YRKESHÖGSKOLAN
NOVIA

Utveckling av samtyckeskomponent

Benjamin Thylin

Examensarbete för ingenjörs (YH)-examen

El- och automationsteknik

Vasa 2022

EXAMENSARBETE

Författare: Benjamin Thylin

Utbildning och ort: El- och automationsteknik, Vasa

Inriktning: Informationsteknik

Handledare: Susanne Österholm, Novia

Christoffer Björkskog, Genero Oy Ab

Titel: Utveckling av samtyckeskomponent

Datum: 16.3.2022 Sidantal: 21

Abstrakt

Detta examensarbete handlar om hur man utvecklar en samtyckeskomponent med hjälp av Stencil. Examensarbetet är utfört på uppdrag av Genero Oy Ab, som är en marknadsföringsbyrå som jobbar med growth hacking, innehållsskapande, sökmotoroptimering och webbutveckling. Genero har cirka 120 anställda och fyra kontor i Finland och Sverige.

Arbetets viktigaste uppgift var att starta en utveckling av en ny samtyckeskomponent för Genero. Komponenten skulle samstämma med alla de krav som ställs runt integritetspolicy. Kraven för den nya samtyckeskomponenten var att den skulle vara lättare att konfigurera och vidareutveckla vartefter nya krav fastställs. Komponenten skulle skapas för att användas på de flesta av Generos webbsidor och framtida webbsidor.

Programmeringsverktygsdelen i arbetet behandlar hur alla verktyg, programmeringsspråk och processer jobbar tillsammans för att komponenten skall fungera.

Resultatet av arbetet blev en första version av en tydlig och användarvänlig samtyckeskomponent. Samtyckeskomponenten implementerades på en av Generos kunders webbsida. I resultatet lämnades också tydliga mönster för hur komponenten skall vidareutvecklas.

Språk: svenska

Nyckelord: webbutveckling, samtyckeskomponent, webbkomponent, gdpr, integritetspolicy

BACHELOR'S THESIS

Author: Benjamin Thylin

Degree Programme: Electrical and automation engineering

Specialisation: Information technology

Supervisors: Susanne Österholm, Novia

Christoffer Björkskog, Genero Oy Ab

Title: Development of consent component

Date: 16.3.2022 Number of pages: 21

Abstract

This thesis is about how to develop a consent component with the help of Stencil. The thesis was commissioned by Genero Oy Ab, which is a marketing agency that works with growth hacking, content creation, search engine optimization, and web development. Genero has approx. 120 employees and four offices in Finland and Sweden.

The most important task of the thesis was to initiate the development of a new consent component for Genero that complies with all the requirements set around privacy policy. The requirements for the new consent component were that it would be easier to configure and further develop as new requirements are established. The component would be created for use on most of Genero's web pages and future web pages.

The programming tools part of the thesis deals with how all tools, programming languages and concepts work together and are needed for the component to work.

The result of the work was the first version of a clear and user-friendly consent component. The consent component was implemented on one of Genero's customers' websites. The result also provided clear patterns for how the component should be further developed.

Language: english

Key words: web-development, consent component, web component, gdpr, privacy policy

Innehållsförteckning

Definitionslista

Growth hacking	Ett samlingsnamn för strategier fokuserade enbart på tillväxt.
SEO	Förkortning av ”Search Engine Optimization”. En process som används för att förbättra en webbplats sökordsranking när man söker i tex. Google.
Copywriter	Skribent som skriver för marknadsföringsbyråer.
GDPR	EU:s dataskyddsförordning ”General Data Protection Regulation”.
GDS	Genero designsystem.
Tredje partens cookies	Cookies som ställs in från en annan webbsida än den man besöker.
Googles privacy sandbox	Ett initiativ av Google att skapa Webstandarder för att få tillgång till användares information utan att kompromissa på säkerheten.
Öppen källkod	Användare har rätt till att använda, läsa och vidaredistribuera koden som programmet använder.
Objektorienterad programmering	Ett sätt att dela upp ett program i mindre delar, så att det blir mer lätthanterligt. De mindre delarna kallas klasser.
Dynamiskt programmeringsspråk	Programmeringsspråk som kör funktioner under körning där statiska programmeringsspråk skulle köra funktionerna under kompilering.
Funktionell programmering	En programmeringsstil som använder sig av ett antal matematiska funktioner för att lösa ett problem.

Webbläsarextension	Ett litet mjukvarutillägg man kan installera till din webbläsare.
SCSS	CSS med tilläggsfunktioner för att skriva CSS effektivare och snabbare.

1 Inledning

Examensarbetet utfördes för Genero Oy Ab i Jakobstad. Företaget var i behov av en ny samtyckeskomponent för att uppfylla de krav som finns på integritetspolicy.

