Applikationer: Teknisk vejledning

Erhvervsfremmeplatform

Contents

1 Introduktion

På Virksomhedsguiden (VG) er der et krav om at kunne præsentere indlejret indhold i form af små apps, som er bygget af eksterne leverandører. Dette indhold bygges som små selvstændige komponenter og er refereret til som applikationer. Eksempler på applikationer kunne være:

- Risikovurderingsværktøj
- Eksempel 2

1.1 Applikationer

Applikationer er som sagt små selvstændige apps som skal sikre at eksterne leverandører kan bidrage med at producere indhold til Virksomhedsguiden. Samtidig sikrer applikationer at brugeren får en komplet brugeroplevelse uden at skulle navigeres væk fra VG.

En applikation kan ikke kommunikere med VG i form af database kald, men kan kun gøre brug af de bibliotekter som bliver stillet til rådighed af VG. En applikation har dog mulighed for at kommunikere og hente data fra eksterne web services. Dette bliver beskrevet i flere detaljer i afsnit 3 (Teknisk guide).

1.2 Formål

Formålet med applikationer er at præsentere brugeren af VG for spændende apps, som er bygget af eksterne leverandører, og dermed sikre at brugeren får den bedst mulige oplevelse på én samlet platform.

1.3 Målgruppe

Denne guide er henvendt til de eksterne leverandører som skal udvikle applikationer til Virksomhedsguiden. Det forventes derfor at læseren har kendskab til web udvikling, herunder HTML, CSS og Javascript.

2 Processer

For at en applikation kan blive præsenteret på VG er der en del processer der skal gennemføres. Disse processer bliver beskrevet i følgende afsnit og dækker både over de processer som en leverandør skal igennem, samt dem som administrator og redaktør kommer igennem.

2.1 Bliv applikation leverandør

2.1.1 Projektbeskrivelse til ERST

Nye leverandører af applikationer skal godkendes af Erhvervsstyrelsen. Se dokumentationen Applikationer: Forretningskrav for at læse mere om de forskellige krav der stilles til nye leverandører.

2.1.2 Login til redaktion

Ved oprettelse af ny leverandør får leverandøren tilsendt en mail med login til VG redaktionen, hvor leverandøren kan administrere sine applikationer og oprette nye.

2.1.3 Adgang til applikation vejledning

Ved oprettelse af ny leverandør får leverandøren tilsendt en mail med link til denne vejledning og til Applikationer: Design vejledning.

2.1.4 Testmiljø

Ved oprettelse af ny leverandør får leverandøren tilsendt en mail med link til et testmiljø som er tilgængeligt via et Git repository. Testmiljøet gør det nemt og hurtigt for leverandøren at komme igang med udvikling og test af applikationer (se afsnit 2.2).

2.2 Bygge og teste nye applikationer

Når man er godkendt som leverandør kan man begynde at udvikle og teste sin applikation. Dette gøres ved at clone det Git repository som man får adgang til ved oprettelse. Testmiljøet sættes hurtigt og nemt op ved at følge de steps som er beskrevet i readme. Disse er:

- Åben et command-line vindue fra ApplikationTest mappen
- Kør npm install
- Kør npm run serve
- Tilgå den localhost url som skrives i powershell efter npm run serve
- Åben ApplikationTest projektet i en teksteditor
- Start udvikling i Applikation.vue som ligger i components mappen

Der medfølger hot reload i testmiljøet, så ændringer i Applikation.vue slår igennem på localhost med det samme. Husk kun at lave ændringer i Applikation.vue, da det kun er kode i denne komponent der vil blive accepteret som applikation kode.

2.3 Godkendelse af ny applikation

Når leverandøren er færdig med at udvikle og teste sin applikation, skal man logge ind i VG administrationen (login er tilsendt på mail i forbindelse med oprettelse af leverandør), klikke på "Administrer applikationer" og indtaste de nødvendige oplysninger.

Leverandøren skal angive en URL til den applikation, der skal godkendes (Applikation.vue).

Da leverandørens kildekoden hentes og bearbejdes programmatisk, er det vigtigt denne kan nås med gængse netværks programmer ala. Curl eller lignende.

Hvis kildekoden ligger i et beskyttet repository, er det leverandørens ansvar at oprette og angive en access_token i URL'en så kildekoden kan tilgås.

