OULUN AMMATTIKORKEAKOULU

HINTASEURANTASOVELLUS PÄIVITTÄISTAVARAOSTOKSIIN

Käytettävyyden ja toiminnallisuuksien parantaminen

Milla Viitapohja
Opinnäytetyö (AMK)
Kevät 2025
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Milla Viitapohja
Opinnäytetyön otsikko: Hintaseurantasovellus päivittäistavaraostoksiin
Työn ohjaaja: Pekka Ojala
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 39

Tämän opinnäytetyön tavoitteena oli kartoittaa olemassa olevan hintaseurantasovelluksen nykytilaa käytettävyyden ja toiminnallisuuden näkökulmista sekä arvioida kehitystarpeita ja esittää parannusehdotuksia.

Teoriaosuudessa käsiteltiin hintavertailun hyötyjä ja haasteita sekä käyttäjäkeskeisen suunnittelun, käytettävyyden ja käyttäjäkokemuksen periaatteita. Näiden pohjalta analysoitiin sovelluksen toiminnallisuuksia ja käytettävyyttä. Työssä tunnistettiin useita kehityskohteita, joista tarkemmin tarkasteltiin verkkosivujen tietojen automaattista keräämistä eli web scrapingia.

Työn aikana aloitettiin hintatietojen hakua automatisoivan scraperin suunnittelu ja alustava toteutus Node.js-ympäristössä Puppeteer-kirjastoa hyödyntäen.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Business Information Systems

Author: Milla Viitapohja
Title of thesis: Price tracking application for grocery shopping
Supervisor: Pekka Ojala
Term and year when the thesis was submitted: Spring 2025
Number of pages: 39

The aim of this thesis was to examine the current state of an existing price tracking application and evaluate its usefulness in terms of usability and functionality.

The theoretical part discusses the benefits and challenges of price comparison, as well as the principles of user-centered design, usability and user experience. Based on these perspectives, the application's features and functionality were analyzed. Several development needs were identified, with particular focus on the automatic retrieval of price data from websites, also known as web scraping.

The design and initial implementation of a scraper for automating price data retrieval was started using the Node.js environment and the Puppeteer library.

SISÄLLYS

1 JOHDANTO

Hintojen vertaileminen on noussut entistä tärkeämmäksi osaksi kuluttajien arkea erityisesti viime vuosina, kun elinkustannukset ovat kasvaneet ja taloustilanne on muuttunut epävarmemmaksi. Kuluttajat pyrkivät seuraamaan omia menojaan tarkemmin ja tekemään harkitumpia ostopäätöksiä arjessa.

Opinnäytetyön tavoitteena on tarkastella olemassa olevan hintaseurantasovelluksen nykytilaa käytettävyyden ja toiminnallisuuksien näkökulmista sekä pohtia kehityskohteita. Työ tarjoaa sekä teoreettisen että käytännön näkökulman siihen, miten tällainen työkalu voi tukea kuluttajia arjen taloudenhallinnassa. Tuloksia voidaan hyödyntää sovelluksen jatkokehityksessä sekä mahdollisesti laajemminkin vastaavanlaisten sovellusten suunnittelussa.

Keskeisenä tarkastelun kohteena on hintatietojen automaattinen kerääminen verkkokauppojen sivuilta, eli web scraping. Työssä perehdytään siihen, miten verkkosivuilta voidaan kerätä dataa ohjelmallisesti sekä millaisia teknisiä, juridisia ja eettisiä kysymyksiä tähän liittyy. Lisäksi käsitellään vaihtoehtoisia toteutustapoja, kuten rajapintojen hyödyntämistä.

Opinnäytetyö esittelee konkreettisia esimerkkejä tiedonkeruun toteutuksesta Node.js-ympäristössä Puppeteer-kirjastoa hyödyntäen. Tarkoituksena on luoda pohja jatkokehitykselle, jossa hintatiedon keruu voidaan integroida osaksi sovelluksen käyttöliittymää.

2 HINTASEURANNAN TARVE

Hintatietoisuus on viime vuosina noussut yhä tärkeämmäksi osaksi kuluttajien arkea. Elinkustannusten nousu ja markkinoiden epävakaus ovat lisänneet tarvetta seurata omia menoja tarkemmin ja tehdä harkitumpia ostopäätöksiä.

Erityisesti arjen perusmenoihin, kuten ruokaan kohdistuvat hinnannousut ovat lisänneet hintatietoisuutta niin kansainvälisesti kuin Suomessa. Capgeminin vuonna 2022 toteuttaman globaalin kyselytutkimuksen mukaan noin 40 % vastaajista oli vähentänyt ruokamenojaan elinkustannusten nousun vuoksi (Capgemini 2023, 20). Samansuuntaisia tuloksia löytyi myös Suomen Kuluttajaliiton kyselystä, jossa noin puolet osallistuneista ilmoitti vähentäneensä ruokamenojaan hintojen nousun vuoksi vuonna 2022 (Kuluttajaliitto 10.11.2022). Taloudelliset paineet ovat siis pakottaneet kuluttajia pohtimaan kulutustottumuksiaan entistä tarkemmin.

Vaikka Tilastokeskuksen (19.2.2024) mukaan, inflaatio on palautunut jo maltilliselle tasolle, ei ole vielä selvää palautuvatko ihmisten kulutustottumukset entiselleen vai ovatko ne muuttuneet pysyvämmin.

Jo aikaisemmat tutkimukset ovat tosin osoittaneet, että hinta voi vaikuttaa olennaisesti siihen missä kaupassa kuluttajat asioivat ja mitä tuotteita he valitsevat (Monroe & Lee 1999; Binkley & Bejnarowicz 2003). Suomessa hinnan merkityksen kasvua ruokavalinnoissa on seurattu kyselytutkimuksilla (Piiroinen & Järvelä 2006; Peltoniemi & Yrjölä 2012). Peltoniemen ja Yrjölän mukaan hinta on noussut nykypäivänä lähes yhtä tärkeäksi kriteeriksi kuin terveellisyys ja vaikuttaakin nykyisin ruoan valintapäätöksiin selvästi aiempaa enemmän. Tätä tukee myös vuonna 2023 toteutettu suomalainen tutkimus, jossa jopa 72 prosenttia vastaajista nimesi hinnan keskeiseksi tekijäksi ostopäätöksessä ja moni kertoi valitsevansa ruokakaupan aiempaa useammin edullisempien hintojen perusteella (Forsman-Hugg, Kinnunen & Yli-Liipola 2024, 15–16). Myös vuoden 2020 Kuluttajaliiton kyselyssä 58 % vastaajista ilmoitti vertailevansa hintoja eri kauppojen välillä ainakin joskus (Kuluttajaliitto 15.11.2020). Hintojen merkitys on siis selvästi

kasvanut suomalaisten keskuudessa ja kuluttajat hyödyntävät yhä aktiivisemmin erilaisia hintaseurantakeinoja arjessaan.

2.1 Hintavertailun hyödyt

Hintaseurannan merkitys korostuu erityisesti päivittäistavaroiden, kuten ruoan, ja muiden arjen välttämättömyystarvikkeiden kohdalla. Nämä ostokset voivat muodostavat suuren osan kotitalouden kuukausittaisista menoista ja hintojen tarkkailu voikin tarjota konkreettisia säästömahdollisuuksia. Esimerkiksi saman tuotteen hinta voi vaihdella huomattavasti eri myymälöiden välillä (Juntunen 25.11.2024) ja pienetkin erot hinnoissa voivat pitkässä juoksussa johtaa merkittäviin säästöihin. Hintojen vertailun onkin myös arvioitu lisäävän kuluttajien tunnetta kontrollista ostopäätöksissä (Kwarteng, Jibril, Botha & Osakwe 2020).

Hintatietoisuus ei tosin auta ainoastaan säästämään rahaa vaan se voi myös tukea harkitumpia ostopäätöksiä ja vähentää heräteostoksia. Esimerkiksi alennusmyyntien on tutkittu lisäävän impulsiivista ostokäyttäytymistä (Zhou & Wong 2004), mutta kuluttajat, jotka tuntevat tuotteiden normaalihintatasot, osaavat arvioida onko tarjous oikeasti edullisempi verrattuna hintaan toisessa kaupassa. Näin kuluttaja pystyy myös hyödyntämään aidosti edulliset tarjoukset entistä tehokkaammin.

Vaikka kansainvälisissä tutkimuksissa (Martin 2020; Lindgren, Daunfeldt & Rudholm 2021) on havaittu, että hintavertailusivustot parantavat markkinoiden tehokkuutta ja kilpailua kuluttajien hyväksi, suomalaisessa päivittäistavaramarkkinoilla vaikutukset voivat olla rajallisempia. Markkinoita hallitsee käytännössä kaksi suurta toimijaa, K-ryhmä ja S-ryhmä (NielsenIQ 26.3.2025), jotka vaikuttavat jo seuraavan aktiivisesti toistensa hinnoittelua pyrkiessään säilyttämään markkinaosuutensa. Näin ollen hintavertailusivustojen käytön hyödyt painottuvat Suomen päivittäistavaramarkkinoilla todennäköisemmin yksittäisten kuluttajien säästöihin kuin koko markkinarakenteeseen.

Hintaseuranta voi ohjata ostokäyttäytymistä myös vastuullisempaan suuntaan tukien samalla kestävää kulutusta. Hintatietoisuus ei nimittäin tarkoita välttämättä

vain halvimman vaihtoehdon valitsemista vaan se voi liittyä myös esimerkiksi hinta-laatusuhteen arviointiin (Lichtenstein, Ridgway & Netemeyer 1993).

2.2 Hintaseurannan haasteet käytännössä

Vaikka digitaalisten työkalujen ansiosta hintojen vertailu onkin nopeampaa ja vaivattomampaa kuin koskaan aiemmin, monet kuluttajat voivat silti kokea hintaseurannan työläänä. Kauppojen omat mobiilisovellukset ja verkkopalvelut saattavat tarjota apua, mutta kuluttajan kannalta keskeinen ongelma on tiedon hajanaisuus. Hinnat eivät ole keskitetysti saatavilla ja osa kaupoista ei julkaise hintojaan ollenkaan verkossa.

Hintavertailua monimutkaistaa myös päivittäistavarakauppojen kanta-asiakasohjelmat. Usein näissä järjestelmissä bonuksia kertyy suhteessa kulutukseen: mitä enemmän tekee ostoksia, sitä enemmän saa etuja. Tällaisia etujärjestelmiä on vaikea ottaa huomioon vertaillessa hintoja, mikä tekee vertailusta väistämättä epätarkempaa. Vaikka tuotteen hinta olisikin korkeampi toisessa kaupassa, voi kuluttaja hyötyä toisen kaupan bonusten tai alennusten kautta enemmän.

