Graduation report

12/06/2022

Graduation assignment

Company	Fixico B.V.
Student	Mark Kravchuk
University	Saxion University of Applied Sciences
Period	February 2022 - July 2022

Table of Contents

1. Introduction

As a Software Engineering student I was able to complete my graduation assignment at Fixico B.V. in Amsterdam. Fixico is the largest vehicle repair platform in Europe. With Fixico, car owners and business partners can easily find a suitable repair company and save on repair costs. Their platform connects supply and demand in the car repair industry.

The assignment I was completing was to analyze existing Fixico websites and especially their drawbacks to design a new website system that would solve and handle maintainer problems.

This document explains the assignment context, the approach taken per each research question, their outcomes, and the design of the solution made. This document is aimed to inform the reader regarding the path, decisions and expectations the student will follow during completion of his graduation assignment.

	Company supervisor	University supervisor	Graduation intern
Name	Gareth Dutrieux	Jan Jaap Sandee	Mark Kravchuk
Position	Projects Manager	Teacher	Graduation Intern
Email	gareth@fixico.nl	j.m.j.sandee@saxion.nl	mark.kravchuk@fixico.nl

Table 1. Graduation contacts

2. Graduation assignment

2.1 Introduction

Fixico is the largest vehicle repair platform in Europe. With Fixico, car owners and business partners can easily find a suitable repair company and save on repair costs. Their platform connects supply and demand in the car repair industry in a completely new and innovative way.

Fixico innovative solutions offer clear benefits for the entire market. Whether customer need them as a car owner or for business, Fixico put ease of use, quality and transparency for all our users first:

- Car owners: Finding the right repair shop is often a tedious and time-consuming task for car owners. Prices can be unclear, warranty is not always included and there are often nasty surprises afterwards. With Fixico, car owners can easily and quickly compare the best repair shops. This way our users can make the best choice with peace of mind, and they know in advance exactly what the damage will be for their wallet.
- Repair shops: Partner repair shops can respond quickly and easily to requests from local car owners via Fixico platform by submitting quotes. Via Fixico they can increase turnover, increase online findability, collect reviews and easily optimize workplace occupancy. Fixico only works with recognized repair shops that meet high quality requirements and offer a minimum of 4 years warranty.
- Business partners: Fixico has unique and innovative solutions for insurers, lease companies, rental companies, fleets and other organizations that deal with car damage on a daily basis. Fixico service is aimed at reducing the total cost of claims, shortening lead times, increasing customer satisfaction and being able to work a lot more efficiently.

2.2 Problem definition

Fixico currently has a wide variety in the website assortment. Right now, 7 business websites are active, and they are built using 3 different content management systems (CMS). While 20 accompanied repositories are handing the source code for them.

Such variety of websites are settled using different CMS (Content management systems) creates multiple maintaining problems:

- Design themes, each CMS used at Fixico has different implementation, so therefore each theme has its source code on different frameworks. This is a problem for the engineering team to make changes (updates) to the theme.
- Updating the core modules used within the website. Due to different technological stack, all the updates are made differently and vary on each CMS that's being used.
- The technological stack, that implies engineers to know a variety of different skill sets for the changes.
- Content Management System implementations, that makes impossible to make changes to some website elements that could be easily changed from the admin panel.

Therefore, right now tickets are being created by other teams within the company for engineers to make desired hardcoded changes.
- Admin accounts, as being said that Fixico has 7 websites, means there are 7 different admin panels where the credential and access per each website differs, this makes employees from different departments a headache to gain admin rights to the right website.
- Technological implementation: current deployment method is hard to complete due to its git-sync implementation. It requires many unnecessary steps that can be handled automatically.

2.3 The assignment

Based on the problems of maintaining current websites, Fixico wants to create a new website base where eventually all websites would be migrated. This base should be implemented with the described problems above in mind.

The technological stack was already defined for the student. And those are:

- The core should be built using Statamic - a Laravel framework on PHP
- Antlers - templating engine will be used for creating template types
- Alpine.js – JavaScript framework for composing JS behavior in templates
- Tailwind CSS – stylesheet framework created for optimization of the stylesheet sizes

Basically, the company has chosen these technologies, because Fixico is already using Tailwind CSS framework within the software they provide, Alpine JS promises to have similar constraints as React (JavaScript front-end framework) which is also widely used within the company.

Statamic is a great framework that satisfies the company's requirements, like:

- It uses PHP programming language and Laravel framework, company makes active use of the Laravel and the Statamic is a nice fit to the company's tech stack
- The framework is lightweight, so the deployment processes should be easy and optimally made
- Statamic already has rich features installed by default that satisfy the technical and business needs

Other parts of my assignment are:

- Integrate the current CI/CD (continuous Integration / Continuous deployment) scripts to the new website base.

The created website base will be served as a minimum viable product (MVP), and also that serves as the proof of concept (POC) that the migration is possible.

Additionally, depending on the time that has been left after getting MVP done, one Fixico website actually migrated would be an additional success factor.

2.4 Research questions

Main question: How can I solve Fixico's "multiple website maintenance problem" which is caused by decentralization and deprecation, using Statamic CMS platform in order to archive unification for all known Fixico websites catered towards developers, end users and content writers?

This question accurately encompasses everything mentioned in the problem description and hints as to what the assignment entails. Answering this requires combining knowledge gained from many modules.

Sub questions (sub chapters):

- What is the marketing team base workflow and how can the new setup improve this?
- How to manage different Fixico websites using 1 shared admin panel?
- How to securely login to the new website admin panel using the company's Google account?
- How to support internationalization and localization (where specific content is only available to specific locales)?
- How to define metrics used for website user experience by improving performance, accessibility, best practices and SEO that are defined by google lighthouse?
- What is going to serve as the deployment for the website-base?
- How to optimize website building and deploying processes to achieve reliability and performance?
- How to securely and reliably store content of the website?

The approach taken per each research question is described in chapter 3.

2.5 Stakeholders

Stakeholders of the new website base:

- Website visitors: They are the most valuable stakeholders, the new system should provide them smooth and fast website interaction. Depending on the type of website, different websites are served.
- Engineering team within Fixico: They should understand the implementation of the website base and be provided with the documentation.
- Marketing team within Fixico: They should access the admin panel and have smooth and easy control over the content of the given website.

2.6 Development

This subchapter describes the environment where the student will be working with to approach graduation assignment.

2.6.1 Entry point
The entry point of the assignment are the prerequisites described in the assignment description (Chapter 2.3 The assignment) and a brand-new Github repository for the future source code of the assignment.

2.6.2 Tooling
Jira management tool will be used as a main board for the graduate student. It is expected for the student to create stories (defined at last sprint planning) and issues. The student is also responsible for delivering up-to-date information about the status of the project to Jira.

2.6.3. Version Control
Version Control is handled with Git, the company is using GitHub as the remote storage of the repositories. New repository for the graduate project will be created and all the source code will be handled there.

Each time the student starts working on a new ticket a new branch is created for that issue. Once the ticket is complete and reviewed, the pull request can be created to merge new functionality to main.

2.6.4 Technologies
The main programming languages are PHP and JavaScript together with HTML and CSS

The student is going to use Confluence for documentation purposes. Confluence is a wiki-like tool that is tightly integrated with Jira and other Atlassian products that Fixico makes use of. The student will have access to all available documentation articles and will document himself all the research decisions and implementation details about the project.

2.6.5 Code quality
Code quality is ensured with the help of tools within the WebStorm IDE, and with the company's internal processes. The student also needs to have a code review upon completion of a ticket or story on the JIRA board.

3 Process description

In the previous chapter, the context of the research was mapped out. In this chapter, the problem statement of the previous chapter is further concretized in order to formulate a solution challenge. The solution challenge is a description of a product that has to solve a specific problem.

The solution challenge is further divided into several sub-questions with the chosen research methods. These research methods are also explained further in this chapter.

The purpose of this chapter is to create an overview of the research questions and how the researching of them will take place. The results are explained in chapter 4.

3.1 Research methodologies

This subchapter describes the approach per each research question the student is going to make during the researching phase of the assignment.

The table below gives an overview about the research questions.

1.0	How to optimize the new website base workflow for marketing team needs?	1.1	What are the requirements from the marketing team that websites should have?	Interview
2.0	How to manage different Fixico websites using 1 shared admin panel?			Technical desk research/ Interview regarding implementation
3.0	How to securely login to the new website admin panel using the company's Google account?	3.1	Does Statamic support login using third party authorities?	Desk research
		3.2	How to make it possible to login using Google corporate account?	Desk research

4.0	How to support internationalization and localization in a new website base?	4.1	What is the solution for internationalization?	Technical desk research
		4.2	What is the solution for localization?	Technical desk research
5.0	How to define metrics used for website user experience by improving performance, accessibility, best practices and SEO that are defined by google lighthouse?	5.1	What is user experience, and what are the metrics for that, what developer tools are used for this?	Desk research
		5.2	How to test and evaluate user experience within a website base?	Technical desk research / Interview
6.0	What is going to serve as the deployment for the website-base?	6.1	What is an existing deployment solution for Fixico websites?	Analyze Fixico stack
		6.2	What is an alternative to the existing Fixico deployment process?	Desk research
7.0	How to build and optimize website building and deployment process?	7.1	What is the CI/CD current solution for Fixico?	Analyze Fixico stack

		7.2	What is the deployment process for current Fixico websites?	Analyze Fixico stack
		7.3	How to implement building and deployment automatic processes to a new website base?	Technical desk research / Interview
8.0	How to securely and reliably store future content of the website?			Technical desk research, Analyze Fixico stack, Competitive research

Table 2. Research methodologies overview.

The following sub questions create a step-by-step approach to answer the main question.

3.1.1 How to optimize the new website base workflow for marketing team needs?

Marketing team is a part of the stakeholders of the websites, because they are actively involved in website content management.

To understand what are the needs and requirements of the marketing team to have in a new website for keeping their website management responsibility.

To answer the question the Interview with a person from the marketing team will be conducted. The person should be somebody who is actively involved in website maintenance, so the interview would be accurate.

The outcome of this research question are requirements from editors perspective to the website base.

3.1.2 How to manage different Fixico websites using 1 shared admin panel?

The idea here is to research how to develop a system that will handle 1 project merging websites to 1 system – Whether possible and how to create a system where multiple, not related to each other, websites will be handled via 1 system and admin panel?

Answering this question should start with the research technological desk research of current technologies/plugins supported by Statamic framework and the existing implementations available on the internet using competitors analysis.

The second step here is to have an interview with the technical supervisor about what he thinks about whether the solution seems to be a good thing to have from functional, technical and architectural wise.

The outcome is expected to be the approach that has to be taken to develop described functionality.

3.1.3 How to securely login to the new website admin panel using the company's Google account?

Using Google authentication is a fast and reliable compromise as the solution for the marketing team as long as it has turned out that it's technically impossible to archive the "shared admin panel" feature.

Therefore, research should be made on this topic, whether it's possible to configure Statamic to login to the admin panel using google identity provider in a secure way that only admins and editors can access that control panel.

To answer the question, a desk research of the official website documentation of Statamic and Google Cloud Platform on the information whether it's possible and how to implement the requested feature.

And as outcome, the implementation of the feature will be made to the pilot version of Statamic website.

3.1.4 How to support internationalization and localization?

Statamic needs to be researched to a topics on how to develop system that will handle:

- Internationalization (i18n) of the website – How will the translations of the same content in different languages be implemented?
- Localization (l10n) – How to create conditions to display parts of specific content in a specific language?

These 2 features above are crucial for starting a Statamic project, because Fixico is interested only in the solutions that handle i18n and l10n problems.

To answer this question, a desk research on existing implementations/official plugins of given framework that support upper described functionality.

The result here would be the design and approach taken to implement requested features.

3.1.5 How to define metrics used for website user experience by improving performance, accessibility, best practices and SEO that are defined by google lighthouse?

This is an important question, because it is about improving the website visitors' experience and improving the grades given by search engine ranking.

To answer this question, a desk research would be made on how to test the user experience, what are the tools available for this, what are their scores, and what should be kept in mind to achieve the most out of it.

An outcome here would be the strategy of testing user experience.

3.1.6 What is going to serve as the deployment for the website-base?

This is an important topic, where insights on the process of website availability, reliability and scalability would be answered.

This research question will be answered using analyzing Fixico stack on how Fixico arranges current websites with similar architecture, as an architecture consistency is something that Fixico looks forward to.

The second step would be to approve the points of view with Fixico's CTO to define the best strategy to secure, reliable and available public versions of the website base.

The outcome would be the decision and its insights on the website architecture alongside the setting-up remote environment process.

3.1.7 How to optimize website building and deploying processes to archive speed, reliability and performance?

Building and deploying processes are an important part of the development Statamic website, because it would reduce risks of finding bugs at a late stage, and ensure the deployment process goes smoothly.

To answer this question, a desk research is made on what approach should be taken to add a new website base to integrate the building and deployment process within the company.

As a proof of right approach is made, an interview with the technical supervisor or Fixico CTO would be made to discuss the approach details.

The answer to the question would be the plan and actual integration of the CI/CD process to graduation assignment.

3.1.8 How to securely and reliably store future content of the website?

This research question was initially not planned, however during answering the research question #6 about the deployment process, it has turned out that the current question has to be researched and answered properly.

Content of the website is the dynamic content that changes during the updating website pages, users, assets and other manipulations via control panel of the website made by website maintainers.

Answering this question consists of 3 different problems where each subquestion should have its own, opinionated solution on how it works currently with existing Fixico websites, and whether there can be any improvements made into it.

3.2 Planning

The company makes use of Scrum as its work methodology. This methodology fits the way the graduate has been conducting their own work up until this point, and it fits the current assignment as well. The graduate will keep themselves within scope and can determine the pace of the progress while employing this work methodology.

The student will be having a 2 week sprint that starts on Monday and finishes on Friday next week. At the time when sprint finishes, the sprint review and sprint planning would be made together with the company and technical supervisors.

Upon competition of 2-3 sprints, the sprint pace would be evaluated, and in case sprint takes longer or shorter time span, re-evaluated to get the most optimal planning.

3.3 Methodology

As far as processes are concerned, the graduate will participate in daily stand-ups, in which the graduate will provide insight into their work to the company supervisor. At the end of each sprint graduate will have to present a summary of their work in front of the whole company during sprint reviews.

As described before, Jira will be used as the tool for tracking progress throughout the development period. The board is split into 5 lanes:

- To Do
- In Progress
- In review
- In functional review
- Done

There might be 2 board sections unclear: “In review” and “In functional review”, the difference is the “In review” section is made for conducting code reviews and technical overview of made solution, and the “functional review” means to check the developed functionality with the company supervisor where the solution functionally fits the assignment correctly.