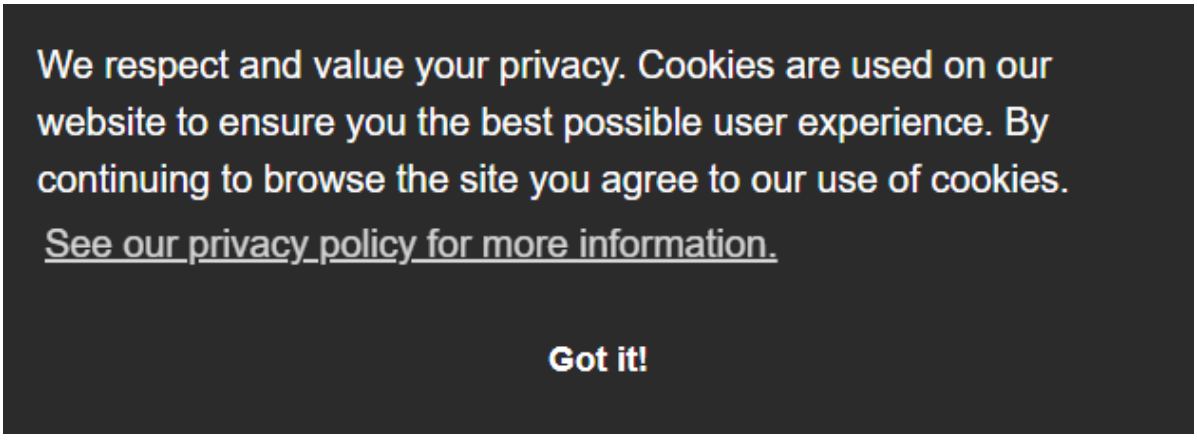
1.1 Uppdragsgivare

Genero grundades 2009 av Rasmus Östman, Sebastian Östman och Jonathan Björkskog vid sidan om deras studier i Helsingfors. Genero är en online-marknadsföringsbyrå som jobbar med growth hacking, SEO, reklamvideor, copywriting och webbutveckling. De har två kontor, ett i Jakobstad och ett i Helsingfors. Genero är idag ett dotterbolag till A-lehdet. 2021 hade Genero över 100 anställda och Genero har hjälpt över 100 företag att växa och synas. (Genero, 2021).

1.2 Bakgrund

Genero har skapat många hemsidor sedan 2009, och end del av dem underhålls än idag. På alla hemsidor som Genero har gjort så används en samtyckeshanteringskomponent.

Fram till 2021 har Genero använt en relativt simpel sådan, där det frågats ifall man samtycker till att webbplatsen använder cookies. (Se figur 1).

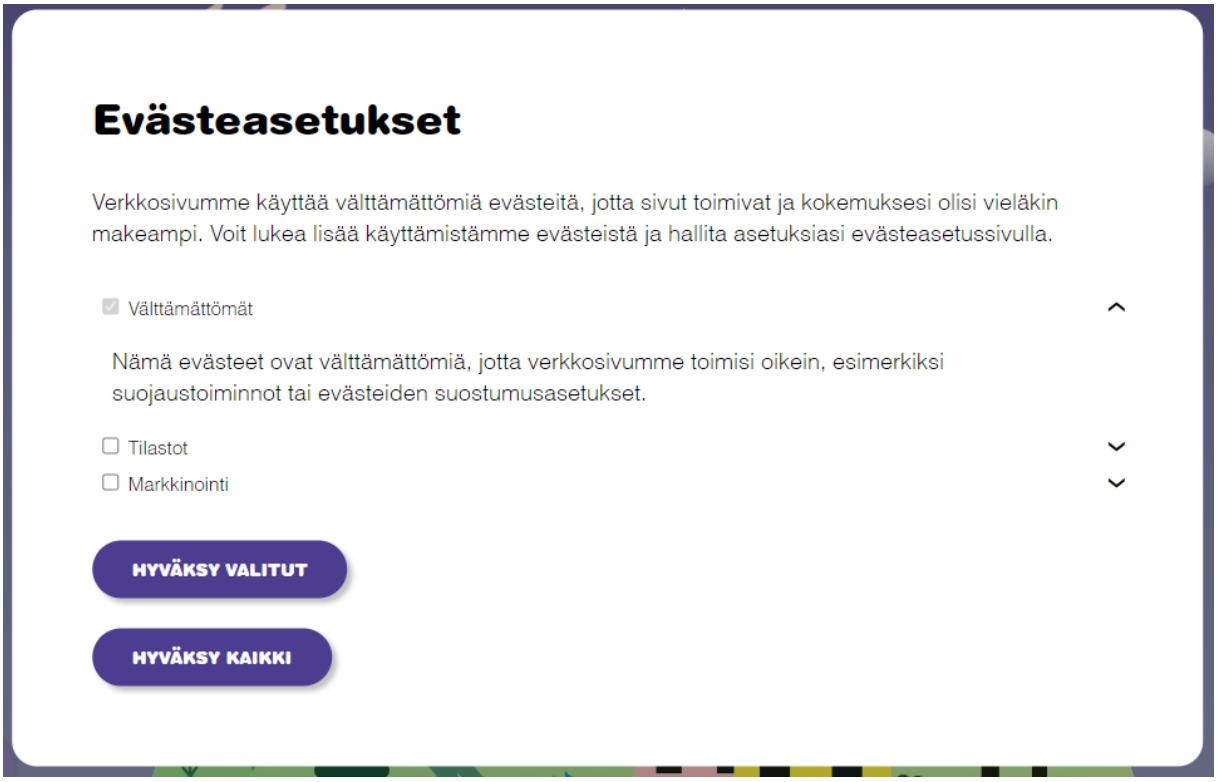


Figur 1. Den gamla samtyckesförfrågningskomponenten som används på Generos hemsida.

Google publicerade i slutet av 2020 en bloggpost där de meddelade att Chrome skulle sluta att stöda tredje partens cookies, då bestämde sig Genero för att skapa en ny samtyckeshanteringskomponent (Goel, 2021). Den nya plattformen skulle vara i linje med Googles privacy sandbox. Den skulle också vara lättare att konfigurera beroende på olika webbsidors samtyckeshanteringsplan.

1.3 Uppgift

Uppgiften var att skapa en samtyckeshanteringskomponent som är lätt att konfigurera, så att kunden kan ändra texterna och samtyckesförfrågningar. På samma gång skulle komponenten uppfylla GDPR-kraven. En tydlig skillnad på den gamla samtyckeshanteringskomponenten och den nya, är flexibiliteten över vad användaren vill godkänna. Komponenten skulle skapas till Generos öppna källkods komponentbibliotek GDS (Genero, 2021). Komponenten kodas med TypeScript med hjälp av Stenciljs.



Figur 2. Den nya samtyckeshanteringskomponenten i användning på en kunds hemsida.