Manuaalinen hintaseuranta, jossa kuluttaja kirjaa tuotteiden hintoja itse esimerkiksi puhelimen muistioon, on nykypäivänä melko vanhanaikainen ja aikaa vievä tapa seurata hintoja. Hintojen päivittäminen vaatii säännöllistä kaupassa käyntiä ja hintojen tarkkailua, mikä voi olla etenkin suurten tuotemäärien kohdalla työlästä. Päivittäistavaroiden hinnat vaihtelevat usein, jolloin myös tiedot vanhenevat nopeasti ja erilaiset tarjoukset ja kampanjat tuovat oman haasteensa ajantasaisen hintatiedon keruuseen. Lisäksi yksittäisten tuotteiden hintaerot voivat tuntua pieniltä ja hyöty suhteessa vaivaan vähäiseltä. Digitaaliset ratkaisut ovatkin monin tavoin syrjäyttäneet tällaisen manuaalisen lähestymistavan.

3 KÄYTTÄJÄKESKEINEN SUUNNITTELU

Käyttäjäkeskeinen suunnittelu (User-Centered Design, UCD) on lähestymistapa, jossa tuotteen suunnittelun keskiössä ovat sen todelliset käyttäjät ja heidän tarpeensa (Steckiw 30.1.2022). Joissakin lähteissä se esiintyy synonyyminä ihmiskeskeisen suunnittelun (Human-Centered Design, HCD) kanssa, vaikka käsitteiden painotukset poikkeavatkin hieman toisistaan. Käyttäjäkeskeinen suunnittelu keskittyy ensisijaisesti loppukäyttäjien tarpeisiin, kun taas ihmiskeskeinen suunnittelu huomioi laajemmin myös muut sidosryhmät. (ISO 9241-210:2019, 3.7.)

Käyttäjäkeskeisen suunnittelun tavoitteena on kehittää ratkaisuja, jotka vastaavat käyttäjien odotuksia ja toimintatapoja mahdollisimman luonnollisella ja vaivattomalla tavalla. Se keskittyy käyttäjien tarpeiden ja tavoitteiden ymmärtämiseen, heidän mieltymystensä huomioimiseen sekä jatkuvaan testaamiseen ja palautteen keräämiseen (Digitaalisen Markkinoinnin Sanakirja s.a.).

Yleensä suunnittelu jaetaan neljään eri vaiheeseen, joita toistetaan iteratiivisesti, kunnes lopputulos on tyydyttävä (ISO 9241-210:2019; Interaction Design Foundation 2016a). Aluksi pohditaan käyttökontekstia, eli missä ja miten käyttäjät käyttävät tuotetta tai palvelua. Tämän jälkeen määritellään käyttäjien tarpeet, tavoitteet ja rajoitteet. Seuraavaksi suunnitellaan ja toteutetaan ratkaisut näiden tietojen pohjalta. Lopuksi arvioidaan, täyttääkö toteutus käyttäjien vaatimukset ja tehdään tarvittaessa parannuksia.

Käyttäjäkeskeisen suunnittelun yhteydessä tarkastellaan usein myös siihen liittyviä käsitteitä, kuten käytettävyys, käyttöliittymäsuunnittelu ja käyttäjäkokemus. Nämä muodostavat perustan sovellusten laadukkaalle suunnittelulle ja tukevat käyttäjien tarpeiden huomioimista.

3.1 Käytettävyys ja käyttöliittymäsuunnittelu

Käytettävyys määritellään sen mukaan, kuinka hyvin käyttäjät voivat käyttää tuotetta, järjestelmää tai palvelua tietyssä käyttötilanteessa saavuttaakseen

tavoitteensa tehokkaasti, vaivattomasti ja tyytyväisinä (ISO 9241-11:2018, 3.1.1). Käytettävyys siis viittaa siihen, kuinka helposti käyttäjä pystyy suorittamaan haluamiaan toimintoja käyttäessään palvelua tai tuotetta. Käytettävyys on osa laajempaa käyttäjäkokemuksen kokonaisuutta, mutta käytettävyys keskittyy erityisesti siihen, että tuote toimii mahdollisimman hyvin käyttäjälle. (Digital.gov 26.2.2025.)

Käyttöliittymä (User Interface, UI) muodostuu kaikista niistä vuorovaikutteisen järjestelmän osista, jotka tarjoavat käyttäjälle tietoa ja mahdollisuuden ohjata järjestelmää tehtävien suorittamiseksi (ISO 9241-110:2020, 3.10). Käyttöliittymäsuunnittelu on keskeinen osa käytettävyyttä, sillä se määrittää, miten käyttäjä kommunikoi järjestelmän kanssa. Hyvin suunniteltu käyttöliittymä tukee käyttäjää ja mahdollistaa tehtävien sujuvan suorittamisen ilman ulkopuolista apua tai erityistietämystä. Suunnittelijan on ymmärrettävä käyttäjien tavoitteet ja mahdolliset rajoitteet voidakseen luoda toimivan käyttöliittymän (Interaction Design Foundation 2016b.)

Jakob Nielsen, tunnettu käytettävyyden asiantuntija ja heuristisen arviointimenetelmän kehittäjä, on esittänyt kymmenen yleistä käytettävyyden periaatetta, jotka auttavat hyvän käyttöliittymän suunnittelussa (Nielsen 1994, 156):

1. **Yksinkertainen suunnittelu.** Käyttöliittymässä tulisi esittää vain olennainen tieto. Kaikki ylimääräinen tai harvoin tarvittava sisältö vie huomiota tärkeiltä asioilta ja heikentää niiden erottuvuutta.
2. **Käyttäjän kielen käyttäminen.** Käyttöliittymän tulee käyttää käyttäjälle tuttuja sanoja, ilmauksia ja käsitteitä, eikä teknistä terminologiaa.
3. **Käyttäjän muistin kuormituksen minimointi.** Käyttäjän ei tule joutua muistamaan tietoa eri käyttöliittymän osien välillä, vaan tarvittavan tiedon tulee olla aina näkyvissä tai helposti haettavissa silloin, kun sitä tarvitaan. Esimerkiksi käyttöliittymän elementit, toiminnot ja valinnat täytyy tehdä näkyviksi käyttäjälle.
4. **Johdonmukaisuus.** Käyttäjien ei tulisi joutua arvailemaan, tarkoittavatko eri sanat, tilanteet tai toiminnot samaa asiaa. Käyttöliittymän tulee olla yhdenmukainen.

5. **Palaute.** Järjestelmän tulisi aina pitää käyttäjä ajan tasalla toiminnan etenemisestä tarjoamalla selkeää ja ajankohtaista palautetta.
6. **Selkeästi merkitty poistumisvaihtoehto.** Käyttäjät tekevät usein virheitä ja tarvitsevat selkeän poistumisvaihtoehdon päästäkseen pois ei-toivotusta tilasta.
7. **Pikanäppäimet ja nopeuttajat.** Kokeneemmille käyttäjille voidaan tarjota näkymättömiä pikanäppäimiä tai muita nopeuttajia, jotka nopeuttavat käyttöä ja palvelevat eri taitotasoja.
8. **Hyvät virheilmoitukset.** Virheilmoitusten tulee olla selkokielisiä, ilmaista ongelma tarkasti ja antaa rakentavia ratkaisuja.
9. **Virheiden ehkäisy.** Parasta on suunnitella järjestelmä niin, että virheet estetään jo ennen kuin ne tapahtuvat.
10. **Ohjeistus ja dokumentaatio.** Vaikka järjestelmän tulisi olla käytettävissä ilman ohjeita, on usein tarpeen tarjota helposti haettavaa ja tehtäväsuuntautunutta ohjeistusta käyttäjille.

Miellyttävä käytettävyys on tärkeä, sillä jos käyttäjät eivät saavuta palvelun avulla haluamiaan tavoitteita, he hylkäävät sen ja etsivät uuden tilalle. Hyvä käytettävyys onkin olennainen edellytys onnistuneelle käyttökokemukselle. (Aliskan 5.3.2021.)

3.2 Käyttäjäkokemus

Käyttäjäkokemus (User Experience, UX) tarkoittaa käyttäjän havaintoja ja reaktioita, jotka syntyvät järjestelmän, tuotteen tai palvelun käytöstä tai odotetusta käytöstä (ISO 9241-11:2018, 3.2.3). Käyttäjäkokemus kattaa kaikki käyttäjän tuntemukset ja arviot tuotteesta tai palvelusta. Eli tunteet, jotka liittyvät muun muassa käytön helppouteen, saavutettavuuteen, visuaaliseen ilmeeseen, käyttöliittymän toiminnallisuuteen sekä yleiseen käyttöön. (Finn & Downie 10.1.2025.)

Suunnittelijat eivät voi täysin kontrolloida käyttäjien kokemuksia tai reaktioita tuotteen käytön aikana. Sen sijaan he voivat vaikuttaa siihen, miten tuote, järjestelmä tai palvelu toimii ja miltä se näyttää. (Interaction Design Foundation 2016c.)

4 SOVELLUKSEN NYKYTILA

Tässä luvussa käsitellään kehitettävän hintaseurantatyökalun tämänhetkinen tila. Sovelluksen kehittäminen sai alkunsa henkilökohtaisesta tarpeestani: halusin yksinkertaisen tavan kirjata ja vertailla päivittäistavaroiden hintoja eri kaupoissa, jotta voisin seurata mistä haluamani tuotteet saa edullisimmin. Kehitysprosessin lähtökohtaan liittyi siis myös käyttäjälähtöistä suunnittelua koska sovellusta lähdettiin kehittämään aidon käyttäjätarpeen pohjalta.

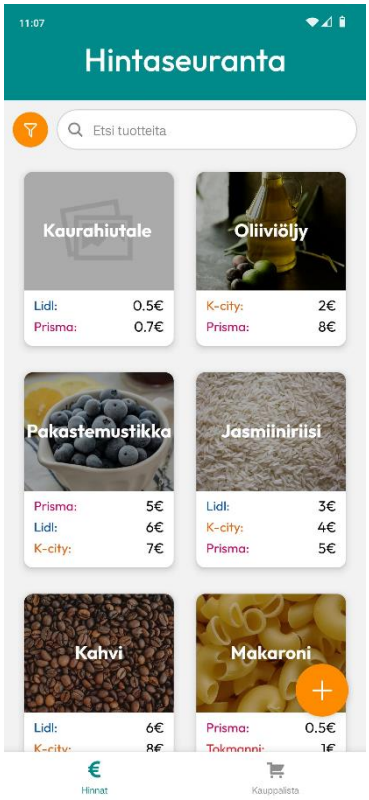
Päädyin kehittämään nimenomaan manuaaliseen tiedonkeruuseen perustuvaa sovellusta, koska yksi säännöllisesti käyttämäni päivittäistavarakauppa ei tarjoa hintatietojaan verkossa. Olin jo tottunut kirjaamaan tuotteiden hintoja manuaalisesti, mutta kaipasin tehokkaampaa tapaa tallentaa ja erityisesti seurata kertyneitä hintatietoja. Hintamerkintöjen määrän kasvaessa niiden selaaminen ja hyödyntäminen alkoi käydä hankalaksi. Tietoisuus manuaalisen seurannan rajoituksista, kuten tietojen nopea vanheneminen, on silti tärkeää sovelluksen jatkokehityksen kannalta.

