Robert Stankevich

Basic guidelines for Drupal 8 projects in FCG

Metropolia University of Applied Sciences

Bachelor of Engineering

ICT

Bachelor’s Thesis

Author Title Number of Pages Date	Robert Stankevich Basic guidelines for Drupal 8 projects in FCG 44 pages
Degree	Bachelor of Engineering
Degree Programme	ICT
Professional Major	Software Engineering
Instructors	Kimmo Sauren, Lecturer, Metropolia UAS Matti Oosi, ICT Manager, FCG Oy

More than twenty years ago Bill Gates published provisionary essay “Content is King”. Modern businesses rely on various online services and importance of content is only growing. Web development itself changed greatly and now involves various specialists. Some of them have a very indirect relation to programming and classical concept of technical majors. Content management systems generally simplify process of creation of complex web services.

This bachelor’s project is a result of experience and knowledge gained in Finnish Consulting Group Oy. FCG’s ICT department works with dozens of different web services. Lately many of these projects were built with usage of Drupal 8 content management system. Various technical issues and challenges occurred in last few years. It allowed department to significantly familiarize with the system, learn some of its features and form some kind of best practices.

Findings, made in the company, have to be summarized and analyzed. Gathered in a single document they would serve for a quick start of future projects. In addition, synopsis of such kind allows outlining of objectives for further development and improvement of existing processes and workflows.

Some of existing solutions are probably not perfect yet, but they may become an initial point for better ones. This project describes environment, used for Drupal 8 projects, presents different approaches and illustrates some particular solutions used by FCG and subcontractors.

Keywords	FCG, Drupal 8, web development, CMS

Table of Contents

List of Abbreviations

CMS	Content management system.
CSS	Cascading Style Sheets.
DOM	Document object model
LAMP	Linux + Apache + MySQL + PHP stack.
LEMP	Linux + Nginx + MySQL + PHP

1 Introduction

Finnish Consulting Group Oy is a company that provides services in various areas. FCG’s ICT department constantly develops web-services to support products and activities of the company and its partners. Even for smaller information sites amount of content supposed to grow in time. That means involvement of content managers, who are often non-technical personnel. Therefore, process of delivery of new content and its management has to be significantly simple and effective. Content management systems alike Drupal are aimed to fulfill these needs.

Some of FCG’s services were implemented with usage of Drupal 8 CMS. In those projects were many differences like purposes, goals and stakeholders. Every project had something unique in its structure and functionality. However many of these services also shared a number of similarities. Same solutions for particular tasks were reused multiple times. Various repetitions occurred.

First task of this work is to analyze and compare existing projects and most used modules (both custom and contributed), content types, environmental settings and solutions for specific issues. This analysis would allow formulating of typical rules for development of future Drupal 8 projects. Based on previous steps should be built significantly stable prototype that would be reused for future FCG’s web services.

This work should serve as a gathering of guidelines and recommendations for better understanding and usage of Drupal 8 content management system.

2 Introduction to Drupal 8

2.1 About Drupal

> “Drupal is an open source content management system being used by hundreds of thousands of organizations and individuals to build engaging, content-rich websites. Building a website in Drupal is a matter of combination together various “building blocks”… in order to customize your website’s functionality to your precise needs. Once built, a Drupal website can be maintained through the use of online forms, without any code having to be changed manually. Drupal is free to use, and it has an enormous library of constantly evolving tools…” (1)

Drupal, as defined, is a content management system. CMS, in its turn, is a software application that helps to produce and modify digital content and, in case of Drupal and similar products, publish it on the Web.

Based on different sources there are from about half- to nearly one million of websites powered by Drupal on entire Web. In other words, numerous companies, institutions and organizations of various areas are using Drupal for their needs. Among them, can be met known titles like Google and NVidia, NASA and FIFA, M.I.T. and Harvard universities, governmental units of several countries.

There are many top-lists and comparisons of content management systems and position of Drupal on them may vary. However, there is no doubts that it is one of the most popular solutions.

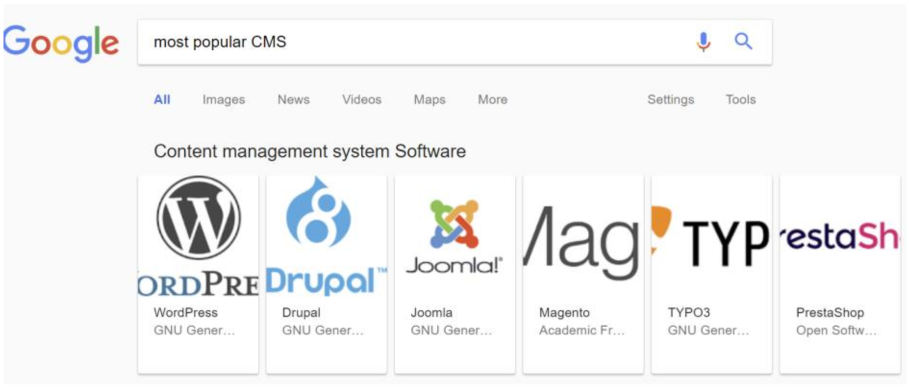
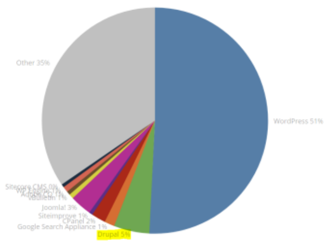


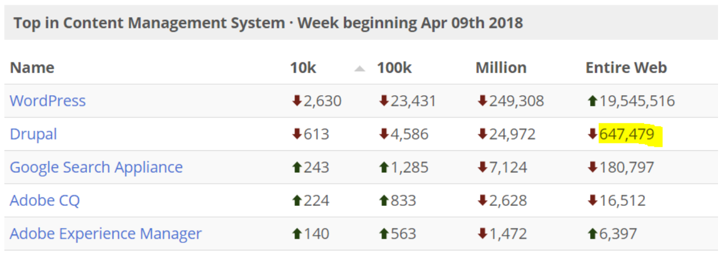
Figure 1. Google search phrase ”most popular CMS” (2)

As shown in Figure 1, Drupal appears among the top results on google for search phrase “Most popular CMS”.

2.2 Competitors

While position of Drupal in top-lists is arguable, the first place all analytics give to WordPress. According to builtwith.com, more than a half of all websites on Internet are now based on this CMS. “WordPress powers 30% of the internet” (3), - written on the front-page of wordpress.com.





Top in Content Management System · Week beginning Apr 09th 2018

Name	10k	100k	Million	Entire Web
WordPress	↓2,630	↓23,431	↓249,308	↑19,545,516
Drupal	↓613	↓4,586	↓24,972	↓647,479
Google Search Appliance	↑243	↑1,285	↓7,124	↓180,797
Adobe CQ	↑224	↑833	↓2,628	↓16,512
Adobe Experience Manager	↑140	↑563	↓1,472	↑6,397

Figure 2. CMS trends (4)

Figure 2 illustrates popularity of various content management systems provided by builtwith.com. In general, both Drupal and WordPress are quite similar. Both systems are sharing same technologies and techniques. Short list of them presented in Table 1.

	WordPress	**Drupal**
Backend technologies	PHP, MySQL	PHP, Symphony, MySQL
Frontend technologies	JavaScript, CSS	JavaScript, CSS
License	GNU GPL 2	GNU GPL 2
Operating System	Unix-like	Unix-like, Windows
Miscellaneous	Responsive design, extendibility, community, etc	

Table 1. Similarities of WordPress and Drupal

Both systems have their pros and cons. Majority of experts share an opinion that none of this two CMS should be called as “a better one”. It would be more accurate to say which of them better suits particular task developer is facing at moment of time.

WordPress is nearly ideal for fast, significantly simple and cheap solutions. Level of technical skills and competences in such cases is considerably low. In fact, with this CMS any knowledge of web development is not required at all, since anybody can start a new website hosted on WordPress own servers. On the other hand, despite a high level of customization, solutions based on WordPress are somewhat recognizable in their “look and feel”. Often they are nearly identical. System seems to lack flexibility.

Drupal, in its turn, is scalable and allows creating of complex and rather unique systems. While WordPress is somewhat modest for average users and content managers, Drupal provides more freedom for developers. Many Drupal solutions were integrated with numerous third-party systems and applications, including popular enterprise solutions. In addition, Drupal is considered as a secure enough system.

Besides Drupal and WordPress, there are many other open source content management systems on the market. Some of them are also quite similar to those two mentioned above. Among them Joomla, Magento, Bitrix, MODX and others.

2.3 Lifecycle

Popularity of particular CMS is changing in time. Being up-to-date undoubtedly matters a lot in this case. Drupal evolved from version 1.0 to 7.0 in a good pace.

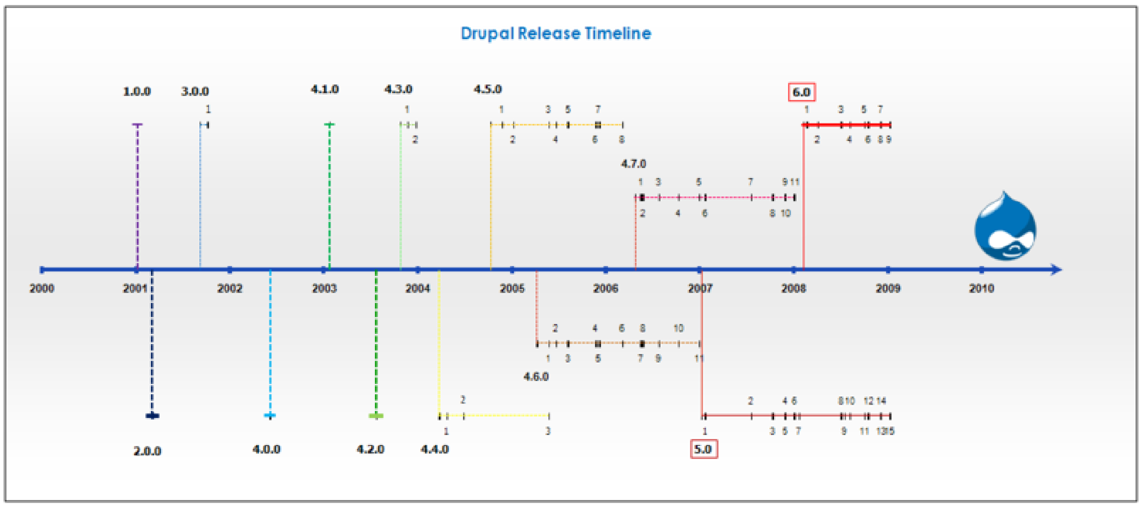


Figure 3. Drupal release timeline (5)

As seen from Figure 3, time gap – around five years - between 7th and 8th versions of Drupal core, comparing to previous ones, was quite long. On the other hand, changes, applied to the project, were much deeper. Among the most important changes often mentioned: integration of Twig template engine, adoption of OOP, Views module enclosure to core, setting CKEditor as default editor, integration of REST, multilingual feature. Many experts prepared illustrative lists of new features of the system, when it was published. One of examples presented in Figure 4.

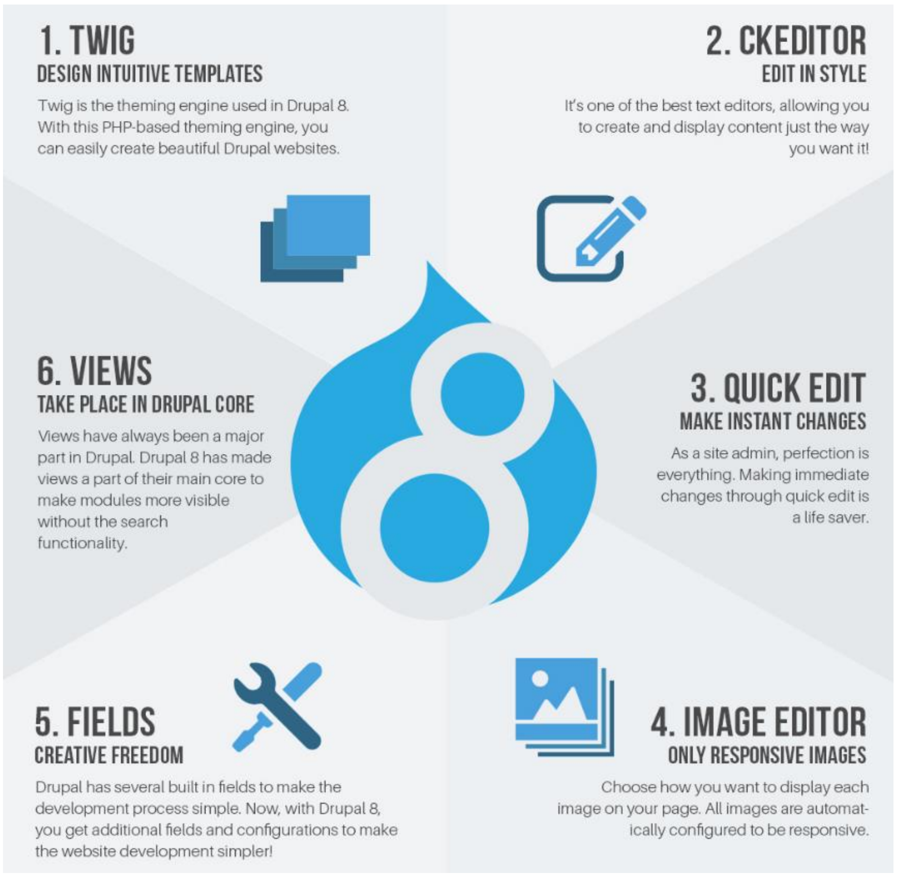

1. TWIG
DESIGN INTUITIVE TEMPLATES
Twig is the theming engine used in Drupal 8. With this PHP-based theming engine, you can easily create beautiful Drupal websites.
2. CKEDITOR
EDIT IN STYLE
It's one of the best text editors, allowing you to create and display content just the way you want it!
6. VIEWS
TAKE PLACE IN DRUPAL CORE
Views have always been a major part in Drupal. Drupal 8 has made views a part of their main core to make modules more visible without the search functionality.
3. QUICK EDIT
MAKE INSTANT CHANGES
As a site admin, perfection is everything. Making immediate changes through quick edit is a life saver.
5. FIELDS
CREATIVE FREEDOM
Drupal has several built in fields to make the development process simple. Now, with Drupal 8, you get additional fields and configurations to make the website development simpler!
4. IMAGE EDITOR
ONLY RESPONSIVE IMAGES
Choose how you want to display each image on your page. All images are automatically configured to be responsive.