2 Programmeringsverktyg

I detta kapitel beskrivs vilka programmeringsspråk och verktyg som används för att skapa samtyckeskomponenten. När man skapar webbkomponenter är det svårt att inte komma i kontakt med JavaScript i någon form. Som tillägg till JavaScript användes TypeScript och Stencil. I och med att komponenten är en webbkomponent användes HTML och CSS för rendering och design.

2.1 JavaScript

JavaScript är ett dynamiskt programmeringsspråk baserat på programmeringsspråken Java och C. JavaScript stöder objektorienterad programmering och funktionell programmering (MDN, u.d.). Enligt Stackoverflows årliga undersökning 2021, så är JavaScript det mest använda programmeringsspråket för nionde året i rad (Stack Overflow, 2021).

2.1.1 Historia

JavaScript skapades 1995 av Brendan Eich när han jobbade som ingenjör på Netscape. Programmeringsspråket skulle egentligen heta LiveScript, men på grund av ett marknadsföringsbeslut så gavs det i stället namnet JavaScript. Detta för att utnyttja det populära Sun Microsystems programmeringsspråket Java, även fast dessa språk har väldigt lite gemensamt. När JavaScript växte, behövdes en standardisering för språket, och i och med det lämnade Netscape in JavaScript till en europeisk standardiseringsorganisation vid namnet Ecma international. (MDN, u.d.). Ecma internationals standardiserade JavaScript och har sedan dess släppt nya versioner för att hålla språket vid liv och utvecklande.

- ECMAScript Edition 1, 1997
- ECMAScript Edition 2, 1998
- ECMAScript Edition 3, 1999
- ECMAScript Edition 5, 2009
- ECMAScript Edition 6, 2015
- ECMAScript Edition 7, 2016

- ECMAScript Edition 8, 2017
- ECMAScript Edition 9, 2018
- ECMAScript Edition 10, 2019
- ECMAScript Edition 11, 2020
- ECMAScript edition 12. (International ECMA, 2021).

2.1.2 Användningsområden

JavaScript är designat för att köras i en värdtjänst. Det vanligaste är att köra JavaScript i en webbläsare, men kan också köras i andra värdtjänster, så som Adobe Acrobat, Adobe Photoshop, SVG bilder, server-side-miljöer som Node.js, NoSQL, inbyggda system, hela skrivbordsapplikationer med mera. (MDN, u.d.).

2.1.3 Svagheter

En av utmaningarna med att använda JavaScript är att variabler i JavaScript inte kräver någon typspecifikation. I och med det kan ett stort JavaScript projekt bli utmanande att felsöka. Fram till 2015 hade JavaScript endast en variabeldeklaration, *var*, som inte har någon specifik räckvidd. 2015 när ECMAScript 6 släpptes, lanserades *let* och *const* variabeldeklarationerna. *let* och *const* begränsar variabelns räckvidd till närmaste *parent block*. *const* används när variabeln inte är avsedd att ändras. När man använder *let* och *const* blir JavaScript lite mindre förlåtande, men som andra programmeringsspråk så underlättar det vid större projekt mera specifika typdeklarationer för variabler. (MDN, u.d.).

Kodexempel 1. Demonstration över hur var variabler fungerar och deklareras.

```
var num1 = 1;
function varDeclaration() {
    var num2 = 2;
    if (num2 > num1) {
        var num3 = 3;
        num3++;
    }
    while(num1 < num2) {
        var num4 = 4;
        num1++;
    }
    console.log(num1); //2
    console.log(num2); //2
    console.log(num3); //4
    console.log(num4); //4
}
varDeclaration();
```

Som i kodexempel 1 deklareras variablerna både innanför och utanför funktionen utan errors när variablerna anropas inuti funktionen. Ifall vi skulle använda ”let” variablerna i stället för ”var” variabler så hade vi fått ett felmeddelande på variablerna ”num3” och ”num4”.

Kodexempel 2. Demonstration över hur let variabler fungerar och deklareras.

```
let num1 = 1;

function letDeclaration() {
    let num2 = 2;

    if (num2 > num1) {
        let num3 = 3;
        num3++;
    }

    while(num1 < num2) {
        let num4 = 4;
        num1++;
    }

    console.log(num1); //OK
    console.log(num2); //OK
    console.log(num3); //Compiler Error: Cannot find name 'num3'
    console.log(num4); //Compiler Error: Cannot find name 'num4'
}

letDeclaration();
```

För att undvika dessa errors och bli varnad i ett tidigt skede så underlättar det att man använder TypeScript istället för JavaScript.

2.2 TypeScript

TypeScript är ett typbaserat programmeringsspråk som byggs på JavaScript. Med TypeScript finns möjligheten att skapa typspecifika variabler så som string, number, boolean, undefined, function, array med mera. Fördelen med TypeScript är att det påpekar inkonsekventa beteenden under felsökning och testning. (Typescript, Typescript Handbook - Typescript in 5 minutes, 2022). TypeScript har alla funktioner som JavaScript plus TypeScripts variabeltypsystem. De vanligaste buggarna och fel i JavaScript är variabeltypfel, vilket betyder

att en variabeltyp används på ställen där en annan variabeltyp förväntas. TypeScript fungerar som ett verktyg som körs och kollar efter variabeltypfel innan koden kompileras till JavaScript. (Typescript, Typescript Handbook - Intro, 2022).

2.3 Stencil.js

Stencil är skapat av Ionic Framework. Ionic använde många traditionella ramverk innan de skapade Stencil. De hittade inget ramverk som fungerade på olika plattformar eller var genererade komponenter som renderades snabbt på alla nätverksuppkopplingshastigheter.

