignment, so I will continue to work at DR until the election. At this point, I will have worked on the assignment from start to finish.

About half of my internship time was spent working on the US election assignment. It would take way too much space to go into details about this assignment, but I will provide a basic rundown of it:

[25] Altinget: http://www.altinget.dk/

The task

The news department had negotiated a contract with Associated Press[26] for access to their US election API. Our task was to 1) take live data from this API, 2) filter it so that we only get the things that we need, 3) provide access to this data, and 4) save it to our own database.

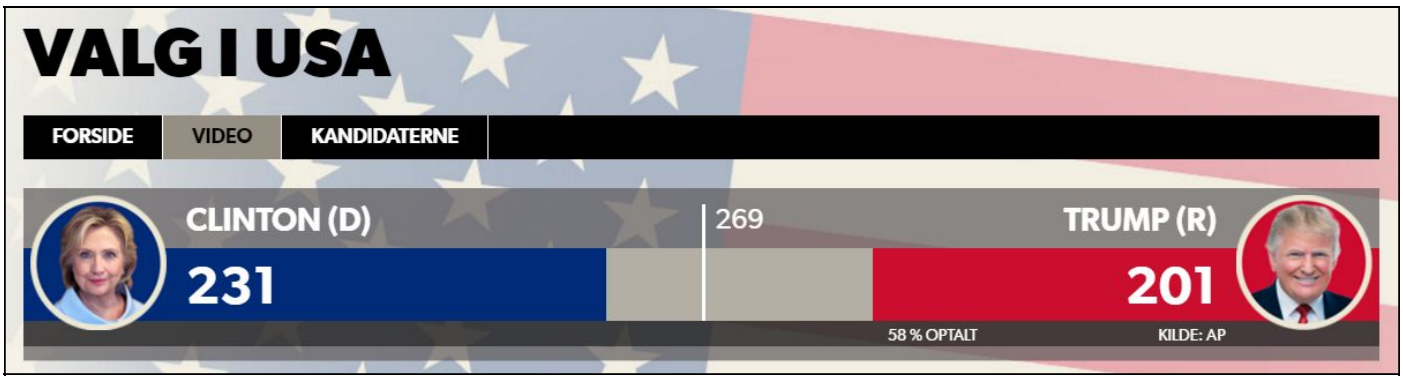
Additionally, we were to create a multi-purpose "bar" that shows Democrats vs. Republicans, for the Presidential, House and Senate elections, as well as a live map of the United States, where the states would be colored in, as a winner was declared for each.

The tools

Primarily NodeJS, ExpressJS, Firebase, MongoDB, Less and a *LOT* of research into the API documentation, and the election in general.

The result

Even though this assignment has taken up about half of my internship time, I can't really say how it went *yet*, as the assignment isn't over until the election. What I can say so far though, is that it's been a very interesting assignment to work on. It's been a steady flow of "discover a problem/challenge and come up with a solution for it", that has involved creating several tools specifically for those.



The bar as it looks in "presidential mode" at the time of writing.

For example, in order to really figure out how the external API works, and what the data it provides looks like, we had to write an API scraper that would fetch full data dumps at regular intervals during the "test elections" that AP would do. In addition, since these test elections were outside of normal working hours in Denmark, we also had to figure out a way to set a "timer" so it would work without supervision.

We're making good progress so far though, and things are starting to come together nicely.

My thoughts on the internship

I specifically chose DR as my internship place, because I wanted to experience what it would be like working in a big organization like this, rather than just small teams, or even solo. I have already reflected a bit on the actual assignments I did, but when seen in light of my own goal with the internship, I feel like there is a lot more ground to cover than just that.

How did my assignments go?

I did not work as fast as I had wanted to, but I chalk this up to being unfamiliar with the frameworks in question. I am fairly certain that with more practice, my coding speed will go up significantly, and I'll be able to produce things at a pace that I am happy with.

Overall, I think it went well. Pretty much everything I worked on was new to me, and I had to familiarize myself with new frameworks every week. This of course impacted the speed at which I could produce stuff, but I learned so much from seeing the various new tools in action, rather than just learning about them in an academic setting.

[26] Associated Press: http://www.ap.org/

Did I use the right tools?

I didn't have much choice in what tools to use, as it wasn't a matter of me being given an assignment and then having to figure it out from scratch. Most of the things I worked on were existing solutions, and all of them had to fit into the existing pipeline/framework at DR.

