VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Veeti Auria

HINNOITTELUTYÖKALUN KEHITYS

Tietojenkäsittely

2025

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Veeti Auria
Opinnäytetyön nimi	Hinnoittelutyökalun kehitys
Vuosi	2025
Kieli	suomi
Sivumäärä	37
Ohjaaja	Päivi Rajala

Opinnäytetyön tavoitteena oli kehittää web-pohjainen hinnoittelutyökalu, joka tukee Yritys X:n hinnoitteluprosessia. Nykyisin yrityksellä on haasteita hinnoittelun tarkkuuden ja tehokkuuden kanssa, koska prosessi on manuaalinen ja riippuu Excel-taulukoista. Työkalun tarkoituksena on parantaa hinnoitteluprosessin luotettavuutta ja nopeutta.

Opinnäytetyössä sovelletaan laadullista tutkimusmenetelmää, jossa analysoidaan projektin eri vaiheita ja kerätään havaintoja. Työssä hyödynnetään ketterän kehityksen periaatteita ja käyttäjälähtöistä suunnittelua. Teknologiat, kuten HTML5, CSS, JavaScript, Node.js ja Azure SQL, valittiin projektin toteutukseen. Aineistona käytettiin projektin dokumentaatiota ja asiakaspalautetta, joka kerättiin iteratiivisessa kehitysprosessissa.

Kehitetty hintalaskuri vastasi Yritys X:n tarpeita ja tehosti hinnoitteluprosessia. Sovellus mahdollistaa reaaliaikaisen laskennan ja tallentaa kaikki tiedot tietokantaan, mikä parantaa prosessin tarkkuutta ja läpinäkyvyyttä. Käyttäjien antama palaute paransi sovelluksen käytettävyyttä ja kehitystyö jatkuu uusien vaatimusten mukaan. Tärkeimmät havainnot liittyivät iteratiivisen kehityksen ja asiakaspalautteen merkitykseen, joka mahdollisti sovelluksen jatkuvan parantamisen.

Avainsanat hinnoittelutyökalu, web-sovellus, ketterä kehitys, laadullinen tutkimus ja tietokannan hallinta

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Veeti Auria
Title	Development of a Pricing Tool
Year	2025
Language	Finnish
Pages	37
Name of Supervisor	Päivi Rajala

The objective of this thesis was to develop a web-based pricing tool to enhance the pricing process of the case company. Currently, the company faces challenges regarding the accuracy and efficiency of pricing, as the process is manual and relies on Excel spreadsheets. The tool aims to improve the reliability and speed of the pricing process.

This thesis applied qualitative research methods, analyzing the different stages of the project and collecting observations. Agile development principles and user-centered design were utilized in the project. Technologies such as HTML5, CSS, JavaScript, Node.js, and Azure SQL were chosen for the implementation. The data used in the study consists of project documentation and customer feedback collected during the iterative development process.

The developed pricing tool met the needs of Company X and streamlined the pricing process. The application enables real-time calculations and stores all data in a database, improving the accuracy and transparency of the process. The feedback from the users improved the application's usability and development continues based on new requirements. The key findings relate to the importance of iterative development and customer feedback, which enabled the continuous improvement of the application.

Keywords pricing tool, web application, agile development, qualitative research, database management

SISÄLLYS

KUVAT

LYHENTEET

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
NPM	Node Package Manager
SQL	Structured Query Language
Frontend	Sovelluksen käyttöliittymä
Backend	Sovelluksen palvelinpuoli
IoT	Internet of Things
API	Application Programming Interface

1 JOHDANTO

Nykyisessä liiketoimintaympäristössä tehokkaat ja tarkat hinnoittelutyökalut ovat keskeisessä asemassa yritysten kilpailukyvyn ylläpitämisessä. Erityisesti pk-yrityksille oikea hinnoittelu voi olla haastavaa resurssien ja asiantuntemuksen puutteen vuoksi. Tämän opinnäytetyön tarkoituksena on kehittää web-pohjainen hintalaskuri yritys X:lle, joka auttaa tehostamaan yrityksen hinnoitteluprosessiaan ja parantamaan päätöksenteon kautta. Hintalaskurin kehityksen on myös tarkoitus helpottaa hinnoittelujen ja parametrien ylläpitoa.

Hintalaskuri on suunnattu pääasiassa yrityksen X myyjille. Myyjien lisäksi sovellusta käyttävät tiimien esihenkilöt, jotka pystyvät ylläpitämään parametrejä, jotka vaikuttavat hinnoitteluun. Hintalaskurissa käyttäjä pystyy tallentamaan hinnoitteluja erilaisten parametrien avulla tietokantaan. Hinnoitteluja voidaan tällöin avata uudestaan tietokannasta ja muokata jälkikäteen tarvittaessa.

Työssä sovelletaan laadullisen tutkimuksen menetelmiä ja aineistokeruunmenetelmänä käytetään projektin eri vaiheiden havainnointia. Tämä mahdollistaa syvällisen tarkastelun projektin etenemistä ja kehityksestä. Aineiston analysoinnissa hyödynnetään teemoittelua ja laadullista sisällönanalyysiä, mikä tarjoaa kattavan kuvan prosessin kulusta ja lopputuloksen laadusta.

Opinnäytetyö pyrkii vastaamaan keskeiseen tutkimuskysymykseen siitä, miten toteutetaan toimiva ja luotettava hintalaskuri. Tämän lisäksi työssä tarkastellaan, miten hinnoitteluprosessi saadaan sujuvoitettua yrityksessä X, sekä kuinka hyvin valitut ohjelmistokehitysmenetelmät tukevat web-sovelluksen toteutusta.

Työssä hyödynnettiin iteratiivista kehitysmenetelmää, jossa sovellusta kehitettiin ja testattiin useassa vaiheessa. Käyttäjätestauksia suoritettiin Yritys X:n edustajien kanssa, ja niiden perusteella tehtiin käyttöliit-

tymän ja toiminnallisuuksien parannuksia. Lisäksi opinnäytetyössä perehdytään ohjelmistokehityksen moderneihin käytäntöihin sekä käyttöliittymäsuunnittelun periaatteisiin kirjallisuuskatsauksen avulla.

Opinnäytetyössä keskitytään sovelluksen frontend- ja backend-kehitykseen, mutta ei syvennytä laajasti tietokantaoptimointiin tai skaalautuvuuden hallintaan. Lisäksi työ rajautuu Yritys X:n tarpeisiin, eikä sovellusta vertailla muihin vastaaviin markkinoilla oleviin ratkaisuihin.

Tässä työssä olen käyttänyt ChatGPT:tä ja Microsoft Copilotia ideoinnin, tiedonhaun ja kielentarkistuksen välineenä. Olen muokannut tekstiä tekoälyn avulla useaan otteeseen, jotta kieli olisi selkeämpää ja helpommin ymmärrettävää, mutta välittäisi asiat yhä alkuperäisen tarkoitukseni mukaisesti.

2 OHJELMISTOKEHITYKSEN KETTERÄ KEHITYS

Ohjelmistokehityksessä noudatetaan useita periaatteita ja käytäntöjä, jotka ohjaavat kohti laadukkaampaa lopputulosta. Kehittäessämme hintalaskuria kollegani kanssa hyödynsimme näitä periaatteita varmistaaksemme projektin onnistumisen.

2.1 Ohjelmistokehityksen periaatteet ja käytännöt

Ohjelmistokehityksessä noudatetaan useita periaatteita, jotka tukevat järjestelmällistä ja tehokasta kehitysprosessia. Yksi tärkeimmistä periaatteista on modulaarisuus, joka jakaa ohjelmiston pienempiin itsenäisiin osiin. Tämä helpottaa sekä ylläpitoa sekä jatkokehitystä. (Huttenlocher & Spoonhower, 2002)

Toinen keskeinen periaate on abstraktio, joka vähentää monimutkaisuutta ja selkeyttää ohjelmiston rakennetta. Abstraktio tarkoittaa ohjelmistokehityksessä korkeammalla tasolla toimivien käsitteiden luomista, jotka piilottavat toteutuksen yksityiskohdat ja tarjoavat selkeän rajapinnan. Tämä parantaa koodin luettavuutta, ylläpidettävyyttä ja uudelleenkäytettävyyttä. Hyvin suunnitellut abstraktiot auttavat kehittäjiä keskittymään olennaisiin asioihin ilman tarpeettomien yksityiskohtien käsittelyä. (Huttenlocher & Spoonhower, 2002)

Lisäksi iteratiivinen kehitys on olennainen käytäntö, jossa ohjelmistoa kehitetään vaiheittain. Jokaisessa iteraatiossa ohjelmistoa parannetaan aiemman version pohjalta hyödyntämällä saatua palautetta ja testituloksia. Tämä mahdollistaa nopean palautteen hyödyntämisen ja ohjelmiston jatkuvan kehityksen. Iteratiivinen lähestymistapa yhdistettynä ohjelmiston testaamiseen parantaa sen laatua ja vähentää virheiden määrää. (Huttenlocher & Spoonhower, 2002)

2.2 Käyttäjälähtöinen suunnittelu ja ketterä kehitys

Modernissa ohjelmistokehityksessä käyttäjälähtöinen suunnittelu ja ketterä kehitys muodostavat kokonaisuuden, jonka avulla voidaan luoda laadukkaita ja käyttäjien tarpeita vastaavia sovelluksia.

