

# Tūturu Dashboard

Utvecklingen av en plattform där vårdgivare och skolor möts

Beatrice Björn

**MITTUNIVERSITETET**

**Institutionen för Data- och elektroteknik**

**Examinator:** Mikael Hasselmalm, [mikael.hasselmalm@miun.se](mailto:mikael.hasselmalm@miun.se)

**Handledare:** Malin Larsson, [malin.larsson@miun.se](mailto:malin.larsson@miun.se)

**Författare:** Beatrice Björn, [bebj2100@student.miun.se](mailto:bebj2100@student.miun.se)

**Utbildningsprogram:** TWEUG, Webbutveckling, 120 hp

**Huvudområde:** Datateknik

**Termin, år:** VT, 2022

## Sammanfattning

Målet med projektet var att utveckla specifik funktionalitet för en plattform där vårdgivare och skolor kan komma i kontakt med varandra. De programmeringsspråk som användes under projektet var JavaScript, PHP och sass, PHP användes för back-end utvecklingen och JavaScript för front-end utvecklingen. Ramverket Vue js användes för frontend-utvecklingen och CMS-systemet SilverStripe användes till viss del för att göra plattformen redigerbar. Rapporten beskriver hur följande funktionalite för plattformen skapats i detalj: Hur BootStrap har ersatts med JavaScript för att skapa en fokusram runt sökfält, hur testdata adderats för att enkelt kunna testa webbplatsen innan produktion, hur sättet som data läses ut på ändrats från SilverStripe till Vue, hur filtrerings- och sorteringsmöjligheter lagts till för att användare enkelt ska kunna hitta den information som sökes och hur testverktyget Wave har använts för att göra webbplatsen mer användarvänlig och tillgänglig för så många som möjligt oavsett förutsättningar.

**Nyckelord:** Vue sj, SilverStripe, CMS, JavaScript, PHP, sass. Front-end, Back-end.

## Abstract

The goal of this project was to develop specific functionality for a platform where health providers and schools can get in touch with each other. The programming languages used during the project were JavaScript, PHP and sass. PHP was used for the back-end development and JavaScript was used for the front-end development. The framework Vue js was use for front-end development and the CMS-system SilverStripe was partially used to make the platform more editable. The report describes, in detail, how the following functionality for the platform was created: how Bootstrap was replaced with JavaScript to create a focus border around a search field, how test-data was added to be able to easily test the platform before production, how the way data is extracted from the database was changed from SilverStripe to Vue, how filtering and sorting options was added to allow users to easily find the desired information, and how the test tool Wave was used to make the platform more user-friendly and accessible for as many people as possible, regardless of circumstances.

**Keywords:** Vue sj, SilverStripe, CMS, JavaScript, PHP, sass. Front-end, Back-end.

## Förord

Jag vill tacka mina lärare, Mattias Dahlgren, Malin Larsson och Mikael Hasselmalm för allt det fantastiska stöd jag fått genom hela webbutvecklingsprogrammet. Ni har ända sedan programstarten varit positiva och stötande i alla lägen och hjälpt till och stöttat upp när olika moment känts svåra.

Jag har via den feedback jag fått på de olika uppgifterna haft möjlighet att alltid förbättra mitt arbete vilket har lett till stor personlig utveckling under hela programmet.

Jag vill även tacka Perrin, Ben, Ralph och Robin på Springtimesoft för att de tagit emot mig och stöttat mig genom mitt arbete. De har tillåtit mig att prova olika lösningar på problem i lugn och ro och alltid funnits nära till hands då jag behövt stöd eller hjälp.

# Innehållsförteckning

<b>Sammanfattning .....</b>	<b>2</b>
<b>Abstract.....</b>	<b>3</b>
<b>Förord.....</b>	<b>4</b>
<b>Terminologi .....</b>	<b>6</b>
<b>1 Inledning.....</b>	<b>1</b>
1.1 Bakgrund och problemmotivering .....	1
1.2 Utförande.....	1
1.3 Avgränsningar.....	2
1.4 Detaljerad problemformulering.....	2
1.5 Översikt .....	4
<b>2 Bakgrundsmaterial.....</b>	<b>5</b>
<b>3 Metod.....</b>	<b>8</b>
3.1 Utvecklingsmiljö .....	8
3.2 Git Flow.....	9
3.3 Testverktyget Wave .....	9
<b>4 Konstruktion.....</b>	<b>10</b>
4.1 Utvecklingsmiljön .....	11
4.2 Vidareutveckling av redan existerande funktionaliteten.....	12
4.2.1 Ändring av antal artiklar .....	12
4.2.2 Felformad ram runt sökfält .....	13
4.3 Testdata .....	15
4.4 Utvecklingen av ny funktionalitet .....	16
4.4.1 Konvertering från SilverStripe till Vue - Backend.....	16
4.4.2 Sortering och filtrering - Backend.....	19
4.4.3 Sortering och filtrering - Front-end.....	20
4.4.4 Konvertering från silverStripe till Vue js i ss templates.....	23
4.5 Tillgänglighet .....	26
4.6 Underhåll av publik webbplats .....	27
<b>5 Resultat.....</b>	<b>28</b>
<b>6 Analys.....</b>	<b>29</b>
6.1 Etisk och social diskussion.....	30
<b>7 Slutsatser .....</b>	<b>31</b>
<b>Källförteckning .....</b>	<b>32</b>

# Terminologi

## Förkortningar

HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
SCSS	Syntactically (or sometimes Sassy) Awesome Style Sheets (CSS preprocessor)
URL	Uniform Resource Locator
API	Application Programming Interface
CMS	Content Management System
MVC	Model View Controller
VSC	Visual Studio Code
WCAG	Webb Content Accessibility Guidelines

# 1 Inledning

## 1.1 Bakgrund och problemmotivering

Springtimesoft är ett litet webbutvecklingsföretag på Nya Zeeland bestående av ett team på fyra personer, varav två webbutvecklare och två koordinators, plus ett par frilansare.

Företaget erbjuder konsultering, webbdesign, webbutveckling och underhåll, och optimering av webbplatser och applikationer. Springtimesoft arbetar i första hand med ideella organisationer men har även utvecklat webbplatser för e-handel och andra större företag.

Företaget använder i första hand programmeringsspråket PHP men även en del JavaScript och JavaScript-baserade ramverk så som Vue. De använder sig även av CMS:et SilverStripe för att skapa användarvänliga och redigerbara webbplatser och applikationer.

Det finns en svårighet för vårdgivare och skolor att på ett smidigt sätt mötas och samarbeta. En plattform där resurser och kunskap kan kommuniceras mellan vårdgivare och skolor för bättre studenthälsa skulle kunna ses som en lösning på problemet. Springtimesoft arbetar just nu med att utveckla en plattform för organisationen Tüturu. Plattformen är menad att användas av vårdgivare och skolor för att de ska kunna lagra och dela information med varandra samt få tillgång till olika verktyg och resurser. Syftet med det arbete som ska utföras är att skapa funktionalitet för att underlätta kommunikationen mellan vårdgivare och skolor.

## 1.2 Utförande

Projektet gick ut på att skapa funktionalitet för en dashboard eller plattform för Tüturu där skolor och vårdgivare kan komma i kontakt med varandra, samt lagra och dela information i form av olika resurser. Det finns höga krav på användbarhet då plattformen ska vara enkel att använda för personer med olika erfarenhet och förutsättningar.

## 1.3 Avgränsningar

Rapporten kommer att täcka utvecklingen av viss kod och funktionalitet för plattformen Tüturu.

En del förarbete har gjorts i projektet. En roadmap som täcker planeringen av projektet togs fram och inkluderade vad som skulle ske, när och hur. Utvecklingen av denna roadmap kommer inte att beskrivas i rapporten då

arbetet utfördes innan mitt arbete med projektet startade. Personas togs fram för att undersöka vad potentiella användare önskade av plattformen. Dessa personas undersökte hur olika personer skulle komma att använda plattformen och vilken funktionalitet som behövdes implementeras för att möta önskemålen. Dessa personas togs fram innan mitt arbete med projektet startades och kommer därför inte att beskrivas i detalj. Huvudstrukturen på plattformen, design och layout togs även den fram och utvecklades innan min projektstart. Den utveckling som beskrivs i rapporten utgår från den redan färdiga strukturen som fanns för plattformen, utvecklinegn av denna kommer inte att beskrivas i detalj.

Utvecklingen av funktionalitet för plattformen har skett av samtliga utvecklare på Springtimesoft. Endast den funktionaliteten listad i problemformuleringen kommer att beskrivas i detalj i rapporten.

## 1.4 Detaljerad problemformulering

Målet med projektet var att skapa en tillgänglig och användarvänlig plattform där vårdgivare och skolor kan mötas och dela information och resurser.

En del av utvecklingen har skett på Tuturus offentliga webbplats.

### Vidareutveckling av redan existerande funktionalitet:

- Ändra antalet artiklar som visas på sidorna Providers, Schools och Community of practice. Antalet artiklar ska ändras från 9 till 15.
- På startsidan av plattformen listas de skolor som inte gjort några uppdateringar den senaste månaden. Denna lista kan bli ganska lång och ska därför ha ett max antal på 9 skolor som visas på startsida. Resterande skolor ska gå att se genom att klicka på en tillhörande länk. Om det finns färre än 9 skolor som inte uppdaterats på en månad eller mer ska länken för att "visa alla skolor" inte vara synlig.
- Sökfält på administrations plattformen och på den offentliga webbplatsen har i nuläget en felformad ram vid fokus. Ramen ska åtgärdas så att den omsluter hela sökfältet och inte bara en del som i nuläget.



### **Testdata:**

För att kunna testa plattformens olika delar måste testdata läggas till både i den lokala miljön och i den miljö som i nuläget ligger publikt för test.

