

Shiva Pandey

IMPLEMENTATION OF A NORWAY WEBSHOP PROJECT USING WORDPRESS AND WOOCOMMERCE

A Case Study of a Snellman's eCommerce Webshop in Designing a New Webshop Instance.

IMPLEMENTATION OF A NORWAY WEBSHOP PROJECT USING WORDPRESS AND WOOCOMMERCE

A Case Study of a Snellman's eCommerce Webshop in Designing a New Webshop Instance.

Shiva Pandey
Bachelor's Thesis
Spring 2023
Degree programme in Information
Technology
Oulu University of Applied Sci-
ences

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology, Bachelor in Engineering

Author(s): Shiva Pandey

Title of the thesis: IMPLEMENTATION OF A NORWAY WEBSHOP PROJECT USING WORDPRESS AND WOOCOMMERCE

Thesis examiner(s): Lasse Haverinen

Term and year of thesis completion: Spring 2023

Pages: 45

This bachelor's thesis was commissioned by Oy Snellman Ab. A diary-based reporting was done for the thesis where the author reflected the tasks that was done during the implementation of a Norway Webshop using WordPress and WooCommerce. The main aim of the project is to design and implement eCommerce site by using WordPress and WooCommerce.

The objective of the thesis is to study an existing Snellman's eCommerce website named SnellmanPetFood (MUSH) Webshop. SnellmanPetFood webshop is designed to shop cat and dog foods. The webshop is designed to sell products in Finland and Sweden. The phases of the case study were planned to know the system structure, data flow of the existing website, WordPress themes and plugins. A case study of SnellmanPetFood (MUSH) Webshop was done during the first phase and implementation of a Norway webshop in the second phase.

The prior implementation plan of a Norway webshop was changed. The new target of the implementation plan was to add a Norwegian language to an existing site along with PLM, Infor M3 and shipping fee logic configuration, instead of implementing completely a new site.

Keywords: Bedrock, Infor M3, product-lifecycle-management, WordPress, WooCommerce, eCommerce, webshop,

CONTENTS

1	INTRODUCTION	5
2	OY SNELLMAN AB	7
3	WORDPRESS.....	9
3.1	WordPress with Bedrock	9
3.2	Webshop plugins.....	10
3.2.1	PLM inventory plugin	11
3.2.2	M3 stock plugin.....	11
3.2.3	M3 order plugin.....	12
3.2.4	M3 warehouse plugin	12
3.2.5	M3 customer plugin.....	13
3.2.6	M3 price list plugin	13
3.3	Multisite feature in WordPress.....	14
4	TOOLS AND TECHNOLOGIES.....	15
5	EXISTING SYSTEM STRUCTURE	21
5.1	System structure	21
5.2	User flow diagram of the webshop	23
6	IMPLEMENTATION OF A NORWAY WEBSHOP.....	25
6.1	Implementation with prior goals and objectives	25
6.1.1	Planning.....	25
6.1.2	Creating a new webshop instance in multisite WordPress.....	26
6.1.3	Installing and customizing plugins.....	27
6.1.4	Themes and layout of the webshop page.....	28
6.2	New target for the implementation of a Norway webshop	29
6.2.1	Add a Norwegian language to the existing site.....	30
6.2.2	Configure PLM.....	30
6.2.3	Configure M3	32
6.3	Results	36
7	ANALYSIS OF THE CHANGES MADE IN IMPLEMENTATION PLAN	40
8	CONCLUSIONS.....	42
	REFERENCES	43

1 INTRODUCTION

Any online business that sells goods or services now relies heavily on an eCommerce platform to manage their website, marketing, sales, and operations. Online purchasing has increased because of advancements in digital and internet technology. The popularity of online purchasing has permanently altered consumer behaviour.

Oy Snellman Ab has launched a B2B ecommerce platform for MUSH Oy to sell products and services directly between two businesses. MUSH Oy is one of the organizational units of Oy Snellman Ab. A webshop named SnellmanPetFood for MUSH Oy was designed with modern WordPress and WooCommerce to sell cat and dog foods. There are 8 organizational units in Oy Snellman AB. Each organizational units have their own unique products and services to sell to the consumers. Therefore, there is a need for building eCommerce platform for each organizational units in Oy Snellman Ab.

WordPress multisite and WooCommerce platform are used in the design and implementation of the SnellmanPetFood eCommerce platform. The webshop's technological objectives are to offer full control over integration with PLM (Product-Lifecycle-Management) and ERP system M3. Webshop also aims to support complex B2B webshops with customized warehouse shipping routing. Webshop implemented in WordPress also offers the option of total control over theming and designing of a simple webshop. In addition, the SnellmanPetFood website is adaptable to various markets, nations, and currencies.

The main aim of this thesis is, to demonstrate the process of designing and implementing a new webshop instance, named Norway webshop, based on an existing Snellman's ecommerce webshop. The primary objective of the thesis is to implement a new webshop instance in multisite WordPress and WooCommerce platform. This thesis also examines an existing SnellmanPetFood webshop to gain knowledge of its system architecture, data flow diagrams, WordPress plugins, themes, and layout. Plans, designs, and implementation of the Norway webshop would be based on existing SnellmanPetFood webshop.

Creating a new webshop instance on Snellman's ecommerce platform, installing and customizing plugins, and completely altering the site with Gutenberg components for themes and page layouts are the fundamental needs for building the Norway webshop. Additionally, setting up PLM, Infor M3, and shipping logic are prerequisites for setting a Norway webshop into operation.

Implementation of the Norway webshop is targeted to the Norwegian retailers. The Norway webshop is implemented to provide B2B customers in Norway to choose, order and buy the MUSH Oy products in a convenient and efficient way. MUSH Oy wants to build the Norway webshop to expand its business in Norway. As a part of the workplace, the author should develop and demonstrate skills of applying theoretical and practical knowledge to use tools and technologies and build a new webshop instance in a multisite WordPress and WooCommerce platform to accomplish the Snellman's ecommerce demands.

The total amount of time spent for the implementation of the Norway webshop was approximately two months. The process involves learning existing webshop and regular meetings and feedback loops, planning, designing, and implementing of the Norway webshop.

2 OY SNELLMAN AB

This thesis is commissioned by Oy Snellman Ab. Oy Snellman Ab is the parent company of the Snellman Group. Markus Hellström serves as the Snellman's chief executive officer (CEO). The Snellman Group is a family business which was established at Jakobstad in 1951 (1).

Meat processing and ready meals are two business areas of Snellman Group. The companies and brands associated with meat processing groups includes Snellmans Köttförädling AB, Figen, Fodax and MUSH. Snellmans Köttförädling AB is a meat processing company. It is Snellman Group's largest and oldest sub-area. The company's headquarters are in Jakobstad on the Ostrobothnia coast. The company operates in Finland and Sweden. (2.)

Companies and brands linked to ready meals include Mr. Panini, Carolines Kök, Kokkikartano, and Snellman pro. Sweden's top supplier of fresh prepared meals is Carolines Kök. A Finnish manufacturer of prepared meals is Kokkikartano. The professional kitchen partner Snellman Pro operates similarly. Snellman's proficiency with Finnish meat, Kokkikartano's understanding of prepared foods, and IceCool's global skill in frozen food are all reflected in the Snellman Pro line. A fast-food brand produced by Snellman Group under the category of ready meals is Mr. Panini. (2.)

The information technology (IT) competency at Oy Snellman Ab can be used to describe a wide range of knowledge bases and skill sets related to business and IT technology. The requirements to an individual's IT competency at Snellman are based on their own core proficiency and knowledge of IT systems. Yet, a company's IT competency typically relates to its understanding of IT and its progress, as well as its fundamental business and professional skills.

IT professionals in Snellman needs to have a solid understanding of hardware, software, and programming skills. IT personnel need to have the ability to learn new skills and adapt to changing conditions. Project management, planning, and adhering to corporate

policies are other requirements. The author has a good understanding of software and hardware knowledge, programming skills and, uses and importance of IT. The need for developing an eCommerce platform is an essential part of business growth in Snellman.

3 WORDPRESS

WordPress is an open-source content management system (CMS) program used to create, alter, and maintain websites. WordPress is technically a PHP application that uses a MySQL or MariaDB database to execute. WordPress has a plugin architecture and a template system, that allows to modify any website to meet company's requirements, blog, portfolio, or online store. (3.)

Snellman's eCommerce platform is built using modern WordPress. The webshop is enabled with multisite feature. This chapter explains the use of Bedrock as a boilerplate in WordPress and webshop plugins, and significance of enabling multisite WordPress feature in existing Snellman's webshop development.

3.1 WordPress with Bedrock

The webshop uses Bedrock as a boiler plate to build modern WordPress. A boilerplate is a standardized framework or set of files that serves as a starting point for building new websites, thereby saving time and effort by eliminating the need to start from scratch for each new website creation (4). Bedrock is a boilerplate for WordPress that includes composer, simpler settings, and an enhanced folder organization. For adaptations, Bedrock employs WordPress core capabilities and has superior git integration for WordPress development. (5.)

The Roots development team builds and updates tools to enhance the WordPress development cycle. Roots is attempting to align WordPress with the Twelve-Factor app development process with Bedrock. The Twelve-Factor app methodology focuses on formalizing the best ways to create web applications with the overarching objective of enhancing work on a growing codebase. (6.)

Figure 2 demonstrates on how the Bedrock structure differs from classical WordPress structure. With certain enhancements, such as renaming wp-content/ to app/, Bedrock's organization is comparable to putting WordPress in its own subfolder.

