TAMPEREEN
AMMATTIKORKEAKOULU

Kosketusnäyttöpohjaisen kassakoneen käyttöliittymän suunnittelu ja toteutus

Aleksanteri Karanka

Opinnäytetyö
Syyskuu 2016
Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

KARANKA, ALEKSANTERI:
Kosketusnäyttöpohjaisen kassakoneen käyttöliittymän suunnittelu ja toteutus

Opinnäytetyö 39 sivua
Syyskuu 2016

Tämän opinnäytetyön tavoite oli tutkia käyttöliittymäsuunnittelun teoriaa ja käyttöliittymän teknistä toteutusta. Teorian pohjalta oli tarkoitus suunnitella ja toteuttaa uusi web-pohjainen kassakoneen käyttöliittymäsovellus. Valmistuessaan sovellus korvaisi työn toimeksiantajan ÖÖ-Evolving Officen nykyisen käyttöliittymäratkaisun. Nykyinen Java-pohjainen käyttöliittymä toimi uuden käyttöliittymän suunnittelun pohjana.

Itse sovellus ei valmistunut opinnäytetyöprosessin aikataulussa, ja sen tulevaisuus on auki. Mikäli sen kehitystä jatketaan, otetaan se valmistuessaan käyttöön toimeksiantajan asiakasyrityksissä. Työssä keskitytään niihin käyttöliittymän näkymiin ja toiminnallisuuksiin, jotka tulivat valmiiksi. Niitä vertaillaan alan kirjoista ja artikkeleista kerätyn teorian näkökulmasta alkuperäisen käyttöliittymän vastaaviin osiin. Lisäksi työssä avataan sovelluksen teknisen toteutuksen ratkaisuja.

Työn aikana selvisi, että vaikka teoria on hyvä lähtökohta käyttöliittymän suunnittelulle, vaatisi paras mahdollinen käyttöliittymä kuitenkin enemmän ja yksityiskohtaisempaa taustatietoa käyttäjäkunnasta ja käyttöolosuhteista. Lisäksi sopivien työkalujen valinnan merkitys teknisessä toteutuksessa tuli vastaan; käyttöliittymän monimutkaisuus ja laajuus on otettava huomioon web-käyttöliittymien toteutustyökaluja valittaessa.

Asiasanat: käyttöliittymä, kassajärjestelmä, web-sovellus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Development

KARANKA, ALEKSANTERI:
Design and Implementation of a Touchscreen User Interface for a Point of Sale System

Bachelor's thesis 39 pages
September 2016

The purpose of this thesis was to study the theory of user interface design and implementation of user interfaces. The goal was to design and implement a new web-based user interface application for a point of sale system. The application would eventually replace the current user interface solution used by ÖÖ-Evolving Office. The current Java-based user interface application was used as a basis for the new design.

The new application was not finished within schedule and its future is still uncertain. In case its development will be continued and eventually finished, it will be deployed at the client companies that are currently using the old Java-based software. This thesis is focused on the finished views and components of the new interface. The finished areas are compared to the equivalent areas in the original interface from the perspective of user interface design theory. Furthermore, the technical choices in the implementation of the application are reviewed.

During study it became apparent that more detailed data of users and user environments is required for best possible user interface design – theory alone provides only a good starting point. In addition, the significance of choosing the right tools for implementation became clear. An important facet to consider when choosing the tools is the complexity and scope of the target application.

Key words: user interface, point of sale software, web application

SISÄLLYS

1 JOHDANTO

Hyvän käyttöliittymäsuunnittelun hyödyt ovat laajasti dokumentoidut. Erään tutkimuksen mukaan yhden ainoan näkymän uudelleensuunnittelu toi yritykselle 20 000 dollarin säästöt vuodessa (Galitz 2007, 6). Paremman, nopeampikäyttöisen näkymän synnyttämä säästö kertyy pitkällä aikavälillä yksittäisten sekuntien kasautuessa työtunneiksi ja -päiviksi. Toisaalta huonosti suunniteltu käyttöliittymä turhauttaa käyttäjänsä – laitteisiinsa ärsyyntyneeltä työntekijältä harvoin saa parasta mahdollista asiakaspalvelua. Hyvin suunniteltu käyttöliittymä mahdollistaa helpomman ja nopeamman työskentelyn, joka hyödyttää kaikkia.

Tämän opinnäytetyön tavoitteena on tutkia käyttöliittymäsuunnittelun teoriaa, jonka pohjalta suunnitellaan ja toteutetaan kosketusnäytöllä toimiva HTML5-pohjainen käyttöliittymäsovellus. Sovelluksen, työnimeltään KassaUI (Kassa User Interface), on tarkoitus korvata toimeksiantajan nykyinen Java-pohjainen myyntikäyttöliittymä. Käyttöliittymän suunnittelun pohjana käytetään nykyistä käyttöliittymää, mutta sen käytettävyyttä kehitetään ja siihen lisätään uusia ominaisuuksia.

Työn toimeksiantaja ÖÖ-Evolving Office on Tampereella toimiva hallintajärjestelmiin, julkaisuratkaisuihin ja mobiili-integraatioon keskittyvä ohjelmistotalo. Virtuaalikonttori on ÖÖ-Evolving Officen toiminnanohjausjärjestelmä, joka on käytössä muun muassa ravintoloissa, pubeissa ja kahviloissa. Näissä ympäristöissä Virtuaalikonttorin myyntikäyttöliittymänä toimii Java-pohjaisesta Openbravo POS -kassajärjestelmästä muokattu käyttöliittymä.

Kyseisen kassajärjestelmän – ja käyttöliittymän – juuret ovat ajassa ennen tablettien ja kosketusnäyttöpuhelimien yleistymistä. Käytössä olevan Openbravo POS:in versio on julkaistu vuonna 2010 – samana vuonna Apple julkaisi tablettien vallankumouksen käynnistäneen iPadin. Kosketusnäyttökäyttöliittymissä sittemmin tapahtunut kehitys näkyy selvästi. Openbravo POS:issa on esimerkiksi käytössä kankeat vierityspalkit kun nykyään kosketusnäytöillä vierittäminen tapahtuu yleisesti raahaamalla.

Opinnäytetyö jakautuu kahteen osaan. Ensimmäisessä osassa käydään läpi käyttöliittymäsuunnittelun teoriaa ja sitä miten sitä käytännössä on hyödynnetty KassaUI:n toteu-

tuksessa. Toisessa osassa perehdytään KassaUI:n tekniseen toteutukseen. Teknistä toteutusta ja käyttöliittymäsuunnittelua on tehty samanaikaisesti yhtenä hiljalleen muovautuvana kokonaisuutena. Käytännössä tämä tarkoittaa sitä, että käyttöliittymään ei ole edes lähdetty suunnittelemaan mitään sellaista, jonka toteuttaminen olisi huomattavan hankalaa. Toisaalta käyttöliittymä on myös hyötynyt kun koodia testatessa on saanut uusia entistä parempia ideoita.

2 KASSAJÄRJESTELMÄT JA KASSAUI

Kassakoneet ovat vuosien varrella kehittyneet yksinkertaisista mekaanisista laitteista tekemään kaikenlaisia muitakin toimintoja kuin laskemaan tuotteiden hintojen summan ja tulostamaan kuitin. Nykyään puhutaan järjestelmistä, joihin voi normaalien myyntitoimintojen lisäksi yhdistyä varastonhallintaa, kirjanpitoa, asiakkuudenhallintaa yms. Tällaista ohjelmistoa kutsutaan kassajärjestelmäksi (point of sale software, POS).

Kassajärjestelmiä räätälöidään kunkin toimialan tarpeisiin sopiviksi. Esimerkiksi ravintolassa tilauksen vastaanottaja ottaa tiedot talteen kassajärjestelmään. Järjestelmään kirjataan tilatut tuotteet, mahdolliset erikoistoiveet keittiöön, pöytä jossa asiakas istuu ja muut tarvittavat tiedot. Järjestelmän kautta kassan käyttäjä tulostaa valmistettavat tuotteet tiedoksi keittiöön. Asiakkaan ollessa valmis kassan käyttäjä tulostaa kuitit ja arkistoi ostotapahtuman.

Openbravo POS on edellä kuvatun kaltainen Java-pohjainen kassajärjestelmä. Työn toimeksiantaja on muokannut siitä oman version Virtuaalikonttorin myyntikäyttöliittymäksi ravintolaympäristöihin. Koska Openbravo POS on kokonaisvaltainen kassajärjestelmä, se sisältää vakiona paljon toimintoja, joita ei tarvita itse myyntitilanteessa; tuotteiden hallinnointia ei tarvita silloin, kun asiakas odottaa vaihtorahaa. Koska sovellusta käytetään vain myynnin käyttöliittymänä, jäävät monet siihen sisäänrakennetut ominaisuudet turhiksi. Esimerkiksi jo mainittu tuotteiden hallinnointi tapahtuu Virtuaalikonttorin omassa käyttöliittymässä, eikä Openbravo POS -käyttöliittymässä.

Toteutettavan KassaUI-sovelluksen ei ole tarkoitus olla parempi versio Openbravo POS -järjestelmästä, vaan parempi versio Openbravo POS:ista tiettyyn käyttötarkoitukseen muokatusta myyntikäyttöliittymästä. Sovellusten erilaiset lähtökohdat on hyvä pitää mielessä, kun seuraavissa luvuissa verrataan KassaUI:ta alkuperäiseen käyttöliittymään.

3 KÄYTTÖLIITTYMÄSUUNNITTELU

Käyttöliittymä on laitteen tai ohjelmiston osa, jonka välityksellä käyttäjä käyttää laitetta. Käyttöliittymäsuunnittelun tavoitteena on tehdä liittymästä käyttäjän ja laitteen välillä mahdollisimman tehokas ja miellyttävä. Hyvän käyttöliittymän tarkoitus on antaa työkalut vaaditun tehtävän suorittamiseen ja muilta osin pysyä poissa tieltä. Onnistunut käyttöliittymä jää käyttäjältä kokonaan huomaamatta – huomio kohdistuu itse tekemiseen. (Galitz 2007, 4.)

3.1 Käytettävyys

Käyttöliittymän laatua käyttäjän näkökulmasta arvioitaessa puhutaan *käytettävyydestä*. Nielsen (2012) määrittelee käytettävyyden seuraavien viiden osatekijän kokonaisuudeksi:

1. Opittavuus: Kuinka nopeasti uusi käyttäjä oppii käyttämään käyttöliittymän perustoimintoja?
2. Tehokkuus: Kuinka nopeasti kokenut käyttäjä suorittaa vaaditut työtehtävät?
3. Muistettavuus: Kuinka helppoa käyttöliittymään palaaminen on pitkän tauon jälkeen?
4. Virheet: Kuinka paljon ja kuinka vakavia virheitä käyttäjälle sattuu? Kuinka helppoa niiden korjaaminen on?
5. Miellyttävyys: Kuinka miellyttävää käyttöliittymän käyttäminen on?

Käyttöliittymän käytettävyyttä voidaan arvioida erilaisten käytettävyystestien avulla. Tehokkain tapa on testata käyttöliittymää oikeilla käyttäjillä (Nielsen 2012). Voidaan esimerkiksi järjestää testitilaisuus, jossa tarkkaillaan testihenkilöiden käyttöliittymän käyttöä. Tarkkailun avulla saadaan selville kuinka eri käyttäjät käyttävät käyttöliittymää ja mitä ongelmia siinä esiintyy. Esimerkki toisesta käytettävyyden mittaustavasta ovat heuristiset arvioinnit, joissa käyttöliittymäasiantuntija käy käyttöliittymän läpi jonkin tarkistuslistan kanssa. Listoja, jotka sisältävät yleisiä käytettävyyttä parantavia ohjeita on tehty eri tarkoituksiin ja eri näkökulmista, mutta tunnetuin niistä lienee Nielsenin kymmenen heuristiikan lista. Kyseiseltä listalta löytyy esimerkiksi ohje ”käyttöliittymän on oltava esteettinen ja minimalistinen” (Nielsen 1995).

Käyttöliittymäsuunnittelussa käytettävyys on keskeistä, mutta se itsessään ei vielä tuota hyödyllistä käyttöliittymää. Käytettävyysdatan tarkoituksena onkin vain toimia varsinaisen käyttöliittymäsuunnittelun pohjana (Rubin & Chisnell 2008, 22). Galitz (2007, 131) tiivistää tavoitteet helpon ja miellyttävän käyttöliittymän suunnitteluun seuraavasti:

- Vähennä käyttäjän visuaalista työtä.
- Vähennä käyttäjän älyllistä työtä.
- Vähennä käyttäjän muistamistyötä.
- Vähennä käyttäjän fyysistä työtä.
- Minimoi kaikki ylimääräinen vaiva ja ohjeistus, joka aiheutuu teknologiasta.

Visuaalisen työn vähentämisellä tarkoitetaan sitä, että käyttöliittymän näkymät ovat visuaalisesti mahdollisimman selkeitä ja yksinkertaisia – informaatio on esitetty käyttäjälle tehokkaasti. Älyllisellä työllä tarkoitetaan ajattelua. Käyttäjän ajattelun pitäisi kohdistua itse suoritettavaan tehtävään, ei käyttöliittymään. Muistamistyö on hieman vastaava: esimerkiksi kaikki kerralla tarvittava tieto pitäisi olla yhtä aikaa saatavilla eikä siten, ettei käyttäjä joudu erikseen painamaan asioita muistiin tehdessään työtehtävää. Toisaalta käyttöliittymän pitäisi olla tarpeeksi intuitiivinen, ettei käyttäjän tarvitse yrittää muistella miten joku vähemmän käytetty toiminto tehdään. Fyysinen työ tarkoittaa muun muassa käyttäjän painalluksia ja muita liikkeitä.