Stencil är ett verktyg för att generera webbkomponenter för designsystem. Fördelarna med att använda Stencil i stället för andra verktyg är att Stencil använder webbkomponentstandarder, vilket betyder att alla komponenter som görs i Stencil stöds av alla moderna webbläsare. Stencil är också implementerbar med alla front-end ramverk. I och med det fungerar Stencil med olika projekt och stacks. Stencil fungerar alltså även för företag som använder olika ramverk för sina projekt. (Thomas, u.d.).

2.4 HTML

HTML är en förkortning av ”Hypertext markup language”. Det är standardspråket för att skapa webbsidor, det är alltså inte ett programmeringsspråk. Det är ett sätt att beskriva strukturen för webbläsaren om hur webbsidan ska visa dess innehåll. HTML underhålls och utvecklas av World Wide Web Consortium.

HTML använder taggar för att berätta åt webbläsaren på vilket sätt innehållet ska visas. Taggar definierar element som renderas på webbsidan.

Ett exempel på en webbsida som har en titel, en paragraf med en länk och en knapp kan man skriva enligt kodexempel 3.

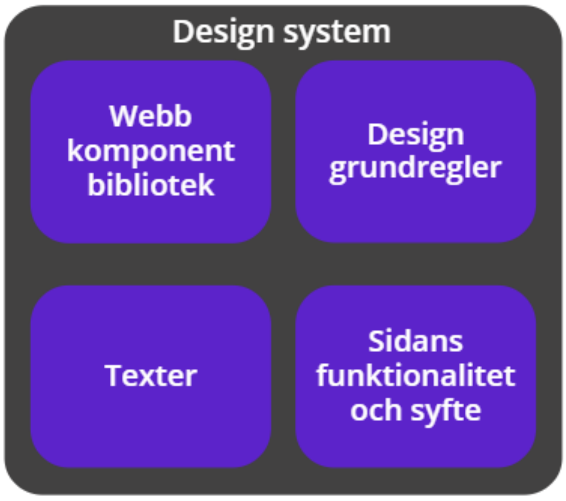
Kodexempel 3. HTML exempelkod.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Det här är titlen för webbsidan</title>
  </head>
  <body>
    <h1>Det här är en huvudrubrik</h1>
    <p>Det här är en paragraf med en <a href=”https://novia.fi/”>länk</a></p>
  </body>
</html>
```

3 Designsystem

Ett designsystem är en bro mellan design och komponenter. Ett designsystem består av komponenter, komponentstrukturer och en visuell stil. Genom att använda komponenterna tillsammans med komponentmönstren och den visuella stilen, får man en enhetlig och flexibel webbdesign. Med hjälp av ett designsystem har utvecklare och designers möjligheten att arbeta enhetligt tillsammans i stället för att behöva skapa samma komponenter om och om igen. (Netkow, Matt, u.d.).

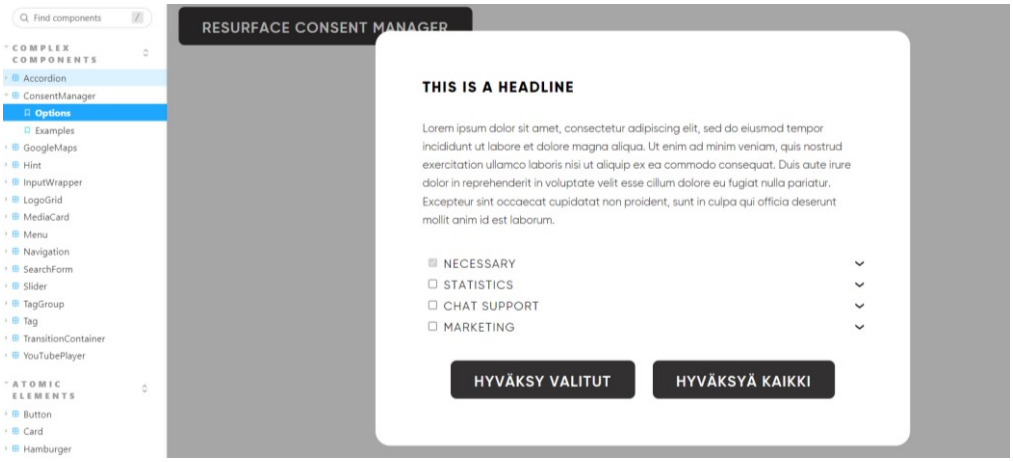
Exempel på designsystem är Googles ”Material Design System”, Apples ”Human Interface Guidelines”, Microsofts ”Fluent Design System” och Adobes ”XD”.



Figur 3. Innehållet av ett designsystem.

3.1 Webbkomponentbibliotek

Webbkomponentbibliotek och designsystem är inte samma sak, men kan lätt förväxlas med varandra. Ett designsystem innehåller alla komponenter ur komponentbiblioteket, dit kan designers injicera stilar för att nå den känsla och stil man önskar för projektet. Webbkomponentbibliotek som görs med Stencil har den fördelen att komponenterna går att användas med olika front-end ramverk. (Thomas, u.d.).



Figur 4. Samtyckeskomponenten i komponentbiblioteket.

4 Cookies

Cookies är ett sätt att lagra personliga data om användare för att förbättra besöksupplevelsen på en webbsida, samt för att samla data från användare, relaterat till marknadsföring. Cookies lagras minimalistiskt i textfiler på webbservern och i användarens webbläsare för att använda dem i framtiden för att till exempel visa vilka varor användaren har tittat på i en webbutik. I detta fall vilka samtycken användaren har godkänt.

När man skriver in en exempel.fi i webbläsaren händer följande:

1. Webbläsaren skickar en förfrågan till exempel.fi server, webbläsaren frågar efter exempel.fi framsida.
2. Webbläsaren kontrollerar efter en cookiefil på användarens dator som tillhör exempel.fi. Om en cookie med till exempel samtycken hittas, så skickas den till exempel.fi server.

3. När exempel.fi server tar emot cookien så svarar exempel.fi server med framsidan utan att visa ett samtyckesfönster. Då visas webbsidan för användaren.

5 General Data Protection Regulation

I maj 2018 trädde den striktaste integritets- och säkerhetslagen i kraft, General Data Protection Regulation eller mera känt som GDPR. Även om lagen utarbetades och godkändes av Europeiska unionen så ställer det krav på alla organisationer som riktar in sig på eller samlar in datarelaterade till människor i EU. GDPR är ett sätt för EU att visa sin starka ståndpunkt på medborgares integritet och säkerhet. (Wolford, u.d.).

5.1 Historia

På samma gång som Internet blev vanligare i alla hushåll, märkte EU att de behövde moderna skyddslagar för dataanvändning. År 1995 godkändes "Europaparlamentets och rådets direktiv om skydd för enskilda personer med avseende på behandling av personuppgifter och om det fria flödet av sådana uppgifter", där de fastställde minimistandarder för datasekretess och säkerhet.

GDPR verkställdes 2016 efter att ha godkänts av Europaparlamentet från och med 2018 var alla organisationer skyldiga att följa reglerna.

GDPR grundar sig på Europeiska konventionen om skydd för de mänskliga rättigheterna, som säger: "Var och en har rätt till skydd för sitt privat- och familjeliv, sitt hem och sin korrespondens". Detta är grunden till Europeiska unionens försök att säkerställa skyddet av sina medlemmars rättigheter. (Wolford, u.d.).