- Minst 20 "Providers" ska läggas till via SilverStripe CMS
- Minst 20 Skolor ska läggas till via plattformen
- Minst 20 Notes med tillhörande actions ska läggas till via plattformen
- Minst 20 Medlemmar på skolor ska läggas till via plattformen
- Minst 20 kommentarer ska läggas till via plattformen

### **Utveckling av ny kod:**

- På sidorna Providers, Schools, Community of practice och den sida som listar skolor tillhörande en specifik Provider ska data läsas ut med hjälp av Vue istället för SilverStripe. Nya end-points, controllers och Vue-filer måste skapas för detta.
- På ovanstående sidor ska det gå att sortera den returnerade datan i bokstavsordning från A till Z och från Z till A, samt för att se de senaste uppdaterade posterna och de äldsta posterna.
- Det ska finnas ett sökfält som i det returnerade sökresultatet visar den data där "name" eller "title" delvis matchar den sökningen som skickats i sökfältet. Finns det ingen match på sökordet ska ett meddelande om detta visas på skärmen.

### **Tillgänglighet:**

- Verktöget Wave ska användas på hela plattformen för att säkerställa tillgängligheten. I den mån som går ska fel och varningar elimineras från plattformen.
- En underhållskontroll ska utföras på den publika webbplatsen. Mjukvaror ska uppdateras och webbplatsens olika delar ska testas.

## 1.5 Översikt

1. Inledning – Projektintroduktion och problemmotivering
2. Bakgrundsmaterial – Defenition av de verktyg som använts i projektet.
3. Metod – Beskrivning av den metod som användes för att utför projektet.
4. Konstruktion – Detaljerad beskrivning av hur projektet utfördes.
5. Resultat – Resultatet av projektet.
6. Analys – Analys av resultat, metod samt en etisk diskussion.
7. Slutsatser – Slutsatser och lärdomar från projektet
8. Källförteckning – Lista med de källor som använts i projektet.

## 2 Bakgrundsmaterial

### Definition av de verktyg som använts i projektet

#### Docker:

Docker är en mjukvara utvecklad för att underlätta arbetet med att skapa webbplatser och applikationer på olika maskiner och i olika operativsystem. Docker kan användas för att utveckla, köra och överföra applikationer i så kallade ”containers”, en slags virtuell miljö. En container innehåller inställningar för utvecklingsmiljöer såsom bibliotek, verktyg, kod och andra konfigurationer. Detta möjliggör att utvecklingen av en applikation eller webbplats kan ske med exakt samma utvecklingsmiljö på olika maskiner och i olika operativsystem.

Om olika delar av kod utvecklats i olika versioner av till exempel PHP och Node.js och sedan slås samman kan komplikationer och buggar uppstå. Genom att använda Docker säkerställs att alla utvecklare använder samma version av olika programvaror [1].

#### Git:

Git är ett versionshanteringssystem som används för att spåra ändringar som gjorts i filer. Genom att använda Git kan man experimentera och skapa ny kod utan att den gamla koden går förlorad. Det går att se olika versioner av kod som utvecklats för en webbplats eller applikation och det går att spåra vilka ändringar som eventuellt skapar problem i koden [2].

#### Git Flow:

Git flow är ett arbetssätt som används för versionshantering i Git. I Git Flow används en uppsättning grenar för olika faser i utvecklingen av en applikation eller webbplats. De vanligast förekommande grenarna att använda i Git Flow är, Master/Main, Develop, Feature och Hotfix.

Master/main är huvudgrenen och där läggs all kod som är redo för publicering. Ingen modifiering av kod sker i Master/Main.

Develop är en gren där all kod som utvecklats kan samlas och testas innan den sammanslås med Master/Main.

Feature är namnet på de grenar som skapas för utveckling av kod. När koden som skapats i en feature bransch är färdig och testad slås den ihop med Develop-grenen.

Hotfix är en gren som kan skapas direkt från Master/Main för snabba korrigeringar av akuta buggar eller fel i koden. Detta kan till exempel vara en felstavning på en huvudrubrik eller liknande. Detta är den enda grenen som gör direkta korrigeringar i Master/Main.

Genom att använda Git Flow kan koden i olika stadier testas och kontrolleras för att undvika större problem vid sammanslagning i Master/Main. I och med att olika typer av grenar används för olika saker är det också enkelt att veta vilka grenar som innehåller vilken typ utav kod [3].

### **Node.js:**

Node.js är en server-plattform bygd på öppen källkod som tillåter utvecklare att köra JavaScript på serversidan. Vid installering av Node.js tillkommer en stor mängd moduler och paket som kan användas för att snabbt skapa komplexa applikationer [4].

### **Composer:**

Composer är en beroende-hanterare för PHP som används för att enkelt kunna hantera och installera de bibliotek och paket som ett projekt kan behöva. I en fil med namnet "composer.json" deklarerar de bibliotek och paket som ska användas för ett projekt och composer laddar sedan automatiskt ner dessa. Composer underlättar även för utvecklare att undvika konflikter som kan uppstå genom att enkelt ladda ner och installera de senaste versionerna av olika bibliotek och paket [5].

### **PSR-2:**

PSR-2 är en kod-standard för PHP. PSR-2 består av en mängd regler/riktlinjer för hur PHP-kod ska vara formaterad. PSR-2 används i första hand för att göra koden lättläst, enkel att förstå och enhetlig.

Några av de viktigaste reglerna i PSR-2 är en tab eller fyra mellanslag vis start av kodblock, att använda "camelCasing", enkla citattecken för strängar och radlängden ska inte vara längre än 80 tecken [6].

### **Jira och job ticketing/uppgifter:**

Jira är en applikation utvecklad av Atlassian för att underlätta arbetet med planering, distribuering av uppgifter och annat som tillkommer projektarbeten. Ett vanligt sätt att använda Jira är att skapa en planering med så kallade job tickets/uppgifter. Varje uppgift eller ticket innehåller en detaljerad beskrivning om vad som ska göras och av vem för att slutföra uppgiften. Det är även vanligt att sätta en prioritering på uppgifterna så som, låg, hög eller medel prioritering för att inblandade kan se vilka uppgifter som är mest brådskande.

Uppgifterna delas sedan upp i olika faser, till exempel, "att göra", "pågående", "frågor och problem", "avklarad". Olika medlemmar i projektet kan

Tüturu Dashboard – Utvecklingen av en plattform där skolor och vårdgivare kan mötas

Beatrice Björn

2023-06-02

sedan flytta uppgifter mellan faserna och alla inblandade får på så vis en tydlig bild av vad som sker i projektet [7].

### **Bootstrap:**

Bootstrap är ett CSS-ramverk bestående av HTML, CSS och JavaScript. Bootstrap används för att snabbt och enkelt skapa användarvänliga och responsiva webbplatser och applikationer. Ramverket är uppbyggt av en mängd färdiga kod-block som kan implementeras via olika Bootstrap-klasser. Detta besparar utvecklare mycket tid på skapandet av layout och design för webbplatser och applikationer. Bootstrap har även stort stöd i olika webbläsare vilket säkerställer att webbplatser och applikationer till stor del ser likadana ut på olika enheter, storlekar på skärm och i olika webbläsare [8].

### **Git Lab:**

Git lab är en plattform baserad på öppen källkod där större företag kan lagra och dela projekt i form av git repositorys. Det går enkelt att skapa ny repositorys och bestämma vilka inom en organisation som ska ha tillgång till dessa [9].

### **SilverStripe:**

SilverStripe är ett CMS skrivit i PHP och baserat på öppen källkod. SilverStripe används i första hand för att skapa dynamiska webbplatser och applikationer där innehållet enkelt kan redigeras utan att användaren behöver skriva kod eller ha större it-erfarenhet. Till skillnad från andra tillgängliga CMS är SilverStripe utformat så att utvecklare kan skapa egen funktionalitet för att möta specifika behov. SilverStripe har även inbyggd SEO och har möjlighet att interagera med andra tredjeparts verktyg. För hantering av webbplatser och applikationers databaser används SilverStripe MVC. Detta är ännu en punkt som gör att SilverStripe skiljer sig från andra CMS-system [10].

## 3 Metod

### 3.1 Utvecklingsmiljö

VCS användes för all utveckling av kod i projektet. VSC är en text editor baserad på öppen källkod. Springtimesoft har en bestämd uppsättning tillägg och konfigurationer som laddas ner för att VSC ska kunna användas på samma sätt oavsett vilken maskin utvecklingen sker på [11].

**De tillägg som laddas ner och används är:**

Vue Language Feature (Volar)

Vetur

Sort Lines

Snippet Creator

SilverStripe Classic

SCSS Formatter

Rewrap

phpcs

PHP Namespace Resolver

PHP DocBlocker

php cs fixer

php cs fixer

Git History

EditorConfig for VS Code

Duplicate Selection or Line

Duplicate Action

Docker

Diff Tool

### **3.2 Git Flow**

För att utvecklingen av de olika delarna på plattformen ska vara enkel att följa användes Git Flow för versionshantering. Git Flow möjliggör att all utveckling av kod för webbplatsen versionshanteras och delas upp i mindre delar. Feature-grenar används för olika uppgifter och kontrolleras innan de sammanslås med den kod som är färdig för produktion.