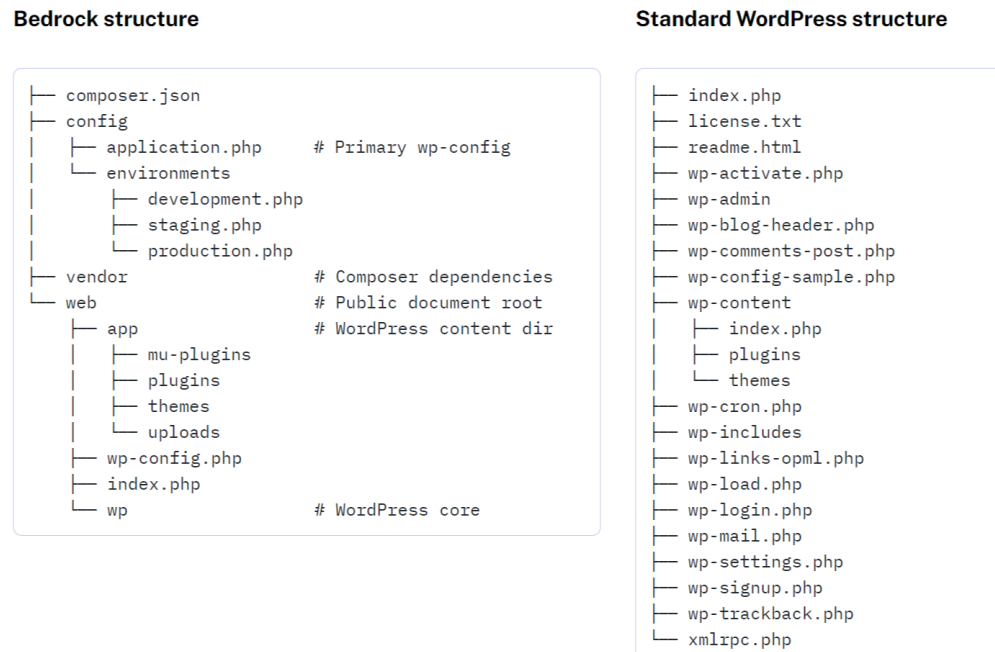


FIGURE 1. Bedrock structure vs standard WordPress structure (5)

Figure 1 shows that WordPress with Bedrock has an improved folder structure with composer and easier configuration. Most of the files necessary for WordPress to function, including the core files as well as pre-configured and customized themes and plugins, are not stored in the repository. Instead, those files are downloaded on the fly as the final applications dependencies, as specified in its composer.json file. (6.)

3.2 Webshop plugins

A piece of software that plugs into your WordPress website is known as a plugin. The use of plugins makes it possible to build almost any type of website, including eCommerce storefronts, portfolios, and directory sites. Plugins can also improve current functionality in the WordPress site. Depending on their functionality, plugins can make minor adjustments or significant alterations to the site. There are numerous uses of plugins in WordPress sites. For example, in eCommerce websites, plugins can be used to adjust payment gateways and allow customers to make reservations online. Similarly, plugins are used to enhance the contact forms, backup site in case of data failure and scan site for broken links. (7.)

In Bedrock structure, WordPress plugins are found in the web/app/plugins directory. There are significant number of plugins used in Snellman's eCommerce site. The site uses several lists of third -party plugins and customized plugins. Below is the list of customized plugins used in the webshop.

3.2.1 PLM inventory plugin

PLM inventory plugin is a necessary plugin that synchronizes products from the PLM system for product information management into the webshop. The webshop should mark goods as unpublished drafts when they are added or deleted so that they may be evaluated without worrying about data loss. There should be an option to only synchronize manually generated and configured products, disabling automated creation and removal. For instance, if an online shop creates 15 products and then manually enters their Product IDs, that will be sufficient for the inventory plugin to behave as though the products have been imported. (8.)

When product data is retrieved from PLM via webhooks, this plugin should initiate an event or action. By default, the plugin will have a basic integration for all the data, but webshops can hook into the event. For example, remove product image updates in the case they want to manage the images themselves. Product's ID, code, managed stock (Boolean), name, description, image, category, ingredients, allergens, feeding guide, nutritional values, storage temperature, origin, size, weight, count per package, warehouse, and product's type such as fresh, frozen and normal are the data responsible through PLM inventory plugin. (8.)

The requirements of API endpoints for PLM includes get all product data based on a Product ID and get all products belonging to a webshop. Webhook is sent when a product is added, removed, or updated. (8.)

3.2.2 M3 stock plugin

An optional plugin that updates the stock levels of goods with "Managed stock" settings that were previously imported using the PLM Inventory plugin. The web store can define

whether a product can be backordered after it runs out of stock or if ordering will be suspended until the stock level is back up. (8.)

Without this plugin, the webshop's stock levels may be manually adjusted, or stock management can be completely turned off. Stock quantities are obtained from M3 and are synced into the website only in one direction. The API endpoint requirements for M3 includes get stock quantity based on Product ID and Webhook is sent when a product stock quantity is updated. (8.)

3.2.3 M3 order plugin

M3 order plugin is used to manage the order system for B2B webshop. A delivery date will be included in the order metadata if the warehouse plugin is activated. Technically, order plugin may be utilized independently if warehouse plugin is disabled. (8.)

The creation of an order with items, a customer number, and a delivery date is one of the criteria for M3's API endpoints. The system will recommend a new date if the delivery date is invalid. Additionally, it must return an active M3 internal ID. The criteria of API endpoints for M3 additionally include getting an order based on ID to validate the order and updating an order based on ID when an order is cancelled or refunded. When an order is updated, a webhook is sent. (8.)

3.2.4 M3 warehouse plugin

M3 warehouse is an optional plugin for B2B web stores that manages the warehouse shipping/routing system. Users who wish to utilize this capability must be logged in and connected to a customer number. (8.)

A component will be added to the web store so that customers may choose the day they want their orders delivered on using a calendar. Holidays and shipping days that are not available internationally will be automatically disabled from the calendar. The Snellman will specify these universal constraints. The delivery days from each warehouse for the

logged-in customer number are requested via a GET request to M3 when a customer checks in. (8.)

Get the delivery day count for each warehouse for each customer number as well as the shipping cost are required API endpoints for M3. The delivery date is recorded in the order metadata at the point of placement. Snellman will specify shipping costs and if they are obtained through an API endpoint. (8.)

3.2.5 M3 customer plugin

M3 customer plugin is a plugin to link webshop user accounts with M3 customer accounts. Multiple user accounts may be linked to a M3 customer account. Users will be created manually however they will have a field for searching and choosing a customer number. The billing and shipping information will be imported from M3 once a customer number is connected. Users cannot change their billing or delivery information if they have a customer number. (8.)

Customer ID, price list ID, email, billing fields (first name, last name, company, billing address, address line 1, postal code, city, country, and phone), and shipping fields (first name, last name, company, address, address line 1 postal code, city, country, and phone) are all required fields for a webshop. Get customers by customer number and get all customers or search by partial match are among the M3 API endpoints requirements. (8.)

3.2.6 M3 price list plugin

To synchronize the product pricing lists, B2B sites use an optional plugin called M3 price list plugin. The pricing lists have a lengthy validity span and are updated often. There is also an access to upcoming pricing lists prior to their implementation. Price lists are exclusive to each webshop, but they may also exist for each user account and can include special offers from campaigns. (8.)

Get price list is one of the API endpoints requirements for M3, and it requires a customer number, a date, the online store, and a system to provide a list of all products and their prices. A webhook is sent whenever a price list is changed. (8.)

3.3 Multisite feature in WordPress

Multisite refers to a form of WordPress installation that enables the creation and administration of a network comprising multiple websites through a unified WordPress dashboard, simplifying the process of making changes and maintaining all websites from a single location. It offers the flexibility to establish a multisite network exclusively for personal use or involve other users who can generate their own sites within the network, with restricted access to advanced WordPress functionalities. WordPress multisite networks are highly beneficial for individuals or entities seeking centralized management of distinct websites. Multisite networks find application in various domains such as corporations, educational institutions, media outlets, eCommerce enterprises, and others. (9.)

Snellman intends to construct a webshop using a multisite WordPress setup, aiming to effortlessly onboard new webshops without significant technical investment. Snellman recommended to host a single WordPress multisite installation capable of accommodating numerous webshops. Consolidating multiple webshops within one system offers advantages such as reduced maintenance costs and the availability of new features across all hosted webshops. This simplifies quality control and minimizes the impact of disruptive changes when implementing updates.

4 TOOLS AND TECHNOLOGIES

Chapter three describes the tools and technologies and architecture employed in development processes of SnellmanPetFood webshop. The Norwegian webshop uses the similar tools and technologies. Following are the tools and technologies used in the webshop.

PHP

PHP is a widely used general-purpose scripting language that is particularly well suited to web development and can be embedded to HTML. PHP powers everything from blogs to the most well-known websites in the world. PHP operates on the web hosting server because it is a server-side language. When someone accesses website, their browser makes a request for the page by contacting the server that hosts it. A HTML page is produced by the PHP code running on the server and sent to the visitor. The visitor's browser then displays the HTML page. (10.)

An introductory example of PHP code is shown in figure 2. The figure shows that PHP code is included within specific start and end processing instructions, denoted by the symbols `<?php` and `?>`. PHP pages contain HTML with embedded code that outputs “Hi, I’m a PHP” in the above figure.

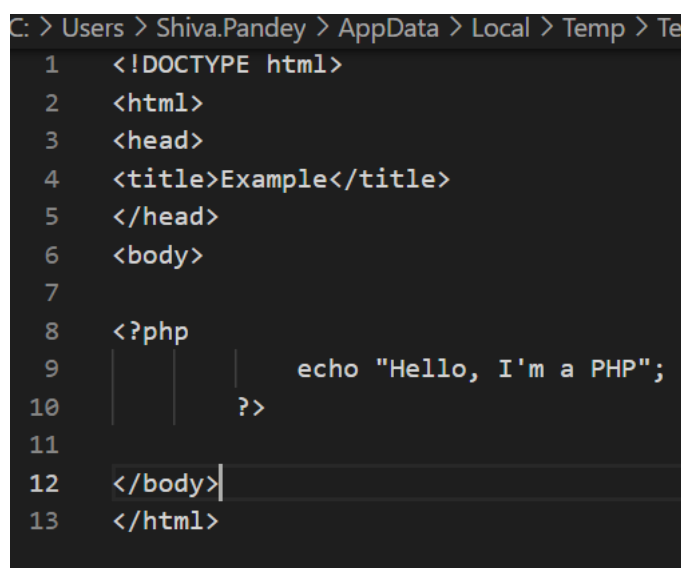
A screenshot of a code editor window with a dark background. The title bar shows the file path: "C: > Users > Shiva.Pandey > AppData > Local > Temp > Te". The code is displayed in a light-colored font. It starts with a line number 1 and the code `<!DOCTYPE html>`. Line 2 has `<html>`. Line 3 has `<head>`. Line 4 has `<title>Example</title>`. Line 5 has `</head>`. Line 6 has `<body>`. Line 7 is empty. Line 8 has `<?php`. Line 9 has `echo "Hello, I'm a PHP";`. Line 10 has `?>`. Line 11 is empty. Line 12 has `</body>`. Line 13 has `</html>`. The cursor is positioned at the end of line 12.

FIGURE 2. PHP code embedded with HTML code (10)

Composer

All the dependencies required for the project's operation can be declared, managed, and installed with Composer's assistance. Composer is used to increase the stability, security, and upkeep-friendliness of the WordPress website. With the help of composer, there is no need to commit WordPress core along with its themes and plugins to Git repository. (11.)

Employing Composer for WordPress site management provides distinct benefits. It facilitates precise dependency specification within a committed file (`composer.lock`), seamlessly integrated into the project's commit history. The lock file is created based on a comprehensive inventory of dependency restrictions specified in the `composer.json` file. Consequently, each subsequent branch maintains an identical set of dependencies, ensuring uniformity across contributors and deployments. This robust process guarantees code uniformity for all stakeholders, irrespective of their involvement or the specific deployment environment. (12.)

DDEV

Local PHP development environments can be quickly launched using the open-source application DDEV. A developer can benefit from a Docker process even without Docker experience or custom configuration because these environments can be expanded, version managed, and shared. Just as simply as they can be begun, projects can be altered, shut down, or abandoned. (13.)

DDEV-Local is an innovative tool designed for local web development, offering a resilient and adaptable setup for platforms like Drupal, WordPress, and TYPO3. By utilizing Docker, a containerization tool, DDEV-Local enables developers to encapsulate services within containers. Docker's strength lies in its hardware-agnostic nature, allowing the execution of Docker containers on any compatible hardware, thus ensuring portability and compatibility across different environments. (14.)

WooCommerce

WooCommerce is an open-source, adaptable software program designed for WordPress-powered websites. It is commonly employed to build online eCommerce stores. Anyone may transform their standard website into a fully functional online store with all the required eCommerce capabilities with this WooCommerce solution. (15.)

Users are able to manage their online shops from creating product displays to processing orders and accepting different payment sources by using WooCommerce software solution. Users may quickly obtain the WooCommerce plugin from the official WordPress plugin directory. WooCommerce also has a user-friendly user-interface (UI). (15.)

Acorn

Using the Acorn framework, Laravel can be integrated with WordPress. Acorn oversees compatibility with various elements of the Laravel ecosystem and incorporates Laravel features into WordPress. It also makes use of WordPress' Laravel blade. Acorn allows for the smooth loading of a Laravel application container inside of WordPress while maintaining WordPress's lifecycle and template hierarchy. Acorn comes with WP-CLI commands that give WordPress a dependable command line interface akin to Artisan-commands such as create components, service providers and clear the view cache. (16.)

Sage

Block editors are supported by Sage, an innovative starting theme for WordPress. To utilize Laravel's features and packages Acorn can be used. Theme templating is made possible using Laravel Blade. Workflow for modern front-end development driven by Bud. Sage's Tailwind CSS configuration automatically generates a theme.json which helps to configure the WordPress editor. The editor gives the possibility to use the defined fonts, colours, and spacing, and use Tailwind classes in the templates to quickly develop WordPress theme. (17.)

API endpoints

An API endpoint is the location where an Application Programming Interface. (API) connects to a software program, which is the code that enables two software programs to communicate with one another. Sending and receiving information requests from a web application or web server is how APIs operate. (18.)

Systems that exchange data through APIs are referred to as integrated systems. One side, referred to as the server, sends the data to the API. Requests are made and API operations are performed by the client, the opposing party. The API endpoint is the server-side component that delivers the resources or data that have been requested. Endpoints define the locations where APIs can access resources and aid in ensuring the integrated software is operating efficiently. The effectiveness of an API depends on its ability to exchange data with API endpoints. (18.)

The client must supply a uniform resource locator (URL), a method, a set of headers, and a body for the endpoint to perform the request effectively. The body contains for a request includes the data that the client sends to the server, while its headers offer information about the request. Together with API methods, endpoints function well. The use of methods like GET, DELETE, PATCH, and POST is permitted for requests. (18.)

Boomi

Boomi, also referred to as Boomi Integration Platform as a Service (iPaaS), is a cloud-centric integration platform developed by Dell Boomi. Its primary objective is to facilitate seamless connections among diverse applications, systems, and data sources for organizations. Boomi provides users with an intuitive graphical interface and a comprehensive selection of pre-designed connectors, empowering them to efficiently design, deploy, and manage integrations. It offers support for various deployment scenarios, encompassing both cloud-to-cloud and on-premises-to-cloud integrations. (19.)

Boomi plays a vital role in optimizing data flows, automating processes, and enabling real-time data synchronization throughout an organization's IT landscape. Notably, it encompasses essential functionalities such as data mapping, transformation, and workflow

orchestration. These features significantly contribute to enhancing connectivity and interoperability between different systems and applications within an organization. By leveraging Boomi, businesses can streamline their operations, achieve greater efficiency, and ensure seamless data exchange across their integrated ecosystem. (19.)

Infor M3

Enterprise resource planning (ERP) systems are comprehensive, integrated platforms that may be installed on-premises or in the cloud and are used to manage every aspect of a production- or distribution-based firm. ERP systems additionally support core accounting function along with all facets of financial administration, human resources, supply chain management, and manufacturing. ERP systems will also increase transparency across the whole business process by monitoring every area of manufacturing, logistics, and finances. When it comes to end-to-end workflow and data, these integrated systems serve as a business's primary hub and provide access to numerous divisions. (20.)

For medium-to-large domestic and international manufacturers, distributors, and after-sales service providers, Infor M3 is a potent Enterprise Resource Planning (ERP) software solution that can provide the groundwork for the digital transformation of the company. Infor M3 is an ERP system platform that supports multiple companies, multiple countries, and multiple sites, and it makes use of the most recent technology to deliver an amazing user experience and strong analytics. (21.)

Product lifecycle management software

Product lifecycle management (PLM) is the process of managing a product as it goes through the typical phases of its life cycle which are development and introduction, growth, stability, and decline. This management includes both the production and marketing of the good. Business decisions, from pricing and advertising to expansion or cost-cutting, can be informed by the idea of the product life cycle. (22.)

PLM software is a solution that controls all the data and procedures throughout the whole lifespan of a good or service in a network of global supply chains. Data from items,

components, products, documentation, specifications, engineering change orders, and quality procedures are included in PLM software. (23.)

Polylang

Polylang is a WordPress plugin that simplifies the process of adding multiple languages to a website and managing translations. It supports both manual and automatic translations and incorporates a translation management system to facilitate collaboration with professional translators. With Polylang, websites can accommodate an unlimited number of languages, making it a versatile solution for creating multilingual sites. While the plugin offers basic features for free, it also provides premium options. However, it is worth noting that the extensive functionality of Polylang may introduce some complexity when using it. Nevertheless, Polylang is a comprehensive tool that effectively facilitates translations, empowering websites to become dynamic and multilingual. (24.)

Understanding the complexities of Polylang can be difficult due to its inherent intricacy. Nevertheless, there are significant practical alternatives available that offer simplified methods for translating WordPress websites and meeting the requirements of International Search Engine Optimization (SEO). Well-known options like WPML, Weglot, Conveythis, TranslatePress, Bablic, and Gtranslate provide user-friendly interfaces and intuitive features that streamline the translation process. These solutions facilitate seamless management of translation workflows, allowing websites to effectively engage with a global audience while enhancing their search engine optimization endeavors. (25.)

As Snellman has already utilized the Polylang plugin for language translation within their multisite WordPress environment, they strongly recommend its continued usage for translating the site. Given their previous positive experience, Snellman recognizes the effectiveness and suitability of Polylang for achieving language translation in the multisite setup. Therefore, they advise leveraging Polylang as the preferred solution to ensure seamless and efficient translation capabilities across their multisite WordPress network.

5 EXISTING SYSTEM STRUCTURE

SnellmanPetFood is a Snellman's eCommerce website designed to sell cat and dog foods. SnellmanPetFood is a B2B webshop for MUSH Oy which is a part of Snellman Group. The webshop platform is built using WordPress and WooCommerce. Modern WordPress named Bedrock is used in building the webshop platform. Bedrock is used as a boilerplate to create modern WordPress sites. This chapter explains the WordPress ecosystem of the SnellmanPetFood webshop, user flow diagrams and uses of necessary third-party plugins.

5.1 System structure

This chapter demonstrates on how the WordPress ecosystem of SnellmanPetFood (MUSH) works to fulfil the requirements of technical goals of Snellman's ecommerce website. The system structure of the webshop is illustrated in figure 3.

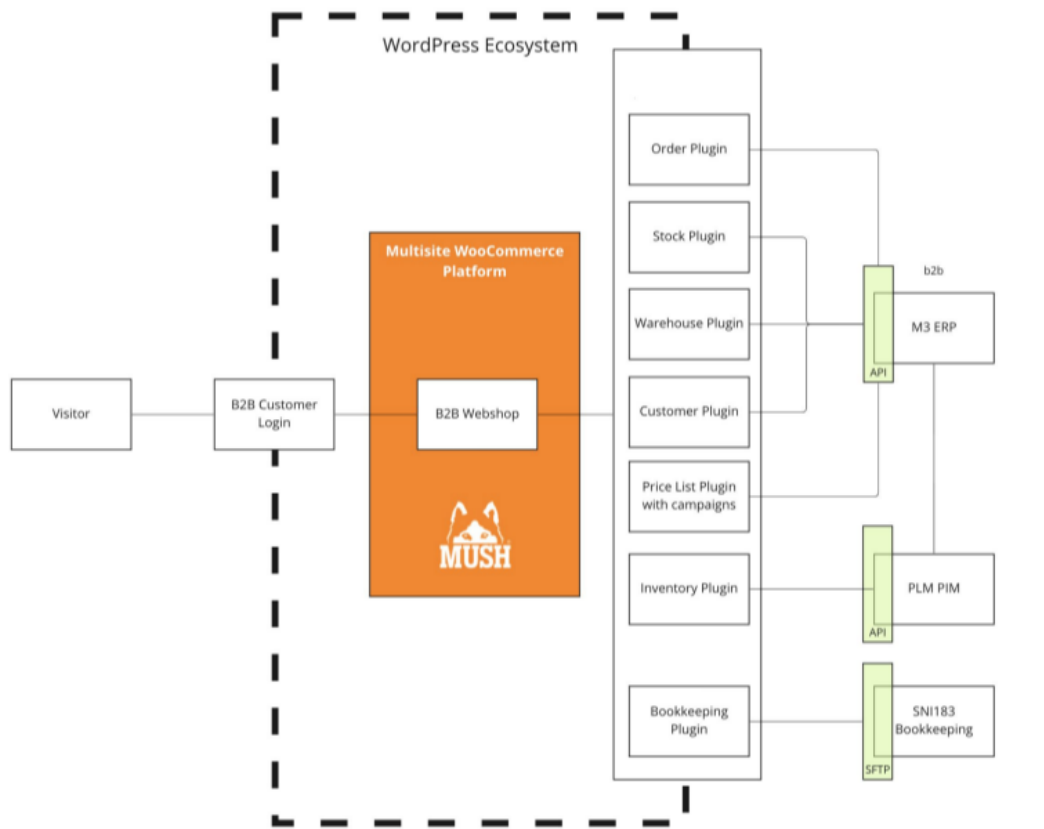


FIGURE 3. System data workflow of SnellmanPetFood (MUSH) B2B webshop (26)

Figure 3 illustrates the data flow in the SnellmanPetFood webshop. A customer login into the B2B webshop with login username with email address and password. When a person is logged in, they will be able to see the products according to their customer number. Assortment was made with the warehouse and price list ID. Then they can choose the product and add to the cart and purchase items. (26.)

The data are fetched via plugins such as product information synchronized between Infor M3 and WooCommerce using Boomi's prebuilt connectors which includes product details such as assortment, product ID, name, description, and price. M3 sends data of customer, order stock, routes, campaigns, and price list through Boomi to the webshop. M3 receives orders from WooCommerce. M3 sends data to the Boomi in JSON file format and then to the webshop. B2B webshop integrates with the data of Infor M3 externally. Boomi fetches data from M3 and formatting it to JSON file format and sends it to WooCommerce. Boomi receives orders from WooCommerce. (26.)

There exists external Boomi and internal Boomi. An external Boomi receives and sends data between Infor M3 and WooCommerce while internal Boomi maps data and connect to Infor M3 API endpoints. External and internal Boomi works independently. A message queue which is called SNMQ is responsible for moving data between external to internal Boomi and vice-versa. Infor M3 has an internal network, which is protected from outside with firewalls and Boomi itself. (26.)

The product code, name, details, description, ingredients of the products and images are fetched to WooCommerce from PLM through inventory plugin. Inventory plugin sends and receives data from WooCommerce to PLM and vice versa. Moreover, PLM and M3 are integrated to ensure an efficient flow of the product life cycle. (26.)

Bookkeeping plugins are used in the B2C websites. These plugins are disabled in B2B webshop since bookkeeping in B2B webshop are done internally with M3 orders. (26.)

5.2 User flow diagram of the webshop

A user flow is a chart or diagram that depicts the steps a user must follow to perform a task in an application. Product teams create user flows to facilitate intuitive product design, give users the relevant information at the right time, and enable them to do desired activities quickly. (27.)

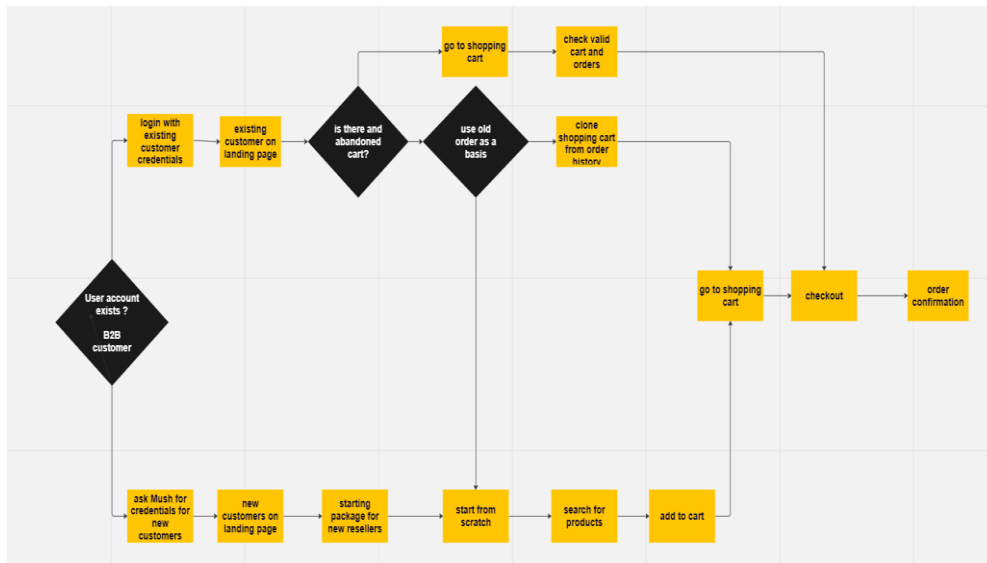


FIGURE 4. User flow diagram of SnellmanPetFood webshop (21)

The above figure 4 illustrates the user flow diagram of SnellmanPetFood webshop. At the starting of the website, B2B customers should login into the webshop with their credentials using email address as username and the password associated with it. A new customer must request the login credentials from MUSH. When a customer signed in into the webshop, they will land into the page that shows the MyAccount page which includes dashboard, orders, addresses, account information, and sign out link. A customer can go to the products page by clicking shop item menu located in the navigation menu bar. (28.)

Each customer in Infor M3 has a limited assortment of the products because assortment of products relies on what warehouses a customer can see. For example, a customer from Finland single warehouse which is Finland's warehouse. A customer from Finland has no options to choose a warehouse. Likewise, a customer from Sweden would have three warehouses to choose products which are Finland's lager, Jordbro fryslager and Nässjö torr lager. If the products are ordered from two different warehouses, there would be two

different orders. Depending on the customer specified in M3, the webshop allows customer to buy only certain products and certain quantities of products. (28.)

When a customer lands to the shopping page, they can choose the products according to which warehouse they are associated with. An existing customer can check if there are any products that were added previously to the shopping cart. If the customer would like to buy the products that were previously added to the cart, they must check the valid date and delivery time and can proceed to the checkout. An existing customer can also clone the orders to the shopping cart from his/her order history and go to the checkout. (28.)

In case of a new customer or an existing customer who would like to choose a product from scratch they can search for all products or make a quick order. A quick order can be made by viewing the entire product catalogue that is related to specified customer and filter by favourite products. A customer can search for all products and view the specific products and learn about the products. There are also articles, news, and videos available with embedded products. (28.)

After choosing and adding the products to the cart with the quantity. The customer would be notified whether the product is out of stock while they add the products to the shopping cart with its quantity. A customer can postpone orders in case of out-of-stock orders. (28.)

When the customer can add the products to the shopping cart, they can view the shopping cart and its delivery time and then proceed to checkout page. The shipping method and payment method are internally specified in Infor M3. Based on the total amount of products a consumer purchases and the warehouse they select, shipping cost will be free. The last part when checkout of the product is done, the customer would see the order confirmation with its delivery date and time. (28.)

6 IMPLEMENTATION OF A NORWAY WEBSHOP

The implementation of a Norwegian webshop involved the utilization of two distinct plans concurrently. Initially, the intention was to develop an entirely new website within a multisite WordPress setup. However, a change in strategy occurred as the company sought to incorporate the Norwegian language into an existing site, thereby creating a dedicated Norway webshop. Consequently, the original implementation plan was interrupted, and a new target was formulated in alignment with the specific needs and requirements of the company. The subsequent implementation efforts were directed towards fulfilling the revised objectives and ensuring compatibility with the company's desired outcome.

6.1 Implementation with prior goals and objectives

Firstly, Snellman planned to create a new webshop instance in a multisite setup. A Norway webshop would have a new site URL and subdomain-based structure. However, the implementation was interrupted and was not completed because of a new implementation plan. The stages of the implementation of a Norway site in multisite WordPress with prior objectives follow the following steps.

6.1.1 Planning

Before the implementation of a Norway webshop, the process involves the planning of the webshop. The planning stage is focused to know the targeted customers, products to sell, themes and template designs for the webshop, assortments of products and associated warehouse and shipping methods.

Norway webshop is targeted to the Norway B2B customers. The design and layout of the page for the Norway webshop is planned to use the same templates and designs layout similar to the SnellmanPetFood webshop. Considering the assortment of products, two warehouses were planned to build for the Norwegian retailers. The shipping settings are

planned to differ from existing webshop. The shipping logic in Norway webshop would be done according to the shipping address where postal code range will determine the price for shipping.

The prerequisite for the development of Norway webshop is to run the Snellman's eCommerce application locally and enable multisite feature. All websites on a WordPress multisite network share the same server resources. This means that the most important part is to have a good WordPress hosting. JNT server is used for the production environment. However, local server is used for the development environment. A Norway webshop uses same tools and technology that was used in building the SnellmanPetFood webshop.

6.1.2 Creating a new webshop instance in multisite WordPress

To create a new site in a multisite WordPress, firstly, it was ensured that the WordPress installation is set up as a multisite network and a user has a super admin account.

One must login to the WordPress admin dashboard with a supper admin account to create a new site. A new site is added in the network admin dashboard and configured site settings. Figure 5 shows the task completed to create a new site in multisite WordPress.

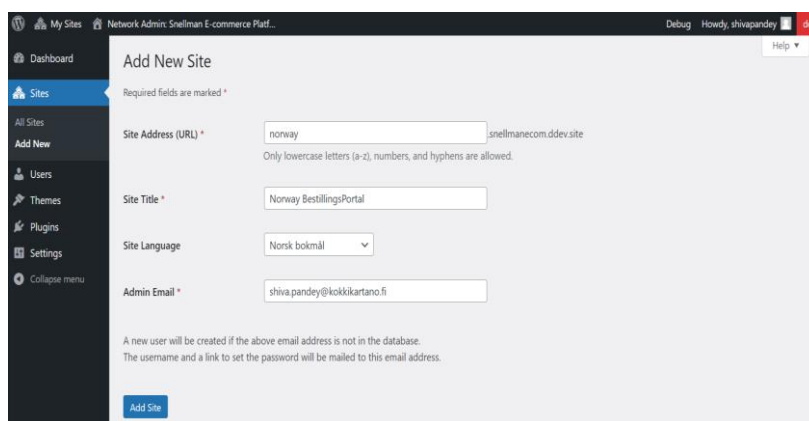


FIGURE 5. Screenshot taken from the development site of Snellman eCommerce platform

At first, the site address URL or subdomain for the new site is enabled. An admin can choose between a subdomain based-structure, for instance, site1.example.com or a sub-directory-based structure, for instance, example.com/site1. In my case, subdomain-based

structure was used for creating a new site. The site address URL was `norway.snellmanecom.ddev.site`. Similarly, the site title was edited for the new site. The site title will be displayed as the site's name throughout the network and on the site itself. The site title was Norway Bestillingsportal. There is also a need to specify admin email that would be associated with the site and used for administrative notifications. Users were managed by adding roles such as administrator, subscribers, and authors. Norwegian language was selected for the preferred language for the new site. After that, a new site was created in the multisite setup.

6.1.3 Installing and customizing plugins

After creating a new site, the site was installed and activated with necessary plugins. From the site dashboard, admin can manage the site's content, appearance, and settings. The plugins were customized to fit for the Norway webshop. The plugins installed on a multisite WordPress network can be network-enabled such that all the plugins are available on all sites within the network. Therefore, the plugins in Norway webshop were chosen from the multisite network-enabled plugins. After that, WooCommerce plugins such as zone, currency, and shipping logic were customized.

Figure 6 shows the incomplete lists of plugins that are available in the multisite network-enabled plugins. The plugins are selected from the lists available in the figure below and activated to the new site according to the preferences.

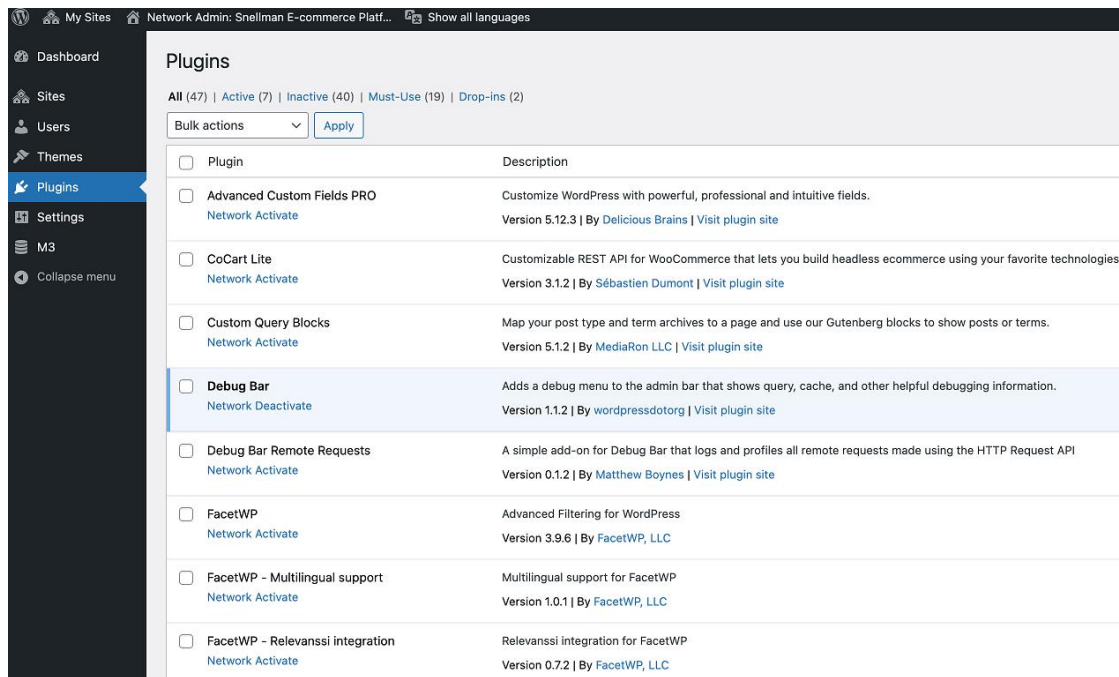


FIGURE 6. Screenshot taken from the Snellman eCommerce platform

6.1.4 Themes and layout of the webshop page

The site's content, appearance and designed was then implemented in the webshop. It was planned that a Norway webshop will be implemented using same template and design pattern. The appearance of the site should look like the existing webshop. New templates are created according to the existing webshop. The template parts are translated into Norwegian language.

To use the same theme and layout designs, a super admin should install and activate the theme to use across multiple sites. By default, themes in multisite WordPress are enabled for all sites. Since it was planned to use the same theme for Norway, the theme was already in use by default.

After activating the theme, webshop pages are then created. The title of each page is composed and configured with design contents. By-default, WordPress uses the Gutenberg editor that provides a block-based approach to content creation. Different types of blocks such as text, images, headings, and footer were added to the site. The layout, styling and content of each block were customized individually.

There are five different pages added to the Norway site. Home page, shop page, shopping cart page, News feed page and checkout page were created. However, the pages were not implemented with the assortments of product and users to the site because of the changes in the project plans. PLM product translation and API endpoint were not released. Price-lists, assortment, and stock were not created in M3 system. Shipping fee logic was not yet decided. To show the products available in the page, the Norway webshop would need PLM, Infor M3 and shipping logic settings defined. Therefore, all the pages are not displayed in this thesis.

Figure 7 shows the home page. Only home page is visible because other pages do not include any data. A home page of a Norway webshop typically serves as the main entry point for B2B customers and showcases essential elements and information related to the site. The home page includes logo and branding image, navigation menu such as Home, Shop, News feed, login and set language to the site.

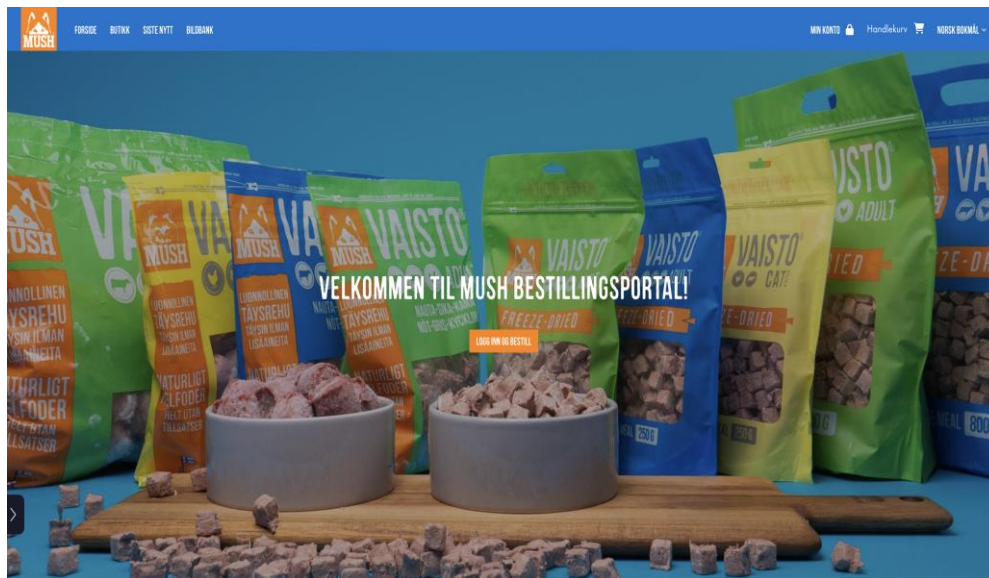


FIGURE 7. Screenshot taken from the development site of Norway webshop

6.2 New target for the implementation of a Norway webshop

The primary implementation plan was interrupted, and a new plan was proposed for the implementation of a Norway webshop. The task requirements for new target are adding a Norwegian language to the existing webshop, configure PLM settings and M3 and add shipping logic. The implementation follows the following steps.

6.2.1 Adding Norwegian language to the existing site

Existing multisite webshop has already an advanced translation feature. A popular translation plugin ‘Polylang’ was installed and activated for the multisite setup. The translation plugin was then configured. The configuration for the translation plugin involves specification of default language, addition of a Norway language as well as configuring language switcher options.

All pages were translated, header and footer were customized using fullsite editing. Templates could not be translated by using Polylang. So, the use of different sections in the templates were translated manually, such as header and footer sections. Image was added to the front page. Pages use same templates defined by default in the multisite. For example, account page uses account page templates, checkout page uses a customized page template named thank you page.

6.2.2 Configure PLM

Experts in PLM updates beCPG PLM attributes and translations which includes product names, descriptions, categories, customer fields or any other data that needs to be localized. beCPG is an open-source software configurable and customizable to business and industry. beCPG stands for be ready to use Consumer Packaged goods (CPG) industries such as food, beverages, and cosmetics. In case, the products details are not translated to Norwegian, by default Swedish language would be displayed because Swedish language seems to be quite similar to Norwegian language.

To synchronize the WooCommerce and beCPG PLM, a plugin named wp-plm plugin is customized and built. A beCPG PLM plugin integrates with WordPress using the WordPress Command-Line Interface (WP-CLI). Figure 8 demonstrates the environment variables related to the configuration of the beCPG PLM connector in WordPress environment file. The environment variables are defined in the .env root file.

Figure 10 shows the ‘getTranslated()’ method which takes three parameters, ‘\$object’, ‘\$key’, and ‘\$language’. The tasks for the implementation were to add a switch statement for Norwegian language. Inside the ‘switch’ statement, the behavior depends on the value of the ‘\$language’ parameter. If the ‘\$language’ is Norwegian ‘no’, an array of unique keys is created using the language suffixes defined in the ‘\$languageSuffixes’ array along with the provided ‘\$key’. For any other value of ‘\$language’, including the default case, the same array of unique keys is created without including the Norwegian language suffix.

```
protected static function getTranslated($object, string $key, string $language)
{
    if (!$object) {
        return null;
    }

    switch ($language) {
        case 'no':
            $keys = array_unique([
                $key . '_' . self::$languageSuffixes[$language],
                $key . '_' . self::$languageSuffixes['sv'],
                $key . '_' . self::$languageSuffixes['fi'],
                $key . '_' . self::$languageSuffixes['en'],
                $key,
            ]);
            break;
        default:
            $keys = array_unique([
                $key . '_' . self::$languageSuffixes[$language],
                $key . '_' . self::$languageSuffixes['fi'],
                $key . '_' . self::$languageSuffixes['en'],
                $key,
            ]);
    }
}
```

FIGURE 10. Code snippets showing switch statements for Norwegian language

6.2.3 Configure M3

All the price lists, route, assortments, customer lists, shipping data (route departures, delivery date) are created in Infor M3 for Norwegian language. Since all the M3 plugins of an existing site already support for newly created fields in M3, there was no need to make any changes to these plugins. All the inputs for the pricelists, route, assortments, and customer lists are handled by M3 experts. The shipping fee logic was created in wp-Snellman-M3-stock customized plugin. The shipping fee logic added to the system is described as follows.

The shipping fee logic in Norway site would be based on zip code ranges regarding a shipping destination of a customer and the price of the order a customer has purchased. Table 1 below demonstrates the logic applied for the cost of the shipping fee.

TABLE 1 Shipping terms (Data taken from Meetings12.05.2023)

Postal code from	Post code to	Free shipping for price of the order (Nkr)
0	2100	3500
2100	4000	4500
4000	6000	5500
6000	8000	7000
8000	9050	8000
9050	over	By request

Table 1 defines the shipping fee logic to be applied to the Norway B2B customers. The postal code shown in the table are the shipping location of the customer. The data in the first row of a table demonstrates, if a customer belongs to the postal code range between 0 and 2100 and the purchase of a product equals to 3500Nkr, then there would be a free shipping for that customer. In case, a customer buys the product that amounts to below 3500, there would be a shipping cost applied. Similar, table shows criteria for free shipping for other customer who belongs to postal code ranges 2100-4000, 4000-6000, 6000-8000, 8000-9050, and 9050 and over. A customer who belongs to the postal code between 9050 and over gets free shipping upon by request.

The shipping fee logic was added to WooCommerce shipping methods in wp-snellman-m3-stock plugin. Figure 11 demonstrates the code that defines a class named 'WooCommerceShippingMethods'. Within this class, several constants are defined using the 'public const' syntax. Constants are class-level variables whose values cannot be changed once defined. These constants are also applied for sites with Finnish and Swedish languages. From the WooCommerce site, admin can choose what to include for shipping fee logic. The last line of the code snippets, 'postal const POSTAL_CODE_RANGE = 'postal_code_range';' is added for the Norway site. Norway site also uses constant-level variables which are, WAREHOUSE_ID, MINIMUM_COST, ITEM_NUMBER and PALLET_NUMBER.

```

class WooCommerceShippingMethods
{
    public const WAREHOUSE_ID = 'warehouse';
    public const APPLY_MINIMUM_WEIGHT = 'apply_minimum_weight';
    public const MINIMUM_COST = 'minimum_cost';
    public const MINIMUM_WEIGHT = 'minimum_weight';
    public const MINIMUM_QUANTITY = 'minimum_quantity';
    public const LIMITED_PRODUCT_IDS = 'limited_product_ids';
    public const LIMITED_CUSTOMER_IDS = 'limited_customer_ids';
    public const LIMITED_CUSTOMER_CHAINS = 'limited_customer_chains';
    public const EXCLUDING_CUSTOMER_IDS = 'excluding_customer_ids';
    public const ITEM_NUMBER = 'item_number';
    public const PALLET_NUMBER = 'pallet_number';
    public const PALLET_COST = 'pallet_cost';
    public const POSTAL_CODE_RANGE = 'postal_code_range';
}

```

FIGURE 11. Code snippets defining postal code range to WooCommerce shipping methods

Figure 12 demonstrates the code snippets for ‘filterPackageRates’ method within a class. This method in the figure is used to filter an array of shipping rates based on certain criteria specifically postal code ranges. The method ‘filterPackageRates’ involves two parameters: ‘\$rates’ and ‘\$package’. \$package is an array representing a shipping package which declares to return an array ‘\$array’ after filtering the rates.

```

public function filterPackageRates(array $rates, array $package): array
{
    foreach ($rates as $id => $rate) {
        $settings = get_option(sprintf('woocommerce_%s_settings', str_replace(':', '_', $rate->get_id())));
        if (!$settings) {
            continue;
        }
        $postalCodeRanges = array_map('trim', explode(',', $settings[self::POSTAL_CODE_RANGE] ?? ''));
        $postalCodeRanges = array_filter($postalCodeRanges);
        if ($postalCodeRanges) {
            $postalCode = (int) Customer::getCurrent()->get_shipping_postcode();
            $isWithinRange = false;
            foreach ($postalCodeRanges as $postalCodeRange) {
                [$from, $to] = explode('-', $postalCodeRange);
                $from = (int) $from;
                $to = (int) $to;

                if ($postalCode >= $from && $postalCode <= $to) {
                    $isWithinRange = true;
                }
            }
            if (!$isWithinRange) {
                $this->debug(
                    'Removed %s because current postal code %s isn\'t within range: %s',
                    $rates[$id]->get_id(),
                    $postalCode,
                    implode(', ', $postalCodeRanges),
                );
                unset($rates[$id]);
            }
        }
    }
}

```

FIGURE 12. Code snippets for ‘filterPackageRates’ method based on postal code range

The method loops through each shipping rate in the `$rates` array and retrieves the settings for each rate using the `'get_option'` function. The `'get_option'` function fetches the option value from the WordPress database based on the option name constructed using `'sprintf'`. If the retrieved settings are empty or false (indicating no settings found), the loop moves to the next rate using the `continue` statement.

The method then enters a nested loop, iterating over each postal code range in the `'$postalCodeRanges'` array using a `'foreach'` loop. Within this loop, each range is processed individually. For each range, the lower and upper bounds are extracted by splitting the range string using `'explode'` with the hyphen `'-'` as the delimiter. The lower and upper bounds are then cast to integers using `'int'` and assigned to the variables `'$from'` and `'$to'` respectively. The method checks if the current postal code falls within the current range by comparing the lower and upper bounds using `'>='` and `'<='` operators. If the postal code is within the range, `'isWithinRange'` flag is set to `'true'`.