Käyttäjälähtöinen suunnittelu keskittyy siihen, että käyttäjän tarpeet ja odotukset otetaan huomioon jo kehitysprosessin vaiheessa. Lähestymistapa sisältää aktiivisen käyttäjäpalautteen saamisen lisäksi käytettävyystestejä ja kyselyitä tulevilta käyttäjiltä, jolloin suunnitteluprosessi mukautuu paremmin todelliseen käyttöympäristöön. (Huttenlocher & Spoonhower, 2002)

Ketterä kehitys puolestaan korostaa iteratiivisuutta ja joustavuutta. Sovellusta kehitetään ominaisuus tai sykli kerrallaan ja jokaisen uuden ominaisuuden jälkeen kerätään palautetta, jonka pohjalta tehdään parannuksia. Tämän lähestymistavan avulla voidaan saavuttaa haluttu lopputulos yhteistyössä yrityksen X kanssa. Ketterä kehitys on ollut loistava lähestymistapa henkilökohtaisesti ensimmäiselle omalle suurelle ohjelmistoprojektille. Jatkuva kontaktointi asiakkaan kanssa on helpottanut vaatimuksien määrittelyä ja toteutusta haluttuun tapaan. (Huttenlocher & Spoonhower, 2002)

Yhdistämällä käyttäjälähtöinen suunnittelu ja ketterä kehitys luodaan aikaan prosessi, jossa ohjelmiston toiminallisuus, käytettävyys ja laatu paranevat merkittävästi. Lähestymistapa edistää jatkuvaa kehitystä ja varmistaa, että kehitetty hintalaskuri on tehokas ja käyttäjäystävällinen. Tämä kokonaisvaltainen kehitysmalli tukee ohjelmistokehityksen käytäntöjen noudattamista, mikä parantaa projektin onnistumista. (Huttenlocher & Spoonhower, 2002)

3 SOVELLUSKEHITYKSEN TEKNOLOGIAT

Tämä luku käsittelee modernin web-kehityksen keskeisiä teknologioita ja työkaluja, jotka mahdollistavat skaalautuvien ja turvallisten verkkosovellusten rakentamisen. HTML, CSS ja JavaScript muodostavat verkkosivustojen perustan, kun taas Node.js tarjoaa palvelinpuolen suorituskyvyn ja joustavuuden.

Azure pilvipalveluna mahdollistaa sovellusten hallinnan ja käyttöönoton tehokkaasti, ja GitHub tukee versionhallintaa sekä yhteistyötä kehitystiimissä. Lisäksi MS Entra tuo tietoturvan ja identiteetinhallinnan osaksi kehitysprosessia, varmistaen käyttäjien tunnistautumisen ja pääsynhallinnan.

3.1 HTML5

HTML on verkkosivujen peruskieli, joka määrittelee sivun rakenteen ja sisällön. HTML:n avulla voidaan luoda ja jäsentää verkkosivujen elementtejä. Elementteihin kuuluvat esimerkiksi otsikot, kappaleet, linkit, kuvat ja lomakkeet. (Duckett, 2011)

HTML5 on HTML:n uusin versio, joka tuo mukanaan merkittäviä parannuksia ja uusia ominaisuuksia verrattuna aikaisempiin versioihin. HTML5 elementteihin kuuluu muun muassa <article>, <section>, <header> ja <footer>, jotka parantavat sivun rakennetta ja luettavuutta. Uudet elementit auttavat hakukoneita ja muita teknologioita ymmärtämään verkkosivun sisältöä paremmin. (Ducket, 2011; Mozilla Developer Network, 2024).

HTML5 tuo myös merkittäviä parannuksia lomakeominaisuuksiin, mikä on erityisen hyödyllistä hintalaskurin kaltaisissa sovelluksissa. Uudet lomakekenttätyypit, kuten ”number”, ”date”, ”time”, ”calendar” ja ”range”, mahdollistavat erilaisten tietotyyppien syöttämisen ja validoin-

nin suoraan selaimessa. Uusien lomakeominaisuuksien avulla käyttäjäkokemus paranee sekä niiden avulla pystyy validoimaan dataa paremmin, joita käyttäjä sinne syöttää. (Ducket, 2011; Mozilla Developer Network, 2024).

Parannuksien myötä HTML5 mahdollistaa tehokkaamman ja käyttäjäystävällisemmän hintalaskurin luomisen.

```html
<!DOCTYPE html>
<html lang="fi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML5 Esimerkki</title>
</head>
<body>

    <header>
        <h1>Hintalaskuri</h1>
    </header>

    <section>
        <form>
            <label for="price">Hinta (€):</label>
            <input type="number" id="price" name="price" min="0" step="0.01" required>

            <label for="date">Päivämäärä:</label>
            <input type="date" id="date" name="date" required>

            <label for="quantity">Määrä:</label>
            <input type="range" id="quantity" name="quantity" min="1" max="100" value="50">

            <button type="submit">Laske</button>
        </form>
    </section>

    <footer>
        <p>&copy; 2025 Hintalaskuri</p>
    </footer>

</body>
</html>
```

Kuva 1. Yksinkertainen esimerkki HTML ominaisuuksista.

Kuvassa 1 on esitetty esimerkkikoodi, josta ilmenevät HTML-rakenteen keskeiset ominaisuudet. Keskeisimpiä ominaisuuksia ovat:

- <header> - elementti määrittää sivun tai osion otsikkoalueen, mikä parantaa semanttisuutta ja hakukoneystävällisyyttä.
- <section> - elementti ryhmittelee sisältöä loogisiksi kokonaisuuksiksi, esimerkiksi lomakkeeksi.

- <form> - elementti on lomakerakenne, jossa voidaan hyödyntää HTML5:n uusia syöttökenttäominaisuuksia.
- <input type=”number”> –elementti sallii vain numeroiden syöttämisen, mikä vähentää virheellistä tietoa. <input> on elementti, ja type="number" on sen attribuutti. <input>-elementti on HTML:n syötekenttäelementti, jota käytetään käyttäjän tietojen syöttämiseen. Käyttäjä voi syöttää mitä tahansa, jos selain ei validoi tietoa oikein tai jos JavaScript muokkaa arvoa.
- <input type=”date”> - tarjoaa kalenteripohjaisen päivämäärän valinnan käyttäjälle. type="date" määrittelee, että syötekentän tyyppi on päivämäärän valinta, jolloin selain näyttää esimerkiksi kalenterin valintamahdollisuutena.
- <input type=”range”> - on liukusäädin, jonka arvo voidaan määrittää tietyllä vaihteluvälillä.
- <footer> - elementti on sivun alatunniste, joka tekee rakenteesta selkeämmän ja parantaa navigoitavuutta (Mozilla Developer Network, 2024.)

Tämä rakenne tukee modernia, käyttäjäystävällistä ja semanttisesti selkeää HTML-dokumenttia. Kuvassa 1 oleva koodi on esimerkki, joka ei ole toteutetusta hintalaskurin koodista.

3.2 CSS

CSS on tyyliohjeiden kieli, jota käytetään verkkosivujen ulkoasun ja muotoilun muokkaamiseen. CSS:n avulla saadaan erotettua sisältö ja ulkoasun toisistaan, jonka avulla saadaan muokattua käyttäjän kannalta käyttäjäystävällinen ja mieluisa lopputulos. Käyttäjäystävällinen ja visuaalisesti miellyttävä lopputulos saavutetaan määrittelemällä erilaisia tyylejä, kuten värejä, fontteja ja asetteluja. (Ducket, 2011; Mozilla Developer Network, 2024).

CSS:n kolme keskeistä perusperiaatetta ovat selektorit, ominaisuudet ja arvot sekä kaskadointi. (Ducket, 2011; Mozilla Developer Network, 2024).

Selektorit määrittävät mihin HTML-elementteihin tyylisäännöt pätevät. Näitä ovat esimerkiksi elementtiselektorit, jotka kohdistuvat suoraan tiettyyn HTML-elementteihin. Luokkaselektoreilla voidaan muotoilla useita samantyyppisiä elementtejä samanaikaisesti ja ID-selektoreilla voidaan kohdistaa säännöt taas tiettyyn elementtiin. (Ducket, 2011; Mozilla Developer Network, 2024).