Følgende Git hosting understøttes:

2.3.1 GitHub

URL til GitHub hostet kildekode skal angives på følgende form:

```
https://${ACCESS_TOKEN?}@raw.githubusercontent.com/${REPO}/${PROJECT}/${SHA}/${FILE_PATH}
```

Eksempel:

```
REPO: vgapplication
PROJECT: app1
FILE_PATH: src/components/Applikation.vue
ACCESS_TOKEN: b68f106cb74f3eddea5c01561384a4536a521fa7
SHA: bf6a684d58366b888aa3ed47f1584327d6612c99
```

Browser URL til kildekode:
https://github.com/vgapplication/app1/blob/bf6a684d58366b888aa3ed47f1584327d6612c99/src/components/Applikation.vue

Skal laves om til:
https://b68f106cb74f3eddea5c01561384a4536a521fa7@raw.githubusercontent.com/vgapplication/app1/bf6a684d58366b888aa3ed47f1584327d6612c99/src/components/Applikation.vue

Bemærk URL'en skal pege på et commit SHA (bf6a684d58366b888aa3ed47f1584327d6612c99), så versionen af kildekoden kan fastfryses. En URL med angivelse af branch vil ikke blive godkendt, da den begvedliggende kode kan ændre sig.

2.3.2 GitLab

URL til GitLab hostet kildekode skal angives på følgende form:

```
https://${GITLAB_HOST}/api/v4/projects/${PROJECT_ID}/repository/files/${FILE_PATH}/raw?ref=${SHA}
```

Eksempel:

```
GITLAB_HOST: gitlab.com
FILE_PATH: src/components/Applikation.vue
REPO: vgapplication
PROJECT: app1
PROJECT_ID: 1817
ACCESS_TOKEN: aUx18aeb31PUfw-PNAMQ
SHA: f957085aeab09f8d18000e98508c85b95655da84
```

Browser URL til kildekode:
https://gitlab.com/vgapplication/app1/-/blob/f957085aeab09f8d18000e98508c85b95655da84/src/components/Applikation.vue

Skal laves om til:
https://gitlab.com/api/v4/projects/1817/repository/files/src%2fcomponents%2fApplikation.vue/raw?ref=f957085aeab09f8d18000e98508c85b95655da84&private_token=aUx18aeb31PUfw-PNAMQ

Bemærk at stier skal URL encodes.

2.3.3 Krav til leverandør

Hvis applikationen benytter sig af NPM moduler, som endnu ikke er tilføjet til testmiljøet, så er det meget vigtigt at leverandøren informerer omkring dette så tidligt i forløbet som muligt, da det vil kræve en ny deployment release af Virksomhedsguiden der skal planlægges i god tid. Følgende information skal angives ved nye NPM moduler:

- NPM modul navn
- Version
- Typescript declaration file (hvis nødvendigt)

2.3.4 Krav til applikation

- Eksterne HTTP kald skal være HTTPS grundet Virksomhedsguiden er
- Stylesheet skal på nuværende tidspunkt være indlejret i Applikation.vue, og må derfor ikke være referer til eksterne filer.
- Der må ikke være fejl eller warnings ved "npm run serve"

2.3.5 Huskelist inden indsendelse

- Leverandør bør teste en applikation med preview knappen inden den sendes til godkendelse
- URL til kildekode skal kunne nås programmatisk evt. med Curl

Når alle nødvendige informationer er angivet, indsendes applikationen til godkendelse. En administrator vil herefter review koden og enten afvise eller godkende applikationen. Hvis applikationen bliver afvist, sender administrator en mail til leverandør med grundlag for afvisningen.

2.4 Visning af godkendt applikation i VG

Så snart en applikation er godkendt af administrator, er applikationen aktiv og tilgængelig for redaktører, så de kan indsættes på VG.

2.5 Opdatering af en eksisterende applikation

Når der skal laves opdateringer til en applikation, kan leverandøren gå ind i VG administrationen, klikke på Administrer applikationer, og vælge den applikation som skal opdateres, fra en liste. Listen indeholder de eksisterende applikationer som leverandøren har oprettet hos VG. Det er muligt at opdatere navnet på applikationen eller indsende ny kode til godkendelse.

