

Jukka Rautanen

Faktabaari.fi

Website, API and Cloud Hosting Solution

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree programme in Electronics

Thesis

10.05.2015

Tekijä	Jukka Rautanen
Otsikko	Faktabaari.fi Websovellus, rajapinta ja Azure pilvipalvelut
Sivumäärä	42 sivua + 9 liitettä
Aika	10.05.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Elektroniikan insinööritutkinto
Ohjaaja(t)	Janne Mäntykoski, Lehtori Jussi Salmo, Sirdar Oy
<p>Tämän opinnäytetyön aiheena oli verkkosivuston suunnittelu ja toteutus käyttäen sisällönhallintajärjestelmää. Sivusto sisältää avoimen rajapinnan ja se toteutettiin käyttäen pilvipalveluita. Työn alussa kerrotaan yleisesti sisällönhallintajärjestelmistä, Microsoft Azuresta sekä toteutettavan sivuston luonteesta.</p> <p>Työn pääasiallisena teemana keskityttiin toteutukseen WordPress-järjestelmän muunnoksella, joka täytti myös toteutetun rajapinnan vaatimukset. Tämän lisäksi työssä käsitellään Microsoft Azure pilvipalvelun teoriaa, sekä käytännön toteutusta tehdyille sivustolle.</p> <p>Opinnäytetyön lopussa on yhteenveto, jossa käydään läpi myös hankkeen nykytilanne.</p>	
Avainsanat	WordPress, HTML, PHP, websovellus, Azure, API, rajapinta

Author	Jukka Rautanen
Title	Faktabaari.fi Website, API and Cloud Hosting Solution
Number of Pages	42 pages + 9 appendices
Date	10th May 2015
Degree	Bachelor of Engineering
Degree Programme	Degree programme in Electronics
Instructor(s)	Janne Mäntykoski, Senior Lecturer Jussi Salmo, Sirdar Oy
<p>The subject of this thesis was planning and implementation of the website using a content management system. The website has an application programming interface and it uses a cloud hosting solution. In the beginning of this thesis there is theory about content management systems, Microsoft Azure and a brief explanation of the nature of the developed website.</p> <p>The main theme of this thesis was the actual implementation with the modified WordPress content management system, which also filled the requirements of the API. In addition to this also Microsoft Azure cloud hosting solution theory and implementation were explained.</p> <p>In the end of this thesis there is a conclusion, in which the current situation of the project is also reviewed.</p>	
Keywords	WordPress, HTML, PHP, Website, Azure, API, Cloud hosting

Contents

Abbreviation

1	Introduction	1
2	Content Management System	2
2.1	Introduction	2
2.2	Definition	2
2.3	Example	3
2.4	Uses	5
2.5	Advantages and Disadvantages	5
2.6	Common Content Management Systems	6
3	Microsoft Azure	7
3.1	Platform	7
3.2	Getting Started	8
3.3	Azure Web Apps	10
3.4	Azure SQL	12
3.5	Azure Redis Cache	13
4	Web Application	13
4.1	Planning and Prototyping	14
4.2	CMS	14
4.3	Layout	16
4.4	Editorial Process	18
4.5	Crowdsourced Editorial Process	21
4.6	Optimization	24
4.7	Analytics	25
4.8	Security	25
5	API	26
5.1	Planning	26
5.2	Plugin Development	27
5.3	Sample Response	31
5.4	Security	31
5.5	Performance	31
5.6	Demo	32

6	Cloud Hosting Solution	32
6.1	Microsoft Bizspark	32
6.2	Planning	32
6.3	Used Azure Services	33
6.4	Implementing Caching	34
6.5	Continuous Deployment	36
7	Conclusion	37
	Bibliography	38
	Appendices	
	Appendix 1 Faktabaari.fi - 3 main layouts	
	Appendix 2 Function generating latest posts in frontpage template	
	Appendix 3 Function sending email on new drafts	
	Appendix 4 Function submitting posts to database	
	Appendix 5 Faktabaari.fi - Tip form	
	Appendix 6 Security rules	
	Appendix 7 Faktabaari's API sample response	
	Appendix 8 Faktabaari's API response time	
	Appendix 9 Faktabaari's API demo	

Abbreviation

CMS	Content Management System
WCMS	Web Content Management System
HTML	Hypertext Markup Language
PHP	PHP: Hypertext Preprocessor
JS	Javascript
WP	WordPress
API	Application programming interface
SQL	Structured Query Language
Ms	Microsoft
JSON	JavaScript Object Notation
JSONP	JSON with padding
RSS	Really Simple Syndication
DNS	Domain Name System
PaaS	Platform as a service
IaaS	Infrastructure as a service
SaaS	Software as a service
DTU	Database throughput unit

1 Introduction

The subject of this thesis work was to redesign and develop a website for startup company called Faktabaari. Although Faktabaari already had a working website, a lot of changes were needed. It was decided that the best way to approach this situation was to build the new website from the ground up. As the specification of this work is broad, in each section the need, solution, results and conclusions are processed separately.

"Faktabaari is a Finnish factcheck service bringing accuracy to the public election debate. Faktabaari is a nonpartisan journalistic project using internet and social media for collecting and distributing factual information [1]." Faktabaari has 3 main channels to distribute its content; website, Facebook and Twitter. In addition to these Faktabaari has had media coverage in the Finnish radio and television. Faktabaari's staff consists of journalists and technical personnel. Eurooppatiedotus and Helsingin Sanomat foundation have funded the project.

First version of faktabaari.fi was made in quick schedule in spring 2014 for the European Parliament Election. The website needed new features, completely new layout, facelift and performance optimizations for the Finnish Parliamentary election. Faktabaari also needed a way to distribute its content to other medias, hence also an API was required. This thesis work covers all previously mentioned aspects.

2 Content Management System

2.1 Introduction

Content Management System is a software which allows users to create, edit, delete and organize content without the need of programming. This thesis is focused on Web Content Management System (WCMS), and it is later referred to as just Content Management System (CMS). The need for Content Management Systems has risen as organizations and businesses need to communicate even faster with their customers. Publishing a news article should be a just few clicks away for the typical user, and not need an experienced IT person in the middle. In addition to saving time, this also saves money, once the Content Management System is developed to use.

2.2 Definition

CMS can be considered a broad term and people define it differently depending on their industry and experiences with Content Management Systems. Bernard Kohan, a mobile app and web application development and technology analyst expert defines it [2] as follows:

The definition of a CMS is an application (more likely web-based), that provides capabilities for multiple users with different permission levels to manage (all or a section of) content, data or information of a website project, or internet / intranet application.

Managing content refers to creating, editing, archiving, publishing, collaborating on, reporting, distributing website content, data and information.

Web-based CMS doesn't explicitly mean that it publishes content on a website, nowadays web-based Content Management Systems are also used as backends for mobile appli-

cations, because of their ease of accessibility. For example Cloud CMS (developed by Gitana Software) is a software which provides API access and the content in JSON, so that it can be used for various purposes. [3].

2.3 Example

This example CMS shown in figure 1 is divided into four parts, content management, layout management, user management and extensions. The CMS in question could be used for a company website for example.

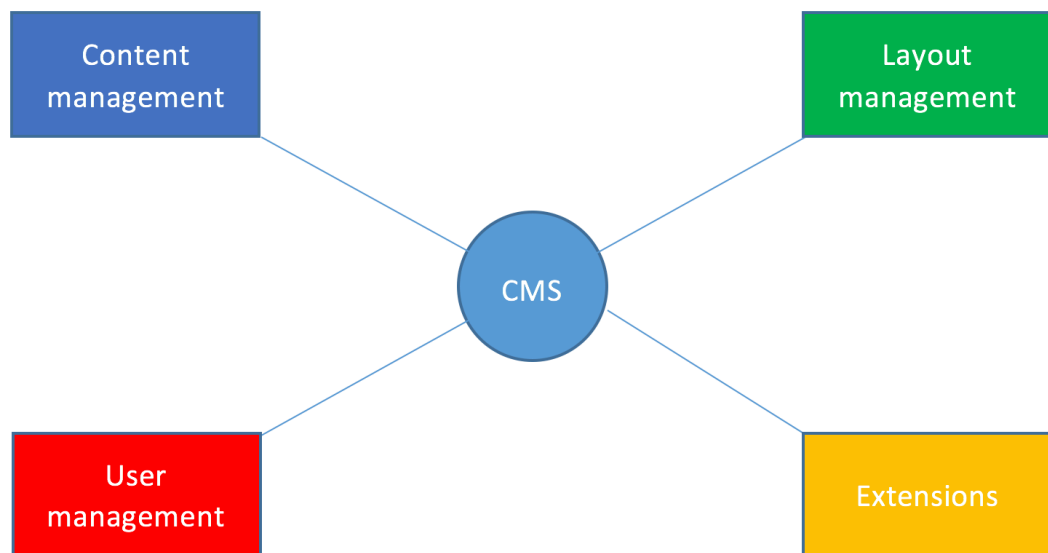


Figure 1: Example CMS

Content management here means creating and updating pages and publishing new articles to the company blog. Users can easily create and edit content using the admin interface, without programming anything. Anyone from secretary to the CEO can have their own credentials and provide content to the website. Figure 2 shows a screenshot from WordPress content editor, where the user has a similar toolbar as in Microsoft Word for example.

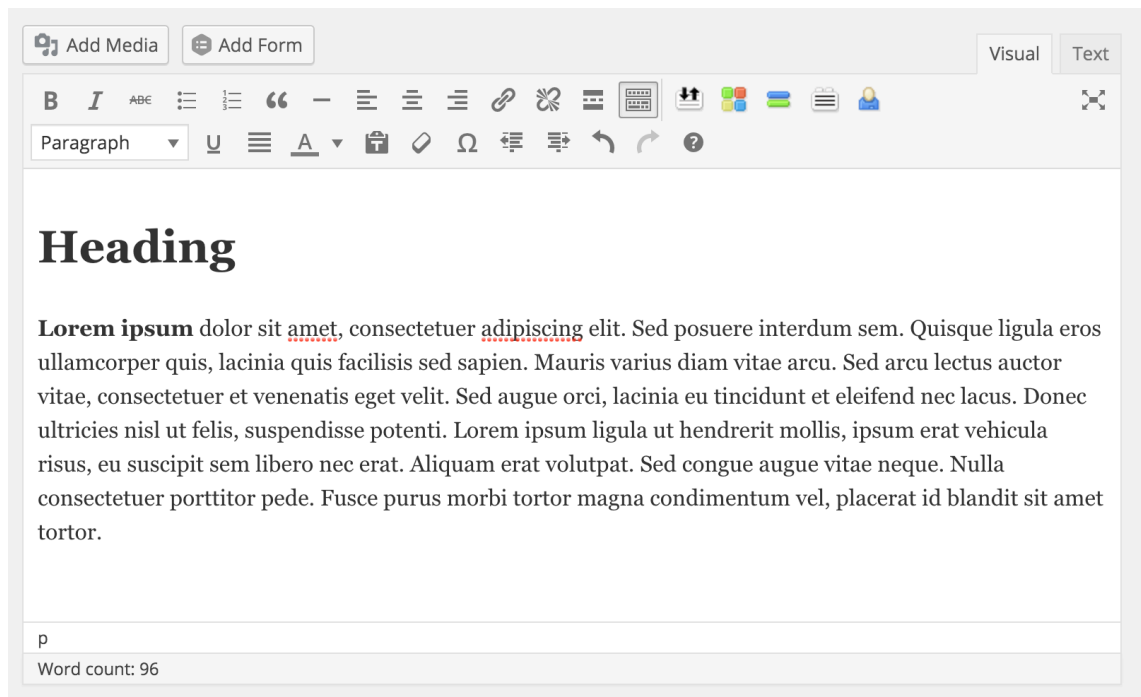


Figure 2: Content editor

Layout management means that during the development of the website, the programmer has created different layouts. One layout could be used for the blog articles, and one for "Contact Us" page, and so on. These layouts are coded, thus they are not editable by the normal users, but the users can select which layout they wish to use.

User management includes everything from creating the users to actually logging in. Most Content Management Systems provide this functionality, which saves time on the development. Users can have different roles, for example the secretary might only edit certain content, where the admin can manage all the settings.

Extensions and plugins are an important aspect of the CMS. Where most of the Content Management Systems provide the same base functionality, they need to be extended to meet the requirements of the client. The requirement can be a general need such as social media sharing buttons, or a really specific one, such as integration to the company's CRM system. Depending on the CMS these extensions are provided by 3rd party developers. They can be open source or commercial.

2.4 Uses

CMS can be developed to fit a specific need, for example a CMS might need to serve only specific type of XML files, or act as backend for another application. However there are common use cases of Content Management Systems for which most of the currently available softwares are developed for.

- The most common example of CMS usage is a simple website, where the author can edit, create, delete and organize content. This might include for example editing contact information or adding a new information page.
- Blogs publish articles, which can be easily created with a suitable CMS. For example WordPress, Anchor CMS, Ghost are ready-made solutions for this purpose.
- All the big news sites use some type of Content Management Systems. Some have developed their own, some use open-source project as their solution. The New Yorker, BBC America and TechCrunch are all using WordPress [4].
- Wikipedia is not the only wiki on the web. Many companies, projects and organizations have their own. There are also a lot of open-source Content Management Systems developed just for this purpose.
- Intranets and extranets are still here, and there are Content Management Systems for them. Most of them are offered as a commercial service, but open-source solutions are available.
- As mentioned earlier web-based CMS can act as backend for a larger application, or fit the specific need of the customer. These types of solutions are often developed for the use case from the ground up.

2.5 Advantages and Disadvantages

Cost can be considered as both advantage and disadvantage when using CMS. On the development phase, using ready-made solution saves a lot of time and development expenses, instead of reinventing the wheel the developers can get access to things like user-management and storage implementations.

Yet when the CMS has been developed, implementation can be expensive. Many of the Content Management Systems might have higher hardware requirements, which leads to higher maintenance costs. And of course the staff needs to be trained to use the developed CMS. Also depending on the implementation, the CMS might have yearly license costs.

Ease of use and customization are the biggest advantages of using a CMS. Updating or adding content does not need any modifications of code and instead of coding every simple feature, one might use plugins available.

In terms of search engine optimization, usage of CMS can be considered an advantage. This is because the content is updated easily, and it is in the correct format. Many of the Content Management Systems for blogs also provide an RSS feed, which can increase the number of readers to the site.

In some cases security is a disadvantage of CMS, because using popular solutions such as WordPress or Drupal means also that hackers tend to target these systems more often. Security breaches are patched quickly, but the installation of these patches are often left for the end user which leaves a lot of CMS installations vulnerable.

2.6 Common Content Management Systems

Content Management Systems are a quite competed area of software. WordPress started of as a blog platform, but has now evolved to a full CMS, and is challenging bigger players like Drupal and EPiServer. Below is a brief listing of some of the most commonly used Content Management Systems [5]:

- WordPress is written in PHP and is an open-source project developed by the WordPress Foundation. It was first released in 2003. WordPress has an active community of developers working on WordPress core, themes and plugins [6].
- Drupal is also written in PHP under open-source licensing. Compared to WordPress

Drupal is considered to be more customizable, but requires more development to achieve the desired results. Drupal was first released in 2001 [7].