Alaluvuissa kuvataan sovelluksen keskeiset toiminnot ja ominaisuudet, käyttöliittymän rakenne ja käytettävyyteen liittyvät ratkaisut sekä tekninen toteutus. Sovelluksen koodirakennetta ei kuitenkaan käsitellä yksityiskohtaisesti, sillä se ei ollut tämän työn kannalta oleellista. Tarkoituksena on antaa kokonaiskuva siitä, millainen sovellus tällä hetkellä on ja miten se toimii.

4.1 Toiminnallisuudet

Sovelluksen nykyiset ominaisuudet on suunniteltu tukemaan käyttäjän omaehtoista hintatiedon keruuta, vertailua ja hallintaa. Käyttöliittymä pyrkii tarjoamaan yksinkertaisen, mutta toimivan ratkaisun manuaaliseen hintaseurantaan. Sovellus sisältää myös yksinkertaisen ostoslistan, joka auttaa arjen ostosten suunnittelussa.

Kuvassa 1 näkyy sovelluksen aloitusnäkymä, jossa on lista tallennetuista tuotteista ja niiden hintatiedoista. Ulkoasu on hyvin minimalistinen, jotta tärkein sisältö eli hintatiedot erottuvat helposti. Tästä päänäkymästä käyttäjä näkee nopeasti tuotteiden hinnat eri kaupoissa helposti vertailtavassa muodossa ja voi siirtyä muokkaamaan olemassa olevia tuotetietoja tai lisäämään uusia. Ruudussa on myös hakupalkki ja suodatinpainike, mutta ne eivät ole vielä toiminnassa, sillä niiden toteutus ei ollut ensisijainen prioriteetti. Alareunassa sijaitsevan valikkopalkin avulla käyttäjä pystyy navigoimaan eri näkymien välillä ja se osoittaa selkeästi millä sovelluksen sivuista käyttäjä on.



Kuva 1. Sovelluksen aloitusnäkymä

Uuden tuotteen lisääminen tapahtuu aloitusnäkymän alakulmassa olevan painikkeen avulla, mikä avaa näkymän tuotetietojen syöttämistä varten (kuva 2). Käyttäjä voi lisätä halutessaan tuotteen kuvan syöttämällä kuvan URL-osoitteen, jonka jälkeen hän kirjoittaa tuotteen nimen sekä valitsee valmiista vaihtoehdoista sen kaupan, jolle haluaa hinnan lisätä. Tuotenimi on ainoa pakollinen kenttä, joten käyttäjä voi halutessaan jättää kuvan ja hintatiedot aluksi tyhjiksi. Hintatietoja

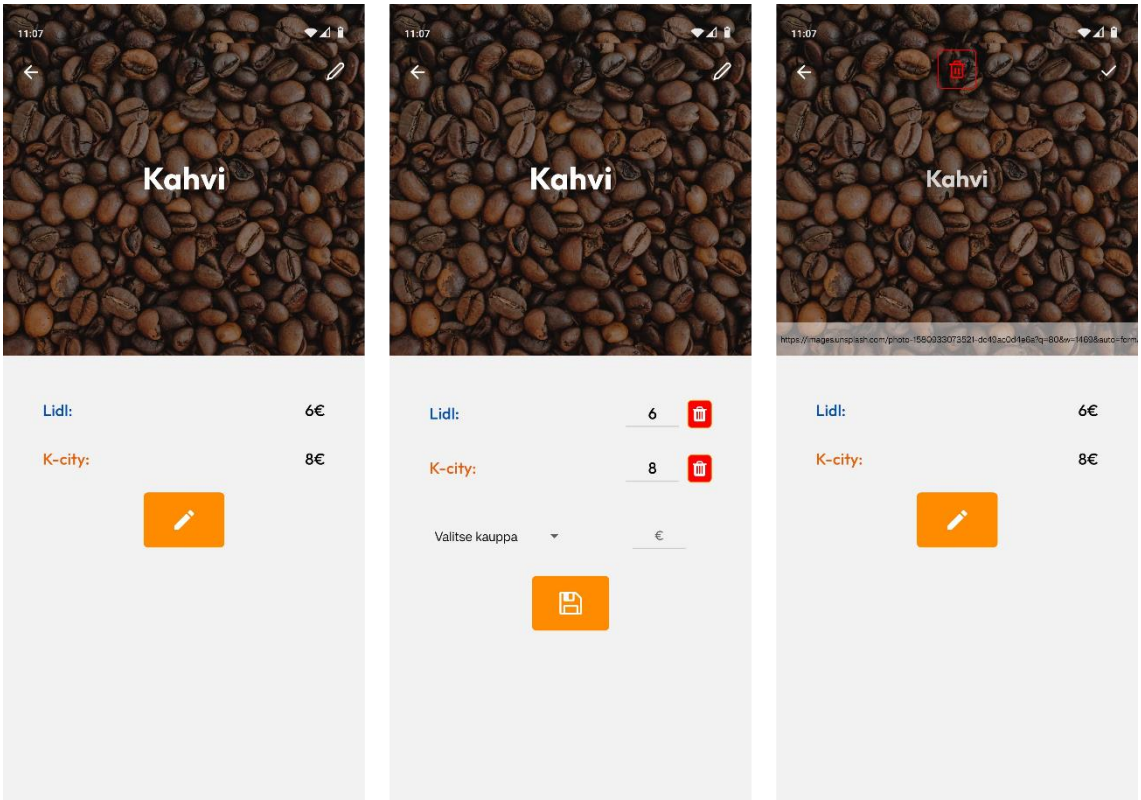
voi myös lisätä useammalle kaupalle samalla kertaa. Sivun yläreunan nuolipainikkeella tai järjestelmänavigaatiolla voi palata aloitusnäkymään ilman tallennusta. Jos käyttäjä on kuitenkin syöttänyt tietoja, järjestelmä kysyy ennen poistumista, tallennetaanko tuote. Mikäli pakollinen nimikenttä on tyhjä, näytetään virheilmoitus eikä tallennus onnistu ennen kuin nimi on lisätty.

Kuva 2. Uuden tuotteen lisääminen

Käyttäjä voi siirtyä muokkaamaan tuotetietoja painamalla aloitusnäkymässä haluamaansa tuotetta. Tuotesivulla (kuva 3) käyttäjä voi muokata erikseen joko tuotteen perustietoja tai kauppakohtaisia hintoja. Muokkaustoiminnot on tarkoituksella jaettu kahteen erilliseen painikkeeseen, jotta usein päivitettävien hintatietojen muokkaus olisi mahdollisimman sujuvaa, mutta harvemmin muutettavat tuotetiedot pysyisivät suojassa tahattomilta muokkauksilta. Käyttäjä voi myös koska tahansa poistua muokkausnäkymästä ilman muutosten tallentamista painamalla takaisin-painiketta puhelimen järjestelmänavigaatiosta. Poistotoiminnoissa on myös vahvistusvaihe, jossa poisto varmistetaan ponnahdusikkunalla vahingossa tapahtuvien poistojen estämiseksi.

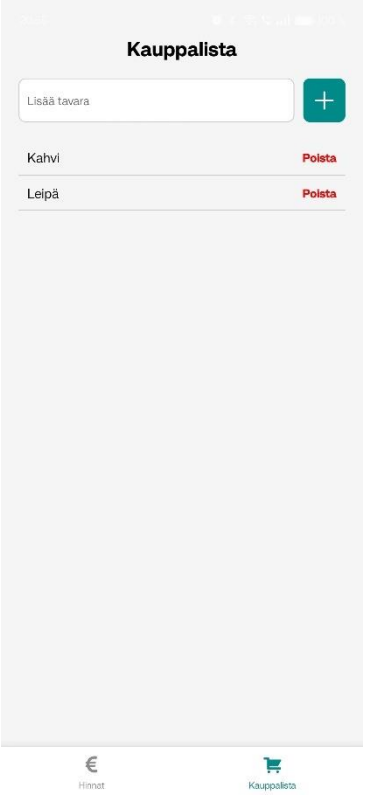
Koska hintatietojen muokkaus on yleisempää, siihen liittyvä painike on myös sijoitettu näkyvästi näytön keskiosaan, hintatietojen alapuolelle, jotta se olisi helposti käyttäjän ulottuvilla. Muokkausnäkymässä käyttäjä voi päivittää olemassa olevia hintoja, poistaa yksittäisen kaupan tiedot tai lisätä uuden kaupan tuotteelle. Uutta kauppaa ei kuitenkaan tallenneta, ellei siihen ole lisätty hintatietoa. Kaupat järjestyvät automaattisesti niin, että edullisin on aina ensimmäisenä. Tämä pätee myös aloitusnäkymän tuotekortteihin.

Tuotetietoja sen sijaan tarvitsee harvemmin muuttaa, joten niiden muokkauspainike on sijoitettu yläkulmaan, missä sitä ei tule painettua vahingossa tai turhaan. Painikkeen kautta voi muokata tuotenimeä ja kuvaa tai poistaa koko tuotteen, jolloin myös siihen liittyvät hintatiedot poistetaan sovelluksesta pysyvästi.



Kuva 3. Tuotesivu ja tietojen muokkaus

Sovelluksen toinen keskeinen toiminto on kauppalista (kuva 4), joka toimii yksinkertaisena muistilistana ostoksia varten. Käyttäjä voi lisätä listalle tuotteita ja poistaa niitä ostosten edetessä. Kauppalista ei ole yhteydessä hintatietoihin, vaan toimii erillisenä työkaluna ostosten suunnittelun tueksi.



Kuva 4. Kauppalista

Sovellus on toiminnallisuuksiltaan vielä yksinkertainen, mutta toimivat perusominaisuudet tarjoavat kuitenkin hyödyllisen työkalun hintatietojen seuraamiseen sekä ostosten suunnitteluun. Sovelluksen toiminta perustuu tällä hetkellä täysin käyttäjän omatoimisuuteen eli käyttäjällä on täysi kontrolli sisällöstä, mutta samalla vastuu sen täydentämisestä.