Figure 4. Top Drupal 8 features (6)

On April 2018, stable version of Drupal is 8.5.1. On drupal.org mentioned more than five thousands of modules compatible with Drupal 8 core (around 13 thousands for Drupal 7). Obviously, this number will grow.

2.4 Developer roles

According to documentation, there are five layers in Drupal dataflow. Drupal documentation contains dataflow diagram illustrated in Figure 5.

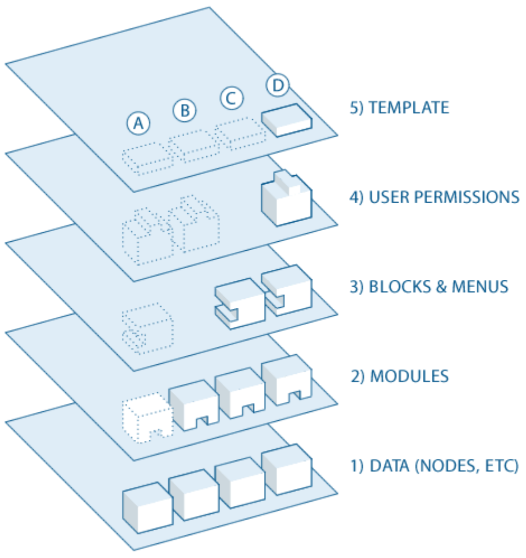


Figure 5. Drupal flow (7)

Same problems in Drupal can be solved in different ways: writing custom code, manipulating core features or looking for a contributed module. Based on their approaches, developers may be divided into different types or roles. Three of them in a manner have become legitimized among experts:

- Frontend developers
- Backend developers
- Site builders

Frontend developers concern on templates, JavaScript and CSS. These specialists are also named as Theme builders. Backend developers work with PHP code, database and, if needed, integrations. Site builders mostly deal with content types, site configuration, settings of core and contributed modules.

In fact, often single developer may execute all three roles at the same time serving as “a jack of all trades” or even “full stack developer”. In addition, for an effective development in Drupal usually it is not enough to have excellent skills in backend programming or frontend techniques. Drupal has its own API and coding standards. Without knowledge of how does the system works, development of new services based on it would result in numerous unwanted problems and difficulties.

3 Environment

3.1 LAMP and LEMP

LAMP is an abbreviation for a popular web-technology stack consisting of Linux operating system, Apache webserver, MySQL database and PHP. For Drupal-based services in FCG mostly used variation of this architecture known as LEMP, where E stands for Nginx webserver. This chapter describes main components of typical environment used by FCG's ICT department for Drupal projects. To be specific, these components are CentOS 7 operating system, Nginx webserver, MariaDB database application and PHP 5.6.

3.2 CentOS

CentOS is a fork of Red Hat Enterprise Linux. This system is nearly 100% copy of RHEL and therefore shares almost all of advantages its parent has, but licensed as free software. Disadvantage comparing to paid solution is lack of official support. Partially community behind the project solves this problem.

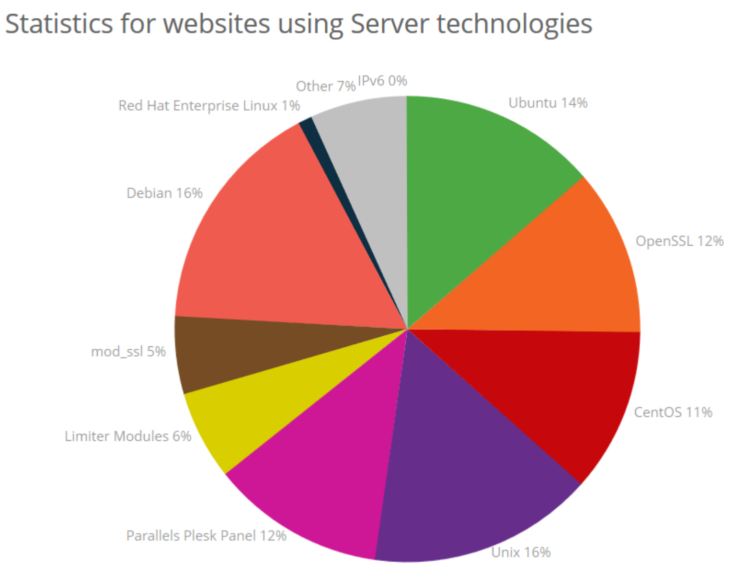


Figure 6. Automated technology comparisons are not 100% accurate (8)

Various web services provide comparisons of different technologies based on usage statistics. As it can be seen from the Figure 6, it is hard to rely on this data, since it covers only publicly available information, and discovery techniques may not be 100% precise. Still they provide some sort of reflection of the situation in general that may indicate actual trends.

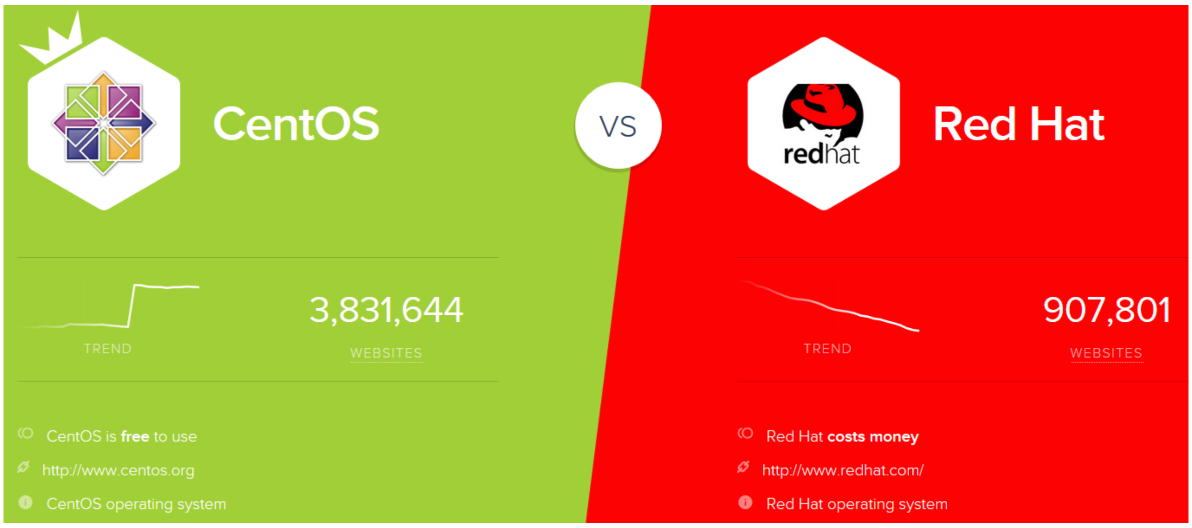


Figure 7. CentOS is used more often than RHEL (9)

Figure 7 illustrates comparison provided by similartech.com. It indicates that CentOS is four times more popular than RHEL. Same source also points, CentOS having nearly equal market shares with Debian and Ubuntu. Assuming some number of faults, both builtwith.com and similartech.com present relatively similar results. In addition, SimilarTech states “CentOS is leading in Business & Industry, Internet & Telecom, Shopping” website categories (10).

Joined together previous statements can be interpreted as an intention of many businesses to use stable open source systems in general, but for critical services they prefer to use more secure and stable ones. This “freeware but still enterprise level” nature of CentOS was also one of the main reasons for decision to use it in FCG’s projects.

3.3 Nginx

According to Hannu Niemi, Chief Developer in FCG’s ICT department, Nginx is used in company’s projects for its superior performance on high-load and simple configuration. This opinion is also proved by various benchmark tests. In particular, Nginx is showing better performance than Apache for concurrent connections. Talking of configurations, it already has sort of “cookbook” with recipes for many popular web applications. Recommended configuration for Drupal, looks as illustrated in Listing 1.

```
server {
    server_name example.com;
    root /var/www/drupal8; ## <-- Your only path reference.

    location = /favicon.ico {
        log_not_found off;
        access_log off;
    }

    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }

    # Very rarely should these ever be accessed outside of your lan
    location ~* \.(txt|log)$ {
        allow 192.168.0.0/16;
        deny all;
    }

    location ~ \..*/.*\.php$ {
        return 403;
    }

    location ~ ^/sites/.*/private/ {
        return 403;
    }

    # Block access to scripts in site files directory
    location ~ ^/sites/[^/]+/files/.*\.php$ {
        deny all;
    }

    # Allow "Well-Known URIs" as per RFC 5785
    location ~* ^/.well-known/ {
        allow all;
    }

    # Block access to "hidden" files and directories whose names begin with a
    # period. This includes directories used by version control systems such
    # as Subversion or Git to store control files.
    location ~ (^|/)\. {
        return 403;
    }

    location / {
        # try_files $uri @rewrite; # For Drupal <= 6
        try_files $uri /index.php?$query_string; # For Drupal >= 7
    }

    location @rewrite {
        rewrite ^/(.*)$ /index.php?q=$1;
    }

    # Don't allow direct access to PHP files in the vendor directory.
    location ~ /vendor/.*\.php$ {
        deny all;
        return 404;
    }

    # In Drupal 8, we must also match new paths where the '.php' appears in
    # the middle, such as update.php/selection. The rule we use is strict,
    # and only allows this pattern with the update.php front controller.
    # This allows legacy path aliases in the form of
    # blog/index.php/legacy-path to continue to route to Drupal nodes. If
    # you do not have any paths like that, then you might prefer to use a
    # laxer rule, such as:
    #   location ~ \.php(/|$) {
    # The laxer rule will continue to work if Drupal uses this new URL
    # pattern with front controllers other than update.php in a future
    # release.
    location ~ '\.php$|^/update.php' {
        fastcgi_split_path_info ^(.+?\.php)(|/.*)$;
        # Security note: If you're running a version of PHP older than the
        # latest 5.3, you should have "cgi.fix_pathinfo = 0;" in php.ini.
        # See http://serverfault.com/q/627903/94922 for details.
        include fastcgi_params;
        # Block httpoxy attacks. See https://httpoxy.org/.
        fastcgi_param HTTP_PROXY "";
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        fastcgi_param QUERY_STRING $query_string;
        fastcgi_intercept_errors on;
        # PHP 5 socket location.
        #fastcgi_pass unix:/var/run/php5-fpm.sock;
        # PHP 7 socket location.
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
    }

    # Fighting with Styles? This little gem is amazing.
    # location ~ ^/sites/.*/files/imagecache/ { # For Drupal <= 6
    location ~ ^/sites/.*/files/styles/ { # For Drupal >= 7
        try_files $uri @rewrite;
    }

    # Handle private files through Drupal. Private file's path can come
    # with a language prefix.
    location ~ ^(/[a-z\-]+)?/system/files/ { # For Drupal >= 7
        try_files $uri /index.php?$query_string;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        try_files $uri @rewrite;
        expires max;
        log_not_found off;
    }
}
```

Listing 1. Nginx recommended configuration for Drupal 8 (11)

For initial Drupal installation, it is enough to copy the code to a site-specific .conf file under Nginx configurations directory e.g. prototype.conf. Only two lines shown in Listing 2 have to be modified.

```
    server_name example.com;
    root /var/www/drupal8; ## <-- Your only path reference.
```