- EPiServer is a commercial CMS which is built with Microsoft .NET technologies. EPiServer provides also solutions for e-commerces. The company was founded in 1994 [8].
- Joomla is a open-source CMS for publishing web content, but it also provided web application framework that can be used independently. Joomla was first released in 2005 and it is written in PHP [9].
- ModX is a smaller CMS and web application framework built with PHP. In addition to MySQL support it also supports Microsoft SQL Server. ModX was first released in 2004 and it is still being developed, but the speed of development has decreased [10].
- Concrete5 is CMS known for its ease of use. Using Concrete5 users can edit content directly from the web page instead of providing external admin interface. It is written in PHP [11].
- SharePoint is a commercial web application platform provided in the Microsoft Office server suite. SharePoint can be a lot more than a traditional CMS, web content management is just a small part of it. It was first released in 2001 [12].

3 Microsoft Azure

3.1 Platform

Microsoft Azure (formerly Windows Azure) is Microsoft's own cloud platform. Cloud platform refers to cloud computing, which means computing as a service over the Internet. By using such service you can scale your cloud application according to the need, for example when a website has suddenly more visitors than normally, it can scale up (add more computing instances) to be able to serve all its visitors without interruption. Instead of constantly having large computing power from which part would go to waste, by using cloud services you can save in costs by paying for only what you are using [13].

Microsoft Azure provides both IaaS and PaaS. IaaS means infrastructure-as-a-service

(for example virtual machines) and PaaS platform-as-a-service (websites, database, development tools..). Even though Azure is developed by Microsoft, it is open to any operating systems, frameworks, tools and programming languages. For example you can run Ubuntu Linux virtual machine or a Windows Server, develop Java applications or ASP.NET applications. [14]

Most commonly used competitors for Azure are Amazon's cloud services, Heroku, Google App Engine. All these provide basically identical services, but the user experience differs. From price point of view one can find little differences, but the general cost is the same. All these cloud service providers give free trials, which one should use before buying. [15]

3.2 Getting Started

In order to be able to use Microsoft Azure the user needs to have a Microsoft account. From the Azure website first time users can start a free trial, to test out the services before buying. Once signed in the users are able to see the dashboard. From the dashboard user can easily see the status of services in use, and manage them. Using the Subscription panel user can also edit the billing options.

Even though Azure provides a free trial, when a credit card is added, it's recommended to follow the costs of your services. For example by setting too high scaling properties, service costs can build up fast, and the user ends up with a big bill to pay. Also it is a common mistake to leave the virtual machine running when the user doesn't need it. The services that don't need to be constantly ran on the background should be shut down when they are not in use. With Microsoft Azure, the user can set up billing alerts on the billing page, so that the user gets notified if the bill gets bigger than 10\$ for example.

The basic Microsoft Azure dashboard is shown in figure 3. All the active services are listed on the table, if they exist. The user can browse all services at once, or show only services under certain category.

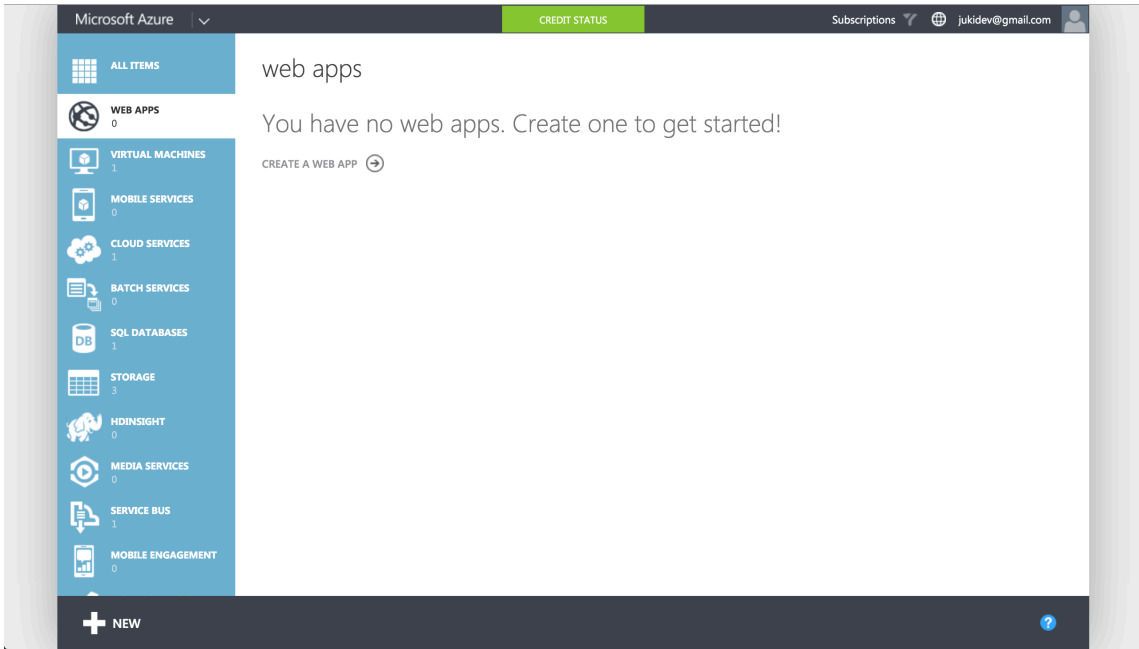


Figure 3: Azure dashboard

New services are created from the panel found after clicking the "New" button on the left bottom corner as shown in figure 4. Services can be chosen from ready made templates (such as the Ubuntu Server shown in the figure below) or from the marketplace, where 3rd party companies can offer their own services to be bought.

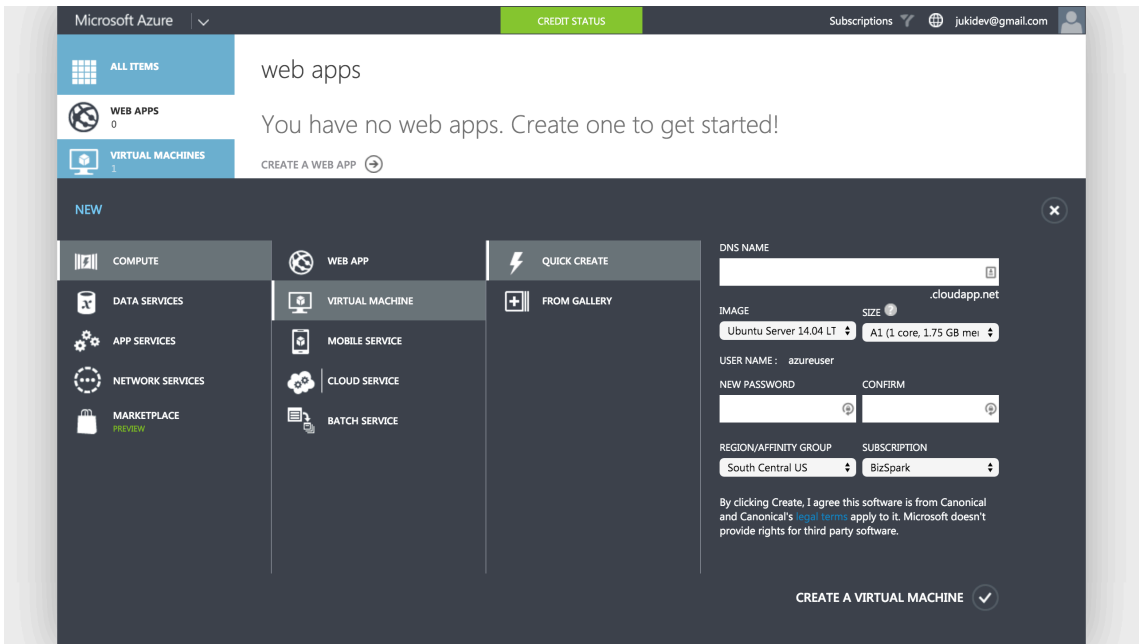


Figure 4: Create a new service

3.3 Azure Web Apps

Azure Web Apps is a platform with which you can build and scale web applications rather quickly [16]. By using web apps the user doesn't have to worry about the infrastructure, because this is managed by Azure. Web apps are for example ASP.NET sites, node.js applications, WordPress sites and so on. Faktabaari is also using web app to host the forked WordPress installation. To clarify, by using Azure web app one could for example in Faktabaari's case skip setting up a virtual machine, installing Linux distribution, Apache web server, MySQL database and PHP on it. Instead of this installation process one can select desired application stack from the App Gallery and all installations are managed for you. Ease of use is not just limited to the installation process, it means also ease of managing and maintaining the infrastructure. By using web apps, user just sets up desired preferences and focuses on coding the app itself. Glimpse of these preferences can be seen in figure 5 below:

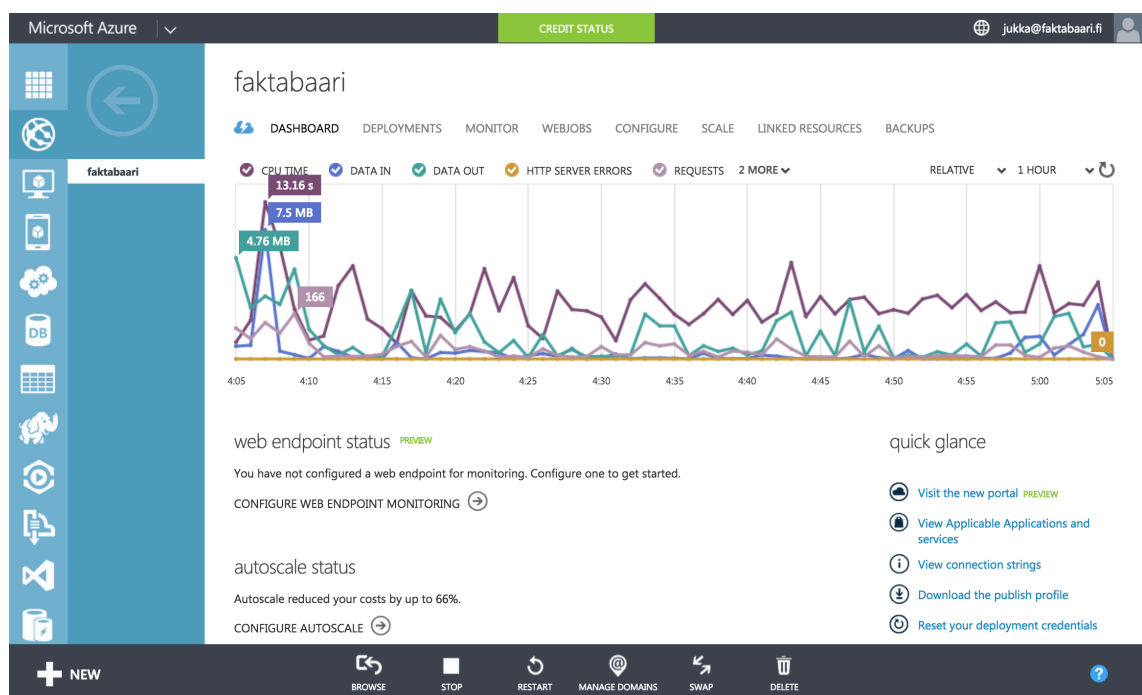


Figure 5: Admin view from Azure web app

One important concept of the web app is auto scaling, which can automatically scale your application up or out so that it is able to handle all the incoming traffic from users. User can set up his/her own preferences, which include scheduled scaling (for example

scale up on rush hours) or set up scaling using parameters like CPU or memory usage. Scaling of course needs to happen downwards as well, when the higher demand is over the application should scale down to lower resources. Figure 6 shows an example on how the user can also see from the admin panel that auto scaling has reduced costs:

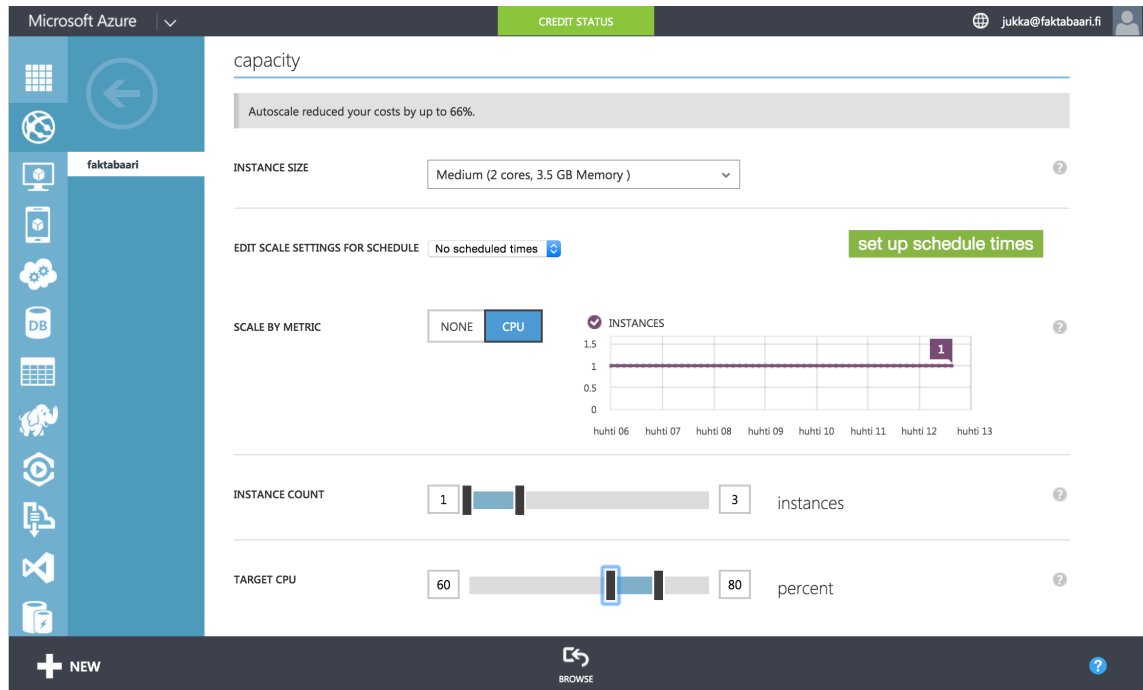


Figure 6: Auto scaling Azure web app

One of the most useful features in Azure web apps is the ability to have multiple deployment slots. This means that one can do staged development very easily. Stage development in general means that one has staging and production versions of the application. Staging is the version under development and production is the one customers can see when visiting the site. Azure provides an easy way to create and swap these slots. This means that when the software has a new version tested and ready on the staging slot, one can swap the slots between staging and production, so that the customers can now see the new version. If something goes wrong, the changes can be reverted easily by just swapping again. Azure ensures that all the instances that are used are made ready before being swapped, which eliminates any downtime. [17]

Azure Web apps also have a feature for using continuous deployment [18]. Continuous deployment integrates in the service selected, which can be for example GitHub or Bit-

bucket repositories or Visual Studio Online account. Continuous deployment can be set for example so that every time a developer pushes new code to the version control system, the new version of the app is deployed. This help teams to save time by removing the step of moving data back and forth to the server.

3.4 Azure SQL

One of the many services in Microsoft Azure is the Azure SQL, which is a relational database-as-a-service. Microsoft states that it "delivers predictable performance, scalability, business continuity, data protection, and near-zero administration to cloud developers and solution architects"[19].