4.2 Käyttöliittymä ja käytettävyys

Sovelluksen käyttöliittymän suunnittelussa on hyödynnetty useita keskeisiä käytettävyyden periaatteita. Se on yritetty suunnitella selkeäksi ja yksinkertaiseksi, mutta samalla kuitenkin visuaalisesti houkuttelevaksi. Suunnittelussa on pyritty pitämään keskiössä helppokäyttöisyys, jotta käyttäjät voivat suorittaa tarvittavat toiminnot mahdollisimman vaivattomasti ja nopeasti. Turhia valikkotasoja on vältetty, jotta käyttö pysyy sujuvana ja toiminnot löytyvät nopeasti. Toiminnot

löytyvät pääosin yhden tai kahden painalluksen takaa, mikä nopeuttaa toistuvaa käyttöä.

Käyttöliittymässä on otettu huomioon useita Androidin suunnitteluperiaatteita, kuten turvallisuusalueiden huomioiminen, elementtien johdonmukainen sijoittelu ja välimatkojen yhdenmukaisuus sekä responsiivisuus eri näyttökokoihin ja dynaamiseen sisältöön (Android Developers 20.12.2024). Suurin osa sovelluksen sisällöstä on sijoitettu näytön keskiosaan, kun taas harvemmin käytettävät toiminnot ovat näytön yläosassa. Lisäksi painikkeissa on käytetty helposti ymmärrettäviä kuvakkeita, jotta käyttäjä hahmottaa vaivattomasti niiden tarkoituksen.

Järjestelmä myös antaa käyttäjälle selkeää palautetta eri toiminnoista. Esimerkiksi uuden tuotteen tallentamisen tai tietojen muokkauksen jälkeen näytetään vahvistusviesti. Mikäli tietojen tallennus epäonnistuu, käyttäjälle ilmoitetaan virheestä selkeästi.

Sovelluksen visuaalinen ilme perustuu huolellisesti valittuun värimaailmaan ja typografiaan. Käytetyt värit eli tumma turkoosi ja tangeriinin sävy, on valittu esteettisen miellyttävyyden perusteella ja niiden visuaalinen yhteensopivuus on varmistettu Figman väripaleteilla. Turkoosi luo vakaan ja luotettavan tunnelman (Piktochart 4.9.2024), kun taas oranssi tuo piristystä ja korostaa tärkeitä toimintoja. Yhdessä ne muodostavat rauhoittavan ja kutsuvan värimaailman (Adobe s.a). Fontti on Outfit, joka on valittu sen selkeyden, luettavuuden ja nykyaikaisen ilmeen vuoksi. Lisäksi on kiinnitetty huomiota fonttikokoihin ja värien kontrasteihin, jotta teksti on helposti luettavissa eri tilanteissa ja laitteilla.

Käytettävyyttä on mietitty arjen sujuvuuden näkökulmasta. Sovelluksen käyttö ei vaadi syvällistä opettelua, ja se on pyritty suunnittelemaan niin, että tärkeimpiä toiminnallisuuksia voisi käyttää helposti yhdellä kädellä. Esimerkiksi tuotteiden nopea lisääminen ja muokkaaminen on tehty mahdollisimman vaivattomaksi, jotta lisäykset ja muutokset saisi helposti tallennettua heti niiden tullessa esiin. Käytettävyyttä kuitenkin heikentävät näkyvillä olevat elementit, jotka eivät toimi vielä, kuten hakupalkki ja suodatinpainike. Lisäksi on syytä pohtia, aiheuttaako esimerkiksi kahden muokkauspainikkeen esiintyminen tuotesivulla epäselvyyttä.

Vaikka ratkaisu on perusteltu toiminnallisuuden kannalta, nappien toimintojen selkeyttä voisi mahdollisesti vielä parantaa.

4.3 Kehitysalusta ja -teknologiat

Sovellus on rakennettu React Native -teknologialla, joka on Metan kehittämä avoimen lähdekoodin framework vuodelta 2015. React Native tuo Reactin ohjelmointimallin mobiilialustoille ja mahdollistaa sovellusten kehittämisen JavaScriptillä. (React Native s.a.) Se hyödyntää niin sanottuja natiivikomponentteja, eli järjestelmäkohtaisia käyttöliittymäelementtejä, joiden ansiosta sovellus näyttää ja toimii luonnollisesti sekä Android- että iOS-laitteilla. Vaikka React Native mahdollistaa saman koodipohjan käytön eri alustoilla, se ei itsessään tarjoa valmista kehitysympäristöä esimerkiksi iOS-sovellusten rakentamiseen Windows-käyttöjärjestelmässä, eikä sisällä kaikkia käytännön sovelluskehityksen tarvitsemia työkaluja valmiina. Tämä voi lisätä kehityksen monimutkaisuutta ja ylläpidon haastavuutta. Siksi React Native -kehityksessä suositellaan käyttämään kehitysalustaa, joka tarjoaa kaikki tarvittavat työkalut sovellusten rakentamiseen. Expo on yksi tällaisista kehysalustoista. (React Native 14.4.2025.)

Kehitystyössä onkin siis hyödynnetty Expoa, joka on React Nativen päälle rakennettu kehitysalusta. Expo helpottaa ja nopeuttaa projektin käynnistämistä ja kehittämistä tarjoamalla valmiita työkaluja ja ominaisuuksia, kuten tiedostopohjaisen reitityksen, universaaleja kirjastoja ja mukautetun natiivikoodin lisäämisen ja muokkaamisen.

Expo tarjoaa kaksi pääasiallista vaihtoehtoa sovelluskehitykseen: Expo Go -ympäristön ja development buildin. Expo Go on hiekkalaatikkoympäristö, joka tarjoaa helpon tavan kehittää ja testata sovelluksia Expo Go -sovelluksessa. Tällöin projekti ei kuitenkaan sisällä natiivitiedostoja ollenkaan, joten käyttäjä ei pysty tekemään muutoksia sovelluksen natiivipuolen asetuksiin. Natiivikoodia sisältävien kirjastojen käyttö on myös hankalampaa, sillä on ensin tarkistettava, että Expo Go -sovellus tukee kyseistä kirjastoa. Development build -vaihtoehdossa sovellus rakennetaan paikallisesti laitteelle tai emulaattoriin, jolloin siihen voidaan

sisällyttää tarvittavat natiivikirjastot ja tehdä muutoksia natiivipuolelle, mutta samalla säilyttää Expon tarjoamat työkalut. (Expo Docs 28.3.2025.)

Kehitysympäristöksi valitsin development buildin Androidille Expo Gon sijaan, sillä tämä valinta tarjoaa enemmän joustavuutta ja hallintaa sovelluksen kehityksessä.

Käytännössä kehitystyössä tarvittiin samoja työkaluja kuin perinteisessä Android-sovelluskehityksessä. Tämä tarkoittaa muun muassa Android SDK:n (SDK, Software Development Kit), Java Development Kitin (JDK), ja halutessaan Android Studion, käyttöönottoa. Android SDK koostuu useista erillisistä paketeista, jotka ovat olennaisia Android-sovellusten kehittämisessä (Android Developers 10.2.2025). JDK taas sisältä välttämättömiä työkaluja Android-sovellusten kokoamiseen.

Kehitysympäristön valinnan taustalla oli ensisijaisesti tarve varmistaa, että ympäristö pysyy johdonmukaisena, eikä tekniset rajoitukset aiheuta ongelmia tulevaisuudessa. Sovelluksesta on tarkoitus tuottaa myöhemmin julkaisuversio (release build), eli itsenäisesti asennettava ja käytettävä sovellustiedosto, kuten Androidin .apk (Android Application Package). Expo Gon käyttäminen kehityksessä olisi saattanut lisätä ylimääräistä monimutkaisuutta tämän vaiheen kohdalla.

4.4 Käytetyt työkalut ja palvelut

Figma on ollut käytössä sovelluksen käyttöliittymän suunnittelussa. Kyseessä on työkalu, joka mahdollistaa käyttöliittymien hahmottelun ja prototyyppien luomisen. Figman avulla pystyttiin etukäteen hahmottamaan näkymien rakennetta ja testaamaan elementtien sijoittelua.

GitHub on toiminut versionhallintajärjestelmänä ja koodin varmuuskopiointipaikkana. GitHub on pilvipohjainen alusta, jonka avulla voi helposti seurata projektin muutoksia, hallita eri versioita ja palata tarvittaessa aiempiin vaiheisiin. Se on rakennettu avoimen lähdekoodin versionhallintajärjestelmä Gitin päälle. (GitHub Docs 13.3.2025.)

Visual Studio Code (VS Code) on ollut pääasiallinen koodieditori koko kehitystyön ajan. Se on monipuolinen editori, joka tukee useita ohjelmointikieliä ja tarjoaa laajasti sisäänrakennettuja ominaisuuksia, kuten integroidun komentopaletin ja sisäänrakennetun versionhallinnan, joka tukee myös Git-versionhallintaa ilman lisäasetuksia. VS Code tarjoaa laajan valikoiman laajennuksia ja tukee saavutettavuutta muun muassa korkeakontrastisilla teemoilla sekä optimoidulla näppäimistönavigoinnilla. Käyttäjä voi myös säätää väriteemoja ja asettelua omien mieltymysten mukaan ja synkronoida omat asetuksensa eri laitteiden välillä. (Visual Studio Code s.a.)

Firebase, tai tarkemmin Googlen Firebase-tuoteperheeseen sisältyvä Cloud Firestore-tietokanta, on toiminut projektin tietokantana. Se on joustava ja skaalautuva NoSQL-pilvitietokanta, joka on suunniteltu mobiili-, web- ja palvelinkehitykseen. Firestore mahdollistaa datan tallentamisen ja synkronoinnin reaaliajassa ja se tukee myös offline-käyttöä, mikä mahdollistaa sujuvan käyttökokemuksen myös ilman verkkoyhteyttä tai viiveellisissä verkoissa. Sen tietomalli tukee joustavia, hierarkkisia rakenteita. Data tallennetaan dokumentteihin, jotka ryhmitellään kokoelmiin. Jokainen dokumentti voi sisältää sekä monimutkaisia, sisäkkäisiä olioita että alikokoelmia, mikä mahdollistaa rakenteeltaan monipuolisten tietokantaratkaisujen rakentamisen. (Firebase 12.5.2025.) Firestore on valittu erityisesti sen helppokäyttöisyyden vuoksi.

Android Studio on virallinen integroitu kehitysympäristö (IDE, Integrated Development Environment) Android-sovellusten kehittämiseen. Tämän projektin kehitystyössä se on kuitenkin ollut mukana lähinnä teknisten vaatimusten vuoksi. Sitä ei ole käytetty varsinaisena kehitysympäristönä, mutta sen sisältämä Android SDK Manager -työkalu on ollut hyödyllinen Android SDK:n asennuksessa ja hallinnassa. Vaikka SDK-komponentit on mahdollista asentaa myös muita reittejä, Android Studio tarjoaa yleisesti suositellun ja vaivattoman tavan niiden hallintaan.