CSS-ominaisuudet ja arvot määrittävät mitä tyylimuutoksia tehdään ja miten ne toteutetaan. Esimerkiksi ”color: black;” määrittää tekstin värin mustaksi. (Ducket, 2011; Mozilla Developer Network, 2024).

Kaskadointi tarkoittaa sitä, että jos samaan elementtiin kohdistuu useita eri tyylisääntöjä, niiden järjestys ja tarkkuus määräävät, mikä niistä lopulta vaikuttaa. Näin varmistetaan, että tärkeimmät tyylit jäävät voimaan ja ristiriitaisuuksia voidaan hallita tehokkaasti. (Ducket, 2011; Mozilla Developer Network, 2024).

CSS:n tärkeimpiin käyttötarkoituksiin kuuluu myös verkkosivun asettelun hallinta. Asettelun avulla määritetään miten HTML elementit sijoittuvat sivulle. Erilaisten työkalujen, kuten ”display”, ”position” ja ”flexbox” avulla voidaan määritellä miten elementit järjestyvät ja reagoivat toisiinsa. (Ducket, 2011; Mozilla Developer Network, 2024).

Nykypäivän asetteluhallinnan tärkein ominaisuus on responsiivisuus, joka tarkoittaa yksinkertaisuudessaan sitä, että miten verkkosivu mukautuu automaattisesti eri laitteille ja näyttöjen kokoihin. CSS:n ”media queries”:n avulla voidaan määritellä, miltä sivuston pitäisi näyttää esimerkiksi puhelimella, tabletilla ja tietokoneella. Responsiivisuuden merkitys on vain kasvanut lähivuosina, koska puhelimet ja tabletit ovat yleistyneet niin paljon. (Ducket, 2011; Mozilla Developer Network, 2024).

```html
<!DOCTYPE html>
<html lang="fi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Esimerkki</title>
    <style>
        /* Elementtiselektori – määrittää kaikkien kappaleiden tekstin värin mustaksi */
        p {
            color: black;
            font-size: 16px;
        }

        /* Luokkaselektori – käytetään useisiin elementteihin, jotka kuuluvat tähän luokkaan */
        .highlight {
            background-color: yellow;
            font-weight: bold;
        }

        /* ID-selektori – kohdistuu vain yhteen elementtiin */
        #main-header {
            text-align: center;
            font-size: 24px;
            color: blue;
        }

        /* Responsiivisuus – muuttaa tekstin koon pienemmillä näytöillä */
        @media (max-width: 600px) {
            p {
                font-size: 14px;
            }
        }
    </style>
</head>
<body>

    <h1 id="main-header">CSS Esimerkki</h1>
    <p>Tämä on peruskappale, johon sovelletaan CSS-tyylejä.</p>
    <p class="highlight">Tämä kappale on korostettu CSS:n avulla.</p>

</body>
</html>
```

Kuva 2. Yksinkertainen esimerkki CSS ominaisuuksista.

Kuvassa 2 on esitetty esimerkki CSS-koodista. Tämä esimerkki ei ole peräisin varsinaisesta hintalaskurin koodista, vaan havainnollistaa CSS:n määrittelyjä. CSS-tyylisääntöjä voidaan käyttää verkkosivuston ulkoasun hallintaan. Kuvan 2 koodissa esiintyvät tyylisäännöt ovat:

- p { color: black; } – Keltaiseksi elementeissä, joihin on lisätty class="highlight".
- .highlight { background-color: yellow; } – Korostaa taustan keltaiseksi.
- #main-header { text-align: center; color: blue; } – Keskittää ja muuttaa otsikon värin siniseksi.

- @media (max-width: 600px) { p {font-size: 14px; } } – Pienentää kappale-elementtien (<p>) fonttikokoa 14 pikseliin, jos näytön leveys on enintään 600 pikseliä. (Ducket, 2011; Mozilla Developer Network, 2024).

3.3 JavaScript

JavaScript on ohjelmointikieli, joka on keskeisessä roolissa nykyaikaisten verkkosivustojen -ja sovellusten kehityksessä. Alun perin se suunniteltiin lisäämään interaktiivisuutta verkkosivuille, mutta nykyään sen käyttö on laajentunut merkittävästi. JavaScript toimii yhdessä HTML:n ja CSS:n kanssa muodostaen modernin web-kehityksen perustan. Sitä käytetään myös palvelinpuolen ohjelmoinnissa, erityisesti Node.js-ympäristössä, mikä laajentaa sen käyttömahdollisuuksia merkittävästi. (Flanagan, 2020; Mozilla Developer Network, 2024).

JavaScriptin merkitys ohjelmistokehityksessä on kasvanut huomattavasti sen syntymästä. JavaScript kehitettiin alun perin vuonna 1995 Netscape-yhtiössä lisäämään interaktiivisuutta verkkosivuille. Sen jälkeen se on kehittynyt monipuoliseksi ja tehokkaaksi kieleksi. JavaScriptiä käytetään nykyään laajasti web-sovelluksissa asiakaspuolella ja palvelinpuolen sovelluksissa. Node.js:n avulla pystyy rakentamaan kokonaisia sovelluksia, jotka toimivat loistavasti eri ympäristöissä. (Flanagan, 2020; Mozilla Developer Network, 2024).

JavaScriptin tärkeimpiin ominaisuuksiin kuuluu se, että sen avulla pystyy tehdä verkkosivuista interaktiivisia ilman sivun uudelleenlataamista. (Flanagan, 2020; Mozilla Developer Network, 2024). JavaScript on keskeinen osa hintalaskurin toteutusta, sillä se vastaa laskennan suorittamisesta ja dynaamisesta toiminnallisuudesta. Sen avulla voidaan käsitellä käyttäjän syöttämiä tietoja reaaliaikaisesti, suorittaa laskutoimituksia ilman sivun päivitystä ja tarjota välittömät tulokset. Tämä vähentää virhemarginaalia, koska käyttäjä näkee muutokset heti. Yhdistettynä HTML:ään ja CSS:ään JavaScript mahdollistaa interaktiivisen ja

responsiivisen hintalaskurin, joka mukautuu käyttäjän tarpeisiin tehokkaasti.

3.4 Node.js backend-kehityksessä

Node.js on avoimen lähdekoodin JavaScript ympäristö, joka perustuu Chrome V8 -moottoriin, jonka tarkoitus oli tuoda tapahtumapohjainen ohjelmointimalli JavaScriptiin. Node.js julkaistiin vuonna 2009 Ryan Dahlin toimesta, ja sitä käytettiin erityisesti palvelinpuolella, mutta sitä voidaan käyttää myös komentorivisovelluksissa, API-palveluissa ja IoT-laitteissa. Node.js on alun perinkin luotu palvelinpuolen applikaatioita varten. Se on tehokas ja skaalautuva ratkaisu verkkosovellusten taustajärjestelmien kehittämiseen. (Jadhav & Gonsalves, 2020)

Teknologia on yksisäkeinen eli se pystyy käsittelemään useita samanaikaisia pyyntöjä ilman, että jokainen pyyntö vaatisi erillisen säikeen, mikä parantaa sovelluksen suorituskykyä. Node.js:n suurimpiin etuihin kuuluu sen laaja modulaarinen rakenne. (Jadhav & Gonsalves, 2020)

NPM tarjoaa tuhansia valmiita kirjastoja ja moduuleja, jotka nopeuttavat kehitysprosessia ja vähentävät tarpeettoman koodin kirjoittamista. (Jadhav & Gonsalves, 2020) Lisäksi sen avulla voidaan rakentaa reaaliaikaisia sovelluksia, kuten hintalaskureita, joissa nopea tiedonkäsittely on elintärkeää.

Express.js on kevyt ja tehokas verkkokehys, joka on suunniteltu helpottamaan Node.js-pohjaisten sovellusten kehittämistä. Sen avulla voidaan yksinkertaistaa http-pyyntöjen käsittelyä, määritellä reittejä ja lisätä toimintoja, jotka mahdollistavat joustavan sovellusrakenteen. (Mozilla Developer Network, 2024).

Express.js:n suurimpiin etuihin kuuluu sen laaja yhteensopivuus muiden Node.js-kirjastojen kanssa sekä sen kyky skaalautua tarpeen mukaan. Tämän avulla voidaan määritellä reittejä, jotka ohjaavat pyyntöjä eri

resursseihin, sekä käsitellä käyttäjän syöttämää dataa. Reitit ja pyynnöt ovat olennainen osa lomakkeiden tai tietokantakutsujen hallinnassa. (Mozilla Developer Network, 2024).

Nykyään Node.js on yksi suosituimmista teknologioista modernien web-sovellusten backend-kehityksessä, koska se mahdollistaa tehokkaan tiedonkäsittelyn ja saumattoman integraation frontendin kanssa.