3.4 Sprint planning

The sprint planning is included in the appendix with the amount of sprints, their timelines and expected work delivered.

4 Research outcomes

This chapter describes sub-questions outcomes that were listed earlier in chapter 3. In this chapter it is made clear what the design choices for the website base are based on.

4.1 How to optimize the new website base workflow for marketing team needs?

The primary research method here is an Interview that was conducted with a marketing manager. The person who was interviewed was Isabell Heller, who is a marketing specialist and main maintainer of current Fixico websites.

During the interview the list of requirements was set together with their prioritization. The list of requirements is included in the chapter 5.1 Functional Design (Interview with person from marketing team, March 2022).

Together with discussion of requirements, the drawbacks of existing websites were listed too, where almost all of them overlap with the listed drawbacks in the Chapter 2 section 2 - Problem Definition.

Moreover, an ideation phase was executed during an interview, where the additional functional features were discussed, such as :

- Solve saving problems of the webpage content that was under work at the same time.
- Create Activity tracking tool, with an overview of the most recent changes made to the website.
- Create the possibility to change content of different Statamic websites using only 1 admin panel.

In conclusion, the answer of the research question is the list of functional and technical requirements that will be added to Functional Design.

4.2 How to manage different Fixico websites using 1 shared admin panel?

Having 1 admin panel that manages the content of all the websites Fixico has is a feature requested by the Marketing team as a should requirement.

The graduate has answered this research question by doing a research on an official framework documentation regarding following questions:

1. How to use 1 admin panel that can change the content on multiple websites that are located in different website projects?
2. How to create 1 common website project, where the theme, content and locales of different not related to each other websites would be stored in 1 place that ensures success using 1 admin panel for management?

Answering 1st sub question: apparently there is no functional support of using 1 admin panel among different website projects, no following information was available on official framework documentation website, and no related discussions were made on framework forums.

The implementation approach discussed in the 2nd sub question, where all the website themes and content will be made in 1 project, so the admin panel would have access over all the content easily, has many disadvantages that overcome the possible advantage for the marketing team. To ensure the graduate is taking into account the right cons, an interview with technical supervisor was made, where the points make a decision to go against developing this feature, because:

- Website structure will get too messy. Each website, and it's locales would make a massive list of all versions of the website, that will be harder and harder to maintain (Interview with developer supervisor, March 2022).
- Website theme will be very hard to configure to have multiple configurations, because:
    - Layouts that serve as an entry point of an html document are impossible to have more than in 1 configuration.
    - Different globals (like header, foot, etc.) would be almost impossible to configure to be displayed only on specific pages.

In conclusion, the answer is that Statamic is not made to support multiple, not related to each other, websites in 1 admin panel. Which means it's sort of impossible to implement a researched feature.

Main reason for such a request is to make login as easy and as fast as possible, which can also be achieved using different methods. During an interview with Isabell from the marketing team, different ideas were suggested, such as using One Password - a Fixico tool that stores safely all passwords on device, using Fixico google account as identity proof. The idea with verifying identity using Fixico Google account was interesting for Isabell, and therefore further research was continued, that is below.

If talking about decentralization and deprecation covered in the problem definition and as a possible solution (answer) in the main question, these 2 things would not be a problem, because in future all websites would be made of Statamic, where the deprecation can be easily controlled and fixed, by doing repetitive solutions to the needed websites, and decentralization, a problem with a big technological stack required to maintain different websites would be replaced with 1 similar architecture around Statamic.

4.3 How to securely login to the new website admin panel using the company's Google account?

Logging in the admin panel using Google corporate Fixico account is possible. Statamic supports extending it's core functionality and their adapters, so therefore a Laravel Socialite OAuth2 identity provider was found and integrated into Statamic website.

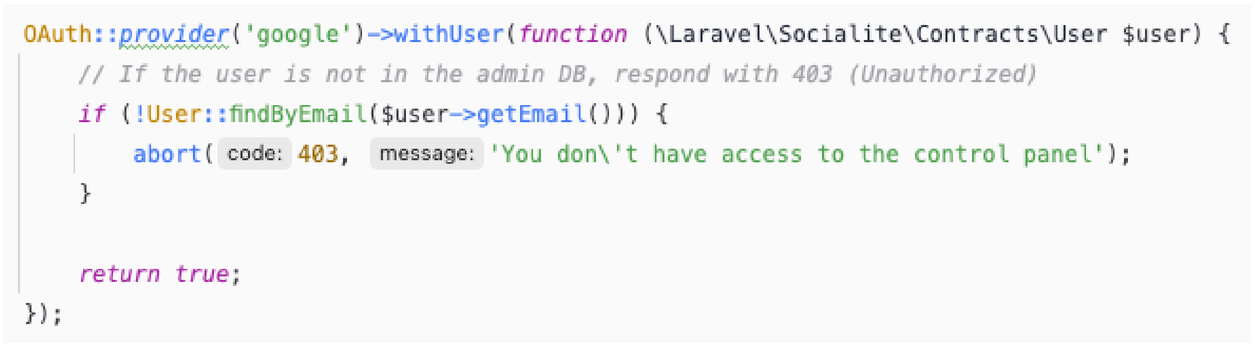
OAuth2 authorization is a specific way to verify or retrieve account information without using the client's password, which makes the website and the client safer (Cobb & Mann, 2020).

Instead of proving the client's credentials, the client is redirected to the official identity provider webpage (in our case - Google), where the user chooses the account to be authorized, and then is redirected back to the original website with access to a temporary token serving as credentials.

To accomplish the described goal, the research can be divided into 2 parts

1. How to configure Statamic for the authentication using third party providers
2. How to create a Google application that will be connected to the Statamic, so the credentials of Google Fixico account will be verified.

Laravel Socialite has a list of providers supported out of the box, where Google is included. In case Google was not included, additional configuration should have been made. Also, a set of configuration manipulations to Statamic configs should be made to treat Statamic using also different identity providers.



```php
OAuth::provider('google')->withUser(function (\Laravel\Socialite\Contracts\User $user) {
    // If the user is not in the admin DB, respond with 403 (Unauthorized)
    if (!User::findByEmail($user->getEmail())) {
        abort( code: 403,  message: 'You don\'t have access to the control panel');
    }

    return true;
});
```

Image 1 Actual authorization process.

The source code above shows an actual authorization process. Google Authentication is successful only if the user with authorized email is already registered in the admin user system. Otherwise the user would be redirected to the main page of the website with an error message.

Answering the second part of a question, a special app-authenticator should be created in Google Cloud Platform. While creating the authenticator project the details about the app should be provided such as: name, data to be used and the logo. In case the project requires too many data parameters from the users who sign in, google support would have to manually verify the project authenticator.

Once the project is successfully created, the secret codes of the app will be revealed, such as Client ID and Client Secret. These keys are needed to be configured in Statamic so the admin panel would know the exact details to redirect users to the Google authentication page.

In the end, users while logging in would have 2 options how to log in to their admin accounts:

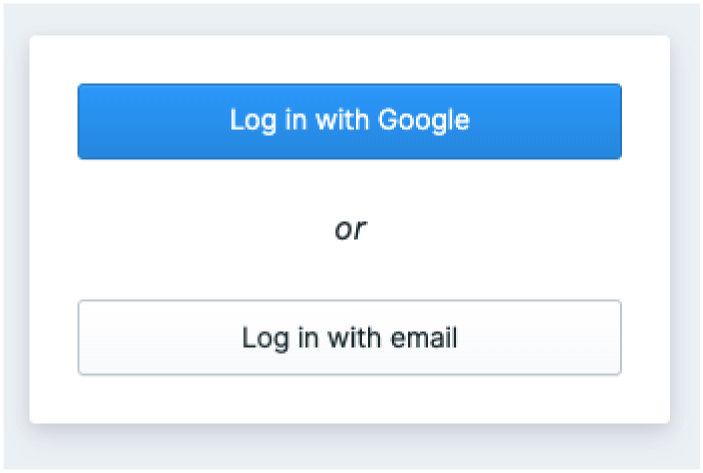


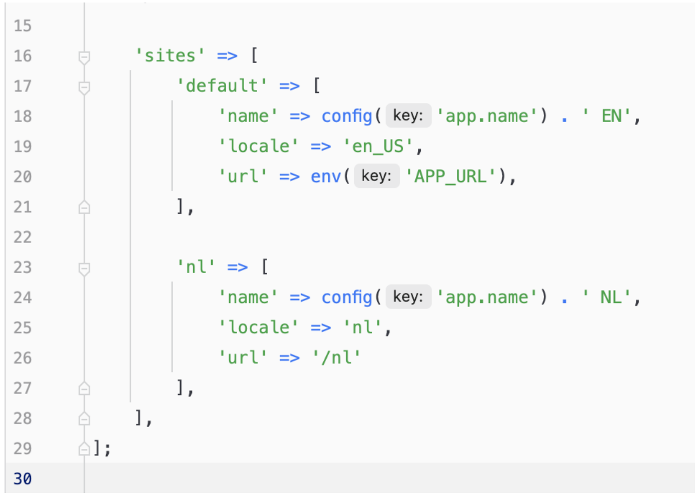
Image 2. Selector of a login option

4.4 How to support internationalization and localization in a new website base?

Internationalization and localization are important features that the marketing team finds as a must requirement, because the company operates in different countries and the websites need to be adopted to each operated region.

First of all, the Statamic supports the researching topic, and mainly it's possible to configure system files, where the routing and localization would be handled. Talking short, the following set of actions need to be performed to get the desired feature:

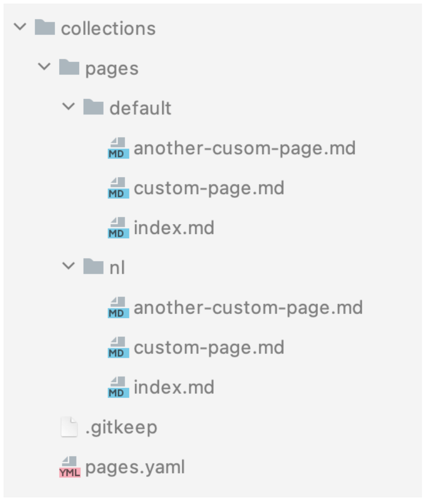
1. Configure website router and add following values: the translated name of the website, language and it's relative URL would be provided to each language of the website.



```
    'sites' => [
        'default' => [
            'name' => config( key: 'app.name') . ' EN',
            'locale' => 'en_US',
            'url' => env( key: 'APP_URL'),
        ],

        'nl' => [
            'name' => config( key: 'app.name') . ' NL',
            'locale' => 'nl',
            'url' => '/nl'
        ],
    ],
];
```

Image 3. The configuration of URLs to provide to certain locale webpages

2. Adding a plugin 'multisite' together with manual replacement of website content trees to the subdirectories with locale names taken as the name of those directories (Statamic Multi-site).



```
collections
  pages
    default
      another-cusom-page.md
      custom-page.md
      index.md
    nl
      another-custom-page.md
      custom-page.md
      index.md
    .gitkeep
  pages.yaml
```

Image 4. Markdown files containing webpage content are split under locale folder

3. Complete the configuration in the control panel of the website, where each page can be on demand available in certain languages and translated as well. Besides, content of each localized version of a website can be easily changed, deleted or added according to the local needs.

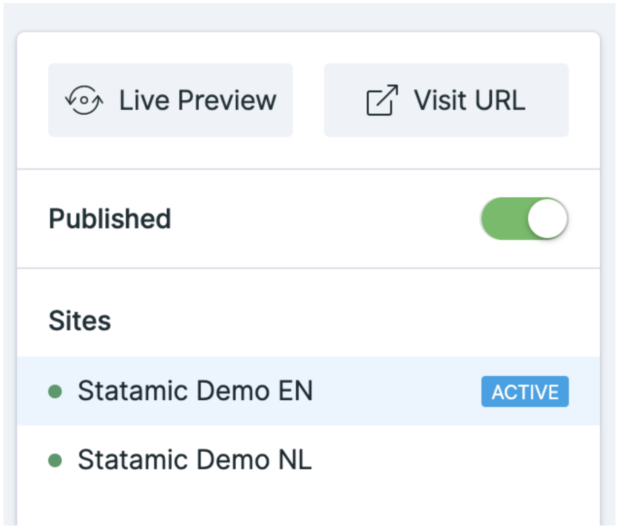
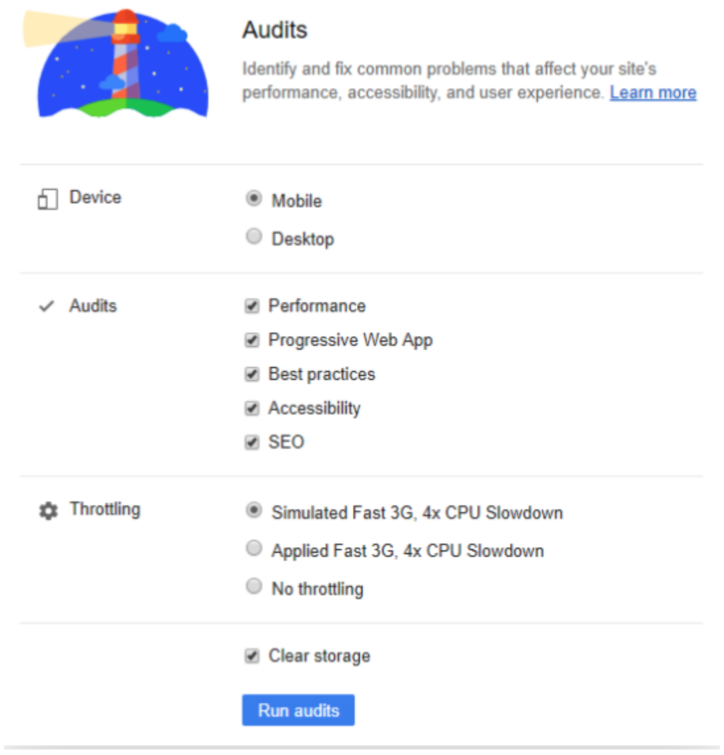


Image 5. The selector of the website local versions, where it's possible to change the content to localized version

4.5 How to define metrics used for website user experience by improving performance, accessibility, best practices and SEO that are defined by Google Lighthouse?

Google Lighthouse is a free tool designed to measure and to help improve website performance (Google Documentation). It's open-source software so it can be used on any website. This tool audits the accessibility and SEO of web pages, with a particular focus on core web vitals (Backlinko).


Audits

Identify and fix common problems that affect your site's performance, accessibility, and user experience. Learn more

Device: [✓ Mobile] Desktop

Audits:
- ✓ Performance
- ✓ Progressive Web App
- ✓ Best practices
- ✓ Accessibility
- ✓ SEO

Throttling: [✓ Simulated Fast 3G, 4x CPU Slowdown] Applied Fast 3G, 4x CPU Slowdown No throttling

✓ Clear storage

Image 6. The Google Lighthouse main screen before running website audits.