5 KEHITYSTARPEIDEN TUNNISTAMINEN JA ARVIOINTI

Sovelluksen kehitystarpeita on pohdittu sekä käytännön havaintojen että omien ideointien pohjalta. Olemassa olevien haasteiden tunnistamisen lisäksi on yritetty etsiä uusia ratkaisuja, jotka parantaisivat sovelluksen käytettävyyttä ja laajentaisivat sen toiminnallisuuksia. Alla on esimerkkiehdotuksia ja arviointia niiden toteutettavuudesta ja hyödyllisyydestä.

Hakutoiminto ja suodattimet

Käyttäjät voisivat etsiä tuotteita hakusanoilla tai suodattaa tuotteita esimerkiksi tuoteryhmän tai kaupan mukaan. Nämä parantaisivat käytettävyyttä erityisesti silloin, kun tuoteluettelo on laaja. Kyseiset toiminnot onkin jo otettu huomioon sovelluksen käyttöliittymässä, mutta toteutus on jäänyt tekemättä. Työ olisi teknisesti suhteellisen yksinkertainen ja se parantaisi sovelluksen käytettävyyttä merkittävästi.

Kauppojen lisäysmahdollisuus

Käyttäjille voisi antaa mahdollisuuden lisätä omia kauppojaan sovellukseen. Tällä hetkellä kauppavaihtoehdot ovat ennalta määritettyjä, eikä niitä voi muokata tai lisätä käyttäjän toimesta. Tämä lisäys voisi parantaa sovelluksen joustavuutta ja houkuttelevuutta erityisesti käyttäjille, joilla on erityisiä kauppavaatimuksia.

Automaattinen tiedonkeruu

Sovellus hakisi automaattisesti hintatietoja kauppojen verkkosivuilta, jolloin manuaalinen kirjaaminen vähenisi. Tämä vähentäisi käyttäjän työtä ja tekisi sovelluksesta huomattavasti houkuttelevamman käyttää. Tämä olisi käyttäjille erittäin hyödyllinen ominaisuus, mutta toteutus on teknisesti suhteellisen haastava. Se vaatisi sovelluksen kehittämistä esimerkiksi verkkosivujen hintojen ohjelmalliseen keräämiseen tai API-yhteyksien hyödyntämiseen, joiden kautta hintoja voitaisiin noutaa.

Kauppasuositukset ostoslistassa

Tämä kehitysidea pohjautuu toiseen opinnäytetyöhön (Nordlund, 2016), jossa kehitettiin sovellus, jonka avulla käyttäjä voi syöttää tuotteiden hintoja eri kaupoista ja ostoslistaa tehdessä sovellus suosittelee tuotekohtaisesti edullisinta kauppaa. Tämä toiminnallisuus voisi parantaa sovelluksen käytettävyyttä, sillä käyttäjä näkisi ostoslistan kirjoitettuaan heti, mikä kauppa tarjoaa edullisimmat hinnat kullekin tuotteelle.

Kulutustottumusten seuranta ostoslistan mukaan

Sovellus voisi seurata käyttäjän ostoslistalle merkittyjä tuotteita ja analysoida kulutustottumuksia eri näkökulmista, kuten rahanmenoa kauppakohtaisesti tai tuoteryhmittäin. Käyttäjä voisi esimerkiksi merkitä missä kaupassa aikoo asioida, jolloin sovellus laskisi kokonaiskustannukset sen mukaan. Vaihtoehtoisesti sovellus voisi laskea summan valitsemalla automaattisesti halvimmat kaupat. Tämä mahdollistaisi käyttäjälle tarkemman kuvan omista kulutustottumuksistaan ja auttaisi optimoimaan ostoskäyttäytymistään. Suomalaisessa tutkimuksessa on nimittäin havaittu, että mahdollisuus tarkastella ostohistoriaa voi vaikuttaa positiivisesti kuluttajien tuleviin ostopäätöksiin (Koski, Kuikkaniemi & Pantzar 2023).

Jatkokehitys

Kaikki esitetyt kehitysideat tukevat sovelluksen käytettävyyden ja toiminnallisuuden parantamista ja onkin todennäköistä, että niitä toteutetaan jossain muodossa tulevaisuudessa. Tämän opinnäytetyön puitteissa päätettiin kuitenkin keskittyä **automaattiseen tiedonkeruuseen**, joka toisi sovellukseen automaatiota verkkosivujen hintatietojen keräämisen kautta. Ominaisuus toisi merkittävää lisäarvoa käyttäjälle ja se tarjoaa samalla mahdollisuuden syventyä teknisesti kiinnostavaan osa-alueeseen, kuten web scrapingiin.

6 TIEDONHARAVOINTI VERKKOSIVUILTA

Tiedonharavoinnilla (data scraping) tarkoitetaan tiedon tai datan, yleensä automaattista, keräämistä eri lähteistä. Kyse ei ole pelkästään verkkosisällön haravoinnista, vaan menetelmää voidaan soveltaa myös esimerkiksi tietokantoihin, asiakirjoihin, taulukoihin tai tekstitiedostoihin. Tiedonharavointi toimii siis eräänlaisena kattokäsitteenä, jonka alle esimerkiksi web scraping ja screen scraping kuuluvat. (Stanley s.a. a.)

Verkkosivujen haravointi (web scraping) on tiedonharavointia, jossa dataa kerätään automaattisesti verkkosivulta. Se perustuu prosessiin, jossa verkkosivuille tehdään HTTP-pyyntöjä, sivujen HTML-sisältö ladataan ja se jäsennetään eli parsitaan haluttuun muotoon. (Stanley s.a. b.) Tämä on erityisen hyödyllistä silloin, kun verkkosivu ei tarjoa tietoa rajapintojen (API) kautta. Verkkoharavointiin on olemassa runsaasti erilaisia työkaluja ja lähestymistapoja kuten esimerkiksi ohjelmallinen haravointi.

Julkiset ja kolmannen osapuolen rajapinnat voivat olla osa tiedonharavointiprosessia, mutta niitä ei voida pitää perinteisinä web scraping -menetelminä. API (Application Programming Interface) on ohjelmointirajapinta, jonka kautta sovellukset voivat vaihtaa tietoa keskenään. Verkkosivujen tapauksessa API (Application Programming Interface) tarjoaa rakenteista tietoa esimerkiksi JSON-muodossa, mikä tekee tiedon käsittelystä helpompaa ja luotettavampaa kuin HTML-sivujen sisällön jäsentäminen. Rajapintojen käyttö on lisäksi yleensä vakaampaa ja tehokkaampaa, mutta niillä saattaa olla erilaisia käyttörajoituksia tai kustannuksia. (Pelakauskas 10.1.2025.) On kuitenkin yleisesti suositeltua käyttää julkista APIa, jos verkkosivusto sellaisen tarjoaa, sillä ne tarjoavat rakenteellisen ja virallisen tavan tiedon keräämiseen.

Screen scraping on tiedonharavointimenetelmä, jossa tietoja poimitaan suoraan verkkosivun tai sovelluksen käyttöliittymästä simuloimalla käyttäjän vuorovaikutusta. Tätä menetelmää käytetään erityisesti visuaalisen, jäsentämättömän tiedon keräämiseen verkkosivuilta. Tieto voidaan poimia esimerkiksi kuvakaappausten tai optisen merkintunnistuksen (OCR, Optical Character Recognition)

avulla suoraan näytöltä, mikä mahdollistaa myös JavaScriptin avulla renderöityjen elementtien tunnistamisen.

Screen scraping ja web scraping ovat käsitteitä, joita käytetään usein toistensa synonyymeina, mutta niillä on pieniä eroja. Web scraping tarkoittaa tyypillisesti tiedon keräämistä suoraan verkkosivujen HTML-rakenteesta ja muista jäsennellyistä lähteistä, kun taas screen scraping keskittyy ruudulla näkyvän sisällön tallentamiseen sellaisena kuin se käyttäjälle esitetään. Nykyaikaiset web-scraping työkalut, kuten Puppeteer, kuitenkin hämärtävät näiden menetelmien välistä rajaa, sillä ne mahdollistavat vuorovaikutuksen verkkosivun käyttöliittymän kanssa ja voivat poimia myös visuaalisesti esitettyä tietoa. Tämä tekee mahdolliseksi sellaisten tietojen keräämisen, jotka ilmestyvät vasta käyttäjän toimenpiteiden jälkeen. (Dilmegani 6.4.2025.)

6.1 Hyvät käytänteet ja eettisyys

Verkkosivujen datan haravointi on hieman harmaata aluetta oikeudellisesta näkökulmasta. Käyttötarkoitus, tekninen toteutustapa, verkkosivun käyttöehdot sekä eri maiden lainsäädännöt vaikuttavat keskeisesti siihen, missä määrin scraping on sallittua.

Esimerkiksi Ryanair Ltd v PR Aviation BV -tapauksessa (2015) PR Aviation keräsi ja käytti Ryanairin verkkosivujen lentotietoja ilman lupaa, vaikka scraping oli kielletty verkkosivuston käyttöehdoissa. Tässä tapauksessa aluetuomioistuin katsoi PR Aviationin loukanneen Ryanairin tekijänoikeuksia ja joutui maksamaan yrityksellä korvauksen. Huomioitavaa on kuitenkin, että kyseisessä tapauksessa tietoja kerännyt osapuoli hyödynsi kerättyä dataa kaupallisesti, mikä saattoi myös vaikuttaa oikeuden arvioon toiminnan luvattomuudesta.

Pienimuotoinen scraping, joka ei aiheuta yritykselle haittaa tai riko selvästi käyttöehtoja, katsotaan usein käytännössä hyväksyttäväksi. Erityisesti jos siitä ei ole taloudellista hyötyä kerääjälle tai kilpailullista haittaa yritykselle.

Vastuullinen kehittäjä tarkistaakin aina sivuston käyttöehdot ja mahdolliset rajoitukset (esim. robots.txt), sekä arvioi tiedonkeruun eettisyyttä ja tarkoitusta ennen

toteutuksen aloittamista. Alle on kerätty keskeisiä periaatteita, joita on hyvä noudattaa tiedonharavoinnissa (Densmore 23.7.2017):

- **Käytä ensisijaisesti rajapintoja:** Mikäli verkkopalvelu tarjoaa julkisen APIn, tulee sitä käyttää ensisijaisena tiedonlähteenä. Tämä vähentää palvelimen kuormitusta ja on palveluntarjoajan sallima tapa käyttää saatavilla olevaa dataa.
- **Noudata robots.txt-tiedostoa ja käyttöehtoja**: Sivuston robots.txt-tiedosto ja mahdolliset käyttöehdot kertovat, mitä osioita saa haravoida ja millä ehdoilla. Robots.txt-tiedostossa määritellyt säännöt eivät tosin ole oikeudellisesti sitovia kuten verkkosivuston käyttöehdot, vaan ne toimivat ohjeistuksena automatisoiduille työkaluille.
- **Tunnista itsesi:** HTTP-pyynnöissä tulee käyttää User-Agent-kenttää, josta käy ilmi kuka olet ja miksi haet dataa. Tämä luo läpinäkyvyyttä ja antaa ylläpitäjälle mahdollisuuden ottaa yhteyttä tarvittaessa.
- **Rajoita pyyntöjen määrää:** Liiallinen pyyntöjen määrä voi kuormittaa palvelimia ja muistuttaa palvelunestohyökkäystä. Tämän vuoksi on suositeltavaa käyttää viiveitä pyyntöjen välillä.
- **Kerää vain tarvittava tieto:** Dataa tulisi kerätä ja säilyttää vain sen verran kuin on tarpeellista.
- **Älä kopioi tai esitä sisältöä omana:** Haravoitua tietoa ei tule esittää omana sisältönä ilman asianmukaista viittausta. Mikäli dataa julkaistaan edelleen, mainitaan lähde asianmukaisesti.
- **Noudata lakeja ja yleisiä käytäntöjä**: Haravointi ei saa rikkoa tekijänoikeuksia, tietosuoja-asetuksia tai muita voimassa olevia lakeja. Esimerkiksi henkilötietojen keruu ilman lupaa ei ole sallittua.