3.5 Tietokanta Azure SQL

Azure SQL on Microsoftin tarjoama pilvipohjainen tietokantapalvelu, joka mahdollistaa tietokantojen hallinnan pilvialustalla. Microsoft vastaa tietojen varmuuskopioinnista, mikä mahdollistaa tietojen palauttamisen vahinkotilanteissa. Palvelun tietoturva on huippuluokkaa, ja siihen sisältyvät ominaisuudet, kuten tietojen salaus, käyttöoikeuksien hallinta sekä roolipohjainen pääsynhallinta, jotka takaavat datan turvallisuuden ja hallitun käytön. Azure SQL on siis täysin hallittu tietokantapalvelu (PaaS), joka mahdollistaa tietojen tallentamisen ja hallinnan ilman fyysisen palvelininfrastruktuurin ylläpitoa. (Krishnaswamy, 2010; Microsoft 2023.)

Azure SQL tukee relaatiotietokantaa ja se mahdollistaa saumattoman integraation muiden Azure-palveluiden kanssa (Krishnaswamy, 2010; Microsoft 2023). Yhdistäminen hintalaskuriin oli vaivatonta integraation myöden. Teknologia sopii erityisesti sovelluksiin, joissa tarvitaan jatkuvaa käytettävyyttä, matalaa viivettä ja helppoa hallittavuutta.

Valitsimme tietokannaksi Azure SQL:n myös sen takia, että sen ylläpito on helppoa. Hintalaskurin tietokantatauluja on myös tarkoitus ylläpitää käyttöliittymän kautta. Integraation tekeminen JavaScriptillä ja Azure SQL:n oli melko vaivatonta ja se onnistui loistavasti. Kuvassa 3 on esimerkkikuva SQL- tietokantataulun ylläpitosivulta.

Kuva 3. Esimerkkikuva tietokantataulun ylläpitosivulta.

3.6 GitHub

GitHub on suosittu versionhallintapalvelu, joka perustuu Git-ohjelmistoon. Se mahdollistaa kehittäjille tehokkaan yhteistyön, koodin versionhallinnan ja muutosten seurannan (GitHub Docs, n.d). Hintalaskuriprojektissa GitHubin avulla pystyimme hallitsemaan koodin eri versioita ja seuraamaan muutoksia.

GitHubin avulla pystyimme työskentelemään omalla haarallaan ilman, että päähaaraan tehtiin muutoksia. Tämän avulla pystyimme varmistamaan, että uusia ominaisuuksia pystyttiin kehittämään ilman, että ne vaikuttivat heti koko sovellukseen. Tietyn toiminnallisuuden valmistuttua se yhdistettiin pääharaan, jonka avulla koodia voidaan tarkastella ja hyväksyä muutokset ennen niiden käyttöönottoa. (GitHub Docs, n.d.)

GitHubin käyttö projektissa oli merkittävä apu versionhallinnassa. Sen avulla pystyimme seuraamaan kehitystä, hallitsemaan eri versioita ja varmistamaan, että muutokset eivät aiheuttaneet virheitä tuotantokoodissa.

3.7 Microsoft Entra

Microsoft Entra on Microsoftin identiteetin ja pääsynhallinnan ratkaisu, joka tarjoaa turvallisen tavan hallita käyttäjien pääsyä sovelluksiin ja palveluihin (Microsoft Learn, n.d.). Hintalaskurissa Microsoft Entra otettiin käyttöön varmistaaksemme, että vain valtuutetut käyttäjät pääsevät sovellukseen käsiksi.

Sovellus hyödyntää Microsoft Entra ID -palvelua käyttäjien tunnistamiseen ja valtuuttamiseen. Käytännössä tämä tarkoittaa sitä, että vain organisaation hyväksymät käyttäjät voivat kirjautua sisään ja käyttää sovelluksen toiminnallisuuksia. Käyttäjä tunnistetaan Entran avulla ja käyttöoikeudet määräytyvät pääsynhallinnan periaatteiden mukaisesti. (Microsoft Learn, n.d.)

Tietoturva on tärkeä osa sovellusta ja Microsoft Entra tarjoaa useita ominaisuuksia sitä varten. Roolipohjaisen pääsynhallinnan avulla voimme määrittää käyttäjäryhmien mahdollisuudet tehdä asioita sovelluksessa (Microsoft Learn, n.d.). Tämän avulla saimme esimerkiksi määriteltyä sen, että vain tietyillä käyttäjillä on pääsy hintalaskurin tietokannan muokkaamiseen.

4 HINTALASKURIPROJEKTIN SUUNNITTELU

Luvussa neljä käsitellään projektin suunnittelua. Suunnitteluvaiheeseen kuuluvat käyttäjätapausten avulla tapahtuva vaatimusmäärittely, käyttöliittymän suunnittelu ja tekninen toteutussuunnitelma.

4.1 Käyttötapaukset

Käyttötapaukset auttavat vaatimusten määritellyssä ja suunnittelussa, sillä niiden avulla saadaan tieto siitä, mitä vaatimuksia käyttäjäryhmillä on tulevassa sovelluksessa.

Pricing tool backlog

Feature name	Descirption	Priority
Hinnoittelun tallentaminen	Käyttäjänä voin tallentaa tekemäni hinnoittelun.	1
Tallennettujen hinnoittelujen tarkastelu	Käyttäjänä voin tarkastella aiemmin luomiani hinnoitteluja	2
Tallennettujen hinnoittelujen muokkaus	Käyttäjänä voin muokata aiemmin tallennettua hinnoitteluani.	2
Käyttäjän tunnistaminen	Järjestelmä tunnistaa työkalua käyttävän käyttäjän.	3
Käyttöliittymän viilaus	Yrityksen X värit, käytettävyysparannukset.	3
Koodin katselmointi ja refaktorointi	Käydään läpi lähdekoodi ja optimoidaan rakennetta ja toteutusta.	2
Tietojen ylläpito	Käyttäjänä voin ylläpitää tietokantatauluja työkalulla.	2
Omat ylläpitokäyttöliittymät joka taulull	Käyttäjänä voin helposti valita mitä taulua haluan ylläpitää. Taulun ylläpidolle on oma käyttöliittymä.	2

Kuva 4. Kuvakaappaus osasta käyttötapauksista.

Käyttötapaukset (kuva 4) olivat olennainen osa hintalaskurin kehitysprosessia. Käyttötapaukset tarjosivat selkeämmän rakenteen siitä, mitä Yritys X on hintalaskurin kannalta halukas saamaan. Tapauksien avulla määriteltiin, miten eri käyttäjäryhmät käyttävät hintalaskuria. Käyttäjätapausten avulla pystyimme selvittämään, miten esimerkiksi hinnan syöttäminen ja kokonaishinnan laskeminen toteutettaisiin hintalaskurissa.

Jokaisen käyttäjätapauksen suorittamisen jälkeen järjestimme palaverin, jossa arvioimme toiminnallisuuden vastaavuutta odotuksiin ja alku-

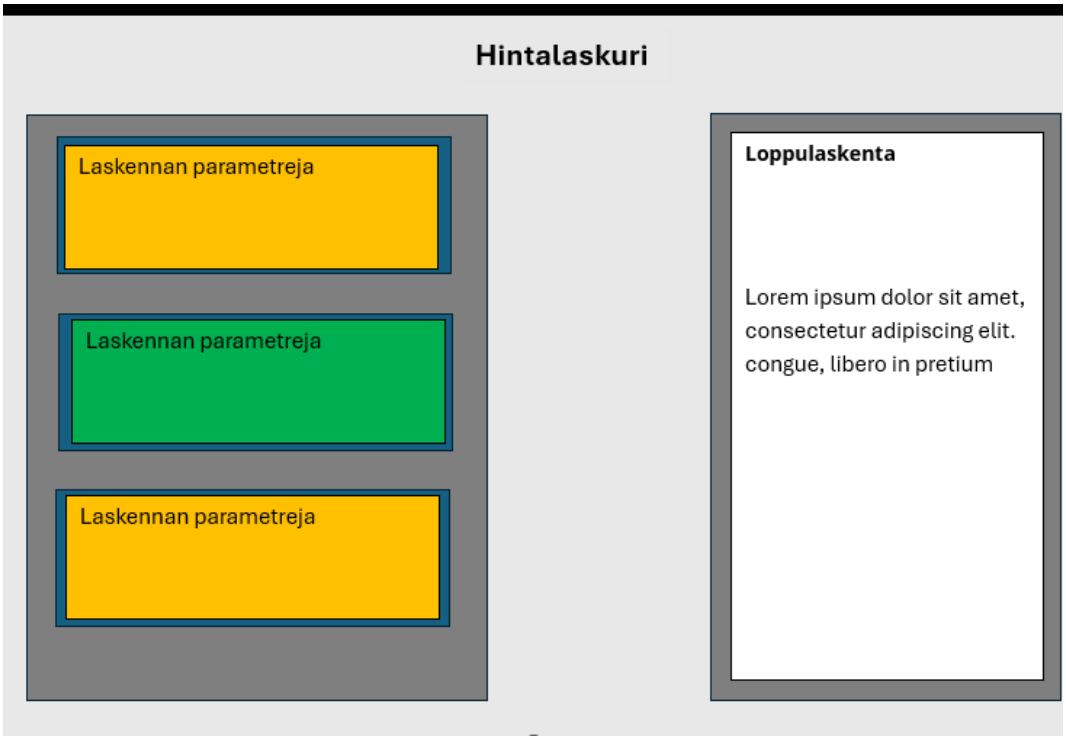
peräisiin suunnitelmiin. Tämä vaihe auttoi tunnistamaan mahdollisia ongelmakohtia ja hienosäätämään laskentaprosessia ennen seuraavaa kehitysvaihetta. Sisäisten tarkastelujen lisäksi pidimme tietyissä vaiheissa palavereita myös Yritys X:n kanssa, jotta varmistimme, että laskentamalli vastasi yrityksen tarpeitaan ja mahdollisia muutostarpeita voitiin tunnistaa ajoissa. Näiden keskustelujen avulla varmistettiin, että hintalaskurin kehitys eteni oikeaan suuntaan ja vastasi loppukäyttäjän odotuksia.