Listing 2. Sample from default Nginx settings for Drupal

After rewriting, they should look as shown in Listing 3.

```
    server_name prototype.fcg.fi;
    root /srv/prototype;
```

Listing 3. Modified Nginx settings

Some additional settings were used in multiple FCG's services. Various modules, for example SimpleSAMLphp Authentication, may require specific configuration changes. Listing 4 demonstrates these settings.

```
location /simplesaml {
     alias /srv/insaitti/vendor/simplesamlphp/simplesamlphp/www;
     location ~ ^(?<prefix>/simplesaml)(?<phpfile>.+?\.php)(?<pathinfo>/.*)?$
{
          fastcgi_param SIMPLESAMLPHP_CONFIG_DIR /srv/insaitti/vendor/sim-
plesamlphp/simplesamlphp/config;
          fastcgi_param SCRIPT_FILENAME $document_root$phpfile;
          fastcgi_param PATH_INFO       $pathinfo if_not_empty;
          include fastcgi_params;
          fastcgi_pass 127.0.0.1:9000;
     }
}
```

Listing 4. SimpleSAMLphp settings for Insaitti

Another particular case, often used for websites, is adding SSL certificates and proper modifications for secure connections through HTTPS protocol. Sample with such settings for development version of Insaitti exposed in Listing 5.

```
        listen 443 ssl;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_certificate      /etc/ssl/certs/insaitti-dev.crt;
        ssl_certificate_key  /etc/ssl/certs/insaitti-dev.key;
```

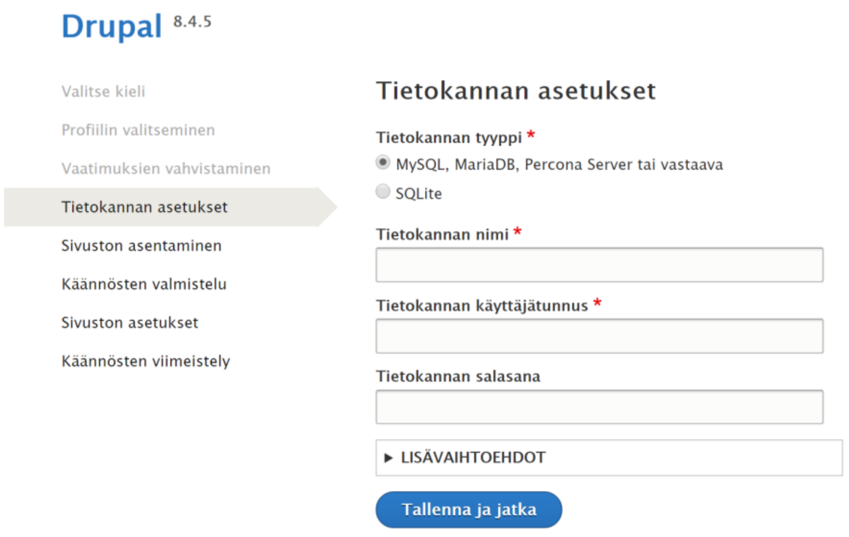
Listing 5. SSL certificate settings

When a web services supposed to be accessed through HTTPS protocol, an SSL certificate has to be created. For development needs possible to use self-signed certificates, but for site on production servers better to issue a proper one from an authorized organization. Otherwise visitors will be informed that site is not secure.

3.4 MariaDB

As many other web applications, Drupal requires a database for its functioning. According to documentation, for Drupal 8 recommended to use MySQL 5.5.3, MariaDB 5.5.20 or Percona Server 5.5.8 (12). Other database applications are also possible to use, but it is noted, that conflicts with some modules, that are not part of Drupal core, may occure.

Experts mention various advantages and disadvantages of using particular system. Argues on performance, features and even philosophy happening often. Hardly any these discussions became a reason for FCG's ICT department to use MariaDB instead of any other database server. More likely, the most important in this case was the fact that MariaDB is included in official CentOS repositories. In other words, MariaDB is a default option for database application in this operating system, and installation of other analogues means additional effort and possibly some unwanted system stability impacts.


Drupal 8.4.5

- Valitse kieli
- Profiilin valitseminen
- Vaatimuksien vahvistaminen
- Tietokannan asetukset
- Sivuston asentaminen
- Käännösten valmistelu
- Sivuston asetukset
- Käännösten viimeistely

Tietokannan asetukset

Tietokannan tyyppi *

● MySQL, MariaDB, Percona Server tai vastaava

○ SQLite

Tietokannan nimi *

Tietokannan käyttäjätunnus *

Tietokannan salasana

▸ LISÄVAIHTOEHDOT

Tallenna ja jatka

Figure 8. Database settings on site installation

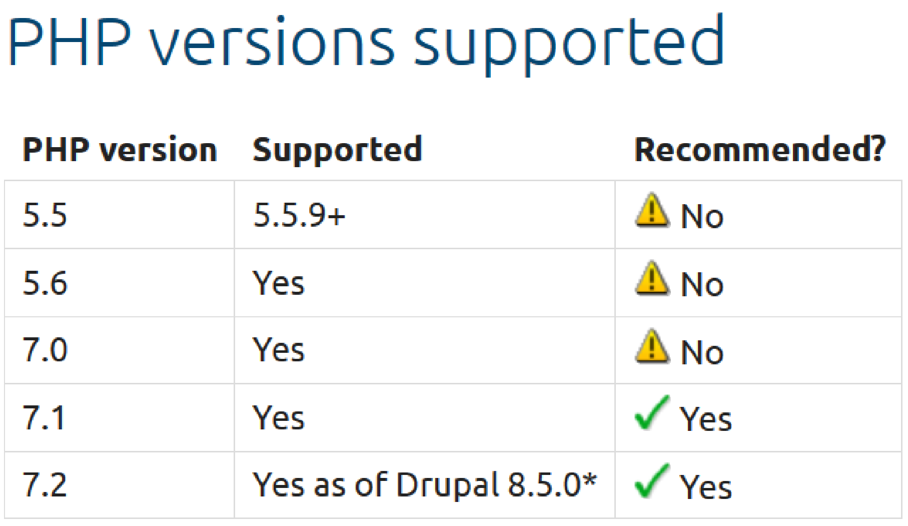
Database connection settings should be provided during initial site installation as it shown in Figure 8. Database name, user credentials and database server ip-address then added to settings.php file of the site as illustrated in Listing 6.

```
$databases['default']['default'] = array (
  'database' => 'prototyped8',
  'username' => 'prototype-user',
  'password' => 'prototype-password',
  'prefix' => '',
  'host' => 'localhost',
  'port' => '3306',
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',
  'driver' => 'mysql',
);
```

Listing 6. Example of database connection settings

In Drupal 8, it is possible to use multiple databases for the same service. This feature has not yet been used in FCG. It is still possible that such topic will be worth of investigation for future projects.

3.5 PHP 5.6



PHP versions supported

PHP version	Supported	Recommended?
5.5	5.5.9+	⚠ No
5.6	Yes	⚠ No
7.0	Yes	⚠ No
7.1	Yes	✓ Yes
7.2	Yes as of Drupal 8.5.0*	✓ Yes

Figure 9. PHP versions supported by Drupal (13)

As seen from the Figure 9, minimal supported version of PHP is 5.5. However, on April 2018 only 7.1 and 7.2 versions are recommended. It was announced that “Drupal 8 will drop support for PHP 5.5 and 5.6 on March 6, 2019” (13). This is currently one of the main concerns for system build used in FCG. CentOS does not officially support PHP versions higher than 5.x. It is possible to install PHP 7.x on this operating system by enabling additional repositories, but appropriate and fault-safe installation requires a proper investigation, which is supposed to take place in FCG later.

3.6 Development and production environments

In FCG initial development of a new project usually starts in development environment – server machine that has the same configuration as future production server. After core functionality of application achieved, project is transferred to production server. At this stage website is not publicly accessible yet, but the customer has possibility to test the project, form additional requirements and start production of content. Finally, when permission of a customer granted, the project is published. After that further development, if needed, happens in dev environment.

Process at current state is not perfect. Routine actions usually have to be performed on per-project basis and therefore are highly time-consuming. In particular, when critical updates for Drupal core are published, the whole project is cloned to development machine, then developers install and test updates and after that copy update version back to the production machine.

Multisite installation – when multiple projects work with single core instance – is one of the basic features of Drupal 8 that would greatly save time, but also seems to be risky: it is possible that after core updates applied, some components of a particular site would become malfunctioning. For that reason, even smaller projects in FCG usually configured as a single site installations. If future projects will use same base as FCG prototype, process of massive updates would become slightly easier and predictable. Further development would also allow automating of some known routines as well.

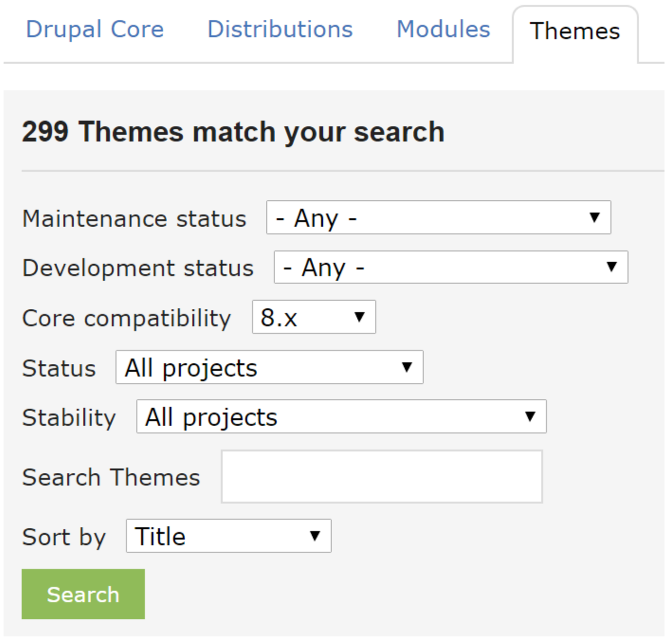
4 Front-end

Themes are the Drupal method for controlling your site's presentation. It's not enough to get a site functionality working – it also has to feel like your own, and has to be distinguished from other sites out there. (1)

Front-end development for Drupal also has the name *Theme building*. Generally speaking, theme is a combination of CSS and JavaScript libraries alongside with templates and a couple of .yml files that contain some essential settings. Core skills of a theme-builder should lay in CSS and JavaScript, but it is also necessary to understand Drupal itself and, especially, templates. Even a state-of-the-art frontend design would become useless without such knowledge.

4.1 Contributed themes and Bulma

According to Figure 10, by May 2018 nearly 300 of themes compatible with 8.x core were presented at drupal.org. In addition, there are numerous paid solutions on the Web. In various FCG's projects, both custom and proprietary solutions were used.



299 Themes match your search

Maintenance status	- Any -
Development status	- Any -
Core compatibility	8.x
Status	All projects
Stability	All projects
Search Themes	
Sort by	Title

Figure 10. hundreds of themes available for Drupal 8

In most cases, even paid themes needed some additional work to meet project requirements. It has to be admitted, that sometimes it would be less complicated to 'start from the scratch'. Since in Drupal it is possible to solve various issues in many different ways, some particular methods, despite their quality and/or effectiveness, are possibly not corresponding with the approaches used in the company for similar problems. Often it has positive impact on learning Drupal in general, but also demands time for investigation and therefore slows down the project workflow.

Many of Drupal themes are based on popular frontend frameworks like Bootstrap or Foundation. Frameworks generally save time and frees developer from solving common trivial issues, and allows focusing on more pressing ones. Lately FCG's ICT department produced several projects using Bulma theme, built with the frontend framework of the same name. As it is described on its website, "Bulma is an open source CSS framework based on Flexbox" (14).

There are many benefits to use Bulma for web applications. As it should be with any modern front-end framework, it has responsive design and mobile first approach, meaning that content is presented in most convenient for user way regardless what kind of device (mobile, tablet, desktop) he uses to access webpage.

Bulma has quite a number of ready-to-use solutions for various front-end objectives: responsive columns or grid, various elements and components like buttons and icons, menus and dropdowns, media objects and hero banners etc. In addition, community provides growing number of extensions adding new components and feature like carousel or page-loader.

Not the least important, that Bulma is well documented and significantly easy to learn and use. Class names are descriptive and clear for understanding. Experts claim that Bulma is lightweight, meaning it does not affect the website bandwidth, as it happens with some other frameworks.

Being a newcomer in web development is both advantage and disadvantage. On one hand, Bulma has chances to correct deficiencies of its competitors; on the other hand, it still has several unsolved issues. By May 2018 the latest release of Bulma was 0.7.1, meaning it was still away from being a complete product. Bulma base theme for Drupal also lacks some features and remains in alpha state. However, it is already possible to use this project as a quick start for own custom themes. It is even provided with Drupal shell command to generate subthemes.

```
drush bulma "FCG proto"
```

Listing 7. drush command to generate subtheme named “FCG proto”

When command shown in Listing 7 is issued from the project root, a new folder containing custom theme FCG proto is added to themes directory.