Eettisen tiedonharavoinnin tavoitteena pitäisi olla molemminpuolinen hyöty. Sovelluskehittäjä saa tarvitsemansa datan ja datan alkuperäinen tarjoaja saa näkyvyyttä tai muuta lisäarvoa.

6.2 Haravoinnissa käytettäviä työkaluja

Ohjelmalliseen verkkoharavointiin on tarjolla useita erilaisia työkaluja ja kirjastoja, joita voidaan käyttää riippuen siitä, millaista sisältöä haetaan ja miten verkkosivusto on rakennettu.

On olemassa kevyitä kirjastoja, jotka soveltuvat erityisesti staattisen sisällön käsittelyyn ilman selainsimulaatiota. Tällaiset työkalut ovat tehokkaita vaihtoehtoja silloin, kun haluttu sisältö löytyy suoraan sivun lähdekoodista ilman JavaScriptin suorittamista. Ne kuluttavat vähemmän resursseja, mutteivat pysty käsittelemään JavaScriptin jälkikäteen lataamaa dataa. Tämä voi olla haaste, sillä nykyään suuri osa verkkosivustoista on dynaamisia.

Dynaamiset verkkosivut eivät näytä kaikkea sisältöä heti verkkosivun ladattua, vaan ne täydentävät tai muuttavat sivun sisältöä JavaScriptin avulla käyttäjän toimien tai muiden tapahtumien perusteella. Tämä tekee perinteisestä tiedon hausta haastavampaa, koska sisältöä ei ole heti näkyvissä lähdekoodissa. Tällaisissa tilanteissa tarvitaan selaimen toimintaa jäljitteleviä työkaluja, jotka pystyvät toimimaan kuin oikea käyttäjä. Suosituimpia kirjastoja tähän tarkoitukseen ovat Selenium, Puppeteer ja Playwright. Ne voivat avata verkkosivun kuin oikea käyttäjä, suorittaa hakuja, odottaa dynaamisen sisällön latautumista ja poimia tarvittavat tiedot suoraan selainympäristöstä.

Selenium

Selenium on yksi vanhimmista ja laajimmin käytetyistä selainten automatisointikirjastoista. Se kehitettiin vuonna 2004 ja on edelleen suosittu erityisesti siksi, että se tukee useita ohjelmointikieliä, kuten Javaa, Pythonia, C#:a, Rubyä ja JavaScriptiä. Selenium toimii kaikkien suurimpien selainten, kuten Chromen, Firefoxin, Safarin ja Edgen kanssa, mikä tekee siitä joustavan työkalun eri käyttötarkoituksiin. Sen vahvuuksiin kuuluu myös laaja ja aktiivinen käyttäjäyhteisö sekä kattava dokumentaatio, mikä helpottaa ongelmien ratkaisemista ja työkalun oppimista. Heikkouksiin kuuluu monimutkaisempi käyttöönotto, selainkohtaiset ajurit ja puutteet uusimmissa ominaisuuksissa. Verkkoharavoinnissa sen hitaampi

suoritusteho ja suuri resurssien kulutus tekevät siitä vähemmän tehokkaan verrattuna nykyaikaisempiin vaihtoehtoihin.

Puppeteer

Puppeteer on Googlen vuonna 2017 kehittämä Node.js-kirjasto, joka on suunniteltu erityisesti Chromium-pohjaisten selainten automaatioon. Puppeteer tukee vain JavaScriptiä ja TypeScriptiä. Se on optimoitu Chrome-selaimelle, mutta Firefoxin tuki on olemassa, vaikkakin yhä kokeellisella tasolla. Puppeteer on nopea ja kevyt, sen asennus onnistuu helposti Node Package Managerin (npm) kautta ja käytön voi aloittaa nopeasti ilman monimutkaista konfiguraatiota. Lisäksi Puppeteerilla on suuri käyttäjäyhteisö ja kattava dokumentaatio, mikä helpottaa ongelmien ratkaisua ja oppimista. Verkkoharavointikäytössä se on tehokas ja nopea vaihtoehto.

Playwright

Playwright on Microsoftin vuonna 2020 julkaisema kirjasto, joka tukee useita ohjelmointikieliä kuten JavaScriptiä, TypeScriptiä, Pythonia, C#:a ja Javaa. Se toimii Chromen, Firefoxin, Safarin ja Edgen kanssa. Playwright on nopea, kevyt ja tarjoaa paljon moderneja ominaisuuksia. Lisäksi sen asennus tapahtuu helposti npm:n avulla. Playwright on tehokas ja nykyaikainen ratkaisu verkkoharavointiin. Kyseessä on kuitenkin suhteellisen uusi työkalu, joten yhteisö on vielä vasta kasvava.

6.3 Kehitystyön aloitus

Ennen työn toteutusta tuli selvittää, millaisia ehtoja ja rajoituksia valitsemillani kaupoilla on verkkosivujensa tietojen käyttöön liittyen. Selvitys kohdistui ruokakauppamarkkinoiden kahteen suurimpaan toimijaan, S-kauppa.fi- ja K-Ruoka.fi-sivustoihin.

K-Ruoka.fi -verkkopalvelussa kiellettiin kaikenlainen sisällön kopiointi ja jakelu ilman nimenomaista lupaa, pois lukien henkilökohtaiseen käyttöön tapahtuva selaaminen ja tulostaminen (Kesko Oyj 2.9.2024). Lisäksi heidän robots.txt-

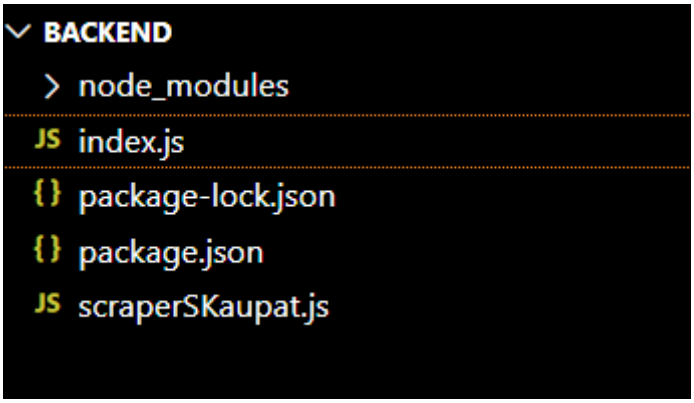
tiedostonsa sisälsi lukuisia Disallow-määrityksiä, joista tärkeimpiä sovelluksen kannalta olivat hakutoimintoihin ja suodattimiin liittyvät rajoitukset. S-kauppa.fi-verkkosivuston käyttöehdoista ei löytynyt suoraa mainintaa sisällön kopioinnista tai sen käytön rajoituksista (SOK s.a.), eikä robots.txt-tiedostossakaan ollut rajoituksia. Kummallakaan sivustolla ei vaikuttanut olevan suoraa pääsyä julkisiin rajapintoihin. Näiden ehtojen ja rajoitteiden perusteella valitsin ensisijaiseksi scraping testikohteeksi S-kaupan sivuston.

Kun käytön luvallisuus oli varmistettu, aloitin verkkosivun rakenteen tarkemman tarkastelun. Ensimmäiseksi tuli määritellä, mitä tietoja halutaan kerätä ja mistä elementeistä nämä löytyvät. Selaimen kehittäjätyökaluja (DevTools) hyödyntämällä oli mahdollista selvittää, mitkä HTML-elementit sisältävät halutut tuotetiedot, kuten tuotenimet ja hinnat. Tarkastelun yhteydessä havaitsin, että tarvittava data oli dynaamista eli se ei ollut suoraan nähtävissä lähdekoodissa, vaan latautui JavaScriptin avulla. Tämän vuoksi tarvitsin työkalun, joka kykenee simuloimaan selaimen toimintaa ja käsittelemään myös JavaScriptin kautta ladattavaa sisältöä.

Valinta kohdistui Puppeteerin ja Playwrightin kirjastojen välille, sillä molemmat ovat nopeita ja kevyitä, ne on helppo asentaa sekä soveltuvat hyvin tiedonharavointiin. Playwright vaikutti teknisesti paremmalta vaihtoehdolta, mutta sen käytöstä oli vaikeampi löytää käytännön esimerkkejä ja ohjeistuksia, sillä kyseessä on suhteellisen uusi työkalu. Tämän vuoksi valinta osui Puppeteeriin, jonka yhteisötuki ja esimerkit vastaavista käyttötapauksista olivat huomattavasti paremmin saatavilla. Lisäksi se toimii Node.js-ympäristössä, joka oli itselle jo jokseenkin tuttu.

Scrapingia varten loin muusta sovelluksesta erillisen projektikansion nimellä backend, koska tiedonkeruu ei varsinaisesti ole osa käyttöliittymän toimintaa vaan toimii taustaprosessina. Tämä helpottaa kehitystyötä ja tekee koodista selkeämmin hallittavaa, erityisesti kun kyseessä oli kokeellinen ominaisuus, jota ei vielä oltu integroimassa käyttöliittymään. Näin myös vältetään mahdolliset ongelmat, jotka voisivat vaikuttaa käyttöliittymän toimintaan tai suorituskykyyn, jos scrapingissa ilmenisi virheitä.

Scraperin kehittämisessä käytettiin Node.js-ympäristöä ja Puppeteer-kirjastoa, jotka asennettiin Node Package Managerin (npm) avulla. Npm luo myös projektikansioon keskeiset tiedostot ja kansiot riippuvuuksien hallintaa varten, kuten node_modules-kansion ja package.json-tiedoston. Selkeyden ja ylläpidettävyyden vuoksi koodi jaettiin kahteen osaan: index.js ja scraperSKaupat.js. Kuvassa 5 näkyy tiedostorakenne.