This instrument will be used to measure the metrics described above for the graduate's website base, so the graduate during building website base will always take into account the ranking Google Lighthouse metrics.

A desk research is made on the metrics and their evaluation on what steps or decisions should be made to plan, and implementation to the website base with the best lighthouse matrics in mind. All the information about the metrics Google Lighthouse uses for measurement can be found in Appendix (Research additional information, Google Lighthouse metrics)

The Google Lighthouse audit tool will be used for the graduate to test and evaluate the performance of the website base the graduate is doing.

The goal for the graduate is to set up Google Lighthouse testing, where metrics will be at the level of not lower than 90% grade, otherwise the search engine ranking will consider the website as not optimized.

4.6 What is going to serve as the deployment for the website-base?

This research question is very important, because it defines the structure of implementing CI/CD and usage/maintenance of other modules, related to the structure of the website-base.

Using a Desk research method, the information about existing deployment was gathered. Fixico uses Kubernetes (platform for managing Docker containerized workloads and services) architecture for their projects, because it has many advantages while handling many different unrelated to each other environments and applications:

- Very simple deployment with Docker containers
- No need to handle server states, as only containers define the environment they work with
- Integrated Version-Control system - a storage of previous versions of the pods (single deployable objects) with their content inside. In case the current version of pod breaks, there is an immediate replacement to fulfill 0-downtime-deployment
- Has a private network, where pods originally can communicate only with other pods. It adds a security aspect and is a great solution for using microservices.
- Easy usage and integration of different configurations.
- Easy environment manager, where different environments (staging and production, etc.) can be handled.

That being said, the graduate has found an alternative to Kubernetes that can also be considered as a deployment server for the project, and that is - usage of Virtual Machine.

It could be an alternative to the existing Kubernetes architecture, because it can be:

- Simpler configuration to store website project,
- Local hard disk environment can be used to store the content of the website

With this given information the interview was conducted, where during discussions and defending different ideas, the result was agreed on - using Kubernetes. As an outcome of an interview, the comparison table was created describing each feature needed for website-base and it’s integration into deployment.

Feature	Virtual Machine (VM)	Kubernetes (KB)
Configuring the development environment before releasing a new project into it	- The whole virtual machine should be configured before accepting projects to work with.	+ Kubernetes is already configured and working, there is a need to define the new project’s name in GUI deployment manager app, where all Kubernetes features and integrations can be configured to work with each specific project
Arranging environments	- VM should be additionally configured to arrange 2 environments (staging and production)	+ KB is configured, and already arranges environment among pods
Zero downtime deployment	- VM should be additionally configured, and website-base project configured as well to support the desired feature	+ KB natively supports feature
Arranging secret keys and secret files	- VM should be configured alongside project as well to securely share keys	+ KB has integrated Google Secret application that has GUI and securely stores and arranges environment variables
Storing website content	+ VM can use own local storage for secure website content storage	- Kubernetes does not have any local storage that can be allocated with desired feature, and each time a new deployment is released, the website content will be replaced with fresh clean install. Therefore, additional content management should

		be provided.
Communication with other Fixico services (e.g. MySQL database)	- Fixico's MySQL database is in KB, and communication layer should be arranged, which would result in exposed public ports of MySQL and therefore potentially lead to security issues.	+ KB has a private network where pods can easily communicate with each other reliably. MySQL is one of the pods located there.

Table 3. Comparison table of different deployment approaches.

Concluding the features and their connectivity in different deployment environments, it is clearly seen that a website base would easier and faster be deployable using Kubernetes.

4.7 How to build and optimize website building and deployment process?

The Continuous Integration and Continuous Deployment (CI/CD) are a coding philosophy and set of practices that drive development teams to frequently implement small code changes and check them into a version control repository and automates application delivery to selected environments (Sacolic, 2022)

This technology is relevant to every project that is in active development or that is used by clients or users. The same applies to the graduate's project.

Fixico makes use of CircleCI as a platform to perform the CI/CD processes in their projects. CircleCI is a powerful small-to-large scale devops tool (P.Jain, 2020) that has nice features that Fixico makes active use of. Main advantages of using CircleCI are:

- Easy to set-up the first pipeline. CircleCI makes use of 1 config.yaml file where the workflows, jobs and steps are described on how to run.
- Runs cloud-based. Every new workflow runs in a new environment which is maintained by CircleCI themself.
- SSH feature. In case the pipeline fails, the developer has an opportunity to SSH to the failing environment and make an inspection for the problems. I myself have been using this feature a lot, and it has turned out helpful to reduce the time significantly to fix pipeline.
- CircleCI orbs. It's a register of packages of YAML configuration files that are being imported for repeated pieces of config YAML file. Fixico has many different projects running on same or similar bases, that use orbs to avoid repeating the same lines of config and adds flexibility.

Usually, the pipeline (set of automated processes) triggers on push events to the configured remote Version-Control System (which is Github for Fixico). In case the pipeline fails, the email is sent to the person who triggered the pipeline with the information what and how it has failed.

There are 2 types of deployment process for Fixico websites, where the CI/CD pipeline is the following:

1. Adding new code to the repository. Developer runs a command Git Push with his changes to the remote master (main) branch of Git repository. The CircleCI detects changes to the repo and starts building the website and the Docker image of the PHP server with the website on it, together with another Nginx web server to process and proxy the requests coming to the website. The next command would replace the Docker images on the hosting environment with the recently built. The new website will be available in a staging environment.

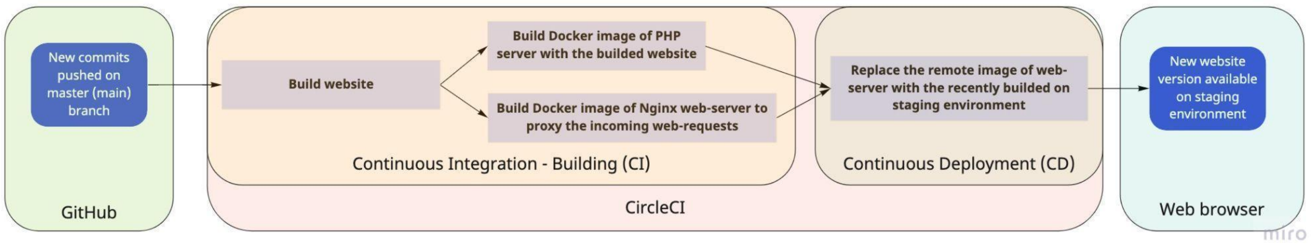
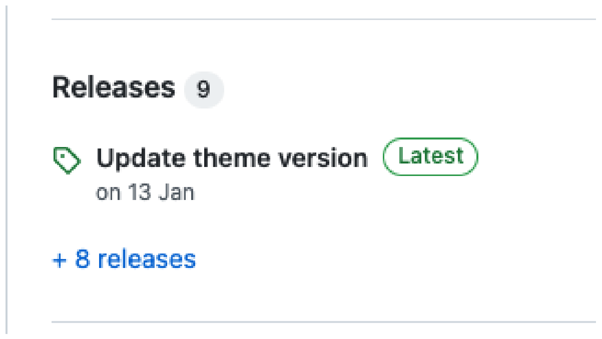

New commits pushed on master (main) branch
GitHub
Build website
Build Docker image of PHP server with the builded website
Build Docker image of Nginx web-server to proxy the incoming web-requests
Continuous Integration - Building (CI)
Replace the remote image of web-server with the recently builded on staging environment
Continuous Deployment (CD)
CircleCI
New website version available on staging environment
Web browser
miro

Image 8. Visualization of the CI/CD process for pushing new code to the repository in existing websites.

2. Creating a new version of the website. In this scenario, a new version of the website is officially released in a representative tab of the website repository.


Releases 9

Update theme version Latest
on 13 Jan

+ 8 releases

Image 9. The overview tab of releases.

Once the new release of a website is made, the same CI scripts run as in section 1, and after that the command runs to deploy Docker images to the production environment, where the changes made to the website are visible and available to users accessing the website.

4.8 How to securely and reliably store future content of the website?

The website, and it's content can be split up into 2 major groups:

1. Static content of the website, such as templates, configurations, and other parts of the website that define the behavior of the system.

   Such content is going to be stored in the representative GitHub repository, and in case the changes need to apply to it, developers can update the website with representative commits to the project's repository.

2. Dynamic content of the website, such content is going to be changed during the website usage. Primarily, it's going to be pages, their components, assets, and users who have access to the control panel.

   Statamic stores such type of content in the project directly under specific directories. This approach is completely fine for small scale projects, where there are no staging and production environments and are deployed to simple web servers.

   For the project's case, where project is going to be deployed to the Kubernetes environment, and have 2 different environments, where content of each would be stored separately, the content storage should be arranged differently.

Therefore in the next 3 subchapters the storage flow would be described for asset, content and user storage. The precise implementation of these 3 topics are described in the Technical Design.

4.8.1 Assets storage

Doing an internal desk research on how the visual content is handled, the following was discovered.

Fixico uses Google Cloud Storage (GCS) as a primary place, where images are stored. Website maintainers store only the specific URL to the images in the website. Which means in case any manipulations should be made, maintainers go to the GCS and apply changes there directly.

That being said, the existing approach of arranging assets on the websites is poor, because it requires maintainers to know the technical aspects of how and where images are stored.

Therefore, doing my desk research, I have researched that Statamic supports GUI for asset management, where the assets' location is on GCS. To achieve the desired functionality, an additional driver connecting GCS with Statamic website-base should be configured. The implementation is described in Technical Design in chapter 5.

4.8.2 User storage

Users are people who have access to the website's control panel.

By default, Statamic stores users in files, under the /users/ project folder. However this solution would mean that users would be lost each time a new deployment is made.

Therefore, a desk research was made, regarding the locations, where users could be stored. Apparently, Statamic has support for storing users in databases. Reasons for Statamic to create a user management system to work with remote environments as a large-scale solution for websites with huge numbers of users, because user arrangement using a file system by itself is slow.

Fixico already has a database in the Kubernetes environment, and it is MySQL.

Therefore, users would be stored in the database as a solution for saving users in the Docker Kubernetes environment.

4.8.3 Storage of web pages and their content

Right now, Fixico stores web pages in the separate Github repositories. So each time there is a change of the content, a special tool, called GitSync would update remote repository with updated content. Main reason for separating a website and its content is because content is much more frequently updated, and its commit history looks like a mess.

A research was made on how to store content files remotely, it has turned out that Statamic can work only with content files stored on the project itself. Main reason for it is the timing (speed) it takes to retrieve the content file from remote storage and return it to the website visitor. While reading local system files is super fast.

In total, there are 3 options to store website content remotely:

1. Using database. Statamic can be configured using many adapters and libraries to store the content on a remote database, but this approach has 2 major disadvantages. First disadvantage is the complicity of transitioning the content data from staging to production environment and other ways around. And the second disadvantage is the complicity of doing such transition. It would require reconfiguring core principles of Statamic workflow, and will increase the difficulty of maintaining the website with a given approach.
2. Using GitHub repo. This method is used among many Fixico websites. Officially Statamic supports automatic github connectivity, but apparently configuring Statamic to use completely another repository for the content is difficult to make. During an interview with Mujib, company CTO, a topic of using repository as a content storage was discussed. Apparently, he strongly advises to avoid doing such implementation, because maintaining such type of content arrangements is a hussle (Interview with Fixico CTO). Main reason for choosing github repo is for its version control system. The

idea here was that other Fixico departments would be able to revert certain changes. Apparently the outcome was different, it is too complicated a solution for other departments and they did not take advantage of it, and it is hard to maintain by developers as well.

3. Using back-up process. This implementation has a similar idea as the content repository, but different implementation. It is a process, where backups of the content directories would be stored on a remote under different folders that specify the date and time the backup was made. Statamic, as a framework built on PHP Laravel has an extension that is capable of arranging backups, which is a plus in the system architecture and simplicity. As the remote place, Google Cloud Storage is a great solution because it would bring more consistency to the project because there already are assets that are stored there.

With an overview of these 3 frameworks and their justifications, the decision goes to the last one that is about backups, because it seemed simpler and more reliable.

5. The solution

In this chapter the solution is described from a functional and non-functional perspective. This chapter covers all the functionality desired and achieved together with the description of processes going under-the-hood.

5.1 Functional Design

5.1.1 Requirements

The requirements were gathered during answering research question 3.1.1. To get more insights about how requirements were obtained, please refer to the Chapter 4.1 of this document.

The requirements are sorted by the type of the website's stakeholder and its priority from top to bottom.

№	Type	Requirement (User story)	Priority	Functional/ Non-Functional	Notes
1	User Editor	As an editor, I want to login to the admin panel, so I can securely work on the website.	Must	Functional	User can only login if he was already defined in the system
2	User Editor	As an editor, I want to login to the admin panel as fast as possible.	Should	Non-Functional	Look for an opportunity to log in using Identity providers (like Google).
3	User Editor	As an editor, I want to have CRUD rights on pages	Must	Functional	Show a prompt to the editor to verify the delete action on a page.
4	User Editor	As an editor, I want to have CRUD rights on components on the pages	Must	Functional	• Update an order of the components on the page. • Show a prompt to the editor to verify the delete action on the component.
5	User Editor	As an editor, I want to optimize the SEO values of the page.	Must	Functional	Provide functionality to add/change meta-data
6	User Editor	As an editor, I want to have a language picker, to work with specific locale versions of the website.	Must	Functional	

7	User Editor	As an editor, I want to have a "Preview" feature to have private access to the page so I can double check for possible errors or mistakes.	Must	Functional	
8	User Editor	As an editor, I want to have access to all websites	Could	Functional	To research whether it's possible to manage all websites using 1 shared admin panel.
9	User Editor	As an editor, I want to be able to edit global components	Should	Functional	Global components are the reusable components seen on many pages and having 1 implementation. Most common are components like header or footer, etc.
10	User Editor	As an editor, I want to see the list of the most recent activities made to the website	Must	Functional	
11	User Editor	As an editor, I want to have the component I work with, locked for other editors who work on website admin panel	Must	Functional	
12	User Editor	As an editor, I want to have interactable window where I have CRUD rights on website assets	Should	Functional	
13	User Admin	As an admin, I want to have CRUD rights on users.	Must	Functional	
14	User Admin	As an admin, I want to update user rights/permissions for all users	Must	Functional	
15	User Admin	As an admin, I want to update user profiles	Could	Functional	
16	Technical	The website has to be optimized for Search Engine Ranking using tool like Lighthouse	Must	Non-Functional	
17	Technical	The website should be	Must	Non-	The website should work with

		configured to work on remote environments.		Functional	Staging environment, that is available for website stakeholders (Editors, Admins, Engineers) and Production environment that works publicly to everyone trying to access it
18	Technical	The website should have a build and deployment pipelines	Must	Functional	
19	Technical	The website should be built using Statamic Laravel, TailwindCSS and AlpineJs	Must	Non-Functional	
20	Technical	The website should use assets from Google Cloud Storage	Should	Non-Functional	
21	Technical	The website should store and retrieve website content from backups	Should	Non-Functional	

Table 4. List of requirements for the graduation assignment

5.1.2 Control Panel core functionality

Statamic by itself does not support rich functionality in the control panel. Therefore a solution was brought to use a starter kit that would satisfy the editor requests functional-wise and satisfy technical-wise implementation.