Listassa toistuva sana *työ* sopii oivallisesti tämän opinnäytetyön asiayhteyteen, sillä kassan käyttäjäkunta koostuu kokonaisuudessaan *työn*tekijöistä, joiden *työ*taakkaa ollaan helpottamassa. Se kuvastaa hyvin näkökulmaa, josta suunnittelua on tässä opinnäytetyössä lähestytty. Lista kattaa kuitenkin hyvin kaikenlaisten käyttöliittymien tavoitteet.

3.2 Käyttöliittymän suunnittelu kosketusnäytölle

Ohjelmiston käyttöliittymän suunnittelussa on tärkeätä huomioida käyttöliittymän syötetapa. Erilaisia ohjelmistojen syötetapoja ovat esimerkiksi hiiri, näppäimistö, kosketusnäyttö, erilaiset liike ja asentoanturit, peliohjaimet, kaukosäätimet ja niin edespäin. Jokaisella syötetavalla on omat hyvät ja huonot puolensa sekä ominaispiirteet, jotka täytyy ottaa huomioon käyttöliittymän suunnittelussa.

Yksi suurimpia kosketusnäytön etuja on suorakäyttöisyys. Suorakäyttöisyys tarkoittaa sitä, että käyttäjä voi esimerkiksi fyysisesti sormea raahaamalla vierittää ruudulla olevaa näkymää. Kun näytöllä oleva näkymä liikkuu käyttäjän sormen mukana, on käyttäjän ja laitteen välinen toiminta mahdollisimman suoraviivaista. Suorakäyttöisyydestä vastaesimerkki on hiirikäyttöliittymä: käyttäjä liikuttaa kättään, joka liikuttaa hiirtä, joka liikuttaa näytöllä näkyvää kursoria, joka liikuttaa vierityspalkkia, joka vierittää näkymää. Kun käyttö on suoraviivaista, on se myös intuitiivista ja helppoa. Muihin kosketuskäyttöliittymän etuihin kuuluu lisäsyötelaitteiden tarpeettomuus. Kun ylimääräistä hiirtä tai muuta syötelaitetta ei ole, se ei voi myöskään mennä rikki, hukkua tai sammua virran loppuessa.

Käyttöliittymänsuunnittelussa pitää kuitenkin erityisesti huomioida syötetavan rajoitukset. Kosketusnäyttökäyttöliittymissä suurimmat rajoitukset aiheutuvat ihmisen kädestä ja sormista. Sormi on huomattavasti epätarkempi kuin hiiren kursori, minkä takia käyttöliittymän painikkeiden täytyy olla suurempia. Näyttöä painaessa käsi ja sormi myös peittää osan ruudun kuva-alasta, mikä on myös otettava huomioon. Hiireen tai näppäimistöön verrattuna käyttö on varsinkin suuremmilla näyttöko'oilla myös raskasta: kättä joutuu liikuttelemaan pidempiä matkoja, eikä kättä voi nojata pöytään tai rannetukeen. Myös kosketusaistiin perustuvan palautteen puute pitää huomioida: fyysisen painikkeen sijainti ja painallussyvyys ovat havaittavissa pelkän sormenpäiden tuntoaistin avulla, mikä ei onnistu kosketusnäytöillä.

4 KASSAUI:N KÄYTTÖLIITTYMÄSUUNNITTELU

Tavoite oli toteuttaa uusi käyttöliittymä vanhan käyttöliittymän pohjalta sekä lisätä siihen joitain toimeksiantajan määrittelemiä ominaisuuksia – esimerkiksi mahdollisuus käyttöliittymän parempaan muokattavuuteen käyttäjän tarpeisiin. Uudet ominaisuudet perustuivat toimeksiantajan käyttäjiltä saamaan palautteeseen. Työ alkoi siitä, että opettelin ensin itse käyttämään alkuperäistä käyttöliittymää tehden samalla muistiinpanoja käytettävyyden näkökulmasta – varsinaista käytettävyystestausta oikeiden käyttäjien kanssa ei kuitenkaan tehty.

Alkuperäisen, todella laajan käyttöliittymän käytettävyys kärsii omasta monipuolisuudestaan; jokainen käyttöliittymä joutuu tekemään kompromissin käytettävyyden ja joustavuuden välillä (Lidwell, Elam, Butler, & Holden 2010, 102). Alkuperäisen käyttöliittymän suunnittelussa painopiste on selvästi ollut mahdollisimman monen eri käyttöympäristön ja -tarpeen kattaminen. Se soveltuisi yhtä hyvin rautakaupan inventaarion ylläpitämiseen kuin ravintolan kassakäyttöön. Lisäksi alkuperäistä käyttöliittymää on toimeksiantajan toimesta muokattu vastaamaan paremmin vaadittuja tarpeita. Käytännössä tämä on tarkoittanut uusien ominaisuuksien lisäämistä – ja kun käyttöliittymää on näin venytetty vielä siitä, mihin se on alun perin suunniteltu, on jouduttu tekemään kompromisseja.

Alkuperäisestä käyttöliittymästä löytyikin todella paljon parannettavaa jo pelkästään omien testailujeni ja käyttöliittymäsuunnittelun perusperiaatteiden pohjalta. Paikannettuani ongelmat, alkoi varsinainen suunnitteluprosessi. Suunnittelun alkuvaiheessa pohdin erilaisia ratkaisuja paperille piirrettyjen käyttöliittymäluonnosten avulla. Saatuani kunkin käyttöliittymän osan toimimaan paperilla, siirryin sen toteuttamiseen käytännössä.

Uudet ratkaisut, esimerkiksi käyttöliittymäelementin uudelleenasemointi tai ohjetekstin sanamuodon vaihto, on perusteltu pääosin teoriasta löytyvillä periaatteilla. Tällaiset käyttöliittymäsuunnittelun perusperiaatteet eivät kuitenkaan yksinään ratkaise ongelmia. Kaikki tehdyt ratkaisut ovat pohjimmiltaan kompromisseja useiden toistensa kanssa ristiriidassa olevien hyötyjen kanssa. Ääriesimerkki: Jos halutaan tehdä jostain painikkeesta mahdollisimman helposti painettava, voi sen muuttaa koko ruudun kokoiseksi peit-

täen alleen kaikki muut elementit – silloin tosin muiden painikkeiden painaminen saattaa osoittautua haasteelliseksi.

4.1 Käyttöliittymän toiminta

Käyttöliittymän päätarkoitukset ovat tilausten kirjaaminen sekä kuittien ja keittiötilausten tulostaminen. Selvennetään käyttöliittymän perustoimintoja yksinkertaisen esimerkkikäyttötapauksen avulla:

1. **Alkutilanne**
   Käyttäjä (tarjoilija) on juuri ottanut yhden pöydän tilauksen vastaan ja tulee kassalaitteen luokse. Tilauksen tiedot (tuotteet, lisätoivcet, yms.) on merkattu muistilapulle.
2. **Tilauksen kirjaus**
   Käyttäjä kirjaa kaikki tilauksen tiedot (pöytä, asiakkaiden määrä, tilatut tuotteet asiakkaittain ja muut lisätiedot) lapulta koneelle. Käyttöliittymässä on oletuksena auki ravintolanäkymä, josta valitaan pöytä. Aukeaa tilausnäkymä, jossa tilauksen tiedot kirjataan.
3. **Tilaus keittiöön**
   Kun kaikki tiedot on kirjattu, tulostetaan tilaus keittiöön tilausnäkymän painikkeesta.
4. **Maksu**
   Myöhemmin, asiakkaiden syötyä, siirrytään maksunäkymään, jossa maksut kirjataan. Tiedot siirtyvät arkistoon ja pöytä tilauksineen vapautetaan seuraavaa käyttökertaa varten.

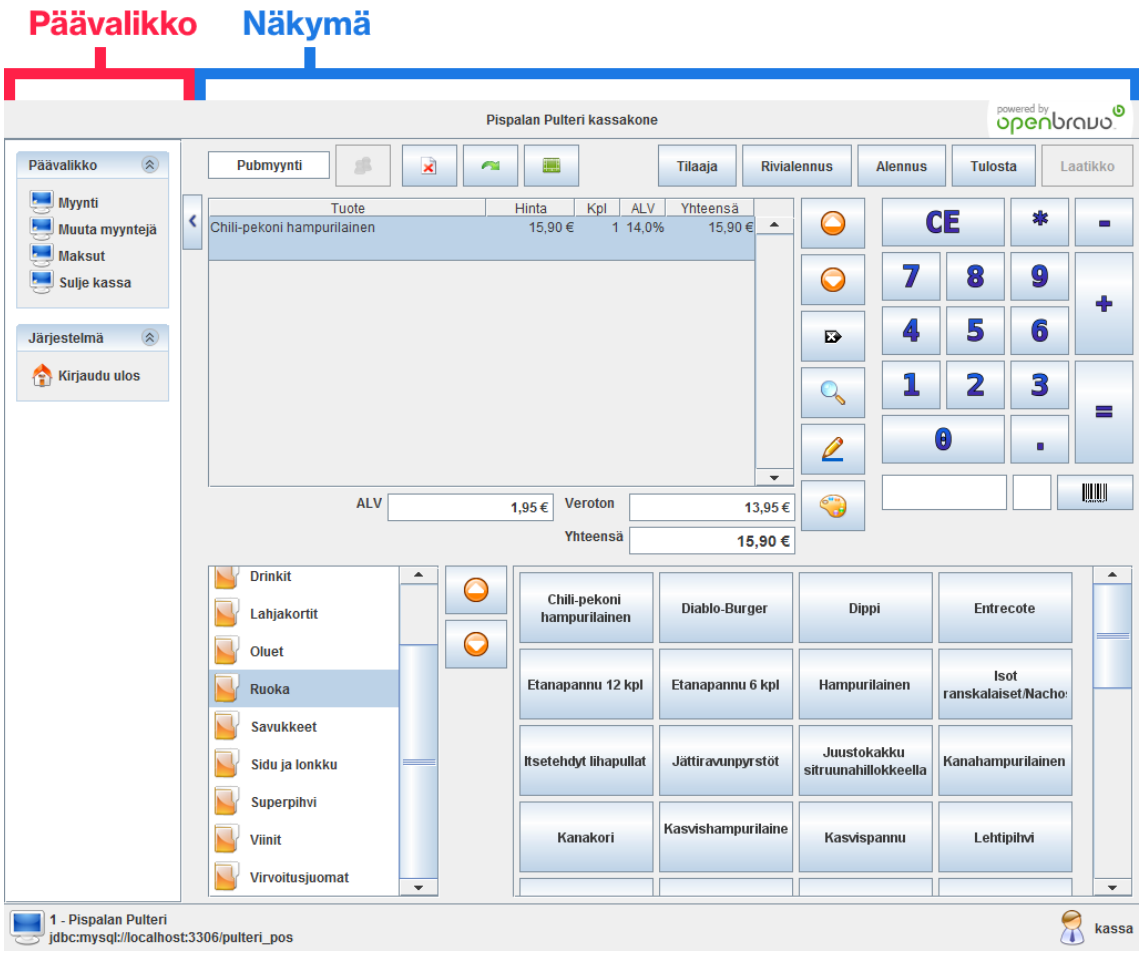
Esimerkissä kaikki tapahtui suoraviivaisesti, mutta myös paljon monimutkaisemmat käyttötapaukset ovat mahdollisia. Ravintolassa on useampia pöytiä, joissa kaikissa voi olla asiakkaita, joiden prosessit kulkevat eri tahtiin. Myös yhden pöydän tilanne voi muuttua kesken, jolloin prosessi voi hyppiä edestakaisin. Jos esimerkiksi jälkiruokatilauksia tehdään ensimmäisen tilauskierroksen jälkeen tai yksi pöydän asiakkaista haluaa maksaa ja lähteä ennen muita.

Seuraavissa alaluvuissa käydään läpi KassaUI:n eri näkymiä ja osia. Kunkin kappaleen alussa on kuvailtu osa ja sen toiminta alkuperäisessä käyttöliittymässä. Sen jälkeen kiinnitetään huomio löydettyihin kehityskohteisiin ja kerrotaan, miten kyseinen asia on huo-

mioitu KassaUI:ssa. Kaikkia näkymiä ja parannuksia ei käydä yksityiskohtaisesti läpi, vaan keskitytään muutamaan, jotka parhaiten havainnollistavat erilaisia käyttöliittymäsuunnittelun teorian periaatteita.

