
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ELIAS HATTUKANGAS

# Verkkomaksujen tekniikka

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN TUTKINTO-OHJELMA
2022

| Tekijä<br>Hattukangas, Elias | Julkaisun laji<br>Opinnäytetyö, AMK | Päivämäärä<br>maaliskuu 2022 |
|---|---|---|
| | Sivumäärä<br>30 | Julkaisun kieli<br>Suomi |

Julkaisun nimi
**Verkkomaksujen tekniikka**

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikka


Tiivistelmä

Verkkomaksujen tekniikka -opinnäytetyö alkoi omasta mielenkiinnosta aiheeseen, jonka seurauksena päätin opiskella aihetta ja tehdä verkkokauppasovellus.

Työn käytettyihin tekniikoihin kuului mm. JavaScript, HTML, CSS, CommerceJS, ReactJS ja Stripen-rajapintoja. Työssä integroitiin ReactJS sovellukseen CommerceJS sovelluspohjaa ja Stripen verkkomaksurajapintoja.

Työssä tutkittiin verkkomaksun toteuttamiseen liittyviä teknisiä käsitteitä ja niiden hyödyntämistä verkkokauppasovelluksessa.

Tutkimustyössä tutkittiin eri lähteistä saatua tietoa, joiden perusteella työn sovellus tehtiin.

Työ onnistui tavoitteiden mukaisesti, jonka tuloksena valmistui toimiva verkkokauppa, johon saa helposti lisättyä tuotteita ja vaihdettua verkkomaksupalvelun tarjoajaa helposti tarpeen vaatiessa. Verkkokauppasovellus seurasi työssä opittuja verkkomaksuun liittyviä aiheita.



Avainsanat
Rajapinta, Käyttöliittymä, Verkkomaksaminen, Verkkokauppa

Author
Hattukangas, Elias

Type of Publication
Bachelor's thesis

Date
March 2022

Number of pages
30

Language of publication:
Finnish

Title of publication
**Technique of electronic payments**

Degree programme
Electrical and automation technology

Abstract

Technique of electronic payments thesis began from my own interest for the subject. My interest for the subject was the reason I began to study about e-payments and decided to make webstore software.

The techniques thesis' used included JavaScript, HTML, CSS, CommerceJS, ReactJS and Stripe interfaces. The ReactJS software got integrated with CommerceJS software base and Stripe e-payment interfaces.

I researched technical subjects about e-payments and their usage for webstore software.

In thesis the knowledge studied was based from various different sources. The knowledge from the sources were used to make the webstore software.

The thesis work was succesful according to the goals which were to make working web-store in which it is easy to add products and change e-payment interface services according to the needs. The webstore followed the information learned from the sources used in this thesis.

Keywords
Interface, User Interface, Paying Online, Electronic Commerce

SISÄLLYS

# 1 JOHDANTO

Opinnäytetyöni käsittelee verkkomaksamisen toteutusta ja sen teknistä osuutta. Tavoitteenani oli tutkia verkkomaksamisen tekniikkaa, jonka vuoksi aloitin tekemään itsenäistä opinnäytetyötä koululle.

Halusin lähteä tutkimaan verkkomaksamisen toteutusta, koska se on minulle erittäin läheinen asia. Itse hyödynnän verkkomaksamista arjessani melkein päivittäin. Satakunnan ammattikorkeakoulu antoi minulle opinnäytetyön aiheeni.
Opinnäytetyöaiheeni kiinnostaa minua ja minä otin aiheen mielenkiinnolla vastaan. Verkkomaksaminen on tekniseltä osaamiseltaan erittäin tuntematon. Tämä työ on antanut minulle hyvän mahdollisuuden tutustua lähemmin aiheeseen ja opettanut käyttämään hyväksi koulussa opittuja tekniikoita verkkomaksamisen toteutuksessa.

Luku 2 kertoo verkkomaksamisesta yleisesti ja luku 3 kertoo erilaisten rajapintojen eroista ja käytöistä. Luku 4 kertoo käyttöliittymistä ja sen tärkeydestä opinnäytetyöhön tehdystä verkkokauppasovelluksesta. Luku 5 käy läpi kyseisen sovelluksen toteuttamisesta tarkasti.

# 2 VERKKOMAKSAMINEN

## 2.1 Yleistä

Yhteiskunta on täynnä sähköistä dataa. Joka hetki rahaa siirtyy sähköisesti. Euroopassa oli liikkunut yli 100 miljardia euroa sähköisesti viime vuonna. (European Central Bank, 2022.)
Useat miljoonat ihmiset käyttävät sähköisiä palveluita, jotka hyödyntävät verkkomaksamista. Verkkomaksamisen kuluihin liittyvät muun muassa maailmanlaajuinen kaupankäynti, sekä useat eri viihdepalvelut. Näitä palveluita voidaan maksaa monilla tavoilla, kuten puhelinmaksuilla ja pankkimaksuilla.

Suuri käyttö verkkomaksuissa kuitenkin vaatii hyvät maksujärjestelmät. Maksujärjestelmien kautta verkkomaksamiset ympäri maailmaa sujuvat hyvin. Verkkomaksuissa on monenlaisia uhkia, jotka estävät verkkomaksamisen toteutusta.
Tällaisiin uhkiin kuuluu digitaalisen rahan säilytys, rahaliikenteen uhat, kuten phishing eli verkkourkinta ja henkilökohtaisten tietojen leviäminen.
Näiden syiden takia lähdin tutkimaan verkkomaksamiseen liittyvää toteutusta ja siihen liittyvää turvallisuutta. Tässä opinnäytetyössä kuvailen verkkomaksamisen toteutuksen tärkeitä teknillisiä osioita, kuten mm. rajapintojen tärkeydestä ja niiden toiminnasta sekä niiden osuudesta verkkomaksamisessa.

## 2.2 Sähköiset maksutavat

Tässä osiossa käydään läpi yleisiä verkkomaksutapoja, joita hyödynnetään verkkomaksamisessa. Näiden maksutapojen ymmärtämistä varten on tärkeää tietää, miten omat käytettävät verkkomaksutavat toimivat ja miten niissä voidaan käyttää verkkomaksuun liittyviä tekniikoita.

### 2.2.1 Laskut

Verkkomaksamisessa voidaan hyödyntää kahta erilaista laskutustapaa: sähköiset laskut ja verkkolaskut.

Sähköiset laskut ovat perinteisiä paperilaskuja, mutta sähköisessä muodossa ja verkkomaksut ovat laskuja, jotka voidaan liittää verkkopankkiin.
Sähköinen lasku saadaan aikaan ottamalla laskun tiedot talteen, jonka jälkeen tiedot muutetaan sovelluksella tiedostomuotoon ja tämä voidaan lähettää eteenpäin. Yleisimpiä tiedostomuotoja sähköisille laskuille on portable document format (.pdf), Word (.doc) sekä OpenDocument (.odt) -päättyiset tiedostot. (Honkala, P. 2008.)

Sähköinen lasku voidaan lähettää asiakkaalle omaan henkilökohtaiseen sähköpostiin. Tämä mahdollistaa erittäin nopean laskun lähetysajan.
Verkkolaskun ero sähköisiin laskuihin on se, että verkkolaskun saa suoraan verkkopankkiin maksettavaksi. Verkkomaksu tulee mahdolliseksi, kun yritykset ja pankit sopivat asiasta.

Verkkolaskun käyttäminen vaatii verkkopankkitunnuksia. Verkkolaskut ovat turvallinen tapa maksaa verkossa asioista, koska maksutapahtuma tapahtuu verkkopankin suojatussa ympäristössä. Tässä tilanteessa pankki toimii sovitun kaupan kolmantena osapuolena, joka huolehtii molempien osapuolien tunnistautumisen.

### 2.2.2 Gateway-palvelut

Gateway-palvelut ovat yleensä verkkomaksupalveluun erikoistuvia yrityksiä, jotka myyvät heidän omaa rajapintaansa. Tämä rajapinta mahdollistaa käyttäjän maksamisen verkkokaupassa eri maksutavoin, kuten verkkopankin tai laskun kautta. Tästä asiasta kerron tarkemmin luvussa 3.

Gateway-palvelun maksutavat ovat yritysten ja pankkien kanssa sovittuja sopimuksia. Tämän takia kaikki gateway-palvelut eivät sisällä kaikkia maksutapoja, koska kaikki yritykset eivät ole tehneet kaikkien pankkien kanssa sopimusta. Tämä rajoittaa paljon verkkokaupoissa maksamista.
Näiden kolmannen osapuolen gateway-palvelun tarjoaja lähettää rajapinnan kautta tietoa maksuprosessista ja ilmoittaa menikö maksu läpi. Tätä tietoa seurataan rajapinnan kautta, jolle ilmoitetaan maksutietojen tilasta.