These databases are divided to different classes. Each class offers different performance, for example with class S1, the user gets 20 DTUs. DTUs (database throughput units) are a relative measure of the resources provided to the database. From the admin panel one can easily monitor the performance of the database and scale it accordingly as shown in figure 7:

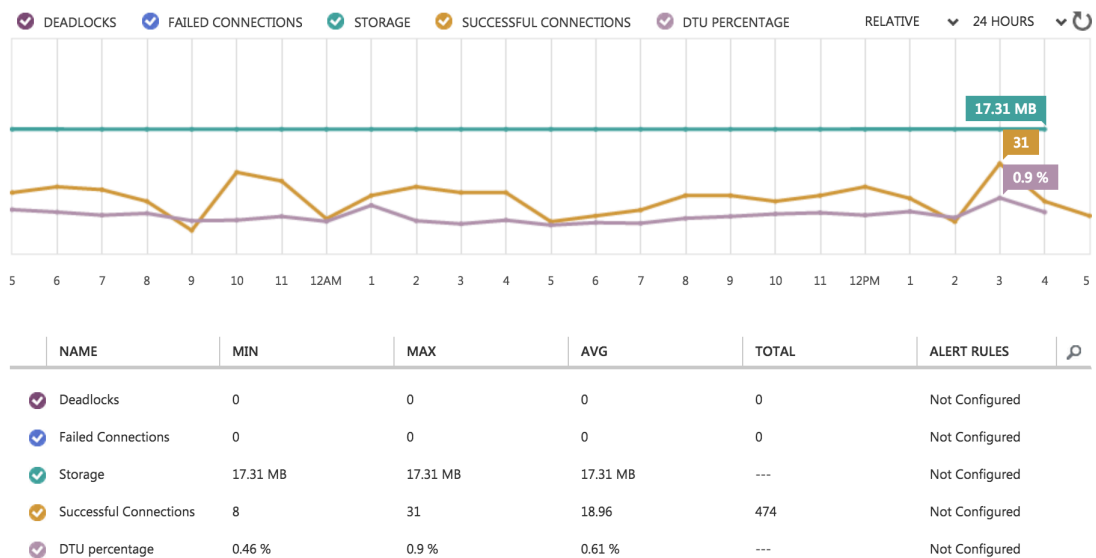


Figure 7: Azure SQL monitoring

If needed Azure SQL also supports geo-replication. By using geo-replication one can make sure that the contents of database are still served from another location if one server farm suffers from problems. Microsoft states the following on its website [20] about the geo-replication of Azure SQL databases:

The Active Geo-Replication feature implements a mechanism to provide database redundancy within the same Microsoft Azure region or in different regions (geo-redundancy). Active Geo-Replication asynchronously replicates committed transactions from a database to up to four copies of the database on different servers. The original database becomes the primary database of the continuous copy.

Geo-replication would be a useful feature for bigger websites for example.

3.5 Azure Redis Cache

Redis is an advanced key-value cache and store, one could say that it is a data structure server. These keys can contain strings, hashes, bitmaps, sets, sorted sets, lists and hyperlogs. It is recommended that Redis is deployed using Linux, but it can be done with most POSIX systems. Redis can be used for a lot of things, and one of these is caching. [21]

Azure Redis Cache is based on the open-source redis mentioned above. It can be bought as a service and it gives one a secure, dedicated Redis cache, which is managed by Microsoft. Azure Redis caches can be used from any application within the Azure ecosystem. [22]

4 Web Application

In this section all layers of the website used in production are covered and snippets from the source code are shared. Even though this project is built upon open source platform some 3rd party plugins were also used, hence the whole source code cannot be shared

directly because of licencing issues. All the source code published in this document can be used according to Creative Commons Attribution license [23]. Faktabaari's API and user-posts plugins are also published in GitHub for open usage and contribution. [24]

4.1 Planning and Prototyping

This website was developed using agile method called Scrum, which is a framework for agile software development [25]. The rules of scrum were altered a bit to match our project. There was one product owner, one scrum master and the development team consisted of one person only. Using one week iterations, for each week set of tasks were defined and at the end of the week the same task were reviewed. If the task was done correctly according to the specification, it was archived, if not, it was set for the next week for a redo. Using this type of agile method, we were able to keep track of spent hours on the project and estimate the time it would be ready. Also constant feedback was gathered from the beta users, and we were quick to adapt to changes in the requirements of the project.

4.2 CMS

For the content management system following features were crucial: user management, content management, media management, security, extendability, scalability and ease of usage. There are several choices for the content management system in the market for this type of media website, for example Drupal [26], but WordPress was chosen for Faktabaari. The staff of Faktabaari already had experience with it and it was a known CMS for the developer also.

Faktabaari.fi uses a modification of WordPress called Project Nami [27]. WordPress is an open-source content management system which is based on PHP and MySQL. It started off as a blogging platform, but has evolved into fully scalable CMS. WordPress excels in customizability and usability [28]. This project uses MsSQL as database instead of

MySQL. MsSQL was chosen as database, because it is natively usable in Microsoft Azure without 3rd party providers.

Using MsSQL database is made possible with multiple edits in WordPress core, in general all MySQL queries need to be converted to MsSQL queries. This generates errors with some of the common WordPress plugins and needs to be taken into account by the developer. Listing 1 shows an example of query that needs to be altered:

```

1 <?php
2 global $wpdb;
3 $author_id = 12;
4 $result = $wpdb->query( $wpdb->prepare( "SELECT ID from
      $wpdb->posts where post_author = %d AND post_status = '
      publish' AND post_type = 'post' LIMIT 10", $author_id )
    );
5 $authors_post_ids = wp_list_pluck( $result, 'ID' );
6 ?>

```

Listing 1: \$wpdb query with MySQL database

Above query needs to be converted to use the WordPress API in order for it to work with MsSQL database in Azure, which is shown in listing 2:

```

1 <?php
2 $author_id = 12;
3 $query = new WP_Query( array( 'author' => $author_id ) );
4 $authors_post_ids = wp_list_pluck( $query->posts, 'ID' );
5 ?>

```

Listing 2: Query with WordPress API

While developing the new version Faktabaari.fi multiple plugins were edited and it took additional time, but at the same time code quality with database queries was optimized and verified. It is important that there are no additional queries to database in order for the website to work fast and efficiently.

For example when querying review from the post meta tags, in previous version of Faktabaari loop shown in listing 3 was used :

```

1 $mykey_values = get_post_custom_values( 'review' );
2 foreach ( $mykey_values as $key => $value ) {
3     ...

```

Listing 3: Old query

And in the new version by querying is done to the specific ID directly, this way the data is only fetched once from the database as shown in listing 4:

```
1 $value = get_post_meta( $id, 'review', true );
2 // check if the custom field has a value
3 if( ! empty( $value ) ) {
4     ...
```

Listing 4: New query

The code in listing 3 will give the wanted result with MySQL, but with MsSQL it will generate multiple hits, because posts include old metadata from the old system. By querying the desired ID directly we also make fewer requests to the database.

The rest of the WordPress core is mostly unedited, the way it should be. By editing WordPress core the developer can also create security breaches. There is a whole open source community behind WordPress, and it is updated frequently. Keeping up to date is crucial for WordPress security, and with multiple edits in the core itself, there is a chance that some of the edits stop working or create other issues [29, Chapter 4].

4.3 Layout

Layout in WordPress is determined by the used theme. Themes consist of template files, such as stylesheets and php templates [30]. These files can be created and modified so that the brand and layout of the site owner is met. In previous section hazards of editing WordPress core were mentioned, and editing theme files is a great alternative to editing core files. In addition to editing theme files, creating plugins can also modify how WordPress works.

When comparing the old site, Faktabaari needed new layout in every aspect. According to the feedback the site was too "blog-like" and the article pages contained too much extra information. When planning and designing the new website this was kept in mind, and the goal was to create a clutter-free website focusing on the content itself.

The layout in Faktabaari.fi is based on a commercial WordPress theme Divi by Elegant-Themes, however this theme is greatly modified for Faktabaari. While having limited resources on budget, it is common to start building your own WordPress theme on top of existing one. Editing theme means editing or creating new php template layouts, editing stylesheet file to match the brand of the product and customizing themes functions [29, Chapter 9]. Faktabaari.fi has 3 main layouts; the front-page, single page and single post as shown in Appendix 1. Each layout uses the same header, navigation and footer with slight modifications on each page. Every layout is designed with minimalistic approach and to be fully responsive. Responsive web design is explained at Google Developers [31] the following way:

Responsive web design, originally defined by Ethan Marcotte in A List Apart responds to the needs of the users and the devices they're using. The layout changes based on the size and capabilities of the device. For example, on a phone, users would see content shown in a single column view; a tablet might show the same content in two columns.

The front-page consists of two main sections, latest facts and latest blog posts as shown in Appendix 1, figure 14. The content for these sections is fetched from the database with similar logic, but the output is altered for both sections. Appendix 2 includes the function for fetching the data and generating the output. The function is used from a shortcode in the frontpage template. Shortcodes are text macro codes that can be used in templates, and in post and page content [29, 174]. This function first extracts user input from the shortcode (lines 4-20) and then parses them to be usable later. The function uses basic wordpress post loop, but the generated output is modified (lines 42-160). The latest facts section has a masonry layout. This masonry layout is created with JavaScript grid layout library [32]. The library calculates position for each post separately on page load thus creating a dynamic grid. If the function is used in the frontpage, the first post will be Faktabaari's own Twitter feed (lines 39-41). Also for each post the review is fetched from the database and corresponding image and text is shown (lines 86-102). For the last post a card containing "Read more facts" text is generated (lines 186-188).

Second of the three layouts is the single post layout which can be seen in Appendix 1, figure 15. This layout is used for all the checked facts and blog posts. It is clutter-free and

focuses on the content itself. If the article has featured image set, it will be shown on top of the text content. Every post is sharable on the social media using the sharing buttons. Logic for fetching data from the database is the same as in front-page layout, but here we are fetching only one specific article.

The single page layout is the third layout in Faktabaari.fi. It is used on all the static pages, mostly with the pagebuilder plugin that comes with Divi theme. This pagebuilder gives the staff members easy way to create visually appealing content using different modules without programming anything. The layout is shown in Appendix 1, Figure 16.

4.4 Editorial Process

Editorial process is made simple. Each member of the staff have their own accounts, which with they first log in to the WordPress admin panel. The WordPress log-in screen is shown in figure 8 below:

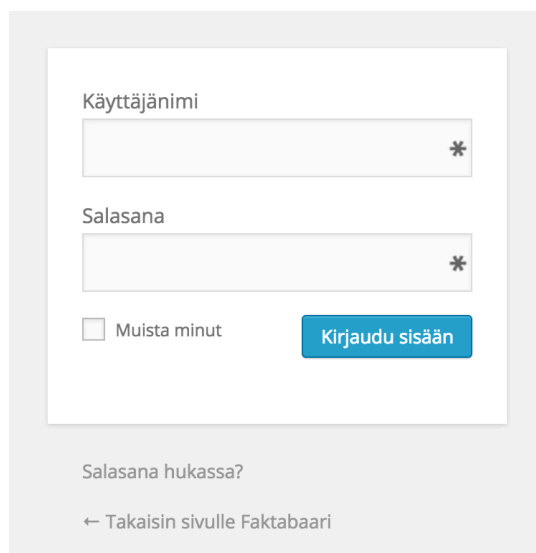
The image shows a WordPress login form. It features two input fields: 'Käyttäjänimi' (Username) and 'Salasana' (Password), both with asterisks indicating required fields. Below the password field is a checkbox labeled 'Muista minut' (Remember me) and a blue button labeled 'Kirjaudu sisään' (Log in). At the bottom, there is a link 'Salasana hukassa?' (Lost your password?) and a back arrow with the text '← Takaisin sivulle Faktabaari' (← Back to Faktabaari page).

Figure 8: Login screen

After successfully logging in, the journalist will select if he/she is writing a blogpost, a checked fact or creating a new page. This is made possible using custom post types. These custom post types are shown in figure 9 below (menu items faktat and EU-faktat):

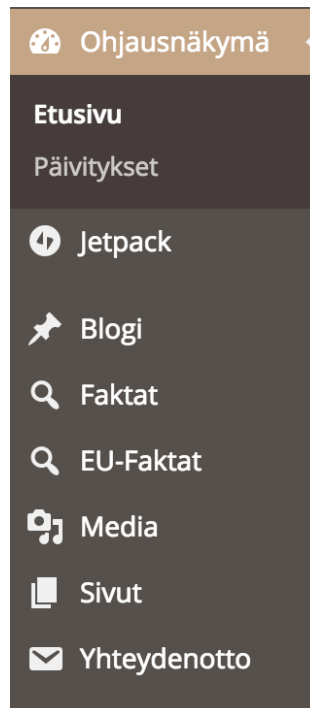


Figure 9: Part of the admin menu in WordPress

By default, WordPress have 5 post types; post (article), page (static content), attachment (uploaded media), revision (backup) and nav menu (menu items). However often these are not enough, thus creating custom post types will help. Custom post types are basically custom defined pieces of content. Common examples of custom post types are products, events and testimonials. [29, Chapter 7] Custom posts types in Faktabaari.fi are facts and eufacts, which are created with WordPress plugin called Types [33].

After selecting for example a fact, the journalist is greeted with default WordPress post editor. Here the writer chooses a heading, writes the content, includes pictures, adds keywords and selects a category. In addition to this, when writing a fact or eufact, the journalist needs to select the review for it (true, 50/50, false). This is made with custom fields in the post type. There is also a field for notes, which the journalist should fill in for internal use as shown in figure 10 below:

The image shows a screenshot of a WordPress custom field interface. It consists of two main sections, each with a title and a small upward-pointing triangle icon in the top right corner.

The first section is titled "Valitse jokin seuraavista" (Select one of the following). Below the title, under the heading "Review", there are three radio button options: "Totta" (True), "50/50", and "Väärin" (False).

The second section is titled "Muistiinpanot" (Comments). Below the title, under the heading "Lisätiedot" (Additional information), there is a line of text: "Kirjoita tähän kenttään muistiinpanoja faktasta esimerkiksi asiantuntijan nimi, työryhmän jäsenet jne." (Write in this field comments from the fact, for example, the name of the expert, the members of the working group, etc.). Below this text is a large, empty rectangular text area with a small cursor icon in the bottom right corner.

Figure 10: Custom fields in WordPress

After writing the content, the journalist clicks the save draft button. He/she could publish it immediately, but it has been agreed that the journalists check each others work before publishing anything. So after saving a draft, there was a need for function which informs the other about the new draft. This is the function in Appendix 3. On line 2 the function first hooks in to the WordPress `save_post` - action hook. Brad Williams D and Hal Stern define hooks in Professional WordPress [29, 127] the following way:

Hooks are simply a standardized way of “hooking” into WordPress. Using hooks, you can execute functions at specific times in the WordPress process, allowing you to alter how WordPress functions and the expected output.

On line 6 the function first checks that the saved post is a draft, as one does not want to be informed about published articles. On the lines 7-23 the function gathers and modifies the information about the post and on lines 27-34 the information is sent with email.

4.5 Crowdsourced Editorial Process

On the Finnish Parliamentary Election Faktabaari decided to crowdsource editorial work even more due to the expected high workload. In the spring 2015 Faktabaari started partnership with Haaga-Helia's journalist students and with Ilmastotieto. Faktabaari's staff was lecturing and teaching students on their course and also Ilmastotieto's staff was instructed.