Kuva 5. Tiedostorakenne

Index.js tiedosto toimii paikkana, joka käynnistää scraperin. Se tuo scraperSKaupat.js-tiedoston käyttöön ja kutsuu sieltä löytyvää startScraping-funktiota, joka vastaa datan keräämisestä kaupan verkkosivulta (kuva 6). Vaikka index.js-tiedosto on vielä itsessään pelkkä taustaprosessi, sen on mahdollista laajentua palvelinpuoleksi, mikäli projektin tulevaisuudessa päätetään siirtyä perinteiseen palvelinratkaisuun. Index.js-tiedostoon on myös helppo lisätä mahdollisten muiden kauppojen scraperit tulevaisuudessa. Tämä mahdollistaa sen, että kaikkien kauppojen scraping-logiikka pysyy erillään ja helposti hallittavana, mutta ne voidaan silti käynnistää yhdestä paikasta.

```
4
5   // Tuodaan Skaupan scraper toisesta tiedostosta
6   const { startScraping } = require('./scraperSKaupat');
7
8   // Käynnistä scraper
9   startScraping();
10
```

Kuva 6. Index.js-tiedosto

ScraperSKaupat.js-tiedosto puolestaan sisältää kaupan verkkosivuston tietojen keräämiseen tarvittavan koodin. Tiedostossa on kaksi pääasiallista toimintoa.

Ensimmäinen on startScraping (kuva 7), joka avaa headless-selaimen automatisoidusti Puppeteerilla ja siirtyy halutulle sivustolle. Tämän jälkeen se odottaa käyttäjän syötettä komentoriviltä. Kun käyttäjä kirjoittaa tuotteen nimen, ohjelma hakee kyseisen haun perusteella ehdotuksia ja näyttää ne terminaalissa (kuva 8).

```
38  // Käynnistää Puppeteer-selaimen ja mahdollistaa tuotteen haun terminaalista
39  async function startScraping() {
40      const browser = await puppeteer.launch();
41      const page = await browser.newPage();
42      // Siirtyy S-kauppojen verkkosivulle
43      await page.goto('https://www.s-kaupat.fi/');
44
45      // Odottaa hakukentän olevan valmis käyttöön
46      await page.waitForSelector('[data-test-id="search-input"]');
47
48      // Setup readline (komentorivi) jotta käyttäjä voi syöttää tuotteita
49      const rl = readline.createInterface({
50          input: process.stdin,
51          output: process.stdout
52      });
53
54      // Kysyy käyttäjältä tuotteen nimen ja näyttää ehdotukset
55      function promptInput() {
56          rl.question('Syötä tuotteen nimi: ', async (input) => {
57              if (input.toLowerCase() === 'exit') {
58                  await browser.close();
59                  rl.close();
60                  return;
61              }
62              const suggestions = await getSuggestions(page, input);
63              // Tulostetaan ehdotukset konsoliin
64              console.log('Suggestions:', suggestions);
65              promptInput();
66          });
67      }
68
69      promptInput();
70  }
71
72  module.exports = { startScraping };  // Export
73
```

Kuva 7. startScraping-funktio

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                chrome
Syötä tuotteen nimi: kahvi
Suggestions: [
  {
    name: 'Paulig Juhla Mokka kahvi suodatinjauhatus 500g',
    price: '9,25 €'
  },
  {
    name: 'Paulig Juhla Mokka Tumma Paahto kahvi suodatinjauhatus 500g',
    price: '9,25 €'
  },
  { name: 'Rainbow 500g kahvi suodatinjauhatus', price: '5,95 €' },
  {
    name: 'Paulig Juhla Mokka Ilta kofeiiniton kahvi suodatinjauhatus 270g',
    price: '6,25 €'
  },
  {
    name: 'Paulig Juhla Mokka Lempeä kahvi suodatinjauhatus 425g',
    price: '9,25 €'
  }
    name: 'Paulig Juhla Mokka Lempeä kahvi suodatinjauhatus 425g',
    price: '9,25 €'
  }
]
Syötä tuotteen nimi:
```

Kuva 8. Tuote-ehdotukset terminaalissa

Toinen keskeinen funktio on getSuggestions, joka suorittaa varsinaisen tiedonkeruun. Se syöttää hakusanan verkkosivun hakukenttään, odottaa hakuehdotusten ilmestymistä ja kerää ehdotukset (kuva 9).

```
// Hakee tuoteehdotuksia käyttäen annettua hakusanaa
async function getSuggestions(page, query) {
    // Aktivoi ja tyhjentää hakukentän
    await page.focus('[data-test-id="search-input"]');
    await page.evaluate(() => {
        const input = document.querySelector('[data-test-id="search-input"]');
        input.value = '';
        input.dispatchEvent(new Event('input', { bubbles: true }));
    });
    // Haku
    await page.type('[data-test-id="search-input"]', query);