### **3.3 Testverktyget Wave**

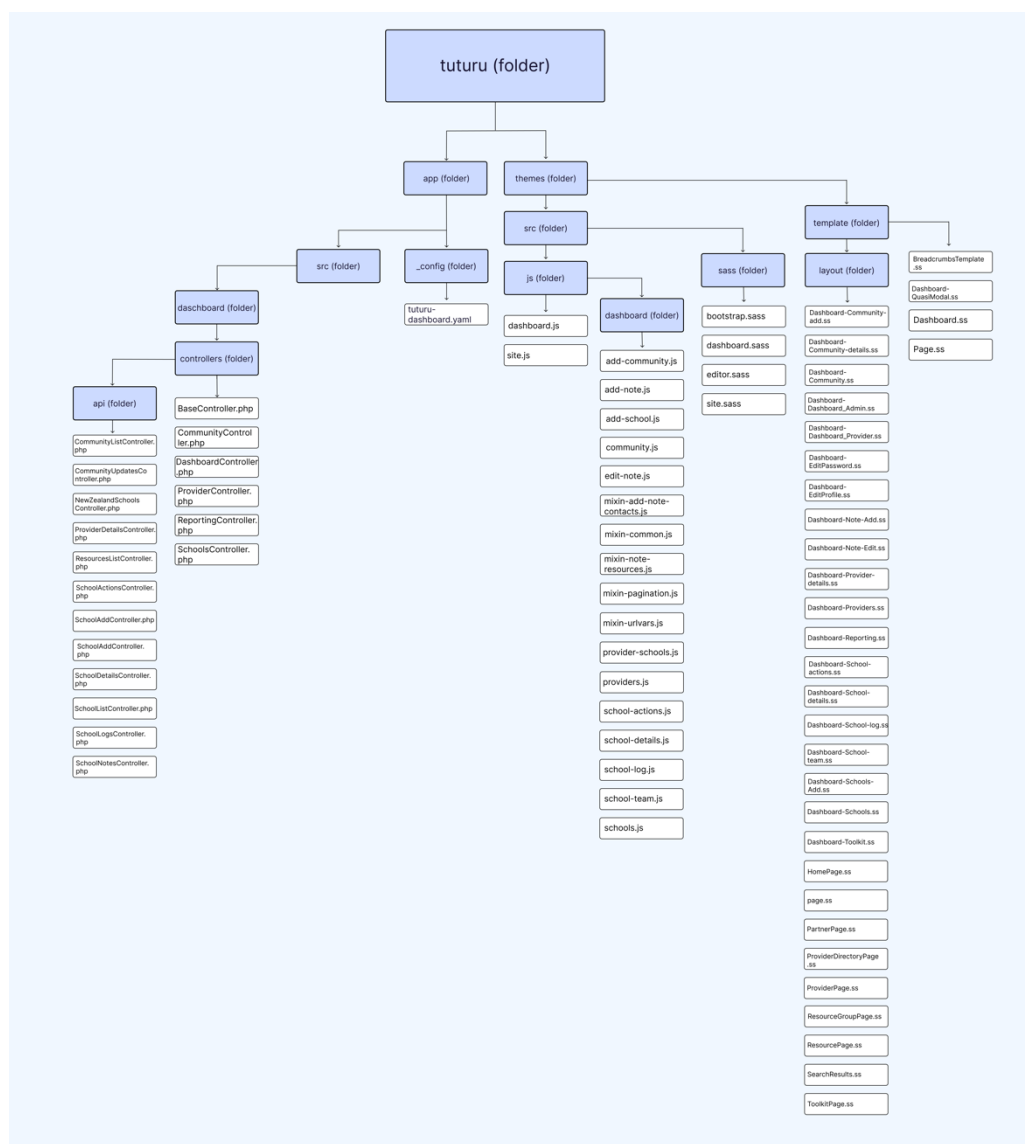
För att säkerställa att plattformen är användarvänlig och tillgänglig användes testverktyget Wave. Wave kan installeras som tillägg i webbläsaren Chrome eller fristående via deras webbplats.

Wave kontrollerar att det finns tillräckligt med kontrast mellan text och bakgrund, att alla element används på rätt sätt och förutsätter därför att webbplatsen som testas blir tillgänglig för så många som möjligt oavsett erfarenhet eller förutsättningar.

Wave genererar en rapport av varje webbsida som testas och föreslår vilka ändringar som bör göras och hur dessa påverkar tillgängligheten för webbsidan. För att uppnå bästa resultat användes Wave kontinuerligt under utvecklingen av den nya funktionaliteten på plattformen [12].

## 4 Konstruktion

Figur 1 visar en site-mat över de filer som användes under projektet. Notera att detta inte är alla filer som finns i projektet utan endast de som användes under mitt arbete med projektet. Se figur 1.



Figur 1. Bild av de filer som användes under projektet.



## 4.1 Utvecklingsmiljön

Springtimesoft har under många år utvecklat ett system för att underlätta arbetet med olika projekt. Detta för att alla utvecklare inom företaget ska skapa så lika kod som möjligt och på såvis minimera konflikter när kod sammanslås.

Springtimesoft använder git lab för att lagra och dela sina projekt. Ett projekt med namnet "tuturu" har klonats från git lab så att utveckling har kunnat ske från den lokala maskinen. Vid start av ett nytt projekt måste rätt grenar installeras. Genom kommandot "git checkout master" skapades master-grenen med all kod för projektet lokalt. Sedan kördes kommandot "git flow init" för att skapa develop-grenen, "git checkout develop" kördes sedan för att byta till den senaste versionen av develop-grenen.

För att få tillgång till rätt utvecklingsmiljö installerades och testkördes Docker. När allt fungerade som det skulle öppnades Tuturu projektet i terminalen och kommandot "sts up" kördes. Detta startar en Docker container med Springtimesofts programinställningar. För att installera den senaste versionen av Composer kördes kommandot "sts sh composer install", detta installerade de paket som krävdes för projektet.

För att börja utvecklingen med projektet skapades en feature-gren, detta görs alltid från develop-grenen. Först kontrollerades om det fanns uppdateringar i develop-grenen och dessa laddas ner genom att synka grenen. Sedan kördes kommandot "git flow feature start" följt av det namn grenen ska ha. På en Jira-ticket finns alltid ett id-nummer, detta nummer används som namn på de feature-grenar som skapas, tex "TU-205".

För att installera de node.js-paket som behövdes för projektet kördes kommandot "sts tools make install" och för att sedan bygga webbplatsen kördes kommandot "sts tools make build". För att få direkta uppdateringar i webbläsaren användes kommandot "sts tools make watch". Detta startar en watch-task som håller koll på de ändringar som sparas i koden och uppdaterar webbläsaren utefter dessa.

För att installationen av de konfigurationer som används i VSC installerades först programmet "code sniffer". Code sniffer är ett test program som kollar efter kod som inte följer en så kallad "Magento Code Standard", kommandot "sudo apt install php-codesniffer" kördes i terminalen för att installera code sniffer. Sedan kördes kommandot "./vscode/set-default-settings.php" för att installera de rätta konfigurationerna.

## 4.2 Vidareutveckling av redan existerande funktionaliteten

### 4.2.1 Ändring av antal artiklar

För att ändra antalet artiklar som visas på sidorna Providers, Schools och Community of practice öppnades filerna "ComunityController.php", "ProviderController.php", "SchoolsController.php", i Tuturu projektet. Sedan letades funktionerna "sortedCommunityUpdates", "sortedSchools" och "sortedProviders" upp och "setPageLength" ändrades från 9 till 15, se figur 1 [13].

```
public function sortedSchools()
{
    $schools = $this->member()->Schools();

    $sortedSchools = $this->getSortByLatest() ?
    $schools->sort('"LastUpdated"', 'DESC') : $schools;

    return PaginatedList::create(
        $sortedSchools,
        $this->getRequest()
    )->setPageLength(15);
}
```

Figur 1. Bild av kod för att ändra antalet artiklar som visas.

För att på startsidan endast visa 9 skolor som inte gjort några uppdateringar den senaste månaden användes filen DashboardController och funktionen getSchoolsUneditedForMonth. Metoden "Limit" följt av siffran 9 lades till för att endast visa 9 skolor på startsidan. Se figur 2 [14].

```
/**
 * Used to return a list of schools unengaged for 30 days
 *
 * @return ArrayList
 */
public function getSchoolsUneditedForMonth()
{
    $lastMonth = date('Y-m-d', strtotime('-30 days'));

    return School::get()->filter('LastUpdated:LessThan', $lastMonth)
    |>sort('LastUpdated', 'ASC')->limit(9);
}
```

Figur 2. Bild av kod för att sätta ett maxantal på 9 artiklar.

#### 4.2.2 Felformad ram runt sökfält

Det finns i nuläget totalt tre sökfält, ett på plattformen och två på den tillhörande publika webbplatsen för Tüturu. Dessa sökfält får vid klick en ram, men ramen har fel form. Se figur 3 och figur 4.



Figur 3. Bild av sökfält med fel-formad ram.



Figur 4. Bild av sökfält med fel-formad ram.

Problemet med dessa sökfält var att fokus ramen som lades till via ramverket Bootstrap endast applicerades på input-elementet. Det element som innehåller ikonen med förstoringsglaset är ett button-element och därav appliceras ingen ram runt detta. Bägge elementen omges av ett div-element [15].

Det första som gjordes för att lösa detta var att ta bort den ram som automatiskt appliceras via Bootstrap. I filen site.scss lades en box-shadow: none; på input-elementen för att ta bort default ramen.

För att skapa en fokus-ram som läggs på det omslutande div-elementet användes JavaScript. Detta på grund av att focus endast går att applicera på ett fåtal element, div-element är inte inkluderade i den listan.

I filen site.js i mappen js skapades längst ner koden för att lägga till en fokus ram på div-elementet. En klass med namnet "search-input" lades på de input-element som används för sökrutor. Sedan användes document.querySelectorAll("input.search-focus") för att hämta in alla input-element med klassen search-focus och lagra dessa i en variabel med namnet "search-Fields".

