Anssi Laurento

Vue 3 -komponentit

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

30.11.2021

Tiivistelmä

Tekijä:	Anssi Laurento
Otsikko:	Vue 3 -komponentit
Sivumäärä:	29 sivua
Aika:	30.11.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	COO Johanna Kalliomäki Lehtori Ilpo Kuivanen

Tämän opinnäytetyön kohde on osa Cloubi Oy:n Cloubi-sovellusta. Työn tavoitteena oli tarkastella, miten hyvin Vue 3 -ohjelmistokehys sopii osaksi Cloubin kehitystyökaluja. Aiheen taustalla on löytää mahdollisimman tehokas ja toimiva työkalu haastaviin ja vaihteleviin käyttötarkoituksiin.

Työssä käydään läpi, mitä ovat tämän päivän haasteet web-kehityksessä ja miten niitä voidaan ohjelmistokehysten ja etenkin Vue 3:n avulla ratkaista.

Työhön kuului Cloubin uusien digikokeiden oppilaiden kirjautumisnäkymän rakentaminen ja sen eri rakennuspalikoiden esittely.

Tuloksena syntyi toimiva ja vakaa sovellus, jonka kehitys ja ylläpito jatkuu myös tämän opinnäytetyön jälkeen. Työ todisti Vue 3:n toimivuuden ainakin tässä sovelluskohteessa ja sen käyttöä laajennetaan luultavasti useampaan paikkaan.

Avainsanat: Vue 3, ohjelmistokehys, web-kehitys, Cloubi

Abstract

Author:	Anssi Laurento
Title:	Vue 3 components
Number of Pages:	29 pages
Date:	30 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Engineering
Supervisors:	COO Johanna Kalliomäki
	Senior Lecturer Ilpo Kuivanen

The subject of this thesis is part of Cloubi Oy's Cloubi application. The aim of the work was to look at how well the Vue 3 framework fits into Cloubi's development toolkit. Underlying the topic is finding the most effective and efficient tool possible for challenging and varied uses.

The work reviews what are today's challenges in web development and how they can be solved with the help of software frameworks and especially Vue 3.

The work included building Cloubi's new digital exam students' login view and presenting its various building blocks.

The result was a functional and stable application, the development and maintenance of which will continue even after this thesis. The work proved the functionality of Vue 3 at least in this application and its use will probably be extended to more places.

Keywords: Vue 3, framework, web development, Cloubi

Sisällys

Lyhenteet

HTML: *Hypertext Markup Language.* Avoimesti standardoitu merkintäkieli, jolla verkkosivujen rakenne määritetään.

CSS: *Cascading Style Sheets.* Erityisesti verkkosivuille kehitetty tyylisivu. CSS:llä annetut säännöt ehdottavat, kuinka dokumentti voidaan esittää.

DOM: *Document Object Model.* Tapa kuvata rakenteisen dokumentin rakenne puuna, jonka eri olioita voi hakea, tutkia ja manipuloida.

REST: *Representational State Transfer*. HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen

1 Johdanto

Nettisivustoista on tullut koko ajan laajempia ja monimutkaisempia. Jonkun rajan jälkeen sivuistoista puhutaankin jo web-sovelluksina. Näiltä sovelluksilta vaaditaan myös enemmän. Sovelluksen tulee toimia ruudusta ja käyttäjästä riippumatta toisin kuin 2000-luvun alun tekstiä, linkkejä ja kuvia sisältävät sovellukset, joiden tuli lähinnä toimia Windows-pöytäkoneen vakioselaimella. Kaikkien toimintojen on toimittava ilman erillisiä lisäosia monilla eri käyttöjärjestelmällä ja selaimella. Samaan aikaan sovelluksen käyttökokemus ulkonäön ja tuntuman osalta on sellainen, että sovelluksen pariin on miellyttävä palata aina uudelleen ja uudelleen.

Työpöytä- ja mobiilipuolella sovellusten rakentamiseen on ollut erilaisia työkaluja saatavilla pitkään. Kasvavan monimutkaistumisen johdosta myös web-sovellukset tarvitsevat yhä enemmän työkaluja kehityksen tueksi. Nämä työkalut ovat erilaisia ohjelmakirjastoja ja nykyään entistä enemmän ohjelmistokehyksiä.

Tämän insinöörityön tavoitteena on tarkastella komponenttipohjaisten ohjelmistokehysten käyttöä ja hyötyä web-kehityksessä. Erityispaino on Vue 3 -ohjelmistokehyksessä, jonka uusia ominaisuuksia avaan ja esittelen eri käyttöpaikoissa.

Luvussa 2 käsittelen sovelluksen kohdetta eli Cloubi Oy:tä, sen sovellusta Cloubia ja Cloubin digikoeosiota.

Luvussa 3 käyn läpi web-kehityksen haasteita. Esittelen myös ohjelmistokehysten ja etenkin Vue 3:n tuomat ratkaisut näihin haasteisiin.

Luvussa 4 käydään läpi itse sovelluksen suunnittelua ja toteutusta. Käyn myös läpi tuloksia.

2 Cloubi Oy

2.1 Yleisesti

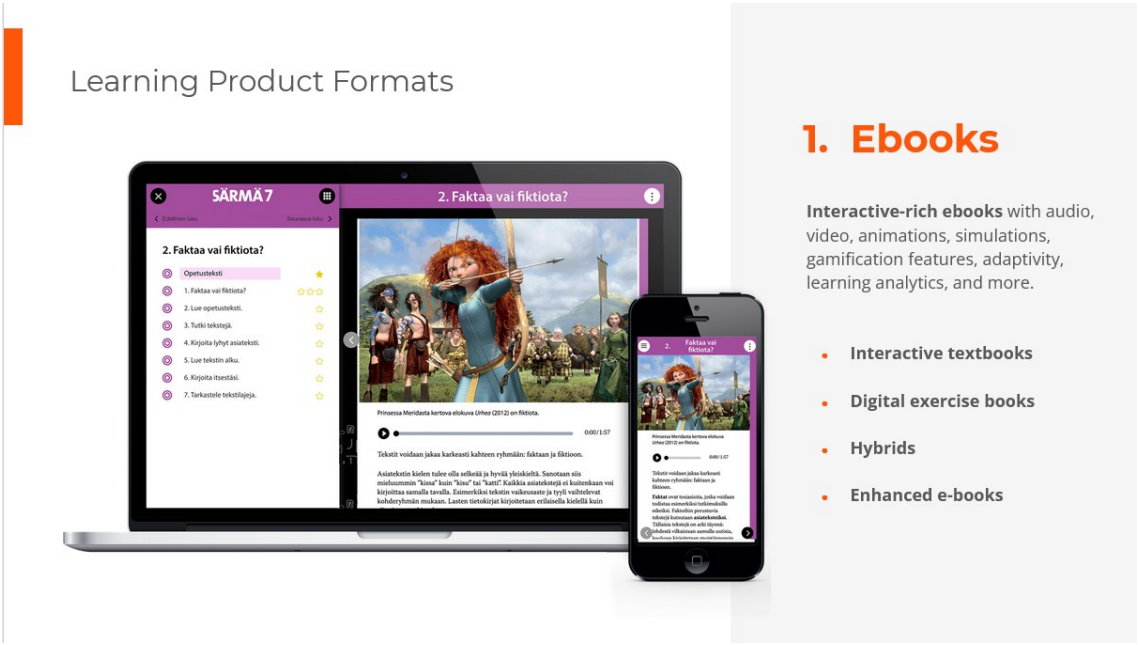
Cloubi Oy on 2011 perustettu suomalainen ohjelmistoyritys. Sen toimistot sijaitsevat Helsingissä ja Tampereella. Cloubi Oy on osa Kustannusosakeyhtiö Otavaa. Vuonna 2021 Cloubi Oy työllisti 35 henkeä ja sen pääasiakkaina ovat opetusalan kustantajat. [1.]