På samme måde som når der oprettes en ny applikation, bliver den opdaterede applikation kode reviewet af en administrator som enten godkender eller afviser koden. Godkendes koden, bliver den gamle kode overskrevet af den nye.

2.6 Slet en applikation

Hvis en leverandør vil have fjernet en applikation fra VG, skal der sendes en mail til Erhvervsstyrelsen med navnet på den applikation som skal slettes. Husk at angive i mail om applikationen bare skal afpubliceres eller fjernes helt fra systemet. Hvis en applikation afpubliceres, kan redaktører ikke længere tilføje applikationen til ydelser, men den eksisterer stadig i systemet og kan publiceres igen på et senere tidspunkt.

2.7 Tilføj applikation til ydelse

Applikationer bliver i VG behandlet som indholdselementer, som er et element der kan indsættes på en ydelse, af en redaktør. Dette kan f.eks. være et tekstfelt, et card, en accordion eller en listevisning. Når der oprettes en ny applikation ydelse kan redaktøren vælge at tilføje en af de eksisterende applikationer som allerede er godkendt.

3 Teknisk guide

Dette afsnit beskriver de teknologiske muligheder og afgrænsninger en leverandør har til rådighed, for at kunne udvikle applikationer.

3.1 Teknologi

3.1.1 VueJS

VG frontend er udviklet med Javascript frameworket VueJS 2.x, som er et open-source framework der kan sammenlignes med andre Javascript frameworks som React og Angular. VueJS er et moderne framework som gør det nemt at bygge brugerflader ved at tilbyde en masse features der skaber forbindelsen mellem HTML og Javascript. En af kernepunkterne i VueJS er et system der tillader udvikleren at lave reaktiv rendering i DOM. Så når der laves ændringer i Javascript er det ikke længere nødvendigt at manipulere med DOM for at ændringerne slår igennem. Det sker helt automatisk med VueJS.

For at læse mere om de mange fordele som VueJS tilbyder, kan man finde dokumentationen på https://vuejs.org/v2/guide/.

3.1.2 Typescript

For at sikre høj kodekvalitet er alle VueJS komponenter i VG udviklet i Typescript, som er et programmeringssprog der er bygget på Javascript og som sikrer typestærk Javascript kode. Det er derfor også et krav at der anvendes Typescript i udviklingen af applikationer. Hvis man vil lære mere om Typescript, finder man den officielle dokumentation på https://www.typescriptlang.org/.

3.1.3 SCSS

VG anvender SCSS til styling af brugerfladen. SCSS er en udvidelse af CSS som tillader udvikleren at gøre brug af variabler, metoder og nested styling. Alle applikationer skal derfor bruge SCSS syntax i deres styling. Vil man lære mere om SCSS, kan man finde dokumentationen på https://sass-lang.com/.

3.2 Single file components

Alle applikationer som skal vises på VG skal være udviklet som VueJS single file components (https://vuejs.org/v2/guide/single-file-components.html). Dette sikrer at en applikation renderes korrekt på VG og at man har alt kode samlet i én fil. En single file component er en .vue fil som indeholder alle aspekter af en web applikation. HTML, Javascript og styling. Derfor består alle .vue filer af tre sektioner: <template>, <script> og <style>.

```
<template>
  <div>
    <!-- HTML here -->
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';

@Component({
  name: 'Template',
})
export default class Template extends Vue {
 //Typescript here
}
</script>

<style scoped lang="scss">
  //SCSS Styling here
</style>
```

I <template> tagget skriver man HTML til sin applikation. <script> tagget indeholder Typescript, og det er derfor nødvendigt at angive attributten lang="ts", for at fortælle at der kodes med Typescript. Endeligt skal <style> tagget indeholde applikation styling. Ligesom med script skal man her også angive en lang attribut for at fortælle compileren at man bruger SCSS. Udover dette skal style tagget også indeholde attributten "scoped", da dette sikrer, at SCSS kun gælder for den aktuelle applikation.