5.2 Regler

GDPR innehåller generella regler och ospecifika regler, men riktlinjerna är enligt följande. (Tietosuojavaltuutetun toimisto, u.d.).

5.2.1 Lagenlighet, korrekthet och transparens

Vid behandling av personuppgifter kan följande grunder vara lagenliga:

- ett samtycke av den registrerade
- ett avtal
- en rättslig förpliktelse för den personuppgiftsansvarige
- skydd av vitala intressen
- en uppgift som gäller ett allmänt intresse eller offentliga makt eller
- ett berättigat intresse hos den registeransvariga eller en tredje part.

Informationen som delges om hur personuppgifterna behandlas bör vara tydliga och givna på ett begripligt sätt.

Den redigerade skall informeras om:

- vilka uppgifter som samlas in om honom eller henne
- varför hans eller hennes personuppgifter behandlas
- hur hans eller hennes personuppgifter behandlas
- vilka rättigheter han eller hon har.

5.2.2 Bundenhet till användningsändamålet

Vid databehandling behöver syftet med personuppgifts insamling och behandling planeras och fastställas innan behandlingen inleds. Under behandlingen får uppgifterna användas och samlas enbart för ett uttryckt och lagligt syfte, uppgifterna får inte senare behandlas på ett sätt som inte har blivit specificerat i planeringen.

5.2.3 Uppgifts- och lagringsminimering

Under insamlingen och behandlingen är det inte tillåtet att samla in uppgifter som inte är nödvändiga för användningsändamålet. Personuppgifter får inte sparas för säkerhets skull, lagringstiden bör hållas så kort som möjligt. Den ansvarige för personuppgifterna ska kunna motivera lagringstiden och bör följa lagstiftningen om lagringstider.

5.2.4 Uppgifternas korrekthet

Den som är ansvarig för personuppgifter ska säkerställa uppgifternas korrekthet. Den redigerade ska ha möjlighet att begära om korrigering och radering av felaktiga eller onödiga uppgifter.

5.2.5 Konfidentialitet och säkerhet

Den ansvarige ska se till att personuppgifterna behandlas på ett konfidentiellt och säkert sätt. Risk- och dataskyddsåtgärder ska vara proportionerliga enligt risknivåerna.

6 Utförande

Innan man börjar utveckla ett projekt är det bra att definiera vad projektet skall innehålla. Med samtyckeskomponenten betydde det att skapa så kallade interfaces för komponenten. Interfaces är en definition av hur ett objekt är uppbyggt. Genom att definiera delar av komponenten på ett och samma ställe med interfaces, undviker man bland annat att i misstag feldefiniera variabler.

6.1 Uppbyggnad

När komponenten planerades, deklarerades även delar för framtida utvecklingsmöjligheter. Versionsnummer och språkvalmöjligheter utvecklades inte i den första versionen av komponenten.

6.1.1 Komponenten

Resultatet av vad komponenten skulle innehålla var följande:

- Versionsnummer, Versionsnumrets uppgift är att verifiera att användaren har godkänt de senaste samtyckesvalen som har definierats på webbsidan.
- Möjlighet att välja språk. Ifall en webbsida har flera olika språk behöver möjligheten finnas att välja språk redan i samtyckeskomponenten. Detta för att komponenten

döljer allt annat på sidan och komponenten är det första som syns när du landar på en webbsida.

- En rubrik.
- En knapp för att ändra samtycken. Samtyckesvalen visas inte innan man trycker på knappen ”Ändra samtycken”.
- En knapp för att godkänna valda samtycken. Knappen syns inte innan man har tryckt på knappen ”Ändra samtycken”. Efter att ha valt de samtycken man vill godkänna trycker man på knappen ”Godkänn valda samtycken”.
- En knapp för att godkänna alla samtycken. Ifall man godkänner alla samtycken trycker man på knappen ”Godkänn alla samtycken”.
- En lista innehållande alla samtycken. Denna lista blir synlig när man trycker på knappen ”Ändra samtycken”.

Kodexempel 5. Interface för samtyckeskomponenten

```
export interface CookieNoticeSetting {
   version: string
   language: string
   heading: string
   buttonEdit: string
   buttonAcceptSelected: string
   buttonAcceptAll: string
   consents: Array<Consent>
}
```

6.1.2 Samtyckesvalen

Samtyckesvalen skulle innehålla följande:

- En unik id. För att skilja på samtycken i koden.
- En rubrik.
- En beskrivning. För att förklara vad samtycket innebär.
- Kravstatus, ifall samtyckesvalet är ett måste. Samtyckesval som kan sättas som måsten kan vara till exempel att sidan behöver spara informationen om vilka samtycken användaren har godkänt.

– En indikation på ifall samtyckesvalet har blivit valt. Dessa val sparas på en webbserver som en cookie. Processer startas beroende på vilka samtycken användaren har godkänt.

Kodexempel 6. Interface för samtyckesvalen.