Liikevaihto on kasvanut tasaisesti ja vuoden 2020 tilikauden tulos oli 3,2 miljoonaa euroa. Tilikauden tulokset ovat vaihdelleet vuosina 2017–2020 0,5 ja 1,1 miljoonan euron välillä. Vuonna 2020 tulos ylsi 0,7 miljoonaan euroon. [2.]

Cloubi Oy:n tavoitteena on auttaa oppimateriaalikustantajia luomaan parempia digitaalisia oppimiskokemuksia tarjoamalla heidän käyttöönsä helppokäyttöisen sisällöntuotantoalustan. Kustantajille olemme ennen kaikkia kumppani, joka toimii oppaana digitaalisen transformaation keskellä. Olemme tarjonneet suomalaista opetusosaamista maailmalle perustamisvuodesta 2011 asti ja käyttäjiä onkin tänä päivänä jo yli 700 000 kymmenestä eri maasta.

2.2 Cloubi

Cloubi on Cloubi Oy:n kehittämä sovellus ja yhtiön päätuote. Sovelluksella voi rakentaa oppimiseen tueksi soveltuvaa materiaalia ja tehtäviä. Materiaalit sisältävät sisältösivuja, jotka rakennetaan tekstistä, videosta, äänistä ja tehtävistä (kuva 1).



Kuva 1. Sisältösivuesimerkki

Mediat luovat täytettä sivuille, mutta tehtävät ovat sisältösivujen suola. Sisällöntuottaja voi luoda tehtäviä useiden tehtävätyyppien valikoimasta, joihin lukeutuvat muun muassa avoimet tehtävät sekä monivalinta- ja aukkotehtävät. Avoimeen tehtävätyyppiin voi myös lisätä haluamaansa mediasisältöä. Vain sisällöntuottajan mielikuvitus on rajana.

Cloubi sisältää myös monia muita ominaisuuksia, jotka auttavat loppukäyttäjien elämää. Moduulipohjaisen back-endin ansiosta eri ominaisuuksia voidaan asiakkaan toiveiden mukaan ottaa heidän tuotteissaan vaivatta käyttöön tai pois käytöstä, jos tarpeet ajan myötä muuttuvat.

2.3 Cloubin bisnesmalli

Cloubi on white-label-tuote. Cloubi myy lisenssin asiakkaille, ja asiakkaat tarjoavat Cloubia osana portfoliotaan kuin omaa tuotettaan. Cloubi Oy saa myös osan asiakkaiden Cloubia käyttävien tuotteiden tuotosta.

Asiakkaiden Cloubi kustomoidaan heidän tarpeeseensa mukaisesti etenkin ulkoasun osalta. Tämän johdosta etenkin front-end vaatii sellaisen rakenteen, että kustomointi on mahdollista vähällä vaivalla. Kustomointiin on myös luotu helppo työkalu asiakkaille heidän omien teemojensa muokkaamiseksi.

2.4 Digikokeet

Tärkeä tai ehkä jopa tärkein osa, jota myös tämä insinöörityö käsittelee, on Cloubin uudet digikokeet. Tehtäviä sisältävistä sisältösivuista opettajat muodostavat kokonaisuuden, jota he haluavat testata oppilaillaan. Tehtävätyypistä riippuen tehtävät voidaan automaattisesti tai manuaalisesti arvioida.

Digikokeet koostuvat useasta osasta. Opettajan näkymässä opettajat voivat luoda, muokata, kopioida, ajastaa, käynnistää, lopettaa ja tarkastella kokeita. Aluksi luodaan koe. Opettaja nimeää kokeen ja latoo haluamassaan järjestyksessä haluamansa määrän sisällöntuottajan laatimia tehtäviä kokeeseen, jonka jälkeen kokeen voi julkaista. Tämän jälkeen kokeen voi ajastaa tai manuaalisesti aloittaa ja lopettaa.

Arviointinäkymässä opettajat voi pisteyttää oppilaiden suorituksia tehtäväkohtaisesti. Kommentteja voi lisätä yksittäiseen tehtävään, avoimen tehtävän tekstiin korostemerkinnällä. Koko kokeelle voi myös antaa yleisen kommentin. Oppilaiden suoritusten kokonaispisteet ja keskiarvot lasketaan, joten opettaja saa helposti yleiskuvan kokeen onnistumisesta.

Oppilaan puoli koostuu kolmesta osasta. Itse kokeessa ovat opettajan lisäämät tehtävät sekä aloitus- ja lopetussivu. Aloitussivu sisältää muun muassa yleiset ohjeet, ja lopetussivu sisältää muun muassa painikkeen kokeen palautukseen. Tulosnäkymässä oppilas näkee opettajan antamat pisteet ja kommentit.

Kirjautumissivut toimivat porttina digikokeisiin. Ne rakennettiin erillisenä osana muusta Cloubista ja ovat myös se osa, mitä tarkastelen lähemmin tässä insinöörityössä. Kirjautumiseen oppilaat tarvitsevat kokeen koodin, jonka opettaja heille antaa. Validin koodin syötettyään oppilaat pääsevät täyttämään nimensä ja sähköpostinsa tietokenttiin, jonka jälkeen oppilas siirtyy odotussivulle. Odotussivulta oppilas näkee henkilökohtaisen koodinsa ja voi liittyä kokeeseen sen alkaessa.

3 Ohjelmistokehykset web-kehityksessä

3.1 Web-kehityksen perusteet tiivistettynä

Web-sivustot koostuvat kolmesta elementistä: HTML:stä, CSS:stä ja JavaScriptistä. Kaikilla on oma tehtävänsä. HTML:n avulla määritetään elementit ja rakenne sivulle, CSS muotoilee ja värjää elementit, ja JavaScriptin ansiosta sivulla on interaktiivisuutta ja toiminnallisuutta, eli se mahdollistaa elementtien muuttumisen ja vaihtumisen.

3.2 Haasteet web-kehityksessä

Nettisivustot ovat nykyään monesti paljon muutakin kuin staattinen sivu tekstin ja kuvien kera. Sivustot vastaavat monesti toiminnallisuudellaan perinteistä työpöytäsovellusta. Lisääntyvät ominaisuudet vaativat uusia työkaluja ja toimintatapoja web-kehitykseen.

Käyttöliittymän rakentamisessa on otettava samaan aikaan huomioon miellyttävyys ja saavutettavuus. Loppukäyttäjän on haluttava käyttää sivustoa mahdollisimman pitkään, mutta toisaalta kaikkien on pystyttävä käyttämään sivustoa rajoitteistaan huolimatta.

Sivuston on oltava nopea. Ihmiset eivät jää odottamaan hitaasti latautuvaa sivustoa tai viiveellä toimivaa nappia sivustolla, jos heidän ei tarvitse. He menevät toiselle vastaavalle sivustolle. Sivuston elementit ja data on ladattava sopivaan aikaan sopivan kokoisissa paloissa, ja laskennalliset toiminnot on oltava sopivan keveitä, jottei käyttökokemus kärsi.