Once all rates have been processed, the method completes execution and returns the filtered `'$rates'` array. The `'filterPackageRates'` method filters shipping rates based on postal code ranges. It retrieves the necessary settings and checks if the postal code falls within any specified range and remove rates.

Figure 13 demonstrates the code snippet that updates the method description of a shipping method object based on the postal code ranges specified in the instance settings. the code retrieves the postal code ranges from the instance settings of a shipping method and appends them to the method description in HTML format if the ranges are available. This can be used to display the postal code ranges associated with a particular shipping method in a user-friendly manner.

```
$postalCodeRanges = $method->instance_settings[self::POSTAL_CODE_RANGE] ?? null;

    if ($postalCodeRanges)
    {
        $method->method_description .= sprintf('<p><em>Postal Codes: %s</em></p>', $postalCodeRanges);
    }
```

FIGURE 13. Code snippets for shipping method object

Figure 14 demonstrates the code snippet that represents the settings configuration for the “Postal code range” field in a shipping method. It is typically used in the context of a settings page or form for configuring shipping options in a WooCommerce.

```
$settings[self::POSTAL_CODE_RANGE] = [
    'title' => __('Postal code range'),
    'type' => 'text',
    'placeholder' => __('Eg. 2100-4000, 6000-8000'),
    'description' => 'A comma separated list of postal code ranges using a dash (-) for from-to.',
    'desc_tip' => true,
    'default' => '',
];

return $settings;
```

FIGURE 14. Code snippets for settings shipping configuration in WooCommerce

6.3 Results

All the implemented tasks were sent to the development team. The development team has progressed to test the materials. PLM and Infor M3 experts are now handling the Norway webshop. The test cases for the Norway webshop are underway. The following errors occurred after adding new language in webshop as shown in figure 15. A cache issue was appeared on a newly set up language. The above error was solved by running a command line: `ddev wp@ddev.mushb2b cache flush`.

TOTAL WARNINGS:	TOTAL NOTICES:
1	0
<p>WARNING: /var/www/html/web/app/plugins/polylang-pro/vendor/wpsyntax/polylang/frontend/frontend-filters.php:78 - Undefined array key 782</p> <p>require('wp-blog-header.php'), require_once('wp-includes/template-loader.php'), include('wp-includes/template-canvas.php'), get_the_block_template_html, do_blocks, render_block, WP_Block->render, render_block_core_template_part, WP_Query->__construct, WP_Query->query, WP_Query->get_posts, get_option, apply_filters('option_sticky_posts'), WP_Hook->apply_filters, PLL_Frontend_Filters->option_sticky_posts</p>	

FIGURE 15. Webshop error while adding new language to the site

The PLM configuration for the WooCommerce is not synchronized yet because the API responses from the PLM website is not responding. The PLM call that was made is not responding as it should be. However, if there is no translation of the products in Norwegian language, the site would use Swedish language by default. Therefore, the result pages for the Norway site still have the Swedish language translation for the products. The

following paragraphs provide a description of the outcome observed in the result pages after the implementation of a Norwegian language site up to this point.

Home page

Figure 16 shows the home page of Norwegian site in SnellmanPetFood webshop. The home page consists of the logo of the company, navigation menu, cover image and the title of the webshop. The footer section of the home page will consist of privacy and cookies settings which is not implemented yet. The main navigation bar includes the main categories such as FORSIDE translated as Front page, BUTIKK as shop page, SISTE NYTT as latest news and BILDBANKK as image bank.

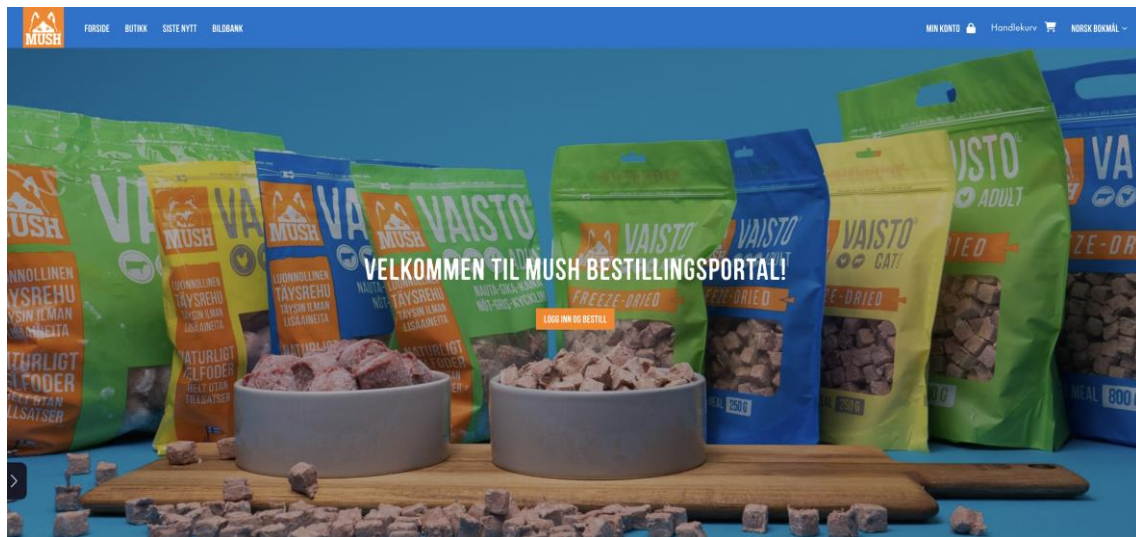


FIGURE 16. Screenshot of home page of Norwegian site

Product page

Figure 17 shows the products listed for the customer where a customer can search and choose the product. The customers can see assortment of products based on the warehouse and M3 customer ID. In the products page, a customer can also see the details of the products such as product's name, ingredients, and storage information.

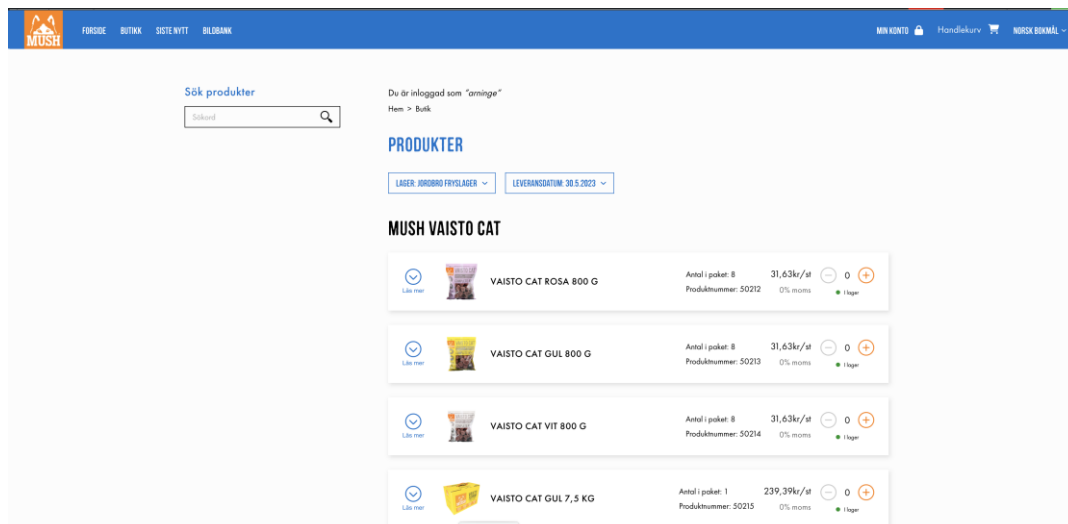


FIGURE 17. Screenshot of product page of Norway site

Checkout page

Figure 18 shows the checkout page of a Norwegian language site. The checkout page enables customers to review their order details and confirm their purchase. The shipping, billing and payment information are handled internally in Infor M3. Therefore, the shipping, billing and payment information are not shown in this figure. The order summary in checkout page shows a concise overview of the items in the customer's order, including product names, quantities, summary of all charges and discounts applied, shipping fees and applicable taxes.

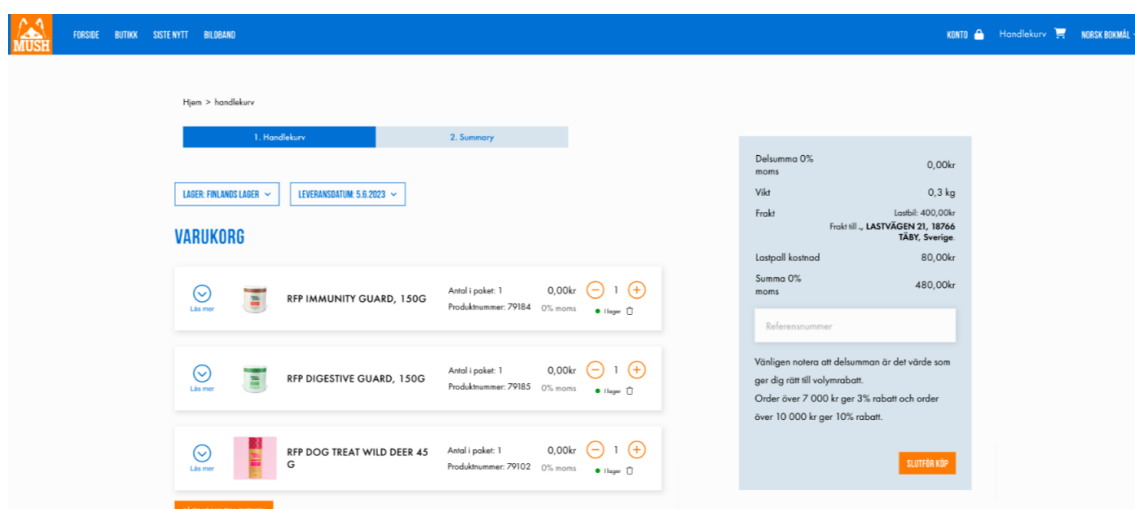


FIGURE 18. Screenshot of checkout page of a Norwegian site

The ongoing development of the Norway webshop has achieved progress in terms of product assortments and prices, which have been completed in Infor M3. However, the implementation of the Norwegian language site on the production site is still pending. The webshop is currently facing issues with API responses from the M3 website, and the setup for route handling in M3 has not been implemented. Additionally, the shipping fee logic for the WooCommerce site has not been finalized yet. These remaining tasks require attention and further development to enhance the functionality and completeness of the webshop.

