SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

FUTURISTINEN JA SUJUVA KÄYTTÄJÄKOKEMUS WEBSOVELLUKSESSA

TEKIJÄ: Heini Saastamoinen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Heini Saastamoinen	
Työn nimi Futuristinen ja sujuva käyttäjäkokemus websovelluksessa	
Päiväys 25.4.2021	Sivumäärä/Liitteet 28
Toimeksiantaja/Yhteistyökumppani(t) Mikael Jokela	

Tiivistelmä

Työn tavoitteena oli toteuttaa websovellus, joka toimii liveroolipelin tukena. Työssä kiinnitettiin erityistä huomiota sovelluksen yhteensopivuuteen liveroolipelin futuristisen teeman kanssa sekä sujuvaan käyttökokemukseen. Sovelluksen tärkeimpiä ominaisuuksia olivat peliin liittyvien tietojen tallentaminen ja muokkaus, pikaviestikeskustelut ja maksaminen virtuaalivaluutalla. Työn tilaaja oli yksityishenkilö, joka toimii Totentanz-liveroolipelien järjestäjänä.

Sovellus toteutettiin Javascriptillä hyödyntäen React-kirjastoa selainpuolella ja Node.js-kirjastoa palvelimen puolella. Tietokanta toteutettiin MongoDB-tietokannalla.

Projektin lopputuloksena saatiin tavoitteisiin vastaava websovellus, joka vaatii vielä hiomista ja testaamista, mutta sisältää kaikki vaaditut ominaisuudet.

Avainsanat

websovellus, liveroolipeli, saavutettavuus, ulkoasu

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Heini Saastamoinen	
Title of Thesis Futuristic and Smooth User Experience in a Web Application	
Date 25 April 2021	Pages/Appendices 28
Client Organisation /Partners Mikael Jokela	

Abstract

The purpose of this project was to produce a web application that would work alongside a live-action roleplaying game. This project was especially focused on creating a smooth user experience that would match the futuristic theme of the live-action roleplaying game. The main features were saving and editing game related info, instant messaging, and virtual currency transactions. The thesis was commissioned by a private person who organizes Totentanz games.

The application was developed with Javascript using the React framework on front-end and Node.js on back-end. The database was made with MongoDB.

The result of this project was an application that requires more polishing and testing but fulfils all demanded features.

Keywords

web application, live-action roleplaying game, accessibility, layout

SISÄLTÖ

1 TERMILUETTELO

ARIA tai WA-ARIA = Web Accessibility Initiative's Accessible Rich Internet Applications, internetin saavutettavuuden standardi, jota näytönlukuohjelmat noudattavat (Google.com, 2020).

CSS = Cascading Style Sheets, websovelluksille tarkoitettu ohjelmointikieli, joka määrittää sen, miltä HTML:n kautta rakennetut komponentit näyttävät (Refnes Data, 2021).

Framework = viitekehys, koodikirjasto, joka tuo valmista koodia sovelluksen pohjaksi.

HTML = Hyper Text Markup Language, websovelluksille tarkoitettu ohjelmointikieli, joka keskittyy ulkonäön ja komponenttien rakentamiseen (Refnes Data, 2021).

JavaScript = websovelluksille tarkoitettu ohjelmointikieli, joka tuo sovellukseen toiminnallisuuden.

JSON = JavaScript Object Notation, tallennusmuoto, jossa JavaScript-oliot tallennetaan tavallisena tekstinä (Refnes Data, 2021).

Kyberpunk = 1980-luvulla muodostunut tieteiskirjallisuuden alagenre, joka tarkastelee suuryhtiöiden hallitsemaa lähitulevaisuuden dystopiaa (Neondystopia.com, 2018).

REST = Representational State Transfer, http-protokollaan perustuva arkkitehtuurimalli.

Tietokanta = palvelu, joka tallentaa sovelluksen datan, jotta siihen pääsee käsiksi missä vain.

2 JOHDANTO

Tämän opinnäytetyön tavoitteena on tuottaa kokonainen websovellus, joka on ottanut visuaalisuuteensa inspiraatiota tulevaisuuteen sijoittuvista videopeleistä Cyberpunk 2077 ja Shadowrun Returns. Molemmat kuuluvat tieteisfiktion alagenreen nimeltä kyberpunk, joka käsittelee jättiyritysten hallitsemaa lähitulevaisuuden dystopiaa (Neondystopia.com, 2018). Genren visuaaliseen tyyliin kuuluvat synkät suurkaupungit, neonvalot, kehittynyt teknologia ja punktyyli.

Sovelluksen käyttötarkoitus on olla samaan genreen kuuluvan liveroolipelin tukena. Liveroolipeli on eräänlainen improvisaatioteatterin ja pelin yhdistelmä, jossa pelaajat elävät roolihahmoinaan pelin ajan. Sovelluksen tehtävänä on toimia kommunikointikanavana pelinjohdon ja pelaajien välillä, tallennuspaikkana hahmotiedoille, pikaviestikanavana hahmojen välillä ja maksusovelluksena virtuaalivaluutalle.

Käyttöympäristö luo tietyt vaatimukset sovellukselle: sen täytyy toimia varmasti ja sujuvasti tositilanteessa sekä tuntua pelimaailman jatkolta, yhteensopivalta sen kanssa. Tämän takia sen täytyy noudattaa samaa futuristista tyyliä kuin sen inspiraationa olevat pelit. Sen kehityksessä pitää myös kiinnittää erityistä huomiota optimointiin mobiililaitteille, jotka muodostavat suurimman osan sen tarkasteluun käytetyistä laitteista.

Tässä opinnäytetyössä on kuvankaappauksia valmiista sovelluksesta. Näissä kuvissa esiintyvät nimet ja muut tiedot ovat keksittyjä, eikä niillä ole mitään yhteyttä oikeisiin henkilöihin.

3 FUTURISTISUUDEN MÄÄRITELMÄ

Monella on jonkinlainen mielikuva mielessään, kun puhutaan futuristisuudesta. Siitä saattaa tulla mieleen 1970-luvun paikkeilla alkunsa saaneet tieteisfiktiot kuten Star Trek ja Star Wars, tai ehkä kirkkaanvalkoinen sisustus ja robotti-imurit. Tulevaisuus näyttää erilaiselta eri aikoina, ja viime vuosisadan puolella luodut tulevaisuuskuvat näyttävät nykyään auttamattoman vanhanaikaisilta.

Tässä projektissa futuristisuudella tarkoitetaan nimenomaan tämän ajan tulevaisuudenkuvaa, joka kuitenkin ammentaa kyberpunk-genren estetiikasta. Kyberpunk syntyi samoihin aikoihin Star Trekin ja Star Warsin kanssa (Neondystopia.com, 2018), mikä tekee myös siitä alttiin vanhanaikaisuudelle. Tilannetta kuitenkin korjaavat uudet julkaisut kuten videopeli Cyberpunk 2077, joka on tuonut genreä nykyaikaan ja näyttänyt, millainen se olisi nykyaikana luotuna.

On tärkeää tehdä ero kyberpunkin ja muun tieteisfiktion välille: kyberpunk sijoittuu lähitulevaisuuteen, kun muu tieteisfiktio voi tapahtua jopa tuhansien vuosien päässä tai vierailla planeetoilla. Kyberpunkissa kaikki tuntuu modernilta ja jännittävältä, mutta ei liian kaukaiselta nykyaikaan verrattuna. Ehkä sen houkuttavuus on juuri uskottavuudessa: maailma voi näyttää tältä muutaman vuosikymmenen kuluttua.

Genren estetiikassa tuntuu olevan ikuinen yö ja loppumattomat kaupungin kadut. Luontoa ei ole tai se on jossain kaukana kamerasta. Musta ja neonvärit ovat valtaosassa. Kun katsoo esimerkiksi inspiraationlähteeksi asetettujen pelien käyttöliittymiä, niissä on paljon pientä tekstiä, numeroita ja statistiikkoja. Pelaajalle luodaan vaikutelma siitä, että jotain monimutkaista ja teknistä on tapahtumassa. Alla olevassa kuvassa näkyy suurennettuna kuva koristetekstistä, joka on Cyberpunk 2077:n latauspalkin yhteydessä (Kuva 1).

Kuva 1. Esimerkki latauspalkin koristeesta (CD Projekt Red, 2020)

4 KÄYTTÖKOHTEENA LIVEROOLIPELI

Sovellusten käyttäminen liveroolipelin tukena on Suomessa vielä melko vähäistä. Ainoa julkisuutta saanut sovellus on IN-Feels-sovellus, jota voi käyttää peliohjeiden jakamiseen minkälaisissa liveroolipeleissä tahansa (ellioi, 2020). Tämän projektin tavoite erosi IN-Feels-sovelluksesta siinä, että se oli räätälöity yhdelle, tietylle pelille ja sisälsi paljon muita ominaisuuksia.