Skaalautuvuus on myös otettava huomioon. Käyttäjämäärät voivat vaihdella ajasta riippuen todella paljon: kymmenistä ihmisistä tuhansiin tai jopa miljooniin. Käyttäjämäärästä huolimatta sivuston on toimittava riittävällä tasolla ja mielellään aina yhtä sulavasti.

Tulevaisuudenkestävyys on etenkin suurille sovelluksille tärkeä ominaisuus. Sovellusyritykset haluavat sovelluksensa pysyvän toimivana ja kiinnostavana mahdollisimman pitkään. On siis valittava teknologioita, jotka eivät ole katoamassa, ja niiden kehitys jatkuu riittävän pitkään. Tämä mahdollistaa sovelluksen päivittämisen vastaamaan aina uusiin tarpeisiin. [3.]

3.3 Front-end-ohjelmistokehykset

Front-end-ohjelmistokehykset on kehitetty edellisen kappaleen front-end-puolen haasteiden kampittamiseen sekä nopeuttamaan kehitystyötä. Ne tarjoavat erilaisia ominaisuuksia ja valmiita työkaluja kehittäjien käyttöön.

Vuonna 2006 julkaistu jQuery oli ensimmäisiä ja suosituimpia front-end -ohjelmistokehyksiä. Se eroaa nykypäivänä käytetyistä ohjelmistokehyksistä olemalla

vain JavaScript-kirjasto. JQuery oli luokkansa suosituin mittaustavasta riippuen, aina vuoteen 2016–2017 asti, kunnes modernit ohjelmistokehykset valtasivat alaa.

Nykypäivän suosituimmat ohjelmistokehykset tarjoavat ominaisuuksia myös CSS- ja HTML-puolelle. 3 suurinta ohjelmistokehystä ovat React, Angular ja Vue. Ne kaikki ovat komponenttipohjaisia.

React on Facebookin kehittämä 2011 julkaistu ohjelmistokehitys. Se on tällä hetkellä suosituin näistä kolmesta suosituimmasta. React käyttää omaa JSX-syntaksiaan, mikä on laajennus JavaScriptin syntaksiin. Syntaksi yhdistää JavaScriptin ja HTML:n yhteneväiseksi sekoitukseksi. Reactin etuina ovat Facebookin tuki, useat päivitykset ja laaja käyttäjäkunta.

Angular on Googlen kehittämä, 2016 julkaistu ohjelmistokehys. Se erottuu kahdesta muusta TypeScriptin käytöllä. Reactissa ja Vuessa TypeScriptiä voi käyttää, mutta sen käyttö ei ole pakollista. Angularissa näkymäpuolen muutokset vaikuttavat tietopuoleen ja toisinpäin.

3.4 Vue 3 -ohjelmistokehys

3.4.1 Vue.js-taustaa

Vue.js on käyttöliittymien rakentamiseen kehitetty progressiivinen ohjelmistokehys, joka tarkoittaa, että web-sovelluksen saa tuntumaan natiivilta mobiilisovellukselta. Vuen pääkirjasto on keskittynyt näkymäkerrokseen, mutta Vuea voi käyttää modernien työkalujen ja kirjastojen kanssa.

Vuen avulla voi rakentaa erilaisia komponentteja, joita voi käyttää useassa sovelluksen osassa uudelleen. Vuessa muuttujia voi antaa lapsikomponenttien käyttöön propsien avulla. Propsien tyyppi voidaan määritellä ja ne voidaan myös validoida. Lapsilta emo-komponentille välittyvä tieto tapahtuu emit-funktion avulla. [4.]

3.4.2 Vertailu Reactiin

Koska React on vuonna 2021 suosituin web front-end -ohjelmistokehys, on Vuea hyvä verrata siihen.

Sivuillaan Vue on listannut seuraavat asiat, jotka yhdistävät vue.js:ää ja Reactia.

- Käyttävät virtuaalista DOM:ia.
- Keskittyvät ylläpitämään ydinkirjastoa. Monet toiminnot voi toteuttaa kumppanikirjastojen avulla.
- Tarjoavat reaktiivisia ja yhdistettäviä näkymäkomponentteja.
- On yhtä hyvä ajonaikainen nopeus.
- Molemmilla on hyvät reititys- ja tilanhallintaratkaisut laajoihin sovelluksiin
- Tarjoavat mahdollisuuden tehdä natiivin sovelluksen Androidille ja iOS:lle.

Vue ja Reactia erottavia tekijöitä ovat taas

- Vuessa komponentit päivittyvät aina vain muuttuvassa komponentissa, kun taas Reactissa myös komponentin alakomponentit päivittyvät samalla.
- Vuessa HTML, JavaScript ja CSS on perinteisemmin eroteltu toisistaan, kun taas Reactissa lähes kaikki rakennetaan JavaScriptin avulla
- Vuessa on Komponenttikohtaiset CSS-määritykset.
- Vuen kumppanikirjastot on virallisesti tuettuja.

- Reactin oppimiskäyrä on jyrkempi. [5.]

3.4.3 Vue 3

Vue.js:n uusin versio Vue 3 on suuri ja myös paljon tunteita herättänyt uudistus.

Kompleksisien sovelluksien kehitystä helpottamaan on lisätty provide / inject -ominaisuus. Vue 2:ssa on muuttujia propsien avulla siirretty aina vanhemmalta lapsikomponentille ja taas sen komponentin lapsikomponentille aina niin kauan kun muuttujan käyttöpaikka tulee vastaan. Provide-funktiolla halutut muuttujat annetaan kaikkien sisäkkäisien komponenttien käyttöön (kuva 1). Inject-nimiseen listaan listataan muuttujat, joita komponentissa halutaan käyttää (kuva 2).

```
export default {
    name: "Vanhempi",
    components: {
      Lapsi,
    },
    data() {
      return {
        selectedFood: "pizza",
        foods: ["pizza", "burger", "hot-dog"],
      };
    },
    provide() {
      return {
        foods: computed(() => this.selectedFood),
      };
    },
  };
```

Esimerkkikoodi 1. Provide-funktion käyttäminen

```
<template>
  <div>{{`My favorite food is ${food} 😊`}}</div>
</template>
<script>
  export default {
    name: "Lapsenlapsi",
    inject: ["food"],
  };
</script>
```

Esimerkkikoodi 2. Injectin käyttäminen

Aiemmin kirjastona ollut ominaisuus ja nyt Vue 3:n pääkirjastoon sisältyvä Teleport mahdollistaa elementtien siirtämisen komponentista toiseen. Ominaisuus mahdollistaa vapaamman ja jossain tapauksessa siistimmän komponenttirakenteen. Teleport-elementin ”to” -propsiin asetetaan elementin nimi, luokka, id tai hakusana, jonka perusteella Teleport päättelee, minkä elementin sisään Teleport-elementin sisältämät elementit asetetaan.