En forEach-loop användes för att loopa igenom alla element i variabeln searchFields och en händelselyssnare som reagerar när fältet är i fokus lades på input-fälten. När elementet är i fokus triggas en anonym funktion som

lägger till klassen "focus" på parent-elementet (det element som om- sluter input-elementet och button-elementet). För att ta bort klassen "focus" från elementen när input-elementet inte är i fokus längre skapades ännu en likadana funktion som den för att applicera focus men i stället för en eventlyssnare som lyssnar på focus så användes "blur". I stället för att lägga till en klass så tas klassen "focus" bort från elementen när funktionen triggas.

Genom att ha löst problemet på detta vis blir det enkelt att applicera samma fokus-ram på framtida sökfält genom att lägga till klassen "input-search" på elementen. Se figur 5. [16]

```
// add a focus border to search-fields
const searchFields = document.querySelectorAll("input.search-focus");
searchFields.forEach((el) => {
  el.addEventListener('focus', () => {
    el.parentElement.classList.add('focus');
  });
  el.addEventListener('blur', () => {
    el.parentElement.classList.remove('focus');
  });
});
```

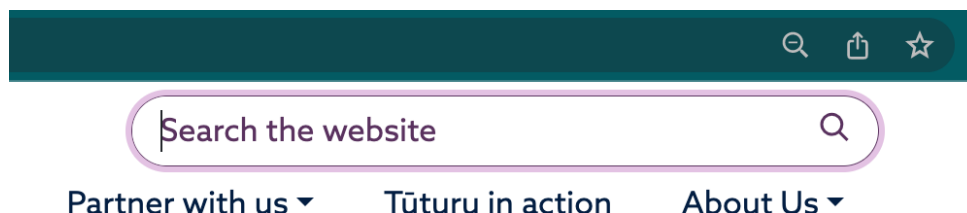
Figur 5. Bild av koden för att lägga till och ta bort klass vid focus och blur.

I filen site.scss skapades den ram som läggs på alla element med klassen "focus" inom ett div-element. En ram lades till med samma färg och stil som den originalram som lades till av BootStrap. Se figur 6

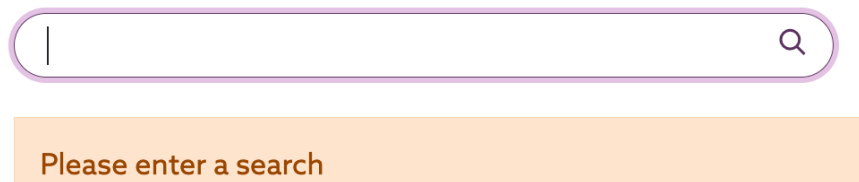
```
//Adds focus border to search fields and removes default bootstrap focus border
div.focus {
  outline: solid 3.5px #981d9846;
}
```

Figur 6. Bild av kod för hur element med klassen "focus" stylas.

Efter att den nya koden applicerades har fokus-ramen runt sökfälten fått rätt form. Se figur 7 och figur 8.



Figur 7. Bild av korrekt formad fokusram

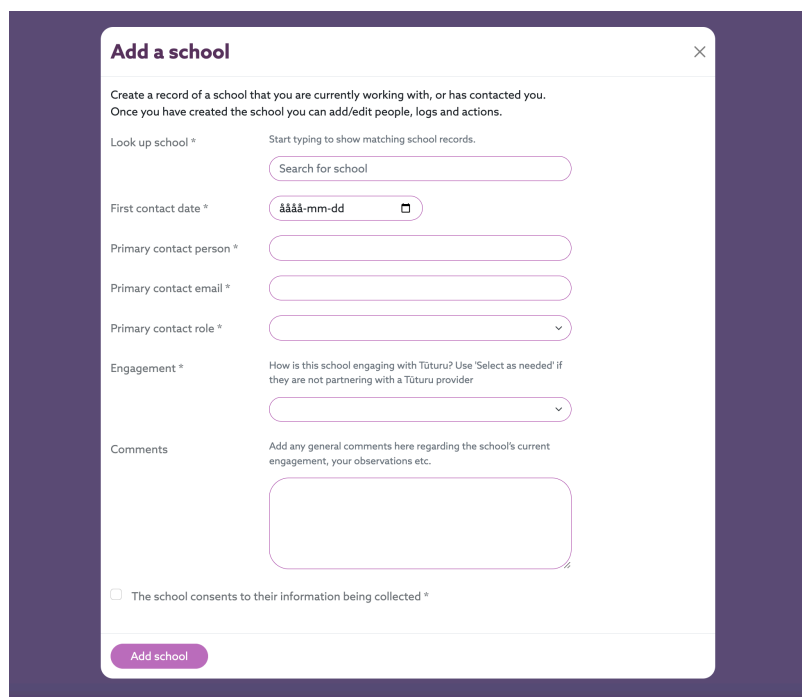


A search bar with a magnifying glass icon and a placeholder text "Please enter a search".

Figur 8. Bild av korrekt formatad fokusram

### 4.3 Testdata

För att kunna testa paginering och annan funktionalitet krävs testdata. Mycket av den data som krävdes för att testa plattformen kunde läggas till via den funktionalitet som redan var implementerad på plattformen. 20 nya skolor skulle läggas till och detta gjordes genom att klicka på knappen "Add School" och fylla i formuläret för att lägga till en skola. När 20 skolor lagts till i databasen skapades "Notes" för dessa, det behövdes minst 20 notes och varje note skulle ha tillhörande "Actions". Se figur 9.

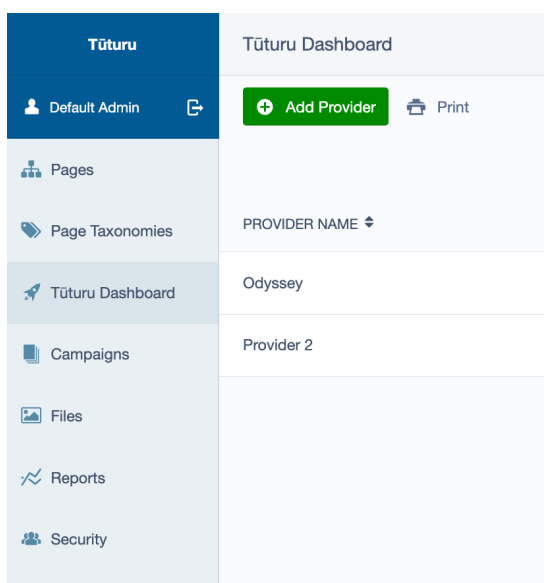


A screenshot of the "Add a school" form in the Tüturu Dashboard. The form is titled "Add a school" and includes a close button (X). It contains the following fields and instructions:

- Look up school \***: Start typing to show matching school records. Search for school
- First contact date \***:
- Primary contact person \***:
- Primary contact email \***:
- Primary contact role \***:
- Engagement \***: How is this school engaging with Tüturu? Use "Select as needed" if they are not partnering with a Tüturu provider.
- Comments**: Add any general comments here regarding the school's current engagement, your observations etc.
- ☐ The school consents to their information being collected \*
- Add school** button

Figur 9. Bild av formulär för att lägga till skolor.

För att fler vårdgivare skulle läggas till användes SilverStripe CMS. Via fliken "Tüturu Dashboard" kunde nya "Providers" läggas till. 20 Providers lades till för att kunna testa pagineringen på plattformen. Se figur 10



Figur 10. Bild av SilverStripe CMS för att lägga till Provider.

## 4.4 Utvecklingen av ny funktionalitet

### 4.4.1 Konvertering från SilverStripe till Vue js – Back-end

För att koden för plattformen skulle bli mer unison och lättläst samt att sök och sorterings-funktionalitet skulle kunna implementeras med Vue js, ändrades koden för att läsa ut data från SilverStripe till att använda Vue js. De sidor som ändrades var Providers, schools (tillhörande en provider), Community of Practice och schools. För att konvertera hur data läses ut från SilverStripe till Vue på sidan "Providers" skapades först en rout för de nya anropen. Detta gjordes i filen tuturu- dashboard.yml. "dashboard/api/providers//\$Action/\$ProviderID" lades till som URL där "\$Action" står för den funktion som ska köras. Sedan lades sökvägen till den Controller som ska användas till, "Tüturu/Controllers/API/ProviderDetailsController".

Sedan skapades filen ProviderDetailsController i api-mappen. "use" användes för att inkludera modellerna där databastabellerna skapas för School och Provider.

För att bestämma vilka funktioner som ska tillåtas köras via Controllern listades funktionerna "ListSchools" och "ListProviders" inom allowed\_actions [17].

För att rätt url ska vara kopplad till rätt funktion skapades url\_handlers som länkar den aktuella URL:en till rätt funktion. Om url:en slutar med list-schools/ samt ett ID från en Provider så ska funktionen ListSchools köras. Om Url:en slutar med ListProviders så ska funktionen ListProviders köras. Se figur 11 [18].

```
class ProviderDetailsController extends BaseApiController
{
    /**
     * Allowed actions.
     *
     * @var array
     *
     * @config
     */
    private static $allowed_actions = [
        'listSchools',
        'listProviders',
    ];

    /**
     * URL handlers
     *
     * @var array
     */
    private static $url_handlers = [
        'listSchools/$ProviderID!' => 'listSchools',
        'listProviders' => 'listProviders',
    ];
}
```

Figur 11. Bild av allowed\_actions och url\_handlers.

En init funktion skapades för att kontrollera att besökaren av sidan har rätt behörighet för att kunna se innehållet. Först används "parent::init()" för att köra init funktionen i den överordnade klassen. Sedan används en if-sats för att kontrollera om den nuvarande användaren inte är en admin, är så fallet visas en text på skärmen som berättar för besökaren att hen inte har behörighet att se materialet på denna sida och felkoden 401 returneras.