Therefore, the best solution on the market is Statamic Peak - a free custom starter kit that extends the default control panel with most of User requirements in the table above. On top of that, peak has integrated support for the frontend frameworks described in chapter 5.3.8 Development frameworks.

5.1.3 User Groups

Statamic framework supports having multiple user groups, where the rights per each group are divided. Analyzing Fixico stakeholders of the website admin panel, there are 2 user groups: Editor and Admin users. The difference is the limitations to the content changes editors have, while admin users are granted with all permissions. It is expected that the marketing team would inherit editor roles, while developers would be treated as super users in the control panel.

Editor rights:

- CRUD rights on collection of pages
- CRUD rights on navigation bars of pages
- Read, Update rights on global (reusable) components
- CRUD rights on Assets
- CRUD rights on Forms (if they are present on the website)

- Read rights on Utilities, a tab where user can get functional and technical information about the website

While Admis have more rights on top of Editors:

- CRUD rights on Blueprints, the page editor template editor
- CRUD rights on Fieldsets, the editor of fields to be displayed on page/component in control panel
- CRUD rights on Users and their roles (rights)

Blueprints may be considered dangerous to be used by editors. Because they represent the functionality behind the UI, and in case an attribute would be slightly changed, it might break the functionality of the website. The same applies to Fieldsets, in case a handler (interpreter variable supplying html template with variable's data) would change, the website would stop displaying some parts of the content.

5.1.4 Addons

During an interview with Isabell from marketing department where the ideas and requirements for the website were gathered, an idea was introduced to implement the following additional features:

1. Multi-user-collaboration - is an add-on feature that locks the component of the page that somebody is editing. It is a helpful tool that prevents overriding the same component while 2 or more people want to save their changes on it.

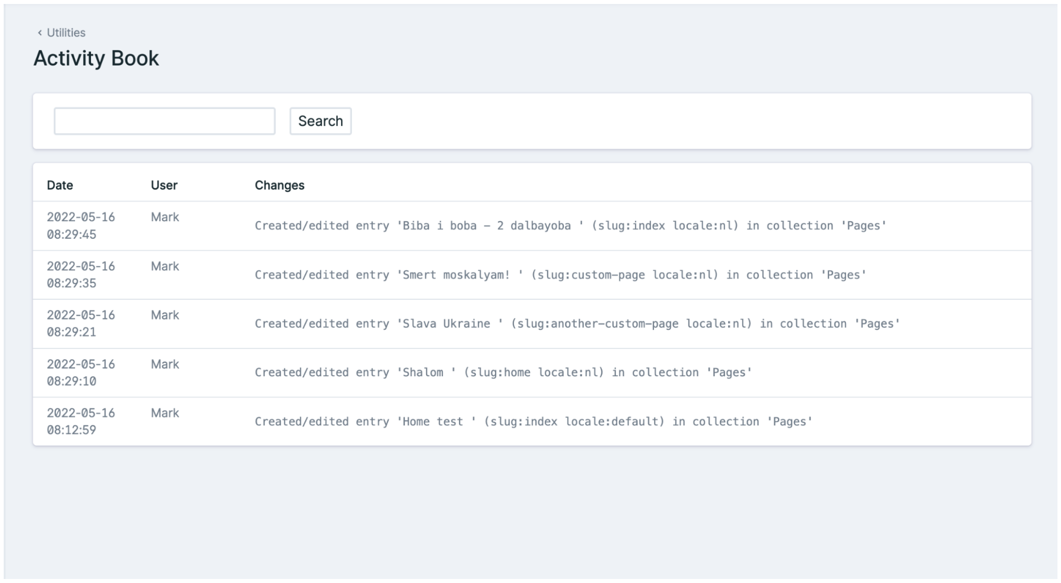
   This add-on shows that the component is locked and disables the interaction with it until the 1st person changing it is done.

   This is a Statamic add-on and has easy integration to the project.

2. Activity Book - is an addon that stores all the changes made to the content of the website. It stores information like when, who and what change was applied.

   The page with changes in the control panel can be found on Utilities tab -> Activity Book.

   The UI of addon looks the following:



Date	User	Changes
2022-05-16 08:29:45	Mark	Created/edited entry 'Biba i boba - 2 dalbayoba ' (slug:index locale:nl) in collection 'Pages'
2022-05-16 08:29:35	Mark	Created/edited entry 'Smert moskalyam! ' (slug:custom-page locale:nl) in collection 'Pages'
2022-05-16 08:29:21	Mark	Created/edited entry 'Slava Ukraine ' (slug:another-custom-page locale:nl) in collection 'Pages'
2022-05-16 08:29:10	Mark	Created/edited entry 'Shalom ' (slug:home locale:nl) in collection 'Pages'
2022-05-16 08:12:59	Mark	Created/edited entry 'Home test ' (slug:index locale:default) in collection 'Pages'

Image 10. UI of Activity Book addon.

5.2 Technical Design

5.2.1 Deployment

This topic is one of the most important, because it's an important architecture decision that has direct impact on decisions made on other modules of the website.

The website-base is configured to support staging and production environments:

- Production environment is an environment where the website is live to the intended users (Techopedia, 2020).
- Staging environment is an isolated copy of the production website that is accessible only to Fixico employees. It is a place where the testers or developers run the inspections on the website behavior or design (Umbraco).

However the website works right now only on staging, because the website-base by itself has nothing to show to Fixico clients.

The deployment of the website-base is implemented as part of existing Fixico infrastructure - called Kubernetes, which is a platform for managing Docker containerized workloads and services. It is a perfect large-scale tool to create and maintain containers (Kubernetes Documentation, 2022).

The website-base works in a pod - single deployable object in Kubernetes, where 2 containers (services) are held inside:

1. PHP FastCGI Process Manager (PHP-FPM) is a PHP extension, making high optimization for online small to large scale websites. The website project together with the source code and library packages are stored there. The container exposes port 9000 - which is typical for PHP servers as an income for HTTP requests.

2. Nginx is a web server that supports out of the box tools like reverse proxy, load balancer, mail proxy and HTTP cache (Kinsta, 2022). Nginx is the best combination solution for PHP FPM for its high-scalable asynchronous structure and low memory consumption performance.

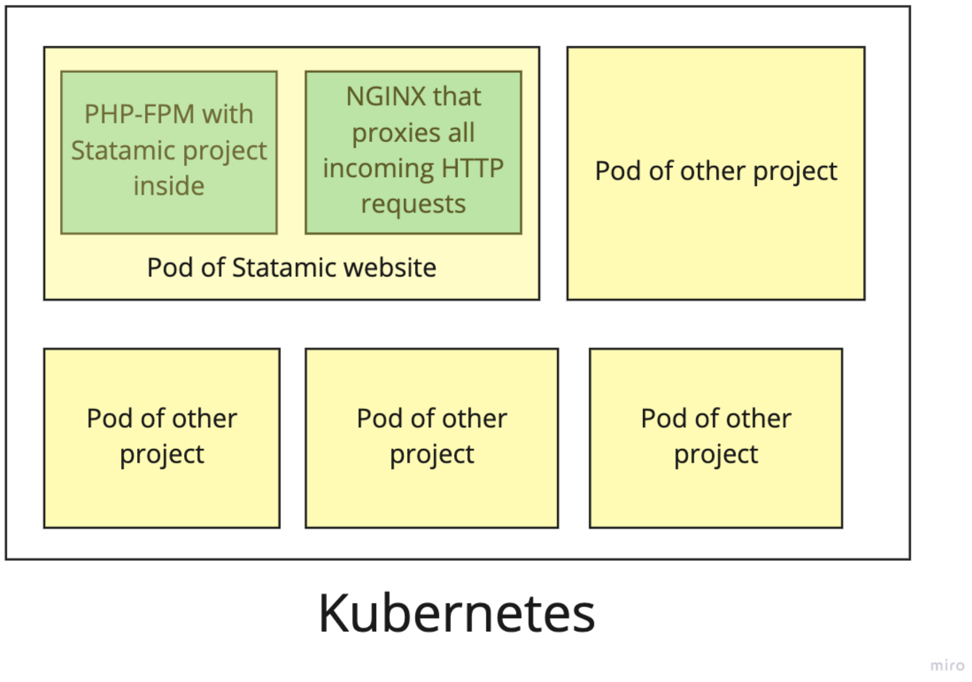


Image 14. Visualization of Statamic website being part of Kubernetes environment

Besides, additional configuration of the Kubernetes itself is stored in .yaml files under `/deployment/templates/` relative from the project's root.

Some of the Kubernetes configurations that are used by website-base:

1. Deployment - a set of rules to Kubernetes on how to manage the pod, how many previous versions of pod should be stored, and other behavior of pod itself
2. Service - internal pod router, that exposes the ports of the pod's containers to communicate inside of Kubernetes environment.
3. Ingress - outer pod router, arranges network traffic that goes outside of the Kubernetes.
4. Certificate - a less-encrypt service that arranges the valid SSL certificates for secure HTTPS connection.
5. Secret - a service that arranges secure allocation of credential environment variables to the pod services environment.

In case the website goes down, the special registry called Sentry would detect that and run the command to restart the Pod of the website, in case the website does not goes live, the special message would be sent to relevant Slack channel

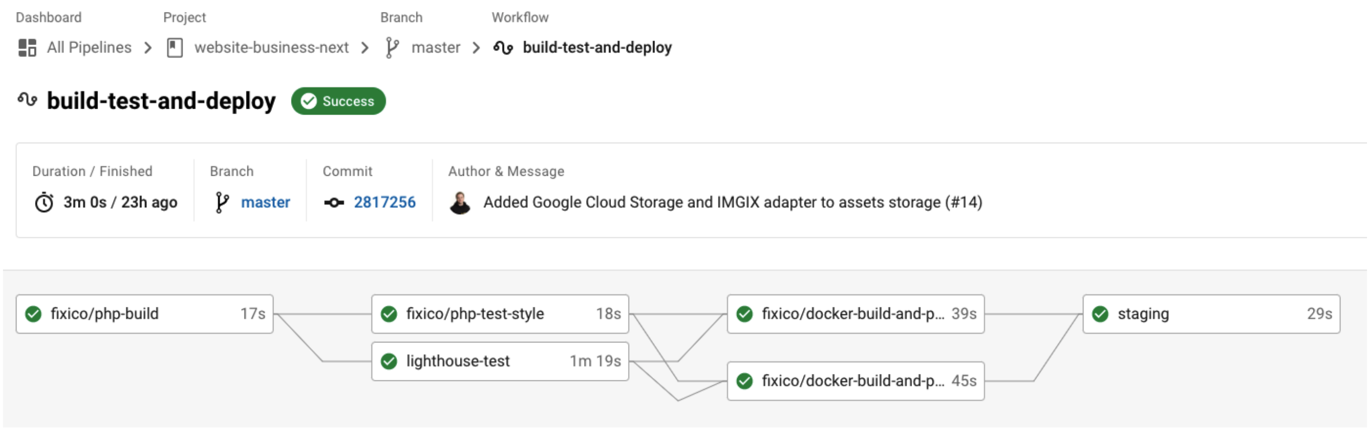
Taking the approach of using Pod architecture with Kubernetes environment would mean to keep the architecture in mind, because a solution for storing any content-based data in the pod would have to be restructured.

The main feature using Kubernetes is the impossibility to store relevant to the website data in the container. It is related to the architecture and the main cause for this is the way how the containers are created and how they work. Each time the new version of the website is created and 2 Docker images are created (more information on this in CI/CD part), the deployment pod gets replaced with a newer version. While the previous pod with the possible content data inside - is stored to the history of unused pods.

The outcome of this is to use the approach of not storing the website data in the pod, but rather use other 3rd party remote data storages, where the Statamic website-base would have the CRUD rights on. Such possible solutions are remote Databases and Cloud Storages (like Google Cloud Storage that is actively used by Fixico).

5.2.2 Introduction to Continuous Integration and Deployment

The pipeline created for the Statamic CMS for website base project is the following:


Dashboard / All Pipelines > Project / website-business-next > Branch / master > Workflow / build-test-and-deploy

build-test-and-deploy Success

Duration / Finished	Branch	Commit	Author & Message
3m 0s / 23h ago	master	2817256	Added Google Cloud Storage and IMGIX adapter to assets storage (#14)

fixico/php-build 17s
fixico/php-test-style 18s
lighthouse-test 1m 19s
fixico/docker-build-and-p... 39s
fixico/docker-build-and-p... 45s
staging 29s

Image 11. The pipeline for website base

The pipeline has 6 stages in total, where first 3 of them are CI and the rest - CD.

5.3.2.1 Continuous Integration

The pipeline has 3 jobs defined, and they are the following:

1. fixico/php-build - Is a Fixico orb that checkouts the code, defines the devops environment variables, installs composer dependencies and saves everything to localstorage (to use in later stages)
2. fixico-php-test-style - is a Fixico orb that checks the styling of PHP source code of the project. Fixico has created its own coding standards and this job inspects this specifically.
3. lighthouse-test - is a job that prepares the environment and actually runs Lighthouse. More about Lighthouse testing is described in the next subchapter.

Usually the jobs 1 and 2 run the same amount of time as on image 11, while the job number 3 with lighthouse tests is time consuming. Main reason is because the Lighthouse runs 3 times on each page of the website, and testing each page in total takes around 30-50 seconds.

5.3.2.2 Continuous Deployment

After the tests, other jobs start running:

4. fixico/docker-build-and-push-php-fpm - is a job where Docker image of PHP server with the project in it is built and pushed to Docker registry.
5. fixico/docker-build-and-push-nginx - is a job where the Docker image of the NGINX server is built and pushed to the Docker registry.
6. staging - is a job, where a Fixico tool, called Deployment Manager, receives a signal about creating/updating deployment environment and starts the Deployment process.