The staff needed a way to receive checked facts from these partners easily, and the staff's email was already swamped. Thus the editorial staff's cooperation form was created. It is password protected so only verified partners can submit facts with it. The user submits his/her name, contact information, heading of the fact, keywords, attachments, fact review, electoral district and then the actual content. The user can use all the basic tools and styles (same which you can find in Microsoft Word for example), so that the staff doesn't need to edit the text afterwards. The submitted facts go directly to the WordPress database, with all the corresponding data, and are saved as drafts. When the new draft is saved, Faktabaari's staff is notified of it by email. Then the staff can login to WordPress, make modifications if needed, and publish the submitted fact. The form accessible for selected users is shown in figure 11 below:


```

6 $review = $_POST['user-submitted-review'];
7 $atc = sanitize_text_field($_POST['user-submitted-atc']);
8 $category = intval($_POST['user-submitted-category']);
9 $content = stripslashes($_POST['user-submitted-content']);

```

Listing 5: Sanitizing inputs

Then the submission is handled and submitted to database. Error handling is also implemented. After submission (or fail) the user is redirected and the corresponding message is shown (success or fail) as shown in listing 6:

```

1     $publicSubmission = usp_createPublicSubmission($title,
2         $content, $authorName, $authorEmail, $authorID,
3         $authorUrl, $tags, $review, $atc, $category,
4         $fileData);
5     if (is_numeric($publicSubmission)) {
6         $redirect = empty($usp_options['redirect-url']) ?
7             $_SERVER['REQUEST_URI'] : $usp_options['redirect-
8                 url'];
9         if (!empty($_POST['redirect-override'])) $redirect =
10            sanitize_text_field($_POST['redirect-override']);
11        $redirect = remove_query_arg(array('submission-error',
12            'error'), $redirect);
13        $redirect = add_query_arg(array('success' => 1, '
14            post_id' => $publicSubmission), $redirect);
15        do_action('usp_submit_success', $redirect);
16        wp_redirect($redirect);
17        exit();
18    } else {
19        $errorMessage = empty($usp_options['error-message']) ?
20            __('An error occurred. Please go back and try
21                again.', 'usp') : $usp_options['error-message'];
22
23        if (!empty($_POST['redirect-override'])) {
24            $redirect = sanitize_text_field($_POST['redirect-
25                override']);
26            $redirect = remove_query_arg(array('success', '
27                post_id'), $redirect);
28            $redirect = add_query_arg(array('submission-error'
29                => '1', 'error' => $publicSubmission), $redirect)
30                ;
31        } else {
32            $redirect = sanitize_text_field($_SERVER['
33                REQUEST_URI']);
34            $redirect = remove_query_arg(array('success', '
35                post_id'), $redirect);
36            $redirect = add_query_arg(array('submission-error'
37                => '1', 'error' => $publicSubmission), $redirect)
38                ;
39        }
40    }

```

```
22     do_action('usp_submit_error', $redirect);
23     wp_redirect($redirect);
24     exit();
25 }
```

Listing 6: Submitting user post

The above code block calls `usp_createPublicSubmission` function, which handles inputs even further and does the actual submission to the database, this function is shown in Appendix 4.

In addition to this editorial form, Faktabaari.fi has 2 different contact forms. The first one is shown in Appendix 5, and it is used for tips about claims to be checked and the another one is traditional contact form. Both of these forms are made with WordPress plugin called Contact Form 7, which is a very popular contact form plugin [35]. Faktabaari.fi is using this plugin's code exactly like it is provided. There was no need for code modification.

4.6 Optimization

Most of the optimization is done by caching, which is addressed later in this thesis work. However, there are also a couple of other optimizations in place. All the images are optimized before they are showed to the public. This is done with service called WP Smush.it [36]. Optimizing images is an important part of web development, Google Developers [37] explain it the following way:

Images often account for most of the downloaded bytes on a web page and also often occupy a significant amount of visual space. As a result, optimizing images can often yield some of the largest byte savings and performance improvements for your website: the fewer bytes the browser has to download, the less competition there is for the client's bandwidth and the faster the browser can download and render useful content on the screen.

In addition to image optimization, Faktabaari.fi is also minifying js and css files. While one might think this is really small save of bandwidth on page load, but when that save is multiplied with thousands of users, the minification comes beneficial. The minification is done with WebJob in Azure, which is explained later in this thesis work.

4.7 Analytics

Faktabaari.fi is using both Google Analytics and Jetpack statistics for analyzing site's traffic and referrers. Google Analytics [38] is a service for showing statistics about a website's traffic and its traffic sources. Jetpack is a WordPress plugin [39] providing similar functionality directly inside WordPress. The reason two different providers were selected is that there is always inaccuracy in these analyzes, and by using two sources we get more accurate reports on traffic.

In order to develop Faktabaari's marketing strategy to the right direction, we especially need to analyze where the users are coming from. For example statistics show that the top 3 referrers for Faktabaari in week 11 were Facebook, Twitter and Google. Thus, Faktabaari could spend most of the marketing money on Facebook promoting its posts, to get even more users to visit the site. The rest could be divided between Twitter and Google AdWords marketing campaigns.

4.8 Security

The most important act concerning security is keeping WordPress and used plugins up to date, but also small modifications are done for improving security even further. For security reasons all of them can not be explained here.

Just by using non MySQL version of WordPress, Faktabaari.fi has an advantage over traditional WordPress sites, because most of the attackers and bots target the database, and MySQL syntaxes don't work on this installation.

Faktabaari.fi is also blocking widely known bad user agents, by using custom rules defined in web.config file. Also according to the article "Hardening WordPress" [40] custom rules have been made for securing WordPress source files in the folder wp-includes. The rules

can be seen in Appendix 6.

5 API

5.1 Planning

As mentioned earlier, Faktabaari needed a way to distribute its content to other parties. WordPress does offer a RSS feed of posts [41], but this did not meet all the criteria for Faktabaari. Faktabaari's API needed to give response as JSON instead of XML and also needed to support JSONP for js clients. JSON is a lightweight data-interchange format [42], and JSONP is "JSON with padding", which is used in JavaScript programs running in clients web browser to request data from a different domain. This is normally prohibited because of the same-origin policy [43], but JSONP uses the fact that this policy is not enforced on <script> tags.

In addition to these technical requirements, the API needed to give response according to the specific parameters; post type, post category, search by text and search by author. This enables the API user for example to query all the latest posts or the latest posts concerning only economy. Another usage example is that Ilmastotieto can query only posts written by themselves, and show these on their own website.

There were no additional specific requirements for the API, because it was developed to this point before any actual client case. The API is being developed further using agile methods, meaning that the API can and will be modified and updated for client cases providing all the data or query parameters they need.

5.2 Plugin Development

It was obvious that someone else has already had a similar task, and there are plenty of plugin choices from where to start from. The reason Thermal-API was chosen as a starting point was that it is completely built on top of WordPress WP_Query object [44]. First of all, the syntax of WP_Query was already familiar, so there was no need to learn new syntaxes. Secondly, and most importantly, Thermal uses WordPress' own internal API, where caching is already provided.

The plugin was edited with a mindset that the client is only downloading data from the API, not uploading anything or setting any options, so the API didn't need any other methods to be allowed than GET. In future POST-method maybe also enabled, if it is needed for some client case. The code that this section is referring to is published in GitHub for open usage and contribution [24]. Only some of the modifications are addressed in this thesis work, because of the whole plugin consists of thousands of lines of code.

First of all, Faktabaari needed its own API base, meaning the slug in the URL which the API lives in. This can be changed as shown in listing 7:

```

1  if ( !defined( __NAMESPACE__ . '\\API_BASE' ) ) {
2    define( __NAMESPACE__ . '\\API_BASE', '/fapi' );
3  }

```

Listing 7: API Base

Listing 8 shows that in addition to this the URL is using the API version. This way, we can have two versions of the API running at the same time if needed, and they can have different query syntaxes.

```

1  public function __construct() {
2    //if requested url starts with api_base_url()
3    if ( false !== strpos( $_SERVER['REQUEST_URI'],
4      get_api_base() ) ) {
5      require_once( 'api/v1/API.php' );
6      add_action( 'wp_loaded', array( $this, 'dispatch_api' ) );
7    }
8  }

```

```
7 }
```

Listing 8: API Constructor

Thermal API provides a lot of different controllers by default (posts, users, taxonomies, rewrite_rules and comments), however Faktabaari needs only to serve clients information about the posts as shown in listing 9:

```
1 public function __construct( \Slim\Slim $app ) {
2     parent::__construct( $app );
3     $this->registerRoute( 'GET', '?', array( __NAMESPACE__ .
4         '\\controllers\\Posts', 'find' ) );
5     ...
6 }
```

Listing 9: API routes

Now we are at the point, where the client has queried the URL and is being "redirected" to the Posts controller. Listing 10 shows that the post controller then parses clients parameters safely to be used in order to give the right response:

```
1 protected static function convert_request( $request_args ) {
2     // Remove any args that are not allowed by the API
3     $request_filters = array(
4         'category_name' => array( ),
5         'name' => array( ),
6         's' => array( ),
7         'post_type' => array( '\\Voce\\Thermal\\v1\\toArray' ),
8     );
9     //strip any nonsafe args
10    $request_args = array_intersect_key( $request_args,
11        $request_filters );
12    //run through basic sanitation
13    foreach ( $request_args as $key => $value ) {
14        if ( isset( $request_filters[$key] ) ) {
15            foreach ( $request_filters[$key] as $callback ) {
16                $value = call_user_func( $callback, $value );
17            }
18            $request_args[$key] = $value;
19        }
20    }
21 }
```

Listing 10: Query handler

At this point the code is also modified so that if when for example the name parameter is set, the API will give response from all the post types, instead of just the default post (blogpost). Listing 11 shows the query handler which is responsible for this:

```

1 //Force all post types to be fetched if name parameter is
  set
2 //This increases response time but we need to get posts from
  all types
3 //If we wouldn't do this we could only get name matches from
  blogposts
4 if ( (isset($request_args['name'])) || (isset($request_args[
  's'])) || (isset($request_args['category_name'])) ) {
5 $request_args['post_type'] = array();
6 array_push( $request_args['post_type'], 'eu', 'fakta', 'post
  ');
7 }

```

Listing 11: Query handler

Then the query is processed in a similar way like in default Thermal installation. However the output is modified so that only the required data is given in the response as shown in in listing 12 below:

```

1 $post_more = get_extended( $post->post_content );
2 $content_display = $post_more['extended'] ? $post_more[
  'extended'] : $post_more['main'];
3 $author_id = $post->post_author;
4 $author_name = get_the_author_meta('display_name',
  $author_id);
5 $post_type = "";
6 //Clean post_type names for output
7 if ($post->post_type == "post"){
8   $post_type = "blogpost";
9 }
10 if ($post->post_type == "eu"){
11   $post_type = "eufakta";
12 }
13 if ($post->post_type == "fakta"){
14   $post_type = "fakta";
15 }
16 $data = array(
17   'type' => $post_type,
18   'author' => $author_name,
19   'published' => ( string ) get_post_time( 'j.n.Y - H:i',
  true, $post ),
20   'title' => $post->post_title,
21   'content' => apply_filters( 'the_content',
  $content_display ),
22   'url' => get_permalink( $post ),
23 );

```

Listing 12: Output modification

In addition to this, Faktabaari's posts also have some extra data that needs to be included, the post category needs to be set as electoral district and the review of the fact also needs to be shown in the response as shown in listing 13. The media URL is the featured image of the article, if such exists.

```

1 //Add media url only if featured image exists
2 $mediaurl = wp_get_attachment_url(get_post_thumbnail_id(
    $post->ID));
3 if($mediaurl != false){
4     $data['image'] = $mediaurl;
5 }
6 //Get post category (=vaalipiiri)
7 $catlist = "";
8 $categories = get_the_category( $post->ID );
9 $separator = ' ';
10 $output = '';
11 if($categories){
12     foreach($categories as $category) {
13         $output .= $category->cat_name.$separator;
14     }
15 $catlist = trim($output, $separator);
16 }
17 if ($post->post_type == "fakta" || $post->post_type == "eu")
    {
18     if($catlist != false){
19         $data['vaalipiiri'] = $catlist;
20     }
21 }
22 //Add review to array if it exists
23 $review;
24 $mykey_values = get_post_custom_values( 'wpcf-review' );
25 if (isset($mykey_values)){
26     foreach ( $mykey_values as $key => $value ) {
27         $review = $value;
28     }
29 $data = array_slice($data, 0, 4, true) +
30 array("review" => $review) +
31 array_slice($data, 4, count($data) - 1, true) ;
32 }

```

Listing 13: Additional data in the output

The above snippets use the Post model which can be found in the GitHub repository [24] with full source code.

5.3 Sample Response

The API returns an array of posts, and each post object contain following information: type, author, publish time, title, content, URL and image URL. Sample response can be seen in Appendix 7. Even though all this data is returned, the client doesn't need to use all of it.

5.4 Security

Faktabaari's API is secure, because basically clients using it can not request anything else than normal site visitors would request while visiting the site. As written before, the only supported method for requesting data is GET. APIs sometimes open vulnerabilities by providing sensitive information or by giving access to normally restricted actions [45]. While developing the API further, this needs to be kept in mind. Clients requesting same resource constantly (DDOS attack [46] for example) can not create interruptions in service, because responses are served from cache and if needed the service will scale up automatically. Also email alerts are in place for such behaviour. Scaling is explained later in this thesis work.

5.5 Performance

APIs performance is often measured in the response speed and uptime. The response speed is a key factor, because using 3rd party APIs on your own website should not significantly increase the page's load time. Faktabaari's API is being constantly monitored using service called Runscope [47]. If Runscope can not get response from the API, an email is sent to the admin. At the moment of writing this, the general API response time is about 500ms as seen in Appendix 8, which is still a bit high. The API is being developed further to fully take advantage of the Redis Cache, which is addressed later in this thesis work. After this update, the response speed is expected to be less than 100ms.

5.6 Demo

The demo for Faktabaari's API (Appendix 9) was developed in order to visualize the functionality for non-technical personnel. The demo includes fetching the latest fact, fetching specific fact by name, searching for facts using keywords, combining and visualizing data and general information. The demo's layout is very basic as it is not an important factor in showcasing the functionality. Demo is written using JSONP callback from the API and the pie chart is generated with JavaScript library called chart.js [48]. This demo is being developed with the API, and in the future is going to be used as documentation for the clients.

6 Cloud Hosting Solution

6.1 Microsoft Bizspark

At the beginning of the year 2015 Faktabaari got in program called BizSpark [49] organized by Microsoft. BizSpark helps startup companies to get started with technologies and software that would normally be expensive to use as the business is not generating any revenue yet. Microsoft states that "Your startup qualifies if it is less than 5 years old, is privately held, and earns less than \$1M annually." For Faktabaari this meant moving to a better, scalable cloud hosting.

6.2 Planning

Moving from previous shared hosting solution needed some planning, as some changes were needed also to the code itself in addition to moving the actual data. Faktabaari wanted to release the new version in the cloud, and the old version was hosted on the previous hosting provider. While planning the move, it was noted that the BizSpark sponsorship includes only those services in Azure that are provided by Microsoft, not from any

3rd party vendors. This meant moving from MySQL database to MsSQL database. It was decided that the moving would be done in the following steps:

1. Copying database from current installation to be used with the new version.
2. Developing the new version on the cloud, accessible by domain provided by Azure.
3. Editing DNS records to point to the new cloud hosting when the new version was ready to be published.

This way there would be no downtime at all, and editing DNS records was to be done overnight, so visitors would see the new site in the morning when Faktabaari was featured in the Finnish morning television show Huomenta Suomi [50].

6.3 Used Azure Services

Faktabaari is using multiple services in the Azure platform. The CMS is published as a Azure Web App, which is using continuous deployment from Bitbucket. The web app uses deployment slots so that Faktabaari can have a staging environment for testing new changes, before publishing them. The web app also has automatic backup on, so that if things go wrong, it is easy to restore the working environment.

As mentioned the MsSQL database is provided by the Azure SQL service. At this point Faktabaari is not using geo-replication, but might use it in the future. The database is running in the standard service tier with performance level S1. During development also faster tiers were tested, but they didn't provide visible performance boosts.

In addition to these Faktabaari is also using Azure blob storage (Azure service for storing data) for files, backups and cache storing. One blob storage can contain multiple containers. Files can be fetched from the containers using the Azure API or by browsing the containers in the Azure dashboard.