Liveroolipelissä jokaisella pelaajalla on oma hahmonsa, jolla on taustatarina, tavoitteita ja kontakteja muihin hahmoihin. Pelaaja voi pelata millaista hahmoa tahansa, eivätkä ulkonäkö tai sukupuoli rajoita mahdollisuuksia. Joihinkin peleihin pelaajat luovat itse omat hahmonsa, mutta Totentanz tarjoaa pelaajille valmiiksi suunnitellut hahmot. Hahmojen kirjoitus on usean henkilön projekti, johon yleensä käytetään pilvipalvelua, jossa työn jakaminen ja tallentaminen on helppoa. Epäkäytännöllisyys tulee vastaan sitten, kun hahmot jaetaan pelaajille esimerkiksi sähköpostin välityksellä. Jotkut pelaajat tulostavat omat hahmotietonsa, mutta jopa kymmenen sivun paperinivaskan kanssa ei ole käytännöllistä kulkea pelipaikalla. Toiset joutuvat kaivamaan dokumenttinsa sähköpostin ja pilvipalveluiden syöveristä, mikä ei sekään ole kaikkein kätevintä.

Pelissä tietenkin kannustetaan kommunikoimaan kasvotusten, mutta on vain järkeenkäypää, ettei hämäriä tapaamisia sovita keskellä ihmismassaa. Monissa nykyaikaisissa liveroolipeleissä käytetään pikaviestipalveluita kuten Whatsapp tai Facebook Messenger hahmojen väliseen viestittelyyn, mutta niiden heikkous tulee siinä, ettei pelinjohto voi pysyä kärryillä tilanteesta kysymättä sitä suoraan pelaajilta. Tämä ongelma ratkeaa toteuttamalla pikaviestipalvelu sovelluksen sisällä. Näin voidaan myös järjestää sujuva pelaajanvaihto, jos varsinainen pelaaja estyy tulemasta peliin ja varapelaaja ottaa hahmon haltuunsa. Kun viestiketjut ovat sidottuja hahmoon eikä pelaajaan, myös uusi pelaaja näkee ne.

Useissa peleissä valuutalla on jonkinlainen rooli. Keskiaikapeleissä selvitään kolikkopussilla, mutta kun nykyaikaisissa peleissä hahmolla voi olla rahaa muutamasta eurosta tuhansiin, on epäkäytännöllistä simuloida ekonomiaa leikkirahoilla. Kun sovellukseen sisällyttää yksinkertaisen maksuominaisuuden, ekonomia toimii sujuvammin. Maksamisen ei tarvitse myöskään rajoittua vain toisiin hahmoihin: pelissä olevat palvelut kuten kasino ja baari voivat myös hyödyntää maksuominaisuutta.

Tekniikan suhteen liveroolipeli ei ole erityisen haastava käyttökohde. Totentanz järjestetään kaupunkiympäristössä, joten nettiyhteys ja sähkö ovat kaikkien saatavilla. Käyttäjämäärä, mahdollisesti jopa 60 pelaajaa, ei ole paljon nykyaikaiselle sovellukselle, etenkään kun kaikki eivät käytä sovellusta yhtä aikaa.

5 SOVELLUKSEN TEKNINEN PUOLI

Sovelluksen tehtävänä on toimia kommunikaatioväylänä pelaajien ja pelinjohdon välillä. Sovelluksen teknisessä toteutuksessa kiinnitettiin erityistä huomiota sen nopeuteen ja toimintavarmuuteen, sillä pelin aikana kukaan ei halua alkaa pohtia, miksi omat hahmotiedot eivät näy. Tämä tarkoitti huolellista testausta, virheiden nappausta ja niistä palautumista sekä selkeää kommunikaatiota käyttäjälle.

Kokonaisuutena sovellus on hyvin tyypillinen websovellus. Se näyttää tietoja käyttäjälle, kerää tietoja lomakkeiden avulla ja tallentaa niitä tietokantaan. Poikkeavin ominaisuus siinä on pikaviestipalvelu, joka sekin on nykyään yleistynyt erilaisten asiakaspalveluchattien myötä.

Vaikka tämä opinnäytetyö ei keskity websovelluksen tekniikkaan, se on silti olennainen osa sujuvan käyttäjäkokemuksen luomista. Pelkkä huolella suunniteltu ulkoasu ei takaa käyttäjäystävällisyyttä. Toisaalta tekninen puoli myös ohjaa ulkoasun suunnittelua: se asettaa tarpeen napeille ja lomakkeille, sekä tekee joistakin tyyliseikoista paljon yksinkertaisempia toteuttaa kuin toisista.

5.1 Valitut tekniikat

Websovellusta suunnitellessa tekotavoista löytyy huimasti valinnanvaraa. Sovelluksen tekeminen pelkän HTML:n ja Javascriptin varaan ei ole kannattavaa enää nykypäivänä, ellei kyse ole äärimmäisen yksinkertaisesta sovelluksesta. Framework eli viitekehys nopeuttaa ja helpottaa ohjelmointia, kun kaikkea ei tarvitse toteuttaa itse alusta lähtien (Andras, 2017). Tämän takia projektin pohjaksi valittiin tämän hetken suosituin viitekehys, React. Se jakaa sovelluksen pienempiin komponentteihin, joilla jokaisella on oma ”muistinsa” eli state. Komponentit voivat kommunikoida keskenään ja kun jokin osa sovelluksesta päivittyy, koko sivua ei tarvitse ladata uudelleen (Facebook Inc, 2021). React sopi tähän projektiin erityisesti siksi, koska sovelluksen osana on pikaviestitoiminto ja siihen kuuluvat ilmoitukset.

Hahmojen luontia helpottamaan otettiin tekstieditorikirjasto. Aluksi testattiin kirjastoa nimeltä Slate, joka mainosti itseään äärimmäisen mukautettavana (Slate, 2021). Slate osoittautui kuitenkin turhan monimutkaiseksi tietokantaan tallentamisen yhteydessä, joten se vaihdettiin toiseen kirjastoon nimeltä Pell. Se oli paljon yksinkertaisempi ja soveltui ympäristöön paremmin.

Tyylien muotoiluun saa kulumaan yllättävän paljon aikaa, vaikka ulkonäkö ei olisikaan pääasia sovelluksessa. Koska tässä projektissa se nimenomaan oli, prosessia tukemaan valittiin tyylikirjasto Bootstrap 5. Kirjasto oli entuudestaan tuttu ja siinä on panostettu sivuston skaalautuvuuteen ja nopeaan toimivuuteen (Bootstrap Team). Bootstrap auttaa erityisesti ulkoasun palasten asettelussa ja perustyyleissä, kuten lomakkeiden ja nappien muotoilussa.

Palvelimen puolella käytettiin Node.js- ja Express -kirjastoja sekä tietokantaan MongoDB:tä. Tätä kirjastojen yhdistelmää kutsutaan nimellä MERN-stack (MongoDB Inc., 2021). MongoDB on tietokanta, joka tallentaa tietoja kokonaisina dokumentteina selkeän hierarkkisuuden sijaan, jota useimmat muut tietokannat käyttävät. Koska sovelluksen tallentama tieto on hyvin vaihtelevaa mutta silti JSON-pohjaista, MongoDB sopi tähän tarkoitukseen mainiosti.

Pikaviestien toteutukseen käytettiin WebSocket-kirjastoa. WebSocket-protokolla mahdollistaa reaaliaikaisen kommunikoinnin palvelimen ja sovelluksen välillä. Siinä missä hahmojen data haetaan perinteiseen tapaan tekemällä sovelluksessa pyyntö palvelimelle, johon palvelin vastaa lähettämällä vaaditun datan, WebSocket mahdollistaa datan lähetyksen myös palvelimen aloitteesta.

5.2 Toiminnallisuus

Sovelluksella oli useita eri toimintoja. Sen päätoiminto oli hahmojen ja käyttäjien luonti, muokkaus, tarkastelu ja poisto. Toinen tärkeä ominaisuus oli kirjautuminen ja uloskirjautuminen, jotta pelaajat pääsevät käsiksi vain omiin tietoihinsa ja pelinjohto kaikkiin tietoihin. Koska kyseessä oli pienelle, suljetulle käyttäjäryhmälle tarkoitettu sovellus, sen tietoturvaan ei panostettu suuresti. Sovellukseen ei tulla tallentamaan henkilökohtaisia tietoja.

Myös pikaviestipalvelu oli isossa osassa sovellusta. Pelaajat pystyivät kommunikoimaan keskenään hahmoinaan, ja pelinjohto pystyi tarkkailemaan näitä keskusteluita ja ottamaan yhteyttä hahmoihin. Jokaisesta viestistä tuli myös ilmoitus kirjautuneelle käyttäjälle, jos hän oli muussa osassa sovellusta kuin chatissa.