   Deployment manager is a tool, created by Fixico, that is responsible for creating and updating the project containers on Kubernetes environment together with connecting the following containers:

   - Secret - Is a Google Kubernetes container storing the environment variables together with credential variables that will be used by the project container to authenticate and work with the other 3rd party systems.

   - Other containers that are not used by website base

   Once the step with deployment manager is finished, the next step makes sure the deployed website is available by doing HTTP requests to the deployed website's URL.

   After the verification about successful deployment, the website is registered to Sentry watcher - a tool used by Fixico that monitors the websites for their availability and speed. In case the website goes down at any time, a special notification would be sent to Slack channel - with the basic information about the website failure.

If the pipeline passes successfully, the updated website-base is accessible.

5.2.3 Testing

Testing is an important part in quality assurance of the project. For the graduation assignment, 3 main testing approaches are described below.

5.2.3.1 Source code testing

The project uses the Statamic CMS package as the core source code for the project. Statamic have their own quality assurance standards that assure that the platform is stable and reliable to use out-of-box.

On top of the Statamic CMS package, a new add-on was developed for graduation assignments. This package does not have any unit or integration testing written, mainly because the package relies on specific development choices (like working with external databases). Therefore as a QA part for the addon, the following tests were performed after the delivery of the package:

- Monkey testing - a technique where the user tests the application or system by providing random inputs and checking the behavior, or seeing whether the application or system will crash (Wikipedia).
- Functional testing - is a type of software testing that validates the software system against the functional requirements/specifications (T.Hamilton, 2022).

The tests described above were made manually after the deployment of the addon.

5.2.3.2 Editors acceptance

As editors, who are primarily marketing team, as an important stakeholder of the websites in Fixico, the control panel should be as much adopted to marketing teams needs as possible.

Therefore, an acceptance interview with a person from the marketing team was held, where the control panel functionality was reviewed (Acceptance interview with Isabella).

An outcome of an interview is that the control panel has all the core features for editors to be able to work with the website in future. Additionally, some new requirements were gathered to improve the editor's work.

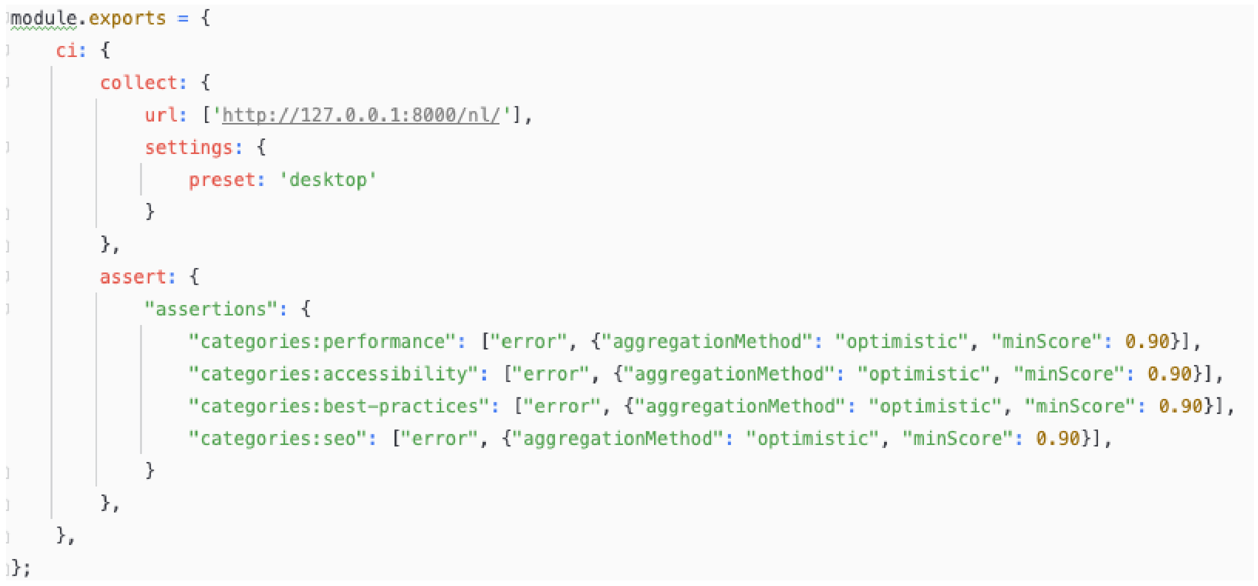
5.2.3.3 Visitors experience testing

Currently, Fixico uses only Google Lighthouse as a main metrics tool to measure the website parameters for the visitor experience. More details into what metrics and how they are used are in the research outcomes chapter.

The best idea to develop and maintain a production website is to always be sure about the highest ranks of Google Lighthouse, because it has a direct impact on Google Search Engine's ranking.

Therefore the Google Lighthouse testing was integrated into a continuous integration pipeline, where a fresh server installs all required dependencies and runs the tests automatically.

Lighthouse has released a tool that is easily integratable into CI/CD's pipelines - Lighthouse CI, it is a command line tool, where the configuration per lighthouse is defined in lighthouserc.js file in the root of the project. The tests run 3 times on each defined page



```
module.exports = {
    ci: {
        collect: {
            url: ['http://127.0.0.1:8000/nl/'],
            settings: {
                preset: 'desktop'
            }
        },
        assert: {
            "assertions": {
                "categories:performance": ["error", {"aggregationMethod": "optimistic", "minScore": 0.90}],
                "categories:accessibility": ["error", {"aggregationMethod": "optimistic", "minScore": 0.90}],
                "categories:best-practices": ["error", {"aggregationMethod": "optimistic", "minScore": 0.90}],
                "categories:seo": ["error", {"aggregationMethod": "optimistic", "minScore": 0.90}],
            }
        },
    },
};
```

Image 13. The Google Lighthouse configuration file.

The file has the following configuration:

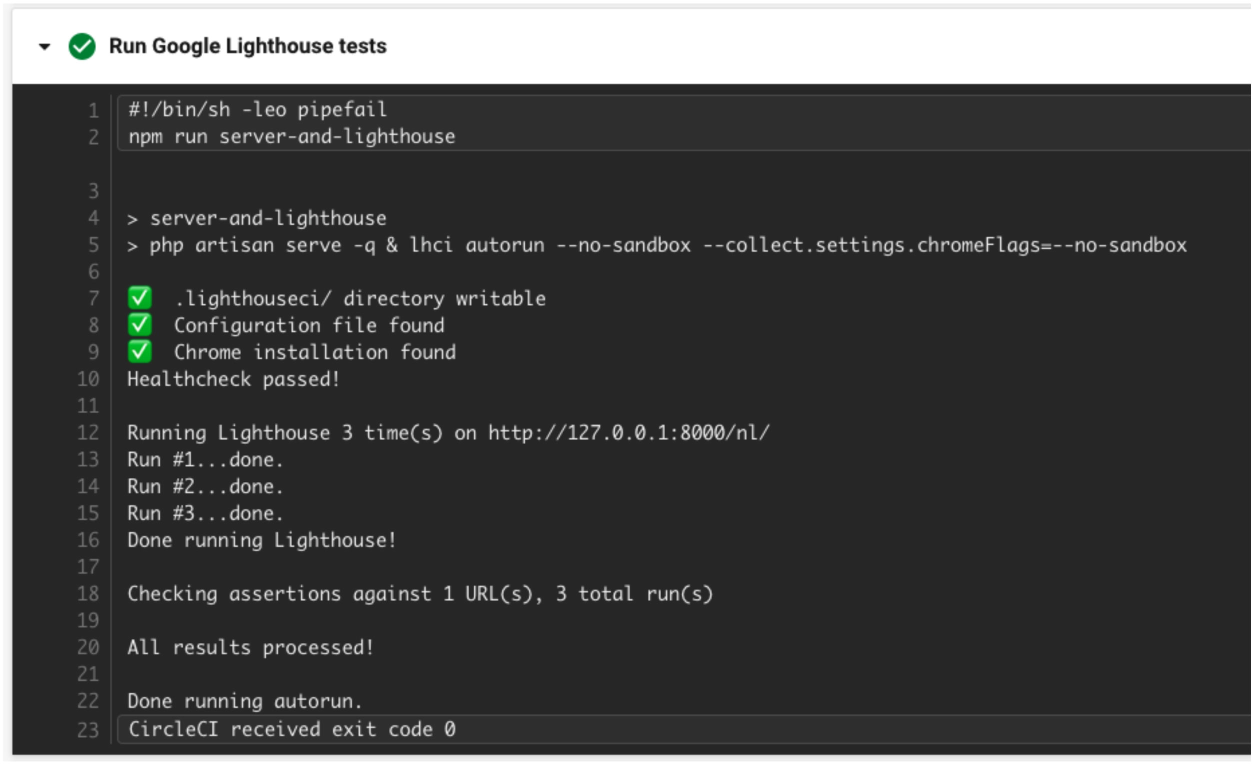
- URLs that have to be tested. Unfortunately, the solution to test all available website URLs automatically was not found, so therefore URLs have to be specified manually.
- Simulation concerning the device to be tested. As it can be seen, the website is tested for desktop sizing, however the website must support all possible screen sizes. The desktop is used to measure the most contentful variant of the website, because there is content that is hidden if the display size is not sufficient (phones, tables) to display it.
- The metrics and their values. Each metric should have at least 90% score to pass the pipeline.

  The 1st argument of metric definition is logging type, what for the project's case is "error", that means in case the metric result does not fulfill the minimum score, the error logging would be printed with expected and actual results.

  The 2nd argument of metric definition is the aggregation method - that defines what resulting value should be compared with expected value. For the project purpose, the best value out of 3 runs will be compared

  And the 3rd argument is the expected score to be for the metric, it ranges from 0 till 1, where 1 means 100%. The project is configured to expect a 90% score on each metric. That is the bottom line of the green testing level.

When the metrics succeed, the pipeline obviously continues, and the similar logging for testing will be the following.

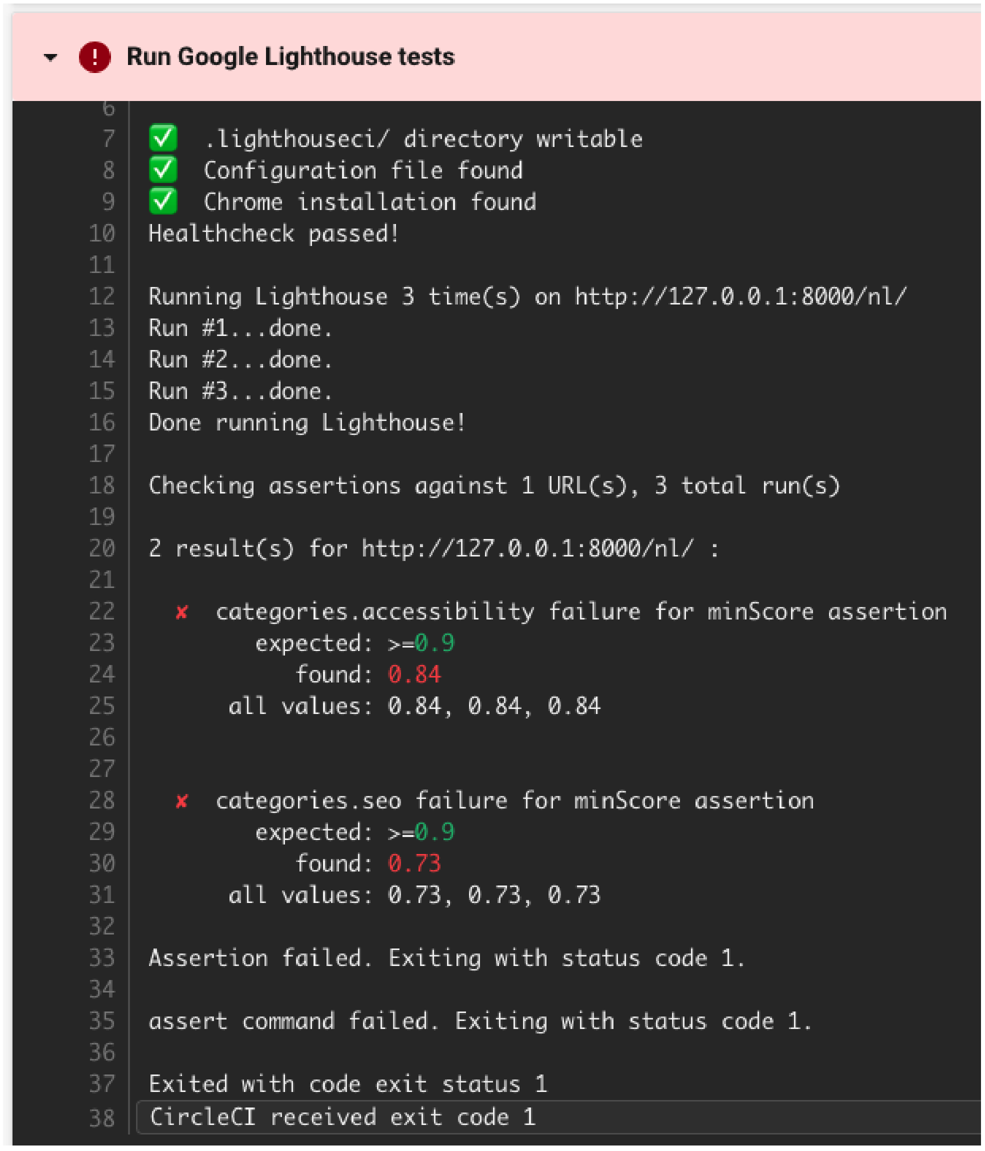


Run Google Lighthouse tests

```
#!/bin/sh -leo pipefail
npm run server-and-lighthouse


> server-and-lighthouse
> php artisan serve -q & lhci autorun --no-sandbox --collect.settings.chromeFlags=--no-sandbox

✅  .lighthouseci/ directory writable
✅  Configuration file found
✅  Chrome installation found
Healthcheck passed!

Running Lighthouse 3 time(s) on http://127.0.0.1:8000/nl/
Run #1...done.
Run #2...done.
Run #3...done.
Done running Lighthouse!

Checking assertions against 1 URL(s), 3 total run(s)

All results processed!

Done running autorun.
CircleCI received exit code 0
```

Image XX. The successful pass of the tests in CircleCI.

However, it might be the case that tests do not run successfully. In that situation the pipeline would fail, and the job logging would be as in an example below.