4.2 Päävalikko

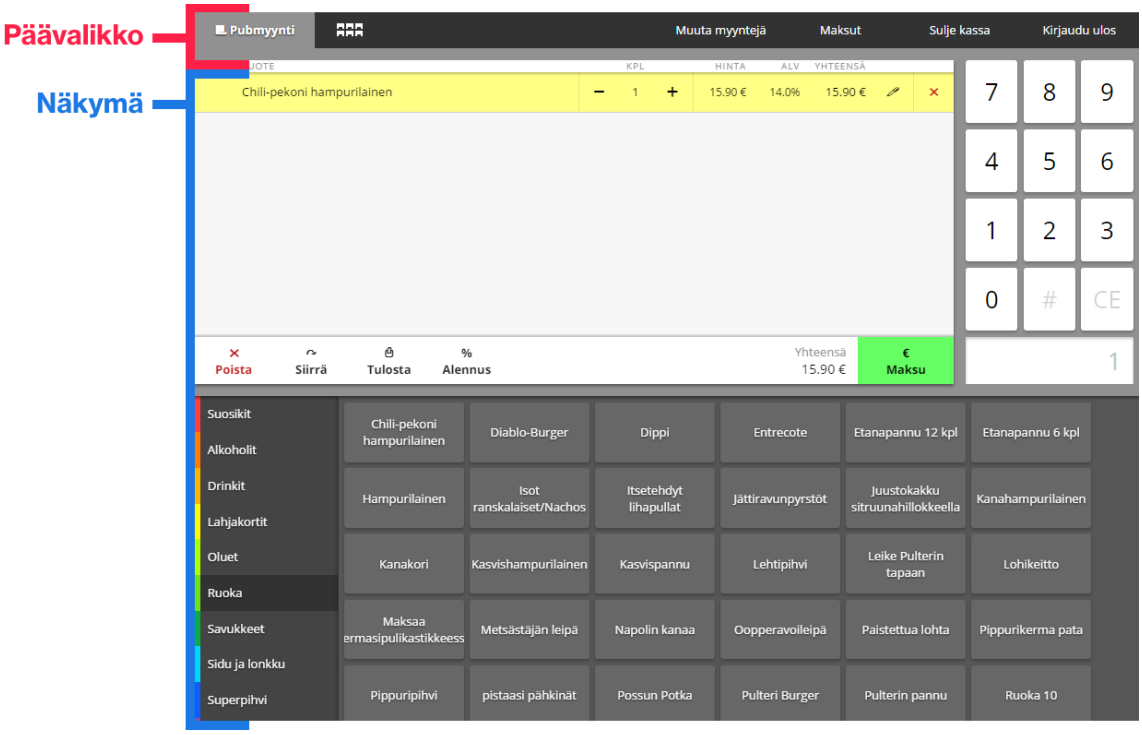
Alkuperäinen kassakäyttöliittymä jakautuu viiteen päänäkymään: myynti, muuta myyntejä, maksut, sulje kassa ja kirjautuminen. Päänäkymiksi luokittelen näkymät, joiden välillä navigoidaan ruudun vasemmassa reunassa olevasta päävalikosta (kuvio 1). Päävalikon voi piilottaa sen oikealla puolella olevasta nuolipainikkeesta vapauttaen lisää tilaa itse näkymälle. Piilotetusta valikosta on näkyvissä vain nuolipainike, josta se myös aukeaa takaisin näkyviin.



KUVIO 1. Päävalikko alkuperäisessä käyttöliittymässä

Päävalikosta itsestään ei näy mikä näkymä kulloinkin on valittuna, vaan sen joutuu päättelemään itse näkymästä. ”Maksut”- ja ”Sulje kassa”-näkymissä on itsessään otsikot, mutta muut näkymät joutuu päättelemään näkymän ulkoasusta.

Painikkeiden tekstit vievät eniten tilaa vaakasuunnassa, joten KassaUI:ssa koko valikko muutettu vaakasuuntaiseksi tilankäytön tehostamiseksi (kuvio 2). Valikko viekin vähemmän tilaa, vaikka painikkeiden kokoa on kasvatettu alkuperäisestä. Alkuperäisen valikon painikkeet ovat testilaitteella n. 5 mm korkeita, kun esimerkiksi Microsoft (Guidelines for targeting) suosittelee kosketusnäyttökäyttöliittymissä painikkeiden kooksi 9 mm tai enemmän suuntaansa.



KUVIO 2. Päävalikko KassaUI:ssa

Miten uusi vaakasuuntainen päävalikko vaikuttaa kaikin tavoin paremmalta kuin ennen? Jälleen esimerkki siitä, kuinka käyttöliittymät joutuvat usein tekemään kompromissiin joustavuuden ja käytettävyyden välillä. Koska alkuperäinen päävalikko on suunniteltu jokaisen mahdollisen käyttötarkoituksen huomioivaan geneeriseen kassajärjestelmään, on sen pystyttävä mukautumaan huomattavasti suurempaan määrään mahdollisia valintoja. Pystysuuntainen valikko onkin siitä näkökulmasta huomattavasti parempi.

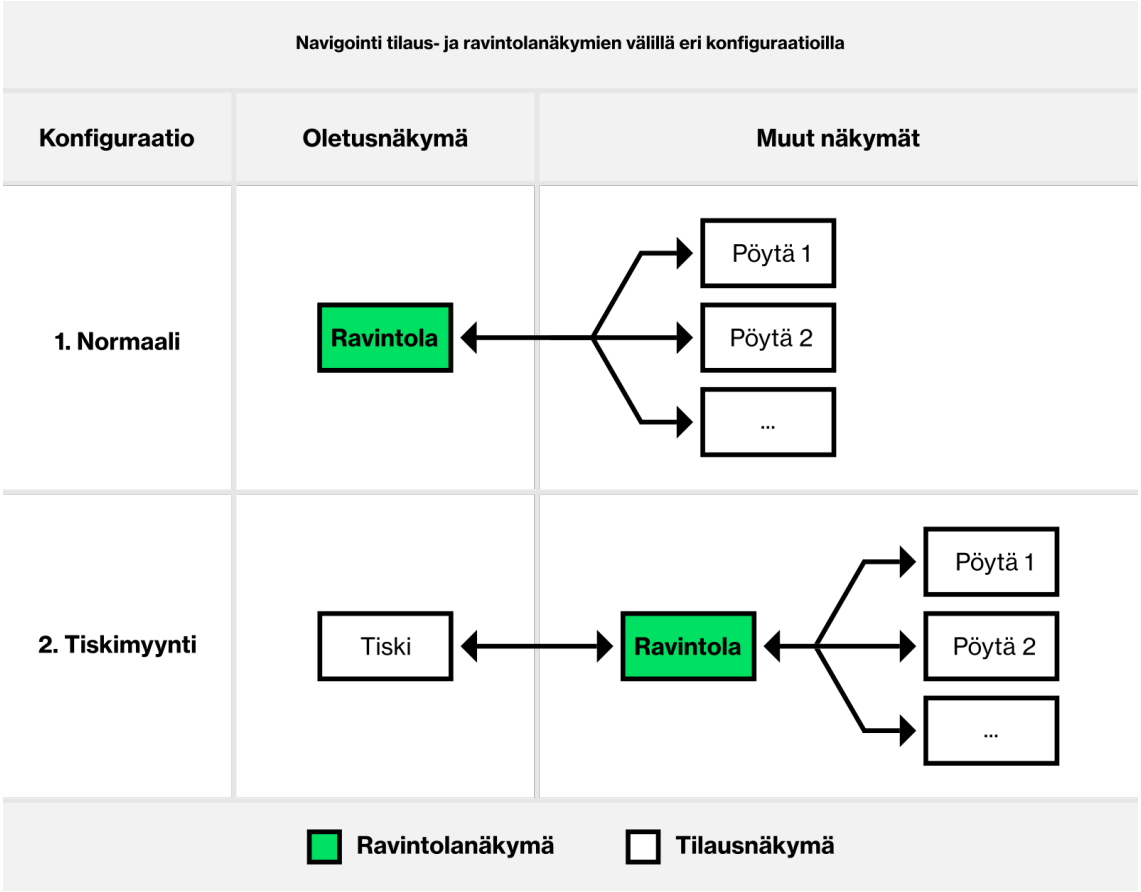
Koska valikko vie vähemmän tilaa, on piilotusominaisuus voitu poistaa. Käytännössä tämä vähentää käyttäjän painalluksia, kun käyttäjän ei tarvitse enää erikseen piilottaa ja ottaa valikkoa esiin. Se myös vähentää ylimääräistä kognitiivista kuormitusta, kun käyttäjän ei tarvitse tilannekohtaisesti harkita kannattaako valikon piilottamiseen tuhlata aikaa vai ei (Whitenton 2013).

Uudesta päävalikosta käyttäjä näkee suoraan sijaintinsa päänäkymien välillä. Avoinna oleva näkymä on selkeästi korostettu muista valikon painikkeista ja yhdistyy taustavärinsä ansiosta osaksi itse näkymää. Visuaalisesti päävalikko siis muistuttaa välilehtipalkkia, jossa jokainen näkymä on omalla välilehdellään. Tämän ansiosta myöskään erillistä näkymän sisäistä otsikkoa ei enää tarvita ja näkymälle jää enemmän tilaa muuhun käyttöön.

4.3 Navigointi myyntinäkymän sisällä

Myyntinäkymä poikkeaa muista päänäkymistä siten, että se on todellisuudessa kaksi erillistä näkymää. Normaalisti oletuksena auki on ravintolanäkymä, josta valitaan, minkä pöydän tilausta käsitellään. Sen lisäksi on tilausnäkymä, jossa tilaus kootaan. Tilausnäkymässä on painike, josta pääsee takaisin ravintolanäkymään. Myyntinäkymän oletusnäkymä on se näkymä, johon käyttöliittymä palaa aina kun edellinen tilaus on maksettu tai kun käyttäjä tulee muusta päänäkymästä myyntinäkymään.

Lisäksi myyntinäkymällä on vaihtoehtoinen tiskimyyntikonfiguraatio erilaista käyttöympäristöä varten. Siinä oletusnäkymänä on erillinen tilausnäkymä, jonka tilaus ei ole liitettynä mihinkään pöytään. Tähän tiskimyyntinäkymään pääsee erillisestä painikkeesta ravintolanäkymässä. Myyntinäkymän sisäisen navigoinnin erot molemmilla konfiguraatioilla on kuvattu kuviossa 3. Huomionarvoista on se, kuinka tässä konfiguraatiossa liikkuminen pöytätilausnäkymistä *oletusnäkymään* vaatii käyttäjältä kaksi painallusta.

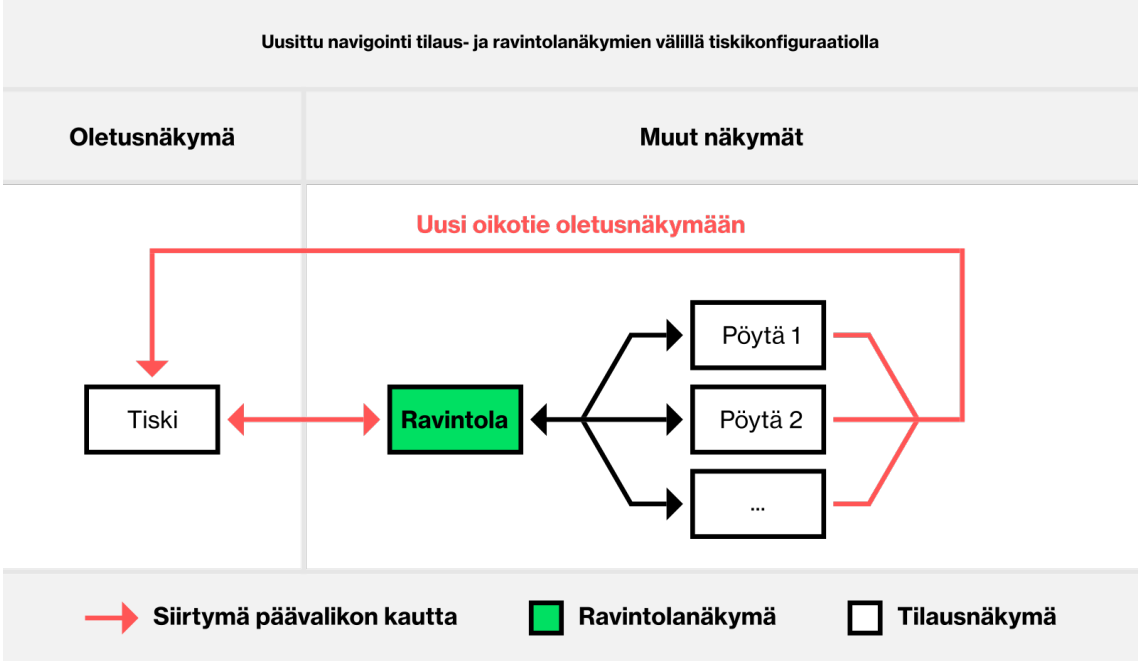


KUVIO 3. Navigointi tilaus- ja ravintolanäkymien välillä eri konfiguraatioilla

KassaUI:ssa myyntinäkymän sisäinen navigointi on uudistettu kokonaan. Ennestään kaksi erillistä näkymää sisältänyt myyntinäkymä on jaettu varsinaisiksi päänäkymiksi, joilla molemmilla on oma painike päävalikossa. Koko ajatus erillisestä ”myyntinäkymästä” on poistettu käyttöliittymästä. Oletusnäkymä, eli normaalisti ravintolanäkymä, on ensimmäisenä päävalikon oikeassa reunassa. Kun ravintolanäkymässä valitsee pöydän, aukeaa se uudelle pöydän mukaan nimetylle tilausnäkymävälilehdelle. Päävalikkoon siis ilmestyy uusi väliaikainen välilehti, joka katoaa näkymästä poistuttaessa.

Tilausnäkymästä ravintolanäkymään palataan päävalikon kautta, eikä erillisestä painikkeesta tilausnäkymän sisältä. Tämä on selkeyden kannalta tärkeä seikka. Vanhassa päävalikossa ravintolanäkymä ja tilausnäkymä on yhdistetty samaan ”Myynti”-painikkeeseen. Mikäli käyttäjä on jo jommassakummassa näkymässä, ei päävalikon ”Myynti”-painike tee mitään, kun taas kaikissa muissa näkymissä siitä pääsee suoraan myynnin oletusnäkymään, esimerkiksi ravintolanäkymään.

Mikäli konfiguraatioon kuuluu vaihtoehtoinen tiskimyyntinäkymä, se on ensimmäisenä päävalikon vasemmassa reunassa ylimääräisenä tilausnäkymävälilehtenä. Oletusnäkymä on siis tällöinkin loogisimmalla paikalla valikossa. Näin tähän oletusnäkymään pääsee mistä tahansa päänäkymästä yhdellä painalluksella, kun ennen siihen vaadittiin pöytätilausnäkymästä kaksi. Kiinnitin tähän erityistä huomiota, koska oletusnäkymä on se näkymä, jota käyttäjän ajattelin tarvitsevan useimmiten. Käyttäjällä siis pitäisi aina olla sinne on mahdollisimman helppo pääsy. Kuvio 4 havainnollistaa tätä uutta tiskimyyntikonfiguraation oikotietä.