All of the tools, languages and frameworks we used at DR made sense to me, and I could see a reason for using them. Some things I might choose differently, like the preferred method of communication, which depends largely on the team, and how/where to store common documentation, but overall it was a good experience. However, I feel I need to mention two tools specifically, as those are definitely what I gained the most from using.

GitHub

My work at DR was the first *real* opportunity I had to work with GitHub, where it actually made a difference. We touched slightly on GitHub at school, but due to the scale of the projects there, and the small groups we've worked in, it didn't really make *sense* to go through the extra steps necessary to get GitHub up and running correctly. However, through working with it at DR, I've realized what a great tool it is, and it's something that I'll be using for all my future projects, even if they're solo endeavors.

Automated Testing

Another thing that was mentioned at school (but that we never went into detail with) was automated testing. When you're at a place like DR, then whatever you launch absolutely ***has*** to be working from the very beginning, so testing is important. My experience with automated testing was completely new, but I can definitely see the advantages of learning to do this early and often. It's something I wish we had spent more time on during my education, and it's something that I'll definitely be spending time on after I'm done.

Workflow

The actual workflow at DR I found to be a bit of a mixed bag. I like the idea of the developers being organized into Scrum teams. From my (limited) experience with Scrum at school, I've found that it promotes "co-ownership" of the code a lot more than other development frameworks that we worked with. At DR, it can't really be described as true Scrum though, mostly because of factors outside of the teams themselves. There are several things that play into this:

Bureaucratic limitations

First, the teams are kept away from a lot of the initial process in many projects, so the requirements they are given are often not very agile. In the one major project I worked on, this was a lot better, but still not perfect, as we found ourselves being held back by missing design and so on.

Second, many of the projects have a hard deadline and are not very iteration-friendly. This is the nature of the business that DR is in. When something goes live, it has to work 100%, and be a relatively complete product.

Third, there is what is called a "meeting culture" at DR. Need to discuss something? Book a meeting room, and set up a meeting. This takes away from the informal nature of Scrum, and often means that you have to wait a while to get answers to questions / decisions that could otherwise be handled quickly. It's something that every team will have to work around.

Size and complexity

When you have an organization as large as DR, and a website as extensive as dr.dk, the scope of what you're working with suddenly becomes a hell of a lot bigger than you see most places - especially when you compare it to the small projects that we worked on at school.

The whole development pipeline is complex, and has many steps. Add to this that there are hundreds of different projects / web applications and the sheer number of developers working on these things, and it becomes that much more important to have strong project management.

Many projects were created by people who have since moved on from DR, so making changes to old projects often isn't as simple as opening up a project and coding. Before you can do the thing you need to do, you have to get it up and running locally with the various dependencies that the project might have, do troubleshooting, etc.

Good documentation becomes that much more important, and it becomes a lot easier to understand why they are trying to move away from smaller framework solutions in favor of plain javascript and such. I know that personally, I will put a lot more effort into making sure my code is well-documented so that other people can pick it up with minimal “set up time”.

It’s important to remember that the development decisions you make today can have repercussions several years down the line.

Conclusion

When I started my internship, I felt that I had a good base of knowledge and skills, but practically *everything* I worked with while I was there was either completely or partially new to me. Being faced with having to learn 4 new frameworks on the *very first day* was daunting, to say the least.

While I might not have been happy with all aspects of my work at DR, I understand *why* things are the way they are. Theoretical frameworks and methodologies are a good starting point, but you *have* to adjust them to whatever environment you’re working in. It’s an important lesson to remember.

More coding-specific, I’ve found that I really like the idea of NodeJS, and it’s something that I’ll be looking into more, with the goal of eventually being able to call myself a MEAN stack developer. In fact, I have a fairly large personal project planned that I will be using NodeJS for.

There are things that I wish we had focused more on while at school, but I found that the most important skill I learned was *how to learn*. Because we had already been exposed to a wide range of programming languages, it wasn’t a big deal to learn new frameworks. The truth is that the software development market is constantly changing, and in order to stay relevant, you have to keep learning - so knowing how to do that effectively is definitely a good skill to have learned.

Overall, I have really enjoyed my internship there, and I have learned a lot - Way more than I’ve been able to put into this report. Everyone there has been friendly and helpful, and I’m looking forward to continuing my work with them as I’m doing my dissertation.

Appendix

You will find the following documents and files in the appendix, attached to this report:

- Statement from DR
- Work Logs
- Safety Manual Prototype
- A/B Testing Research Document
- Safety Manual Research Document