Run Google Lighthouse tests

```
✅  .lighthouseci/ directory writable
✅  Configuration file found
✅  Chrome installation found
Healthcheck passed!

Running Lighthouse 3 time(s) on http://127.0.0.1:8000/nl/
Run #1...done.
Run #2...done.
Run #3...done.
Done running Lighthouse!

Checking assertions against 1 URL(s), 3 total run(s)

2 result(s) for http://127.0.0.1:8000/nl/ :

  ✘  categories.accessibility failure for minScore assertion
        expected: >=0.9
           found: 0.84
      all values: 0.84, 0.84, 0.84


  ✘  categories.seo failure for minScore assertion
        expected: >=0.9
           found: 0.73
      all values: 0.73, 0.73, 0.73

Assertion failed. Exiting with status code 1.

assert command failed. Exiting with status code 1.

Exited with code exit status 1
CircleCI received exit code 1
```

Image 14. Example of failing job on testing website using Google Lighthouse

5.2.4 Storage

As described above in the 5.2.1 Deployment with the given architecture the extra storage places should be involved in the website base for the correct and accurate performance of Statamic.

Statamic has 3 core things that have to be considered to be located remotely, and they will be described below.

5.3.4.1 Assets storage

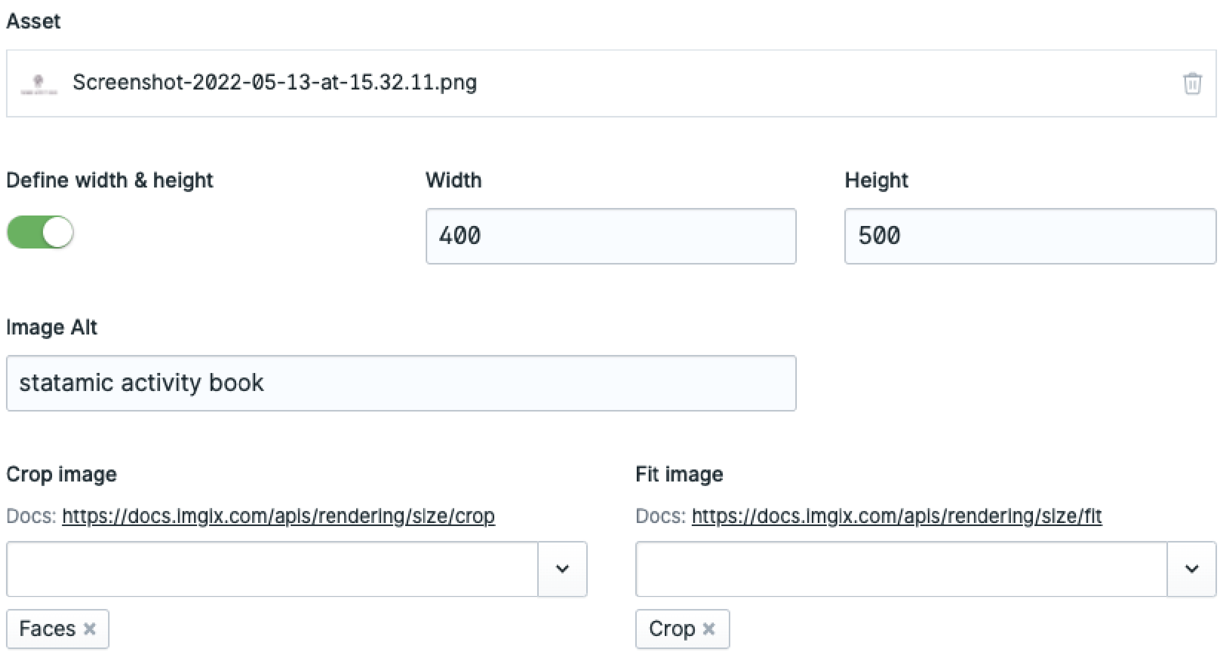
Assets are visual contents that are used to be displayed on the website.

Fixico already uses Google Cloud Storage as the main remote location for storing the assets in connectivity with an image driver that makes images dynamic, compressed and responsive - Imgix.

Right now Fixico employees have direct CRUD rights to Google Cloud Storage (GCS) and Imgix to insert image URLs on the website. But apparently, this approach is not convenient to those, who are editors of the website and do not know how the asset architecture works.

Imgix has an official support plugin to the Statamic CMS, but its functionality does not align with the Fixico needs: building URL queries to display images exactly how the company wants. Therefore an adapter for Imgix was created where with the given properties of width, height and crop attributes an image with desired URL is created.

Statamic supports image containers, basically the GUI to have CRUD rights on the images. Out of box Statamic supports only local assets driver, meaning only from the localstorage. Therefore an additional driver was installed to the website-base that configures Statamic to support an additional GCS driver using Google Application Credentials. Once the adapter works successfully, the Google Cloud Storage disk is available for editors to use:



Asset
Screenshot-2022-05-13-at-15.32.11.png

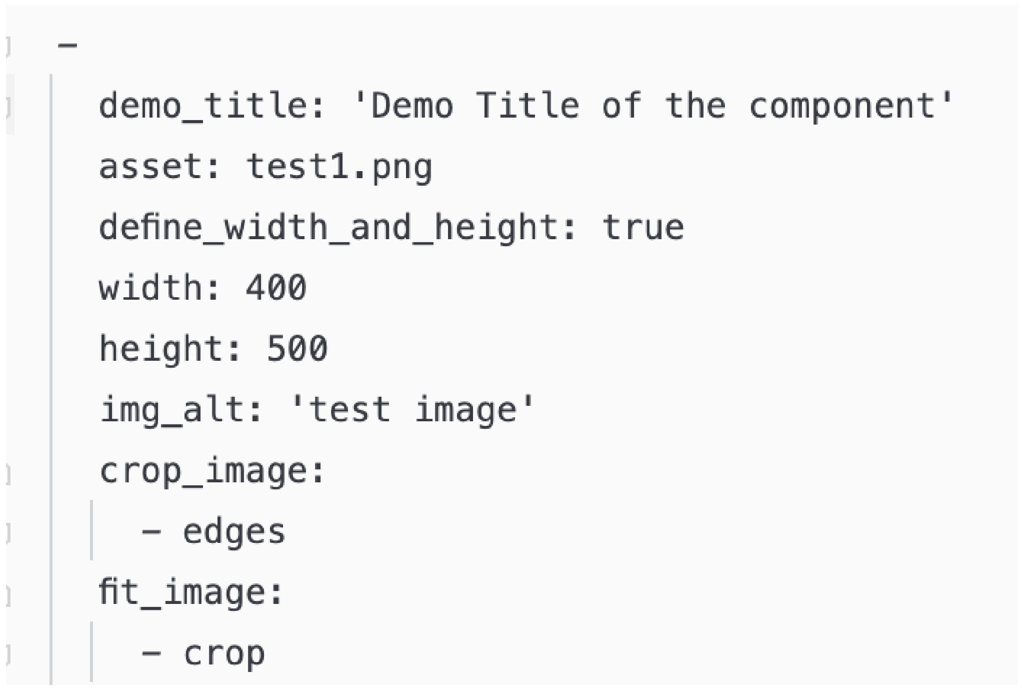
Define width & height	Width	Height
✓	400	500

Image Alt
statamic activity book

Crop image	Fit image
Docs: https://docs.imgix.com/apis/rendering/size/crop	Docs: https://docs.imgix.com/apis/rendering/size/fit
Faces	Crop

Image 15. UI of asset picker.

The image above shows the UI of the image picker component. Here an editor can choose the image under the label "Asset" in a pop-up window and give it settings below. Once the image with configuration is saved in the control panel, the following data would be stored in the content of the component.



```yaml
-
  demo_title: 'Demo Title of the component'
  asset: test1.png
  define_width_and_height: true
  width: 400
  height: 500
  img_alt: 'test image'
  crop_image:
    - edges
  fit_image:
    - crop
```

Image 16. Example of what data Statamic stores for each image.

The Statamic saves the relative path from the Google Cloud Storage bucket name to the path of the file together with the sizing properties of the image.

When the component containing an image is rendering, an Imgix adapter indicates all the image settings properties, and generates the complete URL to the images with desired settings in it.

```html
<img
src="https://next-website-assets.imgix.net/test1.png?w=400&h=500&crop
=edges&fit=crop" alt="test image">
```

The HTML image result sent to the website visitor.

Imgix was configured to transpile the subdomain of the URL like above (next-website-assets) to the Google Cloud Storage bucket URL, where the filename (test1.png) is the relative path from the root of the bucket. Besides this, Imgix is configured to store cache already requested images and to automatically compress images to webimage-standards.

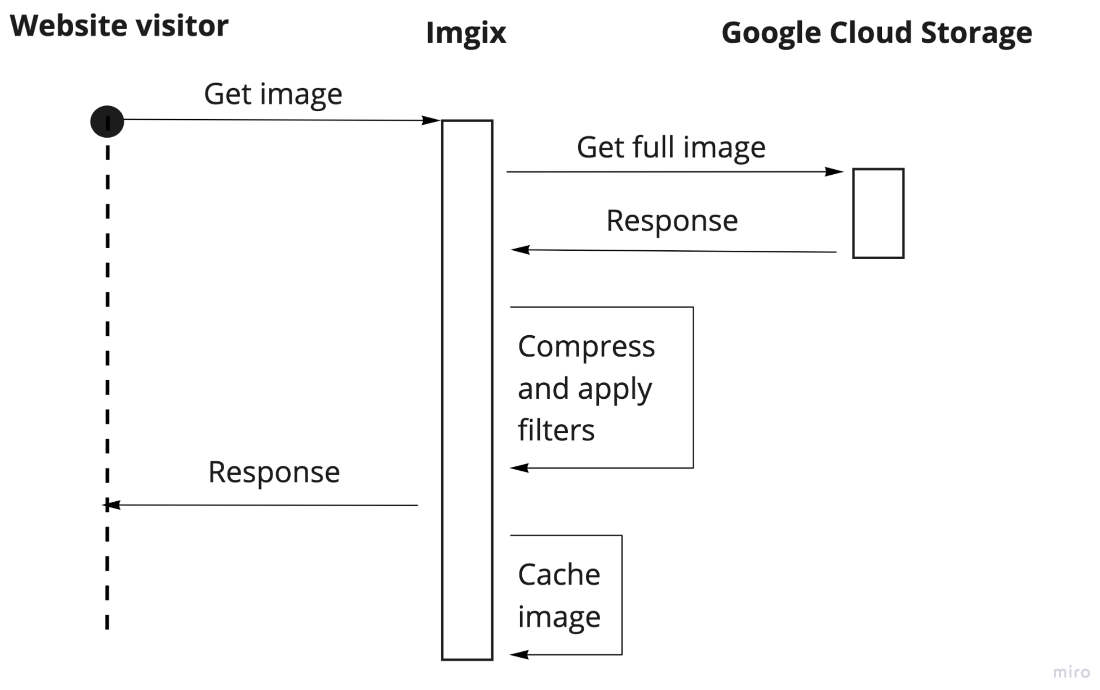


Diagram 1. Sequence diagram of image request.

On the diagram above the flow of image request is shown. Here it is clearly seen that Imgix is a middleware that serves as the caching and filtering images tool.

And once the image is already cached, the image retrieval flow is smaller and faster.

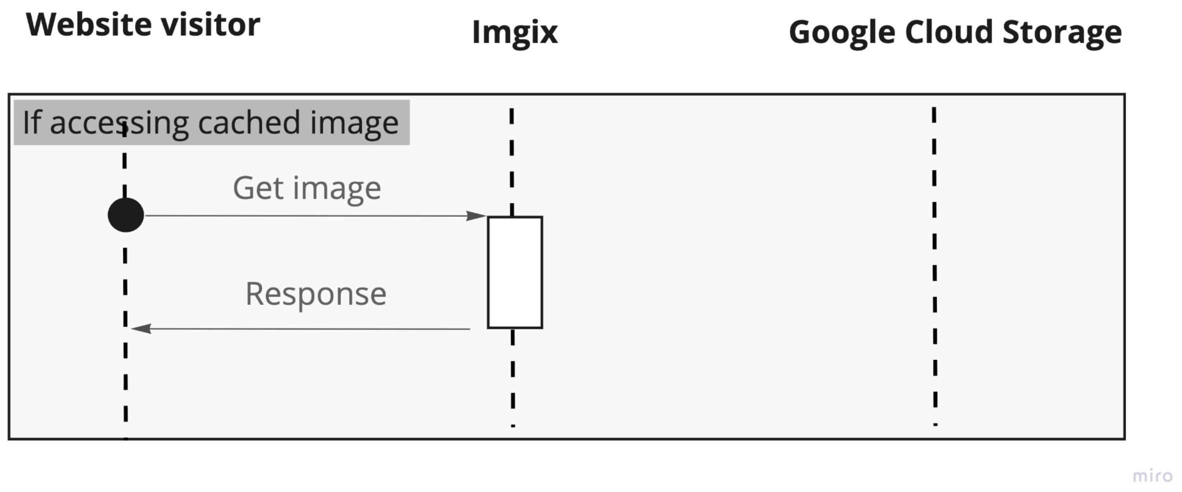


Diagram 2. Sequence diagram of cached image request.

5.3.4.2 User storage

Users are people who have access to the website's control panel.

By default, Statamic stores users in files, under the /users/ project folder. However this solution would mean that users would be lost each time a new deployment is made.

Therefore, an alternative solution was brought, which is to arrange the user storage in the database. Statamic supports this functionality, because for large-scale websites user arrangement using a file system would be really slow.

Fixico obviously already has database in the Kubernetes environment, and it is MySQL data base.

In order to start the process of migration, MySQL credentials should be allocated to the **`/config/database.php`** list of adapters.

Statamic makes use of the Eloquent driver, which does all operations with databases. It is a handy tool to work with databases and build small-to-large queries. Using Eloquent all the user authentication and further processes would be held.

To start the flow of using database, the migration - a database schema should be created. This schema is a set of PHP files, where the insights about tables and their columns are described. For the Statamic users, a handy command line tool creates migration files.

When there is anything to be migrated to the database, PHP Laravel has a command line tool that automatically arranges that - **`php artisan migrate`**

5.3.4.3 Content storage and retrieval

The website-base content is stored under **`/content/`** folder. it has the information about pages, their content, navigation links, and tree structure of the website pages with components.

Statamic natively supports only content storage in the local filesystem, which is not the solution for using Kubernetes deployment.

Therefore a solution was brought to implement the back-up feature for the content. All the website-base content would be stored locally and remotely, the contents of those 2 will be identical. And a backup tool will be installed and integrated that will synchronize local version with remote each time the change is detected. The remote would be Google Cloud Storage (GCS), because it's fast and reliable storage.

This approach is good, because it would also include the version control system. As it has been described above, Kubernetes stores unused previous versions of the same pod with all the information in it. Meaning that in case of the current website production, there is always a chance to revert the content changes made in the last build to the previous version.