### 2.2.3 Mobiilimaksaminen

Mobiililla voidaan maksaa mobiilien maksusovellusten ja verkkopankkien kautta tai tekstiviesteillä mobiilidataa käyttäen. Tekstiviesteillä maksaminen tapahtuu, kun asiakas ilmoittaa tiettyyn numeroon koodin. Viesti menee tekstiviestiyhdyskäytävään, mikä käsittelee viestin. Tekstiviestiyhdyskäytävä on rajapinta, joka suorittaa maksun puhelimen operaattorin kautta. Jos operaatio menee onnistuneesti läpi, silloin lisätään tuotteen hinta mobiilidatan hintaan tai otetaan nykyisestä mobiilidatasta pois. Operaation jälkeen annetaan tuote asiakkaalle. Suuri ongelma mobiilimaksamisessa on gateway-palvelu, joka voi viedä ylimääräistä rahaa asiakkaalta. Tätä asiakas ei välttämättä tiedä. (Honkala, P. 2008.)

Mobiilimaksaminen voi myös olla lähimaksamista, jolloin maksaminen tapahtuu viemällä puhelin kassan maksupäätteen viereen. Kaupoissa toimivat mobiilimaksut perustuvat lähimaksuominaisuuteen, jotta voi maksaa mobiilimaksusovelluksella. Puhelimessa tulee siis olla NFC-lähimaksuominaisuus. Mobiilimaksamisella voidaan tarkoittaa myös tilisiirron tekemistä sovelluksen avulla. Myös verkkokaupoissa ostosten maksaminen voi tapahtua mobiilimaksulla, hyväksymällä ostos sovelluksen sisällä. (Ketonen, J. 2019)

## 2.3 Toteamus

Suurin osa verkkomaksamisesta vaatii kolmannen osapuolen toteuttamaan verkkomaksun teknistä osaamista eli gateway-palvelun sekä rajapintasovelluksen. Usein maksutapahtuman toteuttaa pankki, josta on päätetty toteuttaa kyseinen verkkomaksu. Nämä verkkomaksut toteutetaan usein kokonaan pankkien suojatussa verkossa. Jos on sen sijaan kyseessä gateway-maksupalvelu toiselta yritykseltä, silloin maksupalvelu viedään suojattuun pankkipalveluun, jossa maksutapahtuma toteutetaan. Olemassa on useita erilaisia vaihtoehtoja, miten voi suorittaa verkkomaksamista.

# 3 RAJAPINNAT

Tässä kappaleessa esittelen erilaisia rajapintoja ja niiden ominaisuuksia. Kappaleen lopussa esitän osion, jossa päättelen, mitä rajapintaa käytän tekemääni verkkomaksusovellusta varten.

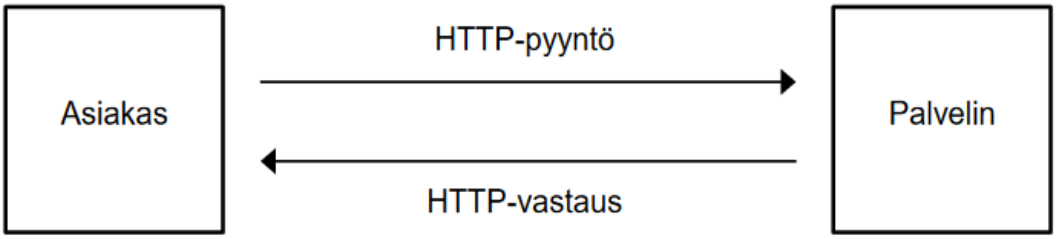
Rajapintoja on monenlaisia, joilla voi olla erittäin pieniä eroja. Erilaisten rajapintojen suurimmat erot ovat logiikassa ja siinä, miten ne on rakennettu. Millainen kieli ja mihin tarkoitukseen rajapinta on tarkoitettu. (avoinrajapinta.fi, 2014.)

Nämä rajapinnat, joita voitaisiin hyödyntää verkkomaksamisen toteutuksessa ovat REST-rajapinnat, SOAP-rajapinnat ja RPC-rajapinnat.
Rajapinnat ovat suunniteltu avustamaan toisia sovelluksia, vaikka rajapinnat voivat itsessäänkin olla yksin toimivia sovelluksia. Rajapinnat ovat yksinkertaisuudessaan sovelluksia, jotka tekevät yhtä tiettyä toimintaa. Rajapintoja kutsutaan toisista sovelluksista, silloin kun näitä toimintoja tarvitaan tai halutaan käyttää.

### 3.1.1 Ohjelmointirajapinta

Ohjelmointirajapinta on järjestelmä, joka välittää tietoa tai palveluita muille sovelluksille. Ohjelmointirajapinnat voivat olla datapohjaisia tai toiminnallisia. Datapohjaiset rajapinnat siirtävät dataa tai lukevat dataa toisiin palveluihin.

Toiminnalliset rajapinnat sen sijaan tarjoaa datan muuttamisen mahdollisuuksia. Nämä rajapinnoilla voidaan tehdä algoritmeja tai muuttaa dataa, jota on syötetty rajapinnalle. Ohjelmistorajapinnat seuraavat nykypäivän koodien sääntöä eli ne voivat olla yksityisiä tai avoimia rajapintoja. (Kankaanpää, S., 2016.)
Avoin rajapinta tarkoittaa rajapintaa, jota saa vapaasti käyttää omiin tarkoituksiin ilman rajapinnan tekijän lupaa. Avoimissa rajapinnoissa dokumentaatiot ja ominaisuudet täytyvät olla myös vapaasti saatavilla rajapinnan käyttäjille.
Yksityinen rajapinta on rajapinta, jota ei ole vapaasti saatavilla ja sen käyttöä rajoitetaan. Yleensä yksityiset rajapinnat ovat yritysten omia rajapintoja, joita käytetään yrityksen omiin toimintoihin. (Kankaanpää, S. 2016.)




Kuva 1. HTTP-rajapinnan yksinkertainen toiminta. (Kankaanpää, S. 2016)

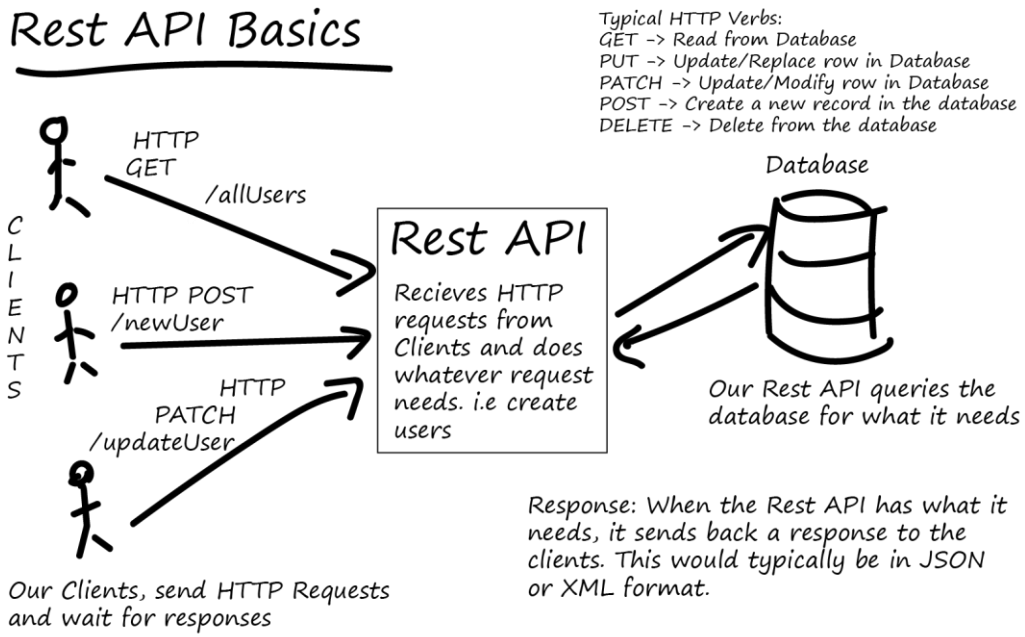
### 3.1.2 REST-rajapinta

REST eli Representational State Transfer on web-pohjainen arkkitehtuurimalli sovellusten väliseen kommunikointiin. REST on yleisimmin käytetty rajapintamalli internetin välityksellä kommunikoiville laitteille. REST-arkkitehtuuri täyttää universaalit arkkitehtuuri vaatimukset, mikä mahdollistaa useamman ohjelmointikielen käyttämisen saman REST-rajapinnan kanssa.