```
interface Consent {
  id: string
  label: string
  description: string
  necessary: boolean
  consent?: boolean
}
```

6.1.3 Komponentens funktionalitet

Komponenten namngavs enligt formatet som används för alla komponenter i Generos komponentbibliotek. Varje namn eller tag namnges enligt följande: *gds-* och sedan blockets namn, därmed namngavs samtyckeskomponenten till *gds-consent-manager*.

Kodexempel 7. Deklaration av tag-namn och SCSS-fil.

```
@Component({
 tag: 'gds-consent-manager',
 styleUrl: 'gds-consent-manager.scss',
 shadow: true,
})
```

När komponenten är definierad enligt kodexempel 7, implementeras komponenten till webbsidan genom att använda HTML-taggen *<gds-consent-manager></gds-consent-manager>.* Komponenten hämtar sitt utseende från filen *gds-consent-manager.scss*.

Innan komponenten renderas på sidan körs funktionen *componentWillLoad*, i funktionen händer följande:

– Default språket laddas in

- Komponenten kollar ifall en samtyckes cookie finns sparad i användarens webbläsare. Ifall webbläsaren har en cookie sparad så renderas inte komponenten.
- Komponenten laddar in de definierade samtycken till komponentens variabler.

Kodexempel 8. Funktionen ComponentWillLoad.

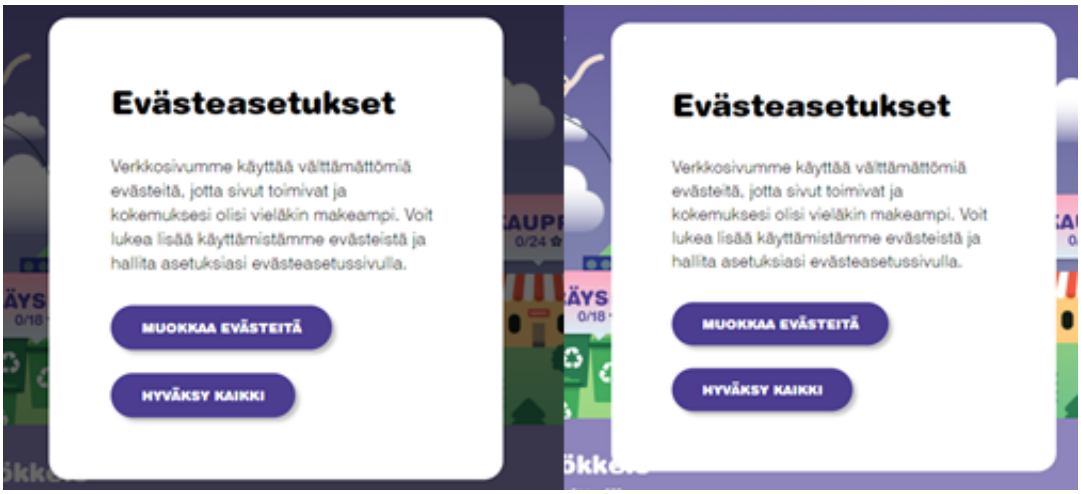
```
componentWillLoad() {
  // Parsing content from frontend
  this.configMap = JSON.parse(this.configs)
  // Selecting correct language
  this.settings = this.configMap[this.language]
  this.configMap[this.language]
  const consentString = this.getCookie('gds-consent')
  if (consentString) {
    // Emits event if consent has been given earlier
    this.runEvent()
    // TODO: Request consent if settings.version and cookie version don't match
    this.isopen = false
    let consentValues = consentString.split(',')
    consentValues.shift() // Removes version number
    this.settings.consents.forEach((consent, i) => {
    consent.consent = consentValues[i] === '1'
  })
}
```

För att bättre förstå komponentens funktionalitet behöver vi titta på komponentens HTML struktur i kodexempel 9. Man ser också i kodexempel 9 att komponenten har en div-tag som omsluter hela komponenten. Innan div-taggen renderas kollar programmet ifall variabeln *isopen* har värdet *true* eller *false*. Ifall komponenten i ett tidigare skede har noterat att användaren har en cookie sparad i sin webbläsare som säger vilka samtycken som har blivit godkända, då är *isopen* variabeln *false* och komponenten göms. Om *isopen* variabeln är *true* renderas komponenten.

Kodexempel 9. Kontrollerar variabeln isopen.

```
<div class={
  // Kollar ifall variabeln isopen är ”true” eller ”false”
  (this.isopen ? 'gds-cm-wrapper is-open' : 'gds-cm-wrapper') || {
  languageNavigation : this.languageNavigation
  }} >
    <div class="gds-cm-overlay">
      <div class="gds-cm">
```

En annan div-tagg som också omsluter den synliga delen av komponenten är taggen med klassen *gds-cm-overlay*. Det elementet skymmer resten av sidan, och på så sätt får användarna känslan av att de först måste svara på samtyckesförfrågningen. (Se Figur 6).



Figur 6. Komponenten med och utan gds-cm-overlay taggen.

Rubriken och beskrivningen laddas från variabler som har blivit tilldelade dess värde innan komponenten renderas. (Se Kodexempel 10 och 11).

Kodexempel 10. HTML för rubriken.