(Activity diagram on how this stuff would work in different cases will be added before the final submission, because the solution is not stable on 29.05)

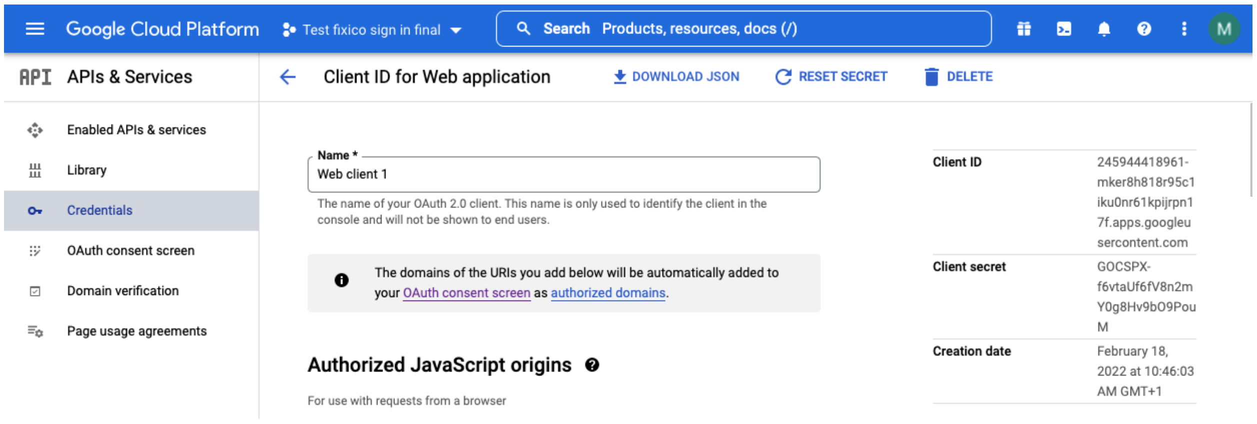
5.2.5 Authentication using Google Identity Provider

Statamic supports extending it's core functionality and their adapters, so therefore a Laravel Socialite OAuth2 identity provider was found and integrated into Statamic website by adding a new dependency to composer.json file.

Laravel Socialite by default supports only the most popular identity providers, and Google is one of them. The configuration of Google OAuth2 requires to provide Statamic the following values:

- Google client id - a string value of Google project that will be authorized to client's google account
- Google client secret - a password of Google project that will be authorized to the client's google account.
- Redirect - a URL that the user will be redirected to after the login is successful.

These values firstly need to be generated and retrieved in the Google Cloud Console web application user APIs&Services -> Credentials tab.



Name *
Web client 1

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

Client ID	245944418961-mker8h818r95c1iku0nr61kpijrpn17f.apps.googleusercontent.com
Client secret	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
Creation date	February 18, 2022 at 10:46:03 AM GMT+1

<u>Image 17. An example of how the Google Authentication identity provider looks like.</u>

Besides this, additional configuration is required to the Statamic modules to finish the Google OAuth2 implementation. The details can be found on Statamic Documentation - OAuth.

5.2.6 Addons

The website project has 2 addons that work with Statamic:

5.3.6.1 Statamic Activity Book addon

Activity Book is activity tracking addon that tracks all changes made by the users of Control Panel.This addon stores the activity log to the database configured next to the project. Which makes it a perfect solution to the Kubernetes/Docker architecture. (Statamic Activity Logger, 2022)

This is a requested feature by maintainers of websites.

Statamic addon marketplace does not have a solution for desired feature, so therefore a new addon was created.

This addon registers event listeners on CRUD operations with content of the website, and each time an event listener triggers, a new log is stored.

With the given Kubernetes architecture, logs should be stored securely, so they won't get lost when new deployment releases. Therefore, all logs are stored in the database configured to the project.

A new tab with the logs overview is created in the control panel. The page is located in the Utilities tab. It has a list of activities ordered by time from the latest to the earliest supporting search by activity functionality.

Addon has a ServiceProvider.php file that serves as the starting point for registering routes and subscribes on CRUD Statamic events.

Statamic Activity Logger was considered to be published to the Statamic Addon Marketplace, because it adds flexibility to updating the package. Statamic website base is expected to be expanded and maintain many Fixico websites, and therefore a solution to have reusable activity book should be implemented. Therefore, Activity Book has its own repository and is publicly registered in composer packages. This way, all future Fixico websites working on Statamic would have a composer dependency with activity book. In case the changes should apply for activity book, the representative repository will be updated, and it will be only a matter of updating an Activity Book version in the composer file of Statamic projects.

The manual on how to install and configure an addon is available on the official website of Statamic addons: https://statamic.com/addons/mark-fixico/statamic-activity-logger

5.3.6.2 Multi-User Collaboration addon

This is another addon requested by the marketing team for a secure collaboration with different people in 1 webpage.

Statamic has released its own addon that supports requested functionality.

Addon uses a pusher broadcast tool as a notification tool for other Statamic users working in a control panel with information about client and locked components.

More information about the addon is available on its readme manual.

5.2.7 System Architecture

With the given information about architecture parts described above, an overview of the whole system architecture is present.

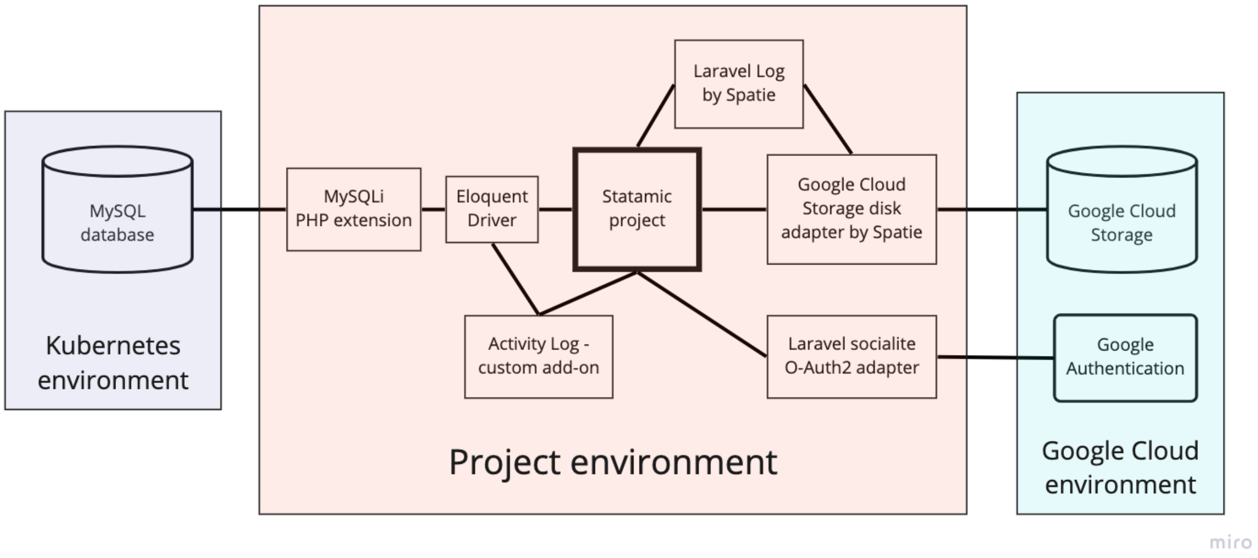


Diagram 3. Architecture diagram

In the diagram above there are all the outer and inner components of the system that are communicating. The same architecture works on local environments (if somebody runs the project) and on remote Kubernetes environments.

5.2.8 Development frameworks

The project right now serves as the base that is easily extendable, so therefore an overview of the technologies to be used to create new website pages and components is described below.

5.3.8.1 Statamic Antlers

Antlers is a simple and powerful templating engine provided with Statamic. It can fetch and filter content, display and modify data, tap into core features like user authentication and search, and handle complex logic. (Statamic Documentation - Antlers templates). I have used it for templating demo components, and apparently it's super easy to use, and makes the html templates much more readable to user developers than Blade, Twig and other templating engines used by Fixico.

As it's a Statamic's inner templating engine, it's available out of box, and to start using it, the template file in **`/resources/views/`** should have **`.antlers.html`** file extension.

5.3.8.2 Alpine JS

A frontend lightweight framework that inherits JQuery behavior and performs all frontend manipulations. It can be used to hide, show, change, or replace any sort of information depending on the behavior inside of the web page or component.

Alpine is loaded to the website base and can be used by adding the behavior that needs to be followed to the templated html file.

5.3.8.3 Tailwind CSS

It's a CSS framework that is used to decrease the CSS class files loaded to the website visitor. Using only Tailwind CSS classes, it's possible to achieve any visual appearance. The framework loads its classes depending if they are called from the project, which makes it an optimal solution for storing and using only used classes.

During development of website components and their appearance, it is common to use development Tailwind watcher that would update CSS whenever there are new changes to them, it can be called using the following command `npm run watch`. That would trigger a watcher that updates the website each time there are new changes to it.

Before the newly developed website pages and components would go to remote environments, the production compilation of javascript and Tailwind CSS will be generated using a command `npm run production`.

Tailwind still needs to be configured to support the color palette the website is using, and whether it's needed to change the default definitions of classes used by Tailwind. Website base has a `tailwind.config.js` file, where all custom properties used by the demo html files and color palette were added.

5.3.8.4 Helper environment

As it can be noticeable above, the project uses NPM (Node Package Manager) during development. NPM is a great tool that assists working with frontend frameworks during development.

Laravel Mix is used as a powerful bundler that is compatible with PHP and prepares JavaScript and CSS for the browser. It inherits webpack core principles.

There is a `webpack.mix.js` configuration file of Laravel Mix that defines the behavior of primarily usage of Tailwind CSS classes.

5.2.9 Manual on how to run the project

5.2.9.1 Environment prerequisites

To run the Statamic project, the environment should have:

- PHP version 8.0
- Composer - 2.0 - a dependency management tool for PHP packages

To develop Statamic project, more development should be:

- Node js - 16.0 - a JavaScript runtime build environment
- NPM (Node Package Manager) - a dependency management tool for JS packages

5.2.9.2 How to run the project

It is expected that the person who reads this chapter already has a clean installed project.

1. Install dependencies using composer install

2. Generate the Application key using command php artisan key:generate
3. Create a `.env` file and allocate there the values of the internal and external components that Statamic would work with

If the project needs to run, so it will be accessible on browsers:

- `php artisan serve`

If the project is going to be extended, and more functional and frontend development is going to be performed:

- `npm run watch`

5.2.9.3 Further website development recommendations

The website base that I have been working on is complete. It is expected that a new repository for each website migrated will be created, and website development would be done in a separate repository.

To start developing the website, the following should be configured:

1. Define the languages the website is going to use, by following the approach described in answer to a research question 4.2.4 - support of internationalization and localization.
2. Configure remote storages, such as Google Cloud Storage and MySQL database with new buckets and databases. This step is necessary, because system architecture highly relies on external providers described above. Also it's important to store all the credential environment variables in a `.env` file, and update .env.example file with the key names of secret values.
3. Start developing website components by following the guides given in the Statamic Peak manual on how to create components (Peak Screencasts, 2021), and following the frontend structure of using configured frontend frameworks that are described in chapter 5.3.8 - Development frameworks.
4. Configure CircleCI app to expect the website repository's CI/CD pipeline, otherwise CircleCI would not run the pipeline.
5. Keep in mind Lighthouse testing that will be running each time a new push is launched to GitHub repository, so therefore Lighthouse optimization should be kept in mind during the whole development process.
6. Configure deployment environment, where 2 website environments will be arranged such as staging and production, with representative website URLs.

These are the main concepts of starting development process on a Fixico website migration.

6. Conclusion & Recommendation

This project was proposed by Fixico BV to answer the following question:

"How can I solve Fixico's "multiple website maintenance problem" which is caused by decentralization and deprecation, using Statamic CMS platform in order to archive unification for all known Fixico websites catered towards developers, end users and content writers?"

The answer I came up with is the optimal solution, where all website stakeholders should benefit from. This chapter describes an overview of what has been delivered at the end of the graduation project, what are the further recommendations on continuing with the website base, and reflection on the assignment that was done.

6.1 Proof of concept

The Proof of concept was described in chapter 2.3 The assignment , in short POC has the following:

- Website components that together serve as the homepage of an existing Fixico website
- Configurable remote environment, where website is running
- All implementation decisions that constrains Fixico websites architecture
    - Uses existing environments for deployment of the website
    - Handles all possible website data savings in predefined places
    - Uses existing continuous integration and deployment environment
    - Is adopted to editor needs
    - Is adopted to website user experience

All the functionality described and desired from POC has been archived. This means, the website base made on the Statamic CMS framework is ensured for Fixico websites to be rebuilded on.

However, it could be debated that the additional success factor of actual migration of 1 website, or at least a part of it was actually done. Unfortunately, there wasn't much time for website migration, and therefore only 1 page was migrated and serves as a proof of concept that Statamic is capable of handling FIxico website architectures.

6.2 Recommendation

The Fixico company has come up with an idea to find a way to create a new, united CMS that would be used to arrange all websites Fixico has. Moreover, the field of view was predefined for me, as I had to use a specific CMS because it has many justified good advantages. After researching a topic to have 1 shared admin panel among all different websites that will be made on Statamic (Research question 3.2) the research has correlated with answering the question whether it's possible to have multiple, not related to each other websites in 1 project, or in 1 place. As the research has shown, Statamic is not supported for the described feature (Research question 4.2). However, I find Statamic as a great framework that can be used for optimal solution that would benefit all new websites stakeholders: to have many different Statamic projects for different websites, that would share the same base between each other.

This way, an optimization of management and control of locales on some websites can be achieved using 1 Statamic project as a handle for the website with its locales. For example: Fixico customer websites have 3 different languages and each website locale is stored and handled as a separate website, so the admin panel is only 1 per locale, rather than 1 admin panel per website.

In general, approaching having multiple Fixico websites running on Statamic would be beneficial to the system architecture consistency and would help all the developers to be proficient with 1 specific technological stack that would be needed for adjusting Statamic projects whenever needed, rather than knowing 5-10 different frameworks that are used among all Fixico websites.

The recommendation on how to start with the development can be found in chapter 5.2.9.3 Further website development.

To summarize, I recommend Fixico to go with Statamic as a core framework for website migration, because it supports a lot of new good features and would be a good upgrade as functional-wise as technical-wise for Fixico, where additionally more websites can be grouped together.

6.3 What is delivered to Fixico?

At the end of graduation assignment, Fixico would receive a proof of concept Statamic project that ensures that migration to Statamic CMS is possible, and also Fixico would get the Graduation report document, where all the decisions and technical implementation are described.

The Statamic project has a demo page, and implemented all architecture decisions from the CMS perspective that comply and align with Fixico architecture.

6.4 Reflection

All in all, it was not an easy assignment, where I had to prototype an ambitious project that would be really beneficial. To have more insights on the timewise perspective, please go to chapter 2 project timeline in Appendices.