REST pohjautuu tilattomaan kommunikointiprotokollaan. Tilaton kommunikointi tarkoittaa kommunikointia, jossa pyyntö lähetetään sovellukselle ja sovellus antaa vastauksen sen hetkisen tilan mukaisesti. REST-rajapintaa voi käyttää esimerkiksi HTTP-protokolla ja silloin kaikki HTTP-protokollaan kuuluvat voivat käyttää kyseisiä metodeja. Tarkoittaen, että rajapinnalla pystytään hakemaan, lisäämään, poistamaan sekä päivittämään rajapinnan sisältämää dataa. (Jäppilä, M. 2020.)

REST-arkkitehtuurityylissä jaetaan arkkitehtuuriset elementit kolmeen eri pääluokkaan: komponentteihin, liittimiin ja dataelementteihin. Komponenteilla tarkoitetaan ohjelmistoja, joilla on tarve kommunikoida muiden komponenttien kanssa. Liittimet mahdollistavat komponenttien kommunikoinnin. Dataelementit puolestaan ovat tietoa, jota komponentit siirtävät tai muokkaavat liittimien avulla. (Fielding 2000, 86-97.)

REST-tyyli pyrkii parantamaan arkkitehtuurisia ominaisuuksia, kuten suorituskykyä ja skaalavaisuutta, sekä REST-tyylin yksinkertaisuutta.




Kuva 2. REST-rajapinnan toiminta periaate yksinkertaistettuna englanniksi. (TutorialEdge. 2017)

Tilattomuudella tarkoitetaan, miten asiakkaan ja palvelimen tulisi kommunikoida. REST edellyttää, että näiden kommunikointi olisi tilatonta, jonka takia kaikki pyynnöt tulisi sisältää kaikki tarvittava informaatio. REST-pohjainen palvelin siis käsittelee kaikki pyynnöt yksittäisinä kokonaisuuksina. (Fielding 2000, 81–82.)

REST erottuu muista rajapinta-arkkitehtuurityyleistä neljän määritellyn rajoitteen mukaisesti. Nämä rajoitteet edellyttävät asiakkaan ja palvelimen välisten rajapintojen tulisi toimia aina yhdenmukaisesti riippumatta siitä, mitä tietoa niiden välillä kulkee. Kaksi ensimmäistä rajoitetta käsittelevät rajapinnan tietojen tunnistamista ja niiden manipulointia. Kolmas rajoite vaatii viestit, jotka sisältävät kaiken informaation, ja neljäs käsittelee hypermedian käyttöä sovelluksen tilakoneena. (Fielding 2000, 81–82.)

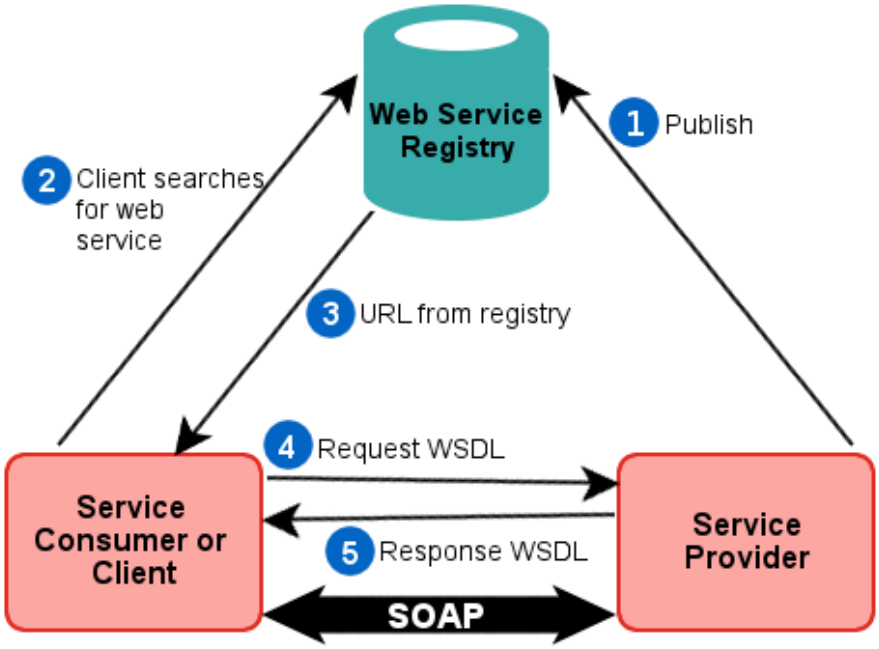
### 3.1.3 SOAP-rajapinta

SOAP on lyhenne sanoista: Simple Objects Access Protocol, mikä tarkoittaa kommunikointiprotokollaa. SOAP alun perin suunniteltiin ja kehitettiin Microsoftille. Nykypäivinä sitä käytetään verkkosovellusten väliseen kommunikointiin.
SOAP on samanlainen kuin REST, sillä molemmat välittävät dataa HTTP-protokollaan. SOAP koostuu neljästä osasta: SOAP-kirjekuoresta, tiedon koodaustavasta, RPC:stä ja HTTP:sta. (Vettainen, J. 2022)

Verrattuna kuitenkin REST-arkkitehtuuriin, SOAP voi ainoastaan käyttää dataa XML-muodossa. Tämän seurauksena SOAP-arkkitehtuurin viestirakenteet ja koodaussäännöt ovat erittäin vahvasti standardoituja. SOAP-tyylissä määritellyt viestit ovat kuitenkin yhdensuuntaisia tiedonsiirtoja, joista voidaan koota suurempia kokonaisuuksia, kuten pyyntö ja vastauspareja.

SOAP-tyylin määritellyt viestit ovat aina yhdensuuntaista tiedonsiirtoa, joista pystytään kokoamaan suurempia kokonaisuuksia. Tällaisia ovat esimerkiksi pyyntö- ja vastauspareja. SOAP ei itsessään välitä siitä, miten viestit tulisi välittää verkon kautta. Tiedonsiirtoon voidaan sen takia käyttää http:n sijaan esimerkiksi sähköpostia tai pelkkää TCP/IP:tä. SOAP-viestit kuljetetaan yleisimmin HTTP:n kautta, jonka takia SOAP määritteleekin HTTP-sidontaa. HTTP-sidonta kertoo, kuinka HTTP:tä tulisi käyttää viestien välittämiseen. (Järvinen 2002.)

SOAP-tyylissä palvelimen vastaus on pyynnön mukaisesti jaettu kahteen osaan. HTTP:n mukaisiin tietoihin, sekä SOAP-protokollan määrittelemään runkoon eli XML-dokumenttiin. Ensimmäinen rivi kertoo http-protokollan mukaisen tilakoodin, kuten esim. ”Maksu on onnistunut.” Pyynnössäkin runko-osan muodostaa SOAP-protokollan kirjekuori.
SOAP-viestejä voidaan kutsua myös kirjekuoriksi, mikä on aina XML-muotoinen tiedosto. SOAP-kirjekuori alkaa aina Envelope-elementillä, joka on pakollinen.