KUVIO 4. Uusittu navigointi tilaus- ja ravintolanäkymien välillä tiskikonfiguraatiolla

4.4 Tilausnäkymä

Käyttöliittymän tärkeimmät toiminnot löytyvät tilausnäkymästä (kuvio 2). Koko sovelluksen toiminta pyörii tilausten ympärillä ja tilausnäkymä on pohjimmiltaan tilauseditori, jossa tilaukset kootaan tietokannassa olevista tuotteista. Muita toimintoja ovat esimerkiksi tilauksen maksaminen, tulostaminen keittiöön tai tilauksen poistaminen. Näkymä on suunnittelun näkökulmasta haasteellinen: se sisältää valtavasti toimintoja ja elementtejä sekä on lisäksi käyttäjälle koko käyttöliittymän keskipiste.

Openbravo POS:in tilausnäkymä jakautuu visuaalisesti kahteen allekkaiseen osaan. Alemmalta puoliskolta löytyy tuotelistat ja ylemmältä puoliskolta tilauslista ja muut toiminnot. Käyttöliittymä toimii siten, että tuotelistalta valitaan tuotteita, jolloin tuotteet ilmestyvät yläosassa olevaan tilauslistaan tilausriveiksi. Näkymän oikean reunan numeronäppäimistöltä valitaan, montako tuotetta lisätään. Mikäli numeronäppäimistöä ei näppäile ennen tuotteen valintaa, lisätään vakiona aina yksi tuote.

Tuotelistaus on hierarkkinen: vasemmalta valitaan tuotekategoria ja valitun kategorian tuotteet näkyvät oikealla. Mikäli tuotteita tai kategorioita on enemmän kuin ruudulta varattuun tilaan mahtuu, saa loput tuotteet näkyviin vierittämällä molemmista listoista löytyvistä vierityspalkeista.

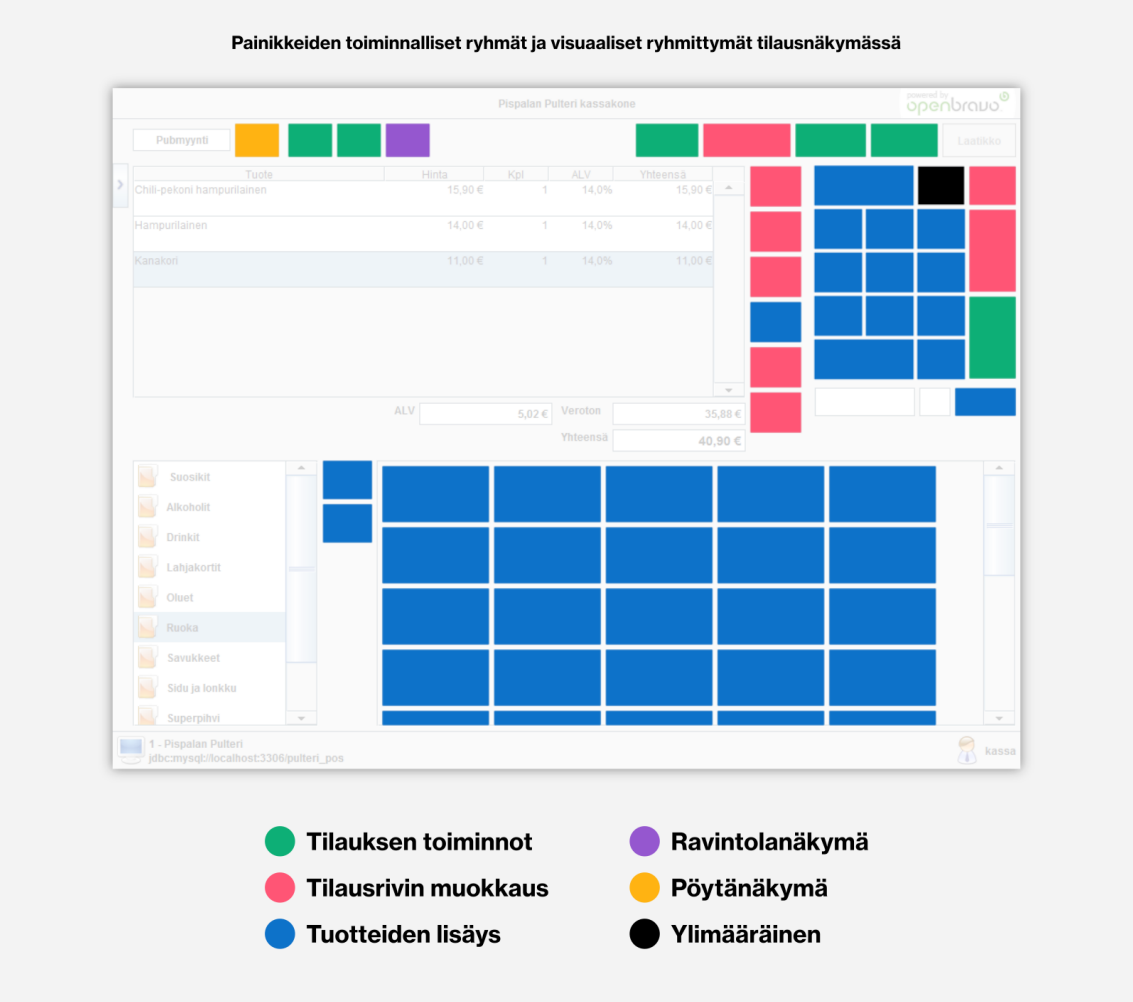
Tilauslistassa näkyvät tilauksen tuotteet tietoineen, kukin omalla tilausrivillään. Yhdellä tilausrivillä näkyy tuotteen nimi ja hintatiedot, tuotteiden lukumäärä ja mahdolliset lisätiedot. Lisäksi rivillä näkyy tarvitseeko riviä tulostaa keittiöön ja onko sitä vielä tulostettu. Varsinaisten tuotteiden lisäksi mahdolliset kokonaisalennukset veroluokittain ja loppuhinnan pyöristys näkyvät tilauslistassa omilla riveillään. Myös tilauslistaa vieritetään vierityspalkista.

Tilauslistan rivejä voi muokata erilaisilla toiminoilla. Esimerkiksi tuotteen poistaminen tapahtuu siten, että ensin valitaan tilausrivi painamalla sitä ja sen jälkeen painetaan tuotteen poistopainiketta. Sekä tilauslistassa, että tuotekategorialistassa on vierityspalkin lisäksi erilliset painikkeet, josta painamalla valinta liikkuu yhden askeleen ylös- tai alaspäin.

Tilausnäkymän painikkeet voidaan jaotella sen perusteella, mihin painikkeen toiminto kohdistuu. Yhden ryhmän muodostavat esimerkiksi painikkeet, joista muokataan valittua tilausriviä. Muita selkeitä ryhmiä ovat koko tilaukseen kohdistuvat painikkeet ja painikkeet joista lisätään tuotteita tilauslistaan. Näiden kolmen ryhmän ulkopuolelle jäävät painikkeet ovat: jo tutuksi tullut painike, josta pääsee takaisin ravintolanäkymään; pöytänäkymäpainike, johon perehdytään tarkemmin myöhemmin ja vielä yksi ylimääräinen painike, jolla ei ole mitään funktiota.

Kuviossa 5 näkyy, miten tilausnäkymän eri toimintoryhmiin jakautuvat painikkeet on asemoitu. Kuvassa kullekin toiminnalliselle ryhmälle on annettu väri, jolla ryhmään

kuuluvat painikkeet on korostettu. Alapuoliskon tuotelistaus on selkeä, mutta ylemmän puoliskon painikkeiden asemoinnissa ei tunnu olevan mitään yhtenäistä logiikkaa. Yläpuoliskon painikkeet on jaoteltu neljään visuaaliseen ryhmittymään: yläreunan vasemman- ja oikeanpuoleiset vaakarivit sekä tilauslistan viereinen pystyrivi ja sen vieressä oleva numeronäppäimistö. Jokaisessa visuaalisessa painikeryhmittymässä on painikkeita useasta eri toiminnallisesta ryhmästä. Esimerkiksi tilausrivin eri muokkauspainikkeita (kuvassa punaisella) löytyy kolmesta eri paikasta.



KUVIO 5. Painikeryhmittymät alkuperäisen käyttöliittymän tilausnäkymässä

KassaUI:n tilausnäkymän suunnittelussa ensimmäinen tavoite oli tehostaa tilankäyttöä. Tilausnäkymässä tila on erityisen tärkeässä roolissa, sillä sekä tuote- että tilauslista vaativat paljon tilaa. Mitä enemmän tilaa listoilla on käytössä, sitä suurempi osa niistä on näkyvillä ja valittavissa kerralla, mikä puolestaan vähentää tarvetta listan vierittämiseen.

Tilankäyttöä on tehostettu järjestelemällä elementtejä uudelleen ja poistamalla kaikkea ylimääräistä, kuten ylimääräisiä painikkeita ja marginaaleja. Testatulla näyttökoolla hyödyt ovat huomattavat. Tuotelistan tuotteista näkyy kerralla 50% enemmän kuin alkuperäisessä käyttöliittymässä. Tuotekategorialista näyttää yhtä monta kategoriaa kuin ennen vaikka listaelementtejä on suurennettu. Sama koskee tilauslistan tilausrivejä. Lisäksi kaikkien edellä mainittujen listojen kirjasinkokoa on suurennettu alkuperäisestä, mikä puolestaan parantaa listojen luettavuutta.

Listojen vieritystä on helpotettu huomattavasti. Kosketusnäyttökäyttöliittymien yleistymisen myötä voidaan olettaa käyttäjillä olevan ajatusmalli, jonka mukaan kosketusnäytöillä vierittäminen tapahtuu pääasiassa raahaamalla. Alkuperäisessä käyttöliittymässä raahaaminen ei kuitenkaan ollut mahdollista. Käyttöliittymän kuitenkin pitäisi hyödyntää käyttäjien olemassa olevia ajatusmalleja virheiden välttämiseksi (Lidwell ym. 2010, 154). Tästä johtuen KassaUI:ssa listojen vieritys tapahtuu raahaamalla listaa itsessään samaan tapaan kuin useimmissa nykyaikaisissa kosketuskäyttöliittymissä.

Kosketusnäytöllä toimiva vierityspalkki vie huomattavan paljon tilaa, sillä palkin on oltava tarpeeksi leveä sormella käytettäväksi. Koska vieritys tapahtuu raahaamalla, on listojen vierityspalkeista voitu poistaa kaikki tilaa vievä toiminnallisuus. Vierityspalkeista on jäljellä pelkästään pienet sijainti-indikaattorit, minkä ansiosta ne vievät huomattavasti vähemmän arvokasta tilaa kuin vanhassa käyttöliittymässä.

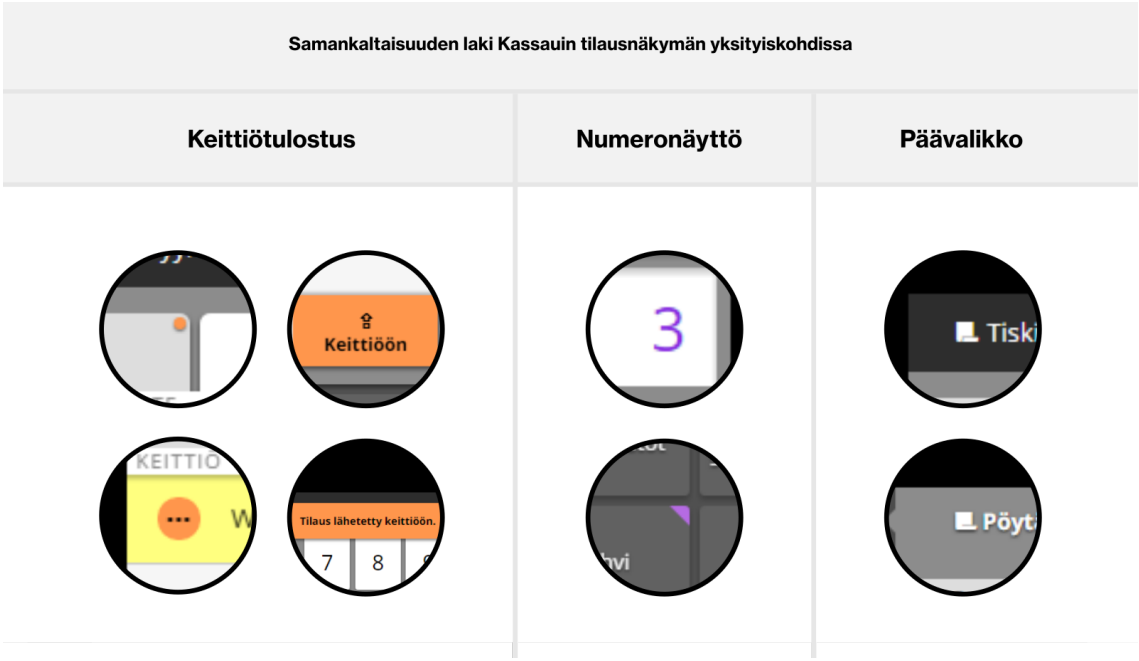
KassaUI:ssa tilausnäkymän painikkeita on yhdistelty ja järjestelty uudelleen. Kuviossa 6 näkyy väreillä korostettuna kuinka saman toiminnallisen ryhmän muodostavat painikkeet ovat sijoiteltu loogisiksi visuaalisiksi ryhmittymiksi. Ero alkuperäiseen on huomattava – vertaa kuviota 6 kuvioon 5. Kuvissa näkyy myös kuinka uusi elementtien sommittelu on huomattavasti yksinkertaisempi. Painikkeita ja elementtejä on vähemmän ja ne tasautuvat entistä yhtenäisempiin linjoihin mikä tekee näkymästä esteettisemmän (Galitz 2007, 152), edistäen näin myös näkymän käytettävyyttä (Lidwell ym. 2010, 20).

KUVIO 6. Painikeryhmittymät KassaUI:n tilausnäkymässä

Sen lisäksi, että KassaUI:ssa painikkeet on loogisemmin sijoiteltu toisiinsa nähden, on niiden sijoittelu loogisempi myös muihin elementteihin nähden. Näkymä jakautuu kahteen visuaaliseen osaan: tilaukseen ja tuotepainikkeisiin. Tilaus on selkeästi omassa laatikossaan, jonka ylä- ja alareunassa on siihen liittyviä painikkeita. Näiden painikkeiden välissä on itse tilauksen sisältö. Tilausrivien toimintopainikkeet löytyvät loogisesti itse tilausrivien sisältä.

Näiden uudelleenjärjestelyjen tuottama näkymän loogisuus perustuu hahmopsykologian läheisyyslakiin. Läheisyyslain mukaan ihminen mieltää lähekkäin olevat elementit samaan ryhmään kuuluviksi (Lidwell ym. 2010, 196). Toinen, hieman vastaava hahmolaki on samankaltaisuuden laki. Sen mukaan ihminen mieltää samankaltaiset elementit samaan ryhmään kuuluviksi. Samankaltaisuus voi tarkoittaa joko samaa väriä, muotoa tai kokoa. (Lidwell ym. 2010, 226) Samankaltaisuuden lakia on hyödynnetty tilausnäkymän pienissä yksityiskohdissa (kuvio 7) joiden tarkoitus on selventää näkymän syy-seu-

raussuhteita. Kun käyttäjä näppäilee luvun numeronäppäimistöllä, ilmestyy tilauslistan tuotteiden yläkulmaan lukumääräindikaattori ilmoittamaan että tuotetta ollaan lisäämässä useampi kappale. (Indikaattori itsessään on uusi ominaisuus KassaUI:ssa) Indikaattori on saman violetinvärinen kuin numeronäppäimistön numeronäytön numerot. Vastaava yhteys on kaikessa keittiötulostukseen liittyvässä: sama oranssi väri toistuu ympäri käyttöliittymää. Esimerkki muodon käyttämisestä elementtien yhdistävänä tekijänä löytyy tilausnäkymän painikkeista päävalikossa. Tilausnäkymä, oli se sitten ”oikotie tiskimyyntinäkymään”, tai pöydän tilausnäkymä on merkitty päävalikossa samalla ikonilla.



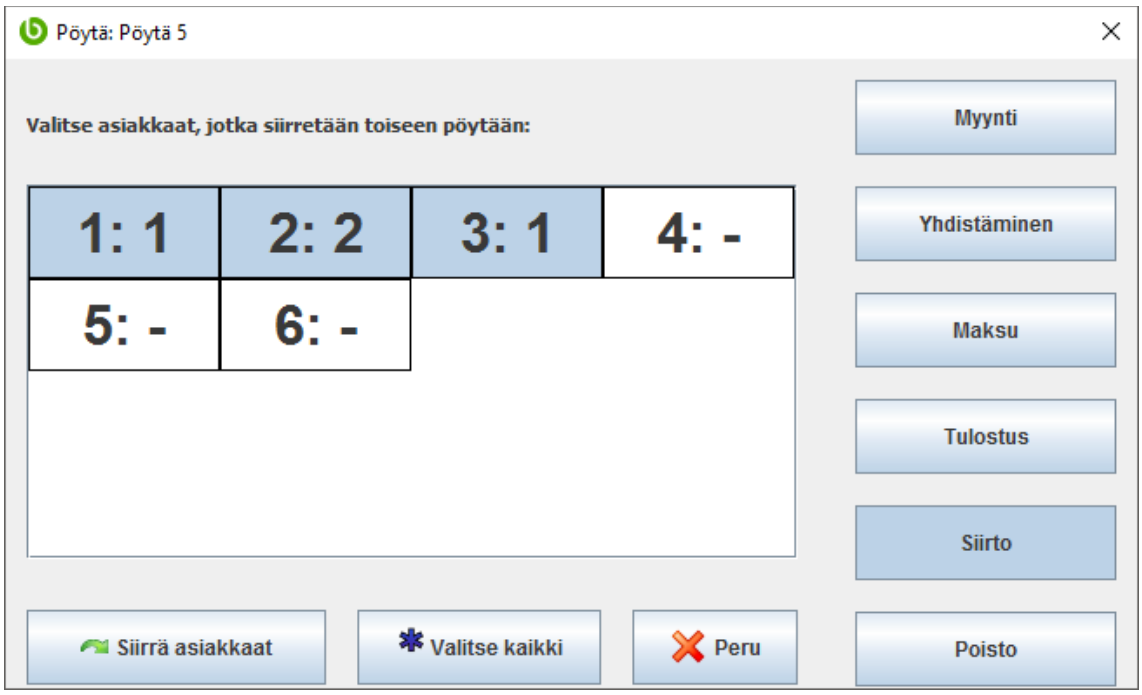
KUVIO 7. Samankaltaisuuden laki KassaUI:n tilausnäkymän yksityiskohdissa

4.5 Pöytänäkymä

Päänäkymien lisäksi on vielä erilaisia alanäkymiä. Alanäkymiksi luokittelen kaikki näkymät, jotka aukeavat varsinaisen näkymän päälle erilliseen ikkunaan. Esimerkiksi tilausnäkymän alla on rivin muokkausnäkymä, maksunäkymä, tuotehaku ja pöytänäkymä.

Pöytänäkymä (kuvio 8) on osa vaihtoehtoista ominaisuutta, jossa yhden pöydän tilaus on jaettu useaksi erilliseksi tilaukseksi siten, että jokaisella asiakkaalla on oma tilauksensa. Pöytänäkymässä hallitaan näitä tilauskokonaisuuksia. Sieltä lisätään, poistetaan, siirretään ja yhdistetään pöydän asiakkaiden tilauksia. Sieltä voi tulostaa kuitteja ja

maksaa kerralla useampia tilauksia. Pöytänäkymän kautta myös valitaan, ketä pöydän asiakkaista tarkastellaan varsinaisessa tilausnäkymässä.

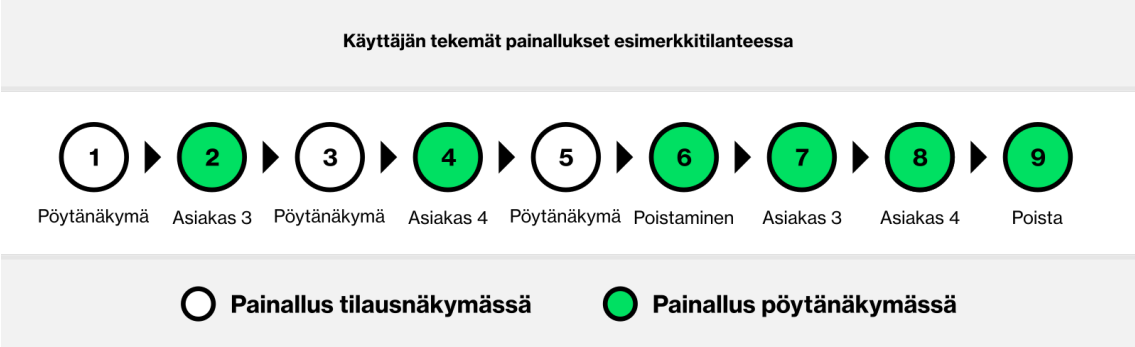


KUVIO 8. Alkuperäinen pöytänäkymä

Pöytänäkymä aukeaa käyttäjän valitessa pöydän ravintolanäkymässä. Siihen pääsee myös erillisestä painikkeesta tilausnäkymässä. Pöytänäkymä toimii siten, että ensiksi valitaan, mikä toiminto halutaan tehdä, sitten valitaan yksi tai useampi asiakas asiakas-ruudukosta ja lopuksi painetaan vasemman alareunan toimintopainiketta, josta valittu toiminto toteutuu. Poikkeuksena on oletustoimintona oleva asiakkaan avaaminen tilausnäkymään, joka tapahtuu heti kun asiakas on valittu ruudukosta. Silloin toimintopainikkeen tilalla on painike, josta lisätään uusia asiakkaita pöytään.

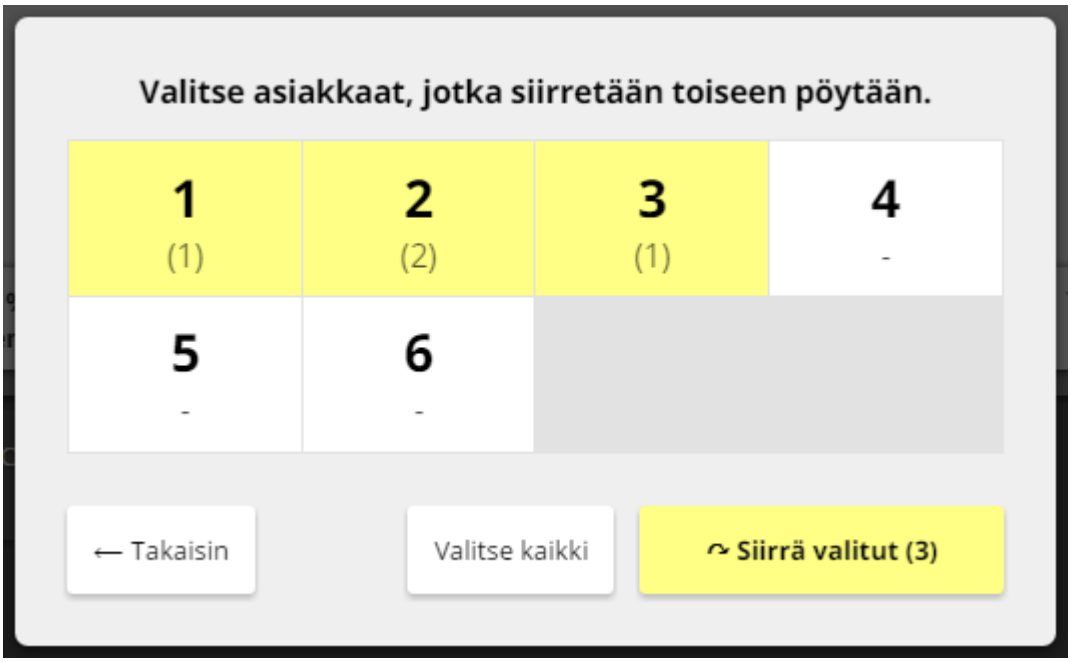
Esimerkkitilanne: käyttäjä katselee tilausnäkymässä nelihenkisen pöydän asiakkaan 1 tilausta. Tavoitteena on tarkastaa asiakkaiden 3 ja 4 tilausten sisällöt ja sitten poistaa ne. Käyttäjä aloittaa painamalla näkymän yläreunassa olevaa pöytänäkymäpainiketta. Tilausnäkymän päälle aukeaa pöytänäkymä, josta käyttäjä valitsee asiakkaan 3. Tilausnäkymään aukeaa asiakkaan 3 tilaus, jonka käyttäjä tarkistaa. Seuraavaksi käyttäjä tekee samat askeleet asiakkaan 4 kanssa. Lopuksi käyttäjä avaa pöytänäkymän vielä kerran. Tällä kertaa käyttäjä valitsee pöytänäkymän poista-toiminnon, valitsee käyttäjät 3 ja 4 ja lopuksi painaa poista-painiketta. Koko prosessi on kuvattu kuviossa 9. Kuviosta ilme-

nee, että prosessi vaatii käyttäjältä yhteensä yhdeksän painallusta. Huomionarvoista on myös se, että prosessin aikana näkymä vaihtuu viisi kertaa.

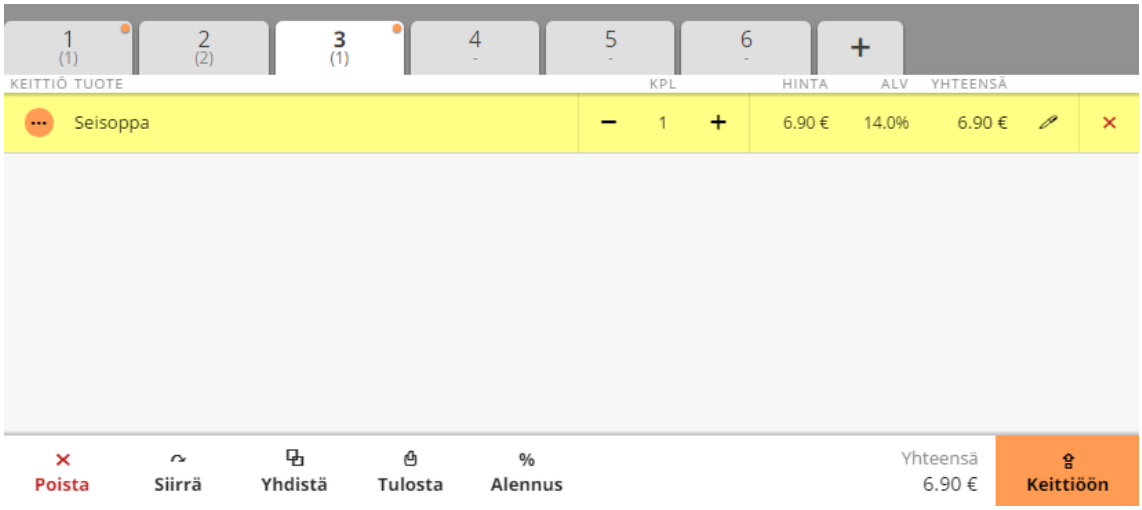


KUVIO 9. Käyttäjän tekemät painallukset esimerkkitilanteessa

KassaUI:ssa pöytänäkymää (kuvio 10) on suoraviivaistettu siirtämällä toimintoja tilausnäkymän yhteyteen (kuvio 11) ja yhdistämällä toimintoja tilausnäkymän vastaaviin toimintoihin. Pöydän asiakkaasta toiseen siirtyminen tapahtuu tilausnäkymän yläreunassa olevasta välilehtivalikosta. Samasta välilehtivalikosta myös lisätään asiakkaita tilaukseen. Erillistä pöytänäkymää ei siis enää edes tarvita kumpaankaan näistä toiminnoista.



KUVIO 10. Pöytänäkymä KassaUI:ssa



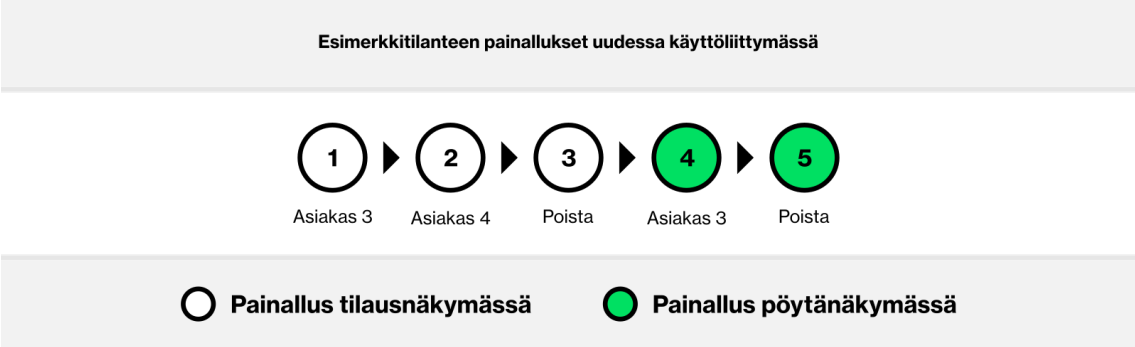
KUVIO 11. Pöytänäkymästä tilausnäkymään siirretyt toiminnot

Pöytänäkymää kuitenkin käytetään yhä kun halutaan poistaa, yhdistää, siirtää, maksaa tai tulostaa myyntejä. Pöytänäkymää ei avata erillisestä tilausnäkymän painikkeesta kuten ennen, vaan se ilmestyy ylimääräiseksi askeleeksi aina kun käyttäjä valitsee jonkun näistä jo tilausnäkymässä olevista toiminnoista. Eli käyttäjän valitessa tilausnäkymässä esimerkiksi poista-toiminnon, ei poistetakaan avoinna olevaa tilausta, vaan aukeaa pöytänäkymä josta käyttäjä voi valita asiakkaat joiden myynnit poistetaan ja poistaa ne. Kun pöytänäkymä aukeaa, on siinä jo valmiina valittuna tilausnäkymässä avoinna ollut asiakas. Tämä helpottaa käyttöä tilanteissa joissa käyttäjä haluaa toiminnon kohdistuvan nimenomaan avoinna olevaan tilaukseen.

Yksi huomionarvoinen seikka on siinä miten painikkeiden uusi jako pöytänäkymän ja tilausnäkymän välille hyödyntää recognition over recall -periaatetta. Periaate tarkoittaa esimerkiksi sitä, että kannattaa suosia vaihtoehtojen näkyvyyttä niiden piilottamisen sijaan. Käyttäjän on helpompi muistaa, miten joku asia toteutetaan kun se on näkyvillä, eikä piilotettuna toiseen näkymään (Johnson 2010, 113; Lidwell ym. 2010, 200). Kun käyttäjä haluaa tulostaa kaikkien saman pöydän asiakkaiden kuitit, hänen ei tarvitse muistaa mistä painikkeesta pääsee tulostamaan useita kuitteja kerralla ja mikä painike tulostaa vain yhden kuitin. Alkuperäisessä tilausnäkymässä oleva tulosta-painike tulostaa vain kyseisen tilauksen kuitin – KassaUI:ssa riittää että käyttäjä löytää sen yhden ainoan tulosta-painikkeen systeemin huolehtiessa lopuista.

Parhaiten uuden pöytänäkymän edut vanhaan nähden nähdään kun otetaan aikaisempi esimerkkitilanne ja katsotaan kuinka sama prosessi onnistuu uudella käyttöliittymällä

(kuvio 12). Alkuperäisessä käyttöliittymässä yhdeksän painallusta vaatinut toimenpide vaatii nyt vain viisi. Ylimääräisen näkymien välillä kikkailun vähentäminen on poistanut huomattavan määrän painalluksia. Huomaa myös, että asiakasta 4 ei valita lainkaan pöytänäkymässä, sillä se on jo valmiiksi valittuna.

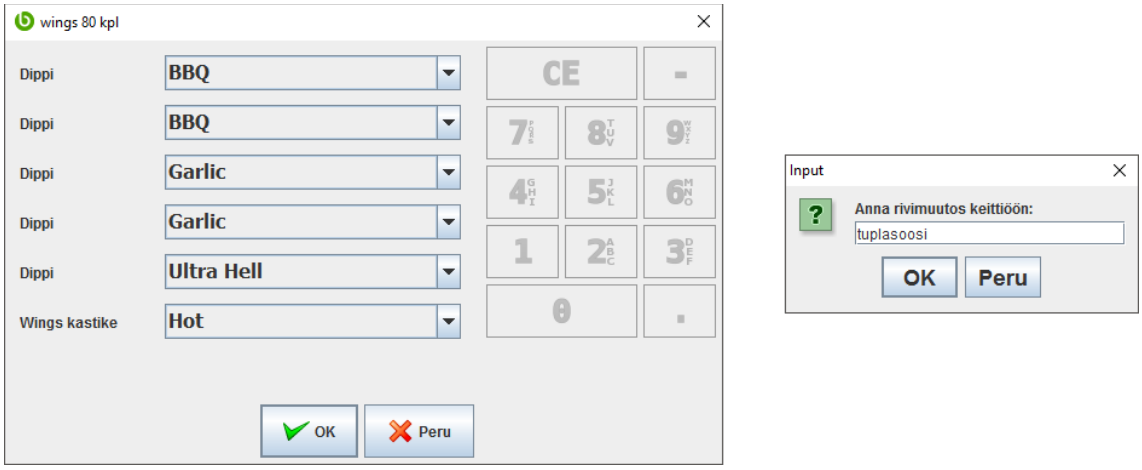


KUVIO 12. Esimerkkitilanteen painallukset KassaUI:ssa

4.6 Lisätiedot keittiöön

Viimeisenä esimerkkinä vielä näkymäpari, joka on yksinkertainen ja selkeä esimerkki siitä kuinka paljon turhaa tekemistä käyttöliittymä voi käyttäjälleen tuottaa ja niistä kompromisseista joiden avulla ongelmaa on lähdetty ratkaisemaan. Homma vaati vain hieman uudelleenjärjestelyä ja tinkimistä käyttöliittymän joustavuudesta.

Keittiöön lähetettäville tilauksille voi antaa tuotekohtaisia lisätietoja. Ensinnäkin on ennalta määrättyjä ominaisuuksia, kuten esimerkiksi vaihtoehto vaalean ja tumman leivän välillä. Lisäksi on vapaa kenttä, johon voi kirjoittaa kaikki muut muutokset. Alkuperäisessä käyttöliittymässä nämä keittiön lisätiedot ovat jaettu omiin tilausnäkymästä aukeaviin alanäkymiinsä (kuvio 13).



KUVIO 13. Keittiöön lähtevät lisätiedot alkuperäisessä käyttöliittymässä

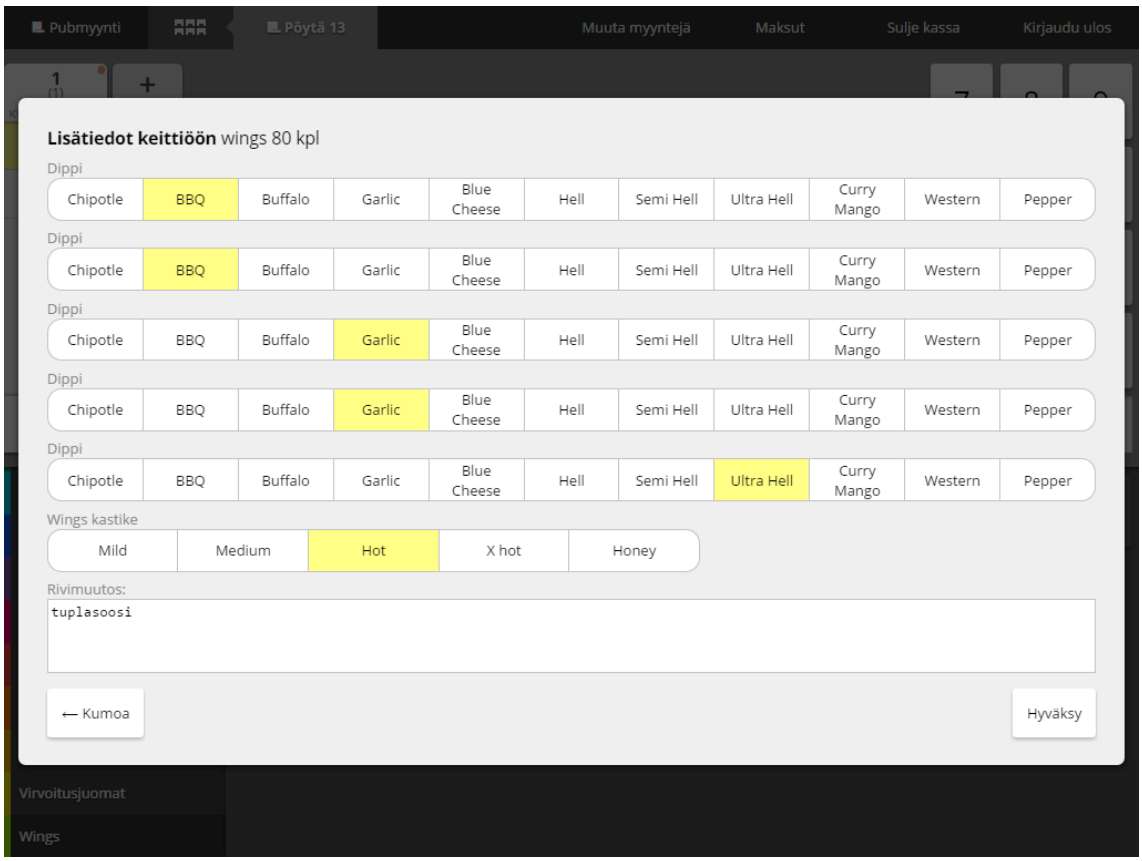
Tuotteen ennalta määrätyt lisätiedot ovat kukin omassa pudotusvalikossaan. Valinnan tekemiseen tarvitaan siis vähintään kaksi painallusta: ensimmäisellä avataan pudotusvalikko ja toisella tehdään valinta. Mikäli haluttu valinta ei ole pudotusvalikossa ensimmäisen kahdeksan vaihtoehdon joukossa, joutuu käyttäjä myös vierittämään valikkoa. Vierittäminen tapahtuu auttamattoman pienestä vierityspalkista, jonka leveys on testilaitteella vain 4 mm – alle puolet aiemmin mainitusta Microsoftin suosituksesta.

Lisätietojen kirjaaminen keittiöön vaatii kohtuuttoman paljon painalluksia. Esimerkiksi kuvion 13 valintojen tekeminen, vapaan kentän täyttäminen ja hyväksyminen vaati yhteensä 16 painallusta (pois lukien sanan ”tuplasoosi” kirjoittaminen) ja yhden pudotusvalikon vierityksen.

KassaUI:ssa nämä kaksi näkymää on yhdistetty yhdeksi ”tuotteen lisätiedot keittiöön”-näkymäksi (kuvio 14). Tämä vähentää sekä painikkeiden määrä tilausnäkymässä (jossa tila oli kortilla), että käyttäjän painalluksia silloin kuin tuotteen molempia tietoja halutaan muuttaa. Vähentääkseni vaadittujen painallusten määrää päätin myös ottaa käytettävissä olevasta tilasta kaiken irti: levitin pudotusvalikot auki, jotta käyttäjän ei tarvitse tuhlata siihen omaa aikaansa. Lopputuloksena se, mikä vei alun perin 16 painallusta ja yhden vierityksen, vie KassaUI:ssa vain yhdeksän painallusta – eikä vieritystä tarvittu ollenkaan.

Minkälaisia sitten ovat kompromissit, jotka johtivat tähän lopputulokseen? Vaakasuuntaiset valintalistat eivät ole absoluuttisesti pudotusvalikkoja parempia: ne vievät enemmän tilaa, niihin ei mahdu rajaton määrä valintoja, tehdyt valinnat eivät ole asemoitu

selkeäksi vasemmalle tasatuksi listaksi, valinnat ovat fyysisesti kauempana toisistaan, jne. Toisaalta vaakasuuntaisilla valintalistoilla on omat puolensa. Ratkaisun hyviä ja huonoja puolia on punnittu kokonaisuuden kannalta ja sen perusteella päädytty parhaaseen mahdolliseen lopputulokseen.



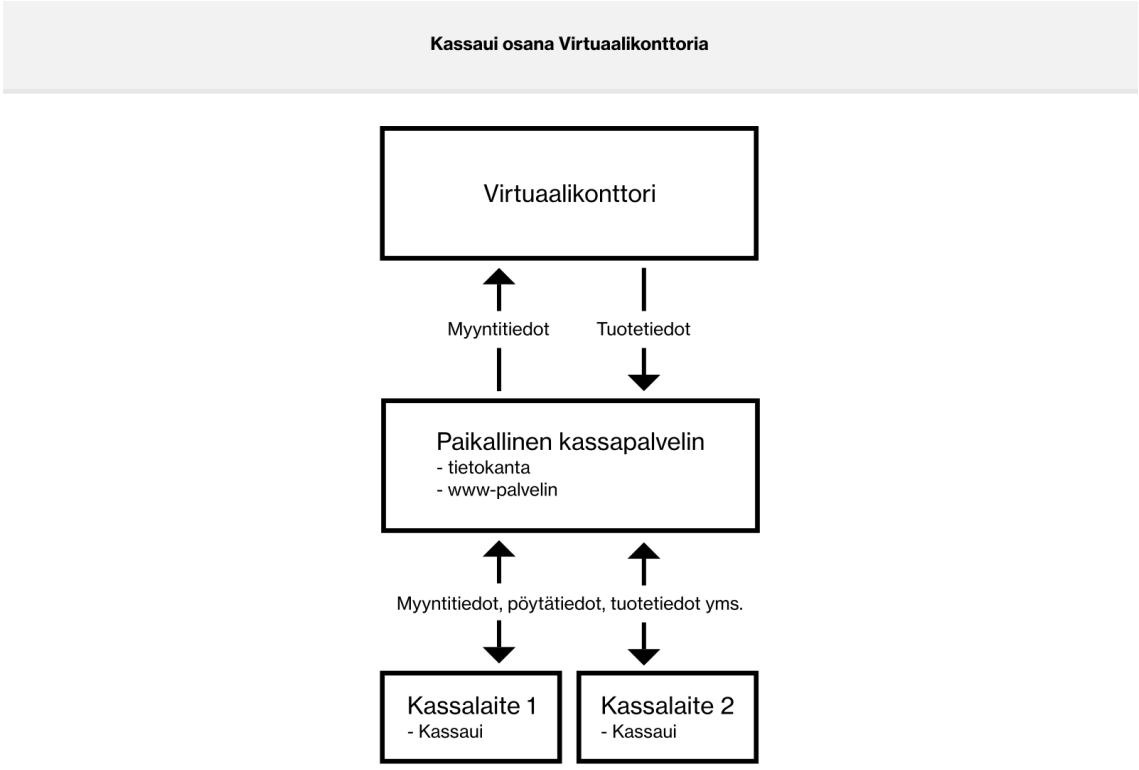
KUVIO 14. Keittiöön lähtevät lisätiedot KassaUI:ssa

5 KÄYTTÖLIITTYMÄN TEKNINEN TOTEUTUS

Käyttöliittymä toteutustapa on single-page application -tyyppinen web-sovellus. Sen ulkoasu koostuu CSS-tyylitetystä HTML-sivusta, jonka toiminnallisuus on toteutettu JavaScriptillä ja jQueryllä. Varsinaisen käyttöliittymäsovelluksen lisäksi palvelimella on PHP-sovellus, joka luo rajapinnan käyttöliittymän ja palvelimella olevan tietokannan väliin. Seuraavissa kappaleissa avataan tarkemmin sovelluksen toteutukseen käytettyjä tekniikoita.

5.1 Sovellus osana laajempaa kokonaisuutta

Toteutettu käyttöliittymäsovellus toimii osana laajempaa Virtuaalikonttori-toiminnanohjausjärjestelmää. Toteutetun sovelluksen on tarkoitus toimia nimenomaan ravintolan myynnin käyttöliittymänä. Kaikki välittömästi myyntiin liittymätön toiminnallisuus, kuten tuotteiden hallinnointi, tapahtuu Virtuaalikonttorin puolella. Kuviossa 15 on havainnollistettu sovelluksen eri osien suhteita ja asemaa Virtuaalikonttori-järjestelmässä.



KUVIO 15. KassaUI osana virtuaalikonttoria

5.2 Single-page application

Single-page application (SPA) tarkoittaa sellaista web-sovellusta, jonka kaikki toiminnallisuus on sisällytetty yhdelle web-sivulle. SPA-sovelluksen ero perinteiseen web-sivustoon tai -sovellukseen on se, että sovellus ladataan selaimeen vain kerran (Mikowski & Powell 2014, 4). Siinä missä perinteisesti selaimella kuljetaan sivuston sivulta toiselle, SPA-sovellus toimii kokonaisuudessaan yhdellä sivulatauksella.

SPA-sovellus sekä lähettää että vastaanottaa dataa palvelimelta asynkronisesti. Asynkronisuus tarkoittaa sitä, että sovelluksen toiminta ei keskeydy lataamisen ajaksi, vaan käyttöliittymä reagoi normaalisti myös sen aikana. SPA-sovelluksen etu perinteiseen web-sovellukseen verrattuna piilee osin juuri käyttöliittymän ripeämmässä reagointinopeudessa käyttäjän syötteisiin (Mikowski & Powell 2014, 20).

Ajax (Asynchronous JavaScript and XML) on keskeisessä osassa SPA-sovelluksia. Ajax-termin määritteli Jesse James Garrett kuvaamaan tekniikoiden joukkoa, jotka mahdollistavat datan asynkronisen lataamisen, käsittelyn ja esittämisen web-sivulla (Garret, 2005; Holzner 2007, 6). Alkuperäisen määritelmän mukaan Ajax viittasi nimenomaan XML-muotoisen datan käyttämiseen, mutta termi on ajan myötä löyhentynyt kuvaamaan myös muita datamuotoja. KassaUI:ssa data liikutellaan JSON-muotoisena.

Edellä kuvatuista seikoista johtuen SPA-sovellus muistuttaakin enemmän tavallista työpöytäsovellusta, kuin perinteistä web-sovellusta. Merkittävä ero on kuitenkin siinä, että SPA-sovellus vaatii taustalle www-palvelinohjelman ja sen käyttöliittymä pyörii verkkoselaimessa.

5.3 Sovelluskehykset

SPA-sovellusten kehitykseen on olemassa erilaisia valmiita sovelluskehyksiä, joiden tarkoituksena on antaa valmis kehikko ja työkalut, joiden päälle sovellus rakennetaan. Ne sisältävät työkaluja esimerkiksi näkymien luontiin, datan päivittämiseen näkymissä, datan hakemiseen palvelimelta ja päivittämiseen sinne. Esimerkiksi Angular JS ja Sencha Touch ovat tällaisia kokonaisia JavaScript-kehyksiä. (Angular JS; Sencha) Lisäksi on olemassa erillisiä kirjastoja, jotka ratkaisevat samoja ongelmia, mutta eivät ole varsinai-

sia kokonaisia sovelluskehyksiä. Esimerkiksi Facebookin alulle panema React-kirjasto on ratkaisu vain datan päivitykseen näkymässä mahdollisimman tehokkaasti (React).

Yleisesti ottaen sovelluskehysten hyödyt ovat sitä suurempia, mitä monimutkaisempi itse sovellus on. Pienissä projekteissa kokonaisen sovelluskehyksen käyttäminen vain monimutkaistaa asioita ilman varsinaista hyötyä. Tästä johtuen yksi haaste projektin alkuvaiheessa on sovelluksen laajuuden arviointi, jotta osataan valita parhaat mahdolliset työkalut. Päätin heti alussa pitää asiat mahdollisimman yksinkertaisina, sillä en halunnut toistaa ongelmia, joita olin aiemmin kohdannut työharjoittelussa Sencha Touch -sovelluskehyksen kanssa. Siispä valitsin vain pari kolme kirjastoa ja muilta osin ohjelmoin omat rakenteet ja systeemit, lisäten koodiin abstraktiota ja rakennetta aina tarpeen mukaan.

5.4 JavaScript

KassaUI:n selaimessa tapahtuvan toiminnallisuuden toteuttavat JavaScriptillä kirjoitetut moduulit. Jokainen moduuli toteuttaa yhden sovelluksen osan ja sijaitsee omassa JavaScript-tiedostossaan. Kullakin käyttöliittymän näkymällä on oma moduulinsa, joka huolehtii kyseisen näkymän toiminnoista. Muita moduuleja ovat muun muassa moduuli, joka hallitsee kaikkia näkymiä ja navigointia niiden välillä sekä moduuli, joka sisältää tuotteiden, tuoteominaisuuksien ja tuotekategorioiden muodostama tietorakenteen ja toiminnallisuuden.

KassaUI:n moduulit käyttävät revealing module pattern -mallia, joka mahdollistaa tiedon kapseloinnin. Kapselointi tarkoittaa sitä, että moduulin muuttujat ja metodit voi jakaa julkisiin ja yksityisiin. Yksityisiin moduulin muuttujiin ja metodeihin pääsee käsiksi vain kyseisen luokan sisältä – muut moduulit eivät pääse sotkemaan moduulin sisäisiä asioita. Julkiset muuttujat ja metodit taas muodostavat moduulin julkisen rajapinnan, jota muut moduulit pääsevät käyttämään. (Osmani 2015)

5.5 jQuery ja jQuery UI

jQuery on JavaScript-kirjasto, joka helpottaa HTML-sivun manipulointia ja esimerkiksi Ajax-pyyntöjen tekemistä (jQuery;). KassaUI:ssa HTML-sivun manipulointia tapahtuu näkymämoduuleissa ja se on toteutettu jQuerylla. Esimerkiksi kun käyttäjä lisää tuotteen tilaukseen, luodaan HTML-sivun tilauslistaan jQueryn avulla uusi elementti kuvastamaan lisättyä tuotetta.

jQuery UI on kokoelma jQueryllä toteutettuja käyttövalmiita graafisia käyttöliittymäelementtejä, efektejä ja työkaluja (jQuery UI). jQuery UI sisältää muun muassa käyttövalmiin dialogi-ikkunasysteemin. KassaUI:ssa jQuery UI:ta on hyödynnetty kuitenkin vain vähän. Siinä oli paljon ominaisuuksia, joita kaavailin käyttäväni, mutta erinäisistä syistä koko jQuery UI jäi vähän taka-alalle.

5.6 HTML

KassaUI:hin sisältyy vain yksi HTML-tiedosto: index.html. Tiedosto on se, jonka web-selain lataa, kun sovellus käynnistetään. Siinä on määritelty kaikki tarvittavat JavaScript-, CSS- ja kuvatiedostot, jotka ladataan käynnistyksen yhteydessä. Lisäksi tässä HTML-tiedostossa määritellään web-sivun rakenne eli elementtien suhteet toisiinsa nähden. Suurin osa sivun elementeistä on valmiina HTML-tiedostossa, mutta joitain osia myös luodaan dynaamisesti JavaScriptillä.

Perinteisessä web-sivustossa jokainen näkymä on omalla sivullaan, mutta SPA-sovelluksessa kaikki näkymät sisällytetään yhdelle sivulle. KassaUI:ssa jokaisen näkymän jokainen elementti on samalla web-sivulla, mutta käynnistyksen yhteydessä kaikki elementit päävalikkoa lukuun ottamatta piilotetaan jQueryllä. Aina uuteen näkymään siirtyessä tuodaan esiin kyseisen näkymän elementit ja piilotetaan edellisen näkymän elementit.

Kaikki käyttöliittymän staattinen teksti, kuten ohjetekstit ja painikkeiden tekstit, on sisällytetty samaan index.html tiedostoon oikeille paikoilleen. Dynaamisesti lisättyjen elementtien sisältämät tekstit haetaan joko tietokannasta tai määritellään erikseen JavaScript-koodissa.

5.7 CSS ja Flexbox

Suurin osa KassaUI:n CSS-tyyliohjeista on sisällytetty yhteen kassaui.css-tiedostoon. Tiedostossa määritellään kaikkien elementtien ulkonäkö ja asemointi sivulla. Tarkoituksena on ollut, varsinkin CSS-tiedoston laajentuessa ajan myötä yli 1200-riviseksi, siirtyä suoran CSS:n kirjoittamisesta johonkin kehittyneempään ratkaisuun, esimerkiksi Less-kieleen. Less on CSS:n kaltainen kieli, joka sisältää lisäominaisuuksia, esimerkiksi muuttujat, jotka helpottavat tyyliohjeiden päivittämistä ja lisäämistä. Selaimet eivät suoraan tue Less-kieltä, vaan se pitää erikseen kääntää CSS-kieleksi. (Less. Getting started)

Eräs CSS:n ongelmista on se, että eri selaimet tulkitsevat sivuja hieman eri tavoin. Siitä johtuen sama sivu näyttää eri selaimilla usein hieman erilaiselta kuin millaiseksi se on alun perin suunniteltu. KassaUI:n tapauksessa tätä ongelmaa ei ole, sillä toimeksiantajani voi kontrolloida millaisena pakettina kassalaitteet ja ohjelmistot lähtevät asiakkaalle. Tästä johtuen KassaUI:n CSS-ohjeet on voitu optimoida vain yhdelle selaimelle – tässä tapauksessa Mozilla Firefoxille.

Flexbox (CSS Flexible Box Layout Module Level 1) on suhteellisen uusi CSS:n ominaisuus, joka mahdollistaa elementtien asemoinnin ja venyttämisen yksinkertaisesti ja joustavasti. Flexbox-systeemi helpottaa huomattavasti joustavasti mukautuvien elementtien asemointia. Tästä johtuen lähes kaikki käyttöliittymäelementit on asemoitu Flexbox-mallia käyttäen.

Kirjoitushetkellä (29.8.2016) Flexbox on Candidate Recommendation -vaiheessa, eli siihen voi vielä tulla muutoksia ennen kuin se hyväksytään W3C:n (World Wide Web Consortium) suositukseksi (W3C 2016). Flexboxin tuoreudesta johtuen se myöskään toimi täydellisesti kaikilla selaimilla eikä sen käyttäminen sellaisenaan internetissä olevilla web-sivuilla ei välttämättä vielä ole kannattavaa. Koska käytettävä selain on KassaUI:n tapauksessa tiedossa, on Flexboxin käyttö kuitenkin mahdollista.

Lopputuloksena syntynyt elementtien asemointi on helposti muokattava ja laajennettava. Flexboxin ansiosta elementtien uudelleenjärjestely on mahdollista pelkkiä CSS-ohjeita muuttamalla. Elementit venyvät automaattisesti käyttämään kaiken tyhjän tilan kun niiden järjestystä ja kokoa muutetaan. Kun yhden elementin kokoa kasvatetaan, pienenevät viereiset elementit siten että kokonaistilankäyttö pysyy vakiona.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Työn tavoitteena oli tutkia käyttöliittymäsuunnittelun teoriaa ja teknistä toteutusta. Molemmat saavutettiin: sekä suunnittelusta että toteutuksesta kehkeytyi selkeä kokonaiskuva. Sen sijaan KassaUI-käyttöliittymäsovellus vaatii vielä työstämistä. Koko projektin laajuus tulikin yllätyksenä. Työtunteja kaiken ohjelmoinnin, suunnittelun, teoriaan perehtymisen ja raportin kirjoittamisen kesken on kertynyt vajaa 550, mikä vastaa noin 14:ää viikkoa kokopäiväistä työtä. Koodirivejäkin on kaikkiaan yhteensä lähes 6000, joista vajaa 4000 on JavaScriptiä. Sovelluksessa pystyy avaamaan, muokkaamaan, poistamaan ja tallentamaan tilauksia siten, että kaikki palvelimeen yhteydessä olevat kassat näkevät samat pöydät ja tilaukset. Tuotetiedot, pöytäkartta yms. tulevat palvelimelta ja ne toimivat suunnitellusti. Tilauksia myös pystyy yhdistelemään ja siirtelemään pöytien välillä, mikä ei tosin toimi vielä useamman kassan kanssa.

Sovelluksessa on myös paljon osia, jotka vaativat vielä työstämistä. Maksaminen, keittiötulostussysteemit ja alennussysteemi ovat kesken. Ne on suunniteltu ja osittain toteutettu, mutta niistä puuttuu vielä toiminnallisuutta. Sisäänkirjautuminen, maksettujen myyntien muokkaus ja kassan sulkeminen puuttuvat kokonaan.

Syitä projektin hitaaseen edistymiseen on ollut useita. Aikaa on mennyt välillä harhaillessa sellaisten systeemien parissa, jotka eivät alunperinkään kuuluneet projektin laajuuteen. Kului varmasti viikkoja tietokantojen ja palvelimen systeemien parissa ihan vain koska se oli mielenkiintoista. Lisäksi aikaa meni välillä jonkun näkymän, koodinpätkän tai pikselin asennon kokonaisessa uudelleensuunnittelussa. Lisäksi oman siivunsa ajasta on vienyt uuden opettelu – välillä hallitusti, välillä kantapään kautta.

Tämän laajuuden projektin saisi helposti jaettua ainakin kahdeksi tai useammaksi erilliseksi opinnäytetyöksi. Näin olisi mahdollisuus kunnolliseen, paremmin mietittyyn ja ennen kaikkea valmiiseen tekniseen toteutukseen, parempaan käyttöliittymäsuunnitteluun käytettävyystestauksineen, hienompaan ulkoasuun, animaatioihin jne. Tietysti kaiken voi tehdä yksikin henkilö, mutta silloin eteneminen on auttamattoman hidasta.

6.1 Käyttöliittymän suunnittelu

Tavoitteena oli alkajaisiksi tehdä sama kuin alkuperäinen käyttöliittymä muutamalla muutoksella ja uudella tekniikalla. Sovellukseen oli myös tarkoitus lisätä kaikenlaisia muokkausominaisuuksia, jotta käyttöliittymää voisi helposti säätää eri ympäristöihin. Nämä valmiiksi määritellyt muutokset pohjautuvat toimeksiantajan käyttäjiltä saamaan palautteeseen. Kaikki muut muutokset on tehty pelkän teorian pohjalta ilman varsinaista käyttäjädataa.

Käyttöliittymäsuunnittelussa on suuri virhe jättää käyttäjätestaus ja käyttöympäristö huomiotta. (Scoresby 2008) Käyttäjädatan puuttuessa on tehtyjä valintoja vaikeaa perustella kaikkein tärkeimmän kriteerin, eli loppukäyttäjän, kannalta. Nyt ne ovat parempia vain teoriassa. Suunnitteluprosessiin kyllä kuului keksittyjen käyttöskenaarioiden ja käyttäjistä tehtyjen oletusten hyödyntäminen. Koska nämä skenaariot ja oletukset eivät perustu mihinkään oikeaan dataan, ei niille ole annettu juurikaan painoarvoa, vaan käyttöliittymä on jätetty mahdollisimman alkuperäistä vastaavaksi. Suunnittelussa onkin keskitytty vain parannuksiin, jotka vaikuttavat paremmilta lähes kaikilla kuviteltavissa olevilla tavoilla. Myönnettäköön kuitenkin se, että myös täysin kyseenalaistettavissa olevia muutoksia on tehty. Esimerkiksi tilausnäkymän painikkeiden asemointi voi olla selkeämpi ja loogisempi kuin ennen, mutta painikkeiden uusi järjestys voi vaatia käyttäjältä enemmän käden liikuttelua – painikkeet ovat kauempana toisistaan kuin alun perin. Kumpi ratkaisu on tässä tapauksessa parempi? Mysteerin ratkaisu vaatisi enemmän tietoa käyttäjäkunnan tarpeista.

Miksi käyttäjädatan hankkiminen jäi kokonaan projektin ulkopuolelle? Projektin alkuvaiheessa ei ollut vielä tarkkaa kuvaa, missä projektin painopisteet tulisivat olemaan ja sitä lähestyttiin lähinnä tekniikan näkökulmasta. Oltiinhan kuitenkin tekemässä vain ”samaa kuin alkuperäinen käyttöliittymä, mutta muutamalla muutoksella.” Laajempi pohjatyö olisi joka tapauksessa lisännyt jo valmiiksi suurta työmäärää koska alkuperäistä käyttöliittymää käytetään erilaisissa ympäristöissä ja erilaisilla asetuksilla. Yhden käyttöympäristön vaatimukset täydellisesti huomioiva ratkaisu voi olla toisaalla täysin käyttökelvoton. Sen takia oli luontevaa pohjata suunnitteluprosessi siihen, mikä kaikkia ympäristöjä tiedettävästi yhdistää, eli alkuperäiseen käyttöliittymään.

6.2 Tekninen toteutus

Sopivien työkalujen valitseminen toi oman haasteensa – siinä päädyttiin mahdollisimman yksinkertaiseen matalan tason ratkaisuun. Käyttöön otettiin vain pari hyödyllistä kirjastoa – ei erillistä sovelluskehystä. Seuraavissa kappaleissa käydään läpi ratkaisun hyvät ja huonot puolet.

Aloitetaan huonoista puolista. Käyttöliittymän monimutkaisin näkymä, tilausnäkymä, oli ensimmäinen, jonka parissa varsinainen ohjelmointityö aloitettiin. Yksinkertainen tilauslista oli toiminnassa suhteellisen nopeasti ilman suurempia ongelmia. Kun näkymään tuli lisää ominaisuuksia – tilauksen lisätiedot, keittiön indikaattorit, loppusumma, jne. – piti aina myös vähän muuttaa ja laajentaa datan päivityslogiikkaa. Näin logiikasta tuli pikkuhiljaa todella monimutkainen ja vaikeasti ylläpidettävä. Lopulta huomattiin, että näkymään tultaessa erään painikkeen tila päivitettiin kaksi kertaa jokaista listan riviä kohden, vaikka kerran *koko listaa* kohden olisi riittänyt. Esimerkiksi 12-rivisessä listassa tämä tarkoittaa 23:a ylimääräistä painikkeen tilan päivitystä. Virheen paikannuksen yhteydessä tuli selväksi että koko systeemi on pakko suunnitella alusta. Uudelleensuunnitteluun meni aikaa eikä lopullinenkaan ratkaisu ole täydellinen. Vastaavia, systeemin laajetessa ja monimutkaistuessa syntyneitä ongelmia tuli vastaan myös muilla sovelluksen osa-alueilla. Sovelluskehykset, joita projektin alkuvaiheessa haluttiin välttää, ovat alun perin suunniteltu ratkaisemaan juuri tämänkaltaisia ongelmia.

Toisaalta ratkaisu pitäytyä matalan tason kirjastoissa oli oman oppimiseni kannalta erinomainen. Aikaisemmin työharjoittelussa kertyneet kokemukset Sencha Touch sovelluskehyksen parissa eivät olleet kovin innostavia – sovelluskehys tuntui vain hankaloittavan työskentelyä ja tekevän siitä tarpeettoman monimutkaista. Sovelluskehystä tulikin silloin käytettyä pitkin hampain vain koska se oli projektissa määriteltynä. KassaUI:n monimutkaisen käyttöliittymän toteutuksesta saadun kokemuksen perusteella näyttäytyvät sovelluskehyksetkin uudessa valossa. Näin on saavutettu tarkempi ymmärrys siitä, mitä sovelluskehyksen pinnan alla tapahtuu ja miten monimutkaiset web-sovellukset toimivat matalalla tasolla.

Sovelluskehyksestä pidättäytyminen syntyi halusta tehdä asiat mahdollisimman yksinkertaisesti – kaikki systeemin tuotu monimutkaisuutta lisäävä tekijä piti perustella jonkin tiedossa olevan ongelman ratkaisuna. Mitään ongelmaa, joka olisi perustellut sovel-

luskehyksen käytön, oli projektin alkuvaiheessa vaikea nähdä. Ongelmat ilmenivät vasta myöhemmin, sovelluksen laajetessa ja monimutkaistuessa. Jälkeenpäin ajateltuna sovelluskehyksen tai korkeamman tason ongelmia ratkaisevien lisäkirjastojen käyttö on täysin perusteltua.

6.3 Projektin tulevaisuus

Koodin siirtämistä jollekin valmiille sovelluskehykselle tai ainakin joidenkin monimutkaisimpien osa-alueiden siirtämistä valmiiden kirjastojen vastuulle on harkittu. Koodin muuntaminen veisi tietysti lisää aikaa, mutta pitkällä aikavälillä siitä olisi varmasti hyötyä. Otetaan kuvitteellinen tilanne: käyttöliittymän näkymät siirretään React-kirjaston vastuulle. Työ, sisältäen React-kirjaston käytön opiskelun, voisi viedä vaikka 100 tuntia (2-3 työviikkoa). Muutosten tekeminen ja koodin ylläpitäminen myöhemmin kuitenkin helpottuisi huomattavasti. React hoitaa automaattisesti näkymien datan päivityksen – työn, joka varsinkin monimutkaisissa näkymissä on nykyisellään sekä virhealtista että työlästä. Toinen huomattavasti nopeampi ja helpompi muutos olisi koko CSS-koodin yhtenäistäminen ja muuttaminen fiksummaksi Less-koodiksi, mikä onkin ollut tarkoituksena jo projektin alkupäivistä lähtien. Siihen menisi arviolta 1-2 työpäivää, riippuen vähän siitä, kuinka paljon asioita yhtenäistettäisiin.

Toimeksiantajan kanssa on alustavasti puhuttu projektin jatkamisesta, mutta mitään ei ole toistaiseksi vahvistettu. Sovellus kuitenkin nykymuodossaan vaatii vielä kehitystä. Tekninen toteutus pitäisi saada valmiiksi: loput kriittiset ominaisuudet pitäisi toteuttaa ja esimerkiksi tarkemmin selvittää, miten palvelin on yhteydessä Virtuaalikonttorin kanssa. Kun sovellus on kehitetty sellaiseen pisteeseen, että sitä voi testata tositoimissa, voidaan alkaa miettimään kaikkia uusia ominaisuuksia, jotka olivat tavoitteena jo projektin alkuvaiheessa. Silloin olisi myös otollinen hetki käytettävyystestaukseen tai ainakin jonkinlaisen palautteen keräämiseen.

LÄHTEET

Angular JS. HTML enhanced for web apps!. Luettu 29.8.2016. https://angularjs.org/

Galitz, W. O. 2007. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. 3. painos. USA: Wiley.

Garrett, J. J. 2005. Ajax: A New Approach to Web Applications. Luettu 19.2.2016. http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

Holzner, S. 2007. Ajax Bible. 1. painos. USA: Wiley.

Johnson, J. 2010. Designing with the mind in mind: simple guide to understanding user interface design rules. 1. painos. Lontoo: Morgan Kaufmann cop.

jQuery. What is jQuery?. Luettu 16.3.2016. http://jquery.com/

jQuery UI. About jQuery UI. Luettu 16.3.2016. http://jqueryui.com/about/

Less. Getting started. Luettu 29.8.2016. http://lesscss.org/

Lidwell, W., Elam, K., Butler, J. & Holden, K. 2010. Universal Principles Of Design: 125 Ways To Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, And Teach Through Design. 2. painos. USA: Rockport Publishers.

Microsoft. Guidelines for targeting. Luettu 29.2.2016. https://msdn.microsoft.com/en-us/library/windows/apps/Hh465326.aspx

Mikowski, M. S. & Powell, J. C. 2013. Single Page Web Applications: JavaScript end-to-end. 1. painos. USA: Manning Publications.

Nielsen J. 1995. 10 Usability Heuristics for User Interface Design. https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen J. 2012. Usability 101: Introduction to Usability. Luettu 23.8.2016. https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Osmani, A. 2015. Learning JavaScript Design Patterns. Luettu 15.3.2016. https://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternjavascript

Otero, C., Gorkov, A. & Larsen, R. 2012. Professional JQuery. 1. painos. USA: Wrox.

React. React – A JavaScript library for building user interfaces. Luettu 29.8.2016. https://facebook.github.io/react/

Rubin, J. & Chisnell, D. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. 2. painos USA: Wiley.

Scoresby J. 2008. Luettu 25.8.2016. http://www.zdnet.com/article/how-to-design-a-system-that-everybody-hates/

Sencha. Sencha Touch. Luettu 4.9.2016. https://www.sencha.com/products/touch/#overview

W3C. 2016. CSS Flexible Box Layout Module Level 1. Luettu 17.3.2016. https://www.w3.org/TR/css-flexbox-1/

Whitenton K. 2013. Minimize Cognitive Load to Maximize Usability. Luettu 29.2.2016. https://www.nngroup.com/articles/minimize-cognitive-load/