6.4 Implementing Caching

Creation of Redis Cache in Azure was done quickly, but using it with WordPress installation needed more work. In order to be able to use Redis with PHP one needs to install phpredis package [51]. Packages used with Azure need to be VC9 and non-thread-safe (nts) compatible. Once one has downloaded the zip, it needs to be extracted and the .dll file must be uploaded to the servers bin folder. Then one must configure Azure Websites app settings according to the figure 12 below:

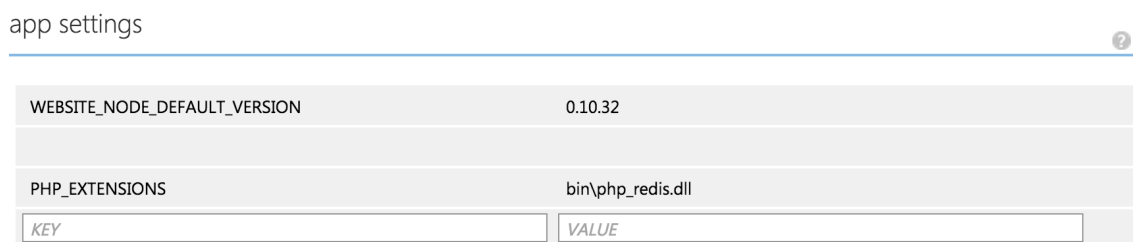


Figure 12: Azure Website's app settings under config tab

After these steps one should be able to use Redis classes in PHP code. Instead of coding a plugin for WordPress it was decided that it is better to add caching to the WordPress core itself. The index.php is modified to use index-wp-redis.php file. In this new index file the Redis client is first created as shown in listing 14:

```

1 //Init Redis client
2 if (class_exists('Redis')) {
3 $redis = new Redis();
4 $redis->connect('redis.cache.url.here', 6379);
5 $redis->auth(getenv("MyAzureRedisKey"));
6 }

```

Listing 14: Initialization of Redis

The URL used above is configured in Azure Redis settings with the port used. Authentication key is stored as server variable for security purposes. The full source code is shared in GitHub repository, but below is an overview of the most important parts. Listing 15 shows that serving cache files is done by using Redis hget function:

```

1 // Check if a cache of the page exists and exclude some pages
  with contact forms etc
2 if (!$loggedin && !$submit && !strpos($url, '/feed/') && !
  strpos($url, '/lomake/') && !strpos($url, '/ota-yhteytta/
  ') && !strpos($url, '/toimitusyhteisty/') && $redis->
  hexists($dkey, $ukey)) {

```



```

3      //Check if we are serving an API response
4      if (strpos($url, '/fapi/')) {
5          //Lets serve it but don't create any message on the
           json
6          echo $redis->hget($dkey, $ukey);
7          $cached = 1;
8      }
9      else {
10     echo $redis->hget($dkey, $ukey);
11     $cached = 1;
12     $msg = 'this is a cache';
13     }

```

Listing 15: Redis fetching

Messages mentioned in above code are used for debugging purposes. If the page is not cached, it will be cached by using PHP's output buffering. With PHP's output buffering one can instead of showing the output of code save it to a variable for example by using internal buffer [52]:

```

1  // Cache the page
2  else {
3      // Turn on output buffering
4      ob_start();
5      require('./wp-blog-header.php');
6      // Get contents of output buffer
7      $html = ob_get_contents();
8      // Clean output buffer
9      ob_end_clean();
10     echo $html;
11     if (strpos($url, '/fapi/')){
12         //Store and no message for api response
13         $redis->hset($dkey, $ukey, $html);
14     }
15     // Store to cache only if the page exist and is not a
           search result.
16     if (!is_404() && !is_search()) {
17         // store html contents to redis cache
18         $redis->hset($dkey, $ukey, $html);
19         $msg = 'cache is set';
20     }prof:wp

```

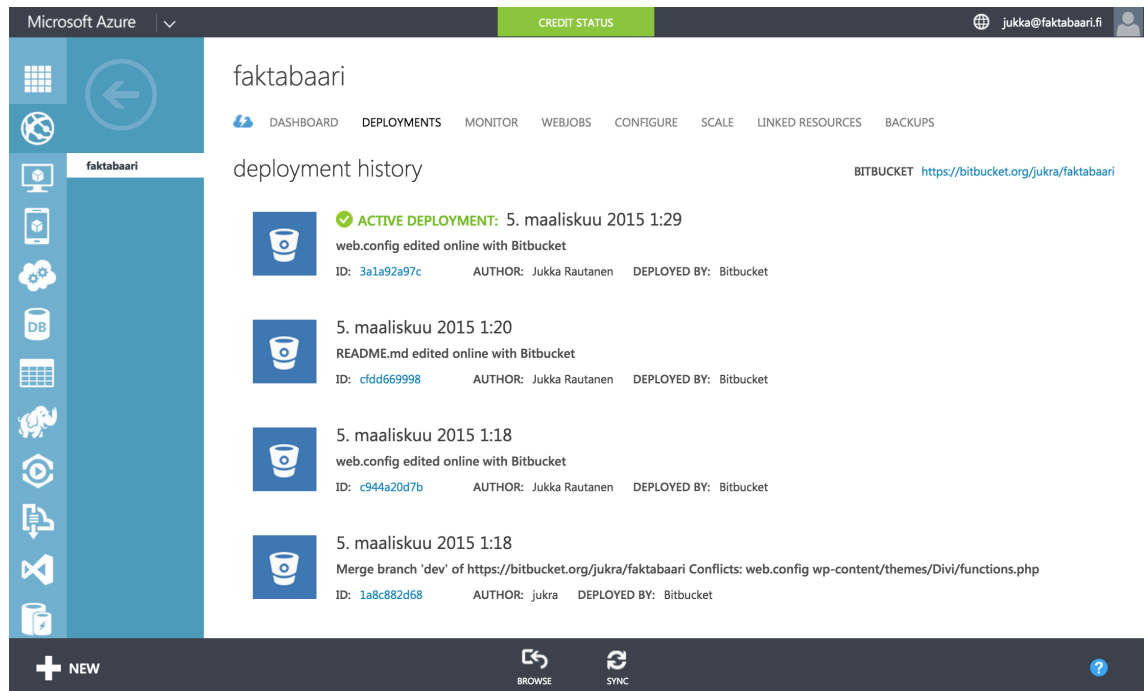
Listing 16: Redis caching

In addition to these functions one has the ability to clear the cache or specific page from it. Also handling states liked logged in users or new post submissions are implemented in the full source code.

Redis caching is a very powerful performance upgrade, for example Faktabaari's pages are served commonly between 0.3-0.8 seconds. Redis cache is relatively new caching system, which is getting more popular as it beats it's commonly used competitor memcached [53].

6.5 Continuous Deployment

Faktabaari has deployment set so, that every time code is pushed to the Bitbucket repository, the Azure website is updated to this newest deployment. One can also revert back to old deployments if needed. Figure 13 below shows the deployment history of Faktabaari's website:



The screenshot displays the Azure portal interface for the 'faktabaari' website. The top navigation bar shows 'Microsoft Azure' and 'CREDIT STATUS'. The left sidebar contains various service icons. The main content area is titled 'faktabaari' and includes a 'deployment history' section. The history lists four deployments, each with a Bitbucket icon, a timestamp, and details about the deployment source and author.

Deployment ID	Timestamp	Author	Deployment Method
3a1a92a97c	5. maaliskuu 2015 1:29	Jukka Rautanen	web.config edited online with Bitbucket
cfd669998	5. maaliskuu 2015 1:20	Jukka Rautanen	README.md edited online with Bitbucket
c944a20d7b	5. maaliskuu 2015 1:18	Jukka Rautanen	web.config edited online with Bitbucket
1a8c882d68	5. maaliskuu 2015 1:18	jukra	Merge branch 'dev' of https://bitbucket.org/jukra/faktabaari Conflicts: web.config wp-content/themes/Divi/functions.php

Figure 13: Azure Website's deployment

7 Conclusion

After releasing the new website Faktabaari has got positive feedback about it. As assumed traffic during the Finnish Parliamentary elections has been higher than before, thus Azure hosting solutions has been a good choice. During these three months Faktabaari.fi got over 250 000 visitors and there were no performance issues at any point.

Microsoft BizSpark credits have been sufficient for hosting and Redis so far. Azure web app has not have any considerable downtime since release and the hosting platform has worked flawlessly.

A few bugs were discovered in the crowd sourced editorial process as users used it in real-life scenarios with different operating systems and browsers. These bugs were fixed rather quickly and the form has helped Faktabaari's journalists in the editorial process.

Faktabaari won €30,000 of seed money in the Media innovation competition organized by the Finnish Ministry of Transport and Communications and award for Best Journalism Act in the 2014 Bonnier journalism awards organized in March, so the development of the site and platform continues. In the future Faktabaari might be going global by participating in other countries elections, and the coded platform needs to be localized and modified so that it can be shipped further.

Bibliography

- 1 Faktabaari.fi - In English [Online]. Faktabaari; 2015.
URL: <http://faktabaari.fi/in-english>.
Last accessed May 01, 2015.
- 2 What is a Content Management System (CMS)? [Online]. Bernard Kohan; 2010.
URL: <http://www.comentum.com/what-is-cms-content-management-system.html>.
Last accessed May 01, 2015.
- 3 Cloud CMS [Online]. Gitana Software, Inc; 2015.
URL: <https://www.cloudcms.com/>.
Last accessed May 01, 2015.
- 4 Most Notable Big Name Brands that are Using WordPress [Online]. WPBeginner LLC; 2014.
URL: <http://www.wpbeginner.com/showcase/40-most-notable-big-name-brands-that-are-using-wordpress/>.
Last accessed May 01, 2015.
- 5 CMS Usage Statistics [Online]. BuiltWith® Pty Ltd; 2015.
URL: <http://trends.builtwith.com/cms>.
Last accessed May 01, 2015.
- 6 WordPress [Online]. WordPress Foundation; 2015.
URL: <http://wordpress.org>.
Last accessed May 01, 2015.
- 7 Drupal [Online]. Dries Buytaert; 2015.
URL: <http://drupal.org>.
Last accessed May 01, 2015.
- 8 EPiServer [Online]. EPiServer; 2015.
URL: <http://www.episerver.com/>.
Last accessed May 01, 2015.
- 9 Joomla [Online]. The Joomla Project Team; 2015.
URL: <http://joomla.org>.
Last accessed May 01, 2015.
- 10 ModX [Online]. Raymond Irving, Ryan Thrash; 2015.
URL: <http://modx.com>.
Last accessed May 01, 2015.

- 11 Concrete5 [Online]. Concrete5; 2015.
URL: <http://concrete5.org>.
Last accessed May 01, 2015.
- 12 Microsoft Sharepoint [Online]. Microsoft Corporation; 2015.
URL: <http://sharepoint.com>.
Last accessed May 01, 2015.
- 13 What is cloud? [Online]. IBM; 2015.
URL: <http://www.ibm.com/cloud-computing/us/en/what-is-cloud-computing.html>.
Last accessed Apr 7, 2015.
- 14 Microsoft Azure [Online]. Microsoft; 2015.
URL: <http://azure.microsoft.com/fi-fi/overview/what-is-azure/>.
Last accessed Apr 12, 2015.
- 15 Microsoft Azure vs. Amazon Web Services: Cloud Comparison [Online].
William Van Winkle; 2015.
URL: <http://www.tomsitpro.com/articles/azure-vs-aws-cloud-comparison,2-870.html>.
Last accessed Apr 12, 2015.
- 16 Web Apps overview - Azure [Online]. Microsoft; 2015.
URL: <http://azure.microsoft.com/en-us/documentation/articles/app-service-web-overview/>.
Last accessed Apr 12, 2015.
- 17 Set up staging environments for web apps in Azure App Service [Online].
Microsoft; 2015.
URL: <http://azure.microsoft.com/en-us/documentation/articles/web-sites-staged-publishing/>.
Last accessed Apr 12, 2015.
- 18 Deploy a web app in Azure App Service [Online]. Microsoft; 2015.
URL: <http://azure.microsoft.com/fi-fi/documentation/articles/web-sites-deploy/>.
Last accessed Apr 12, 2015.
- 19 Azure SQL Database [Online]. Microsoft; 2015.
URL: <https://msdn.microsoft.com/en-us/library/azure/ee336279.aspx>.
Last accessed Apr 12, 2015.
- 20 Active Geo-Replication for Azure SQL Database [Online]. Microsoft; 2015.
URL: <https://msdn.microsoft.com/en-us/library/azure/dn741339.aspx>.
Last accessed Apr 12, 2015.
- 21 Redis [Online]. Salvatore Sanfilippo; 2015.
URL: <http://redis.io/>.
Last accessed Apr 12, 2015.
- 22 How to Use Azure Redis Cache [Online]. Microsoft; 2015.
URL:

- <http://azure.microsoft.com/fi-fi/documentation/articles/cache-dotnet-how-to-use-azure-redis-cache/>.
Last accessed Apr 12, 2015.
- 23 Creative Commons Attribution License [Online]. creativecommons.org; 2015.
URL: <http://creativecommons.org/licenses/by/4.0/>.
Last accessed May 01, 2015.
- 24 Faktabaari API plugin [Online]. Jukka Rautanen; 2015.
URL: <https://github.com/jukra/faktabaari-api/>.
Last accessed May 01, 2015.
- 25 The Scrum Guide [Online]. Ken Schwaber and Jeff Sutherland; 2013.
URL: <http://www.scrumguides.org/scrum-guide.html>.
Last accessed May 01, 2015.
- 26 WordPress challenges Drupal in media sites [Online]. Perttu Tolvanen; 2013.
URL: <http://northpatrol.com/2013/10/31/wordpress-challenges-drupal-in-media-sites/>.
Last accessed May 01, 2015.
- 27 Project Nami [Online]. Team Project Nami; 2015.
URL: <http://projectnami.org>.
Last accessed May 01, 2015.
- 28 About Wordpress [Online]. Wordpress.org; 2015.
URL: <https://wordpress.org/about/>.
Last accessed May 01, 2015.
- 29 Brad Williams D, Hal Stern. Professional Wordpress 2nd edition. Wrox; 2013.
- 30 Wordpress.org - Theme Development [Online]. Wordpress.org; 2015.
URL: http://codex.wordpress.org/Theme_Development.
Last accessed May 03, 2015.
- 31 Google Developers - Responsive Web Design [Online]. Google; 2015.
URL: <https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/>.
Last accessed May 03, 2015.
- 32 Masonry JavaScript grid layout library [Online]. David DeSandro; 2015.
URL: <http://masonry.desandro.com/>.
Last accessed May 03, 2015.
- 33 Types Wordpress Plugin [Online]. Multiple authors; 2015.
URL: <http://wordpress.org/plugins/types/>.
Last accessed May 07, 2015.
- 34 User Submitted Posts Wordpress Plugin [Online]. Jeff Star; 2015.
URL: <http://perishablepress.com/user-submitted-posts/>.
Last accessed May 02, 2015.

- 35 Contact Form 7 Wordpress Plugin [Online]. Takayuki Miyoshi; 2015.
URL: <http://wordpress.org/plugins/contact-form-7/>.
Last accessed May 04, 2015.
- 36 WP Smush.it [Online]. WPMU DEV; 2015.
URL: <https://wordpress.org/plugins/wp-smushit/>.
Last accessed May 08, 2015.
- 37 Google Developers - Image optimization [Online]. Google; 2015.
URL: <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization>.
Last accessed May 08, 2015.
- 38 Google Analytics [Online]. Google; 2015.
URL: <http://www.google.com/analytics>.
Last accessed May 07, 2015.
- 39 Jetpack Wordpress Plugin [Online]. Multiple authors; 2015.
URL: <http://wordpress.org/plugins/jetpack/>.
Last accessed May 07, 2015.
- 40 Hardening Wordpress [Online]. wordpress.org; 2015.
URL: http://codex.wordpress.org/Hardening_WordPress.
Last accessed May 02, 2015.
- 41 WordPress feeds [Online]. wordpress.org; 2015.
URL: http://codex.wordpress.org/WordPress_Feeds.
Last accessed May 08, 2015.
- 42 JSON [Online]. json.org; 2015.
URL: <http://json.org>.
Last accessed May 08, 2015.
- 43 Same-origin policy [Online]. Mozilla Developer Network; 2015.
URL: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy.
Last accessed May 08, 2015.
- 44 Thermal: A RESTful API for WordPress [Online]. VocePlatforms; 2015.
URL: <http://thermal-api.com>.
Last accessed May 08, 2015.
- 45 APIs – The Next Hacker Target Or a Business and Security Opportunity? [Online]. Tim Mather; 2015.
URL: http://www.rsaconference.com/writable/presentations/file_upload/sec-t07-apis-the-next-hacker-target-or-a-business-and-security-opportunity.pdf.
Last accessed May 08, 2015.
- 46 What is a DDoS Attack? [Online]. Arbor Networks, Inc; 2015.
URL: <http://www.digitalattackmap.com/understanding-ddos/>.
Last accessed May 08, 2015.

- 47 Runscope [Online]. Runscope Inc.; 2015.
URL: <https://www.runscope.com>.
Last accessed May 08, 2015.
- 48 Chart.js [Online]. Nick Downie; 2015.
URL: <https://www.chartjs.org>.
Last accessed May 08, 2015.
- 49 BizSpark [Online]. Microsoft; 2015.
URL: <http://www.microsoft.com>.
Last accessed Apr 7, 2015.
- 50 MTV Katsomo - Faktabaari [Online]. MTV3; 2015.
URL: <http://www.katsomo.fi/#!/jakso/445482?toista>.
Last accessed Apr 7, 2015.
- 51 PECL Snaps for Redis [Online]. php_redis; 2015.
URL: <http://windows.php.net/downloads/pecl/snaps/redis/2.2.5/>.
Last accessed Apr 12, 2015.
- 52 ob_start at PHP documentation [Online]. The PHP Group; 2015.
URL: <http://php.net/manual/en/function.ob-start.php>.
Last accessed Apr 12, 2015.
- 53 How fast is Redis [Online]. Salvatore Sanfilippo; 2015.
URL: <http://redis.io/topics/benchmarks>.
Last accessed Apr 12, 2015.

1 Faktabaari.fi - 3 main layouts

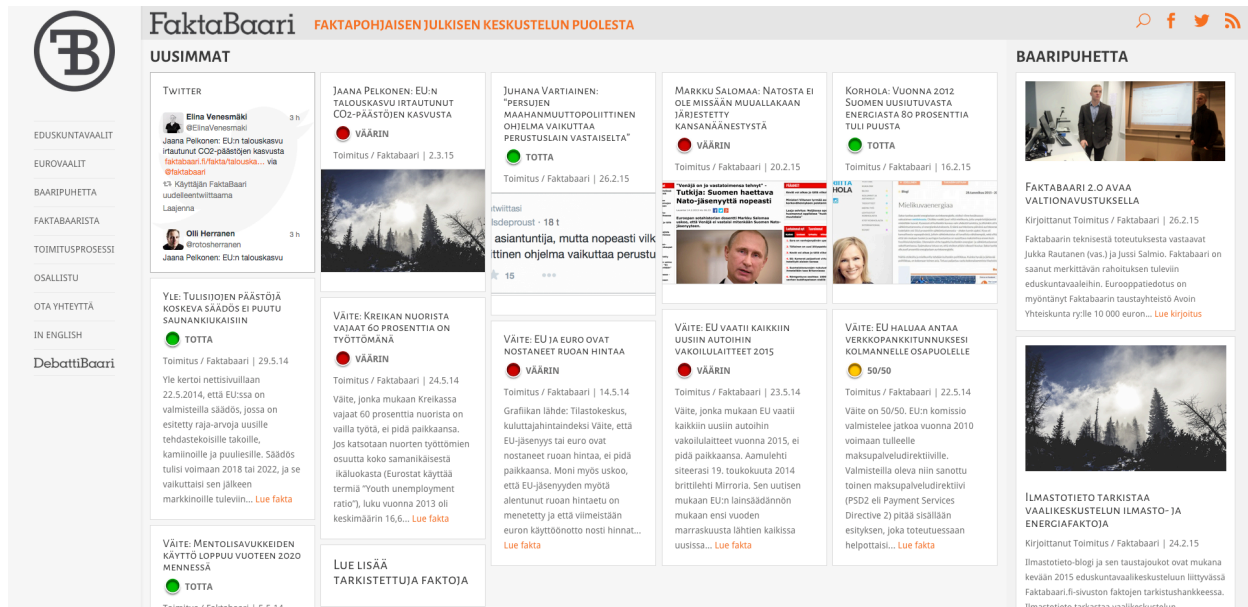


Figure 14: Frontpage



Figure 15: Article layout

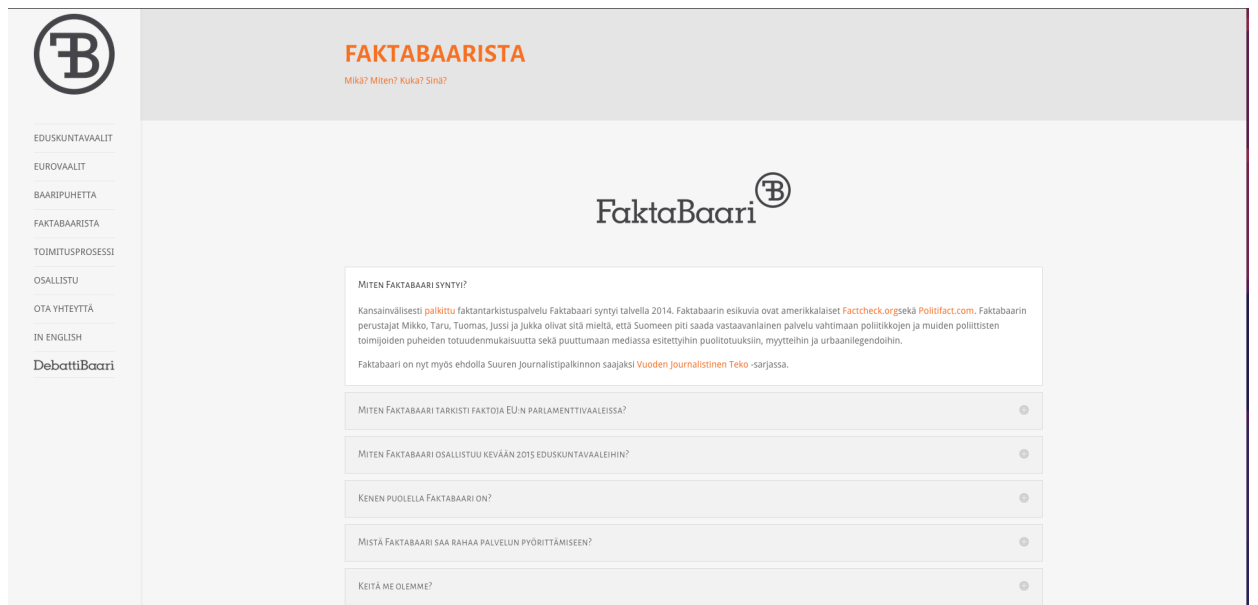


Figure 16: Page layout

2 Function generating latest posts in frontpage template

This is part the of the fronpage template which is generating latests posts in frontpage.

```
1 <?php
2 function get_latest_posts( $atts ) {
3     //Input arguments from the shortcode
4     extract( shortcode_atts( array(
5         'module_id' => '',
6         'module_class' => '',
7         'fullwidth' => 'on',
8         'posts_number' => 10,
9         'include_categories' => '',
10        'meta_date' => 'M j, Y',
11        'show_thumbnail' => 'on',
12        'show_content' => 'off',
13        'show_author' => 'on',
14        'show_date' => 'on',
15        'show_categories' => 'on',
16        'show_pagination' => 'on',
17        'background_layout' => 'light',
18        'show_more' => 'off',
19    ), $atts
20    ) );
21    //Global variables
22    global $paged;
23    $container_is_closed = false;
24    if ( 'on' !== $fullwidth ){
25        wp_enqueue_script( 'jquery-masonry-3' );
26    }
27    $args = array( 'posts_per_page' => (int) $posts_number, '
                post_type' => array('eu', 'fakta') );
28    $et_paged = is_front_page() ? get_query_var( 'page' ) :
                get_query_var( 'paged' );
29    if ( is_front_page() ) {
30        $paged = $et_paged;
31    }
32    if ( '' !== $include_categories )
33        $args['cat'] = $include_categories;
34    if ( ! is_search() ) {
35        $args['paged'] = $et_paged;
36    }
37    ob_start();
38    //Check if frontpage, if it is, then generate Twitter feed
```

```
39     if (is_front_page()){
40         echo "<article class='et_pb_post et_pb_no_thumb post
            type-post status-publish format-standard et_pb_post
            ' style='height: 269px; border: 1px solid #A3A3A3;
            background: #fff url(/wp-content/themes/Divi/images
            /twitter.png) 50% 50% no-repeat;'><h2>Twitter</h2><
            a data-chrome='nofooter noborders transparent
            noheader' class='twitter-timeline' width='300'
            height='250' href='http://twitter.com/Faktabaari'
            data-widget-id='442628258274623488'>Twititejä
            käyttäjältä @Faktabaari</a><script>!function(d,s,id
            ){var js,fjs=d.getElementsByTagName(s)[0],p=/^http
            :/.test(d.location)?'http':'https';if(!d.
            getElementById(id)){js=d.createElement(s);js.id=id;
            js.src=p+'://platform.twitter.com/widgets.js';fjs.
            parentNode.insertBefore(js,fjs);}}(document,'script
            ','twitter-wjs');</script></article>";
41     }
42     //Start the post query
43     query_posts( $args );
44     if ( have_posts() ) {
45         //Post loop
46         while ( have_posts() ) {
47             the_post();
48             $post_format = get_post_format();
49             $thumb = '';
50             $width = 'on' === $fullwidth ? 1080 : 400;
51             $width = (int) apply_filters( 'et_pb_blog_image_width'
                , $width );
52             $height = 'on' === $fullwidth ? 675 : 250;
53             $height = (int) apply_filters( '
                et_pb_blog_image_height', $height );
54             $classtext = 'on' === $fullwidth ? '
                et_pb_post_main_image' : '';
55             $titletext = get_the_title();
56             $thumbnail = get_thumbnail( $width, $height,
                $classtext, $titletext, $titletext, false, '
                Blogimage' );
57             $thumb = $thumbnail["thumb"];
58             $no_thumb_class = '' === $thumb || 'off' ===
                $show_thumbnail ? ' et_pb_no_thumb' : '';
59             if ( in_array( $post_format, array( 'video', 'gallery'
                ) ) ) {
60                 $no_thumb_class = '';
61             } ?>
62             //Start output of the post
63             <article id="post-<?php the_ID(); ?>" <?php post_class(
                'et_pb_post' . $no_thumb_class ); ?>>
64             <?php
65                 et_divi_post_format_content();
```

```
66     if ( ! in_array( $post_format, array( 'link', 'audio',
67         'quote' ) ) ) {
68         if ( 'video' === $post_format && false !== (
69             $first_video = et_get_first_video() ) ) :
68             printf(
69                 '<div class="et_main_video_container">
70                 %1$s
71                 </div>',
72                 $first_video
73             );
74         elseif ( 'gallery' === $post_format ) :
75             et_gallery_images();
76         elseif ( '' !== $thumb && 'on' === $show_thumbnail )
77             :
78             if ( 'on' !== $fullwidth ) echo '<div class="
79                 et_pb_image_container">';
80             print_thumbnail( $thumb, $thumbnail["use_timthumb"],
81                 $titletext, $width, $height );
82             if ( 'on' !== $fullwidth ) echo '</div> <!-- .
83                 et_pb_image_container -->';
84         endif;
85     }
86     if ( 'off' === $fullwidth || ! in_array( $post_format,
87         array( 'link', 'audio', 'quote', 'gallery' ) ) ) {
88     if ( ! in_array( $post_format, array( 'link', 'audio'
89         ) ) ) { ?>
90     <h2><a href="<?php the_permalink(); ?>"><?php
91         the_title(); ?></a></h2>
92     <?php }
93         //Get the review of checked fact
94         $value = get_post_meta( get_the_ID(), 'wpcf-
95             review', true );
96         // check if the custom field has a value
97         if( ! empty( $value ) ) {
98             if($value=="Totta"){
99                 echo "<img class='reviewpic' src='wp-
100                     content/themes/Divi/images/totta.png
101                     '>";
102                 echo "<span class='review'>Totta</span>"
103                     ;
104             }
105             if($value=="50/50"){
106                 echo "<img class='reviewpic' src='wp-
107                     content/themes/Divi/images/5050.png'>
108                     ";
109                 echo "<span class='review'>50/50</span>"
110                     ;
111             }
112             if($value=="Väärin"){
```

```
99         echo "<img class='reviewpic' src='wp-
           content/themes/Divi/images/vaarin.png
           '>";
100         echo "<span class='review'>Väärin</span>
           ";
101     }
102 }
103 //Author and date according to settings
104 if ( 'on' === $show_author || 'on' === $show_date ||
      'on' === $show_categories ) {
105     printf( '<p class="post-meta">%1$s %2$s %3$s %4$s
           %5$s</p>',
106         (
107             'on' === $show_author
108             ? sprintf( __( '%s', 'Divi' ),
109                 et_get_the_author_posts_link() )
110             : ''
111         ),
112         (
113             'on' === $show_author && 'on' === $show_date
114             ? ' | '
115             : ''
116         ),
117         'on' === $show_date
118         ? sprintf( __( '%s', 'Divi' ), get_the_date(
119             $meta_date ) )
120         : ''
121     ),
122     (( 'on' === $show_author || 'on' ===
        $show_date ) && 'on' === $show_categories)
123     ? ' | '
124     : ''
125 ),
126 (
127     'on' === $show_categories
128     ? get_the_category_list(', ')
129     : ''
130 )
131 );
132 }
133 //Post content, showing thumbnail if
           featured image exists
134 if ( 'on' === $show_content ) {
135     global $more;
136     $more = null;
137     the_content( __( 'read more...', 'Divi' ) );
138 } else {
```

```
139         if ( has_excerpt() ) {
140             if ( '' == $thumb){
141                 the_excerpt();
142             }
143         } else {
144             if ( '' == $thumb){
145                 truncate_post( 270 );
146             }
147         }
148         printf( ' <a href="%1$s" class="more-link" >%2$s</
                a>' , esc_url( get_permalink() ), __( 'Lue
                fakta', 'Divi' ) );
149         echo $more;
150     } ?>
151     <?php if ( '' != $thumb){?>
152     <div class="et_pb_image_container">
153     <?php printf( ' <a href="%1$s" class="more-link" >' ,
                esc_url( get_permalink() ), __( 'Lue fakta', 'Divi'
                ) ); ?>
154     <?php print_thumbnail( $thumb, $thumbnail["
                use_timthumb"], $titletext, $width, $height ); ?></
                a>
155     </div>
156     <?php }
157 }
158 </article> <!-- .et_pb_post -->
159     <?php
160 } // endwhile
161     //Check if we need pagination
162 if ( 'on' === $show_pagination && ! is_search() ) {
163     echo '</div> <!-- .et_pb_posts -->';
164     $container_is_closed = true;
165     if ( function_exists( 'wp_pagenavi' ) )
166         wp_pagenavi();
167     else
168         get_template_part( 'includes/navigation', 'index' );
169 }
170     //Stop query
171     wp_reset_query();
172 } else {
173     get_template_part( 'includes/no-results', 'index' );
174 }
175 $posts = ob_get_contents();
176 ob_end_clean();
177 $class = " et_pb_bg_layout_{$background_layout}";
178     //Editing output so that we have one Read More box at
        the end
179 $output = sprintf(
180     '<div%5$s class="%1$s%3$s%6$s">
181     %2$s
```

```
182     %4$s',
183     ( 'on' === $fullwidth ? 'et_pb_posts' : 'et_pb_blog_grid
        clearfix' ),
184     $posts,
185     esc_attr( $class ),
186     ( ! $container_is_closed ? '<article class=" post type-
        post status-publish format-standard et_pb_post">
187 <a class="moar" href="/faktat"><h1>Lue lisää tarkistettuja
        faktoja</h1></a>
188 </article></div><br><br> <!-- .et_pb_posts -->' : '' ),
189     ( '' !== $module_id ? sprintf( ' id="%1$s"', esc_attr(
        $module_id ) ) : '' ),
190     ( '' !== $module_class ? sprintf( ' %1$s', esc_attr(
        $module_class ) ) : '' )
191 );
192
193 if ( 'on' !== $fullwidth )
194     $output = sprintf( '<div class="et_pb_blog_grid_wrapper
        ">%1$s</div>', $output );
195     //Return the output for later use
196     return $output;
197 }
198 ?>
```