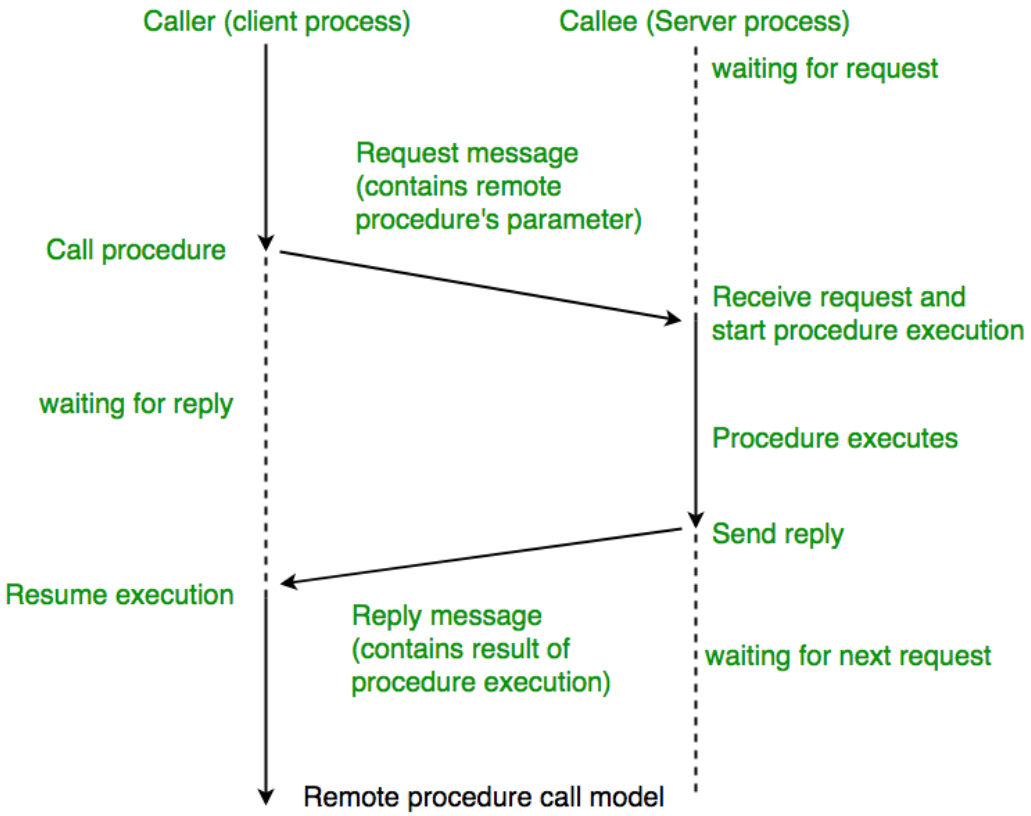



Kuva 3. SOAP-rajapinnan toiminta englanniksi. (Medium.com. 2019.)

### 3.1.4 RPC-rajapinta

RPC eli Remote Procedure Call:n ideana on välittää IDP:n ja TCP:n avulla verkon kautta proseduurin kutsuja argumentteineen ja palauttaa takaisin etäällä suoritetun proseduurin tulos. (Leppänen, V. 2012.)

RPC-arkkitehtuurissa kutsutaan etäistä proseduuria, jolle syötetään dataa. Proseduuri tarkoittaa toimintoa, joka suorittaa aina samaa tehtävää, toimintoja tai operaatioita. Etäinen proseduuri tarkoittaa proseduuria, joita ei suoriteta lokaalisesti eli paikan päällä. Yleisesti tämä tarkoittaa toista laitetta, jossa pääsovellusta ei pyöritetä. Etäinen proseduuri suorittaa halutun toiminnan ja palauttaa kutsujalle dataa.




Kuva 4. RPC-rajapinnan toiminta periaate englanniksi. (GeeksForGeeks. 2021)

## 3.2 Rajapinnan valinta

Rajapinnan valinnassa kuuluu ottaa huomioon rajapinnan tarkoitus. Eri arkkitehtuurityyliset rajapinnat toimivat eri lailla toisistaan ja niillä on omat vahvuudet eri toiminnoissa. Tässä opinnäytetyössä päädyin hyödyntämään kaikista yleisintä ohjelmointirajapintatyyliä, REST-tyyliä. REST sopii hyvin verkkokauppasovelluksen toteuttamiseen. Työssäni käytän REST-sovelluspohjaa. Tämä REST-sovelluksen yleisyys helpotti omaa päätöstäni verkkokauppasovelluksen sovelluspohjan ja rajapintatyylin valinnassa. REST-rajapintasovelluksessa helpottaa paljon verkkomaksuja HTTP-protokollien kautta. REST-rajapintojen kautta toimii opinnäytetyön verkkokauppasovelluksen datan hakeminen ja verkkomaksun toteutus.

# 4 KÄYTTÖLIITTYMÄ

Käyttöliittymä on sovellus, jonka kautta pystytään käyttämään sovellusta. Verkkomaksuissa käyttöliittymä on esimerkiksi ostokori ja maksamispainike. Käyttöliittymän tarkoitus on saada käyttäjä vuorovaikuttamaan ohjelman kanssa.
Käyttöliittymän on tärkeä sisältää ominaisuudet, mitä ohjelman käyttäjä tulee tarvitsemaan sovelluksen käytössä. Verkkomaksamisen toteutuksessa tärkeimpiä asioita, joita käyttöliittymän tulee sisältää, on pääsy verkkomaksamiseen, maksun toteutus ja maksun toteutumisen varmistus. Käyttöliittymän tärkeä ominaisuus on verkkomaksamisessa päästää käyttäjä käyttämään verkkomaksusovellusta.

## 4.1 Komentorivipohjaiset käyttöliittymät

Komentorivikäyttöliittymät (CLI) ovat yksinkertaisia tekstiperusteisia käyttöliittymiä, jotka ajavat ohjelmia ja hallitsevat tietokoneen tiedostoja. Komentorivikäyttöliittymä hyväksyy tekstikomentoja ja kehotteita sisääntuloina ja toimii komentojen perusteella. Suurin osa komentorivikäyttöliittymistä ovat perusohjelmia nykypäivän suurimmissa käyttöjärjestelmissä, kuten Windowsissa ja Linuxissa. (Loshin, P. 2021)

CLI hyväksyy monenlaisia eri komentoja ja ne sisältävät luonnollisesti tiettyjä komentoja. Jotkin ohjelmat myöskin sisältävät komentoja ja kehotteita, joita CLI pystyy ajamaan.

## 4.2 Menu-pohjaiset käyttöliittymät

Menu-perusteiset käyttöliittymät perustuvat sarjaan erilaisia näyttöjä. Nämä näytöt tekevät jonkinlaisen toiminnan tai funktion silloin kuin, niitä kosketaan tai klikataan. Esimerkiksi kun mobiililaitteesta klikataan asetuksia, se vie laitteen uudenlaiseen näyttöön. Menu-perusteiset käyttöliittymät ovat yksinkertaisia ja niitä voidaan hyödyntää monissa eri asioissa, kuten pankkiautomaattilaitteessa ja kännyköissä. (Falk, C. 2014)

## 4.3 Graafiset käyttöliittymät

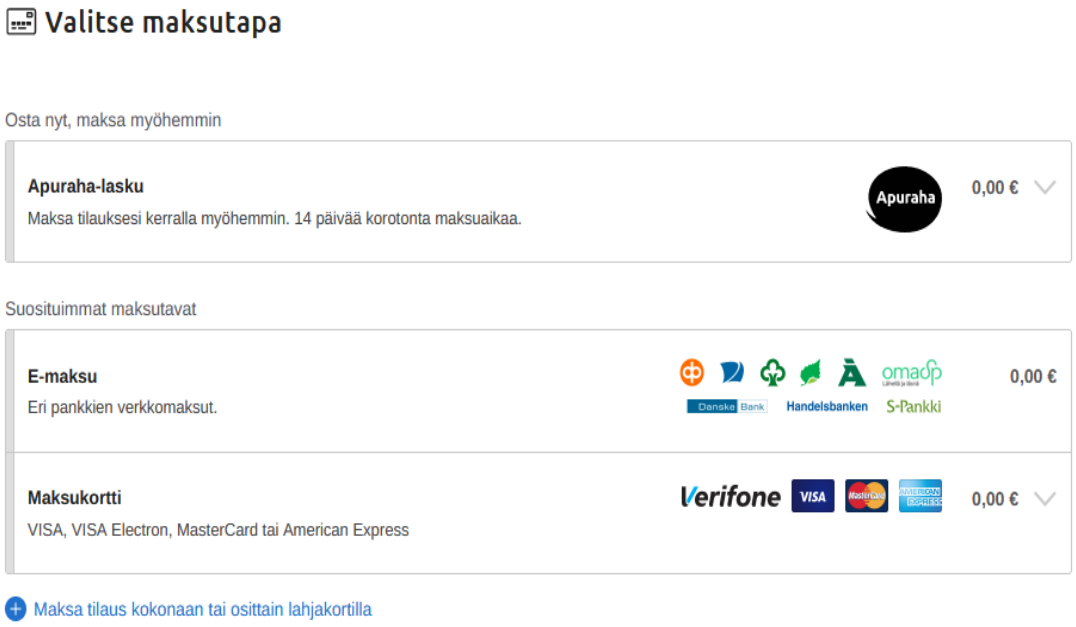
Graafiset käyttöliittymät (GUI) ovat interaktiivisia visuaalisia komponentteja ohjelmassa. Graafiset käyttöliittymät näyttävät objekteja, jotka näyttävät ja tuovat tietoa ja näyttää, mitä käyttäjä voi tehdä. (Computer Hope. 2021)