    // Odottaa ehdotuksien päivittymistä
    try {
        // Odottaa että vähintään yksi ehdotus ilmestyy
        await page.waitForSelector('[data-test-id="search-product-suggestion__productName"]', { timeout: 1500 });
    } catch (e) {
        // Jos ehdotuksia ei löydy aikakatkaisun aikana, palauttaa tyhjän listan
        return [];
    }
    // Kerää enintään 5 ehdotusta hakutuloksista, palauttaa objektina jossa on nimi ja hinta
    const suggestions = await page.$$eval(
        '[data-test-id="search-product-suggestion"]',
        nodes => nodes.slice(0, 5).map(el => ({
            // Etsii tuotteen nimen ja poista ylimääräiset
            name: el.querySelector('[data-test-id="search-product-suggestion__productName"]')?.textContent.trim() || '',
            // Etsii tuotteen hinnan ja poista ylimääräiset
            price: el.querySelector('[data-test-id="display-price"]')?.textContent.trim() || ''
        }))
    );
    // Palauta löydetyt ehdotukset
    return suggestions;
}
```

Kuva 9. getSuggestions-funktio

6.4 Kehitystyön tulevaisuus

Tulevaisuudessa olisi tarkoitus kokeilla scraperin integrointia sovelluksen nykyiseen käyttöliittymään. Tämä edellyttäisi käyttöliittymän muokkaamista, jotta käyttäjä voisi syöttää tuotehakunsa suoraan sovelluksessa. Kun tämä on saatu toimimaan, voisi seuraavaksi kokeilla toteuttaa kauppakohtaista tuotehakua, jossa käyttäjä voisi hakea ja valita haluamansa kaupat, joihin rajaa haun koskemaan.

Kehityksen aikana kävi kuitenkin ilmi, että vaikka S-kaupan sivustolta ei löytynyt virallista, dokumentoitua APIa, sivun taustalla havaittiin käytettävän niin sanottua “piilotettua” rajapintaa. Termi ”piilotettu” voi olla kuitenkin hieman harhaanjohtava, koska kyse ei ole varsinaisesti salatusta toiminnallisuudesta, vaan enemmänkin dokumentoimattomasta, jota ei ole tarkoitettu julkiseen käyttöön (Segar 29.2.2024). Niiden käyttö voi juridisesti olla hieman tulkinnanvaraista. Tästä

huolimatta rajapinnan hyödyntäminen olisi teknisesti helpompi ja selkeämpi vaihtoehto kuin scraping. Jatkokehityksen kannalta olisi siis perusteltua tutustua tähän mahdollisuuteen tarkemmin ja pohtia eettisiä näkökulmia dokumentoimattomien rajapintojen käytöstä.

7 YHTEENVETO JA POHDINTA

Opinnäytetyön tavoitteena oli kartoittaa olemassa olevan hintaseurantasovelluksen nykytilaa ja tarkastella sen toiminnallisuuksia, käytettävyyttä sekä kehityspotentiaalia. Tuloksia arvioitaessa on kuitenkin huomioitava, että sovelluksen käytettävyyttä ei testattu ulkopuolisten käyttäjien kanssa vaan arviot ja kehitysehdotukset perustuivat kirjoittajan omaan näkemykseen sekä aiempaan teoreettiseen tietoon.

Työssä esitettiin konkreettisia jatkokehitysideoita, joista keskeisimpänä tarkasteltiin tiedonharavoinnin hyödyntämistä hintatietojen automaattiseen keräämiseen verkkokauppojen sivuilta.

Tiedonharavoinnin osalta työssä onnistuttiin rakentamaan toimiva prototyyppi, joka kykenee hakemaan tuote-ehdotuksia verkkokaupan hakutoiminnon kautta. Tämä on hyvä pohja jatkokehitykselle, kuten tuotehakujen kohdistamiselle tiettyihin kauppoihin ja integrointi käyttöliittymään. Vaikeuksia toi tosin verkkokauppojen käyttöehdot, jotka osa kieltävät tietojen keräämisen tai jakamisen, mikä rajoittaa ratkaisun soveltamista laajempaan käyttöön.

Jatkokehityksessä olisi hyödyllistä toteuttaa käyttäjätestauksia, joiden avulla voitaisiin arvioida paremmin sovelluksen käytettävyyttä ja tunnistaa kehityskohteita. Lisäksi olisi tärkeää selvittää dokumentoimattomien rajapintojen käytön sallittavuus ja tutkia, missä määrin pienimuotoinen scraping omaan käyttöön on käytännössä hyväksyttävää.

Omasta näkökulmastani työ tarjosi mahdollisuuden yhdistää käytännönläheinen sovellusanalyysi tekniseen taustatyöhön. Vaikka kehitystyötä ei ehditty toteuttaa loppuun saakka, sain arvokasta kokemusta tietojenharavoinnista ja siihen liittyvistä käytännön haasteista. Tulevaisuudessa toivon voivani kehittää sovellusta eteenpäin näiden oppien pohjalta.

LÄHTEET

Adobe s.a. Inspiration in the color tangerine. Luettavissa: https://www.adobe.com/express/colors/tangerine. Luettu: 15.3.2025.

Android Developers 10.2.2025. Update the IDE and SDK tools. Luettavissa: https://developer.android.com/studio/intro/update. Luettu: 3.4.2025.

Android Developers 20.12.2024. Layout basics. Luettavissa: https://developer.android.com/design/ui/mobile/guides/layout-and-content/layout-basics. Luettu: 15.3.2025.

Aliskan, A. 5.3.2021. Usability and its importance. Medium. Luettavissa: https://bootcamp.uxdesign.cc/usability-and-its-importance-1e6c63e81614. Luettu: 25.2.2025.

Binkley, J.K. & Bejnarowicz, J. 2003. Consumer price awareness in food shopping: the case of quantity surcharges. Journal of Retailing, 79, 1, s. 27–35. Luettavissa: https://doi.org/10.1016/S0022-4359(03)00005-8. Luettu: 8.1.2025.

Capgemini 2023. What matters to today's consumers 2023: consumer behavior tracker for the consumer products and retail industries. Capgemini Research Institute. Luettavissa: https://prod.ucwe.capgemini.com/fi-en/wp-content/uploads/sites/26/2023/01/Final-Web-Version-Report-Consumer-Trends.pdf. Luettu: 8.1.2025.

Densmore, J. 23.7.2017. Ethics in Web Scraping. Medium. Luettavissa: https://medium.com/data-science/ethics-in-web-scraping-b96b18136f01. Luettu: 27.4.2025.

Digitaalisen Markkinoinnin Sanakirja s.a. UCD – User-Centered Design. Luettavissa: https://digitaalisenmarkkinoinninsanakirja.fi/ucd-user-centered-design/. Luettu: 25.2.2025.

Digital.gov 26.2.2025. Usability. Luettavissa: https://digital.gov/topics/usability. Luettu: 28.2.2025.

Dilmegani, C. 6.4.2025. Roadmap to Web Scraping: Methods & Tools in 2025. AIMultiple Research. Luettavissa: https://research.aimultiple.com/web-scraping/. Luettu: 1.5.2025.

Expo Docs 28.3.2025. Introduction to development builds. Luettavissa: https://docs.expo.dev/develop/development-builds/introduction/. Luettu: 1.4.2025.

Finn, T. & Downie, A. 10.1.2025. What is user experience (UX)? IBM. Luettavissa: https://www.ibm.com/think/topics/user-experience. Luettu: 26.2.2025.

Firebase 12.5.2025. Cloud Firestore. Luettavissa: https://firebase.google.com/docs/firestore. Luettu: 13.5.2025.

Forsman-Hugg, S., Kinnunen, P. & Yli-Liipola, P. 2024. Kuluttajien näkemyksiä ruuan hinnan nousun vaikutuksista kulutukseen ja ostokäyttäytymiseen. PTT raportteja 288. Helsinki. Luettavissa: https://www.ptt.fi/wp-content/uploads/2024/04/PTTraportteja288.pdf. Luettu: 10.1.2025.

GitHub Docs 13.3.2025. About GitHub and Git. Luettavissa: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git. Luettu: 4.4.2025.

Interaction Design Foundation 2016a. What is User Centered Design (UCD)? Luettavissa: https://www.interaction-design.org/literature/topics/user-centered-design. Luettu: 25.2.2025.

Interaction Design Foundation 2016b. What is Usability? Luettavissa: https://www.interaction-design.org/literature/topics/usability. Luettu: 26.3.2025.

Interaction Design Foundation 2016c. What is User Experience (UX) Design? Luettavissa: https://www.interaction-design.org/literature/topics/ux-design. Luettu: 24.2.2025.

ISO 9241-11:2018. Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. International Organization for Standardization. Geneva.

ISO 9241-210:2019. Ergonomics Of Human-System Interaction – Part 210: Human-Centred Design For Interactive Systems. International Organization for Standardization. Geneva.

ISO 9241-110:2020. Ergonomics of human-system interaction – Part 110: Interaction principles. International Organization for Standardization. Geneva.

Juntunen, E. 25.11.2024. Ruokakorivertailu 2024 – Prisma vs. Citymarket vs. Lidl. Luettavissa: https://www.omavaraisuushaaste.com/ruokakorivertailu-2024-prisma-vs-citymarket-vs-lidl/. Luettu: 10.1.2025

Kesko Oyj 2.9.2024. Palvelun käyttöehdot. Luettavissa: https://www.k-ruoka.fi/artikkelit/palvelun-kayttoehdot. Luettu: 1.5.2025.

Koski, H., Kuikkaniemi, K. & Pantzar, M. 2023. Do Grocery Feedback Systems Enabling Access to Past Consumption Impact Individual Food Purchase Behavior? ETLA Working Papers 103. Elinkeinoelämän tutkimuslaitos. Luettavissa: https://www.etla.fi/wp-content/uploads/Etla-Working-Paper-103.pdf. Luettu: 13.1.2025.

Kuluttajaliitto 15.11.2020. Ostoskorin tuotteiden hintavertailu. Aula Researchin toteuttama kyselytutkimus. Luettavissa: https://www.kuluttajaliitto.fi/artikkeli/suomalaiset-vertailevat-hintoja-erityisesti-ruokakaupassa/. Luettu: 10.1.2025.

Kuluttajaliitto 10.11.2022. Mistä suomalaiset ovat säästäneet? Aula Researchin toteuttama kyselytutkimus. Luettavissa: https://www.kuluttajaliitto.fi/2022/11/10/mista-suomalaiset-ovat-saastaneet/. Luettu: 7.1.2025.

Kwarteng, M.A., Jibril, A.B., Botha, E., Osakwe, C.N. 2020. The Influence of Price Comparison Websites on Online Switching Behavior: A Consumer Empowerment Perspective. Teoksessa Hattingh, M., Matthee, M., Smuts, H., Pappas, I., Dwivedi, Y. & Mäntymäki, M. (toim.). Responsible Design, Implementation and Use of Information and Communication Technology. Lecture Notes in Computer Science, 12067, s. 216–227. Luettavissa: https://doi.org/10.1007/978-3-030-44999-5_18. Luettu: 12.1.2025.

Lichtenstein, D.R., Ridgway, N.M. & Netemeyer, R.G. 1993. Price Perceptions and Consumer Shopping Behavior: A Field Study. Journal of Marketing Research, 30, 2, s. 234–245. Luettavissa: https://doi.org/10.2307/3172830. Luettu: 13.1.2025.

Lindgren, C., Daunfeldt, S.-O. & Rudholm, N. 2021. Pricing in Retail Markets with Low Search Costs: Evidence from a Price Comparison Website. HFI Working Paper 18. Institute of Retail Economics. Stockholm. Luettavissa: https://hdl.handle.net/10419/246772. Luettu: 12.1.2025.

Martin, S. 2020. Market transparency and consumer search - Evidence from the German retail gasoline market. DICE Discussion Paper 350. Heinrich Heine University Düsseldorf, Düsseldorf Institute for Competition Economics (DICE). Düsseldorf. Luettavissa: https://hdl.handle.net/10419/223410. Luettu: 12.1.2025.

Monroe, K.B. & Lee, A.Y. 1999. Remembering versus knowing: issues in buyers' processing of price information. Journal of the Academy of Marketing Science, 27, 2, s. 207–225. Luettavissa: https://doi.org/10.1177/0092070399272006. Luettu: 8.1.2025.

Nielsen, J. 1994. Enhancing the explanatory power of usability heuristics. SIGCHI Conference on Human Factors in Computing Systems (CHI '94), Boston, s. 152–158. Luettavissa: https://doi.org/10.1145/191666.191729. Luettu: 23.5.2025.

NielsenIQ 26.3.2025. Päivittäistavaramyymälärekisteri 2024. Lehdistötiedote. Luettavissa: https://www.epressi.com/tiedotteet/kauppa/paivittaistavaramyymalarekisteri-2024.html. Luettu: 12.1.2025.

Nordlund, M. 2016. Ostoslista Android-sovelluksen kehittäminen. Opinnäytetyö. Satakunnan ammattikorkeakoulu, Tietotekniikan koulutusohjelma. Luettavissa: https://urn.fi/URN:NBN:fi:amk-201605259675. Luettu: 12.4.2025.

Pelakauskas, A. 10.1.2025. Web Scraping vs API: Which to Choose in 2025. Blogi. Luettavissa: https://oxylabs.io/blog/api-vs-web-scraping. Luettu: 25.4.2025.

Peltoniemi, A. & Yrjölä, T. 2012. Kuluttajien ja tuottajien näkemyksiä ruoan ostopäätöksistä ja tuotantotavoista. Työselosteita ja esitelmiä 138. Kuluttajatutkimuskeskus. Luettavissa: http://hdl.handle.net/10138/152337. Luettu: 8.1.2025.

Piktochart 4.9.2024. What Color is Dark Cyan? Meaning, Code & Combinations Luettavissa: https://piktochart.com/tips/what-color-is-dark-cyan. Luettu: 15.3.2025.

Piiroinen, S. & Järvelä, K. 2006. Kokemuksella ja tiedolla: tutkimus kuluttajien ruoan valinnasta. Julkaisuja 8/2006. Kuluttajatutkimuskeskus. Luettavissa: http://hdl.handle.net/10138/152255. Luettu: 8.1.2025.

React Native s.a. React Native – Learn once, write anywhere. Luettavissa: https://reactnative.dev/. Luettu: 3.4.2025.

React Native 14.4.2025. Get Started with React Native. Luettavissa: https://react-native.dev/docs/environment-setup. Luettu: 3.4.2025.

Ryanair Ltd v. PR Aviation BV 2015. Unionin tuomioistuimen tuomio (toinen jaosto). C-30/14, EU:C:2015:10. Ennakkoratkaisupyyntö. Alankomaat. Luettavissa: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:62014CJ0030. Luettu: 27.4.2025.

Segar, G. 29.2.2024. How to find and use hidden APIs to automate processes. Aatt.io. Luettavissa: https://aatt.io/newsletters/how-to-find-and-use-hidden-apis-to-automate-processes. Luettu: 11.5.2025.

SOK s.a. S-ryhmän digitaalisten palveluiden käyttöehdot. Luettavissa: https://www.s-kanava.fi/ehtojen-infosivu/digitaalisten-palveluiden-kayttoehdot/. Luettu: 1.5.2025.

Steckiw, M.J. 30.1.2022. Demystifying Service Design: Decoding the Terminology. Think Design Manage. Luettavissa: https://www.thinkdesignmanage.com/post/decrypting-service-design-terminology. Luettu: 24.2.2025.

Stanley, EJ. s.a. a. What is Data Scraping and How to Use It: A Complete Guide. Luettavissa: https://automate.fortra.com/resources/guides/what-is-data-scraping-and-how-use-it. Luettu: 24.4.2025.

Stanley, EJ. s.a. b. What is Web Scraping? A Complete Guide. Luettavissa: https://automate.fortra.com/resources/guides/what-is-web-scraping. Luettu: 24.4.2025.

Tilastokeskus 19.2.2024. Inflaatio palautunut maltilliselle tasolle – tammikuussa inflaatio oli 3,3 %. Uutinen. Luettavissa: https://stat.fi/fi/uutinen/inflaatio-palautunut-maltilliselle-tasolle-tammikuussa-inflaatio-oli-33. Luettu: 7.1.2025.

Visual Studio Code s.a. Your code editor. Luettavissa: https://code.visualstudio.com/. Luettu: 20.4.2025.

Zhou, L. & Wong, A. 2004. Consumer Impulse Buying and In-Store Stimuli in Chinese Supermarkets. Journal of International Consumer Marketing, 16, 2, s. 37–53. Luettavissa: https://doi.org/10.1300/J046v16n02_03. Luettu: 12.1.2025.