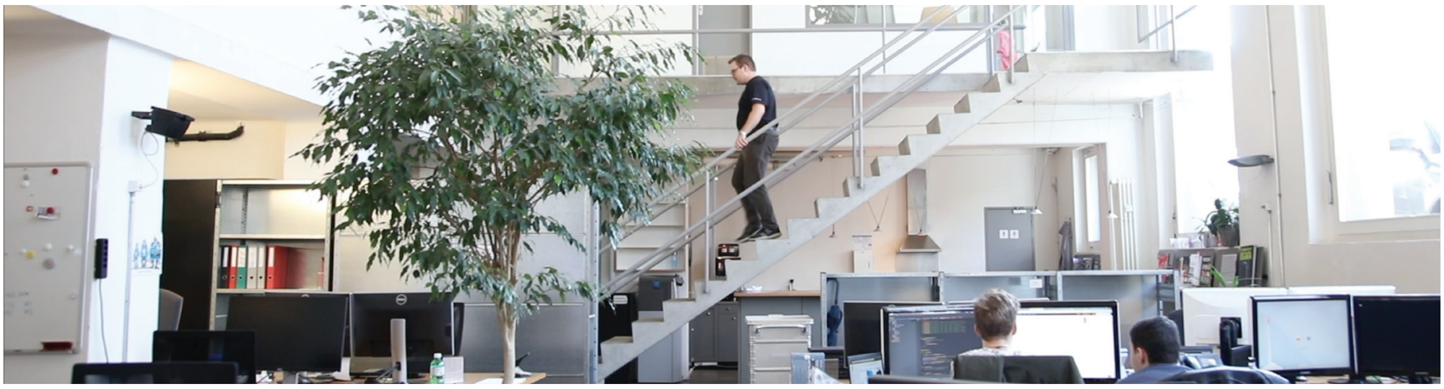
# How Netcetera used Magnolia's fast front-end development to produce software solutions that count

Magnolia's front-end developer tools helped software company Netcetera deliver client projects on time and on budget



**Industry**
Technology

**Country**
Switzerland

**Implemented by**
Netcetera

**Site**
www.netcetera.com

## Front-end approach gives speed and agility to finish web projects quickly

Netcetera is a software development company with over 400 employees in six different countries. Netcetera aims to produce "software that matters", meaning that it develops software that helps their clients reach their digital business goals. Its diverse projects include for example the timetable planning software for the Swiss Federal Railways, easy and secure digital payment processing systems, or virtual reality apps for smartphones. Netcetera often integrates Magnolia in its custom solutions to get the best results.

A software company's website is its most important calling card and communication tool. Netcetera re-designed its corporate website with Magnolia. It also used Magnolia to create a client website for energy provider IWB. Both websites are based on Netcetera's own code-collaboration platform called Hibiscus. Magnolia's fast front-end approach, called light development, gave Netcetera the speed and agility to finish web projects quickly, plus the flexibility to build and optimize the development pipeline that works best for their team.

## The challenge

Netcetera had a new vision of how they want to be perceived when refreshing their corporate identity, design, language and images. For a company producing "software that matters", the corporate website had to showcase their best work and innovative solutions that help clients solve their IT issues from strategy to implementation.

## The solution

### Working with components instead of pages

When Netcetera was re-designing its website, it first focused on pages, but quickly switched to easy-to-use components. Netcetera's marketing department gained more freedom and control over how the pages looked: they could mix and match components, assemble and re-arrange them on a page, re-use them in different contexts, and not be restricted to pre-defined pages.

The component-centric approach meant that developers and marketers were talking about, working on, and viewing and testing the same thing. Components are very flexible to work with and changes can be made at any stage of the project. The team could grow the website iteratively: building the most important components first, then adding new ones and new functionality—components are very versatile.

### Lightweight configuration with YAML

YAML configuration is a huge benefit for developers. Because the key configuration is in simple files, all of the changes are checked into GIT with the actual template files they relate to. Developers can work on them with their favorite text editors rather than having to learn a new tool. And the system detects file changes and automatically reloads the configuration, no server restarts required. In a pinch, a developer can change a configuration on a live server by logging into the resources app on Magnolia AdminCentral. It's fast and easy to use.

### Using content apps to create an information network

Magnolia's apps approach made it very easy for Netcetera to handle content. Netcetera used Magnolia's content apps to create an information architecture that was structured more like a network than a hierarchy. Content apps allowed Netcetera to manage structured data efficiently, e.g. markets, products, contact persons, awards, and were a perfect way to navigate and organize the data.

### Improved collaboration

Magnolia's light development enabled the different Netcetera teams to collaborate better: Java developers working on the back-end, user experience experts working on front-end code. Only the back-end developers needed a full Java development environment; everyone else worked in their preferred (lightweight) text editor or IDE.

### Using light development for IWB

After Netcetera re-designed its website, it used Magnolia and light development features to create the website for IWB, an energy provider in Basel, Switzerland.

## The result

### Faster and on time

Different developer teams—front-end, Angular and Magnolia - could work in parallel. The marketing department could continuously test, review and accept components during the project, resulting in less testing at the end. The process was more efficient and there was a lower barrier for front-end developers to contribute to the project. Thanks to light development, components could easily be tuned until the last minute before project launch.

Netcetera plans to continually improve its website and its front-end connector called Hibiscus. It wants to apply the component-centric approach to more of its code base and build a store of re-usable components for use on multiple projects. The software company also added new features such as drone videos featuring employees.

### A living style guide

Netcetera also leveraged light development to create a connector, called Hibiscus, between Pattern Lab and Magnolia. Hibiscus is a platform that enables front- and back-end developers to work together and work fast by editing the same master files. Pattern Lab is a "living style guide" where developers can put all their front-end components together, preview them in browsers, test them with sample data and discuss them with clients before integrating the components into their software. Hibiscus allows developers to work both ways: you can develop pure front-end components first, and add the CMS integration afterwards. Or you can develop the raw back-end CMS templates first, and then let the front-end developers take over and polish them.

Hibiscus solves the long-standing problem of the design artefacts getting out of sync with the actual website. Previously, front-end devs would develop a design "static prototype" independently, then back-end devs would copy parts of it into the website CMS. Because the file copies are separate and independent, they can get out of sync with subsequent changes. If the front-end devs update the prototype, those changes need to be done in the CMS. Likewise, if the back-end devs update the CMS, those changes need to be done in the static prototype. Hibiscus removes the need for the static prototype. Each component file holds both the front-end design prototype and the actual CMS template—everything now stays in sync because both parts are in one file and are easy to track and update.

Check out Hibiscus on Github:
https://github.com/netceteragroup/hibiscus