Listing 17: Part of functions.php

3 Function sending email on new drafts

```
1 <?php
2 add_action('save_post', 'er_send_email_on_post_draft_save' )
3 ;
4 function er_send_email_on_post_draft_save( $post_id ) {
5     global $post;
6     //Verify post is a draft
7     if ( $post->post_status == 'draft' ) {
8         $post_title = get_the_title( $post_id );
9         $post_url = get_permalink( $post_id );
10        $post_type = get_post_type( $post_id );
11        $author_id = get_post_field( 'post_author', $post_id
12        );
13        $author_details = get_user_by( 'id', $author_id );
14        $author_name = $author_details->first_name . ' ' .
15        $author_details->last_name;
16        //Generate a nicer name for post type to show in
17        //email
18        $post_type_nice = "";
19        if ($post_type=="fakta") {
20            $post_type_nice = "fakta";
21        }
22        if ($post_type=="eu-fakta") {
23            $post_type_nice = "EU-fakta";
24        }
25        if ($post_type=="post") {
26            $post_type_nice = "blogikirjoitus";
27        }
28        //Message body
29        $subject = 'Uusi luonnos FaktaBaarissa';
30        $message = "Uusi " . $post_type_nice . " FaktaBaarissa
31        :\n\n";
32        $message .= "Otsikko: " . $post_title . "\n\n";
33        $message .= "Kirjoittanut: " . $author_name . "\n\n";
34        $message .= "" . $post_url . "\n\n";
35        //Send email to staff
36        //We don't want revisions
37        if($post_type!="revision"){
38            wp_mail( 'toimitus@faktabaari.fi', $subject,
39            $message );
40        }
41    }
42 }
43 ?>
```

Listing 18: Part of functions.php

4 Function submitting posts to database

```
1 <?php
2
3 function usp_createPublicSubmission($title, $content,
4     $authorName, $authorEmail, $authorID, $authorUrl, $tags,
5     $review, $atc, $category, $fileData)
6 {
7     global $usp_options, $usp_post_meta_IsSubmission,
8         $usp_post_meta_SubmitterIp, $usp_post_meta_Submitter,
9         $usp_post_meta_SubmitterUrl, $usp_post_meta_Image;
10    $authorName = strip_tags($authorName);
11    $authorUrl = strip_tags($authorUrl);
12    $atc = strip_tags($atc);
13    $publishName = "";
14    if (isset($_SERVER['REMOTE_ADDR'])) $authorIp =
15        sanitize_text_field($_SERVER['REMOTE_ADDR']);
16    if (isset($_POST['user-submitted-captcha'])) $captcha =
17        sanitize_text_field($_POST['user-submitted-captcha']);
18    if (isset($_POST['user-submitted-verify'])) $verify =
19        sanitize_text_field($_POST['user-submitted-verify']);
20    $newPost = array(
21        false,
22        null
23    );
24    if (!usp_validateTitle($title)) return 'title';
25    if (!usp_validateTags($tags)) return 'tags';
26    if (!empty($verify)) return 'verify';
27    if ($usp_options['usp_captcha'] == 'show')
28    {
29        if (!usp_spam_question($captcha)) return 'captcha';
30    }
31
32    $postData = array();
33    $postData['post_title'] = $title;
34    $postData['post_content'] = $content;
35    $postData['post_status'] = 'pending';
36
37    // Let's check who is writing
38
39    $targets = array(
40        'ilmasto',
41        'ilmastotieto',
42        'ilmasto',
43        'Ilmasto / Faktabaari'
44    );
45    foreach($targets as $t)
```

```
39     {
40     if (strpos($authorName, $t) !== false)
41         {
42
43         // Okay it's one these guys
44
45         $postData['post_author'] = 8;
46         $publishName = "Ilmasto / Faktabaari";
47         break;
48         }
49
50     // Or it's someone else
51
52     else
53     {
54         $postData['post_author'] = 4;
55         $publishName = "Toimitus / Faktabaari";
56     }
57 }
58
59 $postData['post_type'] = 'fakta';
60 $numberApproved = $usp_options['number-approved'];
61 if ($numberApproved < 0)
62     {
63     }
64 elseif ($numberApproved == 0)
65     {
66     $postData['post_status'] = 'publish';
67     }
68 else
69     {
70     $posts = get_posts(array(
71         'post_status' => 'publish',
72         'meta_key' => $usp_post_meta_Submitter,
73         'meta_value' => $authorName
74     ));
75     $counter = 0;
76     foreach($posts as $post)
77         {
78         $submitterUrl = get_post_meta($post->ID,
79             $usp_post_meta_SubmitterUrl, true);
80         $submitterIp = get_post_meta($post->ID,
81             $usp_post_meta_SubmitterIp, true);
82         if ($submitterUrl == $authorUrl && $submitterIp ==
83             $authorIp)
84             {
85                 $counter++;
```

```
86     if ($counter >= $numberApproved) $postData['post_status'  
87         ] = 'publish';  
88     }  
89     do_action('usp_insert_before', $postData);  
90     $newPost = wp_insert_post($postData);  
91     do_action('usp_insert_after', $newPost);  
92     if ($newPost)  
93     {  
94         wp_set_post_tags($newPost, $tags);  
95         add_post_meta($newPost, 'wpcf-review', $review, true);  
96         add_post_meta($newPost, 'Liitteen osoite', $atc, true);  
97         add_post_meta($newPost, 'Kirjoittajan yhteystiedot',  
98             $authorEmail, true);  
99         wp_set_post_categories($newPost, array(  
100             $category  
101         ));  
102         $post_title = get_the_title($newPost);  
103         $post_url = "http://faktabaari.fi/wp-admin/edit.php?  
104             post_type=fakta";  
105         if ($usp_options['usp_email_alerts'] == true)  
106         {  
107             $to = $usp_options['usp_email_address'];  
108             if ($to != '')  
109             {  
110                 $subject = 'Uusi kirjoitus vastaanotettu  
111                     toimitusyhteistyön kautta!';  
112                 $message = "Hei Faktabaarin ahkera toimitus! Uusi  
113                     kirjoitus odottaa nyt tuomiotanne WordPressin  
114                     Faktat-osiossa. Muistakaahan katsoa myös se  
115                     Liitteen osoite - kenttä, jos kirjoittaja haluaa  
116                     julkaista myös kuvan.\n\n";  
117                 $message.= "Otsikko: " . $post_title . "\n\n";  
118                 $message.= "Oikotie: " . $post_url . "\n\n";  
119                 $subject = apply_filters('usp_mail_subject',  
120                     $subject);  
121                 $message = apply_filters('usp_mail_message',  
122                     $message);  
123                 wp_mail($to, $subject, $message);  
124                 add_post_meta($newPost, 'notificationSend', 'true',  
125                     true);  
126             }  
127         }  
128     }  
129     update_post_meta($newPost, $usp_post_meta_IsSubmission,  
130         true);  
131     update_post_meta($newPost, $usp_post_meta_Submitter,  
132         sanitize_text_field($publishName));  
133     update_post_meta($newPost, $usp_post_meta_SubmitterUrl,  
134         sanitize_text_field($authorUrl));
```

```
122     update_post_meta($newPost, $usp_post_meta_SubmitterIp,  
123         sanitize_text_field($authorIp));  
124     }  
125     return apply_filters('usp_new_post', $newPost);  
126 }  
127  
128 ?>
```

Listing 19: Part of the modified user submitted posts plugin

5 Faktabaari.fi - Tip form

Kuka väitti? (Esimerkiksi ehdokkaan nimi)*

Missä väitti? (Www-linkki lähteeseen)*

Mitä väitti? Missä virhe?*

Hyödyllistä taustatietoa Faktabaarilaisille

Korjausehdotus faktaan:

Yhteystietosi Faktabaarin toimituksen käyttöön

Lähetä

Figure 17: Tip form (faktabaari.fi/lomake)

6 Security rules

```
1 <rule name="Abuse Agent Blocking from HackRepair.com"
  stopProcessing="true">
2 <match url="^.*" ignoreCase="false" />
3 <conditions logicalGrouping="MatchAny">
4 <!--# Abuse Agent Blocking-->
5 <add input="{HTTP_USER_AGENT}" pattern="^BlackWidow" />
6 <add input="{HTTP_USER_AGENT}" pattern="^Bolt\ 0" />
7 <add input="{HTTP_USER_AGENT}" pattern="^Bot\ mailto:
  craftbot\@yahoo\.com" />
8 <add input="{HTTP_USER_AGENT}" pattern="CazoodleBot" />
9 <add input="{HTTP_USER_AGENT}" pattern="^ChinaClaw" />
10 <add input="{HTTP_USER_AGENT}" pattern="^Custo" />
11 <add input="{HTTP_USER_AGENT}" pattern="^Default\
  Browser\ 0" />
12 <add input="{HTTP_USER_AGENT}" pattern="^DIiBot" />
13 <add input="{HTTP_USER_AGENT}" pattern="^DISCo" />
14 <add input="{HTTP_USER_AGENT}" pattern="discobot" />
15 <add input="{HTTP_USER_AGENT}" pattern="^Download\ Demon
  " />
16 <add input="{HTTP_USER_AGENT}" pattern="^eCatch" />
17 <add input="{HTTP_USER_AGENT}" pattern="ecxi" />
18 <add input="{HTTP_USER_AGENT}" pattern="^EirGrabber" />
19 <add input="{HTTP_USER_AGENT}" pattern="^EmailCollector"
  />
20 <add input="{HTTP_USER_AGENT}" pattern="^EmailSiphon" />
21 <add input="{HTTP_USER_AGENT}" pattern="^EmailWolf" />
22 <add input="{HTTP_USER_AGENT}" pattern="^Express\
  WebPictures" />
23 <add input="{HTTP_USER_AGENT}" pattern="^ExtractorPro"
  />
24 <add input="{HTTP_USER_AGENT}" pattern="^EyeNetIE" />
25 <add input="{HTTP_USER_AGENT}" pattern="^FlashGet" />
26 <add input="{HTTP_USER_AGENT}" pattern="^GetRight" />
27 <add input="{HTTP_USER_AGENT}" pattern="^GetWeb!" />
28 <add input="{HTTP_USER_AGENT}" pattern="^Go!Zilla" />
29 <add input="{HTTP_USER_AGENT}" pattern="^Go-Ahead-Got-It
  " />
30 <add input="{HTTP_USER_AGENT}" pattern="^GrabNet" />
31 <add input="{HTTP_USER_AGENT}" pattern="^Grafula" />
32 <add input="{HTTP_USER_AGENT}" pattern="GT::WWW" />
33 <add input="{HTTP_USER_AGENT}" pattern="heritrix" />
34 <add input="{HTTP_USER_AGENT}" pattern="^HMView" />
35 <add input="{HTTP_USER_AGENT}" pattern="HTTP::Lite" />
36 <add input="{HTTP_USER_AGENT}" pattern="HTTrack" />
37 <add input="{HTTP_USER_AGENT}" pattern="ia_archiver" />
```

```
38 <add input="{HTTP_USER_AGENT}" pattern="IDBot" />
39 <add input="{HTTP_USER_AGENT}" pattern="id-search" />
40 <add input="{HTTP_USER_AGENT}" pattern="id-search\.org"
   />
41 <add input="{HTTP_USER_AGENT}" pattern="^Image\ Stripper
   " />
42 <add input="{HTTP_USER_AGENT}" pattern="^Image\ Sucker"
   />
43 <add input="{HTTP_USER_AGENT}" pattern="Indy\ Library"
   />
44 <add input="{HTTP_USER_AGENT}" pattern="^InterGET" />
45 <add input="{HTTP_USER_AGENT}" pattern="^Internet\ Ninja
   " />
46 <add input="{HTTP_USER_AGENT}" pattern="^InternetSeer\.
   com" />
47 <add input="{HTTP_USER_AGENT}" pattern="IRLbot" />
48 <add input="{HTTP_USER_AGENT}" pattern="ISC\ Systems\
   iRc\ Search\ 2\.1" />
49 <add input="{HTTP_USER_AGENT}" pattern="^Java" />
50 <add input="{HTTP_USER_AGENT}" pattern="^JetCar" />
51 <add input="{HTTP_USER_AGENT}" pattern="^JOC\ Web\
   Spider" />
52 <add input="{HTTP_USER_AGENT}" pattern="^larbin" />
53 <add input="{HTTP_USER_AGENT}" pattern="^LeechFTP" />
54 <add input="{HTTP_USER_AGENT}" pattern="libwww" />
55 <add input="{HTTP_USER_AGENT}" pattern="libwww-perl" />
56 <add input="{HTTP_USER_AGENT}" pattern="^Link" />
57 <add input="{HTTP_USER_AGENT}" pattern="LinksManager.
   com_bot" />
58 <add input="{HTTP_USER_AGENT}" pattern="linkwalker" />
59 <add input="{HTTP_USER_AGENT}" pattern="lwp-trivial" />
60 <add input="{HTTP_USER_AGENT}" pattern="^Mass\
   Downloader" />
61 <add input="{HTTP_USER_AGENT}" pattern="^Maxthon$" />
62 <add input="{HTTP_USER_AGENT}" pattern="MFC_Tear_Sample"
   />
63 <add input="{HTTP_USER_AGENT}" pattern="^microsoft\.url"
   />
64 <add input="{HTTP_USER_AGENT}" pattern="Microsoft\ URL\
   Control" />
65 <add input="{HTTP_USER_AGENT}" pattern="^MIDown\ tool"
   />
66 <add input="{HTTP_USER_AGENT}" pattern="^Mister\ PiX" />
67 <add input="{HTTP_USER_AGENT}" pattern="Missigua\
   Locator" />
68 <add input="{HTTP_USER_AGENT}" pattern="^Mozilla\. *Indy"
   />
69 <add input="{HTTP_USER_AGENT}" pattern="^Mozilla\. *NEWT"
   />
70 <add input="{HTTP_USER_AGENT}" pattern="^MSFrontPage" />
```



```

71 <add input="{HTTP_USER_AGENT}" pattern="^Navroad" />
72 <add input="{HTTP_USER_AGENT}" pattern="^NearSite" />
73 <add input="{HTTP_USER_AGENT}" pattern="^NetAnts" />
74 <add input="{HTTP_USER_AGENT}" pattern="^NetSpider" />
75 <add input="{HTTP_USER_AGENT}" pattern="^Net\ Vampire"
    />
76 <add input="{HTTP_USER_AGENT}" pattern="^NetZIP" />
77 <add input="{HTTP_USER_AGENT}" pattern="^Nutch" />
78 <add input="{HTTP_USER_AGENT}" pattern="^Octopus" />
79 <add input="{HTTP_USER_AGENT}" pattern="^Offline\
    Explorer" />
80 <add input="{HTTP_USER_AGENT}" pattern="^Offline\
    Navigator" />
81 <add input="{HTTP_USER_AGENT}" pattern="^PageGrabber" />
82 <add input="{HTTP_USER_AGENT}" pattern="panscient.com"
    />
83 <add input="{HTTP_USER_AGENT}" pattern="^Papa\ Foto" />
84 <add input="{HTTP_USER_AGENT}" pattern="^pavuk" />
85 <add input="{HTTP_USER_AGENT}" pattern="PECL::HTTP" />
86 <add input="{HTTP_USER_AGENT}" pattern="^PeoplePal" />
87 <add input="{HTTP_USER_AGENT}" pattern="^pcBrowser" />
88 <add input="{HTTP_USER_AGENT}" pattern="PHPCrawl" />
89 <add input="{HTTP_USER_AGENT}" pattern="PleaseCrawl" />
90 <add input="{HTTP_USER_AGENT}" pattern="^psbot" />
91 <add input="{HTTP_USER_AGENT}" pattern="^RealDownload"
    />
92 <add input="{HTTP_USER_AGENT}" pattern="^ReGet" />
93 <add input="{HTTP_USER_AGENT}" pattern="^Rippers\ 0" />
94 <add input="{HTTP_USER_AGENT}" pattern="SBIder" />
95 <add input="{HTTP_USER_AGENT}" pattern="^SeaMonkey$" />
96 <add input="{HTTP_USER_AGENT}" pattern="^sitecheck\
    internetseer\.com" />
97 <add input="{HTTP_USER_AGENT}" pattern="^SiteSnagger" />
98 <add input="{HTTP_USER_AGENT}" pattern="^SmartDownload"
    />
99 <add input="{HTTP_USER_AGENT}" pattern="Snoopy" />
100 <add input="{HTTP_USER_AGENT}" pattern="Steeler" />
101 <add input="{HTTP_USER_AGENT}" pattern="^SuperBot" />
102 <add input="{HTTP_USER_AGENT}" pattern="^SuperHTTP" />
103 <add input="{HTTP_USER_AGENT}" pattern="^Surfbot" />
104 <add input="{HTTP_USER_AGENT}" pattern="^tAkeOut" />
105 <add input="{HTTP_USER_AGENT}" pattern="^Teleport\ Pro"
    />
106 <add input="{HTTP_USER_AGENT}" pattern="^Toata\
    dragostea\ mea\ pentru\ diavola" />
107 <add input="{HTTP_USER_AGENT}" pattern="URI::Fetch" />
108 <add input="{HTTP_USER_AGENT}" pattern="urllib" />
109 <add input="{HTTP_USER_AGENT}" pattern="User-Agent" />
110 <add input="{HTTP_USER_AGENT}" pattern="^VoidEYE" />