Vue 3 mahdollistaa myös usean elementin asettamisen templaten sisälle (kuva 3). Aikaisemmin, tämän ominaisuuden puuttuessa, komponenttien rakenteet saattoivat ruuhkautua, koska elementtejä ei voitu siirtää lapsielementtiin rikkomatta elementtirakennetta. Ominaisuutta kutsutaan nimellä ” Fragments” (= palaset).

```
<template>
  <label :for="test>{{ label }}</label>
  <input :type="checkbox" :id="test" :name="test" />
</template>
```

Esimerkkikoodi 3. Fragments Vue 3 komponentissa

Global API on muuttunut hieman. Sovelluksen alustus tehdään Vue 3:ssa createApp-funktion avulla. Se mahdollistaa sovelluskohtaisten directive-määritykset.

```
const myApp = Vue.createApp({})

myApp.directive('focus', {
  mounted(el) {
    el.focus()
  }
})
```

Koodiesimerkki 4. Sovelluskohtainen direktiivimääritys

Events API on riisuttu ominaisuuksista Vue 3:ssa. Ainoa kuunneltava tapahtuma on lapsikomponentin lähettämä $emit, jota käytetäänkin usein. [6.]

Composition API on Vue 3:n suurin muutos. Komponenttirakenteen voi kirjoittaa funktiopohjaisemmin, ja se muistuttaa Reactin Hooks-rakennetta. Vue 2:n komponenttirakenteessa erilaiset muuttujat ja funktiot ovat omissa lokeroissaan.

Composition API:ssa muuttujat ja funktiot voivat asettaa setup-funktion sisälle, miten haluaa. [7.]

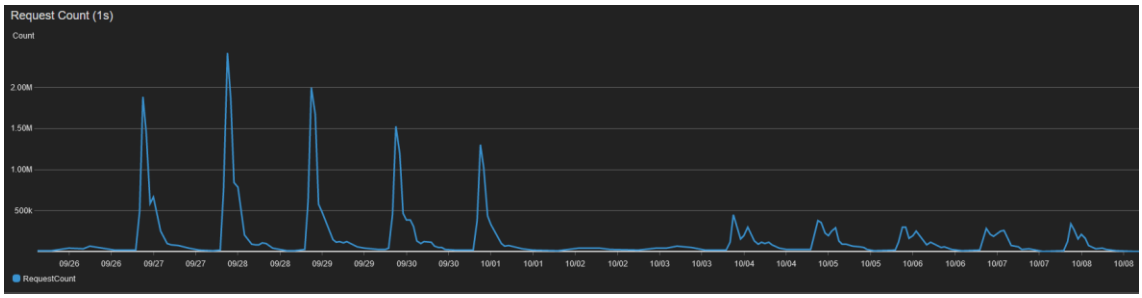
```
<script>
  import { onMounted } from 'vue'
  export default {
    setup() {
      const number = ref(0);

      const addThree = () => {
        number.value += 3;
      }

      onMounted(() => {
        console.log('Im ready')
      });

      return {
        number,
        addThree
      }
    }
  }
</script>
```

Koodiesimerkki 5. Komponenttirakenne Composition API:ssa.

4 Digikokeiden oppilaan kirjautumissivujen komponentit

4.1 Suunnittelu

Normaalisti Cloubin tuotteisiin kirjaudutaan jaettujen käyttäjätunnusten (sähköposti, salasana) tai Cloubiin rakennetun rajapinnan avulla.
Oppilaan kirjautumisvaihtoehdoiksi digikokeisiin nämä eivät sovellu. Syitä tähän ovat tietyt tekniset reunaehdot sekä käytön kankeus luokkahuonetilanteessa. Kokeet järjestetään usein samanaikaisesti. Tämän takia digikokeet ovat suurten kävijäpiikkien rasituksen alaisena, mutta osan aikaa kuorma on olematon.



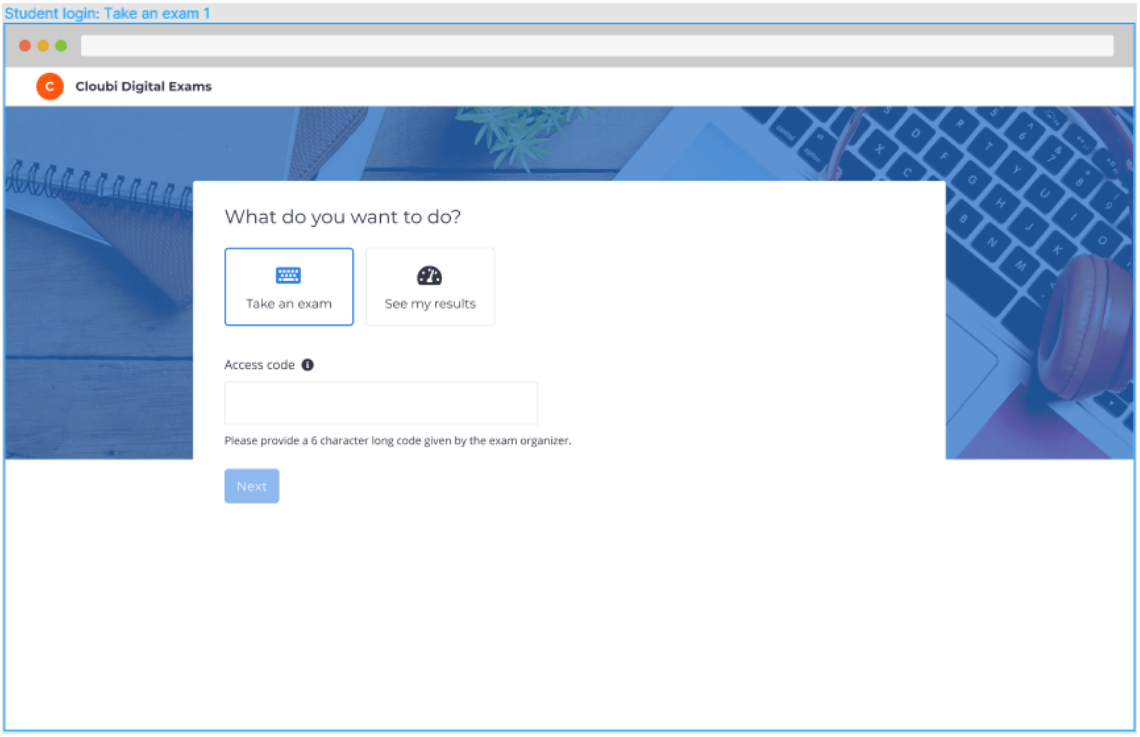
Kuva 2. Digikoepalvelimen kutsut koe- ja normaaliviikolla (kutsua tunnissa).

Digikokeita ei ole suunniteltu käyttöön jokaiselle palvelimelle, joten irrallinen kirjautumissivu muusta Cloubista sopii parhaiten tarpeisiimme.

Näin ollen päätimme oppilaita varten digikokeisiin rakentaa omat kirjautumissivut. Lähtökohtana ovat mahdollisimman yksinkertaiset ja helpot kirjautumissivut, johon esitietona oppilas tarvitsee vain opettajan jakaman pääsykoodin. Muusta sovelluksesta erillään olevan sivuston toivotaan kestävän paremmin kävijäpiikkejä ja vähentävän siten ongelmia ja tyytymättömyyttä.

Kirjautumissivujen on oltava kustomoitavissa helposti monelle eri asiakkaalle. Kustomoinnin piiriin kuuluvat näkymän tekstit, kuvat ja elementtien väritys. Eli vain peruspiirteet ja funktionaalisuus ovat mahdollisesti asiakkaiden kesken yhteneväisiä.

Koska kirjautumissivut ovat osa seuraavan sukupolven digikokeet-projektia, sen suunnittelu tapahtui osana digikokeiden suunnittelua. Design-tiimi valmisteli hyvissä ajoin Figma-sovelluksella suunnitelmat projektia varten, joten toteutuksen suunnittelu oli helpompaa.



Kuva 3. Kirjautumissivun aloitusnäkymän Figma-design

Kirjautumissivujen teknisessä suunnittelukokouksessa tekniikkavaihtoehtoina olivat ilman ohjelmistokehystä rakennettava sovellus sekä Vue 2:a tai Vue 3:a hyödyntävä sovellus. Ilman ohjelmistokehystä rakennettavan sovelluksen etuja ei juuri löytynyt, joten se vaihtoehto ei päässyt jatkoon. Vue 2:n ja Vue 3:n välillä mietittiin etuja. Vue 2 on ollut käytössä Cloubin käyttöliittymäosissa useita vuosia ja sen toimivuudesta on paljon näyttöä. Myös kaikki Cloubin ohjelmoijat osaavat sen avulla kehittämisen.

Vue 3 oli ollut käytössä vain yhdessä Cloubin projektissa noin vuoden ajan, ja vain osa kehittäjistä oli ollut kosketuksissa sen kanssa. Se on kuitenkin todettu toimivaksi ja siirtyminen Vue 2 käytöstä kivuttomaksi.

Käyttöliittymäosat rakennetaan useasti käyttäen jonkunlaista kehitystyökalua, joka tarjoaa kehityspalvelimen, johon voi lennossa tehdä muutoksia. Vue 2:lla tehtyjen käyttöliittymien käytössä on ollut Webpack-kehitystyökalu. Se on ollut toimiva työkalu, mutta se ei ole kovin nopea. Vue 3:n valinta mahdollistaa Viten

tehokkaan käytön. Vite on kehitystyökalu, joka on kehitetty ratkaisemaan käyttöliittymäpuolen ongelmia. Se tarjoaa kehityspalvelimen sekä koodin paketoinnin. Viten avulla Vue 3 -projektipohjan saa myös luotua yhdellä komentorivikäskyllä. Viten kehityspalvelin on käytössä ollutta Webpack-kehityspalvelinta huomattavasti nopeampi pystyttämisen ja lennossa päivittämisen osalta. Tämän ansiosta koodin testattavuus on paljon nopeampaa käsin ja automaattisesti. [8.]

Tämän lisäksi myös toisessa Vue 3:a käyttävässä projektissa olevia komponentteja voitaisiin käyttää hyväksi.

4.2 Sovelluksen rakentaminen

Sovelluksen rakentaminen alkoi loogisessa järjestyksessä luotujen tehtävänantojen määrittelemänä. Sovellukselle rakennettiin Vue 3 -projektin perusrunko, jonka jälkeen aloitimme rekisteröitymislomakkeen parissa.

4.2.1 Lomakekenttä

Oppilaan perustietoja varten tarvitsimme lomakekenttäkomponentin. Lomakkeen esivaatimuksina olivat:

- Oppilaan nimen ja sähköpostiosoitteen validius tulee tarkastaa.
- Epävalidin kentän tulee antaa palautetta käyttäjälle.
- Kahden sähköpostiosoitteen tulee olla samat.
- Epävalidit kentät estävät lomakkeen lähettämisen.

Esitietojen pohjalta päätimme ottaa toisesta Vue 3-sovelluksestamme pohjan lomakekentälle, jota lähdimme muokkaamaan käyttötarkoitukseen sopivaksi.

Pohjaan tuli lisätä validointiominaisuuksia ja tyyliä tuli muokata. Komponentin elementtirakenne koostuu vakio-input-elementistä, johon on asetettu funktiot kuuntelemaan lomakekentän tietojen syöttämisestä aiheutuvia tapahtumia. Merkin kirjoittaminen ja kohdistuksen poistaminen lomakekentästä aiheuttaa viiveellisen validoinnin, ja tekstin liittäminen aiheuttaa oman lähes viiveettömän validoinnin.

Setup-osiossa sijaitsevat validointifunktiot. Ne päivittävät lomakkeen tekstin emokomponentille ja kutsuvat emokomponentissa komponentille asetettua funktiota, jossa validointi tapahtuu. Validointifunktiot ovat tietokenttäkohtaisia.

```
<template>
  <div class="de-text-field">
    <input
      type="text"
      :id="id"
      :autocomplete="autocomplete"
      :class="['text-input', { error: error }]"
      v-model="name"
      @keyup="validate"
      @blur="validate"
      @paste="validateInstantly"
    />
    <custom-transition>
      <template #content>
        <font-awesome-icon
          v-show="error === false"
          :icon="faCheck"
          fixed-width
        />
      </template>
    </custom-transition>
    <custom-transition>
      <template #content>
        <font-awesome-icon
          v-show="error === true"
          :icon="faTimes"
          fixed-width
        />
      </template>
    </custom-transition>
  </div>
</template>
<script type="ts">

import { defineComponent, reactive, toRefs, ref } from 'vue';
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome';
import { faCheck, faTimes } from '@fortawesome/free-solid-svg-icons';
import { default as CustomTransition } from "../transition/CustomTran-
sition.vue";
import { default as debounce } from '../../utils/debounce';

export default defineComponent({
  props: {
    vModel: String,
    autocomplete: String,
    error: Boolean,
    id: String
  },
  components: {
    FontAwesomeIcon,
    CustomTransition
  },
  emits: ["update:modelValue", "validate"],
  setup(props, {emit}) {
    const options = reactive({
      name: props.vModel
    });

    //Paste will send also keyup event without this
    const canBeValidated = ref(true);
```

```
    const validate = debounce(() => {
        if (canBeValidated.value) {
          emit("update:modelValue", options.name);
          emit("validate");
        } else {
          canBeValidated.value = true;
        }
    }, 100);

    const validateInstantly = debounce(() => {
      canBeValidated.value = false;
      emit("update:modelValue", options.name);
      emit("validate");
    }, 10);


    return {
      faCheck,
      faTimes,
      validate,
      validateInstantly,
      ...toRefs(options)
    }
  }
});
</script>
```

Esimerkkikoodi 4. Lomakenttäkomponentin template- ja script-osio

4.2.2 Painike

Toinen sovelluksessa paljon käytetty elementti on painike. Painikeen avulla sovelluksessa navigoidaan ja vahvistetaan asioita. Painikkeen tekstin, taustavärin reunavärin ja kohdistusreunan väri on kustomoitavissa asiakkaille. Painikkeesta tuli olla myös 4 eri tyylivaihtoehtoa eri käyttötarkoituksia varten.

Eri painiketyylit määriteltiin omaan tiedostoonsa, josta ne tuodaan tarvittavaan Vue-komponenttiin. Tyyppien määrittely helpottaa etenkin projektin uusien ohjelmoijien työtä.

Painikkeen rakenne on pidetty mahdollisimman lähellä natiivia HTML-painiketta. Tämän ansiosta saavutettavuus on helpompi toteuttaa, koska natiivit elementit ovat oikein toteutettuna saavutettavia. Peruskomponentin lisäksi painikkeessa on ikoni ja slot-elementti halutulle elementille. Tämän sovelluksen tapauksessa elementti on tekstiä.

```
export enum BUTTON_TYPES {
  LOGIN_BTN = "login-button",
  EXAM_BTN = "exam-button",
  SECONDARY_BTN = "secondary-button",
  ICON_BTN = "icon-button"
}
```

Esimerkkikoodi 5. Painiketyyppivaihtoehdot

Painike-komponenttiin voidaan antaa propsina yleisesti Cloubissa käytössä olevan Font Awesome-ikonin, otsikkotekstin, totuusarvon painikkeen aktivisuudesta ja tyylivaihtoehdon. Otsikkoteksti asetetaan painikkeen aria-merkiksi ja otsikoksi saavutettavuutta tukemaan.

Tyyleissä käytetään kovakoodattujen tyylien lisäksi muuttujia, jotta asiakaskustomointi on helpompaa. Kaikki muuttuvat tyylit on listattu yhteen tiedostoon, mistä ne tuodaan tarvittavaan komponenttiin.

```
<template>
    <button
      :title="label"
      :class="[btnClass, { selected: selected }, {'no-outline': btn-
Class=='exam-button'}]"
      :aria-label="label"
      :disabled="isDisabled"
    >
      <FontAwesomeIcon v-if="icon" :icon="icon" size="lg"></FontAwe-
someIcon>
      <slot />
    </button>
</template>

<script lang="ts">
import { defineComponent, computed } from "vue";
import { FontAwesomeIcon } from "@fortawesome/vue-fontawesome";
import { IconDefinition } from "@fortawesome/free-solid-svg-icons";
import { BUTTON_TYPES } from "./ButtonTypes";

export default defineComponent({
  props: {
    label: String,
    selected: Boolean,
    icon: {
      type: Object as () => IconDefinition
    },
    disabled: Boolean,
    type: String as () => BUTTON_TYPES
  },
  components: {
    FontAwesomeIcon
  },
  setup(props, context) {
    const icon = computed(() => props.icon);
    const btnClass = computed(() => {
      if (Object.values(BUTTON_TYPES).find(o => o == props.type)) {
        return props.type;
      }
      return BUTTON_TYPES.LOGIN_BTN;
    });

    const isDisabled = computed(() => props.disabled === true);

    return {
      icon,
      btnClass,
      isDisabled
    };
  }
});
</script>

<style lang="postcss" scoped>


button {
  cursor: pointer;
  border-radius: 5px;
  font-family: var(--login-btn-font);
```

```
  font-size: 16px;
  transition: all 0.01s ease-in-out;
  border: .1rem solid rgba(255,255,255,0);
  min-width: max-content;
  outline: none;
  &:focus:not(.no-outline) {
    outline: .2rem solid var(--btn-focus-ol-color);
    outline-offset: 0.2rem;
  }
}

button.login-button {
  color: var(--login-btn-color);
  background-color: var(--login-btn-bgcolor);
  padding: 12px 15px;
}
button.secondary-button {
  color: var(--login-btn-bgcolor);
  background-color: var(--secondary-btn-bgcolor);
  padding: 12px 15px;
  border: var(--secondary-btn-border);
  &:hover {
    color: var(--secondary-btn-bgcolor);
    background-color: var(--login-btn-bgcolor);
    border: var(--secondary-btn-border);
  }
}
button.icon-button {
  color: var(--login-btn-bgcolor);
  background-color: transparent;
  padding: 5px;
  margin-left: 10px;
}

button.exam-button {
  color: var(--take-exam-btn-color);
  background-color: var(--take-exam-btn-bgcolor);
  padding: 13px 16px;
  border: none;
  display: grid;
  grid-row-gap: 0;
  justify-content: center;
  align-items: center;
  border: 1px solid var(--take-exam-btn-border-color);
  width: 165px;
  height: 100px;

  &.selected {
    border-color: var(--take-exam-btn-hover-color);
    border-width: 2px;
    padding: 12px 15px;
    & svg {
      color: var(--take-exam-btn-hover-color);
    }
  }
  &:hover, &:focus {
    border: .2rem solid var(--btn-hover-bg-color);
    padding: 12px 15px;
    & svg {
      color: var(--btn-hover-bg-color);
```

```
    }
  }
}
button.exam-button svg {
  margin: auto;
}
button.exam-button[disabled],
button.exam-button[disabled]:hover,
button.exam-button[disabled]:hover svg {
  color: var(--take-exam-btn-disabled-color);
  border: 2px solid var(--take-exam-btn-disabled-color);
}

.keyboard-navigation {
  & button:focus {
    outline: lightgreen solid 2px;
  }
}
button.login-button:hover {
  color: var(--login-btn-hover-color);
  background-color: var(--btn-hover-bg-color);
}
button.login-button[disabled] {
  color: var(--login-btn-disabled-color);
  background-color: var(--login-btn-disabled-bg-color);
  cursor: not-allowed;
  &:hover,
  &:focus,
  & > .btn-label {
    color: var(--login-btn-disabled-color);
  }
}

@media only screen and (max-width: 673px) {
  button.exam-button {
    height: 5.5rem;
    width: 100%;
    grid-template-columns: 4rem 1fr;
    text-align: left;
    grid-gap: 1rem;
    margin-bottom: 1.5rem;
  }
}
</style>
```

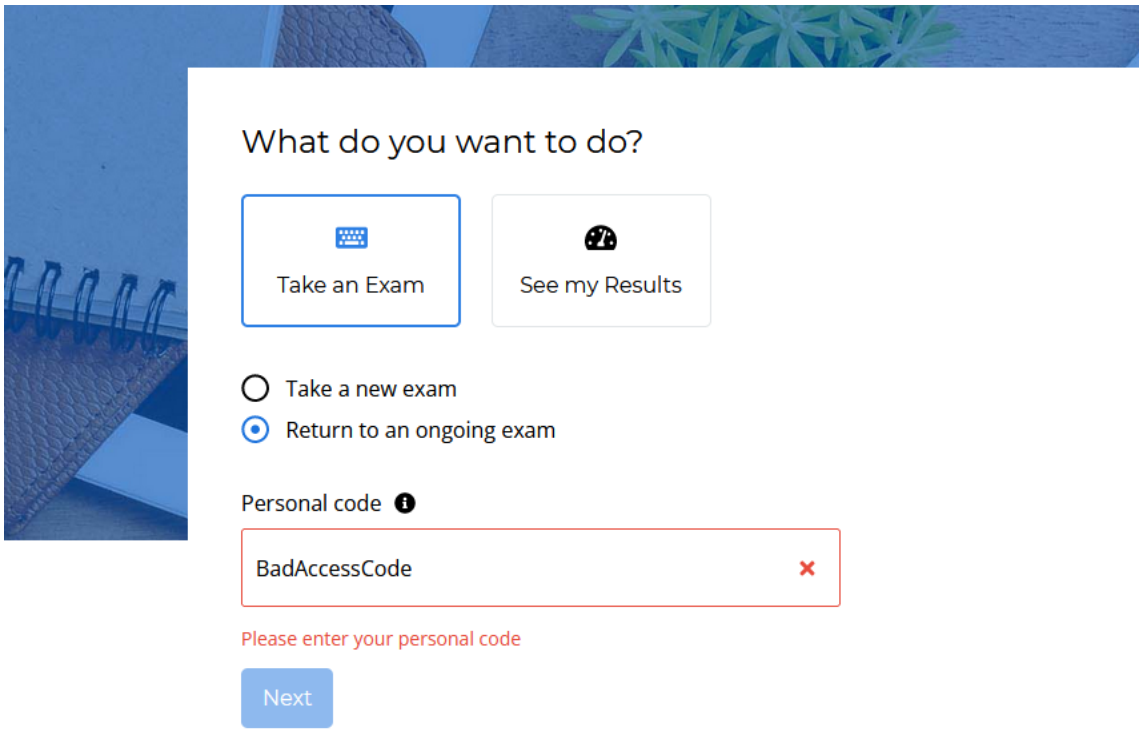
Esimerkkikoodi 6. Painike-komponentti

4.2.3 Lomakkeen ja painikkeen käyttäminen sovelluksessa

Molemmat esitellyt komponentit esiintyvät useamman kerran sovelluksessa. Painike on jokaisessa näkymässä ja lomakekenttä kahdessa eri näkymässä useana eri kenttänä.

Etusivulla molemmat komponentit ovat käytössä. Lomakekenttään voi laittaa joko kokeen koodin tai oppilaan henkilökohtaisen koodin. Molemmat koodit ovat 8-merkkisiä, joten niiden käyttöliittymäpuolen validointi on samanlainen. Käyttöliittymäpuolen validoinnin mentyä läpi voi yrittää siirtyä seuraavalle sivulle.

Koodi lähetetään palvelimelle tarkistettavaksi. REST-kutsuun mukaan lähtee tieto siitä, kumpaa koodia tarkistetaan. Se määräytyy siten, mitkä painikkeet on sivulla valittuna. Jos koodi on validi, käyttäjä siirtyy seuraavalle sivulle. Epävalidi koodi aiheuttaa virheilmoituksen käyttöliittymään (kuva 4). Validin koodin tapauksessa REST-kutsu palauttaa tarvittavat tiedot seuraavaa näkymää varten.



Kuva 4. Epävalidin koodin aiheuttama virheviesti.

```
<template>
  <div :class="['wizard-page', pageName]">
    <div class="de-login-grid" >
      <h2>{{ t("digital-exams-login-app-start-page-header") }}</h2>
      <div class="de-exam-buttons">
        <custom-button
          :type="BUTTON_TYPES.EXAM_BTN"
          :icon="faKeyboard"
          :selected="takeExam"
          @click="toggleExamAction(true)"
          data-cy="de-take-exam-button"
        >
          <span>{{ t("digital-exams-login-app-start-page-take-exam")
}}</span>
        </custom-button>
        <custom-button
          :type="BUTTON_TYPES.EXAM_BTN"
          :icon="faTachometerAlt"
          :selected="!takeExam"
          @click="toggleExamAction(false)"
          data-cy="de-see-results-button"
        >
          <span>{{ t("digital-exams-login-app-start-page-see-results")
}}</span>
        </custom-button>
        <radio-button
          v-if="takeExam"
          name="join-exam-select"
          :options="radioButtonOptions"
          :selected="selectedMode"
          @change="changeMode"
        />
      </div>

      <div class="de-login-to-exam">
        <span class="de-access-code">
          <span class="de-access-code-label">
            <label for="examcode" class="label-text" data-cy="de-ac-
cess-code-label-text">{{
              takeExam && selectedMode === "newExam" ? t("digital-ex-
ams-login-app-start-page-access-code")
                : t("digital-exams-login-app-start-page-personal-ac-
cess-code")
            }}</label>
            <tooltip id="code-tooltip">
              <template #content>
                <FontAwesomeIcon :icon="faInfoCircle" fixed-width />
              </template>
              <template #tooltip>
                <span id="code-tooltip">
                  {{ takeExam && selectedMode === "newExam" ? t("digi-
tal-exams-login-app-start-page-access-code-hint")
                    : t("digital-exams-login-app-start-page-personal-ac-
cess-code-hint")}}
                </span>
              </template>
            </tooltip>
          </span>
          <TextInput
            v-model="examCode"
```

```
            id="examcode"
            :error="showError"
            autocomplete="off"
            @validate="validateCode"
          />
        </span>
        <custom-transition>
          <template #content>
            <span v-show="showError" class="de-access-code-invalid">{{
              errorMessage
            }}</span>
          </template>
        </custom-transition>
        <span class="de-login-to-exam-btn">
          <custom-button :disabled="nextBtnDisabled" @click="nextBtn-
Clicked">{{
            t("digital-exams-login-app-next-button")
          }}</custom-button>
        </span>
      </div>
    </div>
    <LoginSpinner v-if="loading"/>
  </div>
</template>

<script lang="ts">
import { computed, defineComponent, inject, reactive, ref, toRefs }
from "vue";
import { default as CustomButton } from "../buttons/CustomButton.vue";
import  { default as LoginSpinner } from "../spinner/LoginSpin-
ner.vue";
import { FontAwesomeIcon } from "@fortawesome/vue-fontawesome";
import { BUTTON_TYPES } from "../buttons/ButtonTypes";
import { default as CustomTransition } from "../transition/CustomTran-
sition.vue";
import { default as Tooltip } from "../tooltip/Tooltip.vue";
import { default as debounce } from "../../utils/debounce";
import { EXAM_ACTION, StartPageState } from "../../types";
import { WizardNavigation } from "../../types";
import { useI18n } from "vue-i18n";
import { default as TextInput } from "../input/TextInput.vue";
import LoginApi from "../../api/LoginApi";
import {
  faTachometerAlt,
  faKeyboard,
  faInfoCircle
} from "@fortawesome/free-solid-svg-icons";
import { default as RadioButton} from "../buttons/RadioButton.vue";
import { RadioButtonOptions } from "../buttons/RadioButtonOptions";

export default defineComponent({
  name: "StartPage",
  props: {
    pageName: {
      type: String,
      required: true
    }
  },
  components: {
    CustomButton,
```

```
    FontAwesomeIcon,
    CustomTransition,
    Tooltip,
    TextInput,
    RadioButton,
    LoginSpinner
  },
  setup() {
    const loginApi = new LoginApi();
    const { t } = useI18n();

    const options = reactive<StartPageState>({
      showError: undefined,
      examAction: Number(
        localStorage.getItem("examAction") || EXAM_ACTION.TAKE_EXAM
      ),
      examCode: "",
      errorMessage: t("digital-exams-login-app-start-page-input-re-
minder"),
      selectedMode: localStorage.getItem("selectedMode") || "newExam",
      loading: false
    });

    const takeExam = computed(
      () => options.examAction === EXAM_ACTION.TAKE_EXAM || op-
tions.examAction === EXAM_ACTION.REJOIN_EXAM
    );

    const navigation = inject<WizardNavigation>("wizardNavigation");
    if (!navigation) {
      console.error("Missing navigation provider");
    }

    const validateCode = debounce(() => {
      options.errorMessage = t(options.examAction === EXAM_AC-
TION.TAKE_EXAM ?
        "digital-exams-login-app-start-page-input-reminder" : "digi-
tal-exams-login-app-start-page-input-reminder-personal"
      );
      options.examCode = options.examCode?.trim() || "";
      options.showError = options.examCode.length != 8;
    }, 100);

    const trimmedCode = computed(() => {
      return options.examCode?.trim() || ""
    });

    const nextBtnClicked = async () => {
      if (!options.examCode) {
        console.error("examCode not available");
        return;
      }
      options.loading = true;

      switch (options.examAction) {
        case EXAM_ACTION.TAKE_EXAM:
          loginApi
            .fetchExamMetadataByExamCode(trimmedCode.value)
            .then(response => {
              if (response.examId) {
```

```
                sessionStorage.examCode = trimmedCode.value;
                sessionStorage.examId = response.examId;
                sessionStorage.examStatus = response.examState?.state-
Type;
                navigation?.toNext({ examDetails: JSON.stringify(re-
sponse) });
              }
            })
            .catch(e => {
              console.error(e);
              options.showError = true;
              options.errorMessage = t(
                "digital-exams-login-app-start-page-invalid-code"
              );
              options.loading = false;
            });
          break;
        case EXAM_ACTION.REJOIN_EXAM:
        case EXAM_ACTION.SEE_RESULTS:
          loginApi
            .fetchExamMetadataByPersonalAccessCode(trimmedCode.value)
            .then(response => {
              if (response.examId) {
                sessionStorage.studentAccessCode = trimmedCode.value;
                sessionStorage.examCode = response.examCode;
                sessionStorage.examId = response.examId;
                sessionStorage.examStatus = response.examState?.state-
Type;
                sessionStorage.attendee = response.attendee;
                sessionStorage.examLoginLink = response.loginLink;
                navigation?.toPage('result-details', { examDetails:
JSON.stringify(response) });
              }
            })
            .catch(e => {
              console.error(e);
              options.showError = true;
              options.errorMessage = t(
                "digital-exams-login-app-start-page-invalid-code"
              );
              options.loading = false;
            });
          break;
      }
    };

    const nextBtnDisabled = computed(
      () => !navigation?.hasNextPage() || options.showError || !op-
tions.examCode || options.loading
    );

    const toggleExamAction = (takeExam: boolean) => {
      try {
        options.examAction = takeExam && options.selectedMode ===
"newExam"
          ? EXAM_ACTION.TAKE_EXAM
          : takeExam && options.selectedMode === "rejoinExam" ?
EXAM_ACTION.REJOIN_EXAM : EXAM_ACTION.SEE_RESULTS;
```

```
        localStorage.setItem("examAction", options.examAc-
tion.toString());
      } catch (e) {
        console.error("Unable to select exam action");
      }
    };


    const radioButtonOptions: RadioButtonOptions[] = [
      {
        value: "newExam",
        displayValue: "digital-exams-login-app-start-page-take-new-
exam",
      },
      {
        value: "rejoinExam",
        displayValue: "digital-exams-login-app-start-page-return-to-
exam",
      },

    ];


    const changeMode = (newMode: string) => {
      options.selectedMode = newMode;
      localStorage.setItem("selectedMode", newMode);

      const takeExam = options.examAction !== EXAM_ACTION.SEE_RESULTS;
      toggleExamAction(takeExam);
    }

    return {
      t,
      BUTTON_TYPES,
      faTachometerAlt,
      faKeyboard,
      faInfoCircle,
      validateCode,
      nextBtnDisabled,
      nextBtnClicked,
      toggleExamAction,
      ...toRefs(options),
      takeExam,
      radioButtonOptions,
      changeMode
    };
  }
});
</script>
```

Esimerkkikoodi 7. Aloitussivun template- ja script-osio

4.3 Käyttökokemukset

Kirjautumisnäkymä tuli tuotantovalmiiksi muutama viikko ennen loppukäyttäjien ensikosketusta. Ensikosketus tapahtui lukioiden ensimmäisellä koeviikolla, joka

olisi siis koko uusien digikokeiden tulitesti. Käyttäjämäärät olivat kymmenissä tuhansissa laskettavia ja tulivat isoina piikkeinä. Vaikka muista sovelluksen osista löytyi pieniä puutteita kovan rasituksen alla, kirjautumisnäkymästä ei tullut palautetta. Sovellusten tapauksessa saatu palaute on lähes aina vain negatiivista, joten voi sanoa julkaisun olleen todella onnistunut.

Kehittäjät eivät ole nähneet siirtymää Vue 2:sta Vue 3:een haastavana. Tämä on edesauttanut kirjautumisnäkymän nopeaa ja laadukasta rakentamista.

5. Yhteenveto

Insinöörityössä suunniteltiin ja rakennettiin kirjautumissivu Cloubin uusiin digikokeisiin. Tavoitteena oli saada itsenäisesti toimiva kirjautumissivusto oppilaiden kirjautumiskäyttöön käyttäen Vue 3 -ohjelmistokehystä. Lähtökohtana oli mahdollisimman yksinkertainen ja toimiva sivusto, jotta jokainen oppilas kykenee sitä käyttämään. Lisäksi oli otettava huomioon sivuston saavutettavuus, jotta erityistarpeiset oppilaat eivät jää ulkopuolelle. Toimivuus myös kaikenkokoisilla ruuduilla oli taattava.

Projektilla oli kiire ja tähän päälle projektin maali oli heti kesän jälkeen, joten lomat antoivat lisähaasteen projektille. Tästä huolimatta kirjautumissivusto oli jo lukion ensimmäisellä koeviikolla käytössä. Tätä tuki hyvä suunnittelu ja ajoissa valmiiksi tehdyt designit, jotka eivät jättäneet liikaa miettimistä kehittäjille.

Vue 3 osoitti olevansa toimiva työkalu käyttöliittymien rakentamiseen. Sen Vue 2:sta muuttuneet ominaisuudet eivät aiheuttaneet negatiivisia tuntemuksia kehittäjissä, vaan se otettiin hyvin vastaan. Henkilökohtaisesti toivon sille jatkoa myös tulevaisuuden projekteissa.

Kokonaisuudessaan projekti oli erittäin onnistunut. Vue 3:n käyttäminen auttoi rakentamaan kaikilla tavoilla laadukkaan sovelluksen. Sovellusta tai sen komponentteja mahdollisesti käytetään tulevaisuudessa myös muiden käyttäjien kirjautumiseen, koska sen toimivuus on ollut vakaata ja käyttö miellyttävää.

Lähteet

1 About us. Verkkoaineisto. Cloubi. https://cloubi.com/contact-us/. Luettu 12.9.2021.

2 Taloustiedot. Verkkoaineisto. Fonecta. https://www.finder.fi/Suunnittelutoimisto/Cloubi+Oy/Helsinki/yhteystiedot/2637657. Luettu 7.11.2021.

3 The web app development challenges 2020. Verkkoainesto. Rapidops Inc. https://www.rapidops.com/blog/top-challenges-in-web-app-development/. Luettu 12.9.2021.

4 Introduction. Verkkoaineisto. Vue. https://v3.vuejs.org/guide/introduction.html. Luettu 19.9.2021.

5 Comparison with Other Frameworks. Verkkoaineisto. Vue. https://vuejs.org/v2/guide/comparison.html. Luettu 19.9.2021.

6 What's New In Vue 3? Verkkoaineisto. Smashingmagazine https://www.smashingmagazine.com/2020/11/new-vue3-update/. Luettu 30.9.2021.

7 Vue 3 features & changes. Verkkoaineisto. Madewithvuejs. https://madewithvuejs.com/blog/vue-3-roundup/. Luettu 30.9.2021.

8 Why Vite. Verkkoaineista. Vitejs. https://vitejs.dev/guide/why.html. Luettu 7.11.2021.