On a technical level, I believe I conducted myself in a way that satisfied the expectations of my supervisor and other developers. When it comes to the result, management was happy with my progress and the prototype that I delivered. While in the beginning, I seriously doubted my capabilities, thinking that I bit more than I can chew, by working on concepts that flew over my head at the time, I managed to rise to challenge, and I am quite proud of what I have achieved. However, I enjoyed it and learned a lot of new concepts, and how to use them. It was the first time I designed and worked with a static website architecture, and all the people around me were incredibly supportive and helpful, and I am very grateful for the help, for I could never have finished this assignment without their help.

With a 0 background knowledge of how static websites do work, I had to learn a lot of different concepts and practices used among Fixico and competitors' website projects. This has brought me quite some pressure because of the knowledge to be gained and the time I was limited to, but in the end I am happy with what I have learned, archived and delivered.

That's being said, there are plenty of things I feel I could have done better:

1. Sprint failures. In total I have had 2 failed sprints. Main reasons for it were wrong time estimations for the tickets where I had more to guess the amount of hours the ticket would take to be done. This has resulted in shifting planned features to be developed. Also it has resulted in fewer UI components that were migrated to Statamic base.
2. Understanding the editor's concept of working with website page components. I have run into a small misunderstanding problem, where I estimated that only the text or an image can be defined from the control panel, while all the layout properties will be handled from the template files only. This is a mistake, because as the time has shown, editors always want changes in the layouts of components they are working with (such as change the text position, colors, wrappers, background colors) and others. So the problem I have run into is the limitation of a freedom for editors on the layout part of the editing. I realized it too late, at the time when a lot of components were already developed and there was no time for the changes. However if I had a choice to do something differently, I would pick this problem, because the more flexibility the control panel has for components, the less maintenance work developers should do.

Appendices

1. Use of Research methodologies

The graduate has made an extensive research on new and unknown to him topics, where the following methodologies were primary:

1. Desk research. The graduate has learned a lot about the tech stack, how to extend it correctly, and how to arrange new things.
2. Interviews with stakeholders of the system the graduate has built, and with others who are expected to know the answers on given questions.
3. Competitor analysis. The other project having the same or similar structure, where the desired or researched feature works successfully.

1.1 Interviews

Interview with a person from the marketing team.

Marketing team is an active stakeholder of the websites, because they take the main role in maintaining the website from the content part. This team actively maintains existing websites, and therefore it was interesting to talk about what the marketing team expects from the new website-base.

Therefore, an interview with Isabell Heller was conducted in March 2022, she is a marketing manager and an active maintainer of current Fixico websites. Up to a graduate's mind, she is the right person who should know all the drawbacks, problems and good parts that are happening with the websites.

Before an interview, Isabell was informed about it, and asked to think about what would be new great features that would improve the satisfaction level of website maintainers in future.

Another thing before an interview, a list of requirements that graduate has gathered doing a desk research on what are the core things the content-management system should do, alongside with existing features of admin panels. Those requirements were basic and didn’t have prioritization, it was more of a point where to start from.

The list of requirements is described above:

As an editor, I want to login to the admin panel, so I can securely work on the website.
As an editor, I want to have CRUD rights on pages
As an editor, I want to have CRUD rights on components on the pages
As an editor, I want to optimize the SEO values of the page.
As an editor, I want to have a language picker, to work with specific locale versions of the website.

The interview had 2 phases:

1. Requirements gathering and their prioritization.
2. Ideati ng on what could be improved where 2 feature ideas pitched:
    a. Create or integrate a multi-user-collaboration tool - basically an opportunity to work on the same page / component for multiple people without a risk to lose entered data.
    b. Create or integrate an activity tracking tool, where it will be seen who and what has been working on.

As an outcome of the interview, a list of maintainers requirements was gathered alongside with their prioritization using MoSCoW metrics.

Acceptance interview with Isabella from marketing team

This interview was introduced to showcase the functionality of the new website's control panel. The goal for the interview was to receive feedback from the marketing team, as they are important stakeholders in the assignment.

Interview was conducted with Isabell Heller, she is marketing manager, and I already had an interview with her before.

Before an interview, I have prepared 1 page of the Fixico website that Isabell also works with. This way she can compare precisely the experience she has right now working, and potential in future with Statamic. Another thing here can be the layout flexibility of Statamic components.

The outcome of an interview was the satisfaction level of Isabell with flexibility among many things that Statamic is configured to support. This is a great result for me, because I understand, the cms development goes in the right direction. However, Isabell has suggested that I think more about the flexibility of layouts for components on web pages, because the more layout is flexible, the more the editor can do with the control panel alone.

In conclusion, I am glad with the interview, and especially with the useful feedback point to add more flexibility to layouts sections. This is definitely the direction I should take and go this way.

Interview with Fixico CTO

Fixico's CTO is a person who is an architect of all solutions Fixico arranges for their projects, and is an important person on each decision making process.

An interview with Mujib Azizi was conducted in April 2022, he is Fixico's CTO and plays an important role in the decision making process.

The reason an interview was held was to discuss the decisions and it's ground a graduate has on certain topics that do or do not align with company architecture infrastructure.

Before an interview, the following topics were planned to be discussed:

1. Does Fixico use tests or any testing frameworks on websites?
2. What solution should be for deployment: Virtual Machine, or Kubernetes?

This is a tricky question because Kubernetes is known to have many advantages because it's an existing platform and already supports many things that should be configured to VM before accessing environments. However, the question was raised because VM would solve a meaningful problem - content storage of the website.

3. What solution is better to store the content of the website: backups or repository?

   This question has popped up after the Kubernetes environment was chosen in the discussion of the previous question.

   Main reason to ask this question is to get disadvantages of using a repository, because at first glance it is widely used among Fixico websites.

4. How do fixico CircleCI orbs work, and how to get access to them.

   It's a technical question targeting to get insights on how the pipeline defines and what reusable components I could take to build and deploy my project as much inline with Fixico existing software as possible.

The interview was long and very interesting from technical perspectives that were touched during answering planned questions.

As an outcome of this interview were the parts of answers to research questions 4.2.5 Metrics used for website testing, 4.2.6 Deployment, 4.2.7 CI/CD, 4.2.8 Content storage

Interviews with developer supervisor

The graduate has frequent interview checkups, where problem solving figures are arizing, they sometimes are unprepared, because it really depends on the cause of the interview:

- Opinion on unsure decision I am thinking of
- What is the problem with Pod in Kubernetes?
- How to solve X problem
- What would be suggested on a certain topic.

2. Project timeline

Sprint №	Time	What has been achieved
1	Q3 Week 1 - Q3 Week 2	Getting started with Fixico, researching how everything works, researching basics of my assignment
2	Q3 Week 3 - Q3 Week 4	Creating Plan of Approach, defining goals for the gradation, Setting up Statamic boilerplate project.
3	Q3 Week 5 - Q3 Week 6	Answering research questions 1,2,3. Implementing OAuth2 feature.

4	Q3 Week 7 - Q3 Week 8	Answering research questions 4, 6, 7. Implementing CI/CD pipeline, and OAuth2 authorization.
5	Q3 Week 9 - Q3 Week 10	Working on an interim report, connecting CI/CD pipeline with deployment method.
6	Q4 Week 1 - Q4 Week 2	Fixing interim report, Answering research question 8. Starting to implement content storages and connecting them also in a remote environment.
7	Q4 Week 3 - Q4 Week 4	Adding 2 addons to Statamic to fulfill marketing team requirements. 1 Addon had to be created from scratch. Started developing the first page of an existing website to Statamic.
8	Q4 Week 5 - Q4 Week 6	Finished with adding the first page to Statamic. Working on a draft report and its deliverables.
9	Q4 Week 7 - Q4 Week 8	Fixed bugs on the website, prepared final report.

Table 1. An overview of the activities made to the graduation assignment time-wisely.

3. Research additional information

3.1 Google Lighthouse metrics, and how do they influence

Google Lighthouse has 5 basic metrics that rank different aspects of the website or web application:

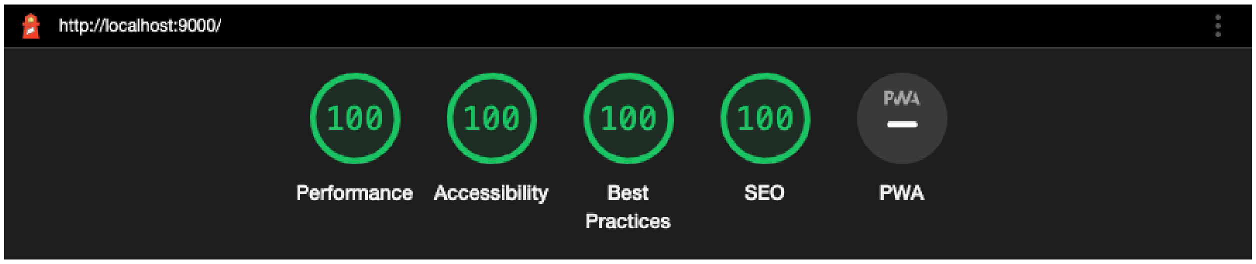


Image 1. The metrics overview after the report is completed.

1. Performance - is one of the most important metrics that measures all technical aspects of loading the webpage. This section covers 6 main aspect keys that are taken into account for the final score:

a. Total Blocking Time (TBT) - also known as First Input Delay, and measures the time in milliseconds between the user's first action and the browser's response time. The other phrasing is a first page response speed score.

   Best practices to avoid poor ranking of TBT are:

   - Minimize the loaded JavaScript. JavaScript files directly
   - Removing or minimizing third-party scripts. Relying on such scripts usually leads to a negative impact on FID.
   - Using browser cache. The scripts that do not change dynamically can be cached after the first website load, which leads to way faster JS loading tasks for website load.

   Google considers this metric as the most important and gives it a score of 30% of the overall ranking.

b. First Contentful Paint (FCP) - the metric measuring the first DOM element to be displayed in the browser. It does not indicate the overall site speed, but reflects the site speed from the user's perspective where the user sees the first element of the site popping up. When any site content appears quickly, their perception indicates a fast-loading site even if the rest of the site takes a bit longer to load.

   Google gives FCP a score of 10% of the overall ranking.

c. Largest Contentful Paint (LCP) - is the time it takes for the largest section of the content to appear on the browser window. It's different from other metrics described here, because the point of view is moved from the content loading things to a user perspective, who should be able to see the majority of the website as soon as possible, which means LCP focuses on what really matters to start interacting with the web page.

   Best practices to improve LCP:

   - Use fast web hosting. The faster web server works, the better the user experience is.
   - Use Lazy loading. It's a script that loads heavy content parts (like images, videos, etc.) only when the content appears on the screen.
   - Avoid using unnecessary third-party scripts. The less data to be loaded to the web browser, the faster it loads.

   Google finds LCP as a highly important metric, and gives 25% of the overall ranking.

d. Cumulative Layout Shift (CLS) - is a measure of how much the website's content shifts position above or below the fold as the page continues to load. It's only when the elements on the web site move around without any input of the user.

   Best practices to avoid CLS problem on the web page:

   - Use size attribute dimensions for any media (videos, images, etc.). This way the user browser knows exactly how much space element takes up on the page, and will reserve the space for the media element on html layout until the media loads.
   - Make sure to dynamically add UI elements below the fold as much as possible. This way the new UI will not push the existing page content down, and user better orients on the page.

   Google gives CLS a score of 15% of the overall ranking.

e. Speed Index - the metric measuring the timing to load and render all content above the fold.

   Google gives Speed Index a score of 10% of the overall ranking.

f. Time to Interactive (TOT) - the amount of time that takes for the page to become fully interactive. Page can be considered fully interactive when the page displays useful content, all event handlers are registered and the page responds to user interaction within 50 milliseconds.

   To improve the TOT, the Javascript files have to be as much as possible optimized together with reducing the main thread work.

   Google gives TOT a score of 10% of the overall ranking.

2. Accessibility - is a metric verifying the website is accessible to people with disabilities too. This includes tests on important elements like buttons or links, to see whether they are sufficiently well described, or whether images have been assigned an alt-attribute so that the visual content can also be described by screen readers for visually impaired users.
3. Best Practices - is a list of audits that check common mistakes in a website to comply with standards of the web.

   Best practices analyzes whether HTTPS and HTTP/2 are used, whether resources come from secure sources and assesses the vulnerability of JavaScript libraries. Other best practices look at secure database connections and avoiding the use of non-secure commands, or incorporating deprecated APIs.

4. SEO - is a ranking where various tests run to establish how well a website can be crawled by search engines and displayed in the search results. These Lighthouse tests that Google describes as "SEO" are extremely limited, but all the returned errors should be fixed, so search engine optimization would offer more potential for other improvements, which should certainly be explored and improved if necessary.
5. Progressive Web Application - these inspections are made for the web apps to be evaluated, what is not the case for the graduate's assignment. The graduate is building a Content Management System, which is different from the PWA website approach.

3. List of references

Backlinko - Core Web Vitals

URL https://backlinko.com/hub/seo/core-web-vitals

Google Documentation - Lighthouse

URL https://developers.google.com/web/tools/lighthouse

I.Sacolic (2022) - What is CI/CD? Continuous integration and continuous delivery explained.

URL
https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuou
s-delivery-explained.html

Kubernetes Documentation (2022) - What is Kubernetes?

URL https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kinsta (2022) - What is Nginx? A basic Look at what is is and how it works

URL https://kinsta.com/knowledgebase/what-is-nginx/

Michael Cobb, Stephanie Mann (2020) - What is OAuth and how does it work?

URL https://www.techtarget.com/searchapparchitecture/definition/OAuth

P.Jain (2020) - An Introduction to CircleCI

URL https://medium.com/xebia-engineering/an-introduction-to-circleci-aa9464a86673

Peak (2021) - Screencasts

URL https://peak.studio1902.nl/other/screencasts.html#statameet-2021

Statamic Official Documentation - Multisite

URL https://statamic.dev/multi-site

Statamic Official Documentation - Antlers Templates

URL https://statamic.dev/antlers

Statamic Official Documentation - OAuth

URL https://statamic.dev/oauth

Statamic Marketplace - Statamic activity logger

URL https://statamic.com/addons/mark-fixico/statamic-activity-logger

Techopedia (2020) - Production Environment

URL https://www.techopedia.com/definition/8989/production-environment

T. Hamilton (2022) - What is Functional Testing

URL https://www.guru99.com/functional-testing.html

Umbraco - What is a Staging Environment?

URL https://umbraco.com/knowledge-base/staging-environment/

Wikipedia - Monkey testing

URL https://en.wikipedia.org/wiki/Monkey_testing