```
<slot name="headline">
  {this.headline && (
    <gds-heading size="s" class="headline">
      {this.headline}
    </gds-heading>
  )}
</slot>
```

Kodexempel 11. HTML för beskrivningen.

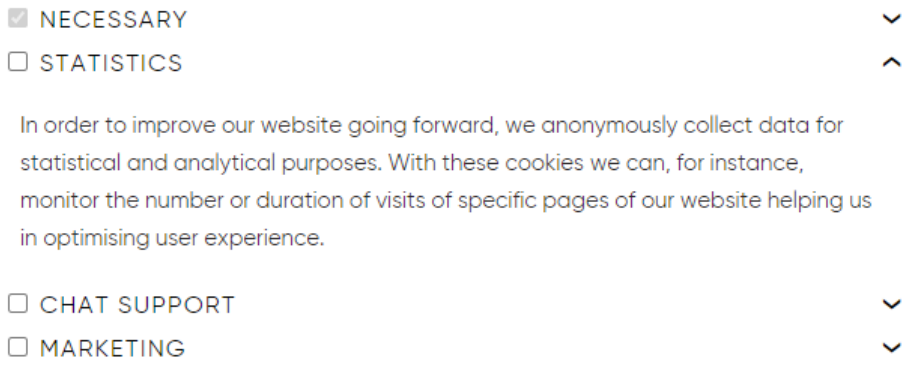
```
<slot name="description">
  {this.description && (
    <gds-paragraph size="l" class="description">
      {this.description}
    </gds-paragraph>
  )}
</slot>
```

Samtycken renderas genom en map-funktion som tar alla samtyckes objekten som har blivit deklarerat och renderar följande HTML element. (Se Kodexempel 12).

Kodexempel 12. HTML för samtycken rendering.

```
{this.settings.consents.map((consent) =>
  <gds-accordion>
    <gds-label slot="label" size="l" onClick={ (event) => event.stopPropagation() }>
      <input
        onClick={ () => this.toggleConsent(consent) }
        type="checkbox"
        disabled={ consent.necessary }
        checked={ consent.necessary || consent.consent}
      />{consent.label}
    </gds-label>
    <div slot="icon-collapse" class="icon iconCollapse">
      ❮
    </div>
    <div slot="icon-expand" class="icon iconExpand">
      ❯
    </div>
    <div slot="content">
      <gds-paragraph size="s" class="gds-cm-accordion-content">
        {consent.description}
      </gds-paragraph>
    </div>
  </gds-accordion>
)}
```

Komponenten renderar endast rubrikerna för samtycken men när man klickar på pilarna till höger om rubriken visas en beskrivning på vad samtycken betyder. Den typen av HTML-element kallas för Accordion. När man klickar på rubrikerna eller direkt på checkboxen markeras checkboxen och samtyckesvalet sparas.



Figur 7. Accordion blocken som visar samtyckesvalen.

Som sista elementen i komponenten renderas knapparna. Knappen ”Ändra samtycken” ändrar till ”Godkänn valda samtycken” och samtyckesvalen visas när knappen blir tryckt. När

man trycker på ”Godkänn valda samtycken” eller ”Godkänn alla samtycken” försvinner samtyckes komponenten och valen blir sparade i en cookie.

Kodexempel 13. HTML för knapparnas rendering.

```
<div class="footer">
   <gds-button onClick={
      this.accordionIsOpen ? () =>
      this.acceptSelectedCookies() : () =>
      this.toggleAccordions()
   } size="m">
      { this.accordionIsOpen ?
      this.settings.buttonAcceptSelected :
      this.settings.buttonEdit }
   </gds-button>
   <gds-button onClick={
      () => this.acceptAllCookies()
   } size="m">
      {this.settings.buttonAcceptAll}
   </gds-button>
</div>
```

När man trycker på ”Godkänn valda samtycken” körs funktionen *acceptSelectedCookies* (Se Kodexempel 14). Funktionen kollar vilka samtycken som har blivit valda och skapar en string med valen som sen blir skickat till funktionen *setCookie* (Se Kodexempel 16).

Kodexempel 14. Funktionen acceptSelectedCookies.

```
acceptSelectedCookies() {
   const consentValues = this.settings.consents.map(({consent}) => consent ? 1 : 0)
   const consentString = '1,' + consentValues.join(',')
   this.setCookie('gds-consent' , consentString)
   console.log(consentString)
   this.runEvent()

   this.isopen = false
}
```

När man trycker på ”Godkänn alla samtycken” körs funktionen *acceptAllCookies* (Se Kodexempel 15). Funktionen markerar alla samtycken som godkända och skapar en string med valen som sen blir skickat till funktionen *setCookie* (Se Kodexempel 16). Både *acceptAllCookies* och *acceptSelectedCookies* funktionerna kör funktionen *runEvent*, som skapar en *event*

händelse som senare kan användas för processer som inte får startas innan samtyckeskomponenten har visats för användaren.

Kodexempel 15. Funktionen acceptAllCookies.

```
acceptAllCookies() {
   const consentValues = this.settings.consents.map(({consent}) => consent ? 1 : 1)
   const consentString = '1,' + consentValues.join(',')
   this.setCookie('gds-consent' , consentString)
   this.runEvent()

   this.isopen = false
}
```

Funktionen *setCookie* tar den string som har skickats till funktionen, skapar ett utgångsdatum och skapar en cookie som innehåller samtyckesvalen. (Se Kodexempel 16).

Kodexempel 16. Funktionen setCookie.

