OULUN AMMATTIKORKEAKOULU

Matias Koivumaa

WEB-SOVELLUS TYÖNTEKIJÄN PEREHDYTYKSEEN JA KÄYTTÖLIITTYMÄKIRJASTOJEN VERTAILU

WEB-SOVELLUS TYÖNTEKIJÄN PEREHDYTYKSEEN JA KÄYTTÖLIITTYMÄKIRJASTOJEN VERTAILU

Matias Koivumaa
Opinnäytetyö
Kevät 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Matias Koivumaa
Opinnäytetyön nimi: Web-sovellus työntekijän perehdytykseen
Työn ohjaaja: Meija Lohiniva
Työn valmistumislukukausi ja -vuosi: Kevät 2024 Sivumäärä: 32

Tämän opinnäytetyön aiheena oli luoda web-sovellus uusien työntekijöiden perehdytyksen tueksi. Sovelluksessa on erilliset näkymät työnantajalle sekä työntekijälle. Työnantajalla on mahdollisuus luoda työntekijälle perehdytyssuunnitelma, joka sisältää erilaisia tehtäviä ja seurata suunnitelman tehtävien etenemistä. Työntekijällä on mahdollisuus tarkastella hänelle määritettyä perehdytyssuunnitelmaa ja merkitä sen sisältämiä tehtäviä suoritetuiksi.

Sovelluksen käyttöliittymä luotiin käyttäen JavaScript-ohjelmointikieltä, Svelte- ja SvelteKit-kirjastoja sekä Visual Studio Code -kehitysympäristöä. Ohjelmistorajapinnan luontiin käytettiin C#-ohjelmointikieltä, ASP.NET-kirjastoa sekä Visual Studio 2022 -kehitysympäristöä. Tietokantana sovelluksessa toimii Microsoftin SQL Server.

Työn teoriaosuudessa tutustuttiin React- ja Svelte-käyttöliittymäkirjastojen historiaan sekä toimintaan. Niiden eroavaisuuksiin, vahvuuksiin sekä heikkouksiin perehdyttiin ja pohdittiin kumpi on sopivampi valinta missäkin tilanteessa.

Lopputuloksena saatiin toteutettua määrittelyitä ja suunnitelmia vastaava sovellus. Työn teoriaosuuden osalta tultiin siihen lopputulokseen, että Reactilla ja Sveltellä on molemmilla omat vahvuutensa sekä heikkoutensa. Se, kumpi on parempi vaihtoehto, riippuu suuresti sovelluksen ominaisuuksista, kuten sen koosta. Tämän opinnäytetyön sovellukseen Svelte oli hyvä valinta sovelluksen pienen koon ja yksinkertaisuuden vuoksi. Sovellus tehtiin tätä opinnäytetyötä varten, eikä se tule kaupalliseen käyttöön.

Asiasanat: Web-sovellukset, JavaScript, Svelte, käyttöliittymäkirjastot

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author: Matias Koivumaa
Title of thesis: Web application for employee orientation
Supervisor: Meija Lohiniva
Term and year when the thesis was submitted: spring 2024
Number of pages: 32

The subject of this thesis was to create a web-application for new employee orientation. The application has separate views for both the employer and the employee. The employer can create an orientation plan for the employee, add different tasks to the plan and monitor the employee's progress on the tasks. The employee can view the plan and mark its tasks completed.

The user interface of the application was created using JavaScript programming language, Svelte and SvelteKit libraries and Visual Studio Code editor. The Application Programming Interface (API) was created using C# programming language, ASP.NET library and Visual Studio 2022 editor. The database in use is Microsoft SQL Server.

The theoretical part of the thesis covers the history and functionality of React and Svelte user interface libraries. Their differences, strengths and weaknesses were reviewed, and it was discussed which one is the more suitable choice in different situations.

The result is an application that matches the specifications and plans. In the theoretical part of the thesis, we concluded that React and Svelte both have their strengths and weaknesses. Which one of them is the better choice depends on the application’s qualities, for example, its size. In the application created for this thesis, Svelte was a good choice because of the small size and simplicity of the application. The application was created for this thesis, and it will not be used commercially.

Keywords: Web applications, JavaScript, Svelte, user interface libraries

SISÄLLYS

1 JOHDANTO

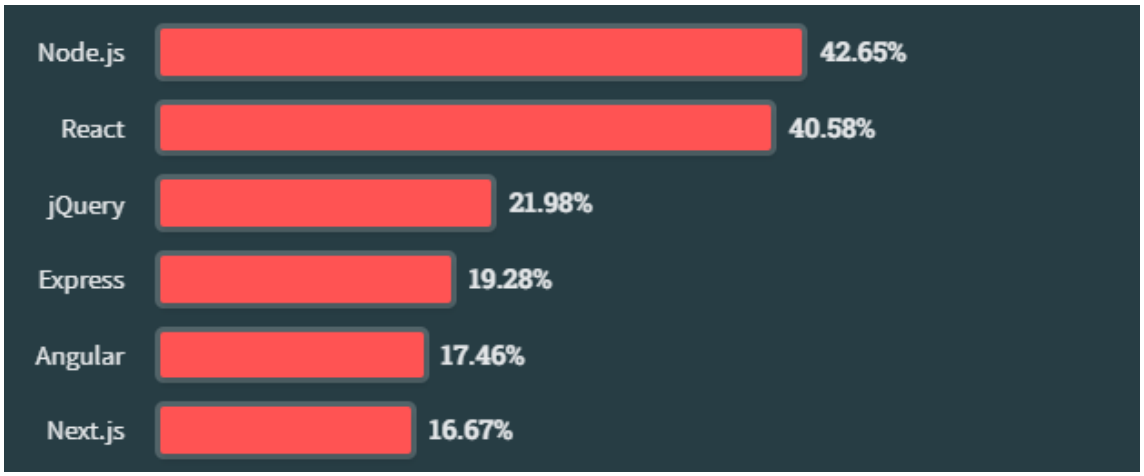
Opinnäytetyön tavoitteena on toteuttaa uuden työntekijän perehdyttämiseen tarkoitettu web-sovellus. Uusien työntekijöiden perehdytys jää usein varsin puutteelliseksi, eikä siinä ole aina selkeitä tavoitteita. Tämän sovelluksen tarkoitus on helpottaa selkeiden perehdytysaskelten luomista ja varmistaa, että uusi työntekijä saa tarpeeksi kattavan perehdytyksen. Sovelluksen käyttäjiä ovat työnantajat ja työntekijät. Työnantaja voi sovelluksessa luoda perehdytyssuunnitelmia sekä seurata niiden edistymistä. Työntekijä voi tarkastella hänelle määritettyä suunnitelmaa. Sovelluksen käyttöliittymä koostuu kirjautumisnäkymästä, suunnitelmien hallintanäkymästä ja suunnitelmien tarkastelunäkymästä.

Perehdytyssuunnitelmien hallintanäkymä on ainoastaan työnantajan käytettävissä, ja sen avulla työnantaja voi luoda suunnitelman eri työntekijöille sekä lisätä suunnitelmaan tehtäviä ja alitehtäviä. Perehdytyssuunnitelmien tarkastelunäkymässä työntekijä voi tarkastella hänelle itselleen määritettyä suunnitelmaa sekä merkitä sen sisältämiä tehtäviä suoritetuiksi.

Selainohjelmointiin käytetään JavaScript-ohjelmointikieltä sekä Svelte- ja SvelteKit-kirjastoja. Palvelinohjelmointiin käytetään C#-ohjelmointikieltä ja ASP.NET Core -kirjastoa. Sovellus käyttää Microsoftin SQL Server -relaatiotietokannanhallintajärjestelmää. Toteutuksessa käytetyt ohjelmointiympäristöt ovat Visual Studio 2022 ja Visual Studio Code.