4.2 Käyttöliittymän suunnittelu

Käyttöliittymän luonti pohjautui Yrityksen X aikaisempaan Excel-pohjaan, jossa hintalaskentaa tehtiin. Käyttöliittymää ei suunniteltu laajasti etukäteen, vaan sitä kehitettiin iteratiivisesti prosessin edetessä. Palautteiden perusteella käyttöliittymään tehtiin muutoksia ja parannuksia. Kehitysvaiheiden jälkeen tehtiin käyttöliittymätestauksia sisäisesti tiimin kesken sekä Yrityksen X kanssa. Testien avulla tunnistettiin kehityskohteita ja saatiin varmistus siitä, että laskenta toimii odotetusti.

Yritykseltä X tuli esimerkiksi vaatimus syötekohtien värikoodaamiseen tietyille väreille, jotta käyttäjä tunnistaa pakolliset ja valinnaiset kentät. Mallina värikoodeille toimi erinomaisesti aikaisempi Excel-pohja. Toinen tärkeä vaatimus oli se, että lopullinen laskenta näkyy koko ajan käyttäjälle, jotta pääsemme erinomaiseen lopputulokseen laskennan kanssa. Kummatkin vaatimukset ovat esimerkkejä siitä, kuinka tärkeitä visuaaliset piirteet ja reaaliaikaisuus ovat käyttöliittymän suunnittelussa ja toteutuksessa.

Kokonaisuudessaan käyttöliittymän suunnittelussa pyrittiin varmistamaan, että hintalaskurin käyttäminen olisi mahdollisimman yksinkertaista, nopeaa ja tehokasta, mikä parantaa käyttökokemusta ja lisää sovelluksen arvoa käyttäjälle.

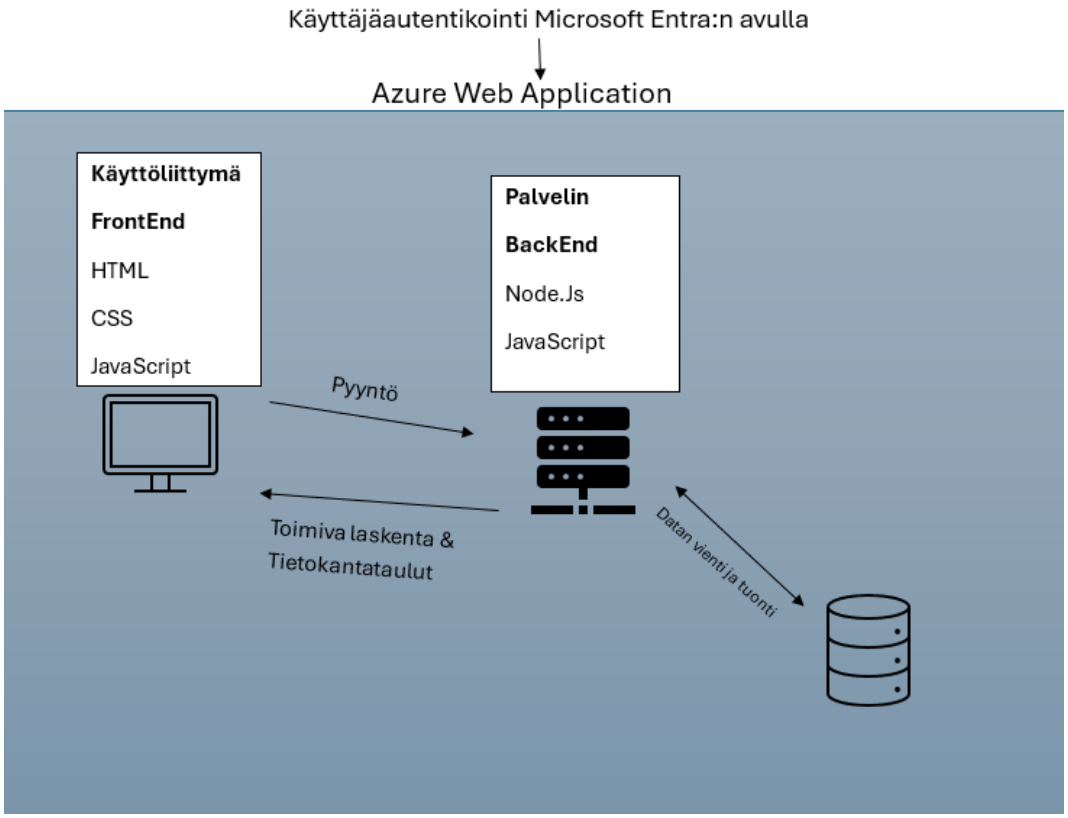


Kuva 5. Kuvakaappaus hintalaskurin käyttöliittymän suunnitelmasta.

Kuvassa 5 on esitetty hintalaskurin layout-suunnitelma ja tärkeimmät vaatimukset, jotka sisällytettiin suunnitelmaan. Loppulaskenta säilyy hinnoitteluparametrien mukana, vaikka sivustoa rullattaisiin alaspäin ja parametrien värikoodaus kertoo käyttäjälle pakollisista syötekentistä.

4.3 Tekninen toteutussuunnitelma

Hintalaskurin tekninen toteutussuunnitelma laadittiin ennen kehitystyön aloittamista, jotta voitiin varmistaa projektin sujuva eteneminen ja oikeiden teknologioiden valinta. Suunnitteluvaiheeseen kuului tarvittavien teknologioiden valinta ja niiden roolien määrittäminen.



Kuva 6. Kuvakaappaus teknisestä toteutussuunnitelmasta.

Kuvassa 6 esitetään hintalaskurin rakenne ja sen pääkomponentit. Käyttöliittymä mahdollistaa käyttäjän vuorovaikutuksen sovelluksen kanssa, palvelin käsittelee pyynnöt ja tietokanta tallentaa tarvittavat tiedot. Sovelluksen turvallisuus on varmistettu käyttäjäautentikoinnilla, ja palvelin huolehtii tietokantakutsujen hallinnasta sekä laskentaprosessin optimoinnista.

5 PROJEKTIN TOTEUTUS

Luvussa viisi käsitellään käyttöliittymän toteutusta. Luvussa kuvataan erikseen käyttöliittymän toteutus ja tekninen toteutus

5.1 Käyttöliittymän toteutus - Frontend

Frontendin eli käyttöliittymän toteutus eteni vaiheittain aiemmin laaditun suunnitelman mukaisesti, mutta kehitystyössä hyödynnettiin pääosin iteratiivista lähestymistapaa, jossa tehtiin jatkuvia parannuksia saatujen palautteiden perusteella. Suunnitelman toteutuksessa painotettiin erityisesti käytettävyyden parantamista ja visuaalisten vaatimusten täyttämistä, jotka oli määritelty yhteistyössä Yritys X:n kanssa.

Hintalaskurin käyttöliittymän toteutus aloitettiin perusrakenteen luomisella HTML:n avulla. Tässä vaiheessa rakennettiin käyttöliittymän keskeiset elementit, kuten syötekentät, laskentapainikkeet ja taulukot, joiden avulla käyttäjälle esitetään laskennan tulokset. Perusrakenteen toteutuksen jälkeen siirryttiin sovelluksen visuaalisen ilmeen kehittämiseen hyödyntäen CSS:ää. Tässä otettiin huomioon Yrityksen X visuaaliset vaatimukset, erityisesti jokaisen syötekentän värikoodaus, jonka mallina käytettiin Excel-pohjaa. Tuttu värimaailma helpottaa käyttäjien siirtymistä uuteen sovellukseen ja tekee käyttöliittymästä käyttäjäystävällisemmän.