Funktionen ListSchools skapades och tar \$request som parameter. "ProviderID" hämtas via Request→param och lagras i en variabel med namnet \$providerID.

"provider::get()->byID(\$providerId)" användes för att hämta all data associerad med den provider vars id har angivits i url:en. En if-sats användes för att kontrollera om det id som angivits inte tillhör någon provider. Om så är

fallet returneras ett meddelande om detta tillsammans med felkoden 404 (not found).

I en variabel med namnet \$schools lagras all data tillhörande schools länkade till en provider i databasen. En variabel med namnet "data" sattes till en tom array och sedan skapades en foreach loop som loopar igenom all data som hämtats in. I stället för ett datum för när posten uppdaterades senast ska antalet dagar sedan uppdatering visas. För detta används data objektet "lastUpdated" följs av →ago. Detta gör att dagarna sedan den senaste uppdateringen räknas och skrivs ut i form av "(antalet dagar) days ago".

Alla "actions" som finns i databasen tillhör olika "notes" i en "många till en" relation. För att kunna läsa ut antalet "actions" måste därför först antalet notes loopas igenom. En foreach loop används för att loopa igenom alla notes och metoden Count() används för att sedan kunna plocka ut antalet actions som tillhör varje Note.

En tom array med namnet data fylls på med de värden som läses ut från databasen och "\$data" returneras i json-format. Se figur 12.

```
public function listSchools($request)
{
    $providerID = $request->param('ProviderID');

    $provider = Provider::get()->byID($providerID);
    if (!$provider) {
        return $this->JSONError('Provider not found', 404);
    }

    $schools = $provider->Schools();

    $data = [];
    foreach ($schools as $s) {
        $date = $s->dbObject('LastUpdated')->Ago();
        $actionsCount = 0;
        foreach ($s->Notes() as $n) {
            $actionsCount += $n->Actions()->Count();
        }
        $data[] = [
            'ID' => $s->ID,
            'Name' => $s->Name,
            'LastUpdated' => $date,
            'NoteCount' => $s->Notes()->Count(),
            'ActionsCount' => $actionsCount,
            'EngagementTitle' => $s->Engagement()->Title,
            'IsStale' => $s->isStale(),
        ];
    }

    return $this->JSONResponse($data);
}
```

Figur 12. Bild av koden för att hämta data från databasen.



På samma sätt som ovan skapades routes, controllers och funktionalitet för att hämta data för Providers, Schools och Community of Practice.

#### 4.4.2 Sortering och filtrering – Back-end

När funktionaliteten för att hämta data från databasen var färdig skapades funktionalitet för att filtera och sortera den data som hämtas. Högst upp i funktionen listProviders skapades tre variabler, sortCol, sortDirection och filter. SortCol får ett strängvärde av "lastUpdated", sortDirection får ett strängvärde av "DESC" och filter får ett värde av en tom textsträng. För att på sidan Schools även kunna läsa ut de skolor som inte uppdaterats på en månad eller mer läggs i funktionen "listAllSchools" även en variabel med namnet "unedited" som får ett default värde av 0.

För att kunna sortera och filtrera returnerad data via namn från A till Z, namn från Z till A, senast uppdaterade poster och äldsta poster och via sökning används först Json-decode för att konvertera den json-data som returneras till en associativ array och lagra denna i en variabel med namnet "post". En if-sats används för att kontrollera om post är en array, om så är fallet används ytterligare en if-sats för att tilldela variablerna sortCol, sortDirection och filter nya värden baserade på den returnerade datan. Metoden strtolower används för att konvertera returnerad data till små bokstäver och metoden trim används för att ta bort eventuella mellanslag i början eller slutet av en sträng.

Om filter har ett värde returneras en lista med de skolor vars namn matchar/delvis matchar (har gemensamma bokstäver) det värde som finns i filter och sorteras beroende på vilket värde som tilldelats sortCol eller sortDirection. Se figur 13.

```
public function listAllSchools($request)
{
    $sortCol      = 'LastUpdated';
    $sortDirection = 'DESC';
    $filter       = '';
    $unedited     = 0;

    $post = json_decode(file_get_contents('php://input'), true);
    if (is_array($post)) {
        if (in_array($this->arrayVal($post, 'col'), ['SchoolInfo.Name', 'LastUpdated'])) {
            $sortCol = $this->arrayVal($post, 'col');
        }

        if (in_array(strtolower($this->arrayVal($post, 'direction')), ['asc', 'desc'])) {
            $sortDirection = $this->arrayVal($post, 'direction');
        }

        $filter = trim($this->arrayVal($post, 'filter'));
        $unedited = ($this->arrayVal($post, 'unedited')) ? 1 : 0;
    }

    $schools = $this->member()->schools();
    if ($unedited) {
        $lastMonth = date('Y-m-d', strtotime('-30 days'));
        $schools = $schools->filter('LastUpdated:LessThan', $lastMonth)
            ->sort('LastUpdated', 'ASC');
        $schools = $schools->filter([
            'SchoolInfo.Name:PartialMatch' => $filter,
        ]);
        $schools = $schools->sort($sortCol, $sortDirection);
    } else {
        if ($filter) {
            $schools = $schools->filter([
                'SchoolInfo.Name:PartialMatch' => $filter,
            ]);
        }
        $schools = $schools->sort($sortCol, $sortDirection);
    }

    if (!$schools) {
        return $this->JSONError('Schools not found', 404);
    }
}
```

Figur 13. Bild av koden för att sortera och filtrera resultat.

#### 4.4.3 Sortering och filtrering – Front-end

För att med Vue kunna läsa ut den data som returneras via ovanstående controllers skapades fyra JavaScript-filer, schools.js, providers.js, provider-schools.js och community.js. Högst upp i filerna importeras "create app", "mixin-common" och "urlVarsMixin" vilket skapar en Vue applikation och inkludera filerna mixin-common.js och urlVarsMixin.js där ytterligare kod för sortering och filtrering finns [19].

I objektet "urlVars" sätts sortBy till lastUpdated, sortDirection till DESC, filter sätts till en tom textsträng och unedited sätts (i schools.js) till 0, dessa är de default värden som sätts när sidan laddas första gången.

I Data objektet sätts schools till en tom array som sedan kommer att fyllas på med skolor och noResults sätts till 0 för att felmeddelande om att inga skolor finns tillgängliga endast ska visas om värdet inte är lika med noll.

För att skolor ska hämtas in så fort sidan laddas in lades en funktion med namnet `getData` i `mounted`.

I `methods` skapades en funktion med namnet `setSort` som triggas varje gång en användare ändrar sorteringen eller filtreringen på sidan. En variabel med namnet `o` lagrar det valda fältet ifrån `select`-elementet för sorteringsalternativ. En `if`-sats används för att kontrollera om `o` inte finns och returnerar om så är fallet hela listan med skolor.

En variabel med namnet `sortCol` skapades och lagrar `o.dataset.col` för att kunna kommunicera direkt med DOM, Detsamma görs för `o.dataset.direction` och denna lagras i en variabel med namnet `sortDirection`.

Två `if`-satser används för att kontrollera om `sortCol` eller `sortDirection` har tilldelats värden och uppdaterar `urlVars.sortBy` och `urlVars.sortDirection` utefter om nya värden tilldelats eller ej. Se figur 14.

```
mounted: function () {
  this.getData();
},

methods: {
  // set school sorting variables - getData() is automatically called via urlVarsOnAfterUpdate()
  setSort: function (el) {
    let o = el.options[el.selectedIndex];
    if (!o) {
      return;
    }
    // if el has selected option
    let sortCol = o.dataset.col;
    let sortDirection = o.dataset.direction;

    if (sortCol) {
      this.urlVars.sortBy = sortCol;
    }
    if (sortDirection) {
      this.urlVars.sortDirection = sortDirection;
    }
  },
}
```

Figur 14. Bild av kod för att sätta värden på sortering och filtrering.

En metod med namnet `getData` skapades och den endpoint som skapats för att göra ett post-anrop används för att hämta in alla skolor. Om `sortCol` och `sortDirections`, (samt `unedited` för skolor) har värden kommer dessa att inkluderas i endpointen och data kommer läsas in sorterad och filtrerad enligt parametrarna i `url:en`. Metoden trim lades på filter för att ta bort eventuella mellanslag i början och slutet av den textsträng som skickas.

I en variabel med namnet `noResult` lagras det nollvärde som returneras om listan med skolor skulle vara tom.

En metod med namnet `urlVarsOnAfterUpdate` skapades och kör metoden `getData`. Detta gjordes för att data ska hämtas in i ny relevant ordning varje gång filtreringen eller sorteringen ändras på sidan.

För att kunna montera vue-komponenten på ett DOM-element användes `.mount` följt av `Id:t` på det DOM-element där komponenten ska monteras. I det här fallet `".mount('#AllSchools')"`. Se figur 15.

```
getData: function () {
  let self = this;
  let data = {};
  data.col = this.urlVars.sortBy;
  data.direction = this.urlVars.sortDirection;
  data.filter = this.urlVars.filter.trim();
  data.unedited = this.urlVars.unedited;

  self.post('/dashboard/api/schoolList/listAllSchools/', data, function (response) {
    self.schools = response.data;
    self.noResult = self.schools.length == 0;
  });
},

// manually invoke this.sortLogs() after URL vars change
urlVarsOnAfterUpdate: function () {
  this.getData();
  this.paginationReset();
},

}.mount('#AllSchools')
```

Figur 15. Bild av funktionen `getData`.

#### 4.4.4 Konvertering från SilverStripe till Vue i ss-templates

Det första som gjordes för att konvertera template-sidorna var att kommentera bort all kod som redan fanns i filerna. Detta gjordes för att fortsatt kunna se koden och kopiera de delar som skulle behållas. All kod i filerna förutom h1-elementen omges av ett div-element med respektive id som satts i .mount i js-filerna. Detta för att länka filerna till de rätta js-filerna. [20].