Graafiset käyttöliittymät sisältävät objekteja, kuten nappeja ja kuvia. Nämä graafiset elementit ovat välillä äänellä tai visuaalisilla efekteillä vahvistettuja. Graafisten käyttöliittymien tarkoitus on olla mahdollisimman käyttäjäystävällisiä, niiden objektien ja funktioiden kanssa. (Computer Hope. 2021)

## 4.4 Käyttöliittymän suunnittelu

Verkkomaksamisen käyttöliittymän suunnittelussa tulee ottaa huomioon helppokäyttöisyys, toimivuus sekä ulkonäkö. Käyttöliittymän suunnitellussa on otettava huomioon sovelluksen käyttäjäkokemus, sillä käyttöliittymä ja käyttäjäkokemus ovat molemmat tärkeitä elementtejä käyttöliittymän toteutuksessa. (Niskanen, J. 2020)

Kuvassa 5. näkyy esimerkki, miten verkkokauppa.com on suunnitellut heidän käyttöliittymänsä. Verkkomaksun valitseminen on jaettu eri kategorioihin, joista voidaan nappia painamalla valita haluttu maksutapa. Valitun maksutavan jälkeen nappi vie maksutiedot gateway-palveluun, jossa suoritetaan maksu.




Kuva 5. Verkkokauppa.com verkkomaksu vaihtoehdon valinta. (Verkkokauppa.com 2022)

Käyttöliittymän suunnittelussa täytyy miettiä verkkomaksusovelluksen toteutusta ja toimintoja, jos verkkomaksamiseen halutaan liittää ostokori. Täytyy silloin miettiä ostokorin toteutusta ohjelmallisesti, sekä sen toteutusta käyttöliittymään.

Suunnittelussa voi myös miettiä, että halutaanko käyttää valmista yksityistä tai avointa ohjelmointipohjaa. Kuuluisa verkkokauppasovelluspohja WooCommerce on yksi mahdollinen yksityinen verkkokauppapohja, jolla voidaan muokata ja lisätä käyttöliittymään asioita. Kuitenkaan ei ole pakko käyttää valmiita ohjelmointipohjia, sen sijaan voidaan luoda oma käyttöliittymä verkkomaksamista varten.

Käyttöliittymän tekemiseen voidaan käyttää monenlaisia eri ohjelmointikieliä, kuten HTML, JAVA, C# ja Python. Monella erilaisella ohjelmointikielillä pystytään tekemään käyttöliittymiä omaan tarkoitukseen.

# 5 TOTEUTUS

Opinnäytetyön toteutuksena päätin tehdä verkkokauppapohjan, jossa hyödynnetään rajapintoja sekä olisi mukavan näköinen käyttöliittymä.
Tärkeänä osana toteutusta oli saada toimiva ostokori ja maksusuoritus, sekä tuotteiden haku tietokannasta rajapinnan kautta. Ostokorin toimintana täytyy tehdä sovellukselle tuotteita ja siirtää haluttu määrä ostokoriin. Ostokoria tehdessä tärkeänä osana on saada tallennettua muistiin lisätyt tuotteet. Ostokorista kuuluu sen jälkeen päästä siirtymään ostamaan valitut tuotteet.
Tätä varten täytyi tehdä tuoteobjektit ja ostokoriobjekti. Tuote-objekteja pitää pystyä lisäämään haluttu määrä ostokoriobjektiin. Ostokoriobjektin pitää pystyä viemään sen sisältämät tuoteobjektit rajapinnalle, josta suoritetaan maksaminen.

Maksusuorituksen suunnittelussa oli tärkeää saada ostokorinobjektit maksatettavaksi ja liittää maksu gateway/verkkomaksu-palveluun, jossa verkkomaksu voidaan suorittaa. Näiden pohjalta saadaan tehtyä toimiva verkkokauppa ja verkkomaksupalvelu.

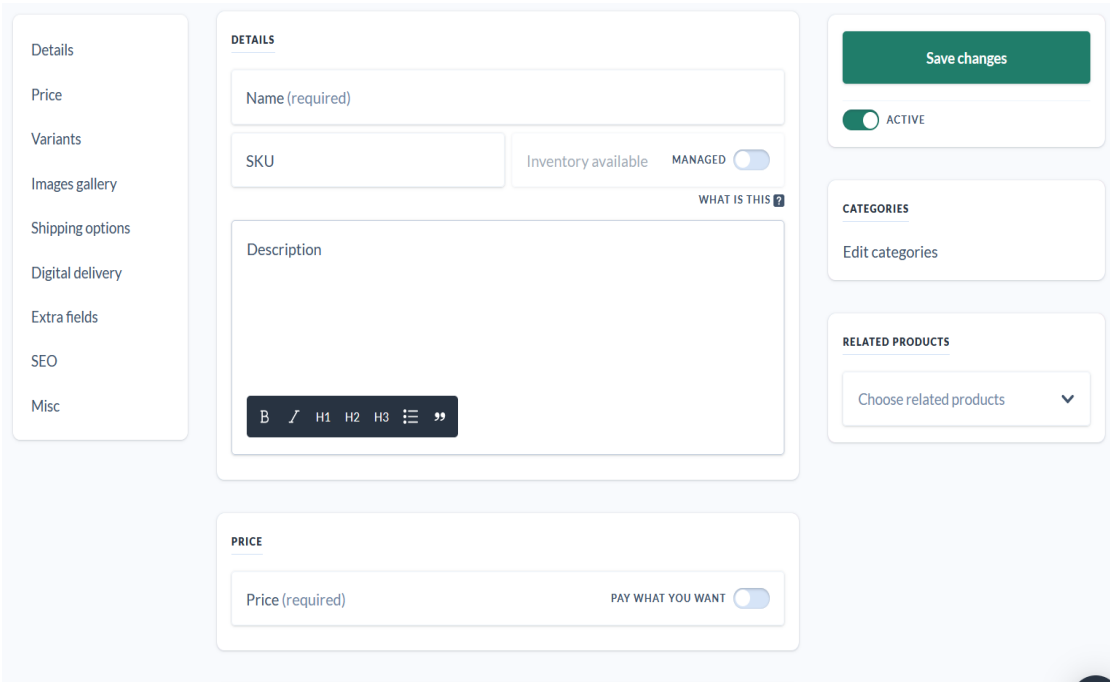
## 5.1 Alustus

Lähdin miettimään näihin toimintoihin ratkaisuja ja parasta ohjelmointikieltä. Pääasiassa vaihtoehtoina oli PHP ja JavaScript. PHP ja JavaScript toimisivat molemmat hyvin verkkomaksujärjestelmän skripteihin ja toteutukseen.
Päädyin tekemään sovellusta JavaScriptillä, koska se on tutumpi ohjelmointikieli ja sen asiakaspuoleiset toiminnot sopivat omaan käyttöön paremmin. Päädyin nimenomaan käyttämään JavaScriptin kanssa ReactJS JavaScript pohjaa, jossa tehdään sovelluksesta komponenttipohjainen.

Komponenttipohjainen sovellus tarkoittaa sitä, että sovelluksen objektit ovat komponentteja ja näitä komponentteja voidaan käyttää hyvin uudestaan, esimerkiksi verkkokaupan tuotteiden tekeminen on paljon yksinkertaisempaa, koska sovellukseen tehdyt tuotteet voidaan kaikki tehdä samasta komponentista.

Päädyin käyttämään CommerceJS sovelluspohjaa, joka toimii tuotteiden ja tilausten tietokantana, josta saadaan haettua kyseiset tiedot, sekä CommerceJS:n rajapintaa JavaScriptin ja ReactJS:n kanssa. CommerceJS:n rajapinta toimii REST-rajapintana verkkomaksupalvelun tarjoajan rajapintojen kanssa, joista voidaan valvoa verkkokaupan toimintaa. Päätin käyttää CommerceJS, koska sinne voidaan tehdä kätevästi verkkokaupan tuotteet, se on helppo liittää sovellukseen verkkoavaimen avulla.

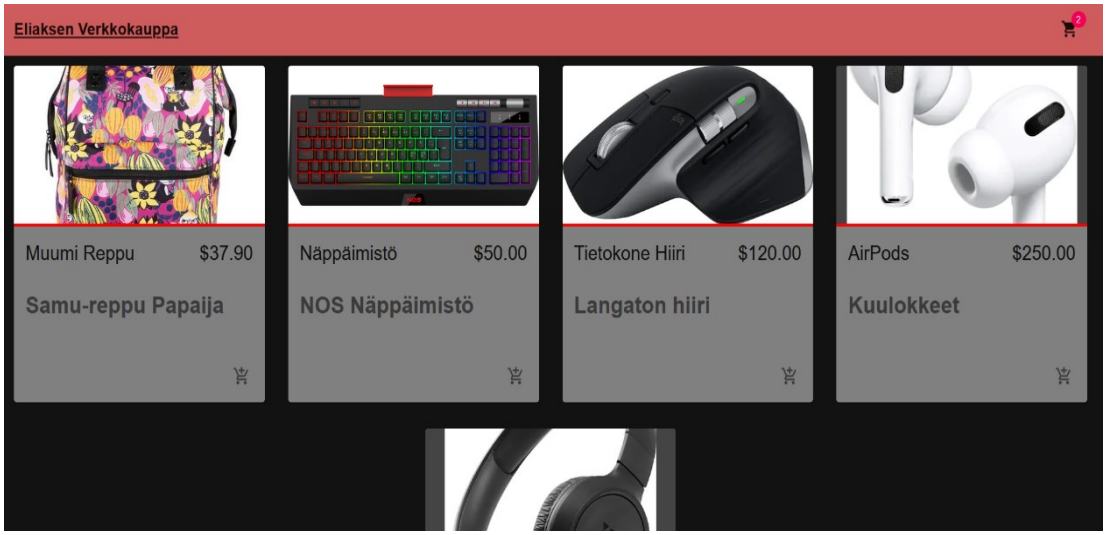
Sovelluksen koodaamiseen käytin Visual Studio Codea. Sovellukseen latasin Reactin ja Node.js. Node.js tuli sovelluksen teossa, käytin Reactia lataamisessa. Reactin asensin sovelluksen kansioon PowerShell-koodilla:" npx create-react-app my-app". Tämän jälkeen latasin sovelluksen kansioon muita hyödyllisiä sovelluspohjia, kuten @material-ui/core ja @chec/commerce.js.  Nämä ovat vapaita lähdekoodeja, jotka auttavat tätä kyseistä sovellusta paljon. Esimerkiksi @material-ui/core sisältää Reactin komponentteja ja @chec/commerce.js sisältää CommerceJS:n komponentteja.




Kuva 6. CommerceJS. Malli CommerceJS:n tuotteen teosta. (CommerceJS 2022)

## 5.2 Sovelluksen toteutus

Tämän sovelluspohjan liitäntä omaan sovellukseen toteutui helposti. CommerceJS:stä voidaan hakea tiedot lisäämällä environment variables -liitäntäkoodin.
Liitäntäkoodilla saadaan CommerceJS-rajapinnan sisältämät tiedot haettua käyttöön. Nämä tiedot sisältävät muun muassa tuotteiden ja tilausten tiedot. Nämä haetut tiedot sen jälkeen liitetään verkkokauppasovelluksessa tuotettuihin tuotteisiin. Jokaiselle tuotteelle on tehty oma medialaatikko, joka tulee sisältämään kaikki olennaiset tuotteiden tiedot käyttäjän tarkastelua varten.




Kuva 7. Verkkokaupan etusivut.

Tuotteiden hakeminen verkkokauppasovellukseen tapahtuu kuva 7:n mukaisesti. JavaScript koodilla haetaan CommerceJS-rajapinnasta kaikki tiedot aktiivisista tuotteista, eli tuotteet, joita halutaan verkkokauppasovelluksessa näyttää. CommerceJS sisältää kaikki sinne tehdyt tuotteet. Tämän jälkeen tuotteet laitetaan sovelluksen sivuille omaan mediapalkkiin, joista ne voidaan lisätä ostokoriin painamalla ostokorinappia tuotteen oikeassa alakulmassa. Ostokoriin siirtymistä varten tehtiin yläkulmaan ostokoriin siirtonappi, josta voidaan helposti siirtyä ostokoriin.

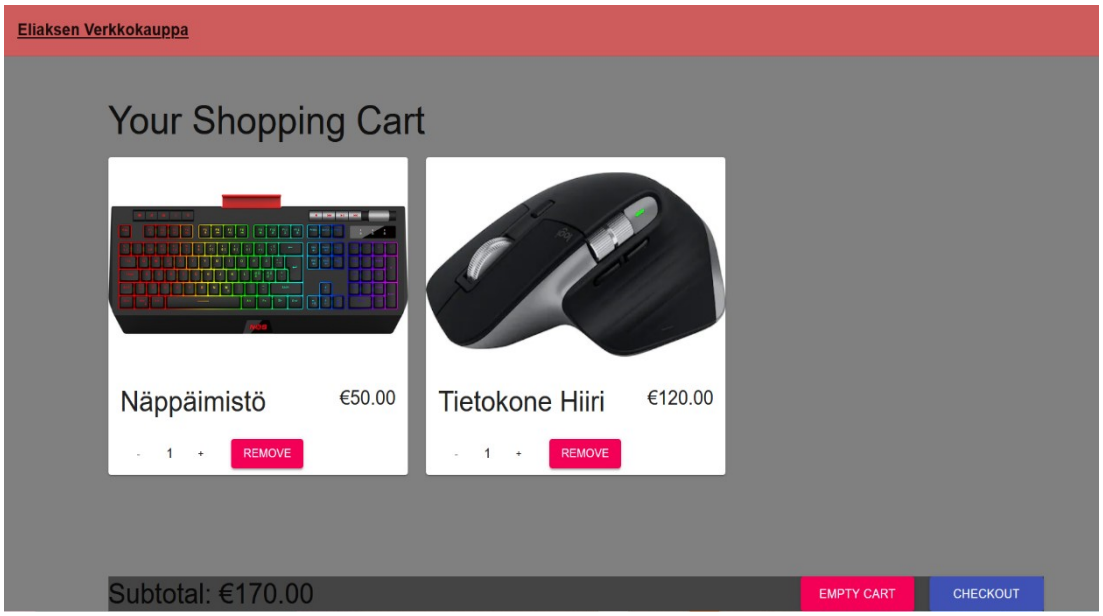
```
return (
  <Card className={classes.root}>
    <CardMedia className={classes.media} image={product.image?.url} title={product.name} />
    <CardContent>
      <div className={classes.cardContent}>
        <Typography gutterBottom variant="h5" component="h2">
          {product.name}
        </Typography>
        <Typography gutterBottom variant="h5" component="h2">
          ${product.price.formatted}
        </Typography>
      </div>
      <Typography dangerouslySetInnerHTML={{ __html: product.description }} variant="body2" color="textSecondary" component="p" />
    </CardContent>
    <CardActions disableSpacing className={classes.cardActions}>
      <IconButton aria-label="Lisää Ostokoriin" onClick={handleAddToCart}>
        <AddShoppingCart />
      </IconButton>
    </CardActions>
  </Card>
);
```

Kuva 8. Tuotteiden haku JavaScriptillä

Tuotteiden jälkeen tein ostokorin, johon saataisiin siirrettyä tuotteiden tiedot. Tämä saatiin aikaan lisäämällä nappi, jota painaessa kyseisen tuotteen data lisätään ostokoriin.

Ostokoria varten tein kolme eri funktiota: koko ostokorin tyhjennys, tuotteen poisto ostokorista ja checkout.

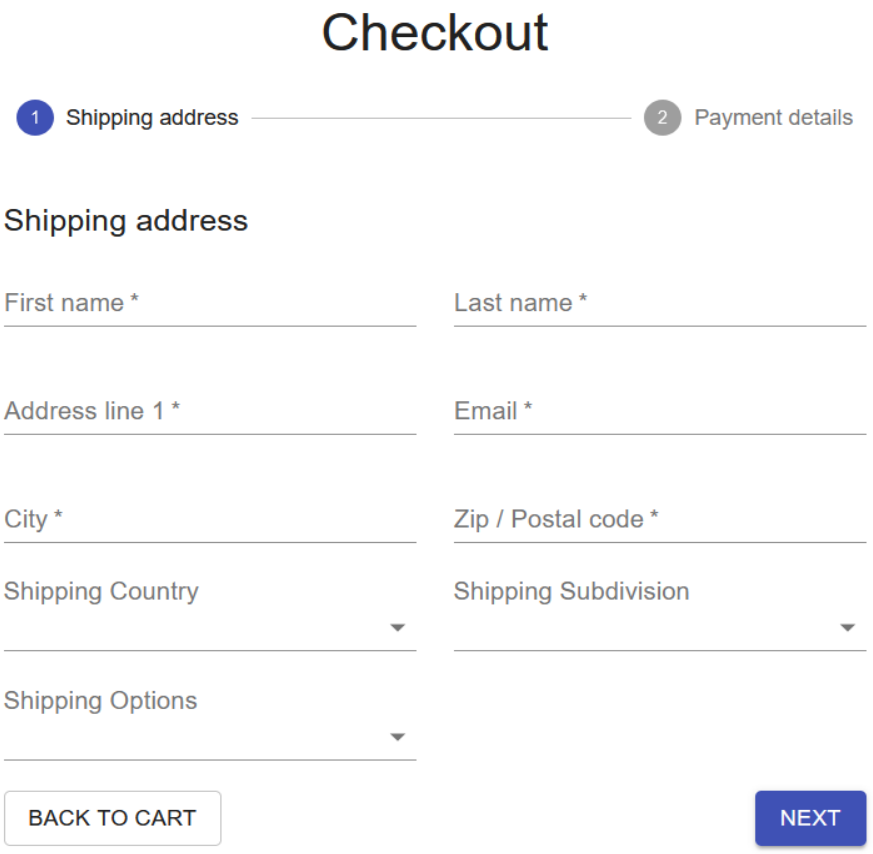
Ostokorin kokonainen tyhjennys saatiin aikaan helposti tyhjentämällä koko ostokorin sisältämä data. Yhden tuotteen poisto ostokorista tapahtui samalla tavalla, kuin sen lisääminen sinne, mutta datan lisäämisen sijaan se poistetaan sieltä. Checkout-funktio vie ostokorin tuotteet maksutapahtuman toteutukseen ja tekee ostokorin tuotteista tilauksen.




Kuva 9. Verkkokauppasovelluksen ostokori

Ostotapahtumaa varten täytyy saada asiakkaan tiedot, kuten nimi ja osoite, jonne tuote lähetetään. Tämän takia checkout-osio pyytää tarvittuja tietoja ennen kuin siirrytään maksutoteutukseen.

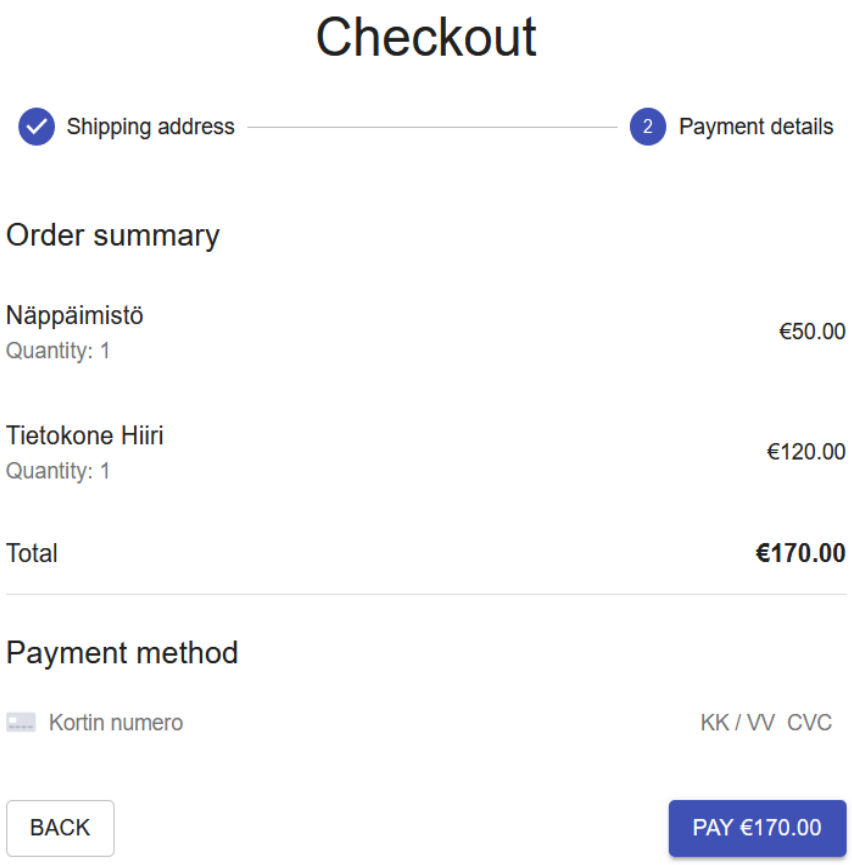
Tässä kohdassa voidaan tarkistaa tuotteiden saatavuus haluttuun osoitteeseen ja maahan. Nämä tiedot ovat määritetty CommerceJS rajapinnan tuotteiden tietojen haun yhteydessä, jonka kautta rajapinta tietää voidaanko niitä kuljettaa. Jos kuljetus onnistuu, niin ilmoittaa sovellus tulevista postikuluista. Tästä kohdasta sovellusta voidaan palata vielä takaisin ostokoriin, jos halutaan tai vaadittujen tietojen täyttyessä siirtyä maksamaan ostokorin tuotteet.




Kuva 10. Verkkokauppa sovelluksen lähetyksen toimitus tiedot.

Seuraavaksi verkkokauppasovelluksessa siirrytään verkkomaksamiseen. Verkkomaksamista varten sovellukseen on lisätty vain pankkimaksaminen. Pankkikorttitietojen täyttämisen jälkeen sovellus vie nämä tiedot gateway-palveluun, jossa verkkomaksu toteutuu.

Verkkomaksun toteutumisen jälkeen sovellus antaa tiedon onnistuiko verkkomaksu vai ei. Jos verkkomaksu on onnistunut, saa CommerceJS-rajapinta tiedon onnistuneesta tilauksesta verkkokauppasovelluksen kautta ja lisää tilaus tiedot myyjälle rajapinnan kautta. Myyjä saa sen suorittaa CommerceJS:n sisällä halutut toiminnot, kuten merkata tilauksen lähetetyksi.




Kuva 11. Verkkokauppasovelluksen maksutiedot.

Gateway-palveluksi verkkokauppasovellukseen valitsin Stripen, koska Stripestä oli opiskelijaprojekteihin valmiiksi gateway-testipalvelimia. Stripen lisääminen tapahtui samalla tavalla, kuin CommerceJS:n, eli liittämällä Stripen ympäristömuuttujat sovellukseen.

Molemmissa rajapinnoissa ympäristömuuttujat lähettävät sovellukselle tietoja ja sovellus lähettää rajapinnoille tietoja.

Verkkomaksun toteutuksessa verkkokauppasovellus lähettää verkkomaksuun liittyvät tiedot Stripeen rajapinnan kautta, jonka jälkeen Stripe yrittää näillä tiedoilla toteuttaa maksun heidän sovelluksillaan ja rajapinnoillaan halutun maksutavan mukaisesti. Tämän jälkeen Stripen rajapinta lähettää verkkokauppasovellukselle tiedon verkkomaksun onnistuneisuudesta. Jos verkkomaksu onnistui lähettää verkkokauppasovellus silloin viestin CommerceJS-rajapintaan, jossa CommerceJS rakentaa rajapinnan saamien tietojen mukaisesti tilauksen, jonka myyjä näkee.

Verkkomaksuun siirtymisen lomakkeeseen käytin kahta eri nappia, kuten kuvassa 6 näkyy sovelluksessa ja kuvassa 7 näkyy JavaScriptillä ja Reactilla.
Ensimmäinen nappi peruuttaa nykyisen maksutapahtuman ja menee takaisin toimitustietoihin. Toimitustiedoista voidaan mennä takaisin ostokoriin ja niin edelleen.
Toinen nappi siirtää nykyisen ostokorin tuotteiden hinnat valittuun gateway-palveluun, joka vie maksun tiedot rajapintaan, jossa suoritetaan verkkomaksu. Verkkomaksun suoritus tapahtuu halutulla tavalla eli pankkimaksuilla verkkokauppasovelluksessa. Rajapinta vie checkout-lomakkeeseen lisätyt pankkikorttitiedot gateway-palveluun, joka ohjaa turvallisesti korttitiedot maksurajapintaan. Maksurajapinta on aikaisemmin valittu ja lisätty maksurajapinta sovelluksessa.

Tässä sovelluksessa se on Stripe, mutta halutusti se voi olla jokin toinen, kuten PayPal tai Paytrail. Valittu verkkomaksupalvelu suorittaa verkkomaksun loppuun. Maksupalvelutarjoajan rajapintojen ja pankkien rajapintojen kautta.

Verkkomaksu viedään verkkomaksupalvelurajapintaan, josta verkkomaksutiedot siirretään verkkomaksurajapintaan, josta verkkomaksu toteutetaan. Silloin kun verkkomaksurajapinta on hyväksynyt ja suorittanut verkkomaksun antaa rajapinta verkkomaksupalvelurajapintaan viestin onnistuneesta tai epäonnistuneesta verkkomaksusta. Verkkomaksupalvelurajapinta antaa verkkokauppasovellukseen viestin, jonka kautta CommerceJS-rajapinta saa saman viestin verkkomaksupalvelurajapinnasta. Tämän jälkeen CommerceJS:n sovelluspohja tekee verkkokaupalle tilauksen tuotteista, jotka on hyväksytysti ostettu.

```
return (
  <>
    <Review checkoutToken={checkoutToken} />
    <Divider />
    <Typography variant="h6" gutterBottom style={{ margin: '20px 0' }}>Maksu Metodi</Typography>
    <Elements stripe={stripePromise}>
      <ElementsConsumer>{({ elements, stripe }) => (
        <form onSubmit={(e) => handleSubmit(e, elements, stripe)}>
          <CardElement />
          <br /> <br />
          <div style={{ display: 'flex', justifyContent: 'space-between' }}>
            <Button variant="outlined" onClick={backStep}>Back</Button>
            <Button type="submit" variant="contained" disabled={!stripe} color="primary">
              Pay {checkoutToken.live.subtotal.formatted_with_symbol}
            </Button>
          </div>
        </form>
      )}
      </ElementsConsumer>
    </Elements>
  </>
);
```

Kuva 12. Verkkokauppasovelluksen maksulomake.

# 6 ARVIOINTI

Verkkomaksamisen toteutus on yllättävän monipuolista ja vaikeaa, mutta sen toteutus onnistui hyvin. Opinnäytetyötä tehdessä opein verkkomaksuun liittyviä tärkeitä asioita.

Verkkomaksamista saadaan toteutettua useiden rajapintojen, tietokantojen ja sovellusten kautta. Verkkomaksamisen toteutuksessa on harkittava, mihin tarkoitukseen tarvitaan verkkomaksamista ja minkälaiset toiminnot sopivat parhaiten siihen.

Usein on tärkeää tietää, että minkälaista tietokantaa verkkomaksamisen toteutuksessa käytetään. Millaiseen tietokantaan tallennetaan verkkomaksamiseen liittyvät tiedot sekä minkälaisia rajapintoja/ja gateway-palvelimia käytetään.

Verkkokauppasovelluksen tekeminen onnistui hyödyntäen opinnäytetyöhön opettelemiani asioita. Verkkokauppasovelluksesta saatiin responsiivinen, eli sovellus toimii hyvin tietokoneella sekä mobiililaitteilla.

Verkkosivun käyttöliittymä toimii hyvin ja eri napit ohjaavat sovelluksen toimintaa halutusti suorittaen verkkokauppaan kuuluvia toimintoja, kuten verkkomaksamisen.

Tuotteiden lisäys ostokoriin toimii hyvin, mutta se voi ajoin olla vähän hidas. Tämä vaatisi käytön mukavuuden kannalta enemmän hiomista jatkossa. Kuitenkin sovelluksen kaikki toiminnot toimivat halutusti ja käyttöliittymäsovelluksessa on selkeä. Tämä helpottaa sovelluksen käyttöä.

# LÄHTEET


Anttila, A. (2021). Kartta.com – WooCommerce verkkokauppa. Haettu osoitteesta 1.2.2022
https://urn.fi/URN:NBN:fi:amk-202103314080

Avoinrajapinta.fi (2014) Avoimen rajapinnan määritelmä. Haettu osoitteesta 7.2.2022 http://avoinrajapinta.fi/

ECB. (2022). Statical Data Warehouse. Haettu osoitteesta 5.2.2022 https://sdw.ecb.europa.eu/browseSelection.do?type=series&q=PSS.A.U2.F000.I00.Z00Z.NT.X0.20.Z0Z.Z&node=SEARCHRESULTS&ec=&oc=&rc=&cv=&pb=&dc=&df=1.2.2022

Falk, C. (2014). Exploring the UI Universe: Different Types of UI. Haettu osoitteesta 21.2.2022 https://www.altia.com/2014/09/22/different-types-of-ui/

Fielding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures. Irvine: University of California. Haettu osoitteesta 18.2.2022 https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

freeCodeCamp.org. 2022. React Course – Beginner's Tutorial for React JavaScript Library. YouTube. Haettu osoitteesta 3.2.2022 https://www.youtube.com/watch?v=bMknfKXIFA8

GeeksforGeeks. (2021). Remote Procedure Call (RPC) in Operating System. Haettu osoitteesta 18.2.2022 https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/

Honkala, P. (2008). Verkkopalvelun maksutoiminnallisuuden toteutus osana J2RR-ympäristöä. Haettu osoitteesta 2.2.2022 https://urn.fi/URN:NBN:fi:amk-201003064806

IBM. (2015). The RPC interface. Haettu osoitteesta 7.2.2022 https://www.ibm.com/docs/en/zos/2.5.0?topic=environment-rpc-interface

JavaScript Mastery. 2020. Ecommerce Web Shop – Build & Deploy an Amazing App. YouTube. Haettu osoitteesta 3.2.2022 https://www.youtube.com/watch?v=377AQ0y6LPA

Järvinen, J. 2002. Hajautetut verkkopalvelut Toolkit.

Kankaanpää, S. 2016. REST-arkkitehtuurin käyttö web-rajapinnoissa. Haettu osoitteesta 7.2.2022 https://urn.fi/URN:NBN:fi:amk-2016060611967

Ketonen, J. 2019. Mobiilimaksaminen ja maksutavan valintaan vaikuttavat tekijät. Haettu osoitteesta 21.2.2022 https://urn.fi/URN:NBN:fi:amk-2019060615155

Lama Dev Group.2021. React Node.js E-Commerce App Full Tutorial. YouTube. Haettu osoitteesta 3.2.2022 https://www.youtube.com/watch?v=y66RgYMAgSo

Leppänen, V. (2012). RPC = Remote Procedure Call. Haettu osoitteesta 7.2.2022 http://staff.cs.utu.fi/opinnot/kurssit/HOJ-2012/kalvot/kalvot7.pdf

Loshin, P. (2021). Command-line interface (CLI). Haettu osoitteesta 21.2.2022 https://www.techtarget.com/searchwindowsserver/definition/command-line-interface-CLI

Mikael, J. (2013). Rajapinnan suunnittelu ja toteutus IOT-Laitteelle. Haettu osoitteesta 1.2.2022 https://urn.fi/URN:NBN:fi:amk-2020052012728

Mursu, S. (2016). REST-tietokantarajapinta mobiilisovellukselle ja web-sivustolle. Haettu osoitteesta 3.2.2022 https://urn.fi/URN:NBN:fi:amk-201605178209

Niskanen, J. (2020). Mobiilikäyttöliittymän toteuttaminen. Haettu osoitteesta 9.2.2022 https://urn.fi/URN:NBN:fi:amk-2020112323747

Ramesh, D. (2019). SOAP Web Services. Haettu osoitteesta 18.2.2022 https://medium.com/@dilshanramesh81/soap-web-services-c62940e0de93

Pulkkinen, J. (2011). Huoltokoneiden hallintajärjestelmän käyttöliittymäsuunnittelu. Haettu osoitteesta 9.2.2022 https://urn.fi/URN:NBN:fi:amk-201104063961

Rauhala, T. (2015). REST-pohjaisen web-rajapinnan kuvaaminen – Case Tax Treatment Service. Haettu osoitteesta 3.2.2022 https://urn.fi/URN:NBN:fi:amk-2015052510186

TutorialEdge.net. (2017). What is a RESTful API? Haettu osoitteesta 18.2.2022 https://tutorialedge.net/software-eng/what-is-a-rest-api/

Verkkokauppa.com. (2022). Verkkomaksu käyttöliittymä Kuva 1. Haettu osoitteesta 9.2.2022 https://www.verkkokauppa.com/fi/etusivu

Vettainen, J. 2014. Tradera-verkkopalvelu ja Arcticas SOAP-asiakassovellus. Haettu osoitteesta 8.2.2022 https://urn.fi/URN:NBN:fi:amk-2014052710186