Pienemmissä rooleissa olivat virtuaalivaluutalla maksaminen hahmojen välillä ja pelinjohdolle maksutapahtumien tarkastelu. Koska tässäkään asiassa ei ollut yhteyttä oikean elämän tietoihin ja varallisuuteen, maksutapahtumien tietoturvaan ei panostettu kovin paljoa. Jos jokin virhe sattuisi maksun aikana, pelinjohto voisi helposti päivittää hahmojen tietoja manuaalisesti.

5.3 Palvelin

Projektiin kuului websovelluksen tueksi palvelin. Sen tehtäviä olivat kommunikaatio selainpuolen ja tietokannan välillä sekä pikaviestien hallinnointi. Kehitysvaiheessa palvelin toimi paikallisella koneella, mutta kun sovellus tulee oikeaan käyttöön, sitä pitäisi ylläpitää joko erillisellä tietokoneella pelipaikan verkossa tai internetissä.

Palvelimella oli käytössä Express-kirjasto, joka auttoi palvelinkutsujen reitityksessä. Kun selain otti yhteyttä palvelimeen, sen käyttämä osoite kertoi, mitä palvelimelta haluttiin. Esimerkiksi pääte ”chat/” palautti kaikki tallennetut keskustelut, kun taas ”chat/102” palautti keskustelun, jonka id oli 102. Myös http-metodilla oli väliä, sillä palvelin noudatti REST-arkkitehtuurimallia. Reititysten kirjoittamisessa auttoi suuresti niiden suunnittelu etukäteen: kun oli tiedossa, millaisia asioita palvelimelta voidaan haluta, oli helppoa kirjoittaa alustavat metodit. Ne toki muuttuivat hieman kehityksen aikana, kun turhia kutsuja jätettiin pois ja uusia tuli tilalle.

Kaikki monimutkainen logiikka tietokantakutsuissa hoidettiin palvelimen puolella. Esimerkiksi käyttäjää muokatessa tapahtui parhaimmillaan kolme toimintoa: käyttäjä muokattiin, käyttäjälle asetetun hahmon tietoihin päivitettiin merkintä käyttäjästä ja vanhalta käyttäjältä poistettiin asetettu hahmo. Oli selkeämpää, kun palvelin hoiti nämä kaikki ja selainpuolen täytyi tehdä vain yksi palvelinkutsu, jossa oli muokatun käyttäjän uusi data. Näiden ketjutettujen tietokantakutsujen toteuttaminen oli melko hankalaa, sillä osa niistä vaati edellisen kutsun valmistumista, kun taas toiset pystyttiin suorittamaan asynkronisesti eli toisistaan riippumatta. Koodissa kannattaa suosia asynkronisia kutsuja aina kun mahdollista, jotta eliminoidaan odottamisesta syntyvä tehokkuuden menetys. Näiden isojen

operaatioiden esittely veisi liikaa tilaa tästä raportista, mutta alla on kuva yksinkertaisesta operaatiosta (Kuva 2). Siinä otetaan yhteys tietokantaan, järjestetään tulokset ja palautetaan ne selaimelle.

```
// Fetch all payments
app.get('/transaction/', (req, res) => {
  // Sort by timestamps, newest first
  db.collection('transactions').find().sort({ "time": -1 }).toArray(function (err, result) {
    if (err) throw err
    res.send(result)
    console.log("Fetched all payments")
    db.close
  })
})
```

Kuva 2. Palvelimen operaatio, joka palauttaa kaikki maksutapahtumat (Saastamoinen 2021, CC BY).

Pikaviestien toteuttamiseen käytetty WebSocket-kirjasto oli yllättävän yksinkertainen ja helppo ottaa käyttöön. Käytännössä käyttäjän kirjautuessa sisään selain lähetti palvelimelle viestin, jossa oli käyttäjän tiedot. Palvelin muisti kaikki kirjautuneet käyttäjät, mutta jos käyttäjä kirjautui ulos tai sulki selaimen, yhteys katkesi ja palvelin unohti käyttäjän. Kun käyttäjä avasi jonkin pikaviestikeskustelun, palvelin lähetti kyseisen keskustelun viestihistorian. Kun käyttäjä lähetti uuden viestin, palvelin kuulutti (eng. broadcast) sen kaikille, jotka ovat osana keskustelua. Jos kirjautunut käyttäjä sai viestin ja oli keskustelussa, keskusteluun ilmestyi uusi viesti. Jos kirjautunut käyttäjä oli muualla sovelluksessa, hän näki ilmoituksen sen sijaan. Palvelin myös tallensi jokaisen viestin tietokantaan.

5.4 Virheisiin varautuminen

Jokainen sovellus tulee kohtaamaan virheitä, vaikka bugit olisikin saatu haravoitua koodista pois. Joskus palvelimeen ei saada yhteyttä, joskus käyttäjän laite on liian vanhanaikainen eikä tue uutta teknologiaa, ja joskus käyttäjä tekee jotain niin odottamatonta sovelluksessa, ettei sitä osattu ottaa huomioon testauksessa.

Varautuminen alkaa jo etusivulla, kun käyttäjä lataa sen ensimmäisen kerran. Koska sivusto tukeutui hyvin vahvasti Javascriptin käyttöön, etusivulla näkyi pelkästään varoitusviesti, jos käyttäjän selaimessa ei ollut sallittu Javascriptin käyttöä. Tämän lisäksi käyttäjällä ei ollut oikeastaan mahdollisuutta törmätä ongelmiin ennen sisäänkirjautumista.

Sovelluksessa jokaiseen tietokantakutsuun lisättiin *catch*-metodi, joka nappaa kutsun aikana tapahtuvat virheet ja estää sovelluksen kaatumisen. Kohdatessaan virheitä se näytti käyttäjälle virheviestin, joka selosti, mitä tapahtui ja miten asiassa voidaan edetä. Alla olevassa kuvassa on esimerkki tällaisesta metodista (Kuva 3).

```
    // Insert/update character
    fetch(url, {
      method: 'POST',
      mode: 'cors',
      headers: {
        'Content-Type': 'application/json'
      },
      body: data
    })
      .then(response => response.json())
      .then(parsed => this.setState({ redirect: <Redirect to="/admin" /> }))
      .catch(error => this.showError(error))
  }
}
showError(error) {
  // Translate the most common error
  if (error.message === "NetworkError when attempting to fetch resource.")
    this.setState({ error:
      "Yhteyttä palvelimeen ei saatu. Yritä hetken kuluttua uudelleen tai ota yhteys pelinjohtoon." })
  // If the error is something else, show it anyway
  else
    this.setState({ error: error.message })
  // Show alert element
  const alert = document.getElementById("errorMessage")
  alert.classList.add('show')
  setTimeout(function () { alert.classList.remove('show') }, 7000);
}
```

Kuva 3. Esimerkki virheen nappaamisesta ja virheviestin näyttämisestä (Saastamoinen 2021, CC BY)

Kaikissa paikoissa, joissa oli vain tietyille käyttäjille näkyvää informaatiota, kirjautuneen käyttäjän rooli tarkastettiin ennen tietojen näyttämistä. Näin esimerkiksi pelkällä osoitteen arvaamisella ei päässyt käsiksi vääriin tietoihin. Kun käyttäjä kirjautui sisään, selaimen evästeisiin tallennettiin tämän kirjautumistunnus, jotta sivua päivitettäessä tai sinne palattaessa sisäänkirjautuminen tapahtuisi automaattisesti. Tässä tilanteessa varauduttiin virheisiin siten, että kirjautuminen tapahtui palvelimen kautta joka kerta, jotta vältetään poistetun käyttäjän sisäänpääsy selaimen muistiin tallennetulla tunnuksella. Kirjautumistunnus myös vanhentui selaimen sulkeutuessa.

Aina kun sovelluksessa käytetään tietokantaa, on tärkeää varautua vahingolliseen syötteeseen. SQL-injektio, eli tietokantakomentojen kirjoittaminen sivuston syötteeseen on yksi yleisimmistä hakkerointikeinoista websovelluksissa (Refnes Data, 2021). Tähän voi parhaiten varautua tarkistamalla käyttäjän syötteen ja estämällä erikoismerkit. Tässä sovelluksessa se tehtiin kahteen kertaan. Ensimmäinen tarkastus on itse HTML-koodissa, jotta virheilmoituksissa voitiin hyödyntää selaimen omaa tekniikkaa. Mutta koska HTML-koodia voi muokata kehittäjätyökalulla, syötteestä poistettiin kaikki erikoismerkit vielä ennen sen lähettämistä palvelimelle. Alla on käytännön esimerkkejä tästä tekniikasta (Kuvat 4 ja 5).

```
const data = {
  login: this.state.login.replace(/["=';]/g, ""),
  userName: this.state.playerName.replace(/["=';]/g, ""),
  character: this.state.selectedCharacter,
  userType: 'player'
}
```

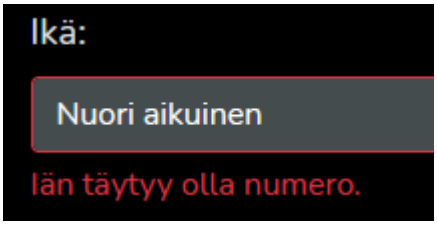
Kuva 4. Haitallisten merkkien poistaminen ennen datan lähetystä (Saastamoinen 2021, CC BY)

```
<input
minLength="6"
maxLength="20"
required
pattern="[a-öA-Ö\d]*"
id="loginInput"
class="form-control"
type="text"
name="login"
value={this.state.login}
onChange={this.handleChange}
placeholder="Käyttäjän kirjautumiseen käyttämä tunnus"
></input>
```

Kuva 5. Tekstikentän koodi, jossa on korostettuna sallittujen merkkien rajoitus (Saastamoinen 2021, CC BY).