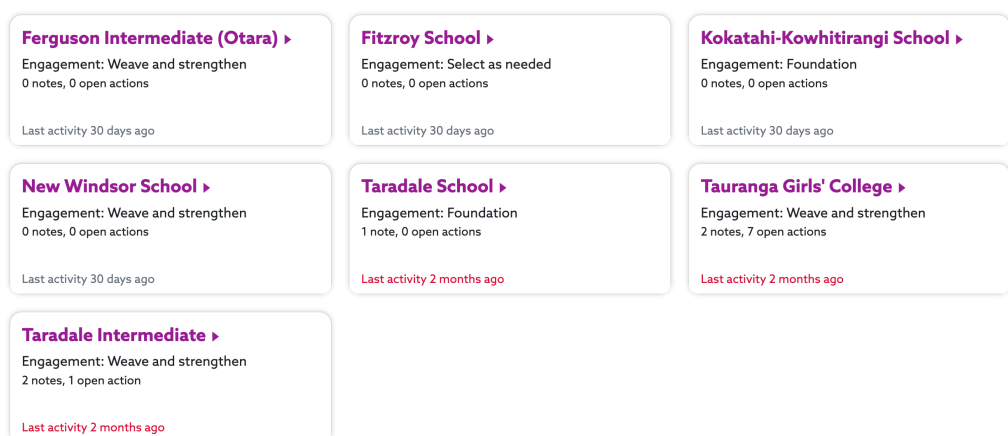
Följande kod har applicerats i filerna Dashboard-community.ss, Dashboard-Provider-details.ss, Dashboard-Providers.ss och Dashboard-Schools.ss. För att enkelt beskriva hur koden applicerats visas nedan endast den kod som applicerats i filen Dashboard-Schools.ss.

V-if används för att loopa igenom alla skolor som finns lagrade. Titeln på skolan lades inom ett a-element för att besökaren till en sida som visar information om den skola vars titel klickats [21].

Med v-bind på href- attributet används "s.Link" för att länken ska leda till rätt skola. Inom h2-elementet läses även skolans namn ut med hjälp av Vue-syntax "{{ s.Name }}". Inom p-element läses "s.EngagementTitle", "s.NoteCount", "s.ActionsCount" ut. Genom att använda en v-if sats kan, om skolan har ett "senast uppdaterad" värde, detta läsas ut. För att texten ska visas i rött om skolan inte uppdaterats på mer än en månad används ":class='s.IsStale ? 'text-danger' : 'text-muted'", detta är en typ utav if-sats eller villkor där "s.IsStale ?" kontrollerar om IsStale är satt till true och lägger till bootstrap-klassen "text-danger" om så är fallet. Frågetecknet används som "else" skulle använts i en if-sats. Om den första kontrollen inte returnerar true, läggs i stället klassen "text-muted" till på elementet och skriver ut texten i en gråskala i stället för rött. Se figur 16 och figur 17 [22].

```
<template v-if="schools.length">
  <div class="row row-cols-1 row-cols-md-2 g-4">
    <template v-for="s in paginatedList(filteredView)">
      <div class="col-xl-4">
        <div class="card h-100">
          <div class="card-body">
            <h2 class="h3 card-title">
              <a :href="s.Link" class="stretched-link">{{ s.Name }} <i class="icon-triangle-right small"></i></a>
            </h2>
            <p class="mb-0">Engagement: {{ s.EngagementTitle }}</p>
            <p class="small">
              {{ s.NoteCount }} note<template v-if="s.NoteCount != 1">s</template>,
              {{ s.ActionsCount }} open action<template v-if="s.ActionsCount != 1">s</template>
            </p>
          </div>
          <div class="card-footer small" :class="s.IsStale ? 'text-danger' : 'text-muted'">
            <template v-if="s.LastUpdated" >
              Last activity {{ s.LastUpdated }}
            </template>
            <template v-else>No activity</template>
          </div>
        </div>
      </div>
    </template>
  </div>
</template>
```

Figur 16. Bild av hur data läses ut med Vue.



Figur 17. Bild av hur resultatet av ovanstående kod ser ut på plattformen

För att lägga till input-fältet där sökningar kan göras och den drop-down som erbjuder sortering av skolor från A-Z, Z-A, senast uppdaterade och icke uppdaterade skapades först ett input-element. I input-elementet används v-bind för att sätta input-elementets value till det värde som lagras i "urlVars.filter". En händelselyssnare som lyssnar på "change" lades till på input-elementet och triggas när värdet i input-fältet ändras. Värdet på "urlVars.filter" uppdateras då med det nya värdet som hämtats från input-fältet. Se figur 18.

```
<div id="AllSchools">
  <div class="row">
    <div class="col-md-4 mb-2 mt-4">
      <label for="Filter" class="d-none">Filter schools</label>
      <div class="search-input input-group">
        <input type="text" id="Filter" class="form-control search-focus border-0 form-control-sm" placeholder="Filter"
          :value="urlVars.filter" @change="urlVars.filter = $event.target.value.trim()" />
        <button type="button" @click="urlVars.filter=''" v-if="urlVars.filter" class="btn border-0"><i class="icon-cross"></i></button>
      </div>
    </div>
  </div>
</div>
```

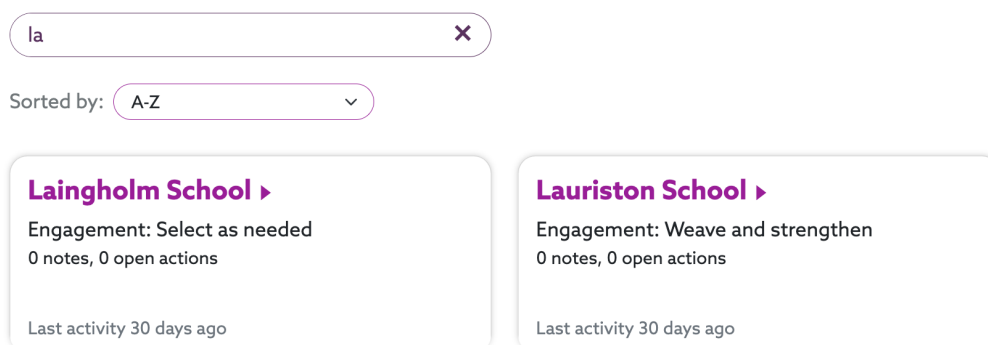
Figur 18. Bild av hur sökfältet är uppbyggt.

För det select-element där det ska gå att välja hur listan med skolor ska sorteras lades ännu en händelselyssnare som lyssnar på "change". I varje option-elementen lades sedan attributet ":selected" följt av det värde som satts på "urlVars.SortBy" och om listan ska sorteras i stigande eller fallande ordning. Se figur 19 och figur 20 [23].

```
<div class="my-3 row">
  <div class="col">
    <label for="SortSchools">Sorted by: </label>
    <select id="SortSchools" class="form-select form-select-sm mb-3 ms-2" @change="setSort(\$event.target)">
      <option data-col="LastUpdated" data-direction="DESC" :selected="urlVars.sortBy=='LastUpdated' && urlVars.sortDirection=='DESC'">Last activity (recent)</option>
      <option data-col="LastUpdated" data-direction="ASC" :selected="urlVars.sortBy=='LastUpdated' && urlVars.sortDirection=='ASC'">Last activity (oldest)</option>
      <option data-col="SchoolInfo.Name" data-direction="ASC" :selected="urlVars.sortBy=='SchoolInfo.Name' && urlVars.sortDirection=='ASC'">A-Z</option>
      <option data-col="SchoolInfo.Name" data-direction="DESC" :selected="urlVars.sortBy=='SchoolInfo.Name' && urlVars.sortDirection=='DESC'">Z-A</option>
    </select>
  </div>
</div>
```

Bild 19. Bild av hur drop-down-fältet är uppbyggt.

## Schools page



Figur 20. Bild av hur sök- och drop-downfälten ser ut på plattformen.

Under sökfältet och drop-down alternativen lades en div innehållandes ett meddelande om att inga matchningar hittats på sökordet. Genom att används v-if kontrolleras om "noResult" är satt till true eller false. Div-elementet visas då endast om noResult returnerar true.

För att den länk som visar alla skolor som inte uppdaterats på en månad eller mer skulle fungera lades parametern unedited in i href-attributet och sattes till 1. För att länken till alla skolor som inte uppdaterats på mer än en månad inte ska visas om det finns färre skolor än 9 lades ett mindre än tecken följt av 9 in i den SilverStripe loop som används för att loopa igenom alla skolor. Se figur 21.

```
<% if $SchoolsUneditedForMonth.Count > 9 %>
<div class="card-footer mx-3">
  <a href="$Link(schools)?unedited=1" class="card-link">View all schools not updated for a month or more</a>
</div>
<% end_if %>
```

Figur 21. Bild av hur länken endast visas om det finns fler skolor än 9.

## 4.5 Tillgänglighet

För att säkerställa tillgänglighet på plattformen användes testverktygen Wave. När Wave startas på en webbsida visas en lista med olika element, hur de använts och om de har en negativ inverkan på hur tillgänglig sidan är.

När den tidigare koden för plattformen testades i Wave så genererades en hel del fel och varningar angående tillgänglighet. Huvudrubriker har inte använts i fallande ordning vilket gör att en skärmläsare kommer att läsa upp innehållet på webbplatsen i oordning. Label-element har använts där det egentligen borde vara p- eller span- element och det fanns även många input och select-element som saknar tillhörande label-element vilket gör det svårt för personer som inte ser skärmen att förstå vilket input-element som hör till vilken information.

För att korrigera ordningen på titel-elementen ändrades de så att de hamnade i rätt ordning på alla sidor där problemet uppstod. För att inte stylingen på elementen skulle påverkas lades en Bootstrap-klass med namnet på original-elementet till. Om elementet ursprungligen var ett h5-element och nu ändrats till ett h2-element kan en klass av h5 läggas till och detta gör att elementet stylas som om det vore ett h5-element fast än det egentligen är ett h2-element. [24]

På de ställen där label-element har använts utan att tillhöra något inputfält ändras elementen till p-element. En del element har en grå text och får därför en klass av "text-muted".

På några ställen i den tidigare koden användes p-element i stället för label-element till input-element. Där har p-elementen bytts ut mot label-element med "for" attributet satt till det id som finns på input-elementet för att koppla ihop dessa. Det fanns även ställen där input-element helt saknade label-element eller inte har kopplats på ett korrekt vis. Där label-element saknas helt lades de till och om de inte ska vara synliga för personer som ser skärmen så lades klassen "d-none" till för att gömma elementet men tillåta skärmläsare att fortfarande se elementet. [25]

På sidorna för att lägga till Community of Practice uppdateringar och Notes finns i källkoden en drop-down lista som genererar data från databasen. För dessa drop-downs används ett bibliotek för att möjliggöra multipla val i listan som visas. Detta ramverk konverterar de select- option-element som finns i källkoden till input- och label-element. Dessa input och label-element är dock inte kopplade till varandra och genererar därför felmeddelanden i Wave som inte går att åtgärda.

På ett par ställen på plattformen finns det knappar som öppnar upp modalfönster, för att till exempel lägga till medlemmar eller skriva kommentarer.



Tüturu Dashboard – Utvecklingen av en plattform där skolor och vårdgivare kan mötas

Beatrice Björn

2023-06-02

På dessa knappar lades attributet `aria-labelledby` följt av `id:t` på den modal som öppnas till. Detta för att personer som inte ser skärmen ska veta att något sker på skärmen när knappen klickats. [26]

I den tidigare koden hade inga huvudkomponenter på plattformen lagts till. I stället användes `div`-element för att dela upp plattformen i olika sektioner. Det `div`-element som omger headern/bannern på plattformen byttes ut mot ett `header`-element och det `div`-element som omgav huvudnavigeringen för desktop byttes ut mot ett `nav`-element. Runt huvudinnehållet på plattformen lades ett `main`-element till. Dessa ändringar bidrar till förbättrad tillgänglighet för personer som inte ser skärmen. [27]

Wave genererade varningar om låg kontrast på två knappar på plattformen. Efter att den designer som jobbar för Springtimesoft lagt fram nya färger för knapparna applicerades dessa för att eliminera kontrastfelet.

## 4.6 Underhåll av publik webbplats

Springtimesoft har i en del av sina kontrakt skrivit på att regelbundet underhålla webbplatserna de utvecklat. Detta görs en gång i månaden och för att genomföra en underhållskontroll på Tüturus webbplats följdes en rad punkter. Först startas en `feature` branch med rätt ticket-nummer och `"sts sh composer install"` och `"sts tools make install"` kördes för att ladda ner de mjukvaror och verktyg som används för projektet.

Kommandot `"sts outdated"` kördes sedan för att kolla efter uppdateringar av mjukvaror och verktyg. Kommandot returnerar en lista med uppdateringar som finns tillgängliga. Listan kontrolleras för att se om någon av uppdateringarna verkar vara väldigt stor eller om de kan installeras relativt snabbt och enkelt. Om listan ser bra ut körs kommandot `"sts sh composer update"` och sedan `"sts tools npm update"` för att uppdatera både backend och frontend.

Kommandot `"tools make build"` ködes för att bygga webbplatsen så att den kunde testas lokalt och `"/dev/build"` lades till i URL:en för att göra eventuella uppdateringar i databasen. Sedan kontrollerades alla sökfält, formulär och länkar på webbplatsen så att de fungerar som de ska och en generell koll av webbplatsens olika delar gjordes för att kunna upptäcka eventuella buggar eller andra problem som måste åtgärdas

När webbplatsen kontrollerats lades listan med uppdateringar som gjorts som en kommentar på Jira-uppgiften. Eventuella andra kommentarer lades till och skickades in för att slutföra underhållskontrollen.

## 5 Resultat

Antalet artiklar som visas på sidorna Providers, Schools och Community of Practice ändrades från 9 till 15. Dessa sidor har även ändrats från att läsa ut data via SilverStripe till att läsa ut data via Vue.

Det går att sortera och filtrera skolor, providers och community of practice. Genom att skriva in ett sökord i ett sökfält returneras en lista med data där "Name" eller "Title" matchar/delvis matchar sökningen. Om ett sökord inte skulle ha några matchningar visas ett meddelande om detta på skärmen. Både den data som returneras som sökresultat och all data (ofiltrerad) går att sortera i ordning från A till Z, Z till A, från äldsta/ouppdaterade poster och senast uppdaterade/skapade poster.

Längst ner på startsidan listas 9 av de skolor som inte uppdaterats på en månad eller mer. Finns det fler än 9 skolor som inte uppdaterats visas en länk som vid klick tar besökaren till sidan som listar alla skolor och visar de skolor som inte uppdaterats på en månad eller mer. Om det finns färre skolor än 9 är ovannämnda länk inte synlig.

Den felformade fokus-ramen på sökfälten (både på plattformen och på den publika webbplatsen) har med hjälp av JavaScript korrigerats. Den nya ramen täcker nu hela sökfältet istället för att endast delvis rama in sökfältet.

All den testdata som behövdes för att kunna testköra plattformen har lagts till. Minst 20 providers, schools, notes, medlemmar och kommentarer har skapats.

Testverktyget Wave har använts för att göra plattformen mer tillgänglig. I den mån som varit möjlig har de fel och varningar som genererades av Wave eliminerats från plattformen.

En del av de fel som genererades när testverktyget Wave användes har inte åtgärdats. Detta beror på att felen uppstår via ett ramverk som konverterar en select-lista till input- och label-element. Dessa input- och label-element är inte kopplade till varandra och eftersom de inte finns i koden för plattformen har detta inte kunnat åtgärdas.

För att underhålla webbplatsen Tuturu har denna undergått en månadskontroll. Månadskontroller har följt den punktlista som finns tillgänglig och därmed säkerställt att webbplatsen fungerar som den ska och uppdateringar för källkoden har gjorts.

## 6 Analys

Målet med mitt arbete på springtimesoft var att utveckla specifik funktionalitet för plattformen Tüturu. Detta mål är i stora drag uppnått.

Det finns i nuläget fortfarande en del felmeddelanden från testverktyget Wave vad gäller det bibliotek som konverterar källkoden till input och label-element. Hur detta ska lösas är något som i nuläget diskuteras på Springtimesoft. Ett alternativ är att ta kontakt med de som utvecklat det bibliotek som används och be dem uppdatera koden så att label och input elementen länkas till varandra. Detta kan dock bli en långdragen process då det finns risk att ärendet inte kan prioriteras inom en snar framtid. Ett annat alternativ skulle kunna vara att byta och använda ett annat bibliotek. Detta skulle dock kräva stora ändringar i koden och vara ett tidskrävande moment. Ett tredje alternativ är att inte använda något bibliotek alls utan att göra en egen multi-select drop-down i ren HTML-kod [28].

Att ändra antalet skolor som inte uppdaterats på mer genom att sätta en gräns på 9 i kontrollern har vid testning av lösningen visat sig inte vara optimalt. Eftersom gränsen för antalet artiklar sätts redan i kontrollern innebär detta att endast 9 artiklar hämtas från databasen. Eftersom endast 9 artiklar hämtas kommer länken för att visa alla skolor aldrig att bli synlig eftersom denna endast blir synlig om det finns fler än 9 skolor att visa.

När jag testade lösningen hade jag endast tillgång till 5 skolor som inte uppdaterats på mer än en månad och insåg därför inte att detta var ett problem. Genom att i stället hämta in alla skolor via kontrollern och sedan sätta en gräns på 9 skolor i template filen har problemet lösts.

Under utvecklingens gång har vi även kommit över information om att "text-muted" inte längre stöds och att detta därför inte är ett bra alternativ för att skriva ut text i gråskala. Ett bättre alternativ för framtida projekt är att styla texten direkt i sass i stället för med Bootstrap direkt i html-koden.

Jag har även insett att "d-none" inte är ett bra alternativ för att dölja labels om de inte ska vara synliga på skärmen men läsbara för skärmläsare. När display:none appliceras innebär det att elementen även är dolda för skärmläsare vilket motverkar syftet med att lägga till ett label-element från första början. En bättre lösning för detta skulle vara att använda Bootstrap-klassen "visually-hidden". Detta gör att elementen inte är synliga på skärmen men fortfarande läsbara via skärmläsare [29].

Utvecklingen av plattformen i sin helhet är ännu inte färdig utan kommer att fortsätta tills den funktionalitet som krävs utvecklats och då kommer plattformen beta-testas av framtida användare.

## 6.1 Etisk och social diskussion

Att vårdgivare och skolor kan komma i kontakt med varandra via en plattform är positivt och kommer med många fördelar angående studenthälsa och kontinuitet inom studentvård.

Den data som lagras via på Tuturu dashboard (bortsett från persondata) är offentlig, detta innebär låg risk vad gäller hanteringen av datat i sig självt. Ett problem som kan uppstå är att offentliga data när den kombineras kan komma att klassas som konfidentiell data. Om till exempel en skola och en vårdgivare samarbetar och delar en mängd resurser skulle slutsatser kunna dras om vilka utmaningar just den skolan står inför.

GDPR gäller inte på Nya Zeeland, här finns i stället lagen Privacy Act 2020 som tillämpar lagar och regler som liknar GDPR [30] [31].

En av dessa regler gäller hur länge persondata tillåts lagras i en databas hos ett företag. Data får enligt lagen inte lagras längre än nödvändigt av ett företag. Denna regel kan ses som luddig då det är svårt att fastställa vad "längre än nödvändigt" betyder. Springtimesoft har som regel att spara data så länge den kan behövas för att bevisa att ett samarbete mellan två parter ägt rum.

Genom att regelbundet rensa databasen från data som inte längre är aktuella minskar risken med att stora mängder data läcker om ett intrång i databasen skulle äga rum.

Det finns på Nya Zeeland en hederskodex vad gäller hantering av persondata, denna beskriver vikten i att hantera data med stor vördnad och respekt samt att alla beslut gällande persondata ska övervägas med säkerhet i åtanke.

En annan viktig etisk aspekt med en plattform som Tuturu dashboard är att den måste vara tillgänglig och användarvänlig. Detta innebär inte bara tillgänglighet och användarvänlighet för personer med funktionsnedsättningar så som syn, hörsel eller kognitiv förmåga utan även att plattformen måste vara tillgänglig för personer med till exempel begränsad IT-kunskap eller långsam internetuppkoppling. För en plattform som Tuturu är detta extra viktigt då en stor del av användarna bor på den Nyzeeländska landsbygden och inte kan förlita sig på en stadig internetuppkoppling.

Det är svårt att kontrollera den information som delas på en plattform som Tuturu då plattformen tillåter användare att till exempel skriva kommentarer eller dela artiklar. Detta kan i värsta fall leda till diskriminering eller delning av extrema åsikter. Det kan bli svårt att kontrollera innehållet på plattformen och på så vis förebygga att till exempel rasism, sexism, homofobi eller andra former av diskriminering delas eller sprids på plattformen.

## 7 Slutsatser

Genom att ha utfört mitt examensarbete på Springtimesoft har jag fått tillgång till en stor mängd kunskap från mina kollegor. Jag har lärt mig mer än jag trodde var möjligt och ser fram emot att utnyttja den nyvunna kunskapen i framtida projekt.

Den största skillnaden med att arbeta för ett företag och i ett team har varot att projektet varit så mycket större än någonting annat jag utvecklat tidigare. Det tog många dagar för mig att lära mig navigera mellan alla filer och mappar.

I tidigare projekt har jag vant mig vid att jag kan göra ändringar i koden precis när jag vill. När jag nu arbetat i ett större projekt med fler utvecklare som jobbar med kod samtidigt har jag fått lära mig att endast utveckla den kod som finns listad i den jira-uppgift jag arbetar med för stunden. Ändringar i andra filer kan skapa buggar eller störa den kod som andra utvecklare skapat och skickat in. Detta har varit en stor utmaning och det har tagit tid att komma in i rätt tänk vad gäller detta.

Detsamma gäller designen på webbplatsen. Springtimesoft har en designer som designat plattformen. När jag testade plattformen i Wave uppstod en del varningar vad gäller kontrast och min första tanke var att ändra färgen så att felet skulle försvinna. Detta är dock inte rätt väg att gå utan jag behövde först kontakta designern, vänta på svar, testa hans nya förslag på färger och implementera dessa om det inte längre genererade några fel i Wave.

Det har varit lärorikt att arbeta tillsammans med andra utvecklare. Vi har haft många intressanta diskussioner om kod och olika lösningar på problem. Dessa diskussioner har bidragit till att min utvecklingskurva som webbutvecklare skjutit i höjden. Genom att diskutera och jämföra kod och olika lösningar har jag grävt i minnet och fått nyttja den kunskap jag ackumulerat under de senaste två åren som student.

Jag känner mig nöjd med vad jag åstadkommit med mitt projektarbete men framför allt är jag nöjd och stolt över den utveckling jag gjort på ett personligt plan, både under de två åren som student på Mittuniversitetet och under de senaste veckornas arbete på Springtimesoft.

## Källförteckning

- [1] Docker - <https://docs.docker.com/>  
Senast uppdaterad: 2023  
Hämtad: 2023-04-02
- [2] Git - <https://git-scm.com/>  
Senast uppdaterad: 2023-05-20  
Hämtad: 2023-04-02
- [3] Git Flow - <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>  
Senast uppdaterad: 2023-04-21  
Hämtad: 2023-04-22
- [4] Node.js: <https://nodejs.org/en/about>  
Senast uppdaterad: 2023  
Hämtad: 2023-05-07
- [5] Composer: <https://getcomposer.org/doc/00-intro.md>  
Senast uppdaterad: 2023  
Hämtad: 2023-05-07
- [6] PSR-2 - <https://www.php-fig.org/psr/psr-2/>  
Senast uppdaterad: 2023  
Hämtad: 2023-04-22
- [7] Jira:  
[https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=148251130767&campaign=19306728945&creative=641977782121&device=c&keyword=jira&matchtype=e&network=g&placement=&ds\\_kids=p74602247110&ds\\_e=GOOGLE&ds\\_eid=700000001558501&ds\\_e1=GOOGLE&gclid=CjwKCAjw6IiiBhAOEiwALNqncTq](https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=148251130767&campaign=19306728945&creative=641977782121&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p74602247110&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=CjwKCAjw6IiiBhAOEiwALNqncTq)

[v9hUvcQlcBJvkBZ4FmzFK3XCetwiB7UApZhgnuRYqgFXEQz\\_zSRoCDhYQAvD\\_BwE&gclid=aw.ds](https://v9hUvcQlcBJvkBZ4FmzFK3XCetwiB7UApZhgnuRYqgFXEQz_zSRoCDhYQAvD_BwE&gclid=aw.ds)

Senast uppdaterad: 2023

Hämtad: 2023-04-22

- [8] BootStrap: <https://getbootstrap.com/>

Senast uppdaterad: 2023

Hämtad: 2023-04-22

- [9] Git Lab: <https://about.gitlab.com/>

Senast uppdaterad: 2023

Hämtad: 2023-05-07

- [10] SilverStripe: <https://www.silverstripe.org/>

Senast uppdaterad: 2023

Hämtad: 2023-04-22

- [11] VSC Extentions: <https://marketplace.visualstudio.com/vscode>

Senast uppdaterad: 2023

Hämtad: 2023-05-20

- [12] Wave: <https://wave.webaim.org/>

Senast uppdaterad: 2023

Hämtad: 2023-05-20

- [13] SilverStripe SetPageLength:

[https://docs.silverstripe.org/en/4/developer\\_guides/templates/how\\_tos/pagination/](https://docs.silverstripe.org/en/4/developer_guides/templates/how_tos/pagination/)

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[14] PHP Limit() :

<https://pear.php.net/manual/en/package.database.db-dataobject.db-dataobject.limit.php>

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[15] Bootstrap focus: <https://getbootstrap.com/docs/5.1/forms/form-control/>

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[16] Focus and blur: <https://javascript.info/focus-blur>

Senast uppdaterad: 2022-06-19

Hämtad: 2023-05-20

[17] SilverStripe allowed\_actions: [https://docs.silverstripe.org/en/4/developer\\_guides/controllers/access\\_control/](https://docs.silverstripe.org/en/4/developer_guides/controllers/access_control/)

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[18] SilverStripe url\_handlers: [https://docs.silverstripe.org/en/4/developer\\_guides/controllers/routing/](https://docs.silverstripe.org/en/4/developer_guides/controllers/routing/)

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[19] Vue create app: <https://vuejs.org/guide/essentials/application.html>

Senast uppdaterad: 2023

Hämtad: 2023-05-20

[20] Vue .mount(): <https://masteringjs.io/tutorials/vue/mounted> Senast uppdaterad: 2023

Hämtad: 2023-05-20



- [21] Vue v-if: <https://vuejs.org/guide/essentials/conditional.html>  
Senast uppdaterad: 2023  
Hämtad: 2023-05-20
- [22] Vue v-bind: <https://vuejs.org/guide/essentials/class-and-style.html>  
Senast uppdaterad: 2023  
Hämtad: 2023-05-20
- [23] Vue v-bind :selected:  
<https://vuejs.org/guide/essentials/forms.html#value-bindings>  
Senast uppdaterad: 2023 Hämtad: 2023-05-20
- [24] W3C – headings: <https://www.w3.org/WAI/tutorials/page-structure/headings/>  
Senast uppdaterad: 2017-05-04  
Hämtad: 2023-05-28
- [25] W3C – labels: <https://www.w3.org/WAI/tutorials/forms/labels/>  
Senast uppdaterad: 2019-07-29  
Hämtad: 2023-05-28
- [26] W3C aria-labelledby: <https://www.w3.org/TR/WCAG20-TECHS/ARIA16.html>  
Senast uppdaterad: 2016  
Hämtad: 2023-05-28
- [27] W3C semantic HTML: <https://www.w3.org/TR/WCAG20-TECHS/G115.html>  
Senast uppdaterad: 2016  
Hämtad: 2023-05-28

- [28] Multi-select HTML: [https://www.w3schools.com/tags/att\\_select\\_multiple.asp](https://www.w3schools.com/tags/att_select_multiple.asp)

Senast uppdaterad: 2023

Hämtad: 2023-05-28

- [29] Visually-hidden Bootstrap: <https://getbootstrap.com/docs/5.0/helpers/visually-hidden/>

Senast uppdaterad: 2023

Hämtad: 2023-05-28

- [30] GDPR: <https://gdpr-info.eu/>

Senast uppdaterad: 2023

Hämtad: 2023-05-28

- [31] Privacy Act 2020: <https://www.privacy.org.nz/privacy-act-2020/privacy-principles/>

Senast uppdaterad: 2023

Hämtad: 2023-05-28