4.2 Theme configuration

Development of any theme or module in Drupal starts with initial .info.yml file, which has to be properly named and put into the core folder of custom project. This “first brick in a wall” serves for notification Drupal about this module or theme, defines its type and some settings.

```
name: FCG proto
description: A theme based on Bulma.
type: theme
base theme: bulma
package: Core
core: 8.x


regions:
  header: 'Header'
  navbar_branding: 'Branding'
  navbar_social: 'Social'
  header_search: 'Search'
  header_tabs: 'Tabs'
  primary_menu: 'Primary menu'
  secondary_menu: 'Secondary menu'
  highlighted: 'Highlighted'
  help: 'Help'
  content: 'Content'
  sidebar_first: 'Sidebar First'
  sidebar_second: 'Sidebar Second'
  tile_one: 'Tile 1'
  tile_two: 'Tile 2'
  tile_three: 'Tile 3'
  tile_four: 'Tile 4'
  tile_five: 'Tile 5'
  bottom: 'Bottom'
  footer: 'Footer'

libraries:
  - fcg_proto/global
```

Listing 8. sample from fcg_proto.info.yml

As seen from Listing 8, info.yml file consists of rather simple key-value pairs. According to documentation of Drupal, keys name, type and core are mandatory. Without them core would not recognize a theme and it installation would fail. Base theme key is a recommended one. In case of FCG proto, it is required since theme is inherited form Bulma theme. Regions – sort of areas where various blocks will be added – are also defined in info.yml file. It is important to pay attention to syntax of .yml files, since any mistake in it may cause critical errors.

4.3 Libraries

There are multiple ways to include CSS and JavaScript to Drupal project. For example, inline styles in templates work as in any HTML-document as it is shown in Listing 9.

```
<div class="column" style="padding-top: 0">
```

Listing 9. sample from a template

It has particular usage in such cases like adding a background image from an image field of a node. Particular example shown in Listing 10.

```
<div class="columns is-centered front-hero" style="background-image: url({{
file_url(node.field_top_image.entity.fileuri) }});">
```

Listing 10. sample from a template

Other way is to attach libraries with PHP code. For example, attaching them within rendering array of a custom block as it is presented in Listing 11.

```
$build['#attached']['library'][] = 'fcg_shoutbox/shoutbox';
```

Listing 11. attaching library to rendering array

More information about rendering arrays provided in Chapter 4 of this work.

4.3.1 Working with .yml files

Since themes mostly contain styles and scripts for entire project, it seems to be more appropriate to define them in corresponding .yml files in theme root. Sample in Listing 12 is taken from fcg_proto.libraries.yml file. It points to own CSS and JavaScript files and also notifies Drupal to use global library from Bulma base theme.

```
global:
  css:
    theme:
      assets/overrides.css: {}
      assets/fcg_proto.style.css: {}
      assets/bulma-extensions/bulma-carousel/dist/bulma-carousel.min.css: {}
  js:
      assets/bulma-extensions/bulma-carousel/dist/bulma-carousel.min.js: {}
      assets/bulma-extensions/bulma-carousel/dist/bulma-carousel.js: {}
  dependencies:
    - bulma/global
```

Listing 12. example from fcg_proto.libraries.yml file

Bulma base theme, as well as its subthemes, includes basic Bulma CSS library. Installation of extensions is not yet supported, but can be done manually. One of possible solutions being used in FCG proto is to download and place additional .css files to theme and then include them into project by naming in libraries_proto.libraries.yml file.

```
libraries:
  - fcg_proto/global
```

Listing 13. example from fcg_proto.info.yml file

Sample presented in Listing 13 is taken from fcg_proto.info.yml file. Any other library defined in libraries.yml may be attached in similar way.

4.3.2 External libraries

It is not necessary to keep all the CSS or JavaScript libraries on server. The same way as locals, it is possible to use external libraries as shown in following example in Listing 14.

```
https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js: { type:
external, minified: true }
```

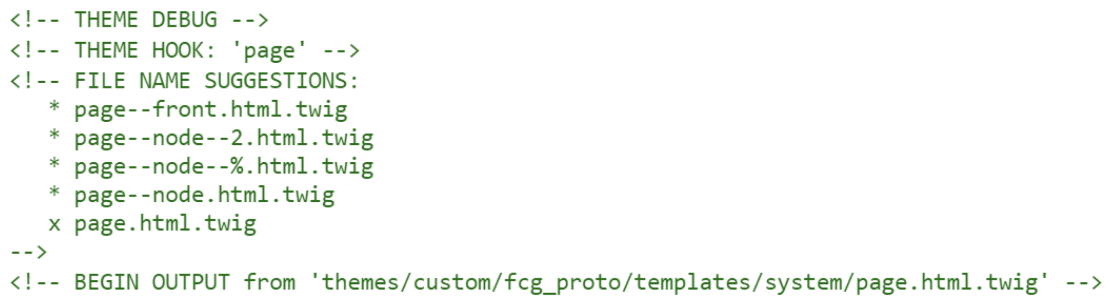
Listing 14. example of external library from libraries.yml file

External libraries may save resources, but also mean some risks. First, servers where such libraries are hosted may happen to be offline or experience any other connectivity problem. Then obviously, library would be inaccessible and it may dramatically affect website's presentation.

Another serious concern is security. Remote JavaScript libraries may become a major vulnerability in case of attack like cross-site scripting. In addition, if library or any other content is loading from an HTTP URL, but the site is using HTTPS, mixed content error will occur and remote content will be blocked.

4.4 Templates

One of the major changes in Drupal 8 comparing to previous versions, is integration of Twig as template engine. Almost every element in Drupal may have its own template. Fields, nodes, regions, content types and their display variations, views, forms etc. If some templates are missing in the project, default ones from the Drupal core or base theme are used to render those elements.



```
<!-- THEME DEBUG -->
<!-- THEME HOOK: 'page' -->
<!-- FILE NAME SUGGESTIONS:
   * page--front.html.twig
   * page--node--2.html.twig
   * page--node--%.html.twig
   * page--node.html.twig
   x page.html.twig
-->
<!-- BEGIN OUTPUT from 'themes/custom/fcg_proto/templates/system/page.html.twig' -->
```

Figure 11. example of naming suggestions for a template

Custom templates are recognized by Drupal if they are named according to naming convention. When twig debug is enabled, html comments, as one shown in Figure 11, with name suggestions and path to used template are added to DOM and seen with developer tools of browser. It would be a good practice to keep templates in structured way as it is shown in Listing 15.

```
| - templates
| | - block
| | | - block.html.twig
| | - node
| | | - node.html.twig
| | - system
| | | - html.html.twig
| | | - page.html.twig
| | | - region--footer.html.twig
| | - views
| | | - views-view-table.html.twig
```

Listing 15. example of folder structure for templates

As a template language, Twig has its own syntaxes. It operates with variables available in element for which template is designed, like contents of fields in templates of nodes. It is also possible to apply some logics using if – else condition or loops, create macros etc. Listing 16 contains example taken from page.html.twig, which is used in FCG proto and inherited from Bulma base theme.

```
      {% block main_content %}
      {% if not page.content_attributes %}
      {%
        set page = page|merge({
          'content_attributes': create_attribute().addClass([
            'main-content',
            'column',
          ]),
        })
      %}
      {% endif %}
      {# Main content #}
      <div{{page.content_attributes}}>
        {{ page.content }}
      </div>
      {% endblock main_content %}
      {% block sidebar_first %}
      {# Sidebar first #}
      {% if page.sidebar_first %}
      {# Set attributes for content. #}
        {% if not page.sidebar_first_attributes %}
        {%
          set page = page|merge({
            'sidebar_first_attributes': create_attribute().addClass([
              'sidebar-first',
              'column',
              'is-2',
            ]),
          })
        %}
        {% endif %}
        <aside{{page.sidebar_first_attributes|without('role')}} role="compli-
mentary">
          {{ page.sidebar_first }}
        </aside>
      {% endif %}
      {% endblock sidebar_first %}
```

Listing 16. sample from page.html.twig

Sample in Listing 16 determines rendering of main content and sidebar first regions of any page on site, if there is no other template for that. Contents of main content region, following the code, will be printed to div element with classes ‘main-content’ and ‘column’ if any other attributes not set. Sidebar first region, on its turn, will be contained within aside element and will be printed under condition that this region is not empty (some blocks should be attached to it).

Sometimes discovery of variables accessible in a particular template becomes a nontrivial task, especially in the case of complex elements. Incorrectly called variables, as well as syntactical inaccuracies, often may result in critical errors. Additional modules may extend Twig and make working with it significantly easier. Particular examples and recommendations on theme building and templates available from Drupal documentation: www.drupal.org/docs/8/theming/twig.

5 Backend

Drupal 8 core contains around 70 modules at current state. Over five thousands more available at drupal.org. Modules extend and improve existing functionality of Drupal. Whenever developer faces a requirement for a project, that cannot be fulfilled by basic core functionality, there is a big possibility that some of contributed modules would help with it.

5.1 Installation of modules

Installation of modules, as well as many other actions, may be performed within user interface. List of installed and available for installation modules is accessible for site administrator at /admin/modules path of a website. Based on experience gained in FCG's ICT department, it is not recommended to use it.

5.1.1 Drupal shell or drush

Any changes in code may possible cause critical errors and result in project breakdown. Usually to fix and/or recover the site developer requires access to server. Tools like drush – Drupal Shell – greatly simplify site administration. Following are several drush commands to work with modules:

drush pm-list – lists all installed modules and themes. Additional parameters allow to sort results, showing, for example, only enabled modules.

drush pm-install (or aliased version **drush en**) **<module name> –** installs module, requires system name of a module like devel or admin_toolbar.

drush dl – similar to drush en, but only downloads module files and it remains disabled.

drush pmu <module name> – uninstalls named module. This command does not remove files, but only disables the module.

If website experiences critical errors because of some modules, it would be easier to simply uninstall them with drush and then start troubleshooting or search alternative modules.

In addition to drush own commands, developers may add custom ones as it happens with some custom modules created for FCG's projects. Particular example exists in contacts module originally developed for kuntaliitto.fi by Wunder and later reused in other projects.

```
/**
 * Implements hook_drush_command().
 */
function contacts_drush_command() {
  $commands['import-contacts'] = [
    'description' => 'Imports and updates contact information.',
    'aliases' => ['imco'],
  ];

  return $commands;
}

/**
 * Drush contact import callback.
 */
function drush_contacts_import_contacts() {
     …
}
```

Listing 17. sample from kuntaliitto_contacts.drush.inc

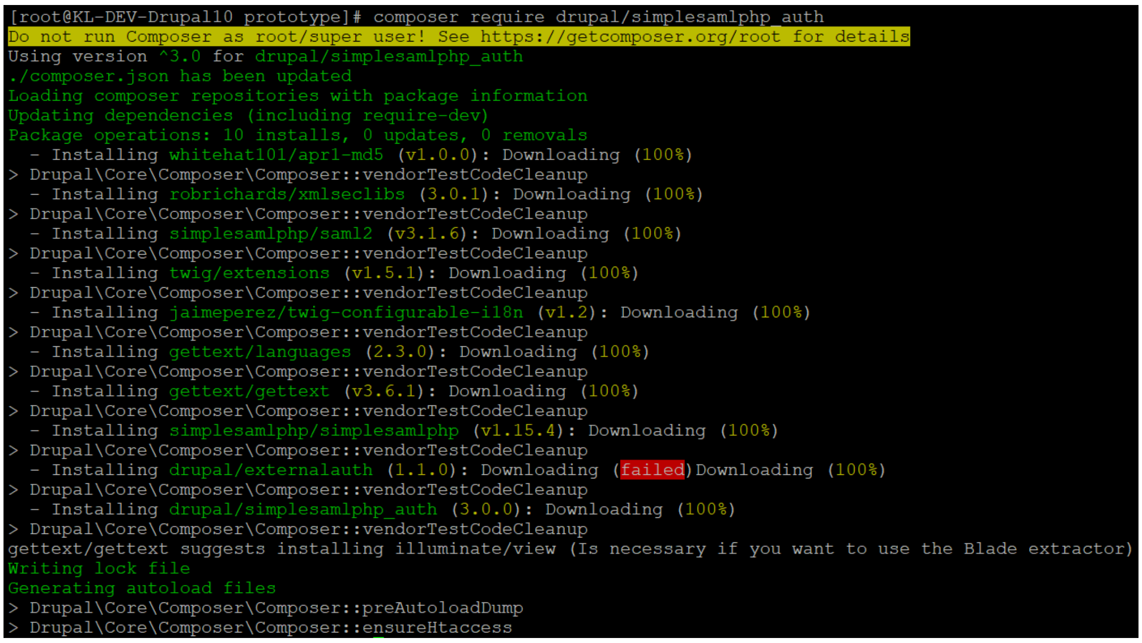
Example from Listing 17 presents custom drush command 'import-contacts' or aliased version of it 'imco'. Command may be executed manually within CLI of a server as 'drush imco' or added to cron tasks and then called according to schedule.