5.5 Kommunikaatio käyttäjille

Tärkeä osa sujuvaa käyttäjäkokemusta on selkeä kommunikointi käyttäjälle. Jokaisen virheen yhteydessä käyttäjälle näytetään viesti, joka selittää, mitä on tapahtunut. Tämän lisäksi viestejä näytetään esimerkiksi kenttiä täyttäessä: jos kentän sisältö ei ole oikea (esimerkiksi ikäkentässä on kirjaimia numeroiden sijaan), kentän lähistölle tulee huomautus asiasta, kuten alla olevassa kuvassa on esitetty (Kuva 6). Käyttäjän tulee nähdä yhtä aikaa sekä virheilmoitus että virheellinen kenttä, eikä virheilmoitus saa peittää kenttiä. Ennen merkittäviä operaatioita, kuten tietojen poistoa tai maksun hyväksymistä, käyttäjälle näytetään varmistusikkuna, jossa hänellä on vielä mahdollisuus peruuttaa toimintonsa. Myös onnistuneista operaatioista näytetään viesti. (Irmeli Sinkkonen, 2009, s. 240.)

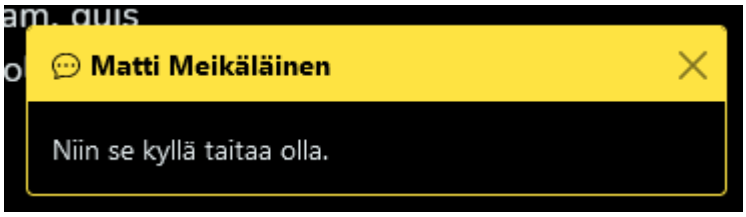


Kuva 6. Esimerkki syötekentän virheviestistä (Saastamoinen 2021, CC BY).

Pudotuslistoissa, eli valikoissa joissa on valittavissa yksi vaihtoehto useamman listasta, käyttöä selkeyttää vaihtoehtojen järjestäminen aakkosjärjestykseen (Irmeli Sinkkonen, 2009, s. 230). Tämä oli helppoa toteuttaa Javascriptin sort-toiminnolla, joka lajittelee datan haluttuun järjestykseen. Sovel-

lusta kehittäessä aakkosjärjestyksen vaikutusta ei vielä juurikaan päässyt toteamaan, mutta kun todellisessa käyttötilanteessa hahmoja voi olla jopa kuusikymmentä, järjestyksen varmasti huomaa. Lajittelun ei pitäisi myöskään vaikuttaa merkittävästi sovelluksen tehokkuuteen, sillä tietokoneelle kuudenkymmenen kohteen lajittelu tehokkaalla algoritmilla on hyvin nopea prosessi, joka tapahtuu ainoastaan silloin, kun data pitää hakea uudestaan.

Tärkeä osa kommunikaatiota ovat myös ilmoitukset, kun jotain käyttäjästä riippumatonta tapahtuu. Kun käyttäjä oli kirjautuneena sovellukseen ja toinen käyttäjä lähetti viestin keskusteluun, jossa hän oli osallisena, ensimmäinen käyttäjä sai ilmoituksen. Jos hän oli katsomassa kyseistä keskustelua, ilmoitusta ei tietenkään tällöin tullut. Ilmoitukset olivat riittävän näkyviä, jotta ne huomattiin, mutta ne eivät estäneet sivuston muuta käyttöä, kuten alla olevasta kuvasta on nähtävissä (Kuva 7). Ilmoituksen pystyi sulkemaan, mutta se myös katosi itsestään muutaman sekunnin kuluttua.



Kuva 7. Ilmoitus, joka kertoo uudesta viestistä keskustelussa (Saastamoinen 2021, CC BY).

5.6 Testaus

Sovelluksen testaus on tärkeää monesta eri syystä. Tärkein syy on tietysti löytää sovelluksen toiminnasta virheitä, jotta ne voidaan korjata ennen sovelluksen julkaisua. Vähemmän ilmiselvä syy on käytettävyystestaus, jossa sovelluksen käytössä kokematon käyttäjä testaa, kuinka helposti sovelluksen käyttö onnistuu (Irmeli Sinkkonen, 2009, s. 302). Käytettävyyden testaus oli erityisen tärkeää tämän sovelluksen kanssa, sillä sen käyttäjäryhmien täytyi oppia sen käyttö helposti ja nopeasti, jotta samaan aikaan pelattava liveroolipeli ei tulisi kärsimään uuden sovelluksen opettelusta.

Myös rasitustestaus oli aiheellista tämän sovelluksen kohdalla. Siinä hyödyntämällä useampaa käyttäjää tai bottia testataan sovelluksen ja erityisesti sen palvelimen toimivuutta, kun palvelinkutsuja tulee usein ja päällekkäin. Tämän opinnäytetyön aikarajoissa siihen ei kuitenkaan käytännössä ehditty.

Sovellusta testasi tämän projektin aikana lähinnä tilaaja, joka samalla tutustui uusiin muutoksiin sovelluksessa. Testaus oli kaukana virallisista testaustilaisuuksista, joita isommat sovelluskehitysprojektit järjestävät ennakkotöineen ja testiraportteineen (Irmeli Sinkkonen, 2009, s. 302), mutta siitä oli silti valtava hyöty sovelluksen kehityksessä. Sovelluksen kehittäjä sokeutuu työnsä heikkouksille ja epäloogisuuksille, kun taas ulkopuolinen käyttäjä törmää niihin nopeasti.

