Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

Eksamensrapport

Titelblad

Projekt er udarbejdet i perioden:
Fra 24/10-2022 til og med 28/10-2022
På Medieskolerne Viborg

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

Indholdsfortegnelse

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

Indledning

Denne rapport er udgivet af Tobias Leth Skov, i forbindelse med webudviklernes afgangseksamen for holdet WebH121-2 på medieskolerne i Viborg. Rapporten er udgivet i led af afgangseksamen, hvor der tages udgangspunkt i et UX/UI mockup udleveret på eksamensdag 1. Det færdig resultat er udviklet i HTML, CSS og JS. Dog er der anvendt Vue3 som framework til JS, Scss med TailwindCSS til styling og HTML5. Hele projektet er skrevet i Visual Studio Code og koden er udgivet på GitHub. Projektets frontend kan ses online på eget domæne.

Der skal gå en stor tak til JCD A/S, da opgaven er udført i næsten samme kode miljø som eleven vil stå foran når hverdag efter uddannelse falder for. JCD A/S har været elevens praktikplads det sidste halv år, hvorefter der skrevet kontrakt om fast ansættelse.

Opsætning af projekt

Projektet er udgivet på GitHub og kan tilgåes via følgende link: https://llk.dk/6fon7l

Her fra kan projektet hentes ned på den ene eller anden måde, det individuelt hvordan projektet hentes ned om det med SSH, HTTP eller som .zip fil der udpakkes.

Folderen kan nu åbnes i en vilkårlig editor, så kan man tilgå og se koden.

Hvis projektet skal køres skal man først åbne terminalen, navigere sig frem så man står i projektmappen og derefter køre kommandoen *'yarn'* her efter kan man anvende *'yarn build'* sår får man tildelt en lokalt host, hvorpå siden kan afprøves.

Se projektet

Ønskes der blot kun at se og anvende frontend projektet som helhed kan det tilgås via følgende link: https://llk.dk/3sy090

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

Teknisk gæld

Projektet er skrevet for at minimere teknisk gæld så vidt muligt. Teknisk gæld eller *'technical debt'* er eller kan være udfaldet af dårlig kode *'lazy coding'* eller udfaldet af forældet system. Når en webudvikler får et nyt projekt kan det gå en af tre veje.

1. Kunden kommer med et stramt budget og en stram deadline, som resultere i hurtig og dårlig kodning fra udvikleren.
2. Kunden kommer med et stramt budget men vil gerne have noget godt kode, dette resultere i deadlinen bliver skubbet til et senere tidspunkt, hvilket ikke altid er en god ting for kunden.
3. Kunden kommer og ber efter et godt stykke kode, med en kort deadline. Dette resultere i at udvikleren køre masser timer af for at nå deadline og dermed bliver prisen dyrere.

Dette scenarie ses sjældent at kunden kommer med et stort budget, en sen deadline og derved kan få en det bedst mulige kode.

Ved overspringshandlinger helt fra start resultere det i teknisk gæld på lang sigt. Der også en faktor der er kommer udefra, det kunne f.eks. være frameworket som den originale kode ikke bliver supportere pr. dagens dato.

Hvilke aspekter viser minimal teknisk gæld?

Projektet er lavet med det mindset at det skal optimeres så kunden Smuk.Nu nemt kan skallere deres projekt. Kunden har ytret at denne løsning der udvikles i denne omgang blot er første fase. Der skal blot udvikles noget de kan komme i luften med, men at website der er skalerbart, dog med det ene formål at skabe trafik og kendskab til Smuk.nu og her.

Derfor har det været vigtigt for udvikleren at lave noget der strømlinet, i en sådan grad at, det nemt for udvikleren at komme tilbage til projektet om x antal tid, uden at skulle akkumulere tekniske gæld for kunden allerede fra start. Dog inden for de rammer eksamen tilbyder.

Nuværende akkumuleret teknisk gæld.

Var der ikke en eksamens ramme der sagde man ikke måtte anvende et CMS, havde projektet set helt anderledes ud nu. Da eksamens rammerne ikke gør det muligt at anvende CMS fra projektet begyndelse, vil det resultere i at store ændringer og en større refaktorering til Smuk.nu. Da udvikleren i næste fase skal bruge tid (penge) på at omskrive koden til at det passer med et CMS. Havde det været i en ideel verden, var kunden på nuværende tidspunkt blevet introduceret til Umbraco 10[1] som cms. Her vil man kunne spare tid og penge samt opsætning timer og omskrivning timer på at opkoble et CMS fra start. Men lad os kigge videre på den nuværende løsning for at undersøge hvorfor denne løsning er en god start.

[1] Læs mere om Umbraco -> https://umbraco.com/

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

Opsætningen af kode

Projektet er opsat med Vue 3 som framework til at styre logik og skabe et komponent baseret layout. Alt dette er stylet i SCSS med Tailwind til at styre parameter.

Vue 3 frem for vanilla JS.

Hele projektet kunne sagtens udføres i vanilla JS, men for lethedens skyld og skalerbar hedens skyld er projektet udført med Vue 3. Vue 3[2] kan på sigt dog akkumulere en del teknisk gæld. Virksomheden udvikleren har stiftet bekendtskab med løber nu ind i en situation hvor alt deres JS er skrevet i Vue 2, som har 'end of life'[3] i slutningen af 2023. Dette har skabe stor teknisk gæld for virksomheden men også for virksomhedens kunder, da flertallet af projekter er udviklet i Vue 2. Hvem skal betale tiden der bliver brugt på at omskrive kundes komponenter til Vue 3 eller et nyt alternativ framework?

Komponent baseret layout.

Vue 3 gør det muligt at man kan lave komponent baseret kode. Et godt eksempel her kunne være hero komponent. I stedet for udvikleren skal lave hero sektionen på forsiden og derefter kopier og indsætte samme kode på alle .html filer.

Det hele starter på index.html siden, hvor der i body er anvendt en app komponent. Denne app anvender VueRouter[4] til at pege rundt på de forskellige sider.

Der er så oprettet en router fil som peger på de forskellige sider og laver logikken bag. På hver side bliver der anvendt komponenter til at vise sidens delelementer. Disse komponenter er muligt at tilgå fra alle filer og derved gør det nemt at genanvende og gøre ens indhold dynamisk.

Denne opsætning er med til at mindske det akkumuleret tekniske gæld fra start, da tingene er let og overskueligt at finde rundt i. Ligeledes gør det det let for næste udvikler at genanvende forskellige sektioner på nye sider.

Et godt skrevet Vue komponent

Lad os tage udgangspunkt i AllProducts.vue og SelectedProducts.vue som begge pegere videre på ProductList.vue. Ude i roden hvor vi første gang kunne tænke os at vise en af komponenterne skal man blot skrive tagget efterfølgt af nogle parameter som er sat.

---

[2] Læs mere om Vue 3 -> https://vuejs.org/
[3] Læs mere om Vue 2 end of life -> https://vuejs.org/about/faq.html#what-s-the-difference-between-vue-2-and-vue-3
[4] Læs mere om VueRouter ->https://router.vuejs.org/

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

```
<section class="py-16">
    <div container>
        <h1 class="text-center">udvagt <span class="text-pink">skønhed</span></h1>
        <selectedProducts class="pt-20 " :ids="[1213213211, 4335542819, 2332233444, 4566543883]"></selectedProducts>
    </div>
</section>
```

På forsiden kalder vi SelectedProducts.vue efterfulgt af id'erne på de produkter vi godt kunne tænke os at se.

```
<section class="py-16">
    <div container>
        <h1 class="text-center">Alt er <span class="text-pink">Skønhed</span></h1>
        <allProducts class="pt-20"></allProducts>
    </div>
</section>
```

På produktsiden kalder vi AllProducts.vue, da vi skal vise alle produkterne følger der ikke nogle parametre med denne gang.

Inde i SelecetedProducts definere vi en props hvor vi får id'erne med ind og dets værdier. Her kalder vi ProductList.vue komponentet og giver den window.products med. Her inkluderes der et filter som filtrerer alt indhold der indeholder id'et med videre til næste komponent. Dette gør at man inde i ProductList ikke får alt muligt med som ikke er nødvendig for følgende opgave, nemlig at vise de udvalgte produkter.

```
const props = defineProps({ ids: Array })
const products = ref(window.products)
</script>

<template>
    <ProductList :products="products.filter((product) => {return ids.includes(product.id)})"></ProductList>
</template>
```

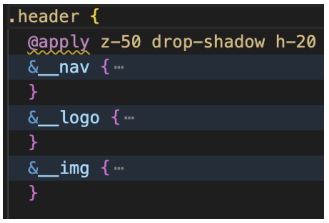
Inde i ProductList.vue modtagere vi en array med de produkter vi ønsker at vise og ikke mere eller ikke mindre. Her kan man blot loop gennem arrayet og vise de forskellige informationer som danner indholdet.

```
<template>
    <ul class="product-list">
        <li v-for="(product, key) in products" :key="key">
            <div class="product-card">
                <div class="product-card__img-area">
                    <img :src="product.image" :alt="'produkt billede af '+product.title">
                    <div v-if="product.discountInPercent" class="discout">
                        <p class="discout__heading">SPAR</p>
                        <p class="discout__procentage">{{product.discountInPercent}}%</p>
                    </div>
                </div>
                <div class="product-card__content">
                    <h3>{{product.title}}</h3>
                    <div row class="justify-between mt-auto">
                        <p class="font-bold mb-1">{{product.price}} kr.</p>
                        <button class="btn-dark" v-on:click="() => {buyHandler(product)}">Køb</button>
                    </div>
                </div>
            </div>
        </li>
    </ul>
</template>
```

Grunden til denne framgangsmåde er valgt er som sagt at de små enkeltdele i komponenterne nemt kan tilgås overalt, derved kan man også nemt tilføje funktioner og danne sig et overblik. Alt dette er med til at mindske risikoen for at skabe teknisk gæld.

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022

SCSS & Tailwind CSS



SCSS el. SASS[5] er industri standard inden for strukturering af CCS koden. Nølgen til brug af SCSS er at man kan neste indhold.

Der er anvendt Tailwind CSS til at style de mange parameter. Dette gør resultater er hurtigere at se og hurtigere at tilgå. Det måske ikke pænere at læse men det hurtigere at skrive. Her har man udgangspunktet mobilefirst, og fra skalere man ellers op efter nødvendigheden. Ligeledes kan man også nemt oprette sine egne klasser til styring at småting som farver såvel som store ting som sektioner.

Overall er SCSS et must have når man skal skrive ordentlig og struktureret kode. Tailwind CSS er kun et plus for udvikleren når det kommer til udførsel af koden. SCSS'en er også struktureret i mange underdele der indgår i den generelle css fil. Derudover kan man nemt finde stylingen der har præcis indvirke på et specifikt komponent.

API kald

Der bliver kaldt ind fra de tre endpoints via 1 funktion. Det forskellige data bliver herefter gemt i vinduet. Dette gør at udvikleren nemt kan tilgå de forskellige endpoint fra roden af siden.

```
Promise.all([axios.get('https://smuknu.webmcdm.dk/products'), axios.get('https://smuknu.webmcdm.dk/questions'), axios.get('https://smuknu.webmcdm.dk/reviews')]).then((res) => {
  window.products = res[0].data

  window.questions = res[1].data

  window.reviews = res[2].data
```

Herefter er det blot at finde ud af hvilke komponenter der skal have adgang til de forskellige api'er. Udvikleren skal nu blot refererer til f.eks. produkter, og loop gennem objektet og derved anvende dataen her i. Her spares der tid både i udførelsen af koden, men også når der på sigt skal bygges flere funktioner til hjemmesiden.

Post til API.

Til post og behandling af api'et anvendes der Axios[6]. Axios er et js promise baseret på HTTP klienter. Det er nemt at anvende og udføre jobbet i mange forskellige forbrugs scenarier. Axios er yderst populært både for små udvikler og store firmaer, og er ofte go to på arbejdspladsen.

```
const handleSubmit = (e) => {
  name = e.name
  loading.value = true
  e.preventDefault()
  const data = new FormData(form.value)
  const output = {}
  for (const pair of data.entries()) {
    output[pair[0]] = pair[1]
  }

  axios.post('https://smuknu.webmcdm.dk/subscribe', output).then(res => {
    console.log(output)
    if (res.status === 201) doneMessage.value = res.data
    else console.error(res.message)

    loading.value = false
  })
}
```

[5] Læs mere om SASS og SCSS -> https://sass-lang.com/
[6] Læs mere om Axions -> https://github.com/axios/axios

Medieskolerne Viborg, Webudvikler
Elev: Tobias Leth Skov
Hold: Web121-2 **Projekt**:
Dato: 28-10 2022
Her ses det stykke kode der håndtere og formatere koden så det bliver sendt rigtig til API'et. Her anvendes der FormData som strukturere det rå data der kommer fra input feltet til noget som server APIet kan læse.

Hvad gik godt og hvad gik ikke så godt

Projektet blev udført efter de specifikke ønsker fra kunden til en sådan grad at det inden for rammerne for eksamensprojektet. Udviklere kunne godt selv have tænkt sig at implementere det hele i Umbraco 10, da kunden selv ønsker at kunne tilføje og redigere produkter på sigt. Et nyt setup i Umbraco havde bestemt ikke gjort det nemmere for udvikleren nu og her. Da udvikleren skal lave side struktur i .CSHTML altså i .net samtidig med at Vue 3 og Tailwind skal sættes op på en lidt anden måde. Dog er det hos kunden et punkt der skaber teknisk gæld, da kunden ikke har fået hvad de vil have på lang sigt, men blot hvad kunden godt kunne tænke sig som en lappeløsning.

Det gode

Projektet er forløbet så godt at udvikleren blev færdig til tiden med alt implementeret. Dette er ikke kun hardkodet løsning men også dynamisk og skalerbart. Der brugt mange alternative metoder som er standard praksis for virksomheder. Ligeledes er koden så universalt skrevet at en kommende programmør nemt vil kunne tage projektet op. Uden at kunden skal betale for inspektion tid. Derved mindsker det den tekniske gæld for kunden.

Det knap så gode

Projektet er skrevet i Vue 3, som på et eller andet tidspunkt også har end of life. Dog er valget gået på at det nu og her er hurtigere og nemmere at streamline sin kode. Ligeledes har der været forsøgt at lave et admin dashboard, som er gået godt, men ikke tilstrækkelig godt til at det medgår i første løsning til kunden. Ved at sende et halvt admin dashboard med, risikere vi at akkumulere teknisk gæld hos kunden, men også aflevere et halvfærdig resultat der bliver for specifikt og hardcoded til at udvikleren var tilfreds herom.