Nedenfor er et eksempel på en meget enkel applikation - den har en datamodel der indeholder en msg, en template der indeholder et div tag og sætter msg fra modellen ind i div tagget og endelig en CSS klasse, der anvender en rød farve.

```
<template>
  <div class="example">
    {{msg}}
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';

@Component({
  name: 'Template',
})
export default class Template extends Vue {
  private msg = 'Hello world';
}
</script>

<style scoped lang="scss">
  .example {
    color: red;
  }
</style>
```

Der er flere Vue.js teksteditor plugins til rådighed, som vi opfordrer applikation leverandører til at bruge (Emacs, Sublime Text, Visual Studio og Visual Studio Code, Intellij og andre).

3.3 Tilgængelige biblioteker

Som nævnt i introduktionen, er det ikke muligt for en applikation at kommunikere med VG. Men VG stiller nogle redskaber til rådighed som en applikation kan gøre brug af.

Hver applikation lever i sit eget scope hvilket vil sige at en applikation ikke kan hente data fra parent komponenter. Derfor skal en applikation importere de tilgængelige biblioteker hvis de skal bruges. I eksemplet herunder importeres bibliotekerne axios og luxon så de er tilgængelige i applikation koden.

```
<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';
import axios from 'axios';
import { DateTime } from 'luxon';

@Component({
  name: 'Widget'
})
export default class Widget extends Vue {
  private currentTime = '';
  private response = {};
  private error = {};
  private loadingResponse = false;

  mounted() {
    this.currentTime = DateTime.local().toISO();
    this.loadingResponse = true;
    axios.get( url: 'https://httpbin.org/get').then((rsp: any) => {
      this.response = rsp;
      this.loadingResponse = false;
    }).catch((error: any) => {
      this.error = error;
    });
  }
}
</script>
```

3.3.1 Det fælles designsystem

Hele brugergrænsefladen i VG er bygget op omkring Det Fælles Designsystem (DFS - https://designsystem.dk/). Det Fælles Designsystem er udviklet for at sikre en ensartet funktionalitet og design i selvbetjeningsløsninger på tværs af myndigheder. Det skaber genkendelse for brugerne af løsningerne og gør det nemmere for dem at foretage selvbetjening.

Alle komponenter som er tilgængelige i DFS er tilgængelige for applikation leverandøren. Man behøver ikke at importere DFS da det allerede er aktiveret globalt. De tilgængelige komponenter er beskrevet i detaljer i applikation Design Guiden (Applikationer: Design vejledning).

Det er et krav at man som applikation leverandør gør brug af de komponenter som er tilgængelige i DFS, og er derfor ikke tilladt at udvikle sine egne standard komponenter.

3.3.2 Axios

VG leverer Axios biblioteket for at gøre det let for applikation leverandører at kommunikere med deres eget REST Web API. For at bruge Axios kan man skrive axios.get eller axios.post i sektionen <script> i .vue filen for at lave et kald til et eksternt API.

Axios bruger promises som standard og gør det nemt at lave asynkrone webservicekald og foretage fejlhåndtering som illustreret ved nedenstående eksempel.

```
<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';
import axios from 'axios';
import { DateTime } from 'luxon';

@Component({
  name: 'Widget'
})
export default class Widget extends Vue {
  private currentTime = '';
  private response = {};
  private error = {};
  private loadingResponse = false;

  mounted() {
    this.currentTime = DateTime.local().toISO();
    this.loadingResponse = true;
    axios.get( url: 'https://httpbin.org/get').then((rsp: any) => {
      this.response = rsp;
      this.loadingResponse = false;
    }).catch((error: any) => {
      this.error = error;
    });
  }
}
</script>
```

Axios kan lave både GET, POST, PUT, DELETE osv., og understøtter også tilføjelse af interceptors til requests/responses.

Som applikationleverandør har du ikke tilladelse til at ændre den globale Axios standardkonfiguration eller tilføje interceptors til den globale Axios instans. Applikationer der bruger den globale Axios instans på denne måde, godkendes ikke.

Se https://github.com/axios/axios for at få flere oplysninger om, hvordan du bruger Axios.

3.3.3 Luxon

Luxon er et bibliotek som bruges i VG til at håndtere datoer og tidspunkter. Med Luxon er det meget nemt at parse forskellige dato/tids formater, og formatere dem til det man har brug for. Samtidig leverer Luxon mange features til at manipulere datoer/tider og gør det nemt at arbejde med tidszoner hvis dette er nødvendigt.