7 ANALYSIS OF THE CHANGES MADE IN IMPLEMENTATION PLAN

The prior implementation plan of a Norway webshop was to create a new site named Norway webshop in a multisite setup. A Norway webshop was planned to have its own subdomain-based site URL. In addition, certain elements such as themes, plugins and global settings would be duplicated for a Norwegian site. The second implementation plan was to just add a Norwegian language to the existing site and configure language preference settings to the site. When deciding whether to add language to an existing site or create completely a new site for a new language using multisite WordPress, it is important to consider the advantages and implications of each approach. The analysis of both approaches is described in the following paragraphs.

The benefits of adding a language to an existing site, as opposed to creating a new site in a multisite WordPress installation, include simplicity, consolidation of content, and the ability to maintain consistent design and branding. Adding a language to an existing site is usually a simpler process since there is no need to create a separate site or multiple installations. It also consolidates all contents in a single location, simplifying management and updates. It also enables seamless language switching for users. Moreover, adding a language to the existing site allows to maintain consistence in the design and brand along with user experience.

The disadvantages of adding a language to an existing compared to a new site creation include performance impact and plugin dependencies. Adding a language using a translation plugin may introduce additional dependencies on third-party plugins. If the plugin becomes unsupported or incompatible with future WordPress updates, it could potentially impact the functionality of the site. Moreover, depending on the translation plugin and the volume of content, adding translations to an existing site may affect the performance of the site. Location based plugins must be enabled to target a specific customer for example, a customer in Norway should see the site in Norwegian language by default.

Unlikely, creating a new site in a multisite setup, as opposed to adding a new language to an existing site, offers advantages such as having a separate site structure, increased flexibility, performance, and improved scalability. Creating a new site for a new language using multisite WordPress allows to have a separate site structure for each language which could be beneficial if the content, layout, or functionality of the new language version significantly differ from the original site. Multisite WordPress gives more control over each language site and possibility to install different themes, plugins and customize settings specific to each language version. In addition, creating a new site in multisite installation provides a scalable solution which makes easier to manage and maintain the site whenever company wants to expand and add more language to the site.

The disadvantage of creating a new site includes increased complexity, and content duplication. Setting up and managing a multisite WordPress installation can be more complex compared to adding a language to an existing site because it requires additional configuration and maintenance efforts. Moreover, with a multisite setup, there is a need to duplicate certain elements such as themes, plugins, and global settings for each language site. This can result in increased disk space and more maintenance work.

Considering the advantages and disadvantages of both approaches, Snellman decided to add a new language to the existing site rather than creating a new site in a multisite WordPress setup. This decision was made to minimize implementation time, reduce cost, reduce maintenance efforts, and save disk space. Since the Norwegian webshop sells similar products to the existing webshop, the company deemed it appropriate to add the Norwegian language to the site.

8 CONCLUSIONS

The subject matter of this thesis is notable and deserves exploration since WordPress and WooCommerce provide developers with a robust platform for creating a wide array of exceptional web applications. The findings and insights from this thesis project can serve as valuable experience and a point of reference for future development assignments.

Through the implementation process, significant key insights and outcomes have been realized. Firstly, the flexibility and extensibility of WordPress and WooCommerce have allowed for the customization and adaptation of the webshop to meet specific business requirements. The availability of a wide range of themes, plugins, and extensions has facilitated the incorporation of diverse features such as product catalogs and inventory management systems. This thesis also displays the utilization of WordPress and WooCommerce in incorporating a new language into an existing website, exemplifying the proficiency and potential of these platforms in the creation of eCommerce solutions.

The utilization of Multisite WordPress and WooCommerce for creating a Norway webshop opens up possibilities for further development and application in different contexts. The experience gained from implementing this solution can be leveraged to streamline future webshop projects, both within the company and beyond. The flexibility of multisite WordPress allows for the easy onboarding of new webshops without significant technical investment. This capability presents a valuable opportunity for the company to expand its online presence and cater to a wider range of customers.

To sum up, the author has acquired knowledge in eCommerce solutions using WordPress and WooCommerce. Through collaboration with colleagues, the author has gained expertise in new frameworks, technologies, and effective communication strategies. Meetings and follow-ups have further enhanced the author's understanding and skills in the field.

REFERENCES

1. Snellman Konserni - Kosernen 2023. History. Date of retrieval 04.04.2023.
<https://www.snellmangroup.fi/sv/snellman-koncernen/tarina/>.
2. Snellman Group 2023. Snellman Group. Date of retrieval 04.03.2023
<https://www.snellmangroup.fi/en/snellman-koncernen/>.
3. ithemes 2023. What is WordPress? Date of retrieval 09.04.2023
<https://ithemes.com/tutorials/what-is-wordpress/>.
4. Instawp 2022. The best way to create a blueprint (Boilerplate) WordPress web-site. Date of retrieval 02.06.2023
<https://instawp.com/create-blueprint-boilerplate-wordpress-website/>.
5. Roots 2023. Bedrock. Date of retrieval 09.04.2023
<https://roots.io/bedrock/>.
6. Carlson, C. 2021. Bedrock for modern WordPress development. In: Platform.sh. Date of retrieval 09.04.2023
<https://platform.sh/blog/bedrock-modern-wordpress-development/>.
7. Fitzgerald, A. 2022. The ultimate guide to WordPress plugins: 19 examples and how they work. In: HubSpot. Date or retrieval 04.05.2023
<https://blog.hubspot.com/website/wordpress-plugins>.
8. Genero 2021. Snellman ECommerce technical requirements. Snellman project documentation. Internal source.
9. WPbeginner 2023. What is Multisite (MU)? Date of retrieval 04.06.2023
<https://www.wpbeginner.com/glossary/multisite/>.
10. The PHP Group 2023. What is PHP? Date of retrieval 03.04.2023
<https://www.php.net/manual/en/intro-what-is.php>.
11. Platform.sh 2023. Upgrade your WordPress site to use Composer. Date of retrieval 07.04.2023
<https://docs.platform.sh/guides/wordpress/composer/migrate.html>.
12. Platform.sh 2023. Why you should manage WordPress with Composer? Date of retrieval 04.06.2023
<https://docs.platform.sh/guides/wordpress/composer.html>.
13. DDEV Docs 2023. Get started with DDEV. Date of retrieval 07.04.2023
<https://ddev.readthedocs.io/en/stable/>.

- 14 OStraining 2023. Local Web Development with DDEV explained.
Date of retrieval 04.06.2023
<https://ostraining.com/books/local-2/>.
- 15 Safira, A.P. 2022. What is WooCommerce? A guide to WordPress ECommerce.
In: MultilingualPress. Date of retrieval 09.04.2023
<https://multilingualpress.org/what-is-woocommerce/>.
- 16 Roots 2023. Acorn. Date of retrieval 10.04.2023
<https://roots.io/acorn/>.
- 17 Roots 2023. Sage. Date of retrieval 10.09.2023
<https://roots.io/sage/>.
- 18 TechTarget Network 2021. API endpoint. Date of retrieval 10.04.2023
<https://www.techtarget.com/searchapparchitecture/definition/API-endpoint>.
- 19 TechTarget 2023. Boomi Atmosphere Platform (Dell Boomi Atmosphere). Date of retrieval 04.06.2023
<https://www.techtarget.com/searchcloudcomputing/definition/Dell-Boomi>.
- 20 Oracle 2023. What is ERP? Date of retrieval 11.09.2023
<https://www.oracle.com/erp/what-is-erp/>.
- 21 ERPFOCUS 2023. Infor M3. Date of retrieval 11.04.2023.
<https://www.erpfocus.com/infor-m3-erp-software-profile.html>.
- 22 Segal, T. 2023. Product Lifecycle Management (PLM): Definition, Benefits, History. In: Investopedia. Date of retrieval 11.04.2023
<https://www.investopedia.com/terms/p/product-life-cycle-management.asp>.
- 23 Oracle 2023. What is PLM (Product Lifecycle Management)?
Date of retrieval 11.04.2023
<https://www.oracle.com/scm/product-lifecycle-management/what-is-plm/>.
- 24 Themeisle 2022. How to create a multilingual WordPress Website using Ploylang. Date of retrieval 04.06.2023
<https://themeisle.com/blog/multilingual-wordpress-website-polylang/#gref>.
- 25 Redaction team 2022. 6 Polylang alternatives in 2023. Digital Marketing Blog.
Date of retrieval 04.06.2023
<https://barrazacarlos.com/polylang-alternatives/#>.
- 26 Genero 2021. Snellman's eCommerce: multi-brand eCommerce concept & project MUSH. Snellman project document. Internal source.
- 27 ProductPlan 2023. User Flow. Date of retrieval 15.04.2023
<https://www.productplan.com/glossary/user-flow/>.

- 28 Genero 2021. Snellman multi-brand eCommerce strategy. How to move forward with success. Snellman project documentation. Internal source.