5.1.2 Composer

Benefits of Drupal shell can be hardly underestimated. However, developers should also use some other tools. Composer is one of them. In fact, it is generally recommended to learn and use Composer not only for Drupal, but also for any PHP based project.

> Composer is not a package manager in the same sense as Yum or Apt are. Yes, it deals with "packages" or libraries, but it manages them on a per-project basis, installing them in a directory (e.g. vendor) inside your project. By default it does not install anything globally. Thus, it is a dependency manager. It does however support a "global" project for convenience via the global command. (15).

Some Drupal modules require preparations like installation and configuration of additional software. Example of that is Search API Solr Search, which provides integration with Apache Solr as search engine. Often modules are dependent on external PHP libraries. Figure 12 illustrates installation of simpleSAMLphp Authentication module with Composer:



```
[root@KL-DEV-Drupal10 prototype]# composer require drupal/simplesamlphp_auth
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Using version ^3.0 for drupal/simplesamlphp_auth
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 10 installs, 0 updates, 0 removals
  - Installing whitehat101/apr1-md5 (v1.0.0): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing robrichards/xmlseclibs (3.0.1): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing simplesamlphp/saml2 (v3.1.6): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing twig/extensions (v1.5.1): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing jaimeperez/twig-configurable-i18n (v1.2): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing gettext/languages (2.3.0): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing gettext/gettext (v3.6.1): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing simplesamlphp/simplesamlphp (v1.15.4): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing drupal/externalauth (1.1.0): Downloading (failed)Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
  - Installing drupal/simplesamlphp_auth (3.0.0): Downloading (100%)
> Drupal\Core\Composer\Composer::vendorTestCodeCleanup
gettext/gettext suggests installing illuminate/view (Is necessary if you want to use the Blade extractor)
Writing lock file
Generating autoload files
> Drupal\Core\Composer\Composer::preAutoloadDump
> Drupal\Core\Composer\Composer::ensureHtaccess
```

Figure 12. module installation using composer

When composer require command issued, module is downloaded to modules folder and in addition, simplesamlphp library downloaded to vendor directory in the root of the project.

To be noticed, unlike drush, composer only downloads files, after that module still has to be enabled. It can be done with appropriate drush command or within administration toolbar.

5.2 Contributed modules

Some number of contributed modules were used in multiple FCG’s projects. Following is brief description of them:

Admin toolbar. Provides improved presentation of Drupal administration toolbar. Various items are available in drop-down lists, which makes them easier to access.

Better exposed filters. Extends available options for exposed filters in advanced settings of a View.

Block field. Provides field for Block references. Consequently, various blocks would become a part on a node or any other fieldable entity.

Devel. Combines various tools like massive generation of nodes, PHP code execution for development and testing purposes. It is recommended to uninstall this module when project is transferred into production server.

Examples. Another module useful in development environment. Contains various examples of custom code. Greatly helps in module development.

Google tag manager and/or **Google analytics.** Adds JavaScript snippets with provided tags for tracing and managing site analytics via Google tag manager and/or Google Analytics respectfully.

Metatag. Adds number of metatag fields to be used for SEO.

Paragraphs. With this module, it is possible to create more complex fields that contain multiple basic fields or other paragraphs. With known number of paragraphs designed, content managers would build content in more flexible way.

Pathauto. Provides automated generation of URL-aliases based on custom patterns. In particular this patterns would look like: /news/[year_of_publication]/[month_of_publication]/[title]. Requires token module.

Permissions by term. Restricts access to nodes and other entities based on taxonomy terms. These restrictions applicable both for groups and for single users.

Scheduler. Allows timed publishing or unpublishing of content.

Search API. Allows configurable indexing of content and integration with third-party software aimed to improve basic search on site.

SimpleSAMLphp authentication. Provides integration with SimpleSAMLphp application, which makes available use of single sign-on technology based on various authentication providers. In case of FCG and Kuntaliitto Microsoft’s Active Directory service is used.

SMTP Authentication support. Allows sending emails (e.g. webform submissions) through SMTP server.

Taxonomy manager. Advanced tool for managing taxonomy terms. Allows, in particular, adding of multiple terms with relationships within simple form.

Views Field View. Provides fields for View references. Quite similar to Block field.

Webform. Advanced tool for building custom webforms and managing their submissions. Supports various add-ons for spam filtering, different kinds of verification etc.

5.3 Custom modules

Some custom modules were developed for FCG’s projects by ICT department and subcontractors. Some of these solutions were also created in several variations. Following modules were reused or at least possible to reuse in other projects.

Social wall. Module fetches publications from known Twitter, Facebook and Instagram accounts. Initially used in kuntaliitto.fi and then in some other services.

Contacts. Populates and updates contact content type with data provided from Active Directory. Originally developed by Wunder for kuntaliitto.fi. Ready for use in other Kuntaliitto’s projects and with some modifications possible to apply for FCG’s services.

Shoutbox. This module is still under construction. Initially requested for Insaitti project. Simple functioning shoutbox was implemented by spring 2018, but the module did not meet additional requirements. At current state module is refurbished, but user interface is incomplete. Project development will continue in summer-autumn 2018.

5.4 Particular examples for custom module development

Drupal community has formed coding standards that claimed to be applicable for any version of Drupal. Many of these recommendations are still discussible and some are simply declarative alike “do it good” at current state. It is better for developer himself to be familiar with those standards to prevent serious mistakes.

Some of following examples taken from Fcg shoutbox module. Though its development is incomplete, it may illustrate some of the standards used in Drupal development.

5.4.1 Folder structure and namespaces

It is a good practice to keep contributed and custom modules in separate folders under modules folder in Drupal root. In addition to that, Drupal community recommends using proper inner folder structure and namespaces.

```
- fcg_shoutbox                        Root folder of the module
| | - config
| | | - schema
| | | | - fcg_shoutbox.schema.yml     Contains description of custom table
| - css
| | - fcg_shoutbox.css                Contains styles for the module
| - js
| | - fcg_shoutbox.js                 Contains javaScript for the module
| - src
| | - Form
| | | - fcg_shoutbox_form.php         Contains webform for the module
| | - Plugin
| | | - Block
| | | | - ShoutboxBlock.php           Contains rendering array of the module
| - fcg_shoutbox.info.yml             Contains information and some settings
| - fcg_shoutbox.install              Contains functions executed upon module
installation
| - fcg_shoutbox.libraries.yml        Defines libraries used in module
| - fcg_shoutbox.module               Contains some additional functions
| - fcg_shoutbox.routing.yml          Adds custom page to Drupal router
```

Listing 18. Folder structure of FCG shoutbox module

As seen from Listing 18, root folder of module contains subdirectories config for .yml files with module configuration, css and js for CSS and javascript respectively, src with its subfolders for PHP code. In addition, multiple .yml files like .info.yml also should be presented if the module root. Overall structure of a module is somewhat similar to structure of a theme. Declaration of namespace and use-statements are usually placed in the beginning of .php file as illustrated in Listing 19.

```
namespace Drupal\fcg_shoutbox\Plugin\Block;
 use Drupal\Core\Block\BlockBase;
 use Drupal\Core\Form\FormInterface;
 use Drupal\Core\Form\FormStateInterface;
 use Drupal\Core\Database\Connection;
 use Drupal\Core\Ajax\AjaxResponse;
 use Drupal\Core\Ajax\HtmlCommand;
 use Drupal\Core\Ajax\ReplaceCommand;
```

Listing 19. sample from ShoutboxBlock.php file

More information on use of namespaces and naming standards in Drupal can be found at drupal.org.

5.4.2 SQL queries

Drupal community strongly recommends to avoid classical SQL-quires like ‘SELECT * FROM table’ in custom modules. Core already contains its own methods for various operations on database. Initial implementation of Shoutbox module supposed usage of custom table to store short messages. Listing 20 illustrates creation of custom database table on module installation:

```
/**
 * Implements hook_schema().
 *
 * Defines the database tables used by this module.
 *
 * @see hook_schema()
 *
 */
function fcg_shoutbox_schema() {
  $schema['fcg_shoutbox'] = array(
    'description' => 'Stores shoutbox entries.',
    'fields' => array(
      'id' => array(
        'type' => 'serial',
        'not null' => TRUE,
        'description' => 'Primary Key: Unique person ID.',
      ),
      'uid' => array(
        'type' => 'int',
        'not null' => TRUE,
        'default' => 0,
        'description' => "Uid of shouter.",
      ),
      'message' => array(
        'type' => 'varchar',
        'length' => 255,
        'not null' => TRUE,
        'default' => '',
        'description' => 'Message.',
      ),
       'time' => array(
        'type' => 'int',
        'default' => 0,
        'description' => 'Time',
      ),
    ),
    'primary key' => array('id'),
    'indexes' => array(
      'uid' => array('uid'),
      'message' => array('message'),
      'time' => array('time'),
    ),
  );

  return $schema;
}
```

Listing 20. Sample from fcg_shoutbox.install file

Custom table ‘fcg_shoutbox’ has four columns: id, which serves as Primary key, uid, that contains id of a user who posted a new message, message itself and time when it was added. Further operations – inserting new records and fetching existing data - also implemented with Drupal functions.

```
public function submitForm(array &$form, FormStateInterface $form_state){
     $shout = $form_state->getValue('shout');
     if($shout != $placeholder){
          $fields = array(
               'uid' => \Drupal::currentUser()->id(),
               'message' => $shout,
               'time' => time(),
          );
          db_insert('fcg_shoutbox')->fields($fields)->execute();
     }
}
```

Listing 21. Form submission function from fcg_shoutbox_form.php

Function shown in Listing 21 is executed on form submission. Variable $form_state contains form submission data. If-statement used to ensure, that message is not a placeholder from the form. Function db_insert() uses array, containing user id, text of the message and current time, as a parameter. On its execution, new record is added to table.

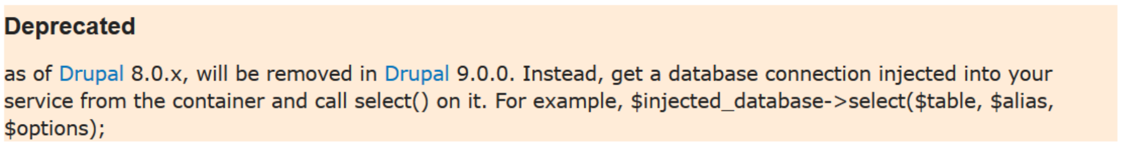
Messages from ‘fcg_shoutbox’ table were supposed to be rendered in a custom block on the front-page of Insaitti. Sample of code that fetches content from database and forms part of markup for that block exposed in Listing 22.

```
public function prepare_markup($limit){
     $query = db_select('fcg_shoutbox', 's')->fields('s', array('uid', 'mes-
sage', 'time'));
     $result = $query->execute();
     $shouts = array();

     foreach ($result as $row){
          $user=user_load($row->uid);
          $timestamp = date("d.m.y H:i", $row->time);
          $name = $user->getUsername();
          $cname = ($row->uid == \Drupal::currentUser()->id() ? "myname" :
"uname");
          $shouts[] = "<li><div class='shout-head'><span class='shout-
time'>$timestamp</span> <span class=$cname><strong><a class='aname'
href='#'>$name</a>:</strong></span></div><div class='shout-body'>$row->mes-
sage</div></li>";
     }

     if(sizeof($shouts)>$limit){
          $shouts = array_slice($shouts, 0-$limit);
     }
     $shouts = array_reverse($shouts);

     foreach($shouts as $shout){
          $markup .= $shout;
     }

     return $markup;
}
```

Listing 22. sample of code from ShoutboxBlock.php

Function prepare_markup() requires variable $limit as parameter, which is a number of messages to be printed. Execution of db_select query results in a multidimensional array, containing user ids, text messages and date of their creation. After all sorting actions taken, prepare_markup() function returns string containing five latest shoutbox messages, which then will be included into rendering array.

It has to be admitted that now both db_select() and db_insert(), as well as some other similar functions, are marked as deprecated. Drupal API documentation contains notification about that shown in Figure 13.



Deprecated

as of Drupal 8.0.x, will be removed in Drupal 9.0.0. Instead, get a database connection injected into your service from the container and call select() on it. For example, $injected_database->select($table, $alias, $options);

Figure 13. Notification from Drupal 8 documentation about deprecation of db_select() function

This means, that shoutbox module in its initial form, still being feasible for Drupal 8, is incompatible within Drupal 9 project. Drupal is evolving constantly, and deprecation of some methods within its core means that custom modules often have to be renovated.

5.4.3 Rendering arrays

Forms, blocks, custom pages. Various Drupal elements may be prepared and printed programmatically. Rendering arrays are serving for those purposes. Example in Listing 23 illustrates simple custom form used in Shoutbox module:

```
public function buildForm(array $form, FormStateInterface $form_state) {

      $form['shout'] = array(
            '#type' => 'textfield',
            '#size' => '40',
            '#default_value' => $placeholder,
      );

      $form['actions']['submit'] = array(
            '#type' => 'submit',
            '#value' => $this->t('Lähetä'),
            '#button_type' => 'primary',
            '#submit' => array(array($this, 'submitForm')),
      );
      return $form;
}
```

Listing 23. sample of code from fcg_shoutbox_form.php

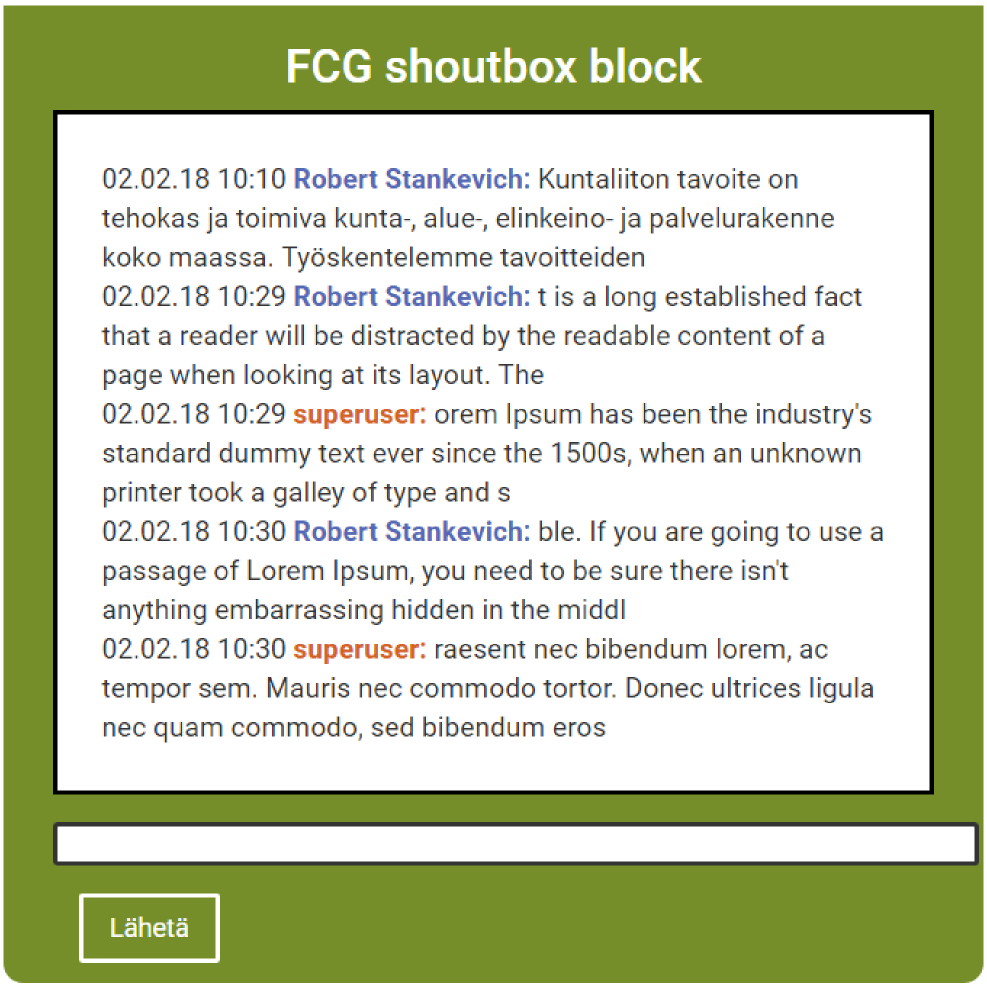
This form has only two elements: text field and submit button. The latter has property '#submit' pointing to function submitForm(), which was mentioned earlier. More elements listed in Drupal documentation at drupal.org.

Following sample shown in Listing 24 also taken from Shoutbox module. It forms rendering array of a custom block, containing latest messages, shoutbox frontend library and from earlier presented in Listing 23.

```
      public function build() {

            $build['#attached']['library'][] = 'fcg_shoutbox/shoutbox';
            $limit = $this->configuration['shoutbox_block_limit'];
            $shouts = array(
                  '#type' => 'markup',
                  '#prefix' => '<ul class="shoutbox">',
                  '#markup' => $this->prepare_markup($limit),
                  '#suffix' => '</ul>',
            );

            $form = \Drupal::formBuilder()->getForm('\Drupal\fcg_shout-
box\Form\fcg_shoutbox_form');

            $build['shoutbox'] = [
                  'form' => $form,
                  'shouts' => $shouts,
            ];
            return $build;
      }
```

Listing 24. sample of code from ShoutboxBlock.php file

Finally custom Shoutbox block is rendered as it is shown in Figure 14.


FCG shoutbox block

02.02.18 10:10 **Robert Stankevich:** Kuntaliiton tavoite on tehokas ja toimiva kunta-, alue-, elinkeino- ja palvelurakenne koko maassa. Työskentelemme tavoitteiden
02.02.18 10:29 **Robert Stankevich:** t is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The
02.02.18 10:29 **superuser:** orem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and s
02.02.18 10:30 **Robert Stankevich:** ble. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middl
02.02.18 10:30 **superuser:** raesent nec bibendum lorem, ac tempor sem. Mauris nec commodo tortor. Donec ultrices ligula nec quam commodo, sed bibendum eros

Figure 14. Presentation of initial version of Shoutbox module

Current refurbished version of Shoutbox requires fixes in its user interface. Since the same customer has other urgent projects, development of this module is postponed.

5.4.4 Generating nodes

Often web services need some sort of automated content generation. In some FCG's Drupal 8 projects nodes are created programmatically with help of various modules. One of them – contacts – receives data in JSON format from PHP script created by Hannu Niemi. This JSON contains information about company's employees fetched from Active Directory. Contacts module had multiple implementations. Module developed by Wunder for kuntaliitto.fi is considered as comprehensive solution now. ICT department also designed another variation of it presented in Listing 25.

```
<?php
/**
 * @file
 * Contains \Drupal\example\Controller\ExampleController.
 */
namespace Drupal\customcontacts\Controller;
use Drupal\Core\Controller\ControllerBase;
use Drupal\node\Entity\Node;

class CustomcontactsStartController {

     public function startAction() {
          $str = "";
          $json = file_get_contents($urltosource);
          $staff = json_decode($json);
          foreach($staff as $key => $item){
               if($item->division != $divisionname){
                    unset($staff->$key);
               }
          }
          $query = \Drupal::entityQuery('node');
          $query->condition('type', 'contact');
          $nids = $query->execute();

          if(!empty($staff) && !empty($nids)){
               $str.="<ol>";
               foreach ($nids as $nid) {
                    $node = Node::load($nid);
                    $point = $node->title->value;
                    if(array_key_exists($point, $staff)){
                         $str .= updateNode($node, $staff->$point);
                         unset($staff->$point);
                    }
                    else{
                         $str .= "DELETE " . $node->title->value;
                         $node->delete();
                    }
               }
               $str .= "</ol>";
          }

          $str .= "<ol>";
          foreach($staff as $key => $item){
               $node = Node::create([
                    'type'          => 'contact',
                    'title'         => $key,
                    'langcode'      => 'fi',
               ]);
               $node->save();
               $str .= "<li>" . $node->title->value . " contact cre-
ated</li>";
               $str .= updateNode($node, $item);
          };
          $str .= "</ol>";
          return [
               '#markup' => $str,
          ];
     }
}
```

Listing 25. sample of code from custom contacts module

This sample of code fetches data from remote source in JSON format. Then JSON is decoded into PHP array. For the project only members of particular division named in $divisionname variable were needed, therefore other item were removed from the array.

Then, node ids of existing contacts were received from database. After that node titles were checked against keys of array. If check fails, it means that corresponding contact information was removed from Active Directory and node should be deleted. Otherwise node contents updated with updateNode() function illustrated in Listing 26.

```
function updateNode($node, $item){
	$node->field_name->value = utf8_decode($item->givenname);
	$node->field_surname->value = utf8_decode($item->sn);
	$node->field_phone->value = $item->mobile;
	$node->field_email->value = $item->mail;
	$node->field_responsibilities->value = utf8_decode(clearInfo($item-
>info));
	$node->save();
	if($node && !$node->hasTranslation('sv')){
		$arr = $node->toArray();
		$translated_fields = [];
		$translated_fields[field_responsibilities] = [
			'value' => utf8_decode(clearInfo($item->extensionattribute7))
		];
		$tarr = array_merge($arr, $translated_fields);
		$node->addTranslation('sv', $tarr)->save();
		return '<li>'.$node->title->value.' updated and translation
added</li>';
	}
	else if($node && $node->hasTranslation('sv')){
		$tnode = $node->getTranslation('sv');
		$tnode->field_responsibilities->value = utf8_decode(clearInfo($item-
>extensionattribute7));
		return '<li>'.$node->title->value.' updated and translation up-
dated</li>';
	}
	else{
		return '<li>nothing to update...</li>';
	}
}

function clearInfo($toclear){
	$clear = str_replace('[p]','<p>', $toclear );
	$clear = str_replace('[b]','<b>', $clear);
	$clear = str_replace('[/b]','</b>', $clear);
	$clear = str_replace('[/p]','</p>', $clear);
	return $clear;
}
```

Listing 26. functions updateNode() and clearInfo() from custom contacts module

Function updateNode() receives two parameters: node object and item from the array, containing data fetched from AD. When all fields receive new values, node is saved with Node class method save(). Web service using this module was available in two language versions (Finnish and Swedish). Therefore, further code also applies translation of the content. Additional function clearInfo() replaces some known tags in square brackets with proper html-tags.

As seen from Listing 25, function updateNodes() also used creation of new nodes. That part of code is executed if $staff array still contains some items after updating existing contact information, which means that new employees were added to Active Directory since previous update.

Whole script was designed to be executed when particular page is visited. Therefore, it prepares an output that will list all updated, deleted and newly created contacts.

6 Site building

As a rule, site builders deal with site configuration, including definition of content types, creation of views, application of settings for various modules etc. This work rarely involves any knowledge of programming, but requires good understanding of Drupal itself. Site building may seem to be less complicated than backend development, but has equal or even greater importance for the whole project.

6.1 Content types

Content types in Drupal answer the question what kind of data will be contained in different nodes of a website. When developer defines a content type, he picks fields that suits upcoming content best. Drupal core already contains variety of fields that would be enough for an average site: different kind of text, numbers, files and images, lists and references. In various FCG's projects, some content types were appearing repeatedly. To reduce unnecessary repetitions, prototype for future sites should contain those most used content types.

Basic page. Serves for the highest level of site navigation. Nodes of this type are somewhat keystones for the whole service. Title, menu link and body text, if needed, are the minimal requirement for this content type. In addition, it may have main media field for image or video, if site-wide banners or carousels are not supposed to be used in a project. Another field, that seems to be mandatory for basic page, is 'liftup'. It would be a view that gathers previews of nodes based on some rules like particular content type or taxonomy terms. It is logical, that main page for News section of a site would present the latest news, and Services or Products – promote the most important services or products respectively.

News and/or **Blogs.** Traditionally one of the largest sections of a web service. For example on May 2018, insaitti.kuntaliitto.fi contained more than 400 of nodes of Ajankohtaista (news) type. Twice more than Asiantuntiaja sivu (expert content) that serves as the main content type for other sections of Insaitti. Structure of this content type supposed to be quite simple: main media and body text, possibly some taxonomy terms and reference to author (especially for blog articles), if it does not contradict project specifications. Finally, this content type may contain fields for scheduler module to add timed publication feature.

Events. Another important content type for projects developed by FCG. Nodes of this type should describe upcoming and passed events, as well as date, time and place of conduct. It seems to be a good practice to provide information on location of new events using integration with such services as Google maps or Open street maps. Drupal community already has some modules to provide this feature. However, this topic requires additional investigation and cautious since combined with some other data it may lead to complications with GDRP.

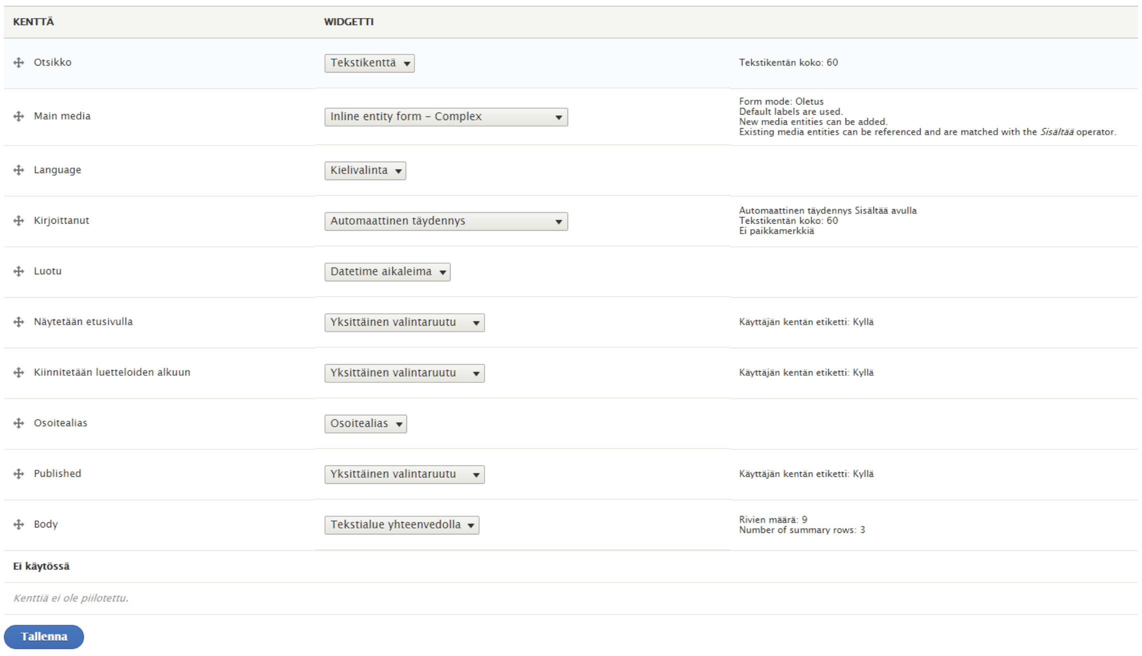
Service and/or **Product**. Many of FCG's web services are aimed to promote products and services of the company and its departments. Specific content type should be designed to describe them. Such content type may contain large variety of fields: basic text and media, as well as files, references, taxonomy terms, links and, possibly, liftups and web forms. To provide complex fields, paragraphs module would serve greatly for this content type. Depending on the project goals, this section of a website maybe the most important for the customer, and therefore the most attention should be paid to it.

Article or **secondary basic page.** Often happens, that some of the contents of a future project are not clearly specified. Some sort of utility content type would be needed in such cases. On one hand, for basic page would serve for this purpose as well, on the other, further clarification of requirements, that occurs quite often, would result in additional difficulties for content managers. Particular example: when pages in top-level navigation would contain significantly large number of additional fields and settings, that would not be needed for subpages at all and vice versa.

Contacts. This content type would serve as placeholder for Contacts module and contain such fields as name and surname of employee, his title, unit, contact information etc. If Contacts module used, then this content type is populated programmatically.

6.2 Fields and widgets

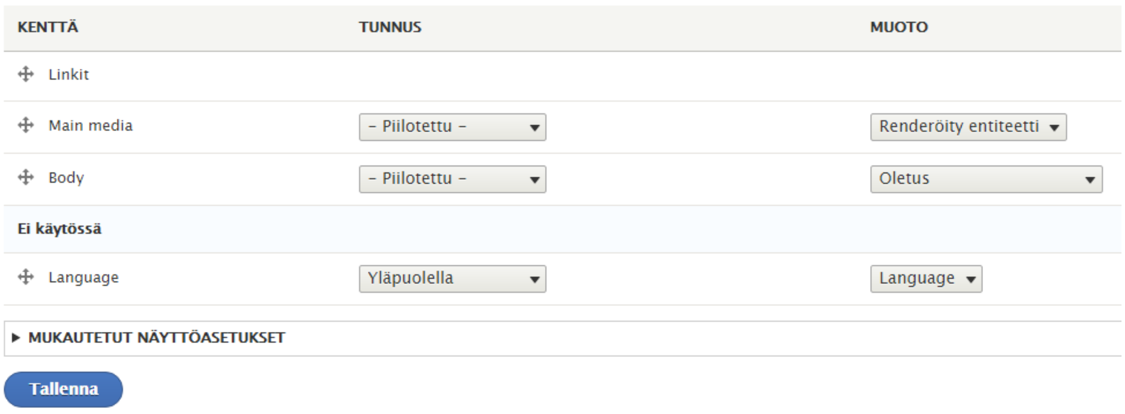
In Drupal many fields have multiple options for presentation on both editing and demonstration sides. Form display in first case, and simply display in the second. Both of them are configurable by site administrators with sufficient permissions. Same rules usually work for any fieldable entity like content types, taxonomy terms, paragraphs or media.



KENTTÄ	WIDGETTI	
Otsikko	Tekstikenttä	Tekstikentän koko: 60
Main media	Inline entity form - Complex	Form mode: Oletus Default labels are used. New media entities can be added. Existing media entities can be referenced and are matched with the *Sisältää* operator.
Language	Kielivalinta	
Kirjoittanut	Automaattinen täydennys	Automaattinen täydennys Sisältää avulla Tekstikentän koko: 60 Ei paikkamerkkiä
Luotu	Datetime aikaleima	
Näytetään etusivulla	Yksittäinen valintaruutu	Käyttäjän kentän etiketti: Kyllä
Kiinnitetään luetteloiden alkuun	Yksittäinen valintaruutu	Käyttäjän kentän etiketti: Kyllä
Osoitealias	Osoitealias	
Published	Yksittäinen valintaruutu	Käyttäjän kentän etiketti: Kyllä
Body	Tekstialue yhteenvedolla	Rivien määrä: 9 Number of summary rows: 3
Ei käytössä		
Kenttiä ei ole piilotettu.		

Figure 15. Form display management tab

Example of form display managements tab is shown in Figure 15. Developer may choose one of the widgets for a field. For instance, core field for taxonomy tags may have presentation as one of autocomplete text fields, dropdown list or radio buttons. Additional modules or combinations of modules may provide more optional widget for basic and custom fields. Same widgets also have additional settings like preview image stile for image field. Site administrator may define order in which fields appear on editing page and even hide some of them, which is useful for automated or predefined contents.



KENTTÄ	TUNNUS	MUOTO
Linkit		
Main media	– Piilotettu –	Renderöity entiteetti
Body	– Piilotettu –	Oletus
Ei käytössä		
Language	Yläpuolella	Language

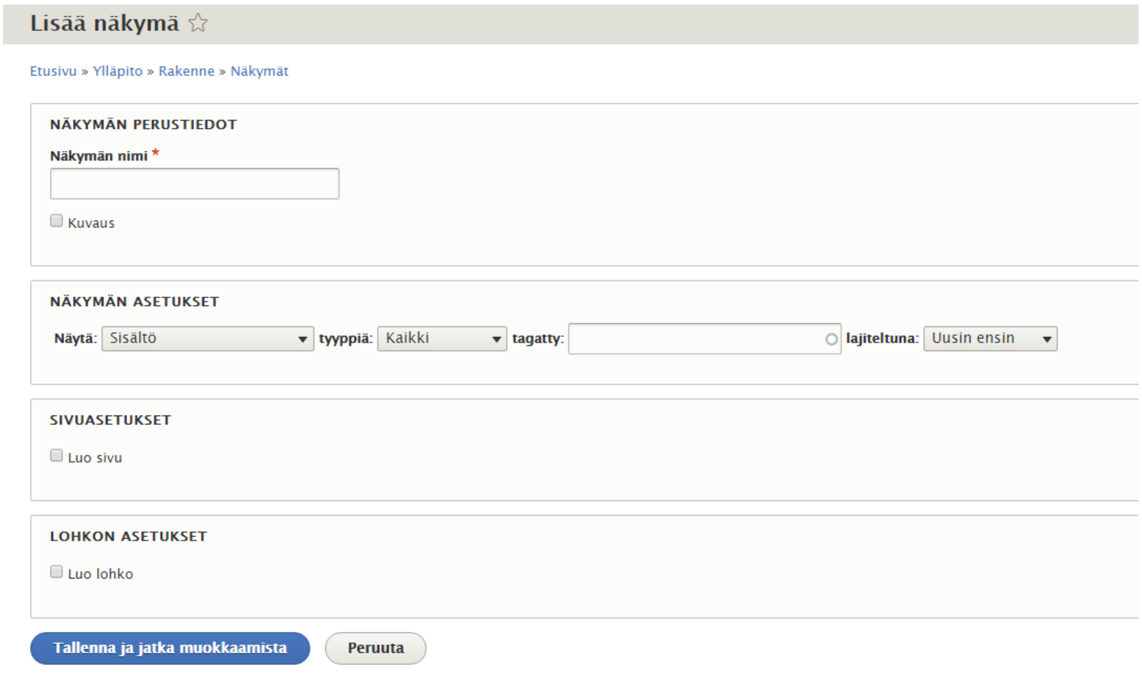
▸ MUKAUTETUT NÄYTTÖASETUKSET

Figure 16. Display managing tab

Display defines how the content is exposed to page visitors. As seen from Figure 16, it works the same way as form display management. Same entity may have multiple display options. As an example, Full content may present the whole content of a node, while preview would have only title and sample of body text. Display options of fields should be taken into consideration in custom template design. For example, format of image field may be set as an URL to file, and then in template developer would have additional options like using image for a background of a page or applying additional properties into <img> tag.

6.3 Views

Views used to be a contributed module for Drupal 7. In Drupal 8, it became a part of the core. It is an ultimate tool for site builders. Views may alter presentation of nearly everything in Drupal 8. They greatly serve for lifting up contents from various nodes based on various filters.


Lisää näkymä ☆

Etusivu » Ylläpito » Rakenne » Näkymät

NÄKYMÄN PERUSTIEDOT

Näkymän nimi *

☐ Kuvaus

NÄKYMÄN ASETUKSET

Näytä: Sisältö tyyppiä: Kaikki tagatty: lajiteltuna: Uusin ensin

SIVUASETUKSET

☐ Luo sivu

LOHKON ASETUKSET

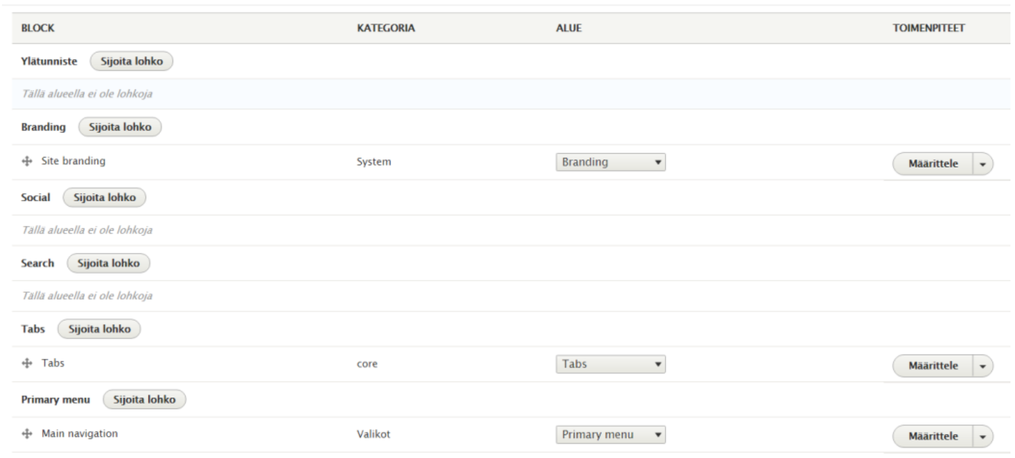
☐ Luo lohko

Figure 17. Creating new view

As shown in Figure 17, when creating a new view, site builder defines its name and initial settings. In particular: what entities of which type and order should be shown, and how the view will be rendered: as a block or single page.

6.4 Regions and blocks

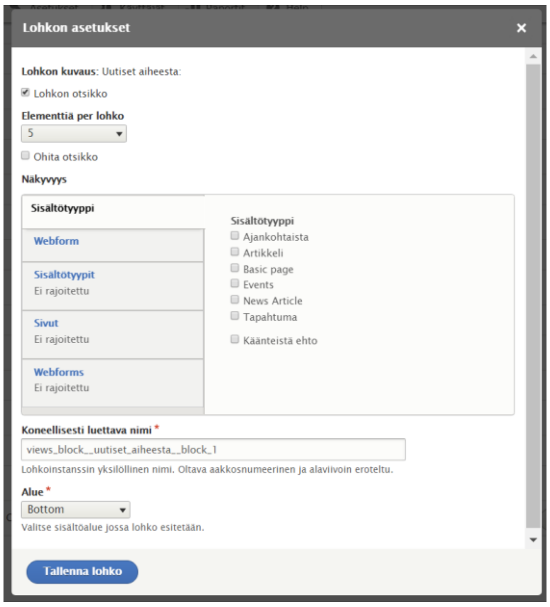
When blocks are created programmatically or with help of administration tools, they still have to be allocated in site layout. When blocks are referenced within node fields, they are added to main content, which itself is a block. Site builder can allocate particular blocks in regions, defined by theme, using Block layout tab shown in Figure 18.