```
 setCookie = (key, value) => {
   const domain = window.location.hostname.split('.').splice(-2).join('.')
   var expires = this.getDate13MonthsFromNow()
   document.cookie =
      key + '=' + value + ';
      domain=' + domain + ';
      path=/; SameSite=None; Secure;
      expires=' + expires.toUTCString()
}
```

6.2 Indata till komponenten

Komponenten blir matad med ett TypeScript-objekt som innehåller följande data:

- Språkkod (ISO 639-1).
- Språk.
- Text för ”Godkänn alla” knappen.
- Text för ”Ändra samtycken” knappen.
- Text för ”Godkänn valda” knappen.
- ID, rubrik, beskrivning och kravstatus för alla samtycken.

All data matas in hårdkodat i webbsidans källkod men för framtida versioner finns möjligheten att uppdatera all data via ett mer lättillgänglig backend system, som till exempel Wordpress.

Kodexempel 17. Exempelobjekt som matas in till komponenten.

```
const configMap = {
  FI: {
    languageCode: 'FI',
    language: 'Finnish',
    buttonAcceptAll: 'Hyväksyä kaikki',
    buttonEdit: 'Muokkaa evästeitä',
    buttonAcceptSelected: 'Hyväksy valitut',
    consents: [
     {
      id: 'consent-necessary',
      label: 'Necessary',
      description: 'These cookies are technically required for our core website to work properly, e.g.
      security functions or your cookie consent preferences.',
      necessary: true,
      consent: true,
     },
     {
      id: 'consent-statistics',
      label: 'Statistics',
      description: 'In order to improve our website going forward, we anonymously collect data for
      statistical and analytical purposes. With these cookies we can, for instance, monitor the
      number or duration of visits of specific pages of our website helping us in optimising user
      experience.',
      necessary: false,
     },
     {
      id: 'consent-chat',
      label: 'Chat Support',
      description: 'We want to make it as easy as possible for you to reach our support teams, e.g. via
      our chat box, which requires certain cookies.',
      necessary: false,
     },
     {
      id: 'consent-marketing',
      label: 'Marketing',
      description: 'These cookies help us in measuring and optimising our marketing efforts.',
      necessary: false,
     },
    ]
  }
}
```

7 Diskussion

Examensarbetets huvudsyfte var att skapa en första version av en ny samtyckeskomponent till Genero. Komponenten fungerar och ser ut som det var planerat. Komponenten togs i användning på två webbsidor vid komponentens första versions lansering. Med hjälp av komponenten kan Genero erbjuda kunder en lite bättre hemsida och försäkra användare att deras data inte används på ett vårdslöst sätt.

Arbetet har också öppnat upp mera diskussioner om integritetspolicy. På samma gång som företaget växer blir ämnet viktigare hela tiden.

Komponenten utvecklades så att den håller de integritetspolicykrav som uppkommer. Vidareutveckling görs också på de delar som kunderna önskar att blir ändrade på komponenten.

7.1 Utvecklingsmöjligheter

Det som har blivit planerat att bli utvecklat på komponenten för inkommande år:

- En språkmeny ifall webbsidan innehåller flera språk, det är en viktig uppdatering i och med att komponenten för tillfället inte går att använda på sidor med mera än ett språk.
- Möjligheten att ändra på texten utan att ha tillgång till källkoden. Tanken är att skapa en flik i Wordpress backend-system där man kommer åt texterna som renderas på komponenten.
- Fortsatt undersökning av kraven för en samtyckeskomponent. Som konstaterats i kapitel 5.2 är GDPR-reglerna relativt ospecifika och bör därför också regelbundet kontrolleras upp och uppdateras.

8 Källförteckning

Cookiebot. (u.d.). *Google third party cookies*. Hämtat från https://www.cookiebot.com/en/google-third-party-cookies/

Europaparlamentet och rådet. (den 23 November 1995). Hämtat från eur-lex.europa.eu: https://eur-lex.europa.eu/legal-content/SV/TXT/PDF/?uri=CELEX:31995L0046&from=EN

Europeiska domstolen för de mänskliga rättigheterna. (u.d.). Hämtat från echr.coe.int: https://www.echr.coe.int/Documents/Convention_SWE.pdf

Genero. (den 22 6 2021). *Genero design system repository*. Hämtat från Github: https://github.com/generoi/genero-design-system

Genero. (2021). *Genero digital*. (Genero) Hämtat från https://generogrowth.com/sv/about-us/

Goel, V. (6 2021). *Google blog*. (Google) Hämtat från https://blog.google/products/chrome/updated-timeline-privacy-sandbox-milestones/

Google. (u.d.). *The Privacy Sandbox*. Hämtat från The Privacy Sandbox: https://privacysandbox.com/

International ECMA. (6 2021). *ECMA-262*. (ECMA international) Hämtat från https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

MDN. (u.d.). *Web Docs*. Hämtat från https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

Netkow, Matt. (u.d.). *What is design system*. (Ionic) Hämtat från https://stenciljs.com/docs/what-is-design-system

Stack Overflow. (2021). *Developer Survey Results*. Hämtat från https://insights.stackoverflow.com/survey/2021

The PHP Group. (u.d.). *What is PHP?* Hämtat från https://www.php.net/manual/en/intro-whatis.php

Thomas, J. (u.d.). *Dokumentation*. Hämtat från StencilJs: https://stenciljs.com/docs/introduction

Tietosuojavaltuutetun toimisto. (u.d.). *Dataskyddsprinciper*. Hämtat från tietosuoja.fi: https://tietosuoja.fi/sv/dataskyddsprinciper

Typescript. (den 24 1 2022) A. *Typescript Handbook - Intro*. (Typescript) Hämtat från https://www.typescriptlang.org/docs/handbook/intro.html

Typescript. (den 24 1 2022) B. *Typescript Handbook - Typescript in 5 minutes*. (Typescript) Hämtat från https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

Wolford, B. (u.d.). *What is GDPR*. Hämtat från gdpr.eu: https://gdpr.eu/what-is-gdpr/