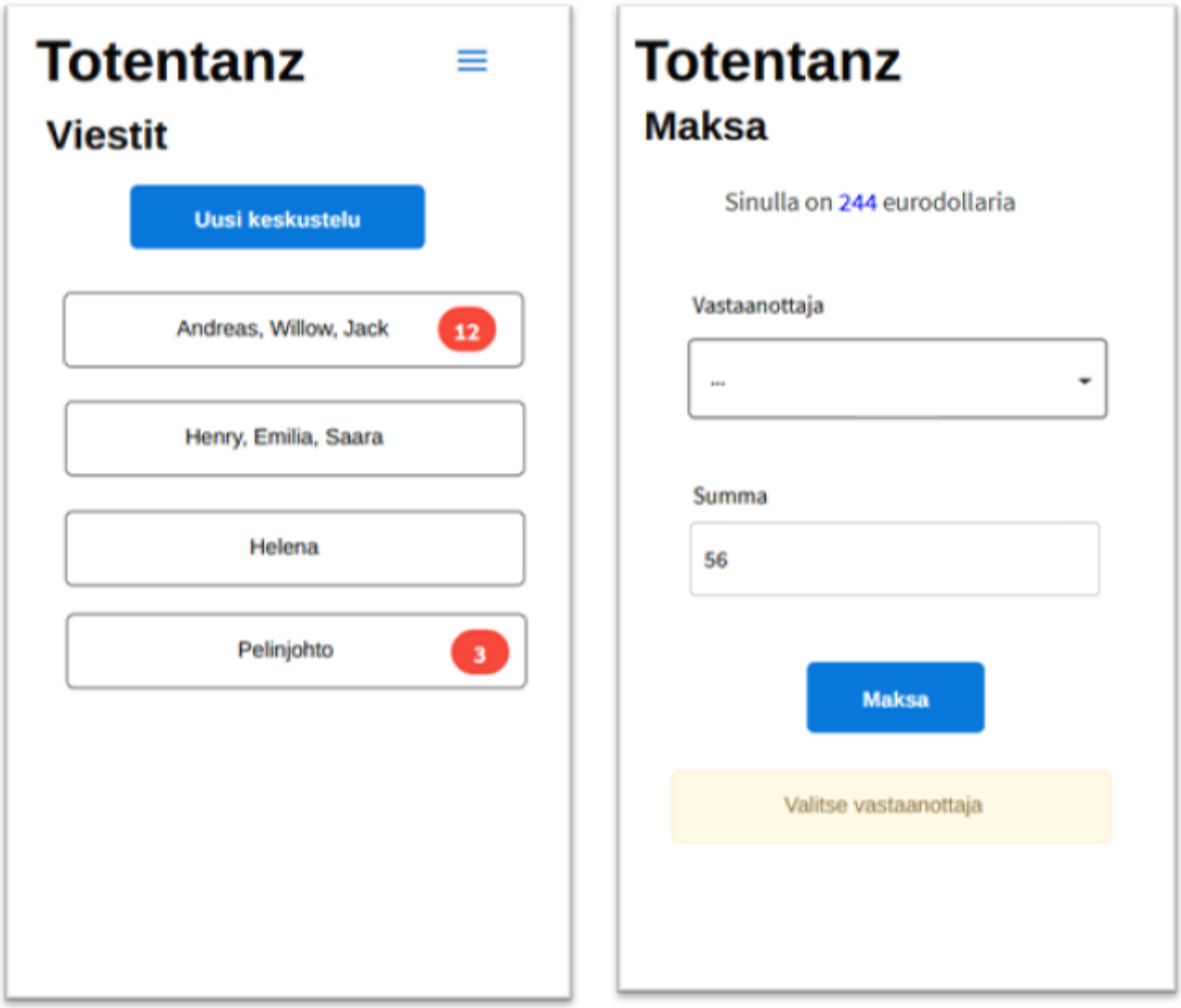
6 ULKONÄÖN SUUNNITTELUN VAIHEET

Kun sovelluksen runko oli valmistunut ja toimi riittävän vakaasti, halu päästä suoraan luomaan tyylejä sen päälle oli kova. Kuten projekteissa ylipäätään, myös visuaalisuuden teossa suunnitteluvaihe on äärimmäisen tärkeä. Huolellisella suunnittelulla vältetään tyylien luominen turhaan.

Visuaalista suunnittelua tehdessä huomio täytyy kiinnittää käyttäjiin ja käyttöympäristöön (Rebecca Hagen, 2017, s. 10). Tätä sovellusta tulisivat käyttämään aikuiset ihmiset, joiden teknologiset taidot voivat vaihdella paljonkin. Joukossa saattaa myös olla henkilöitä, joiden näkökyky tai motoriset taidot ovat heikentyneet. Sovellusta tultaisiin käyttämään eniten mobiililaitteilla ja kiireisissä tilanteissa. Näiden syiden vuoksi sovelluksen käytön täytyi olla helppoa ja nopeaa.

Sovelluksen tavoitteena oli myös olla visuaalisesti miellyttävä, jännittävä ja auttaa uppoutumaan pelin kuvitteelliseen maailmaan. Tasapainon löytäminen kauneuden ja käytettävyyden välillä voi olla haastavaa. Ulkonäön laiminlyönti näkyy amatöörimäisenä jälkenä, kun taas käytettävyyden unohtaminen epäselvinä toiminnallisuuksina, hitaana käyttökokemuksena ja hämmentävyytenä (Irmeli Sinkkonen, 2009, s. 243).

Kun kohderyhmän ja käyttöympäristön määrittely oli tehty, siirryttiin luonnosteluvaiheeseen. Luonnostelussa tavoitteena oli tuottaa suuntaa antavia kuvia siitä, miltä sovellus tulisi näyttämään. Ensimmäiset luonnosversiot tehdään usein yhä paperille (Rebecca Hagen, 2017, s. 20), mutta tässä projektissa käytettiin MockFlow-ohjelmaa luonnosten tekemiseen. Alla on nähtävissä viesti- ja maksusivun luonnokset mobiiliversiolle (Kuva 8).



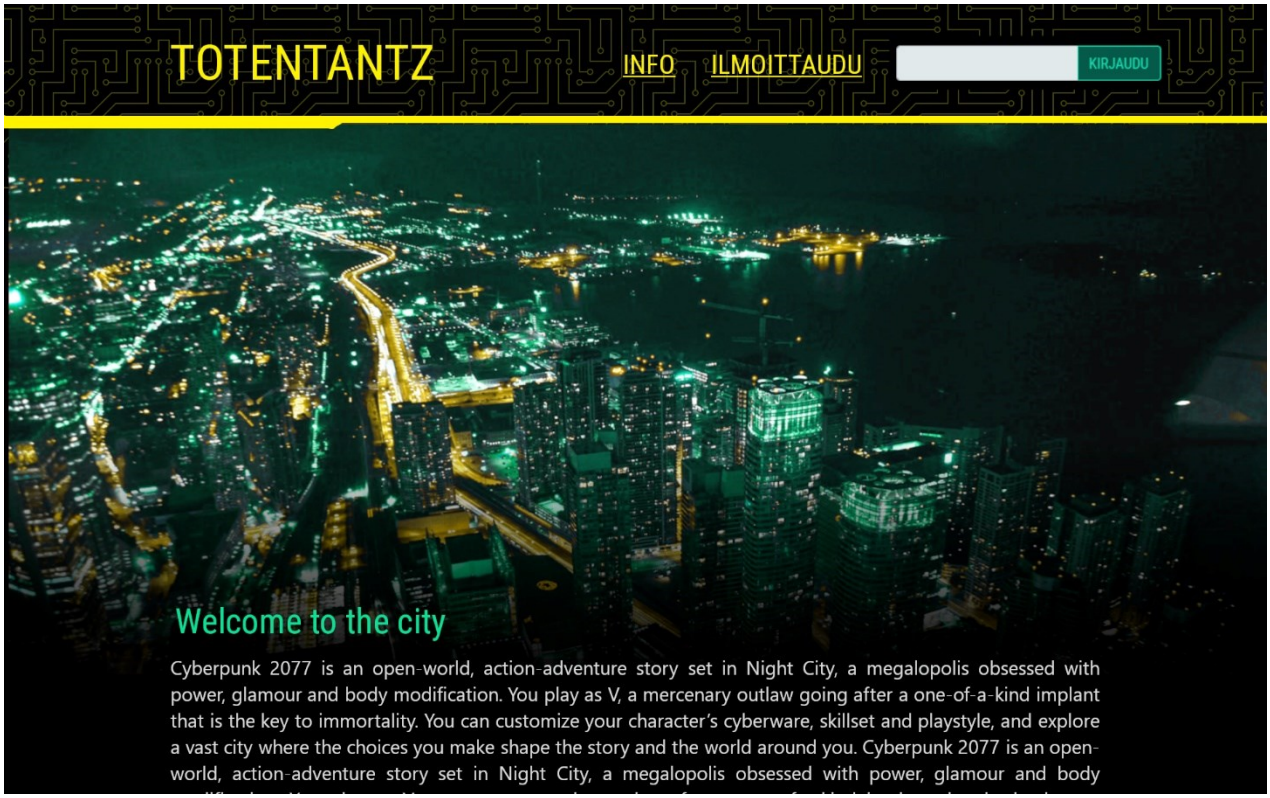
Kuva 8. Luonnos mobiiliversiosta (Saastamoinen 2021, CC BY)

Koska MockFlow antaa luoda kerralla vain kolme näkymää ja suunnitteluohjelman lisenssin hankkimiseen ei ollut budjettia, luonnoksien suunnittelu oli hieman kankeaa. Käytännössä kerralla tehtiin kolme näkymää, jotka lähetettiin hyväksyttäväksi työn tilaajalle, ja sen jälkeen niitä joko muokattiin toiveiden mukaan tai tallennettiin ja jatkettiin seuraaviin. Nämä kyseiset luonnokset kuvaavat ainoastaan elementtien sijoittelua ja kokoa, ei niinkään lopullista visuaalista ilmettä.

Ensimmäisten luonnosten jälkeen oli kaksi tapaa edetä. Voitaisiin joko luoda kerralla kaikki luonnokset, sitten viimeistellyt suunnitelmat lopullisesta ulkoasusta ja vasta sitten käydä käsiksi itse sovellukseen, tai voitaisiin luonnostella ja toteuttaa rakenne ja sen jälkeen viimeistellä sovellus väreillä ja muilla tyylivalinnoilla. Vaihtoehdoista päädyttiin jälkimmäiseen, sillä luonnokset harvoin etenevät käytäntöön ilman minkäänlaisia muokkauksia, joten täydellisen suunnitelman tekeminen voisi pikemminkin rajoittaa ja hidastaa prosessia, kun käytännön toteutuksen aikana heränneet ideat olisi vaikeampaa valjastaa käyttöön. Projektin hyvin vapaa toteutustapa myöskin puolsi tätä valintaa, sillä kehittämisessä sai käyttää omaa harkintaa melko pitkälle, eikä jokaista muutosta ja suunnitelmaa tarvinnut erikseen hyväksyttää tilaajalla.