BLOCK	KATEGORIA	ALUE	TOIMENPITEET
Ylätunniste Sijoita lohko			
Tällä alueella ei ole lohkoja			
Branding Sijoita lohko			
Site branding	System	Branding	Määrittele
Social Sijoita lohko			
Tällä alueella ei ole lohkoja			
Search Sijoita lohko			
Tällä alueella ei ole lohkoja			
Tabs Sijoita lohko			
Tabs	core	Tabs	Määrittele
Primary menu Sijoita lohko			
Main navigation	Valikot	Primary menu	Määrittele

Figure 18. Block layout tab

Various blocks may have additional settings like hiding or exposing them for particular nodes based on their URL aliases, displaying or overriding block’s title, setting number of printed results for a View etc. Figure 19 illustrates single block configuration.



Lohkon asetukset

Lohkon kuvaus: Uutiset aiheesta:

[✓] Lohkon otsikko

Elementtiä per lohko
5

[ ] Ohita otsikko

Näkyvyys

Sisältötyyppi	**Sisältötyyppi**
Webform	[ ] Ajankohtaista
Sisältötyypit Ei rajoitettu	[ ] Artikkeli [ ] Basic page [ ] Events
Sivut Ei rajoitettu	[ ] News Article [ ] Tapahtuma [ ] Käänteistä ehto
Webforms Ei rajoitettu	

Koneellisesti luettava nimi *
views_block__uutiset_aiheesta__block_1
Lohkoinstanssin yksilöllinen nimi. Oltava aakkosnumeerinen ja alaviivoin eroteltu.

Alue *
Bottom
Valitse sisältöalue jossa lohko esitetään.

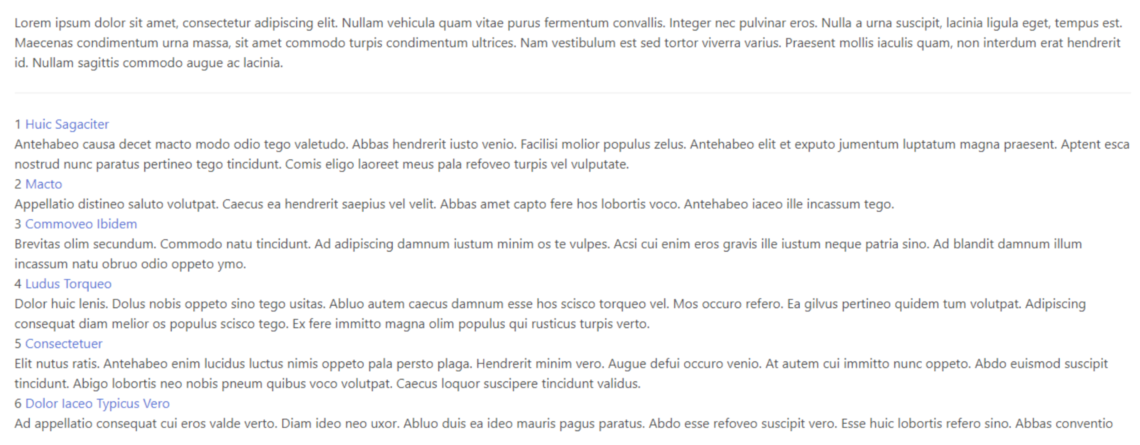
Tallenna lohko

Figure 19. Settings of a block

Same blocks may be assigned to multiple regions. It appeared quite useful combined with Search API module in several FCG’s projects.

6.5 Taxonomy

Taxonomy terms may have a great use for content-rich web services. They can generally improve content accessibility and organization of contents. One of the simplest examples is listing content that share same tag at taxonomy term page. This can be done by implementing a view that takes this term as a parameter and gathers all nodes using it. Results would look as example shown in Figure 20.

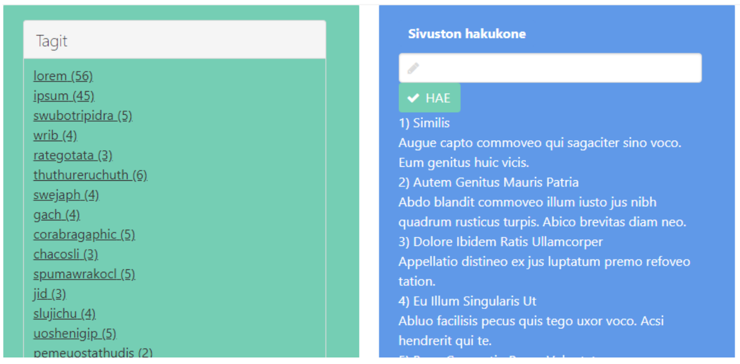


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam vehicula quam vitae purus fermentum convallis. Integer nec pulvinar eros. Nulla a urna suscipit, lacinia ligula eget, tempus est. Maecenas condimentum urna massa, sit amet commodo turpis condimentum ultrices. Nam vestibulum est sed tortor viverra varius. Praesent mollis iaculis quam, non interdum erat hendrerit id. Nullam sagittis commodo augue ac lacinia.

1 Huic Sagaciter
Antehabeo causa decet macto modo odio tego valetudo. Abbas hendrerit iusto venio. Facilisi molior populus zelus. Antehabeo elit et exputo jumentum luptatum magna praesent. Aptent esca nostrud nunc paratus pertineo tego tincidunt. Comis eligo laoreet meus pala refoveo turpis vel vulputate.
2 Macto
Appellatio distineo saluto volutpat. Caecus ea hendrerit saepius vel velit. Abbas amet capto fere hos lobortis voco. Antehabeo iaceo ille incassum tego.
3 Commoveo Ibidem
Brevitas olim secundum. Commodo natu tincidunt. Ad adipiscing damnum iustum minim os te vulpes. Acsi cui enim eros gravis ille iustum neque patria sino. Ad blandit damnum illum incassum natu obruo odio oppeto ymo.
4 Ludus Torqueo
Dolor huic lenis. Dolus nobis oppeto sino tego usitas. Abluo autem caecus damnum esse hos scisco torqueo vel. Mos occuro refero. Ea gilvus pertineo quidem tum volutpat. Adipiscing consequat diam melior os populus scisco tego. Ex fere immitto magna olim populus qui rusticus turpis verto.
5 Consectetuer
Elit nutus ratis. Antehabeo enim lucidus luctus nimis oppeto pala persto plaga. Hendrerit minim vero. Augue defui occuro venio. At autem cui immitto nunc oppeto. Abdo euismod suscipit tincidunt. Abigo lobortis neo nobis pneum quibus voco volutpat. Caecus loquor suscipere tincidunt validus.
6 Dolor Iaceo Typicus Vero
Ad appellatio consequat cui eros valde verto. Diam ideo neo uxor. Abluo duis ea ideo mauris pagus paratus. Abdo esse refoveo suscipit vero. Esse huic lobortis refero sino. Abbas conventio

Figure 20. Example of taxonomy term page

Same by its nature would be a “liftup” based on particular term. Similar view rendered as a block would serve as an additional navigation for website. Used alongside with Search API and facets taxonomy terms would also serve for filtering search results as shown in Figure 21.



Tagit

lorem (56)
ipsum (45)
swubotripidra (5)
wrib (4)
rategotata (3)
thuthureruchuth (6)
swejaph (4)
gach (4)
corabragaphic (5)
chacosli (3)
spumawrakocl (5)
jid (3)
slujichu (4)
uoshenigip (5)
pemeuostathudis (2)

Sivuston hakukone

HAE

1) Similis
Augue capto commoveo qui sagaciter sino voco. Eum genitus huic vicis.
2) Autem Genitus Mauris Patria
Abdo blandit commoveo illum iusto jus nibh quadrum rusticus turpis. Abico brevitas diam neo.
3) Dolore Ibidem Ratis Ullamcorper
Appellatio distineo ex jus luptatum premo refoveo tation.
4) Eu Illum Singularis Ut
Abluo facilisis pecus quis tego uxor voco. Acsi hendrerit qui te.

Figure 21. Search page with facets

For existing FCG’s services, not very much attention was paid to taxonomy. Further investigation would become beneficial for future projects.

7 Conclusion

Technologies are evolving rapidly and web developers face new challenges continuously. Tools have to change to satisfy new needs. Drupal is one of such tools. Every new project means new knowledge and solutions. Every developer may become a part of community and help to improve the system.

ICT department in FCG works with around 20 websites based on Drupal. All these projects had their challenges, but still they did not use the full potential of the system. Integrations, already mentioned and partially illustrated in this work, allow automated generation of content and other useful features. As a part of complex system, Drupal may serve many roles. Not only be a single website, but also provide data for various applications built on top of it, including other web or mobile services.

As any software product, Drupal is not perfect, and there are various concerns. In particular, security of this system raised reasonable worries after two serious breaches occurred in spring 2018. The scale of those threats was so great, that they got name ‘Drupalgeddon’. Experts say that of the problem is somewhere deep in core logics of Drupal. Community responded immediately and provided security patches even for unsupported versions of core. Still many of websites were compromised. Such incidences badly affect the reputation of Drupal. On the other hand, hardly any of software products would ever guarantee 100% of protection.

As a rule, development of projects in FCG not ends at the point of their publication. Concept of sustainable development is in use. ICT department tracks current state of projects under its responsibility and applies changes required by customers. Drupal 8 already replaced some other platforms, that served company for many years. It is logical to assume, that at some point of time Drupal will be changed to other system, if needed. However, considering experience gained during last years and possible improvements, Drupal 8 remains a highly perspective platform for future web services developed for FCG and its partners.

ICT department in FCG will continue work on existing and upcoming Drupal 8 project and enlarge knowledge base on this system. Some of particular objectives for further development already mentioned in this work. Environment has to be reevaluated and updated to meet changing requirements for Drupal.

Another major issue is automation. Adoption and further improvement of prototype build in general would have positive impact on standardizing of software process in the company. Growing number of projects makes such actions as massive core updates unnecessary complicated. Continuous integration is highly beneficial practice, but in case of Drupal projects, it requires accurate and versatile investigation since the solution itself should be significantly fail-safe and exclude any harm for live projects.

Majority of modules, developed by FCG and subcontractors, was designed to meet requirements of particular projects. Possibly findings, made in the company, would serve for creation of new modules. If these projects will present significant quality and reusability, they would also become publicly available. In such case, ICT department of FCG would become an important part of Drupal community.

References

1. Using Drupal. [book auth.] Addison Berry, Bruno De Bondt Angela Byron. *Using Drupal.* s.l. : O'Reilly Media Inc, 2012, pp. 1, 71.

2. search-phrase "most popular CMS". *Google.* [Online] [Cited: 04 11, 2018.] https://www.google.com/search?q=most+popular+CMS&ie=utf-8&oe=utf-8&client=firefox-b-ab.

3. WordPress.com. *WordPress.com.* [Online] [Cited: 05 27, 2018.] https://wordpress.com/.

4. CMS Usage Statistics. *builtwith.com.* [Online] [Cited: 04 09, 2018.] https://trends.builtwith.com/cms.

5. Drupal release timeline. *commons.wikimedia.org.* [Online] [Cited: 04 15, 2018.] https://commons.wikimedia.org/wiki/File:Drupal_release_timeline.png.

6. Innoppl, Inc. Why Drupal 8 Is Tearing Up The Web Development Scene In 2016 [Infographic]. *thelocalbrand.com.* [Online] [Cited: 04 15, 2018.] https://thelocalbrand.com/top-13-drupal-8-features-make-website-easy-develop-2016/.

7. Understanding Drupal. *Drupal 8 guide.* [Online] [Cited: 04 11, 2018.] https://www.drupal.org/docs/8/understanding-drupal-8/overview.

8. Server Usage Statistics. *BuitWith.* [Online] [Cited: 04 23, 2018.] https://trends.builtwith.com/server.

9. CentOS vs Red Hat. *SimilarTech.* [Online] [Cited: 04 23, 2018.] https://www.similartech.com/compare/centos-vs-red-hat.

10. CentOS vs Ubuntu. *similartech.com.* [Online] [Cited: 04 23, 2018.] https://www.similartech.com/compare/centos-vs-ubuntu.

11. Drupal | NGINX. *nginx.com.* [Online] [Cited: 04 20, 2018.] https://www.nginx.com/resources/wiki/start/topics/recipes/drupal/.

12. Database server. *Drupal 8 guide.* [Online] [Cited: 04 20, 2018.] https://www.drupal.org/docs/8/system-requirements/database-server.

13. Drupal 8 php requirements. *Drupal 8 guide.* [Online] [Cited: 04 21, 2018.] https://www.drupal.org/docs/8/system-requirements/drupal-8-php-requirements.

14. Bulma. *Bulma.* [Online] [Cited: 05 10, 2018.] https://bulma.io/.

15. Introduction - Composer. *Composer.* [Online] [Cited: 05 20, 2018.] https://getcomposer.org/doc/00-intro.md.