```

```
111 <add input="{HTTP_USER_AGENT}" pattern="^Web\ Image\  
Collector" />  
112 <add input="{HTTP_USER_AGENT}" pattern="^Web\ Sucker" />  
113 <add input="{HTTP_USER_AGENT}" pattern="Web\ Sucker" />  
114 <add input="{HTTP_USER_AGENT}" pattern="webalta" />  
115 <add input="{HTTP_USER_AGENT}" pattern="^WebAuto" />  
116 <add input="{HTTP_USER_AGENT}" pattern="^[Ww]eb[Bb]andit  
" />  
117 <add input="{HTTP_USER_AGENT}" pattern="WebCollage" />  
118 <add input="{HTTP_USER_AGENT}" pattern="^WebCopier" />  
119 <add input="{HTTP_USER_AGENT}" pattern="^WebFetch" />  
120 <add input="{HTTP_USER_AGENT}" pattern="^WebGo\ IS" />  
121 <add input="{HTTP_USER_AGENT}" pattern="^WebLeacher" />  
122 <add input="{HTTP_USER_AGENT}" pattern="^WebReaper" />  
123 <add input="{HTTP_USER_AGENT}" pattern="^WebSauger" />  
124 <add input="{HTTP_USER_AGENT}" pattern="^Website\  
eXtractor" />  
125 <add input="{HTTP_USER_AGENT}" pattern="^Website\  
Quester" />  
126 <add input="{HTTP_USER_AGENT}" pattern="^WebStripper" />  
127 <add input="{HTTP_USER_AGENT}" pattern="^WebWhacker" />  
128 <add input="{HTTP_USER_AGENT}" pattern="^WebZIP" />  
129 <add input="{HTTP_USER_AGENT}" pattern="Wells\ Search\  
II" />  
130 <add input="{HTTP_USER_AGENT}" pattern="WEP\ Search" />  
131 <add input="{HTTP_USER_AGENT}" pattern="^Wget" />  
132 <add input="{HTTP_USER_AGENT}" pattern="^Widow" />  
133 <add input="{HTTP_USER_AGENT}" pattern="^WWW-Mechanize"  
/>  
134 <add input="{HTTP_USER_AGENT}" pattern="^WWWOFFLE" />  
135 <add input="{HTTP_USER_AGENT}" pattern="^Xaldon\  
WebSpider" />  
136 <add input="{HTTP_USER_AGENT}" pattern="zermelo" />  
137 <add input="{HTTP_USER_AGENT}" pattern="^Zeus" />  
138 <add input="{HTTP_USER_AGENT}" pattern="^Zeus\.*Webster"  
/>  
139 <add input="{HTTP_USER_AGENT}" pattern="ZyBorg" />  
140 </conditions>  
141 <action type="CustomResponse" statusCode="403"  
statusReason="Forbidden" statusDescription="Forbidden"  
/>  
142 </rule>  
143 <rule name="Imported Rule 2" stopProcessing="true">  
144 <match url="^wp-admin/includes/" ignoreCase="false" />  
145 <action type="CustomResponse" statusCode="403"  
statusReason="Forbidden" statusDescription="Forbidden"  
/>  
146 </rule>  
147 <rule name="Imported Rule 3" stopProcessing="true">
```

```
148 <match url="^wp-includes/[^/]+\.\php$" ignoreCase="false"  
    />  
149 <conditions>  
150 <!--# RewriteRule !^wp-includes/ - [S=3]-->  
151 <add input="{SCRIPT_FILENAME}" pattern="^(.*)wp-includes  
    /ms-files.php" ignoreCase="false" negate="true" />  
152 </conditions>  
153 <action type="CustomResponse" statusCode="403"  
    statusReason="Forbidden" statusDescription="Forbidden"  
    />  
154 </rule>  
155 <rule name="Imported Rule 4" stopProcessing="true">  
156 <match url="^wp-includes/js/tinymce/langs/.\.\php"  
    ignoreCase="false" />  
157 <action type="CustomResponse" statusCode="403"  
    statusReason="Forbidden" statusDescription="Forbidden"  
    />  
158 </rule>  
159 <rule name="Imported Rule 5" stopProcessing="true">  
160 <match url="^wp-includes/theme-compat/" ignoreCase="false"  
    />  
161 <action type="CustomResponse" statusCode="403"  
    statusReason="Forbidden" statusDescription="Forbidden"  
    />  
162 </rule>  
163 <rule name="Imported Rule 6" stopProcessing="true">  
164 <match url="^(.*)$" ignoreCase="false" />  
165 <conditions>  
166 <add input="{REQUEST_METHOD}" pattern="^(TRACE|DELETE|  
    TRACK)" />  
167 </conditions>  
168 <action type="CustomResponse" statusCode="403"  
    statusReason="Forbidden" statusDescription="Forbidden"  
    />  
169 </rule>
```

Listing 20: Part of web.config

7 Faktabaari's API sample response

```
{
  "posts": [
    {
      "type": "blogpost",
      "author": "Toimitus / Faktabaari",
      "published": "6.3.2015 - 06:50",
      "title": "Faktabaari valittiin Mediainnovaatiokilpailun finaaliin",
      "content": "<p>Faktabaari on valittu <a href='\"https://mediainnovaatio.fi/ajankohtaista/mediainnovaatiokilpailun-finalistit-valittu\">Mediainnovaatiokilpailun finaaliin</a>. Kilpailun finaali järjestetään 10. maaliskuuta. Voittajille on jaossa yhteensä 220 000 euroa, jotka on tarkoitettu muun muassa kilpailutyön jatkokehittämistä varten.</p><p>Mediainnovaatiokilpailu on Liikenne- ja viestintäministeriön hanke, jonka avulla halutaan vauhdittaa media-alan siirtymistä digiaikaan. Ministeriön mukaan kilpailun tavoitteena on edistää alan rakennemuutosta ja tukea uusien, kilpailukykyisten sisältö-, palvelu- ja ratkaisukonseptien luomista ja kehittämistä digitaalisessa toimintaympäristössä.</p><p>Palkintorahoista kisaa viisi yksityishenkilöä tai joukkuetta sekä kahdeksan yritystä. Kilpailun järjestäjän mukaan kisasta tulee kova, sillä kaikki finaalityöt ovat kiinnostavia.</p><p>Faktabaarin lisäksi finaalisissa ovat</p><p>* Prime, maksullisten mediasisältöjen aggregaattori</p><p>* Prejkfast-projekti, pienmaksupalvelu</p><p>* TalkTV, keskustelusovellus</p><p><p>* Aatos, journalismin spotify</p><p>Yrityssarjassa finaaliin pääsivät: Tässä ja nyt, <a href='\"http://www.ilmoituslaatikko\">www.ilmoituslaatikko</a>, Copyright Finder, Datomik, Avaus Asiakkuusmedia -konsepti, LiquidBlox, Leia ePaperi sekä Rapport-palvelu.</p><p>Opetus- ja viestintäministeri <strong>Krista Kiuru</strong> on aiemmin todennut, että mediainnovaatiokilpailu on yksittäinen pieni keino edistää isoa asiaa. \"Mutta pienenäkin eleenä se voi toimia ponnahduslautana tai olla avain jonkin haasteen ratkaisemiseksi\", Kiuru sanoo.</p><p>Mediainnovaatiokilpailu liittyy median innovaatiotukeen, josta hallitus päätti keväällä 2014.</p><p>Verkossa toimiva faktantarkistuspalvelu Faktabaari tarjoaa seuraajilleen kanavan tarkistaa netissä esitettyjen väitteiden todenperäisyyttä. Faktabaarin tarkistuksilla pystytään ehkäisemään ja torppaamaan virheellisen tiedon leviämistä sosiaalisessa mediassa.</p><p>Faktabaari palvelee kansalaisten medialukutaidon kehittymistä ja viime kädessä yhä asiallisempaa julkista keskustelua infoähkyn keskellä.</p><p>Joukkoistusta hyödyntävä Faktabaari on osoittanut voimansa kansainvälisestikin palkituissa vaalihankeissa.</p><p>Mediainnovaatiokilpailussa Faktabaari tähtää siihen, että uuden rahoituksen turvin palvelu voisi etsiä ansaintamahdollisuuksia, kansainvälistyä ja laajentaa toimintaansa vaalien ulkopuolelle.</p><n",
      "url": "http://faktabaari.fi/faktabaari-valittiin-mediainnovaatiokilpailun-finaaliin/",
      "image": "http://faktabaari.fi/wp-content/uploads/2015/03/Screen-Shot-2015-03-06-at-08.48.36.png"
    }
  ]
}
```

Figure 18: JSON response

8 Faktabaari's API response time

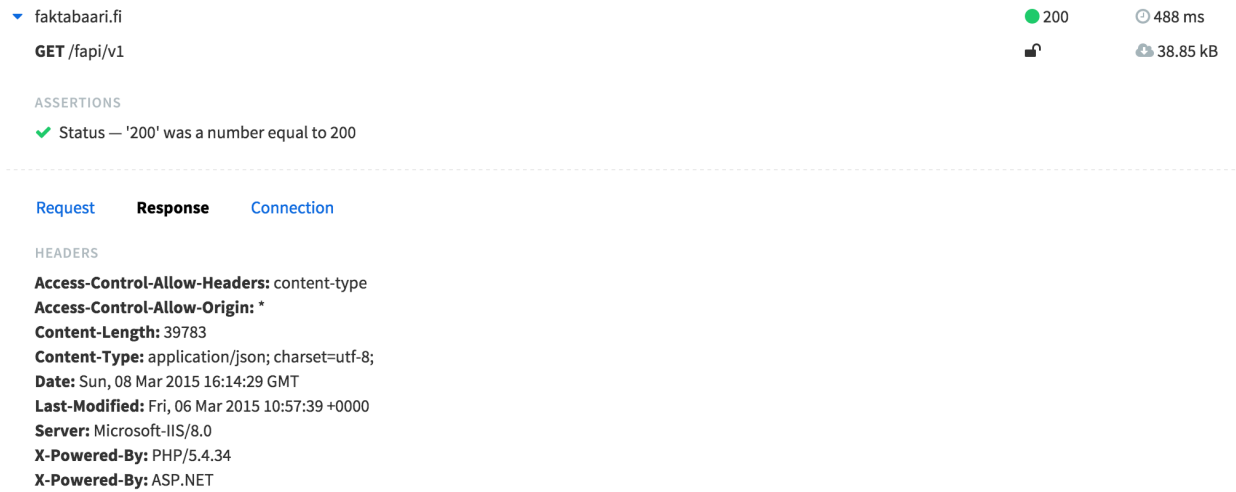


Figure 19: Runscope response time analysis

9 Faktabaari's API demo

FaktaBaari RAJAPINNAN DEMO

Faktojen hakeminen sanalla

Haku voi hakea joko vain eduskuntavaaleista tai eurovaaleista, tai sitten molemmista.

Hakutulokset sanalla talous:

Häkämies: Työn verotus on Suomessa Ruotsia kireämpää monissa tuloluokissa
Jaana Pelkonen: EU:n talouskasvu irtautunut CO2-päästöjen kasvusta
Väite: EU haluaa antaa verkkopankkitunnuksesi kolmannelle osapuolelle
Väite: EU:n budjetti on valtava
Väite: EU haluaa estää kasvisten kotiviljelyn ja kieltää yrtit
Talouselämä ja Iltalehti: EU pakottaa juomaan kylmää kahvia
VÄITE: Euroopan tilintarkastajilta toistuvasti moitteita EU-hallinnon taloudenhoidolle
Turunen (ps): Brysselin miljardilaskua ei Ruotsiin lähetetä, koska Ruotsi ei ole eurossa
Korhola (kok) Twitterissä: Meluisa tuulivoima kasvattaa päästöjä
Lehto (vihr): Jopa puolet EU:n rahoista menee maataloustuottajille

Esimerkki API:n datan käytöstä eri tavalla

Viimeisimmät faktat (7 kpl) eduskuntavaaleista ovat jakautuneet seuraavalla tavalla:

Tulevaisuudessa samaan tyyliin voitaisiin näyttää dataa vaikka puolueittain, vaalipiireittäin jne.

Figure 20: Full demo at dev.sirdar.fi/apidemo