Seuraavaksi toteutettiin dynaamisia toimintoja JavaScriptin avulla. Tämä mahdollistaa reaaliaikaisen laskennan ja syötteiden validoinnin, jolloin käyttäjä näkee laskennan tulokset heti syöttäessään tietoja ilman erillistä päivitystä. Reaaliaikaisuus parantaa käyttökokemusta ja tekee sovelluksesta tehokkaamman.

Lopuksi sovellusta kehitettiin iteratiivisesti saatujen palautteiden perusteella. Ensimmäiset testaukset suoritettiin sisäisesti kollegoiden kanssa, minkä jälkeen käyttöliittymää testattiin myös Yrityksen X edustajien

kanssa. Testausvaiheen aikana tunnistettiin parannuskohteita, kuten käyttöliittymän selkeyttäminen ja responsiivisuuden optimointi eri laitteille. Näiden muutosten pyrittiin varmistamaan, että laskentatoiminnot toimivat odotetusti ja käyttöliittymä oli käyttäjäystävällinen ja intuitiivinen.

5.2 Tekninen toteutus - Backend

Hintalaskurin backend kehitettiin käsittelemään keskeisiä toiminnallisuuksia, kuten käyttäjäautentikointia, laskentaprosessia, tietokantatoimintoja sekä tietoturvaa ja suorituskykyä. Microsoft Entra -integraation avulla varmistettiin, että vain valtuutetut käyttäjät voivat käyttää sovellusta, mikä paransi järjestelmän tietoturvaa ja käyttöoikeuksien hallintaa.

Laskentaprosessin osalta backend vastaanottaa käyttäjän syöttämät tiedot ja suorittaa hintalaskennan Yritys X:n määrittelemien vaatimusten mukaisesti. Laskennan tulokset palautetaan frontendille reaaliajassa, mikä mahdollistaa käyttäjälle välittömän näkymän syöttämiinsä tietoihin ja niiden vaikutuksiin lopputuloksessa.

Tietojen tallennus toteutettiin Azure SQL -tietokantaan, johon käyttäjän syötteet tallennetaan, jotta aiempiin laskelmiin voidaan palata tarvittaessa. Backend tukee CRUD (Create, Read, Update, Delete) -operaatioita, joiden avulla käyttäjät voivat hallita ja päivittää laskentatietojaan tehokkaasti.

Backend kehitettiin vaiheittain ja sitä testattiin jatkuvasti yksikkötesteillä ja integraatiotesteillä. Kehityksen aikana tunnistettiin haasteita, jotka ratkaistiin iteratiivisesti palautteen ja testitulosten perusteella. Alun perin hintalaskurin laskennat toteutettiin frontendin koodissa, mutta tietoturvasyistä ne siirrettiin backendin puolelle, jotta tiedot pysyisivät suojattuina. Tämä estää laskentakaavojen ja tietojen paljastumisen selaimessa sekä parantaa datan eheyttä ja suojausta. Muutos vei

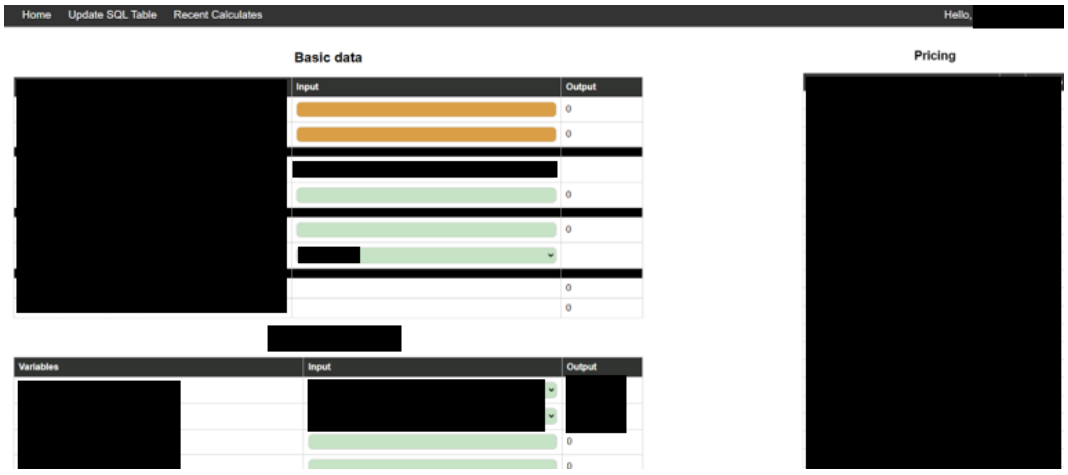
paljon aikaa ja osoitti, että laskentatoimintojen sijoittaminen suoraan backendiin olisi ollut järkevää jo projektin alkuvaiheessa.

Hintalaskurin backend mahdollistaa turvallisen ja tehokkaan tietojen käsittelyn. Node.js- ja Express.js-pohjainen toteutus tarjoaa suorituskykyisen rajapinnan, joka kommunikoi saumattomasti Azure SQL -tietokannan kanssa. Iteratiivinen kehitys ja kattava testaus varmistivat, että backend täyttää laskennalliset ja tekniset vaatimukset, tukien koko sovelluksen sujuvaa käyttöä.

6 PROJEKTIN TULOKSET

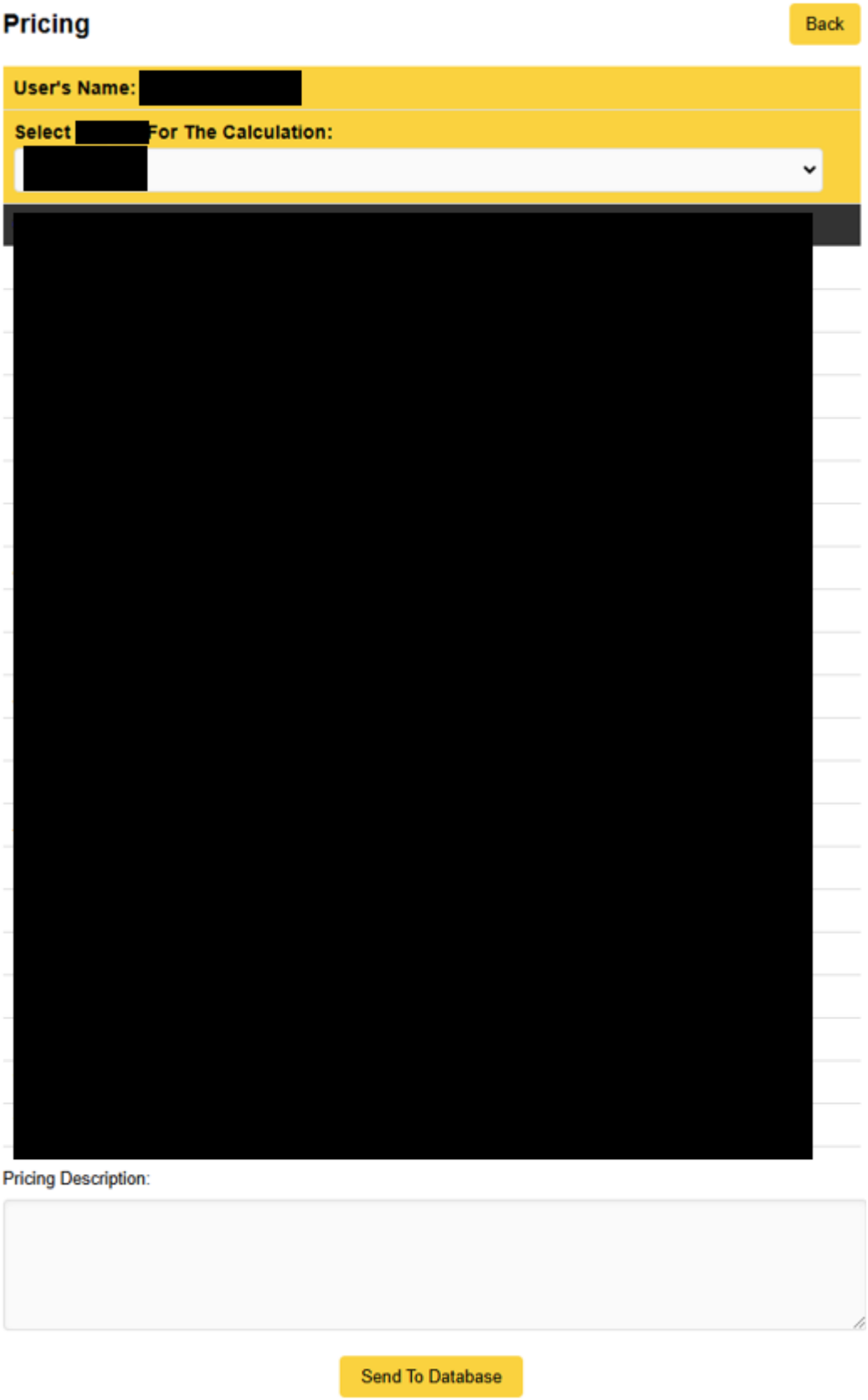
Tässä kappaleessa käydään läpi hintalaskurin kehitysprojektin keskeiset tulokset ja niiden tulevat vaikutukset Yritys X:n toimintaan. Projekti on edennyt suunnitelman mukaisesti ja kehitetty työkalu on vastannut yrityksen tarpeisiin hinnoitteluprosessin tehostamisessa.