Viimeistelyvaiheen luonnostelu aloitettiin keräämällä materiaalia kahdesta pelistä, joita tilaaja halusi käyttää ulkoasun inspiraationa: Shadowrun Returns ja Cyberpunk 2077. Pelien käyttöliittymästä otettiin kuvankaappauksia ja tilaajan kanssa pidettiin suunnittelutuokio, jossa kerättiin hänen mielipiteitänsä pelien elementtien ulkonäöstä. Tämän pohjalta luotiin kollaasi halutuista elementeistä, mikä tuli toimimaan lopun suunnittelun tukena.

Seuraavana vuorossa oli kuvankäsittelyohjelma ja luonnos ulkoasun lopullisesta ulkonäöstä. Kuten luonnoksien yleensäkään, tämän tarkoitus ei ollut olla ehdoton, kiveen hakattu ohjenuora, vaan parempien ideoiden syntyessä luonnoksesta voisi poiketa. Lopputulos on nähtävissä alapuolella (Kuvat 9 ja 10).



Kuva 9. Lopullinen luonnos isoille näytöille (Saastamoinen 2021, CC BY)

Kuva 10. Lopullinen luonnos mobiilinäytöille (Saastamoinen 2021, CC BY)

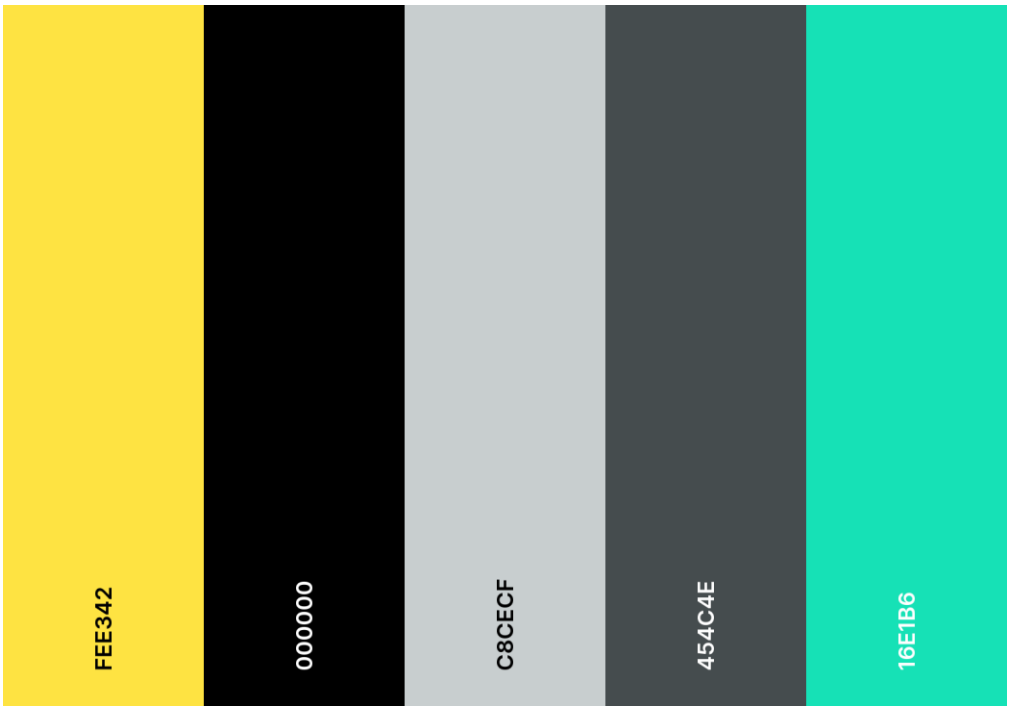
Joissakin projekteissa olisi vaadittua tehdä luonnoksia useammasta sivusta, mutta tässä projektissa aikaa ja vaivaa säästäen pelkkä etusivun luonnos oli riittävä. Kun parannusehdotukset oli toteutettu ja tilaaja oli tyytyväinen luonnosten ulkonäköön, niitä alettiin toteuttamaan käytännössä. Tyylien luominen voi olla sekavaa ja aikaa vievää, mutta selkeä luonnos ohjenuorana auttoi prosessissa. Muutama asia muuttui matkalla lopulliseen versioon, kuten kirjautumisnappi ja etusivulla käytettävä kuva.

6.1 Värit

Värimaailman valinnan taustalla oli paljon teoriaa. Käyttäjää voi ohjailla väreillä luomalla kontrastin avulla kiintopisteitä. Muuten tummassa taustassa kirkkaan värinen nappi vie käyttäjän huomion heti sivulle tullessa. Tätä voi käyttää hyödyksi, mutta siitä voi olla myös haittaa, jos kiintopiste ei edistä sivun haluttua käyttötapaa. Sen lisäksi värit ovat voimakkaita työkaluja tunteiden herättämiseen. Koska sovelluksen käyttäjäryhmä on kapea ja todennäköisesti suurimmilta osin suomalainen, kulttuurieroja värien ymmärtämisessä ei tarvinnut ottaa huomioon. (Rebecca Hagen, 2017, ss. 116-127.)

Suunnittelun pohjana käytetyissä peleissä on toisistaan hieman poikkeavat värimaailmat. Molemmat ovat pohjaväriltään tummia, mutta Shadowrunissa päävärinä on sininen pienillä keltaisen pilkahduksilla, kun taas Cyberpunk 2077:ssä hallitsevana värinä on punainen turkoosilla korostuksella. Toisaalta Cyberpunk 2077 käyttää markkinoinnissaan ja nettisivuillaan aivan erilaista palettia, jossa keltainen on hallitseva väri ja muut päävärit korostavat sitä.

Projektia varten tehtiin erilaisia väripaletteja käyttäen hyväksi coolors.co-sivuston palettityökalua. Osaan värit kerättiin pelien käyttöliittymästä tai mainoskuvista, osa sommiteltiin oman taiteellisen näkemyksen pohjalta. Lopuksi tilaaja valitsi paleteista mieleisensä, joka on nähtävissä alla (Kuva 11). Paletti on yhdistelmä molempia inspiraationa toimivia pelejä: keltainen on hyvin lähellä Cyberpunkin mainosten väriä, kun taas turkoosia esiintyi sekä Shadowrunissa että Cyberpunkin käyttöliittymässä.



Kuva 11. Lopullinen väripaletti (Saastamoinen 2021, CC BY)

6.2 Kirjasimet

Kirjasimet, toiselta nimeltään fontit, voidaan jakaa kategorioihin niiden tunnuspiirteiden mukaan (Rebecca Hagen, 2017, s. 90). Sekä Shadowrunissa että Cyberpunkissa kirjasimina on käytetty Sans Serif -kategorian fontteja. Niiden tunnuspiirteenä on pääteviivojen puuttuminen kirjainten päistä. Tämän lisäksi ne voidaan lokeroida alakategoriaan ”grotesque” eli groteskit fontit (Rebecca Hagen, 2017, s. 90). Kirjaimet ovat pyöreitä ja etenkin Cyberpunkissa kapiteelia on käytetty monessa kohtaa tehostekeinona.

Kirjasimilla on suuri vaikutus sivun ilmeeseen, vaikka sitä ei osaisi aina yhdistää niihin. Kirjasimet voivat antaa virallisen, leikittelevän tai vaikka trendikkään vaikutelman, ja väärin valittu kirjasin voi hämmentää käyttäjää. Jotkut kirjasimet ovat toisia selkeämpiä: esimerkiksi Sans Serif -kirjasimissa on usein ongelmana ison i-kirjaimen ja pienen l-kirjaimen sekoittuminen keskenään.

Nämä esikuvat mielessä kirjasimiksi valikoituivat lopulta Nunito leipätekstiin ja Roboto Condensed otsikoihin. Molemmat kuuluvat Sans Serif -kategoriaan. Kirjasimista on nähtävissä esimerkki alapuolella (Kuva 12).

Roboto Condensed Nunito

Kuva 12. Sovellukseen valitut fontit (Saastamoinen 2021, CC BY)

6.3 Kuvat

Alusta lähtien oli selvää, että etusivulle haluttiin nappaava kuva. Kuvissa ihmiset, ja erityisesti jotain tekevät ihmiset, kiinnittävät katsojan huomion (Rebecca Hagen, 2017, s. 140), joten kuvassa täytyi esiintyä ainakin yksi henkilö. Sivuston väriteema oli jo selvillä, joten kuvan täytyi joko sopia kyseisiin väreihin tai olla muokattavissa sellaiseksi. Kuvan täytyi myös olla vapaasti käytettävissä ja tarpeeksi hyvälaatuinen isoksi etusivun kuvaksi.

Valintaa varten etsittiin useampi kuva, joista tilaaja sai valita mieleisensä. Kuvapalveluna käytettiin sivustoa Pexels, joka sisältää ilmaisia, vapaasti käytettäviä ja ammattilaistason valokuvia. Valinnan voittanut kuva on nähtävissä alapuolella (Kuva 13).

Kuva 13. Alkuperäinen etusivun kuva (Cottonbro, 2020)

Valittu kuva sisälsi ihmisen liikkeessä, futuristiseen teemaan sopivia valoja ja oli vapaasti käytettävissä ja hyvälaatuinen. Asettelu ei ollut täydellinen, sillä henkilö oli hieman liian keskellä kuvaa. Mielenkiintoisin asettelu syntyy, kun kiintopisteet asetellaan kolmasosien paikkeille (Rebecca Hagen, 2017, s. 140). Toisaalta taustalla olevat kirkkaat valot olivat suotuisammissa paikoissa, mikä tasapainotti asetelmaa riittävästi.

Seuraavaksi kuva täytyi muokata väriteemaan sopivaksi. Käyttäen avoimen lähdekoodin kuvankäsittelyohjelma GIMP:iä kuvasta muutettiin harmaasävyinen. Kuvan päälle lisättiin toinen taso, joka täytettiin teemaan kuuluvilla väreillä. Lopuksi tasot yhdistettiin, kuva rajattiin ja sen koko säädettiin sopivaksi. Yhdelläkään näytöllä kuvan ei pitäisi olla leveämpi kuin 2000 pikseliä, joten kuvatiedostonkaan ei tarvitse olla sen suurempi. Tiedosto tallennettiin JPG-muodossa, joka soveltuu parhaiten web-julkaisuun pienen tiedostokokonsa vuoksi, kun kuvassa ei ole läpinäkyvyyttä (Rebecca Hagen, 2017, s. 141). Kuvan tarkkuus oli jo ennestään julkaisuun sopiva, 72 ppi (pixels per inch, pikseliä per tuuma) (Rebecca Hagen, 2017, s. 142).

Koska suuri käyttäjäkunta sovelluksella tulee olemaan mobiililaitteilla, kuvasta tehtiin myös pienemmälle näytölle sovitettu kuva. Molemmat kuvat ovat nähtävissä alempana (Kuvat 14 ja 15). Etusivun kuvaa etsiessä löytyi muitakin kiinnostavia kuvia, jotka päätettiin jättää infosivulle koristamaan tekstiä. Jotta kuvien lataus ei hidastaisi sivuston käyttöä, niihin lisättiin hitaan latauksen ominaisuus (englanniksi lazy-loading), joka varmistaa, että kuvat ladataan vasta muun sisällön latauduttua ja kuvien ollessa näkyvissä.

Kuva 14. Lopullinen etusivun kuva (Saastamoinen 2021, CC BY)

Kuva 15. Mobiililaitteille optimoitu etusivun kuva (Saastamoinen 2021, CC BY)

6.4 Elementtien sijoittelu

Ihmiset ovat tottuneet siihen, että sovelluksissa elementeillä on omat, tyypilliset paikkansa. Nämä on hyvä tiedostaa ja rikkoa ainoastaan hyvin painavasta syystä, sillä muuten käyttäjä hämmentyy helposti. Esimerkiksi päävalikon selkein paikka on vaakatasossa sivun ylälaidassa (Irmeli Sinkkonen, 2009, s. 216).

Toinen tapa miettiä sivun elementtejä on käyttötarinoiden pohjalta. Käyttötarinat kuvaavat sovelluksen kuvitteellista käyttäjää vuorovaikutuksessa sovelluksen kanssa (Irmeli Sinkkonen, 2009, s. 172). Tässä projektissa tilaajan toiveita kartoitettiin juuri tällaisten tarinoiden kautta. Kun oli selvillä, mitä käyttäjän pitäisi pystyä tekemään missäkin osassa sivustoa, mahdollisuudet muutettiin konkreettisiksi elementeiksi ja sijoiteltiin luonteviin paikkoihin sivulla.

Esimerkiksi hallintapaneelissa kirjautuneen käyttäjän piti pystyä tarkastelemaan, muokkaamaan ja poistamaan pelaajien ja hahmojen tietoja. Tähän on yleensä kaksi esitystapaa: joko niin, että halutut kohteet valitaan klikkaamalla valintaruutua, ja jossain taulukon yläpuolella on työkalupalkki, josta valitaan haluttu toiminto, tai sitten jokaisen kohteen kanssa samalla rivillä on nappeja, joilla kohdetta voi käsitellä. Jälkimmäinen tapa sopi tähän käyttökohteeseen paljon paremmin, sillä käyttäjällä ei pitäisi olla juurikaan tarvetta manipuloida useampaa kohdetta yhtä aikaa. Esimerkki tästä on nähtävissä alapuolella (Kuva 16).

Kuva 16. Esimerkki hallintapaneelin asettelusta (Saastamoinen 2021, CC BY)

6.5 Animaatiot

Animaatiot ovat kaksiteräinen miekka: toisaalta ne tuovat sovellukseen laadun ja moderniuden tunnetta, mutta toisaalta ne hidastavat sivun käyttöä etenkin mobiiliselaimilla. Tämän vuoksi niiden käyttöä tulee harkita tarkkaan. Tässä projektissa niitä käytettiin muutamissa kohdissa.

Sovelluskehityksessä käytetyn koodikirjaston Bootstrapin komponentteihin sisältyy valmiiksi joitakin animaatioita. Esimerkiksi modal (varmistusviesti), toast (ilmoitus) ja alert (varoitus) katoavat hidastetusti ja pienille näytöille kasaan taitettu navigaatiopalkki liukuu auki ja kiinni. Bootstrap tukee myös saavutettavuutta ottamalla huomioon, jos käyttäjän selainasetukset pyrkivät vähentämään liikettä verkkosivuilla. Tällöin kaikki animaatiot toistetaan hidastettuina.

Vaikka varsinaisia animaatioita eivät olekaan, linkkeihin ja nappeihin lisättiin erilaisia efektejä, jotka näkyvät hiiren ollessa elementin päällä, klikatessa sitä tai valitessa sen näppäimistöllä. Nämä auttavat tunnistamaan linkit linkeiksi ja luovat tunteen, että sovellus toimii ja vastaa käyttäjän toimiin. Hiiren vieminen linkkien päälle eli hover-toiminto tuo esiin alleviivauksen ja hehkuefektin, ja napeissa taustan väri muuttuu. Näppäimistöllä navigoidessa valitun napin ympärille tulee paksu viiva, mikä on vastaus focus-toimintoon. Alla olevassa kuvassa on nähtävissä tämä muutos (Kuva 17).

Kuva 17. Nappi normaalisti, hiiren alla (hover) ja valittuna (focus) (Saastamoinen 2021, CC BY)

7 SAAVUTETTAVUUS

Projektin alusta lähtien oli selvää, että sovelluksessa otetaan mahdollisuuksien mukaan huomioon erilaiset haasteet, joita pelaajilla saattaa olla. Liveroolipelaaminen on hyvin vastaanottavainen harrastus kaikenlaisille ihmisille, ja usein peliin voi osallistua monenlaisista fyysisistä haasteista huolimatta. Käytännössä tämä tarkoitti sovelluksen optimointia näytönlukijoille sekä suuremmalle fonttikoolle.

Google Chrome -selaimessa on kätevä toiminto nimeltä Lighthouse, joka auttaa tarkastelemaan sivustoa tehokkuuden ja saavutettavuuden näkökulmasta. Kun sivusto loppuvaiheessa kehitystä testattiin sen kautta, tulokset olivat jo melko hyviä. Lighthousen raportin mukaan yläpalkin kirjautumiskenttä vaati selitteen yhteyteensä, mutta koska visuaalinen ilmeeseen sellainen ei mahtunut eikä sille toisaalta ollut tarvettakaan, kun vieressä ollut nappi selitti kentän toiminnallisuuden, kentän viereen lisättiin ainoastaan näytönlukijoille näkyvä selite.

Hyväksytty tulos tuli esimerkiksi sivuston navigoimisesta näppäimistön avulla. Sivustolla eteneminen toimi loogisesti ja kaikkiin linkkeihin pääsi käsiksi. Lisäksi kaikissa linkeissä oli tyylimuutos, joka aktivoitui, kun selaimen focus eli huomio oli kyseisessä linkissä. Tämä helpotti navigoinnin seuraamista. Navigointia tuki myös elementtien semanttinen nimeäminen: siinä missä div-elementti ei merkitse mitään vaan toimii ainoastaan asettelun apuna, main-, header- ja section-elementit sen lisäksi kuvaavat sisältöä.

Varoitus- ja infoviesteihin sekä pikaviesti-ilmoituksiin lisättiin muutama määrite, jotka varmistavat niiden oikean esitystavan avustavalle teknologialle. Ensinnäkin ilmoitukset ovat näytönlukijoilta piilossa silloin, kun niitä ei ole tarkoitus näyttää. Toisekseen niille on asetettu rooliksi ”hälytys”, jotta näytönlukijat lukevat ne tarkoituksenmukaisella tavalla, esimerkiksi äänimerkin kera. Nämä määritteet noudettavat ARIA-standardia (Web Accessibility Initiative's Accessible Rich Internet Applications) (Google.com, 2020). Alla olevasta kuvasta ne näkyvät käytännössä (Kuva 18).

```
<div
class="alert alert-danger position-fixed bottom-0 start-50 translate-middle-x fade hidden"
id="errorMessage"
role="alert"
aria-live="assertive"
aria-atomic="true"
>
```

Kuva 18. ARIA-standardin mukaiset määritteet varoitusviestissä (Saastamoinen 2021, CC BY)

Myös värivalinnoissa on otettu näkemisen haasteet huomioon. Tekstin ja taustan välillä on tarpeeksi kontrastia, jotta lukeminen on helppoa. Tämän lisäksi värisokeuden asettamia haasteita on testattu Firefox-selaimen Let’s get colorblind -laajennuksella, jolla voi simuloida erilaisia värisokeuden tyyppejä. Sivusto näytti yhä selkeältä ja eri värit olivat erotettavissa toisistaan.

8 POHDINTA

Tämän opinnäytetyön painopiste oli visuaalisessa lopputuloksessa ja käytettävyydessä. Ne ovat erottamattomasti osa toisiaan, ja niiden yhdistelmää kutsutaan visuaaliseksi käytettävyydeksi (Irmeli Sinkkonen, 2009, s. 242). Tasapaino niiden välillä säilyi koko projektin ajan: käytettävyyttä ei uhrattu hienojen tyylikikkojen vuoksi.

Visuaalinen suunnittelu on usein erotettu sovelluksien muusta kehittämisestä, ja ostetaan ulkoiselta yritykseltä (Irmeli Sinkkonen, 2009, s. 243). Vaikka koen omaavani visuaalista silmää ja jonkin verran kokemustakin, tämän projektin jälkeen kyseinen seikka on minulle paljon ymmärrettävämpi. Visuaalinen suunnittelu hyötyy koulutuksesta siinä missä muutkin sovelluskehityksen osa-alueet, ja koska oma koulutukseni on keskittynyt pelkästään tekniseen puoleen, ulkoasua suunnitellessa tuli usein kokematon olo. Tiesin, milloin jokin osa näytti ulkoasussa hyvältä, mutta en aina tiennyt, mitkä asiat tekivät siitä sellaisen. Tämä projekti on opettanut minulle paljon.

Vaikka usein aikataulussa on hankala pysyä, tämä projekti onnistui siinä yllättävän hyvin. Sovellus toki vaatii vielä hiomista ja erityisesti testaamista, mutta tämän opinnäytetyön aihealue tuli käsiteltyä aikataulussa. Suurimpia tekijöitä onnistumisen taustalla olivat tarkka aikataulutus, nopea rutiinin löytyminen projektin tekoon ja sopivankokoisen haasteen määrittely. Projektista olisi helposti tullut liian suuri ja vaikea, jos toiminnallisuuksia olisi ollut yhtään enempää tai tekniikat olisivat olleet vähemmän tuttuja.

Erityisesti pidin tässä projektissa itsenäisestä työskentelystä. Sain täysin vapaat kädet toteutustapojen valinnassa ja soveltaa kaikkea oppimaani teknisistä raporteista kokousten sopimiseen ja työn etenemisen valvontaan. Oloni on nyt itsevarma tulevaisuuden projekteja ajatellen, sillä tiedän, miten prosessi etenee ja minulla on näyttää konkreettinen esimerkki kyvyistäni.

Kommunikointi työn tilaajan kanssa oli helppoa ja rentoa, sillä hän oli minulle entuudestaan tuttu henkilö. Siinä mielessä tämä opinnäytetyö olisi voinut olla vielä opettavaisempi, jos tilaaja olisi ollut täysin tuntematon henkilö tai yritys. Uskon kuitenkin kommunikaatiotaitojeni olevan riittävällä tasolla, sillä aiemmissa harjoitteluissa työskentely uusien ihmisten kanssa on tullut tutuksi.

9 JATKOKEHITYS

Kaikkia tilaajan toiveita ei pystytty toteuttamaan tämän opinnäytetyön aikataulussa. Yksi näistä oli minipeli, jonka voittamalla pelaaja saisi ”hakkeroitua” tietoa toisesta pelaajasta tai rahaa tämän tililtä. Tämän voi kuitenkin toteuttaa melko helposti, kunhan sille on määritetty tarkat raamit, joiden sisällä se toimii.

Toiveena oli myös ottaa vielä yksi askel mobiililaitteille optimoinnissa: sivuston muuttaminen PWA-sovellukseksi. PWA eli Progressive Web Application on välimuoto varsinaisen mobiilisovelluksen ja websovelluksen välissä, joka tukee muun muassa verkotonta tilaa, ilmoituksia ja asennusta (LePage & Richard, 2020). PWA:n hyödyt olisivat merkittävät pikaviestipalvelun käytettävyydessä, mutta aika ei riittänyt tähän ominaisuuteen.

Sovellus vaatii vielä hiomista ja testaamista, jotta se varmasti tulee toimimaan oikeassa tilanteessa. Tähän mennessä sitä on testannut lähinnä kehittäjä ja tilaaja, mutta on tärkeää löytää myöhemmin useampi testaaja, jotka eivät vielä tunne sovellusta. Erityisesti mobiililaitteilla testaaminen jäi vähäiseksi projektin aikana, kun paikallisesti toimivaan palvelimeen oli hankalaa saada yhteyttä toiselta laitteelta.

Yksi tärkeä mekaniikka hahmojen muokkauksesta jäi toteuttamatta. Kun sovelluksella tulee olemaan useampi käyttäjä, jotka voivat muokata hahmojen tietoja, olisi tärkeää estää tietojen ylikirjoitus, jos useampi ihminen muokkaa niitä yhtä aikaa. Tämän voisi toteuttaa joko niin, että tietoihin pääsee käsiksi vain yksi käyttäjä kerrallaan, tai varoittamalla useista muokkaajista hahmotietosivulla. Myös muokkaushistoria toisi turvaa tietojen menettämiseen, mutta toisaalta se alkaa mennä tämän sovelluksen skaalan ylitse.

Jos katsoo vielä kauemmas tulevaisuuteen, tätä sovellusta voisi muokata sopivaksi liveroolipeleihin yleisesti ottaen. Se toki edellyttää, että sovellus todetaan toimivaksi Totentanz-pelissä ja käyttäjät kokevat, että haluaisivat nähdä sen muissakin peleissä. Ulkoista tyyliä on helppoa muuttaa kunkin pelin teemaan sopivaksi.

10 LÄHDELUETTELO

Andras, S. (14. 2. 2017). *HelloJS*. Haettu 28. 3. 2021 osoitteesta https://blog.hellojs.org/javascript-frameworks-why-and-when-to-use-them-43af33d0608d

Bootstrap Team. (ei pvm). *Bootstrap*. Haettu 9. 4 2021 osoitteesta https://getbootstrap.com/

CD Projekt Red. (10. 12. 2020). Cyberpunk 2077.

Cottonbro. (2020). Noudettu osoitteesta https://www.pexels.com/fi-fi/kuva/auto-ajoneuvo-neon-korjata-4488636/

ellioi. (14. 7. 2020). *IN-Feels peliohjesovellus*. Haettu 24. 4. 2021 osoitteesta https://inf.purplie.net/

Facebook Inc. (2021). *React - a Javascript library for building user interfaces*. Haettu 9. 4. 2021 osoitteesta https://reactjs.org/

Google.com. (9. 7. 2020). *Web Fundamentals*. Haettu 23. 4. 2021 osoitteesta https://developers.google.com/web/fundamentals/accessibility/semantics-aria/

Irmeli Sinkkonen, E. N. (2009). *Helppokäyttöisen verkkopalvelun suunnittelu.* Helsinki: Tietosanoma Oy.

LePage, P.;& Richard, S. (24. 2. 2020). *Web.dev*. Haettu 23. 4. 2021 osoitteesta https://web.dev/what-are-pwas/

MongoDB Inc. (2021). *MongoDB*. Haettu 9. 4. 2021 osoitteesta https://www.mongodb.com/mern-stack

Neondystopia.com. (2018). *Neon Dystopia*. Haettu 22. 4. 2021 osoitteesta https://www.neondystopia.com/what-is-cyberpunk/

Rebecca Hagen, K. G. (2017). *White space is not your enemy: a beginner's guide to communicating visually through graphic, web & multimedia design* (3rd p.). Boca Franton, FL: CRC Press, Taylor & Francis Group.

Refnes Data. (2021). *W3Schools*. Haettu 9. 4. 2021 osoitteesta https://www.w3schools.com/

Slate. (29. 3. 2021). *Slate*. Haettu 9. 4. 2021 osoitteesta https://docs.slatejs.org/

StrongLoop, IBM, and other expressjs.com contributors. (2017). *Express*. Haettu 23. 4. 2021 osoitteesta http://expressjs.com/

11 KUVALUETTELO