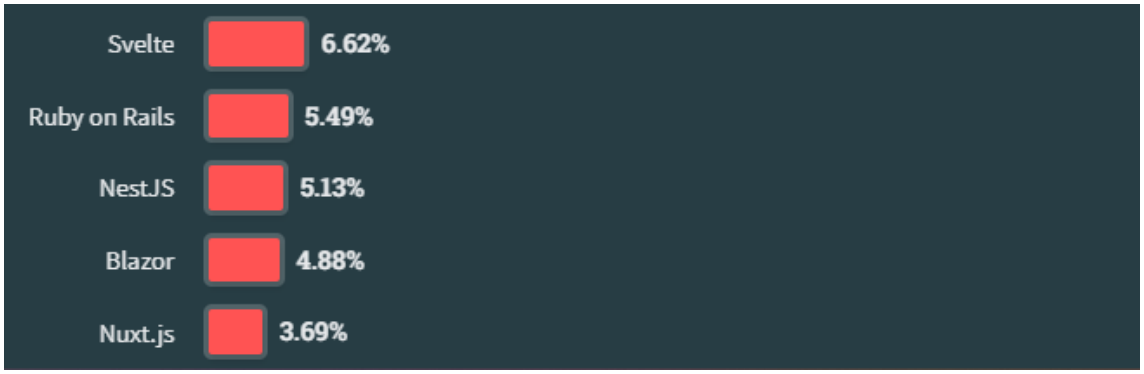
2 KÄYTTÖLIITTYMÄKIRJASTOT

Web-ohjelmistokehykset ovat ennalta koottuja paketteja, jotka sisältävät kirjastoja, ohjelmistotyökaluja ja käytäntöjä. Niiden tarkoitus on tukea sovellusten kehitystä. Niiden tarjoamia hyötyjä ovat esimerkiksi kehityksen nopeutuminen, sisäänrakennetut turvallisuusominaisuudet ja sovelluksen parempi suorituskyky. Web-ohjelmistokehyksiä on sekä käyttöliittymä- että palvelinkehitystä varten. Web-ohjelmistokehyksiä ja kirjastoja on saatavilla lukuisia eri vaihtoehtoja. Suosituimpien teknologioiden joukossa on selainohjelmoinnissa käytettävä React-kirjasto (kuva 1). (Chatterjee 2023.)



KUVA 1. Suosituimmat web-ohjelmistokehykset ja teknologiat vuonna 2023 (Stack Overflow 2023)

Yksi monista vaihtoehdoista Reactille on Svelte. Svelte on uudempi, ja Reactiin verrattuna huomattavasti vähemmän käytetty kirjasto (kuva 2).



KUVA 2. Suosituimmat web-ohjelmistokehykset ja teknologiat vuonna 2023 (Stack Overflow 2023)

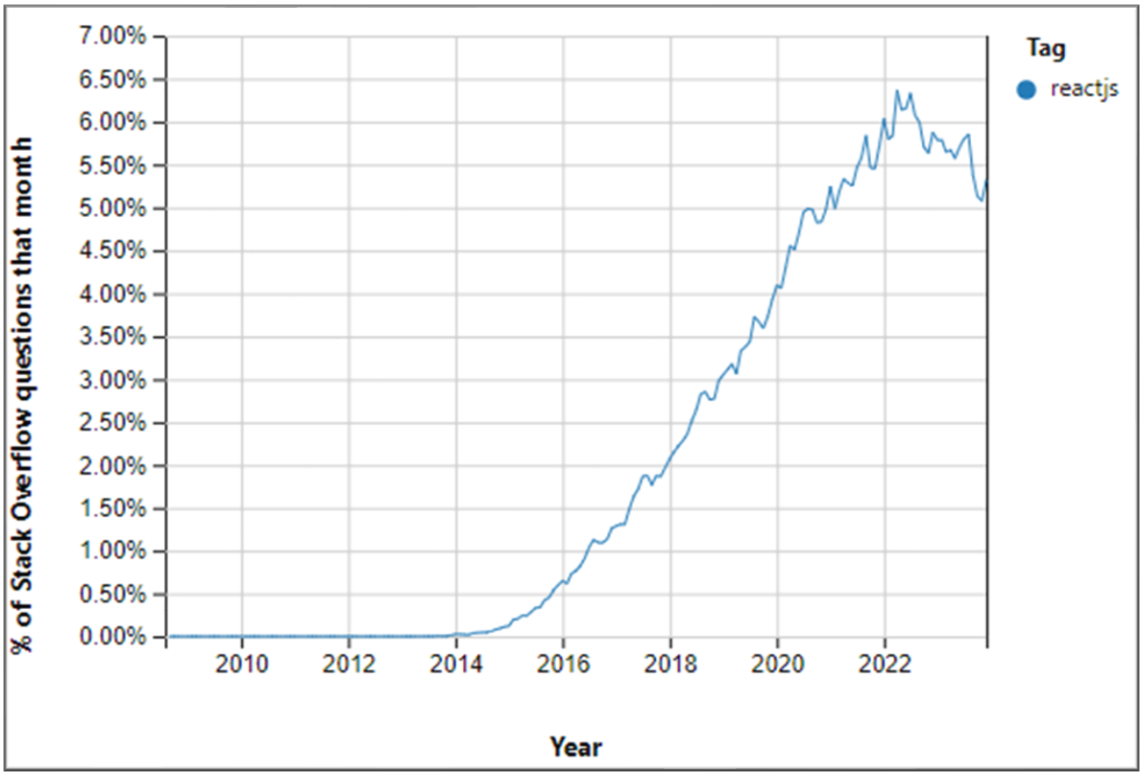
Tässä opinnäytetyössä selainohjelmointiin on valittu Svelte, jota verrataan Reactiin ja yritetään selvittää kumpi on parempi vaihtoehto selainohjelmoinnin tueksi.

2.1 React

React on Facebookin kehittämä avoimen lähdekoodin JavaScript-kirjasto. Sen avulla voi luoda käyttöliittymiä sekä web- että mobiilisovelluksille. React sijoittui Stack Overflown vuonna 2023 tekemässä kyselyssä suosituimpien web-ohjelmistokehysten ja teknologioiden osiossa toiseksi (kuva 1). Syitä siihen, että React on niin suosittu, ovat muun muassa sen helppokäyttöisyys, suorituskyky sekä se, että React huomattavasti yksinkertaistaa sovellusten kehitystyötä. Myös se, että Reactilla voi web-sovellusten lisäksi luoda myös mobiilisovelluksia, on huomionarvoista pohdittaessa miksi se on niin ylivoimaisen suosittu moniin muihin teknologioihin verrattuna. (A-Team Global 2023.)

2.1.1 Historia

Reactin loi Jordan Walke vuonna 2011 työskennellessään Facebookilla. Ensin se otettiin käyttöön ainoastaan Facebookin oman sivuston uutissyötteessä. Vuonna 2012 sitä alettiin käyttää myös Instagram-sovelluksessa. Vuonna 2013 Facebook teki Reactista avoimen lähdekoodin kirjaston, minkä myötä Reactin suosio on kasvanut räjähdysmäisesti (kuva 3). (Banks & Porcello 2020, luku 1.)

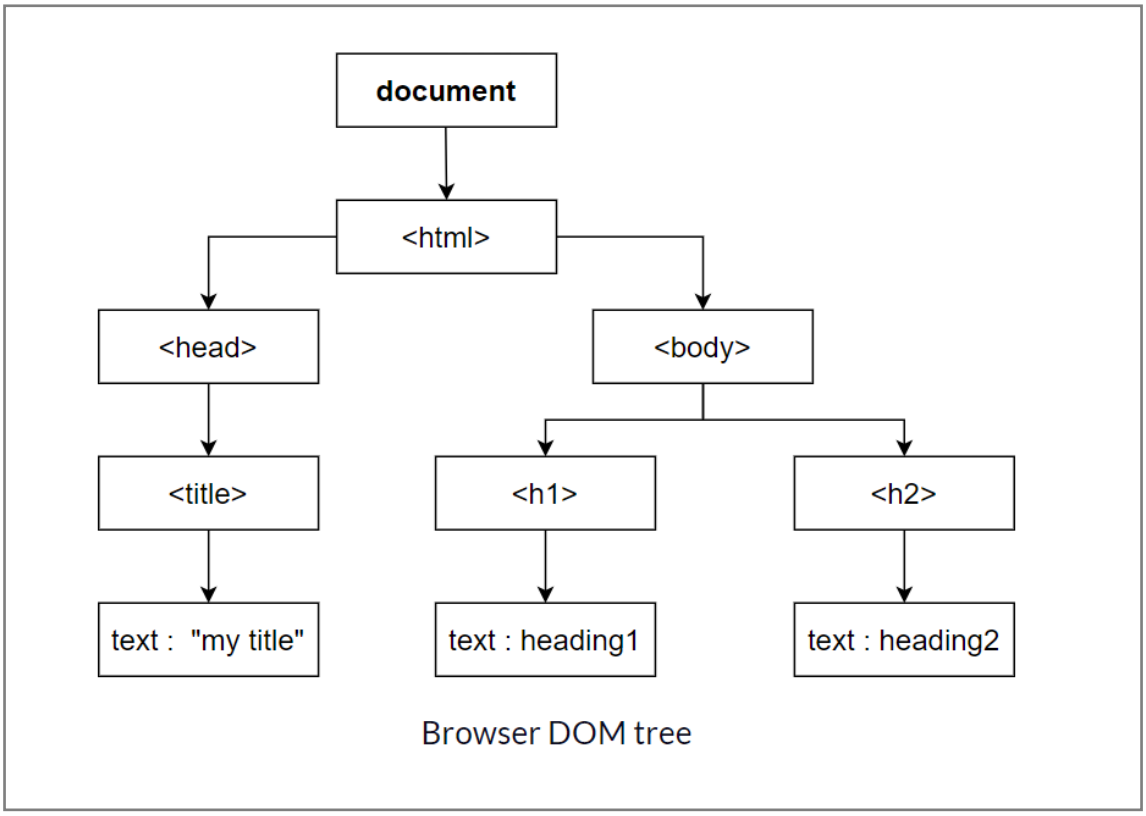


KUVA 3. React-aiheisten Stack Overflow -kysymysten osuus kuukausittain (Stack Overflow 2023)

Tammikuussa 2024 Reactia käyttäviä nettisivuja oli yli 13,7 miljoonaa (BuiltWith 2024). Nykyään Reactia käyttää sovelluksessaan esimerkiksi suoratoistopalvelu Netflix. (Banks & Porcello 2020, luku 1.)

2.1.2 Toiminta

React piirtää ja päivittää sovelluksen käyttöliittymää lisäämällä komponentteja selaimen Document Object Modeliin (DOM). DOM on nettisivun käyttöliittymän runko. Se koostuu HTML-elementeistä, jotka muodostavat eräänlaisen puukaavion (kuva 4). React ei suoraan muokkaa selaimen DOM:ia, koska sen uudelleenpiirtäminen vie huomattavasti aikaa. Sen sijaan React muokkaa virtuaalista DOM:ia, joka on kopio selaimen DOM:ista. React tekee muutokset ensin virtuaaliseen DOM:iin. Tämän jälkeen virtuaalista DOM:ia ja selaimen DOM:ia verrataan toisiinsa, ja selaimen DOM päivitetään vastaamaan virtuaalista DOM:ia. Selaimessa päivitetään ainoastaan muuttuneet elementit, mikä parantaa huomattavasti sovelluksen nopeutta ja tehokkuutta. (KS 2022.)



KUVA 4. Document Object Model (DOM) (KS 2022)

React-sovellukset koostuvat komponenteista. React-komponentit ovat erilaisia uudelleenkäytettäviä moduuleita. Komponentit koostuvat yleensä import-lauseista, komponentin logiikasta ja JSX-elementeistä. Import-lauseiden avulla komponentti voi käyttää esimerkiksi muualla sovelluksessa määriteltyä koodia. (MDN web docs 2024.)

Komponentin logiikka kirjoitetaan JavaScript-ohjelmointikielellä, ja se sijaitsee JavaScript-funktiossa, joka ajetaan silloin kun komponenttia kutsutaan. Logiikka voi muodostua esimerkiksi muuttujien määrityksistä ja erilaisista funktioista. Komponentin näkyvät osat piirretään return-funktion palauttamasta JSX-koodista, joka voi sisältää myös aaltosulkeilla ympäröityä JavaScript-koodia (kuva 5). (MDN web docs 2024.)

```jsx
import "./App.css";

export default function App() {
    let count = 0;

    function increment() {
        count++;
    }

    return (
        <div>
            <h1>Count: {count}</h1>
            <button onClick={increment}>Increment</button>
        </div>
    );
}
```

KUVA 5. React-komponentti

JavaScript XML, lyhyesti JSX, mahdollistaa HTML-koodin kirjoittamisen React-komponenteissa kääntämällä sen React-elementeiksi. JSX:n käyttö ei ole pakollista, mutta se helpottaa ja nopeuttaa työskentelyä, sillä ilman sitä HTML-elementit täytyy kääntää manuaalisesti käyttäen Reactin createElement- tai appendChild -metodeja. (W3Schools 2024.)

Yksi Reactin erikoisuuksista ovat koukut joita ovat esimerkiksi useState, useEffect ja useMemo. UseState-koukun avulla voidaan luoda state-muuttuja, jolle määritetyn arvon muuttuessa kaikki sitä käyttävät sovelluksen osat uudelleenpiirtyvät. UseState-koukkua käyttämällä voidaan luoda koko sovelluksen tilanhallinta, eikä siihen enää tarvita ulkopuolista tilanhallintakirjastoa (kuva 6). (React 2024.)

```
import { useState } from "react";

export default function App() {
    const [count, setCount] = useState(0);

    function handleClick() {
        setCount(count + 1);
    }

    return (
        <div>
            <p>You clicked {count} times</p>

            <button onClick={handleClick}>Increment</button>
        </div>
    );
}
```

KUVA 6. UseState-koukun käyttö React-komponentissa

UseEffect-koukku on tarkoitettu käytettäväksi silloin, kun komponentista halutaan luoda yhteys johonkin ulkoiseen järjestelmään. UseEffect-koukkuun syötetään JavaScript-koodista muodostuva logiikka sekä hakasulkeissa oleva viittaus johonkin muuttujaan. Koukkuun syötetty logiikka ajetaan silloin kun hakasulkeissa olevan muuttujan arvo muuttuu (kuva 7). (React 2024.)

```
import { useEffect } from "react";

function ChatRoom({ roomId }) {
    useEffect(() => {
        const connection = createConnection(roomId);
        connection.connect();
        return () => connection.disconnect();
    }, [roomId]);
}
```

KUVA 7. UseEffect-koukun käyttö React-komponentissa

UseMemo-koukun avulla voidaan tallettaa jokin arvo selaimen välimuistiin. Koukkua käytetään yleensä silloin kun komponentissa suoritetaan jokin järjestelmää kuormittava laskutoimitus, jota ei haluta suorittaa turhaan uudelleen. Koukkuun syötetään funktio, jossa laskutoimitus suoritetaan, sekä laskutoimituksessa käytettävät reaktiiviset arvot. (kuva 8.) (React 2024.)

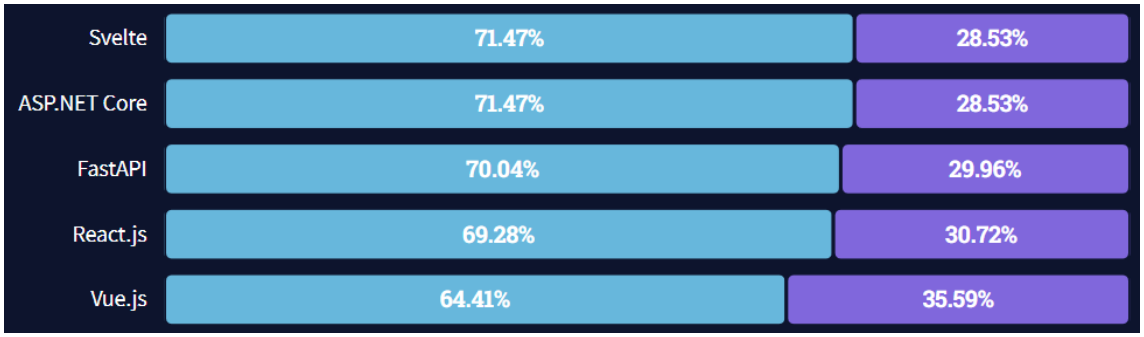
```
import { useMemo } from "react";

function App() {
    const visibleTodos = useMemo(
        () => filterTodos(todos, tab),
        [todos, tab]
      );
}
```

KUVA 8. UseMemo-koukun käyttö React-komponentissa

Funktio ajetaan uudelleen vain silloin kun sille hakasulkeissa syötetyt reaktiiviset arvot muuttuvat. (React 2024.)

2.2 Svelte

Svelte on Rich Harrisin vuonna 2016 julkaisema avoimen lähdekoodin ohjelmistokehys, joka on tarkoitettu web-sovellusten kehittämiseen. Svelte äänestettiin Stack Overflown vuoden 2021 kyselyssä kaikkein rakastetuimmaksi web-sovelluskehykseksi (kuva 9). (Holthausen 2022, luku 1.)



KUVA 9. Rakastetuimmat ja kamalimmat web-ohjelmistokehitykset vuonna 2021 (Stack Overflow 2021)

Svelten vahvuuksiin kuuluvat muun muassa helppokäyttöisyys, hyvä suorituskyky sekä suuri määrä valmiiksi mukana olevia ominaisuuksia, kuten yksinkertainen ratkaisu tilanhallintaan (Holthausen 2022, luku 1).

2.2.1 Historia

Aloittaessaan Svelten kehityksen loppuvuodesta 2016 Rich Harris työskenteli brittiläiselle sanomalehdelle The Guardianille. Siellä hänen työnkuvaansa kuului interaktiivisten visualisaatioiden luominen sanomalehden nettisivuille. Tämän työn helpottamiseksi hän halusi kehittää työkalun, jonka avulla visualisaatioiden luominen oli sekä nopeaa että tarpeeksi helppoa hänen vähemmän tekniikkataitoisillekin kollegoille. (Holthausen 2022, luku 1.)

Vaikka Svelten ensimmäinen versio julkaistiinkin jo vuonna 2016, pääsi se web-kehittäjien huomioon maailmanlaajuisesti toden teolla vasta vuonna 2019 Svelten kolmannen version julkaisun myötä. Kolmannessa versiossa oli keskitytty erityisesti Svelten saavutettavuuteen sekä kehityskokemukseen. Svelte on käytössä todella suurissa yhtiöissä, kuten musiikkipalvelu Spotifyssa sekä huonekalujätti IKEA:lla. Nykyään Rich Harris työskentelee Svelten parissa täysiaikaisesti. (Holthausen 2022, luku 1.)

2.2.2 Toiminta

Svelte-sovellukset koostuvat yhdestä tai useammasta komponentista, jotka käännetään pieniksi JavaScript-moduuleiksi. Komponentit voivat koostua kolmesta osiosta, joista kaikki ovat kuitenkin valinnaisia. Nämä osiot ovat script, styles ja markup (kuva 10). (Learn Svelte 2023, osa 1.)

```
<script>
    let name = 'Svelte';
</script>

<h1>Hello {name}!</h1>

<style>
    h1 {
        color: blue;
    }
</style>
```

KUVA 10. Svelte-komponentti

Script-osio ajetaan kun instanssi komponentista luodaan. Se sisältää JavaScript-ohjelmointikielellä kirjoitettua koodia. Script-osioihin liittyen on olemassa neljä eri sääntöä. Ensimmäinen sääntö on avainsana export. Jos lisää export-avainsanan ennen muuttujan julistamista, on muuttuja käytettävissä myös kaikilla muilla komponenteilla, jotka perivät komponentin, jossa muuttuja on julistettu. Toinen säännöistä on se, että arvojen asettamiset muuttujille ovat reaktiiviisia. Tämä tarkoittaa sitä, että kun paikallisesti julistetulle muuttujalle asettaa arvon, komponentin tila muuttuu ja komponentti piirtyy uudelleen. Tämän tapahtuessa script-osion sisältämiä arvon asettavia komentoja ei kuitenkaan automaattisesti ajeta uudelleen. Kolmas sääntö on se, että ennen komentoa script-osiossa voidaan lisätä dollarin merkki ja kaksoispiste. Tällöin kyseinen komento ajetaan uudelleen silloin kun arvot, joista komento on riippuvainen, muuttuvat (kuva 11). (Svelte 2023. Svelte components.)

```
<script>
    export let person;

    $: ({ name } = person);
</script>
```

KUVA 11. Reaktiiviseksi merkitty komento

Neljäs ja viimeinen sääntö on se, että kun store-objektien arvoja käytetään, täytyy ennen viittausta objektiin lisätä dollarin merkki. Store-objektit ovat objekteja joihin voi kirjoittaa ja joista voi lukea arvoja mistä tahansa komponentista. Svelte-komponentti, jossa storen arvoja käytetään, luo ja katkaisee yhteyden store-objektiin automaattisesti. (Kuva 12.) (Svelte 2023. Svelte components.)

```
<script>
    import { writable } from 'svelte/store';

    const helloWorld = writable('');

    helloWorld.set('Hello World!');

    console.log($helloWorld);

</script>
```

KUVA 12. Store-objektin käyttö

Style-osio sisältää Cascading Style Sheets (CSS) -koodia. CSS-koodia käytetään nettisivujen tyylin ja pohjan muokkaamiseen. Sillä voi muokata esimerkiksi sivun värejä, fontteja ja asettelua. (MDN web docs 2024. CSS first steps.)

Tyylejä voi asettaa esimerkiksi kaikille tietyn tyyppisille HTML-elementeille tai globaalisti koko komponentille (kuva 13). Svelte-komponentissa voi olla ainoastaan yksi style-osio. (Svelte 2023. Svelte components.)

```
<style>
    h1 {
        color: red;
        font-size: 1.5em;
    }

    :global(body) {
        background-color: #f0f0f0;
    }
</style>
```

KUVA 13. Svelte-komponentin style-osio

Markup-osio sisältää HTML-koodia, josta sivuston näkyvät osat piirretään. HTML-elementtien lisäksi markup voi myös sisältää aaltosulkeiden sisään kirjoitettua JavaScript-koodia, kuten viittauksia muuttujiin tai loogisia lausekkeita (kuva 14). (Svelte 2023. Basic markup.)

```
<script>
    const showHelloWorld = true;
</script>

<div>
    <h1>{showHelloWorld === true ? "Hello World" : ""}</h1>
</div>
```

KUVA 14. Looginen JavaScript-lauseke HTML-elementin sisällä

Markup-osiossa on usein myös logiikkalohkoja, joiden avulla määritetään, mitä HTML-osioita näytetään missäkin tilanteessa. Yleisimmin käytettyjä lohkoja ovat if-, else if-, else- ja each-lohkot. If- ja else if -lohkoilla voi olla erilaisia ehtoja, jotka määrittävät, ajetaanko kyseisen lohkon sisältämä

HTML-koodi vai ei. Jos mikään if- tai else if -lohkojen ehdoista ei täyty, ajetaan else-lohkon sisältämä koodi. (Kuva 15.) (Svelte 2023. Logic blocks.)

```
<script>
    const amount = 5;
</script>

<div>
    {#if amount > 10}
        <h1>More than 10</h1>
    {:else if amount < 10}
        <h1>Less than 10</h1>
    {:else}
        <h1>Exactly 10</h1>
    {/if}
</div>
```

KUVA 15. If-, else if- ja else -logiikkalohkot

Each-lohkolle syötetään lista arvoja, jotka lohko käy läpi ja ajaa lohkon sisältämän koodin erikseen jokaiselle listan sisältämälle arvolle (kuva 16). (Svelte 2023. Logic blocks).

```
<script>
    const numbers = [1, 2, 3, 4, 5];
</script>

<div>
    {#each numbers as number}
        <h1>{number}</h1>
    {/each}
</div>
```

KUVA 16. Each-logiikkalohko

HTML-elementeille voi asettaa erilaisia ohjeita, jotka ajetaan määritetyn tapahtuman hetkellä. Yleinen esimerkki tästä ovat sivuston sisältämät nappulat, joita painaessa ajetaan määritetty funktio (kuva 17). (Svelte 2023. Element directives.)

```
<script>
    function handleButtonClick() {
        alert('Button clicked!');
    }
</script>

<div>
   <button on:click={handleButtonClick}>Click me!</button>
</div>
```

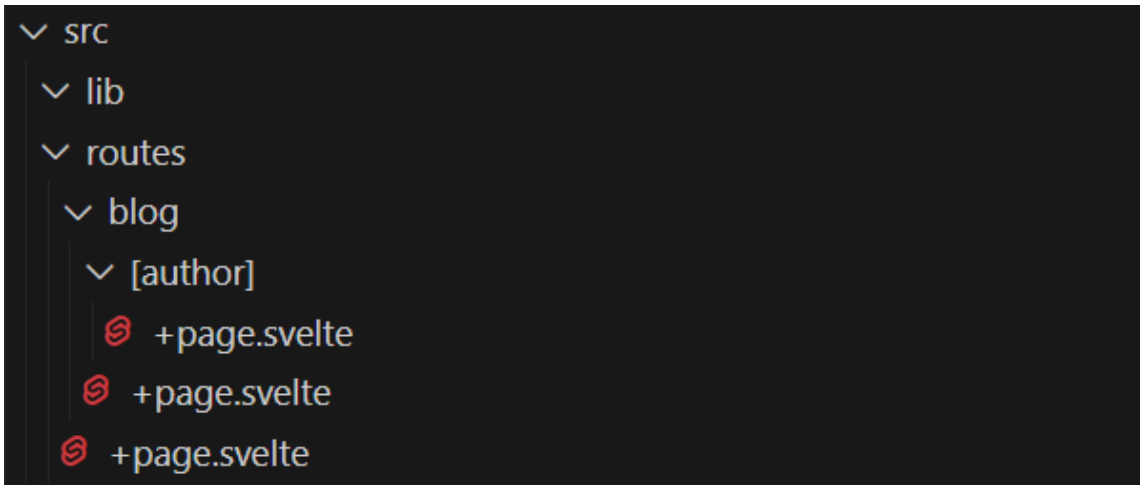
KUVA 17. Nappula, jota painaessa ajetaan määritetty funktio

HTML-elementti voidaan myös liittää johonkin arvoon käyttämällä bind-komentoa. Normaalisti tieto siirtyy parent-elementiltä child-elementille. Bind-komento kuitenkin mahdollistaa tiedon siirtymisen child-elementiltä parent-elementille. (Svelte 2023. Element directives.)

2.2.3 SvelteKit

SvelteKit on ohjelmistokehys, joka on kehitetty web-sovellusten luomiseen käyttäen Svelte-kirjastoa. SvelteKit on samankaltainen kuin Next.js, joka on tarkoitettu Reactin kanssa käytettäväksi. SvelteKit ei korvaa Svelteä, vaan samalla kun sovelluksen sivut luodaan käyttäen Svelteä, SvelteKitin avulla sovellukseen voi luoda esimerkiksi sivuston reitityksen. SvelteKit myös sisältää ominaisuuksia, kuten sovelluksen koonnin optimisoinnin ja palvelinpuolen renderöinnin (SSR). (SvelteKit 2024. Introduction.)

SvelteKitissä on tiedostojärjestelmään perustuva reititys. Tämä tarkoittaa sitä, että selaimessa käytetyt URL-polut muodostuvat sovelluksen routes-kansion tiedostojen ja kansioiden rakenteesta. Jos URL-polussa halutaan syöttää dynaamisia parametreja, täytyy tiedoston nimi ympäröidä hakasulkeilla. (Kuva 18.) (SvelteKit 2024. Routing.)



KUVA 18. SvelteKit-sovelluksen tiedostorakenne

Jos sivulle täytyy ladata jotain tietoa ennen kuin se voidaan piirtää, voidaan +page.svelte-tiedoston kanssa samaan tiedostopolkuun luoda +page.js-tiedosto, joka sisältää load-funktion. Tässä funktiossa voidaan hakea sivun tarvitsemat tiedot. Funktio ajetaan palvelimella palvelinpuolen renderöinnin (SSR) yhteydessä ja selaimessa käyttöliittymällä sivulle navigoitaessa. (Svelte 2024. Loading data.)

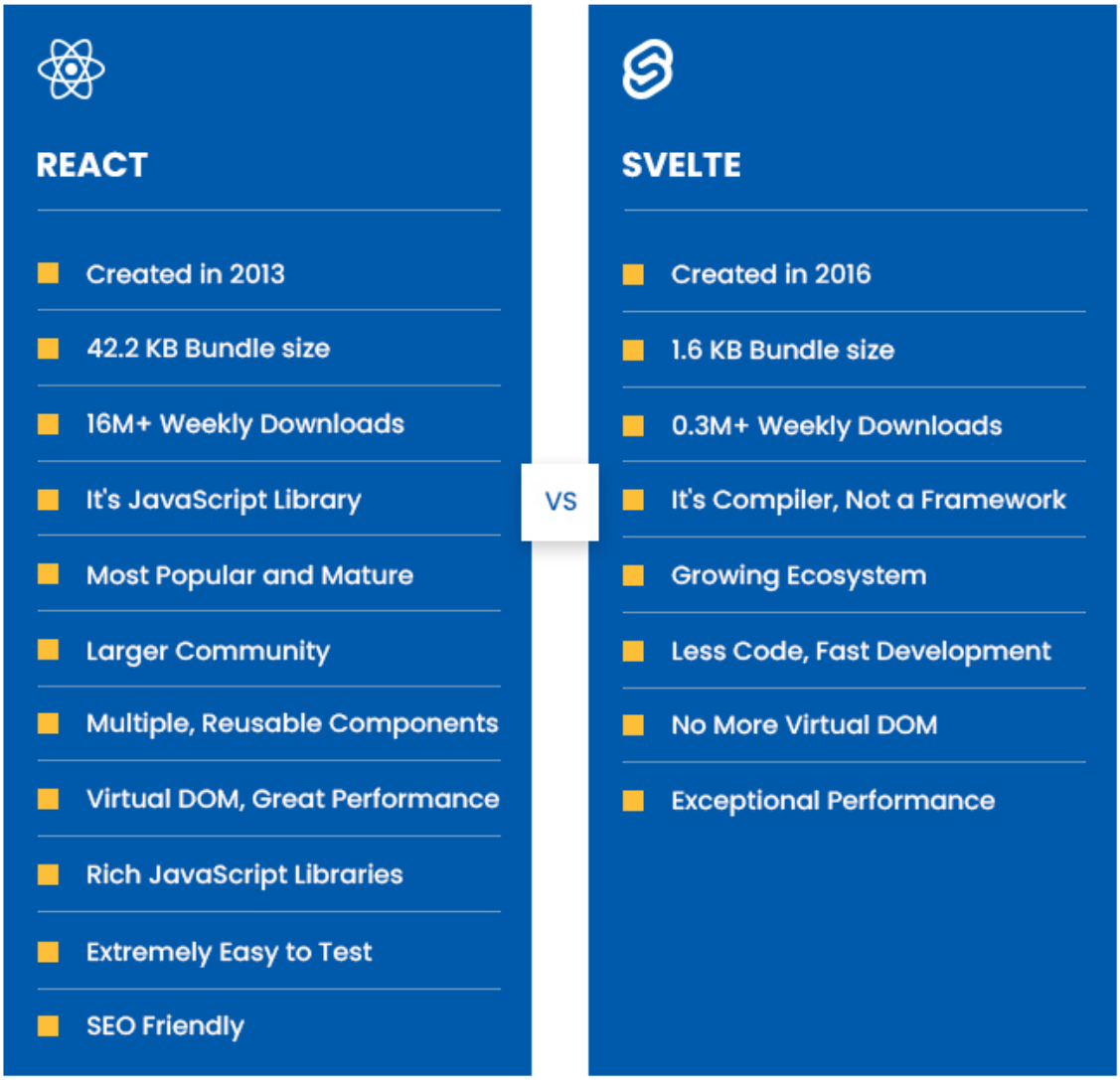
2.3 Reactin ja Svelten eroavaisuudet?

React ja Svelte ovat molemmat luotu samaan käyttötarkoitukseen. Niissä on kuitenkin joitain huomattavia eroja. Suurimpia eroja ovat esimerkiksi seuraavat asiat. React käyttää Virtual Document Object Modelia (DOM). Tämä varmistaa sen, että monimutkaisetkin käyttöliittymät päivittyvät sulavasti. Svelte ei käytä samaa mallia, minkä myötä sovellusten kehitystyö nopeutuu ja ohjelmistopaketit pienenevät. Svelten malli on parempi projekteihin, joissa prioriteettina on paras mahdollinen suorituskyky. (Uchenna 2023.)

Reactin ympärille muodostunut käyttäjien yhteisö on todella suuri. Svelten vastaava yhteisö on huomattavasti pienempi, mutta kasvaa kovaa vauhtia. Svelten käyttö on reaktiivisuuden sekä yksinkertaistetun syntaksin ansiosta jonkin verran helpompi ja nopeampi oppia kuin Reactin. React tarjoaa käyttäjille laajan valikoiman eri työkaluja käytettäväksi kehitystyöhön. Svelten työkaluvalikoima on suppeampi, mutta kasvaa koko ajan. Sveltessä on ominaisuus, jonka ansiosta sovelluksia voi luoda pienemmällä määrällä koodia kuin Reactia käytettäessä. Pienempien sovellusten kehityksessä Reactia käytettäessä, koodin määrä voi olla jopa 30-40 prosenttia

suurempi kuin Svelteä käytettäessä. Todella suurissa Svelte-sovelluksissa JavaScript-pakettien koko kuitenkin kasvaa Reactin vastaavia paketteja suuremmaksi. (Uchenna 2023.)

React sekä Svelte ovat molemmat hyviä vaihtoehtoja sovellusten käyttöliittymien luontiin. Niissä on kuitenkin monia huomion arvoisia eroavaisuuksia (kuva 19).



KUVA 19. Reactin ja Svelten eroavaisuuksia (Rajvanshi 2022)

Se, kumpi kannattaa valita, riippuu lähinnä siitä, millainen sovellus on kyseessä. Hieman yleistäen voi sanoa, että React on parempi ratkaisu suuriin sovelluksiin, joissa on monimutkaisia käyttöliittymiä ja joissa tarvitaan laajaa valikoimaa eri työkaluja. Svelte taas sopii paremmin pieniin tai keskikokoisiin yksinkertaisempiin sovelluksiin, joissa halutaan priorisoidaan nopeutta. (Uchenna 2023.)

3 SUUNNITTELU

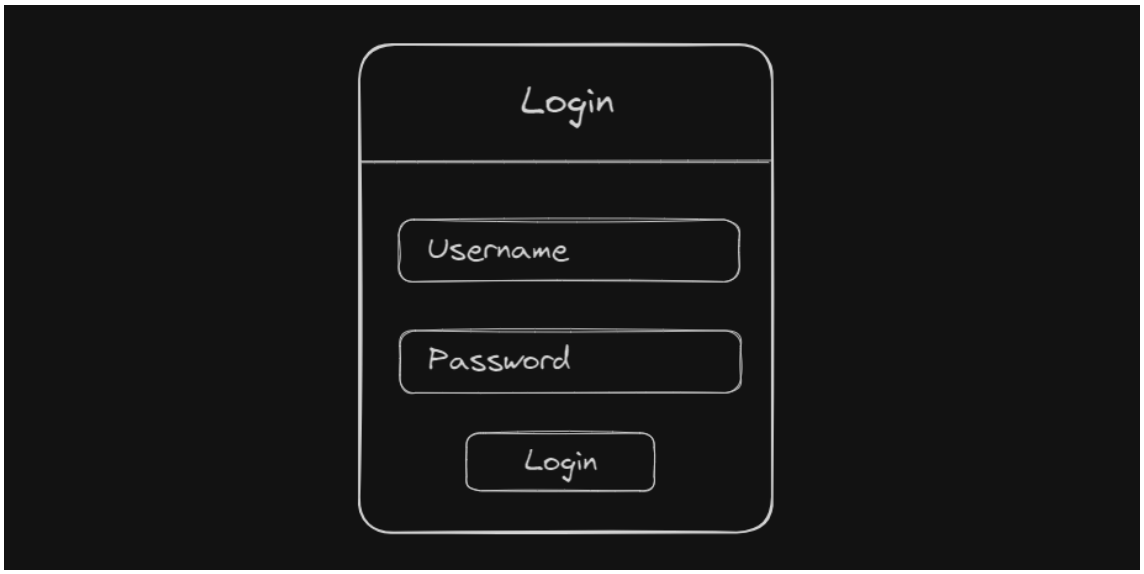
Sovelluksen suunnitteleminen aloitettiin miettimällä, mitä ominaisuuksia sovellukseen tarvitaan, ja miten kyseiset ominaisuudet kannattaa toteuttaa. Kun sovellukseen vaadittavat ominaisuudet olivat tarkentuneet, piirrettiin sovelluksen käyttöliittymästä kuvaajat. Tämän jälkeen mallinnettiin sovelluksen tietokanta.

3.1 Käyttöliittymä

Käyttöliittymän suunnittelu kannattaa aloittaa miettimällä kuka sovellusta tulee käyttämään, ja mikä on projektin lopputavoite. Tämän jälkeen tulee määrittää selkeät tavoitteet valmiille käyttöliittymälle. Kun on selvillä, mitä käyttöliittymältä vaaditaan, voidaan aloittaa käyttöliittymän mallintaminen. (Hannah 2023.)

Käyttöliittymän mallinnus tehtiin piirtämällä käyttöliittymän eri osioista rautalankamallit. Rautalankamallit ovat piirustuksia, joissa hahmotellaan käyttöliittymä pääpiirteittäin. Rautalankamallien tarkoitus on auttaa sovelluskehittäjiä hahmottamaan luotava käyttöliittymä. Mallien piirtämiseen käytettiin Excalidraw-työkalua. Käyttöliittymän eri osiot ovat kirjautumisnäkymä, työnantajan käytössä oleva perehdytyssuunnitelmien hallintanäkymä ja työntekijän käytössä oleva oman perehdytyssuunnitelman katselu- ja edistämisnäkymä.

Kirjautumisnäkymässä on yksinkertainen kirjautumislomake, johon käyttäjä voi syöttää käyttäjänimen ja salasanan sekä painaa Login-nappulaa, jolloin järjestelmä yrittää kirjata käyttäjää sisään syötetyillä tunnuksilla (kuva 20).



KUVA 20. Kirjautumisnäkymä

Perehdytyssuunnitelmien hallintanäkymä sisältää olemassa olevat perehdytyssuunnitelmat, suunnitelmien sisältämät tehtävät sekä tehtävien sisältämät alitehtävät joiden nimen vieressä on valintaruutu, joka ilmaisee onko työntekijä merkinnyt kyseisen alitehtävän suoritetuksi. Näkymässä on myös mahdollisuus luoda järjestelmään uusi työntekijä sekä lisätä, muokata ja poistaa suunnitelmia, tehtäviä ja alitehtäviä. (Kuva 21.)

KUVA 21. Perehdytyssuunnitelmien hallintanäkymä

Oman perehdytyssuunnitelman katselu- ja edistämisnäkymä koostuu käyttäjän perehdytyssuunnitelmaan määritetyistä tehtävistä sekä niiden alitehtävistä. Käyttäjällä on mahdollisuus merkitä alitehtävän nimeä edeltävästä valintaruudusta alitehtävä suoritetuksi. (Kuva 22.)

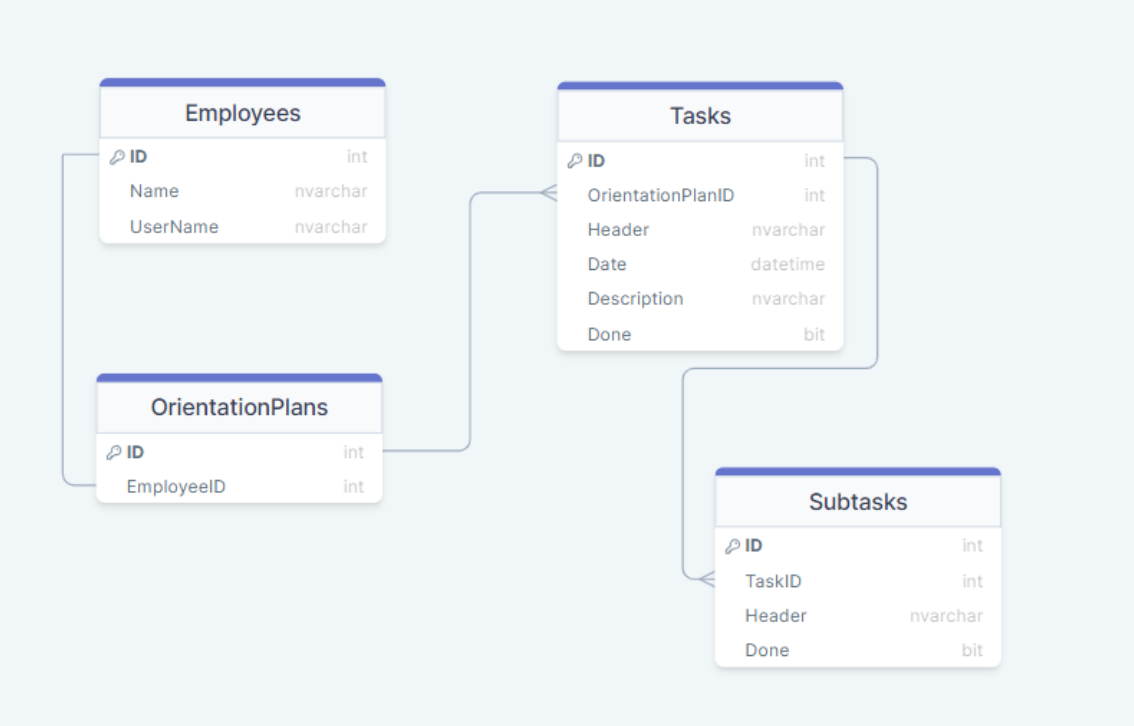
KUVA 22. Oman perehdytyssuunnitelman katselu- ja edistämisnäkymä

Näiden rautalankamallien pohjalta käyttöliittymien luominen on helpompaa, sillä kehitystyön aikana ei tarvitse juurikaan suunnitella käyttöliittymän ulkoasua vaan voidaan keskittyä sen toiminnalliseen puoleen.

3.2 Tietokanta

Hyvässä tietokantasuunnittelussa jaetaan tieto eri aihealueiden perusteella tauluihin, jotta tietokannassa olisi mahdollisimman vähän tarpeetonta tietoa, välitetään tietokannalle tarpeelliset tiedot, jotta se voi yhdistää tauluissa olevat tiedot ja varmistetaan tietokantaan syötetyn tiedon tarkkuus ja luotettavuus (Naeem 2023).

Tietokanta luodaan käyttämällä Microsoft SQL Server -relaatiotietokantojen hallintajärjestelmää. Tietokannan suunnittelu aloitettiin määrittelemällä, mitä tauluja sovellukseen tarvitaan sekä mitä attribuutteja tauluihin tarvitaan. Sovelluksen tietokantaan luodaan neljä taulua. Tauluja ovat työntekijät, perehdytyssuunnitelmat, tehtävät ja alitehtävät. Tietokannasta piirrettiin mallikuvaaja DrawSQL-sovelluksella käyttäen Entity-Relationship (ER) -mallia (kuva 23).



KUVA 23. Tietokannan mallikuvaaja

ER-mallissa kuvataan tietokannan eri taulut, taulujen attribuutit sekä taulujen keskinäiset suhteet järjestelmän sisällä. ER-mallinnuksen loi Peter Chen 1970-luvulla, ja se on edelleen hyvin suosittu mallinnustekniikka tietokantasuunnitellussa. (Lucidchart 2024).

4 TOTEUTUS

Sovelluksen toteutus aloitettiin luomalla ensin käyttöliittymä, sitten käyttöliittymälle tiedon hakemisen sekä käyttöliittymälle käyttäjän syöttämän tiedon talletuksen mahdollistava ohjelmointirajapinta (API) ja viimeiseksi tietokanta.

4.1 Käyttöliittymä

Käyttöliittymän luomiseen käytetty ohjelmointikieli oli JavaScript. Käytetyt teknologiat olivat Svelte ja SvelteKit. Kehitysympäristönä toimi Visual Studio Code. Työ aloitettiin asentamalla tarvittavat paketit, kuten Svelte ja SvelteKit. Käyttöliittymään luotiin rautalankamallin mukaisesti kaikki tarvittavat elementit, jotka aseteltiin oikeaan järjestykseen (kuva 24). Koska tietokantaa ei oltu vielä luotu, täytyi myöhemmin tietokannasta tulevat tiedot toistaiseksi kirjoittaa suoraan käyttöliittymä-koodiin.

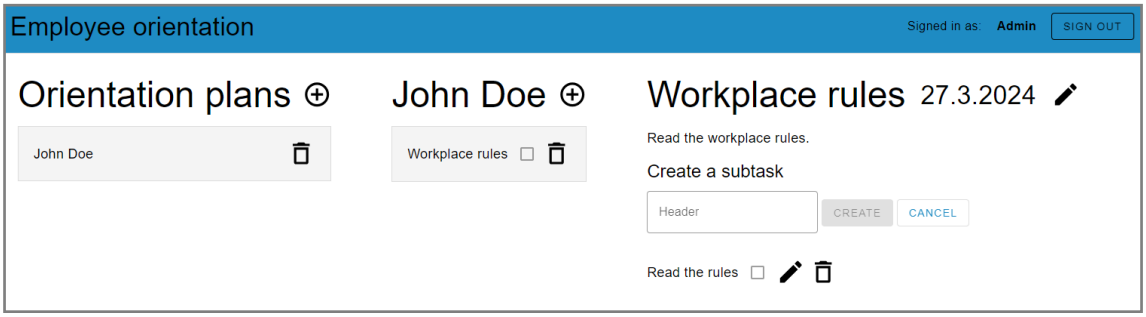
KUVA 24. Perehdytyssuunnitelmien hallintanäkymä ilman tyylejä

Seuraavaksi luotiin käyttöliittymän toimintalogiikka. Jokaiselle käyttöliittymän nappulalle sekä valintaruudulle luotiin funktio, joka ajetaan, kun kyseistä nappulaa tai valintaruutua painetaan. Myös kaikille tekstikentille luotiin toimintalogiikka (kuva 25).

```
function handleCheckSubtaskClick(event, subtask) {
    const done = event.target.checked;
    subtask.done = done;
}

function handleCreatedSubtaskHeaderChange(event) {
    subtaskHeader = event.target.value;
}
```

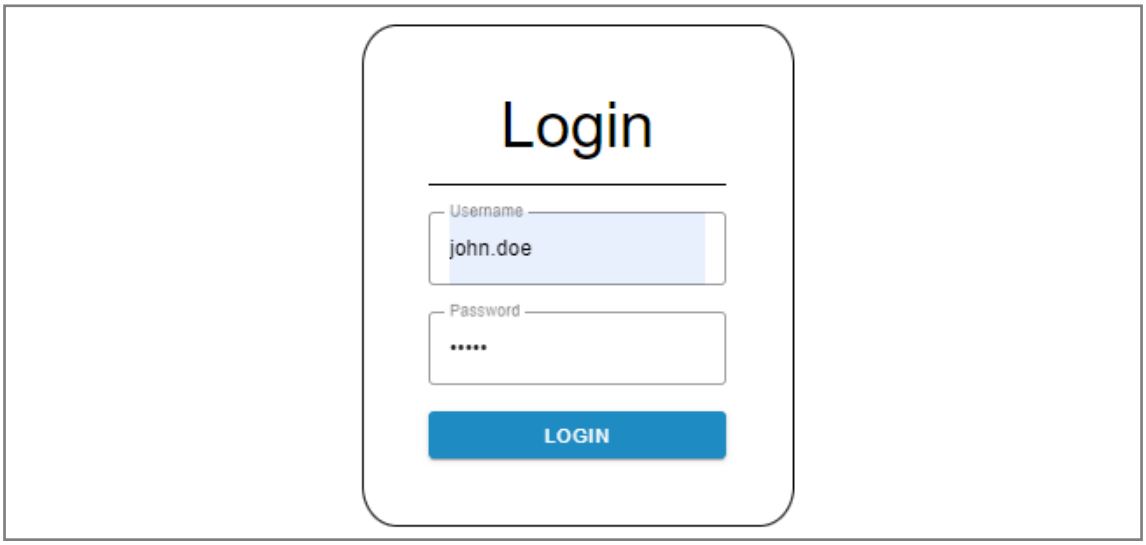
KUVA 25. Valintaruudun ja nappulan toimintafunktiot

Käyttöliittymän ulkoasusta saatiin modernin ja tyylikkään näköinen, kun kaikki elementit korvattiin Svelte Material UI -kirjaston tarjoamilla käyttöliittymäkomponenteilla, Add-, Delete- ja Edit-nappulat korvattiin kuvakkeilla ja käyttöliittymän tyylittely sekä asettelu viimeisteltiin käyttäen CSS-koodia (kuva 26).



KUVA 26. Perehdytyssuunnitelmien hallintanäkymä tyyliteltynä

Käyttöliittymän autentikaatio toteutettiin luomalla käyttöliittymään kirjautumisnäkymä (kuva 27). Sisäänkirjautuneen käyttäjän tiedot talletetaan selaimen evästeisiin. Eri näkymiin lisättiin käyttäjän oikeuksien tarkistus, jotta käyttäjillä on pääsy ainoastaan heille tarkoitettuihin näkymiin.

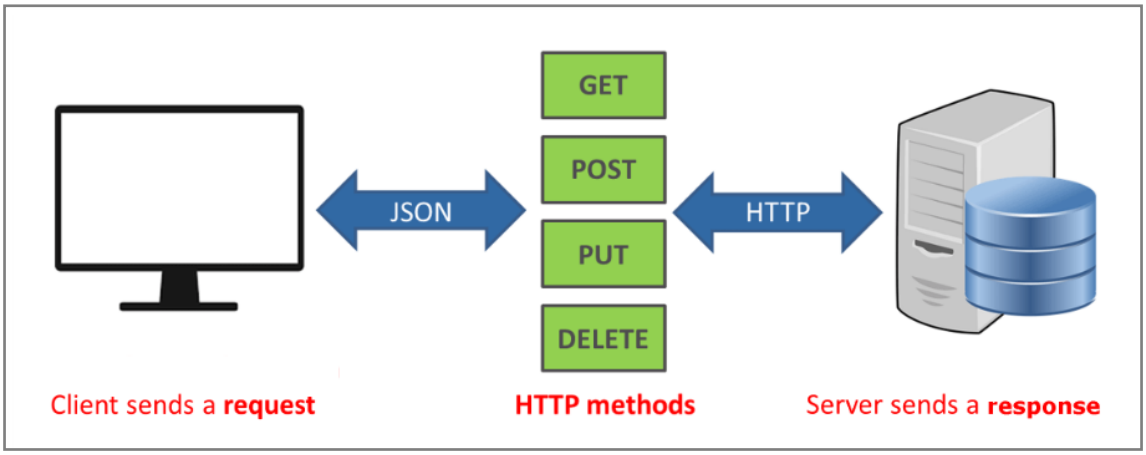


KUVA 27. Kirjautumisnäkymä

Evästeiden käyttö on SvelteKitin tukema ominaisuus ja siihen on tarjolla valmiit metodit Set, Get sekä Delete. Set-metodi ottaa vastaan evästeen nimen, evästeen arvon ja valinnaiset asetukset. Se luo annettujen parametrien perusteella evästeen ja tallettaa sen. Get-metodi ottaa vastaan evästeen nimen ja valinnaiset asetukset. Sen avulla voi hakea aiemmin Set-metodilla talletetun evästeen arvon. Delete-metodi ottaa vastaan evästeen nimen sekä valinnaiset asetukset ja poistaa aiemmin talletetun evästeen. (Meena 2022.)

4.2 Ohjelmointirajapinta

Ohjelmointirajapinta (API) on mekanismi, joka mahdollistaa kahden eri ohjelmistokomponentin kommunikoinnin keskenään. Tässä sovelluksessa luotiin Representational State Transfer (REST) -rajapinta käyttöliittymän ja tietokannan välille. Tällaiseen rajapintaan voidaan määritellä erilaisia metodeja. Nämä metodit ovat GET, POST, PUT ja DELETE. GET-metodilla luetaan tietoa tietokannasta, POST-metodilla tallennetaan tietoa tietokantaan, PUT-metodilla muokataan tietokannassa olevaa tietoa ja DELETE-metodilla poistetaan tietokannassa olevaa tietoa. (Kuva 28.) (Amazon Web Services 2024.)



KUVA 28. REST-rajapinta käyttöliittymän ja palvelimen välillä (Benharosh 2018)

Ohjelmointirajapinnan luominen aloitettiin asentamalla Visual Studio 2022 -kehitysympäristö. Visual Studiossa luotiin uusi projekti käyttäen ohjelmiston tarjoamaa valmista mallia web-rajapinnalle. Seuraavaksi projektiin asennettiin tarvittavat paketit. Näistä merkittävin oli Entity Framework Core -paketti. Entity Framework Coren avulla voi luoda siistin, siirrettävän ja korkeatasoisen tiedonhallintakerroksen. Entity Framework Core toimii useiden eri tietokantojen kanssa. (NuGet 2024.)

Sovellukseen luotiin tarvittavat mallikomponentit. Mallien nimet ja attribuutit vastaavat tietokannalle suunniteltuja tauluja. Jokaiselle mallille luotiin ohjainkomponentti. Näihin komponentteihin luotiin tarvittavat funktiot tiedon hakemista ja tallettamista varten. Kyseisiä funktioita kutsutaan käyttöliittymästä.

4.3 Tietokanta

Tietokannan luominen aloitettiin asentamalla SQL Server 2022 sekä SQL Server Management Studio -työkalu. Sovellus yhdistettiin SQL Serveriin Visual Studio -työkalun avulla. Tietokantaan luotiin taulut käyttäen Entity Framework Core -pakettia. Taulut muodostettiin automaattisesti ohjelmistorajapintasovellukseen luoduista mallikomponenteista.

5 POHDINTA

Opinnäytetyön tavoitteena oli luoda web-sovellus uusien työntekijöiden perehdyttämisen ja sen seurannan tueksi. Sovellukseen määriteltiin eri toiminnot työnantajan ja työntekijän käytettäväksi. Työnantajalla piti olla mahdollisuus luoda perehdytyssuunnitelma uudelle työntekijälle. Työntekijän piti voida tarkastella hänelle määritettyä perehdytyssuunnitelmaa ja merkitä siihen sisältyviä tehtäviä suoritetuiksi. Sovellus saatiin toteutettua ja siitä tuli edellä mainittujen määrittelyjen mukainen.

Sovellus ei tule kaupalliseen käyttöön, vaan se luotiin ainoastaan tätä opinnäytetyötä varten. Halutessaan sovelluksesta voisi saada käyttökelpoisen työkalun kaupalliseen käyttöönkin asti. Tämä kuitenkin vaatisi vielä paljon jatkokehitystä esimerkiksi lisäominaisuuksien muodossa. Samaan käyttötarkoitukseen on kuitenkin markkinoilla saatavilla useita hyviä vaihtoehtoja, joten tämä ei mielestäni olisi vaivan arvoista.

Työn aihe oli mielenkiintoinen ja sopivan haastava. Web-sovellusten tekeminen oli minulle jo entuudestaan tuttua. Tähän projektiin kuitenkin valitsin sellaiset teknologiat, jotka eivät olleet minulle kovinkaan tuttuja. Käyttöliittymän kehitykseen valitsin Svelten, joka oli minulle täysin tuntematon. Ohjelmointirajapinnan luomiseen valitsin ASP.NET Core -kirjaston, jota olin aiemmin käyttänyt vain vähän. Näiden minulle uusien teknologioiden käyttö toi projektiin lisää haastetta ja lisäsi sen työmäärää, sillä uusien teknologioiden käytön opettelemiseen kului merkittävä määrä aikaa.

Opinnäytetyön teoriaosuudessa tutustuttiin React- ja Svelte-käyttöliittymäkirjastojen toimintaan sekä historiaan. Niiden vahvuuksia ja heikkouksia punnittiin, ja niitä verrattiin toisiinsa. Lopullista vastausta siihen, että kumpi on parempi, ei saatu selville. Kumpikaan kirjasto ei ole täydellinen, vaan molemmilla on hyvät ja huonot puolensa. Hieman yleistäen voidaan todeta, että React on parempi vaihtoehto suuriin ja monimutkaisiin projekteihin, ja Svelte sopii paremmin kooltaan pienempiin projekteihin.

Opinnäytetyön teko kehitti ohjelmointitaitojani, ongelmanratkaisukykyäni ja kirjallista tekstintuotto- ja raportointitaitoani. Uusien teknologioiden oppimista tehosti niistä, tätä opinnäytetyötä varten tekemäni, tutkimustyö ja raportointi.

LÄHTEET

Stack Overflow 2023. Technology. 2023 Developer Survey. Hakupäivä 24.1.2024. https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe.

Holthausen Simon 2022. Svelte: A Beginner's Guide. SitePoint. Hakupäivä 25.1.2024. O'Reilly for Higher Education. Vaatii käyttöoikeuden.

A-Team Global 2023. Why is React so popular these days? Hakupäivä 27.1.2024. https://a-team.global/blog/why-is-react-so-popular/.

Banks, Alex & Porcello, Eve 2020. Learning React. Modern Patterns for Developing React Apps. 2. Painos. Yhdysvallat: O'Reilly Media. Hakupäivä 27.1.2024. O'Reilly for Higher Education. Vaatii käyttöoikeuden.

BuiltWith 2024. React Usage Statistics. Hakupäivä 27.1.2024. https://trends.builtwith.com/javascript/React.

Learn Svelte 2023. Introduction / Welcome to Svelte. Hakupäivä 28.1.2024. https://learn.svelte.dev/tutorial/welcome-to-svelte.

Svelte 2023. Svelte components. Hakupäivä 29.1.2024. https://svelte.dev/docs/svelte-components.

MDN web docs 2024. CSS first steps. Hakupäivä 23.3.2024. https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps.

Svelte 2024. Basic markup. Hakupäivä 23.3.2024. https://svelte.dev/docs/basic-markup.

Svelte 2024. Logic blocks Hakupäivä 23.3.2024. https://svelte.dev/docs/logic-blocks.

Svelte 2024. Element directives. Hakupäivä 23.3.2024. https://svelte.dev/docs/element-directives.

SvelteKit 2024. Introduction. Hakupäivä 23.3.2024. https://kit.svelte.dev/docs/introduction.

MDN web docs 2024. Getting started with React. Hakupäivä 23.3.2024. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started.

React 2024. Built-in React Hooks. Hakupäivä 23.3.2024. https://react.dev/reference/react/hooks.

W3Schools 2024. React JSX. Hakupäivä 23.3.2024. https://www.w3schools.com/react/react_jsx.asp.

SvelteKit 2024. Routing. Hakupäivä 24.3.2024. https://kit.svelte.dev/docs/Routing.

Amazon Web Services 2024. What is a RESTful API? Hakupäivä 25.3.2024. https://aws.amazon.com/what-is/restful-api/.

Benharosh, Joseph 2018. What is REST API? PHPenthusiast. Hakupäivä 25.3.2024. https://phpenthusiast.com/blog/what-is-rest-api.

Uchenna, Emmanuel 2023. Svelte vs React 2024: Which is Better? Prismic. Hakupäivä 28.3.2024. https://prismic.io/blog/svelte-vs-react.

KS, Adwaith 2022. React.js Basics – The DOM, Components, and Declarative Views Explained. freeCodeCamp. Hakupäivä 28.3.2024. https://www.freecodecamp.org/news/reactjs-basics-dom-components-declarative-views/.

NuGet 2024. Microsoft EntityFrameworkCore. Hakupäivä 6.4.2024. https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.

Rajvanshi, Neha 2022. Svelte Vs React: Which is the Best for Web App Development? Ace Infoway. Hakupäivä 9.4.2024. https://www.aceinfoway.com/blog/svelte-vs-react.

Hannah, Jaye 2023. A Complete Guide to the UI Design Process. UX Design Institute. Hakupäivä 9.4.2024. https://www.uxdesigninstitute.com/blog/guide-to-the-ui-design-process/.

Naeem, Tehreem 2023. Database Design – Learn How to Design a Good Database. Astera. Hakupäivä 9.4.2024. https://www.astera.com/type/blog/all-you-need-to-know-about-database-design/.

Meena, Shivam 2022. SvelteKit Changes: Cookies and Authentication. DEV Community. Hakupäivä 10.4.2024. https://dev.to/theether0/sveltekit-changes-session-and-cookies-enb.

Chatterjee, Shormistha 2023. Web Frameworks: All You Should Know About. BrowserStack. Hakupäivä 20.4.2024. https://www.browserstack.com/guide/web-development-frameworks.

Lucidchart 2024. What is an Entity Relationship Diagram (ERD)? Hakupäivä 20.4.2024. https://www.lucidchart.com/pages/er-diagrams.

SvelteKit 2024. Loading data. Hakupäivä 21.4.2024. https://kit.svelte.dev/docs/load.