6.1 Kehitetyn työkalun esittely



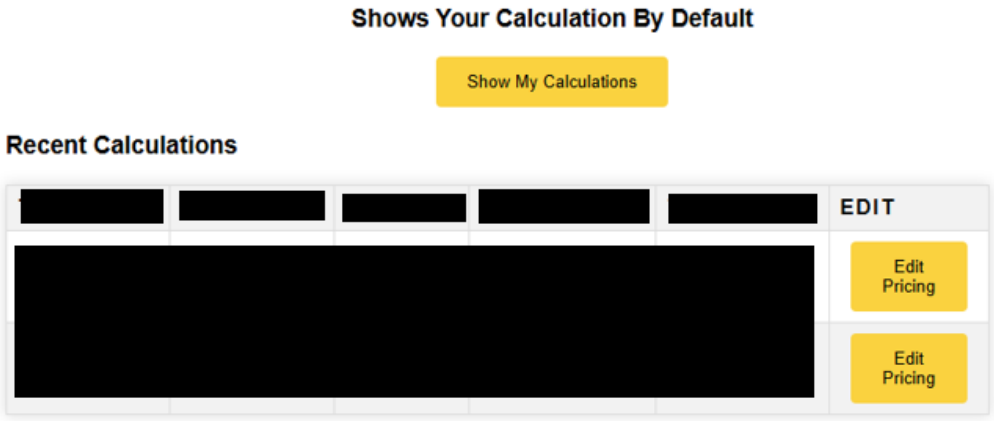
Kuva 7. Kuvakaappaus toteutetusta hintalaskurista.

Kuvassa 7 on esitetty ajantasainen kuvakaappaus hintalaskurista, jossa näkyvät Yritys X:n vaatimusten mukaisesti toteutetut ominaisuudet. Loppuhinnoittelu pysyy näkyvissä oikeassa reunassa riippumatta siitä, kuinka pitkälle käyttäjä selaa hinnoittelun parametreja. Lisäksi kenttien värikoodaus on toteutettu toiveiden mukaisesti, ja värien muuttuminen hinnoittelun täyttämisen aikana toimii odotetusti, mikä parantaa käyttäjäkokemusta ja selkeyttää laskentaprosessia.



Kuva 8. Kuvakaappaus lasketun hinnoittelun esikatselutilasta.

Kuvassa 8 on esitetty lasketun hinnoittelun esikatselutila, jossa käyttäjä voi tarkastella syöttämiensä parametrien perusteella muodostettua lopullista hinnoittelua. Esikatselutila on suunniteltu vastaamaan Yritys X:n vaatimuksia, ja sen avulla käyttäjä voi varmistaa laskennan oikeellisuuden ennen tallennusta. Lisäksi esikatselutilasta on mahdollista palata muokkaamaan tietoja helposti, mikä mahdollistaa joustavan ja tehokkaan hinnoitteluprosessin.



Kuva 9. Kuvakaappaus sovelluksen sivusta, josta näkee vanhat laskelmat.

Kuvassa 9 on esitetty, miten hintalaskurin vanhojen laskelmien sivu on toteutettu. Yritys X toivoi, että vanhoista laskelmista näkyisivät vain tietyt kentät, ja tämä toive huomioitiin toteutuksessa. Lisäksi vanhojen laskelmien muokkaamisen tuli olla käyttäjälle mahdollisimman sujuvaa, joten toiminnallisuus toteutettiin siten, että laskelmaa voi muokata helposti yhdellä painikkeella. Tämä parantaa käytettävyyttä ja nopeuttaa hinnoitteluprosessia.

Yritys X on ollut tyytyväinen hintalaskurin kehitykseen ja sen tuomiin parannuksiin hinnoitteluprosessissa. Sovellus on helpottanut ja nopeuttanut laskennan tekemistä, vähentänyt manuaalisia virheitä ja tuonut

selkeyttä hinnoittelun hallintaan. Erityisesti reaaliaikainen laskenta, selkeä käyttöliittymä ja yrityksen vaatimusten mukainen värikoodaus ovat saaneet positiivista palautetta.

Yrityksen tarpeet ja vaatimukset kehittyvät jatkuvasti, joten sovellukseen lisätään uusia toiminnallisuuksia sitä mukaa, kun niitä tunnistetaan. Esimerkiksi käyttöliittymän parannukset, uusien parametriasetusten lisääminen ja raportointiominaisuudet ovat kehityksessä mukana. Tiivis yhteistyö yrityksen kanssa varmistaa, että sovellus kehittyy vastaamaan käyttäjien tarpeita entistä paremmin, ja se pysyy ajan tasalla muuttuvien vaatimusten kanssa. Jatkossa sovellusta yritetään liittää Yrityksen X muihin sovelluksiin, jotta integraatio on sujuvampaa ja hinnoittelun tekeminen parantuu entistä enemmän.

6.2 Hinnoittelutyökalun vaikutus yrityksen X prosesseihin

Hintalaskurin käyttöönoton odotetaan tehostavan merkittävästi Yritys X:n hinnoitteluprosesseja, sillä se vähentää manuaalisen työn määrää ja lisää laskennan tarkkuutta. Aiemmin hinnoittelu perustui Excel-taulukoihin ja manuaalisiin laskelmiin, mikä saattoi aiheuttaa virheitä ja hidastaa päätöksentekoa. Uuden työkalun avulla hinnoitteluprosessi nopeutuu ja yhdenmukaistuu, mikä parantaa hinnoittelun läpinäkyvyyttä ja luotettavuutta.

Työkalun käyttöönotto mahdollistaa tehokkaamman tiedonhallinnan, sillä hinnoittelulaskelmat tallentuvat keskitetysti tietokantaan. Tämä mahdollistaa laskelmien helpon haun ja analysoinnin myöhemmin, mikä parantaa prosessin läpinäkyvyyttä ja tarkkuutta. Lisäksi hinnoitteluparametrien hallinta tapahtuu tietokannan kautta, mikä antaa esihenkilöille paremman hallinnan parametrien määrittelyssä ja päivityksessä.

Muut hyödyt tulevat esiin ajan myötä, kun työkalua otetaan laajemmin käyttöön Yritys X:n toiminnassa. Käytön lisääntyessä voidaan tunnistaa uusia mahdollisuuksia prosessien optimointiin ja työkalun jatkokehitykseen. Lisäksi kerätty data ja käyttäjäkokemukset auttavat tunnistamaan lisäominaisuuksia, jotka voivat edelleen tehostaa hinnoitteluprosessia.

7 JOHTOPÄÄTÖKSET

Projektin tavoitteena oli luoda selainpohjainen hintalaskurisovellus Yritykselle X. Opinnäytetyöstä kuvataan, miten on toteutettu toimiva ja luotettava hintalaskuri. Projektin tulokset osoittavat, että kehitetty sovellus vastaa yrityksen tarpeisiin ja tarjoaa tehokkaan työkalun hinnoittelun hallintaan. Sovelluksen avulla käyttäjät voivat suorittaa laskennan suoraan järjestelmässä ja kaikki syötteet tallentuvat tietokantaan.

Projektissa käytettiin ketterää kehitysmallia, jossa sovellus kehitettiin iteratiivisesti ja muutoksia tehtiin käyttäjäpalautteen perusteella. Tämä osoittautui toimivaksi lähestymistavaksi, sillä palautteen perusteella pystyttiin tekemään parannuksia käyttöliittymään ja lisäämään käyttäjien toivomia ominaisuuksia.