Udvikleren af Luxon er også udvikler af MomentJS, og derfor vil man også kunne genkende mange af de samme features, hvis man tidligere har brugt Moment.
Se https://moment.github.io/luxon/ for at læse mere dokumentation om Luxon.

3.4 Brug af eksterne API

3.4.1 Personfølsom data

Alle applikationer som præsenterer data fra eksterne API skal overholde GDPR loven (https://gdpr.dk/) i forhold til personfølsom data.

3.5 Begrænsninger/regler

Grundet sikkerhed og stabilitet er der nogle begrænsninger som applikation leverandører skal følge for at kunne få godkendt en applikation. Enhver applikation der ikke følger de nedenstående regler, vil ikke blive godkendt i review processen.

3.5.1 Ingen v-html

Alt HTML-indhold skal være en del af <template> sektionen i .vue-filen. Leverandører har ikke tilladelse til at injecte dynamisk oprettet HTML, f.eks. ved at indlæse den fra en webservice, ved brug af innerHtml, v-html eller lignende tilgange.

3.5.2 Ingen $parent

Det er ikke tilladt at bruge this.$parent for at tilgå parent component i Vue.

3.5.3 Ingen DOM manipulation

JavaScript i en applikation har ikke tilladelse til at bruge JQuery og har ikke tilladelse til at tilgå eller manipulere DOM. Ydermere er det ikke tilladt at til gå window.location, window.history eller enhver anden adgang til window, undtagen window.navigator (for at få adgang til oplysninger om browseren).

3.5.4 Alle links skal bruge target="_blank"

Alle links (<a href>) skal bruge attribut target="_ blank" for at åbne linket i en ny fane / et nyt vindue.

```
<a href="http://www.google.com" target="_blank">Link til Google</a>
```

3.5.5 Alle kald til eksterne API skal være async

Al adgang til Web API'er skal bruge asynkrone kald for at holde VG responsiv.

```
private async callExternalApi() {
  axios.get( url: 'https://httpbin.org/get').then((rsp: any) => {
    this.response = rsp;
    this.loadingResponse = false;
  }).catch((error: any) => {
    this.error = error;
  });
}
```

3.5.6 <style> tag skal gøre brug af scoped attribut

<style> tag skal altid indeholde attributten "scoped" for at begrænse CSS-styling til selve applikation komponenten.

```
<style lang="scss" scoped>
  h1 {
    color: $color-link;
  }
</style>
```

3.5.7 Ingen opdatering af eksisterende cookies

Det er ikke tilladt at manipulere eksisterende cookies eller request headers.

3.6 Best practices

3.6.1 Performance

VG applikationer udføres på samme måde som modulerne inde i selve VG (som også er Vue.js komponenter) og for at opnå en god brugeroplevelse er det vigtigt, at applikationer og deres underliggende Web API'er er performance optimeret.

JavaScript koden i Vue.js applikationer er kun til "frontend-logik", dvs. til at behandle events, eksekvere web API kald og tildele resultater fra API kald til modellen. Javascript bør ikke indeholde forretningslogik eller komplekse beregninger. Dette skal håndteres i leverandørens Web API.

3.6.2 Typestærk kode

For at sikre høj kodekvalitet i applikationer, er det vigtigt at applikationer bliver udviklet med fokus på typestærk kode. Derfor er det et krav at JavaScript i applikationer udvikles med TypeScript.

3.6.3 Error handling

Sørg for, at Typescript koden håndterer uventede fejl og viser en meningsfuld fejlmeddelelse til brugeren:

- Alle Web API-opkald skal have en fejlhåndtering, der viser brugeren, at noget gik galt og muligvis giver dem mulighed for at prøve igen.
- JavaScript kode skal have try/catch, der også viser brugeren, at noget gik galt.

3.6.4 Loading spinner

Ethvert Web API kald kan ende med at tage længere tid end forventet (på grund af latenstid, netværksafbrydelse, serverhickups osv.). Sørg derfor altid for, at brugergrænsefladen viser en "loading"-indikator, så brugeren kan se, at noget loader.
Nedenstående komponent kan bruges til at angive en spinner i en applikation:

```
<div class="spinner" v-if="loadingResponse" aria-label="Henter indhold" />
```