Aiemmin hinnoittelu toteutettiin Excel-taulukoilla, mikä teki ylläpidosta, hinnoittelujen tallentamisesta ja datan käsittelystä haastavaa. Hintalaskurin kehittämisen myötä hinnoittelun hallinta on nyt huomattavasti käyttäjäystävällisempää ja tehokkaampaa. Kaikki laskelmat tallennetaan keskitetysti tietokantaan, mikä mahdollistaa niiden helpon haun, muokkauksen ja analysoinnin. Tämä vähentää manuaalista työtä ja parantaa tietojen eheyttä, sillä kaikki hinnoittelutiedot säilyvät yhdessä paikassa ja ovat helposti saatavilla tarpeen mukaan.

Teknologioiden valinta tehtiin täysin sen perusteella, mitä minä ja kollegani olemme aikaisemmin käyttänyt. Uusien teknologien opettelemiseen olisi mennyt vain turhaa aikaa ja nykyiset teknologiat olivat sopivat sovelluksen kehitykseen. Aikaisemmin käytetyt teknologiat vahvistavat sovelluksen onnistumista ja luotettavuutta, koska jo osattu teknologia vähentää virhemarginaalia.

7.1 Haasteet ja kriittinen arviointi

Projektin aikana kohdattiin useita haasteita, joista keskeisempiä olivat teknologian valinta ja soveltaminen, iteratiivinen kehitys ja asiakaspalaute sekä tietoturvaan liittyvät muutokset. Projektiin valittiin tuttuja teknologioita, mutta niiden soveltaminen uuteen kontekstiin vei aikaa ja osoittautui välillä haastavaksi. Vanhat ja luotettavat teknologiat tarjosivat vakautta, mutta välillä rajoittivat uusien ratkaisujen kokeilua. Jälkikäteen tarkasteltuna uusien teknologioiden opettelu olisi voinut tuoda lisäarvoa pitkällä tähtäimellä erityisesti suorituskyvyn näkökulmasta.

Toinen merkittävä haaste liittyi iteratiiviseen kehitykseen. Sovellukseen Yrityksellä X oli selkeät mieltymykset ja vaatimukset, mikä teki jatkuvasta palautteen huomioimisesta ajoittain haastavaa. Iteratiivinen kehitysprosessi on tehokasta, mutta se vaati tiivistä yhteistyötä ja joustavuutta asiakkaan muuttuvien tarpeiden kanssa. Kehitystapa varmisti lopulta, että sovellus vastasi odotuksia. Jälkikäteen tarkasteltuna kehitysprosessia olisi voitu parantaa järjestämällä tiheämmin palavereita Yrityksen X kanssa, jotta kehityksen suunta olisi voitu varmistaa aiemmin. Sopivien aikataulujen löytäminen osoittautui kuitenkin haastavaksi, mikä korostaa ennakoivan suunnittelun ja tehokkaan ajankäytön merkitystä vastaavissa projekteissa.

Tietoturvaan liittyvät muutokset olivat myös merkittävä oppimiskokemus projektissa. Aluksi laskentatoiminnot oli toteutettu frontendissä, mutta myöhemmin ne siirrettiin backendiin tietoturvasyistä. Päätös tehtiin myöhäisessä vaiheessa, mikä aiheutti ylimäärästä työtä. Kokemus osoitti, että tietoturvan huomioiminen jo alkuvaiheessa olisi säästänyt aikaa ja resursseja. Jatkossa on tärkeää arvioida ohjelmiston tietoturva jo alkuvaiheessa, jotta muutostarpeet voi tunnistaa ja toteuttaa tehokkaasti.

Projekti oli henkilökohtaisesti ensimmäinen suuri ohjelmistoprojekti, niin sen myötä kertyi arvokasta oppia ohjelmistokehitysprosessista ja sen kriittisistä osa-alueista. Yksi projektin suurimmista opeista oli tehokas

kommunikointi asiakkaan kanssa sekä sovelluksen esittely heille. Opin, kuinka tärkeää on selkeästi perustella kehitysratkaisut, kuunnella asiakkaan tarpeita ja mukauttaa sovellusta yrityksen toiveidensa mukaisesti. Lisäksi sain arvokasta kokemusta siitä, miten esitellä teknisiä ratkaisuja ymmärrettävästi ja havainnollistaa sovelluksen toiminnallisuuksia asiakkaalle, jotta he voivat antaa rakentavaa palautetta kehitysprosessin aikana.

7.2 Kehitysehdotukset ja jatkokehitys

Hintalaskuri on jo osittain käytössä ja vastaan Yrityksen X keskeisiin tarpeisiin, niin kehitystyö jatkuu edelleen uusien vaatimusten ja palautteiden pohjalta. Esiin on noussut tarve laajentaa käyttöoikeuksien hallintaa, jotta eri käyttäjäryhmille voidaan määrittää tarkempia oikeuksia järjestelmään.

Kehitysehdotuksia on myös tullut sovelluksen integraatiosta muiden järjestelmien kanssa, kuten toiminnanohjausjärjestelmään. Tämä mahdollistaisi hinnoittelutietojen hyödyntämisen laajemmin ja vähentäisi manuaalisen tietojen siirtämistä.

Sovelluksen käytettävyyttä ja suorituskykyä voidaan edelleen parantaa. Käyttöliittymää voidaan kehittää käyttäjäystävällisemmäksi ja tehokkaammaksi. Suorituskyvyn optimointia voidaan jatkaa erityisesti suurten tietomäärien käsittelyssä, jotta hintalaskuri pysyy nopeana myös käytön laajentuessa.

Mielenkiintoisia jatkokehitysmahdollisuuksia olisi tutkia, miten hintalaskentasovellus voidaan optimoida koneoppimisen avulla tai toteuttaa laajempi käytettävyystutkimus eri käyttäjäryhmien kanssa tai vertailla, kuinka paljon tehokkaampaa selainpohjainen ratkaisu on Excel-pohjaiseen ratkaisuun verrattuna pitkällä aikavälillä.

Yritys X on ilmaissut kiinnostuksena sovelluksen jatkokehitykseen ja sen käyttöönoton laajentuessa uusia tarpeita ja parannuskohteita tulee ilmi. Kehitys tulee jatkumaan ja tavoitteena on varmistaa, että hintalaskuri pysyy yrityksen tarpeiden mukaisena.

LÄHTEET

Duckett, J. (2011). *HTML & CSS: Design and Build Websites*. John Wiley & Sons. Noudettu 12.1.2025 osoitteesta https://wtf.tw/ref/duckett.pdf

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media. Noudettu 15.1.2025 osoitteesta https://books.google.fi/books?hl=en&lr=&id=b2bdDwAAQBAJ&oi=fnd&pg=PT24&dq=what+is+javascript+in+software+development&ots=6geGZxClM2&sig=bLHwCFHKKQmukolrVW8VCKT6API&redir_esc=y#v=onepage&q=what%20is%20javascript%20in%20software%20development&f=false

Jadhav, K., & Gonsalves, A. (2020). *Role of Node.js in modern web application development*. International Research Journal of Engineering and Technology (IRJET), 7(6), 1632–1636. Noudettu 16.1.2025 osoitteesta https://www.irjet.net/archives/V7/i6/IRJET-V7I61149.pdf

Krishnaswamy, J. (2010). *Microsoft SQL Azure Enterprise Application Development*. Packt Publishing. Noudettu 16.1.2025 osoitteesta https://books.google.fi/books?hl=en&lr=&id=gRTKKUAZy-bwC&oi=fnd&pg=PT11&dq=what+is+azure+sql&ots=R6nfsMPJ_w&sig=mDqOYnOllu-MjO9QifLi-_eRoPg&redir_esc=y#v=onepage&q&f=false

Huttenlocher, D., & Spoonhower, D. (2002). *Principles of software development*. Cornell University. Noudettu 19.1.2025 osoitteesta https://www.cs.cornell.edu/~dph/papers/principles.pdf

GitHub Docs. (n.d.). *About GitHub and Git.* Noudettu 22.1.2025 osoitteesta https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

Microsoft Learn. (n.d.). *What is Microsoft Entra?* Noudettu 29.1.2025 osoitteesta https://learn.microsoft.com/en-us/entra/fundamentals/what-is-entra

Mozilla Developer Network. (2024). *HTML: HyperText Markup Language.* Noudettu 2.2.2025 osoitteesta https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla Developer Network. (2024). *CSS: Cascading Style Sheets.* Noudettu 3.2.2025 osoitteesta https://developer.mozilla.org/en-US/docs/Web/CSS

Mozilla Developer Network. (2024). *JavaScript (JS)*. Noudettu 4.2.2025 osoitteesta https://developer.mozilla.org/en-US/docs/Web/JavaScript

Microsoft. (2023). *Azure SQL Database documentation.* Noudettu 6.2.2025 osoitteesta https://learn.microsoft.com/en-us/azure/azure-sql/

Mozilla Developer Network. (2024). *Introduction to Express/Node.js*. Noudettu 7.2.2024 osoitteesta https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction