YRKESHÖGSKOLAN
NOVIA

Kartläggningsverktyg för elbilsladdningsinfrastruktur i fastigheter

Andreas Stortåg

Examensarbete för ingenjörs (YH)-examen

Utbildningsprogrammet för el- och automationsteknik

Vasa 2022

EXAMENSARBETE

Författare:	Andreas Stortåg
Utbildning och ort:	El- och automationsteknik, Vasa
Inriktning:	Automationsteknik
Handledare:	Kaj Wikman

Titel: Kartläggningsverktyg för elbilsladdningsinfrastruktur i fastigheter

---

Datum: 3.5.2022 Sidantal: 62 Bilagor: 1

Abstrakt

Syftet med detta examensarbete var att digitalisera och automatisera processen att kartlägga en fastighet inför installation av elbilsladdare. Examensarbetet gjordes åt företaget Comsel System och behandlar hur man kartlägger en fastighet inför installation av elbilsladdare. Det tas även upp hur belastningen på elanslutningen beaktas, hur man kan sänka sina energikostnader för att ladda en elbil genom att använda ett spot-elavtal och hur skapandet av verktyg för att samla in relevant information gick till. Arbetet består av en teoretisk och en praktisk del.

Den nödvändiga informationen som behövdes för att kunna fastställa situationen vid fastigheten samlades först in direkt från kunden genom ett webbformulär, sedan samlades vidare information in med en smarttelefonapplikation. Förbrukningsmätdata togs även i beaktande för att kunna fastställa hur mycket ledig kapacitet som finns på den aktuella anslutningen. Webbsidan kodades i React för att få en dynamisk och multifunktionell webbsida som kunde visas i allt från smarttelefoner till stora tv-skärmar. Smarttelefonapplikationen kodades i React Native för att på samma gång få en applikation som fungerar för både Android och iOS operativsystemen. För att importera och analysera förbrukningshistoriken för anslutningen, och skapa grafer över datan användes VBA-programmering i Excel.

Slutresultatet för detta examensarbete blev ett webbformulär för kunder att fylla i grundläggande information före kartläggningen, en smarttelefonapplikation för kartläggaren att använda vid besök på plats vid fastigheten, samt en automatiserad Excel-fil som med ett knapptryck importerar data, analyserar den och skapar grafer. Med all insamlad information i digital form underlättas kartläggarens arbetsbörda avsevärt då den slutliga rapporten ska skapas.

Språk: svenska
Nyckelord: kartläggning, elbilsladdning, React

OPINNÄYTETYÖ

Tekijä:	Andreas Stortåg
Koulutus ja paikkakunta:	Sähkö- ja automaatiotekniikka, Vaasa
Suuntautumisvaihtoehto:	Automaatiotekniikka
Ohjaaja:	Kaj Wikman

Nimike: Kartoitustyökaluja sähköauton latausinfrastruktuurille kiinteistöissä

Päivämäärä: 3.5.2022 Sivumäärä: 62 Liitteet: 1

Tiivistelmä

Opinnäytetyön tarkoituksena oli digitalisoida ja automatisoida kiinteistön kartoitusprosessia ennen sähköautojen latausasemien asennusta. Työ on tehty Comsel Systemille ja käsittelee kiinteistön kartoittamista ennen sähköauton latureiden asentamista. Lisäksi käsitellään kuinka sähköliittymän kuormitus huomioidaan, miten sähköauton latauksen energiakustannuksia voidaan alentaa spot-sähkösopimuksella ja miten tarvittavien tietojen keräystyökalun luominen toteutui. Työ koostuu teoreettisesta ja käytännön osasta.

Tarvittavat tiedot kiinteistön lähtökohtaisen tilanteen selvittämiseen kerättiin ensin asiakkaalta verkkolomakkeen avulla, jonka jälkeen lisätietoa kerättiin älypuhelinsovelluksella. Kiinteistön kulutustietoja hyödynnettiin myös, mahdollistaen selvityksen siitä kuinka paljon vapaata kapasiteettia nykyisellä liittymällä on käytettävissä. Verkkosivu ohjelmoitiin Reactin avulla, mahdollistaen dynaamisen ja monikäyttöisen verkkosivuston, jota voidaan käyttää niin älypuhelimilla kuin suurilla televisioruuduilla. Älypuhelimen sovellus ohjelmoitiin React Nativea käyttäen, jotta sovellus on käytettävissä sekä Android- että iOS-käyttöjärjestelmillä. Kiinteistön liittymän kulutustietojen analysointiin, tiedon tuontiin sekä visualisointiin käytettiin VBA-ohjelmointia Excelissä.

Opinnäytetyön lopputuloksiin sisältyy verkkolomake, johon asiakkaat voivat täyttää perustietoja ennen kartoitusta, älypuhelinsovellus kiinteistön kartoituksen tekijälle sekä automaattinen Excel-tiedosto, joka tuo tietoja, analysoi niitä ja luo kaavioita yhdellä napin painalluksella. Digitaalisessa muodossa kerättyjen tietojen avulla kartoittajan työmäärä helpottuu huomattavasti kartoituksen loppuraporttia laadittaessa.

Kieli: ruotsi
Avainsanat: kartoitus, sähköautonlataus, React

BACHELOR'S THESIS

Author:	Andreas Stortåg
Degree Programme:	Electrical engineering, Vaasa
Specialisation:	Automation
Supervisor:	Kaj Wikman

Title: Tools For Mapping Properties Regarding Electrical Car Charging Infrastructure

---

Date: May 3,2022 Number of pages: 62 Appendices: 1

Abstract

The purpose of this bachelor's thesis was to digitalize and automate the process of mapping a property, before installing chargers for electrical vehicles. The thesis was made for the company Comsel System, and it describes the process of mapping a property before installing chargers for electrical vehicles. It also describes how the strain on the property's connection to the electrical grid is taken into consideration, how it is possible to lower the energy costs for charging an electrical vehicle by using a spot-price agreement and how the tools needed for gathering the information were created. The thesis consists of both a theoretical and a practical part.

The necessary information needed to assess the situation at the property was firs gathered through a webform directly from the customer. Then further information was gathered through a smart phone application. Consumption measurement data was also taken into consideration when trying to assess the available amount of power on the electrical connection. The webform was coded using React to get a dynamic and multifunctional website that was able to be displayed in everything from small screens such as smartphones all the way up to big screens like TVs. The smartphone application was coded using React Native, to get a version for both iOS and Android at the same time. VBA-programming in Excel was used to import and analyse consumption data for the electrical connection and create graphs showing the imported data.

The result of this thesis was a webform for the customers to fill out basic information before the survey, a smartphone application for the surveyor to use when visiting the property, and an automated Excel file that, with the press of a button imports data, analyses it and creates a graph. With all the gathered data in digital form, the surveyor's workload is reduced considerably, when it comes to creating the final report.

Language: swedish
Key words: survey, ev-charging, React

Innehållsförteckning

Bilageförteckning

1 Inledning

Detta examensarbete behandlar insamling och hantering av den information som är nödvändig vid kartläggning av en fastighet inför installation av laddare för elbilar. Belastning på elanslutningen och hur den kan regleras med hjälp av smartteknologi kommer även att tas upp, samt hur det med fördel går att använda ett timpris-elavtal för att sänka kostnaderna för laddning av elbilar.

1.1 Uppdragsgivare och beställare

Uppdragsgivaren är Comsel System som är ett företag som utvecklar smarta mätinsamlingssystem för el, fjärrvärme, vatten och gas. Företaget har tidigare främst fokuserat på mätinsamlingssystem där stora datamängder kostnadseffektivt kan hanteras med hög kvalitet och säkerhet. Denna typ av data är värdefull då den möjliggör energioptimeringar vilket i sin tur minskar påfrestningen på miljön, något som har blivit allt viktigare i modern tid. Vidare har Comsel System kunskap och teknologi inom elbilsladdning, vilket har utvecklats tillsammans med sin delägare Vasa Elektriska, för att stödja det växande området för elbilism.

1.2 Bakgrund

Comsel Systems VD Kristian Heimonen har konstaterat att intresset för elbilarna ökar explosionsartat och frågan de flesta nu ställer sig är går det att ladda en elbil hemma i ett egnahemshus eller i ett flerbostadshus? Vad krävs i så fall och vilka kostnader är förknippade med att kunna skapa förutsättningar för detta? Tyvärr finns det inget enkelt svar på dessa frågor utan mer regel än undantag är att det kräver ett besök på platsen och att varje objekt kartläggs enskilt.

1.3 Syfte

För att underlätta, snabba upp och höja kvaliteten av själva kartläggningen för olika typer av fastigheter krävs ett anpassat verktyg för ändamålet. Detta verktyg består av ett webbformulär för kunderna att fylla i, en smarttelefonapplikation som kartläggaren kan använda för att komplettera med ytterligare information vid besök på platsen, samt en Excel-

fil där förbrukningshistorik för fastigheten analyseras. Samtliga delar av verktyget har utvecklats som den praktiska delen av examensarbetet.

1.4 Avgränsning

Eftersom Comsel System erbjuder en mängd olika tjänster inom fjärravläsning kan det hända att kunden även är intresserad av någon av de andra tjänsterna på samma gång som de införskaffar elbilsladdare, exempelvis integrering av solpaneler i systemet. Då skulle det vara smidigt ifall verktyget även skulle samla in information angående denna tjänst på samma gång. Dessutom då varje fastighet är unik går det inte att ta fram någon enkel lösning som passar varje fall. Genom detta uppstår en närmast oändlig mängd scenarion som kan anträffas. Verktyget som ska utvecklas för att samla in den nödvändiga informationen behöver således avgränsas till kartläggning för elbilsladdning och den information som berör detta.

1.5 Framtiden

För tillfället görs dessa kartläggningar med penna och papper. Detta skapar dubbelt jobb för kartläggaren som först ska skriva upp allt för att sedan fylla i allt igen i Comsels datasystem. I framtiden kommer troligtvis fler verktyg utvecklas för att även kunna hantera insamling av information för andra tjänster. Exempelvis ifall kunden är intresserad av att integrera egen elproduktion i systemet.

Det kommer troligtvis att skapas skilda verktyg för kartläggning av andra tjänster då dessa har helt andra aspekter som ska tas i beaktan. För själva verktyget till kartläggning av fastigheter inför installation av elbilsladdare, kommer funktionaliteten i mobilapplikationen att utökas. Då det redan i nuläget finns förbättringar som kunde implementeras, men som av tidsskäl kommer att utebli i den första versionen av applikationen. Det samma gäller även för webbsidan, som av tidsskäl kommer att anses som färdig då all grundläggande funktionalitet är på plats. Även här kommer utvecklingen att fortlöpa efter examensarbetets slut.

2 Förkunskap

För att skapa alla de verktyg som kommer att behövas för insamling av informationen delas helheten upp i ett antal mindre delar. Då en kund kontaktar Vasa elektriska eller Comsel System och vill ha en kartläggning, får de börja med att fylla i ett webbformulär. Här får de fylla i grundläggande information om fastigheten och kontaktpersoner. Sedan ska informationen skickas in till Comsel Service Hub. Vad detta är beskrivs senare under rubriken Comsel Service Hub.

Sedan när själva kartläggningen ska utföras så skickas en kartläggare ut till fastigheten för att få en bättre uppfattning över situationen och samla in mer teknisk information. Kartläggaren kommer då att ha en smarttelefonapplikation som läser ut den information som kunden har angett via webbformuläret från Comsel Service Hub. Applikationen ska även ge kartläggaren möjlighet att anteckna en mängd olika detaljer och ta fotografier på exempelvis elcentraler och parkeringar. Den nya datan ska sedan kunna laddas upp till hubben direkt från applikationen med ett knapptryck.

2.1 React

Enligt rekommendationer från Comsel systems Chief Technology Officer Karl Herler, används React för att skapa webbformuläret. React är flera saker på samma gång, det är dels ett JavaScript baserat bibliotek som är utvecklat av Facebook, dels är det ett verktyg för att bygga dynamiska användargränssnitt.

React underlättar skapandet av interaktiva och skalbara komponenter. Det tar även hand om en del av det tunga lyftet i bakgrunden och underlättar för programmeraren att säga hur komponenter ska bete sig då användaren agerar med dem. React stöder även TypeScript. [1]

2.2 JavaScript

JavaScript är ett av världens mest använda programmeringsspråk. Efter 25 år i branschen verkar popularitet inte avta särskilt mycket. JavaScript är ett objektorienterat programmeringsspråk som används främst för att skapa interaktiva element i webbläsare och på webbsidor, men det har även spridit ut sig inom en mängd olika applikationer över åren. [2]

2.3 TypeScript

TypeScript är i korthet en moderniserad version av JavaScript, som stöder en närmare integration med editeringsprogram. Detta gör att eventuella fel i koden kan upptäckas tidigare och undvikas redan under editering före koden appliceras till webbsidan eller applikationen. TypeScript kod omvandlas till JavaScript då den kompileras vilket ger samma breda stöd som JavaScript redan har. [3]

2.4 MUI

För att underlätta skapandet av användargränssnittet kommer ett React användargränssnitt-bibliotek som heter MUI att användas. Biblioteket innehåller en mängd olika lättanvända komponenter såsom knappar, textfält och listor. Dessa komponenter har ett modernt och bekant utseende som smidigt kan anpassas till applikationens temafärger. Biblioteket används av en mängd olika moderna webbsidor. [4]

2.5 React Native

React Native är en variant av React som används för skapandet av mobila applikationer. Med React Native går det att koda applikationen för både Android och iOS på samma gång. Om applikationen skulle kodas direkt för ett av operativsystemen skulle det inte gå att installera på det andra och vice versa. Men med React native kompileras koden för att kunna köras på båda operativsystemen utan att behöva skriva två olika varianter av den. React Native har sina egna varianter av komponenter som det använder för att kunna anpassas till de båda operativsystemen. Exempelvis i React native heter komponenten som visar en bild Image medan den heter ImageView i Android och UIImageView i iOS. Fördelen med att använda React Native blir genast självklar genom att slippa lära sig två olika sätt att göra samma sak. [5]

2.6 VBA

Visual Basic, som förkortas VBA är Microsofts programmeringsspråk som kan användas i alla Microsoft Office program. Detta är ett sätt att kombinera traditionell programmering med olika funktioner som de olika programmen kan utföra. I Excel går det att exempelvis kontrollera värdet på alla celler inom ett vist område, och göra något ifall ett bestämt värde

upptäcks eller liknande. Det går även att skapa användargränssnitt med knappar som utför olika funktioner. [6]

2.7 Smarttelefonapplikation

För detta projekt kommer fokusen att ligga på att skapa applikationen för Android. Detta eftersom det är mycket sällsynt att ett installationsföretag ger iOS telefoner eller surfplattor som arbetsredskap åt sina anställda ute på fältet. Däremot ifall efterfrågan skulle bli större än förväntat, så borde det inte vara några större problem att kompilera en iOS variant av applikationen då den är gjord med React Native.

2.8 Comsel Service Hub

För att hålla reda på all information och vad som hör ihop med vilken fastighet så använder Comsel System sin egen server, Comsel Service Hub. Här sparas alla fastigheter som har någon av deras produkter och all relevant information kring detta. Då användaren först fyller i webbformuläret så kommer en ny enhet som det internt kallas, att skapas på hubben. Sedan sparas all information som användaren fyller i direkt till enheten på hubben. Sedan i smarttelefonapplikationen kan informationen hämtas ut, läsas av, modifieras och skrivas tillbaka till hubben.

2.9 Oculus

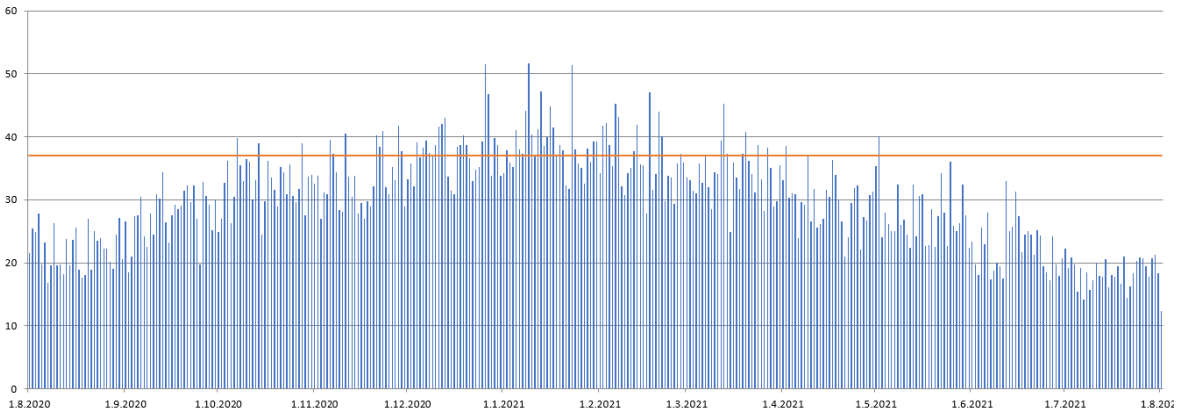
Comsels Oculus är en modul som monteras i närheten av el-, vatten-, fjärrvärme- eller gasmätare. Den kommunicerar trådlöst med mätaren genom standardiserad infraröd kommunikation och sänder mätdata tillbaka till Comsels system. Informationen kan sedan användas för att optimera energianvändningen eller styra användning enligt elpriserna. Den kan även paras ihop med annan utrustning såsom elbilsladdaren. [7]

2.10 Förbrukningsdata

En viktig del av planeringen innefattar att ta reda på fastighetens förbrukningshistorik och bestämma hur mycket ledig effekt som finns tillgänglig på anslutningen. Eller ifall anslutningen inte har tillräckligt med effekt och är i behov av att förstoras. Detta görs genom att processera förbrukningsdata för anslutningen som fås av Vasa elektriska, med kundens samtycke. Den mätdata som fås är i råformat och behöver processeras för att lättare kunna

få en uppfattning om hur förbrukningshistoriken har sett ut. Grafer skapas även för att visualisera datan. Allt detta innebär ytterligare manuellt arbete för kartläggaren. Däremot går i princip allt detta att automatisera med VBA-programmering i Excel.

I figuren syns en färdigställd graf över en fastighets förbrukning under ett år bakåt i tiden. På X-axeln är tidpunkten för mätningen och på Y-axeln är den avmätta skenbara effekten. En ny avläsning görs varje timme under hela året och sammanlagt görs cirka 8760 avläsningar.



Figur 1: Exempel på förbrukningsdata för en fastighet under ett år.

I figuren syns även en orange linje. Denna linje ligger vid ett beräknat värde på 37 kVA där de 100 största topparna har klippts bort. Detta innebär att förbrukningen för fastigheten ligger under linjen 99,99 % av tiden. Samtidigt kan vi se att de toppar som sticker upp över linjen är kraftigt över, med den största på 51,7 kVA. Andreas Willför som i nuläget utför samtliga kartläggningar vid Comsel System har valt att kalla denna linje för praktisk maxförbrukning, och anser att den resterande effekten som finns ledig på anslutningen skulle vara möjlig att reservera åt elbilsladdningen. Sedan får ett nytt beslut tas om den kvarvarande effekten kommer att räcka till, eller om anslutningen behöver förstoras för att kunna få tillräckligt med effekt åt laddarna.

I fall anslutningens resterande effekt är tillräcklig räknas vidare på hur många laddare som husbolaget önskar sig och hur mycket effekt det skulle kunna innebära per laddare. De stora strömpikarna som sköt över den beräknade maxförbrukningslinjen är dock inte bortglömda. Laddaren har en smart belastningsstyrning och kan enligt behov strypa effekten till laddarna, beroende på hur belastningen ser ut på anslutningen för tillfället. Denna funktionalitet beskrivs i mer detalj under rubriken Smart belastningsstyrning.

3 Belastning på elanslutning

Genom att fler och fler personer skaffar elbilar ökar även belastningen på elnätet. Detta är dock inte ett bekymmer för den enskilde husägaren eller husbolaget att lösa utan kanske närmare ett problem för elleverantören och de som äger elnäten. Det som däremot blir den enskilde husägarens ansvar eller snarare kartläggarens, är att se till att den aktuella fastighetens anslutning klarar av belastningen från att ladda elbilar. Vid ett egnahemshus med endast en elbil som ska ladda är det sällan något större bekymmer. Men då ett radhus eller höghus vill sätta upp ett antal stationer på gården börjar det bli mera av en utmaning för anslutningen.

Med en Mode 3-laddare används ändå bilens interna laddare, dessa kan variera från 3,6 kW till 22 kW, och i framtiden troligtvis ännu högre. Detta gör att de inte är helt självklart att beräkna exakt hur mycket effekt som kommer att behövas för att ge alla elbilsladdplatser tillräckligt med effekt. Det är exempelvis stor skillnad på om alla bilar som ska ladda vill uppnå 22 kW eller om hälften av bilarna inte klarar av mer än 3,6 kW. På grund av detta används någon form av medeltal för vad en bil kommer att förbruka, för att kunna avgöra om anslutningen kommer att klara av belastningen eller ej.

3.1 Matarkabel och säkringar

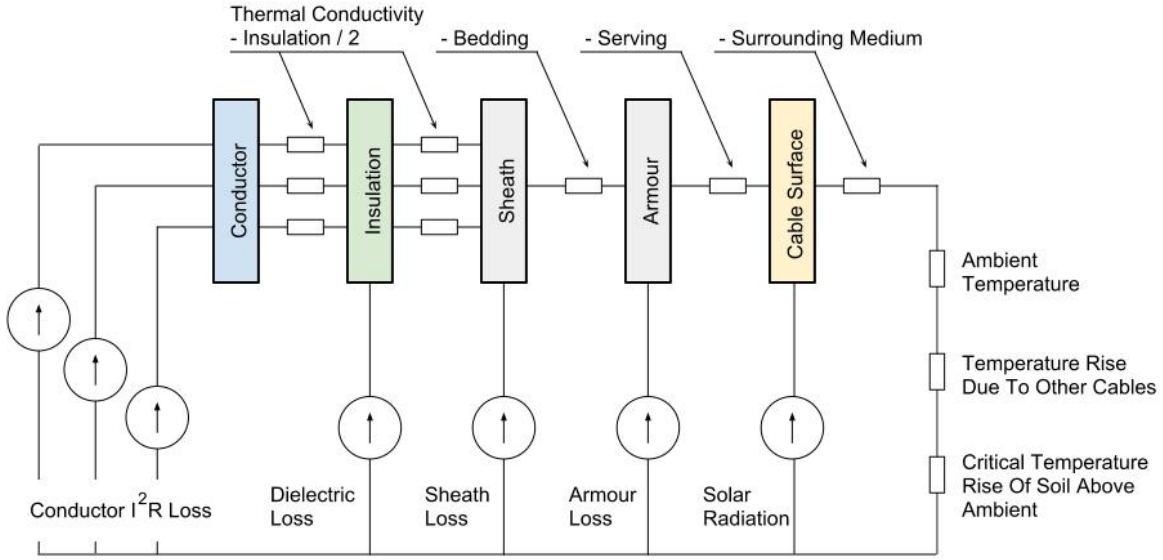
För att kunna hantera tillräckligt med ström till både fastighetens befintliga krav och det nya behovet för en eller flera elbilsladdare, är det viktigt att se över fastighetens anslutande kabel. Den behöver vara rätt dimensionerad för att klara av den nya belastningen som uppstår.

Orsaken till att en kabel har en maximal strömmängd som den klarar av att överföra är värme. Förluster inom kabeln skapar värme, takten som denna värme överförs till omgivningen är beroende på miljön som kabeln befinner sig i. Vid en viss temperatur kommer den mängden värme som skapas att vara i jämvikt med den värme som absorberas av omgivningen, då har kabeln uppnått termisk jämvikt. Förlusterna och den värme som skapas är direkt relativ till mängden ström som flödar genom kabeln. Då strömmängden ökar ser vi samtidigt att förlusterna ökar, och samtidigt ökar temperaturen för att uppnå termisk jämvikt. Vi kommer vid en viss strömmängd att uppnå en termisk jämviktstemperatur som motsvarar den maximala tillåtna temperaturen för kabelns isolering. Detta är den teoretiskt största möjliga strömmen för denna kabel. Det finns dock fler faktorer som spelar roll och orsakar förluster, även dessa bör tas i beaktande ifall ett extra noggrant värde önskas vid beräkning av kabelns maximala strömmängd.

För att noggrannare beräkna en kabels kontinuerliga strömöverföringskapacitet måste en mängd olika faktorer tas i beaktande. Dessa är bland annat:

- Installationssätt: luftburen, nergrävd, exponering för direkt solljus
- Växelström eller likström
- Ledarens material
- Kabelns isolering
- Störningsskydd
- Armering
- Kabelns hölje eller mantel

Samtliga faktorer som inverkar på kabelns maximala strömöverföringskapacitet illustreras i Figur 2. Figuren är mycket detaljerad och beskriver väldigt ingående alla faktorer som inverkar på kabelns förluster. [8]



Figur 2: Termomodell för en kabel. [8]

Enligt IEC 60287 Beräkning av kontinuerlig strömkapacitet för kablar under 100 % belastning finns olika formler för dessa olika tillämpningar. Standarden gäller för alla växelströmmar och för likström upp till 5kV.

För nedgrävda kablar där jorden inte torkar eller kablar som är i luften:

AC: $$I = \left[\frac{\Delta\theta - W_d[0{,}5T_1 + n(T_2+T_3+T_4)]}{RT_1 + nR(1+\lambda_1)T_2 + nR(1+\lambda_1+\lambda_2)(T_3+T_4)}\right]^{0,5} \quad (1)$$

DC: $$I = \left[\frac{\Delta\theta}{R'T_1 + nR'T_2 + nR'(T_3+T_4)}\right]^{0,5} \quad (2)$$

För nedgrävda kablar där jorden delvis torkar upp:

AC: $$I = \left[\frac{\Delta\theta - W_d[0{,}5T_1 + n(T_2+T_3+T_4)] + (v-1)\Delta\theta_x}{R[T_1 + n(1+\lambda_1)T_2 + n(1+\lambda_1+\lambda_2)(T_3+vT_4)]}\right]^{0,5} \quad (3)$$

DC: $$I = \left[\frac{\Delta\theta + (v-1)\Delta\theta_x}{R'[T_1 + nT_2 + n(T_3+vT_4)]}\right]^{0,5} \quad (4)$$

För nedgrävda kablar där jorden är delvis torr:

AC: $$I = \left[\frac{\Delta\theta_x - nW_dT_4}{nRT_4(1+\lambda_1+\lambda_2)}\right]^{0,5} \quad (5)$$

DC: $$I = \left[\frac{\Delta\theta_x}{nR'T_4}\right]^{0,5} \quad (6)$$

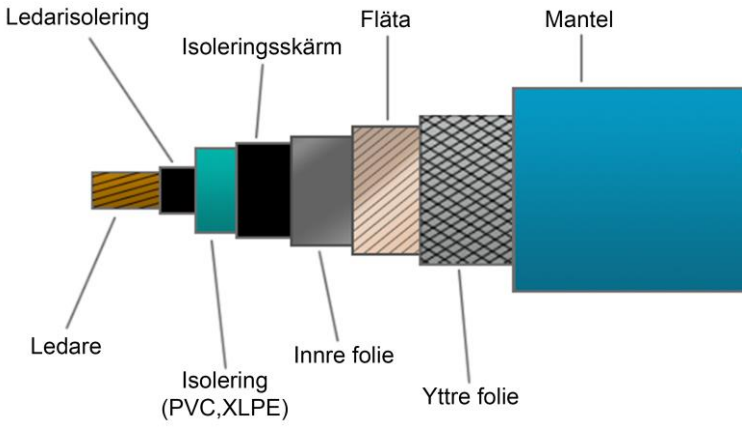
Kablar utsatta för direkt solljus

AC: $$I = \left[\frac{\Delta\theta - W_d[0{,}5T_1 + n(T_2 + T_3 + T_4{}^*)] - \sigma D_e{}^* H T_4{}^*}{RT_1 + nR(1+\lambda_1)T_2 + nR(1+\lambda_1+\lambda_2)(T_3 + T_4{}^*)}\right]^{0,5} \quad (7)$$

DC: $$I = \left[\frac{\Delta\theta - \alpha D_e{}^* H T_4{}^*}{R'T_1 + nR'T_2 + nR'(T_3 + T_4{}^*)}\right]^{0,5} \quad (8)$$

Där:
n = Antal strömbärande ledare
v = Värmeresistansförhållande mellan torr och fuktig jord
I = Strömmen i ledare, A
R = AC Ledarens resistans givet i Ω/m
R’ = DC Ledarens resistans givet i Ω/m
T_1 = Värmekonduktivitet per kärna mellan ledare och inre folie, W/m·°C
T_2 = Värmekonduktivitet mellan inre folie och yttre folie, W/m·°C
T_3 = Värmekonduktivitet för yttre mantel, W/m·°C
T_4 = Värmekonduktivitet för omgivande element, W/m·°C
T_4* = Yttre värmekonduktivitet för luft, justerat för solstrålning, W/m·°C
D_e* = Kabeldiameter inklusive isolering, m
H = Intensitet för solstrålning, W/m^2
W_d = Dielektrisk förlust
λ_1 = Förhållande över förluster i inre folie till samtliga förluster i alla ledare
λ_2 = Förhållande över förluster i yttre folie till samtliga förluster i alla ledare
σ = Absorberingskoefficient för solstrålning över kabelns yta
θ = Maximala arbetstemperaturen för ledaren, °C
θ_a = Omgivningstemperatur, °C
Δθ = Temperaturskillnad (θ- θ_a), K
$\Delta\theta_x$ = Kritisk temperatur för jord, °C

[9]



Figur 3: Kabelns konstruktion. [10]

Notera att kabeln som illustreras i Figur 3 ofta har flera ledare inom samma isolering.

Om ett riktigt noggrant värde önskas går det att använda dessa formler för beräkning av den teoretiskt maximala kapaciteten för en kabel. Men då en ny installation ska göras eller kontrolleras om en befintlig kabel är tillräcklig, behövs nödvändigtvis inte alla dessa detaljer. Det går i stället att använda en officiell tabell, de värden som är angivna har en viss säkerhetsmarginal för att ta olika yttre faktorer i beaktande. Säkringarna går även att ta ur samma tabell.

Tabell 1: Kabeldimensionering vid omgivningstemperatur på 30 °C [11]

Tvärsnittsarea (mm^2)	Nominellt strömvärde koppar (A)	Nominellt strömvärde aluminium (A)	Största säkringar koppar (A)	Största säkringar aluminium (A)
1,5	17	--	13	--
2,5	23	19	20	16
4	31	25	25	20
6	40	32	35	25
10	54	44	40	40
16	73	58	63	50
25	95	76	80	63
35	117	94	100	80
50	141	113	125	100
70	179	142	160	125
95	216	171	160	125
120	249	197	200	160
150	285	226	250	200
185	324	256	250	200
240	380	300	315	250
300	435	344	315	250

3.2 Strömpikar

För en fastighet går det inte att på förhand säga exakt hur mycket ström som kommer att gå åt. Detta gör det svårt att uppskatta exakt hur mycket ledig effekt som behöver finnas på anslutningen, och hur mycket av den som kan utlovas åt elbilsladdningen. Ytterligare kan situationen se olika ut beroende på hur många boende det finns i fastigheten.

En befintlig anslutning har en viss uppsättning huvudsäkringar som tål en viss belastning före de slår ur. Eftersom Comsels kunder i de flesta fall även är kunder hos Vasa Elektriska har vi möjlighet, med kundens samtycke att kontrollera hur förbrukningen har sett ut under det senaste året. Detta gör att vi kan beräkna en teoretisk maxförbrukning med undantag för de 100 största pikarna. Detta för att vi inte i onödan ska vara tvungna att rekommendera kunden att förstora sin anslutning då det troligtvis inte kommer att vara nödvändigt bara för att de har haft ett par gånger under året då förbrukningen tillfälligt varit större än vanligt. Ifall en sådan situation skulle uppstå kan vi enkelt begränsa effekten till laddarna kortvarigt tills belastningen från resten av fastigheten åter sjunker.

3.3 Schuko-uttag

Som fastställts under rubriken Elbilsladdare är laddning ur schuko-uttag tillåtet till en viss del enligt EU direktiv. För Mode 1- och Mode 2-laddning är det möjligt att använda vanliga enfas schuko-uttag beroende på den effekt som förbrukas. Det bör dock tas i beaktande att dessa inte är dimensionerade för att klara av en kontinuerlig förbrukning som ligger nära sin maxkapacitet.

Det är möjligt att byta ut ett vanligt schuko-uttag till ett super-schuko som är tillgängligt från en mängd olika tillverkare. Dessa är dimensionerade för att klara av 16 A kontinuerlig förbrukning. Det bör dock tas i beaktande att 16 A endast räcker till 6,6 kW laddning. De flesta nya bilar i dagens läge är kapabla till 11 eller 22 kW och då räcker inte enfas till längre. Visserligen kan vissa bilmodeller även begränsa laddeffekten och ladda på enfas, fast bilens interna laddare är kapabel till mer effekt än enfas kan leverera. Det går även med fördel att dela upp en belastning som teoretiskt sett skulle klaras av på enfas över alla tre faser för att lätta på belastningen.

Dessa orsaker är troligtvis vad som bidragit till rekommendationerna om att i första hand använda Mode 3-laddare i stället för schuko. Men under övergångsperioden då kunden väntar på att få en laddare installerad är super-schuko det bästa alternativet.

4 Elbilsladdare

Enligt IEC 62196 standarden som behandlar kontakter, uttag, fordonskontakter och konduktiv laddning av elektriska fordon, så finns det fyra olika nivåer som beskriver generell karakteristik för olika laddningstyper, elektriska fordon och laddare. Dessa är:

- **Mode 1** är baserad på växelström och fokuserar mest på lätta fordon med lägre strömmar, så som cyklar och mopeder. Detta läge används inte för laddning av elbilar och är tillochmed förbjuden från att användas till det i ett antal länder. Vid denna typ laddas batteriet ur ett standard schuko-uttag

- **Mode 2** är baserad på växelström och får användas vid laddning av elbilar, men är endast menad att användas tillfälligt. Eller i övergångsperioden innan en bättre lösning har implementerats. Vid denna typ laddas bilen ur ett standard schuko- eller super-schuko-uttag.

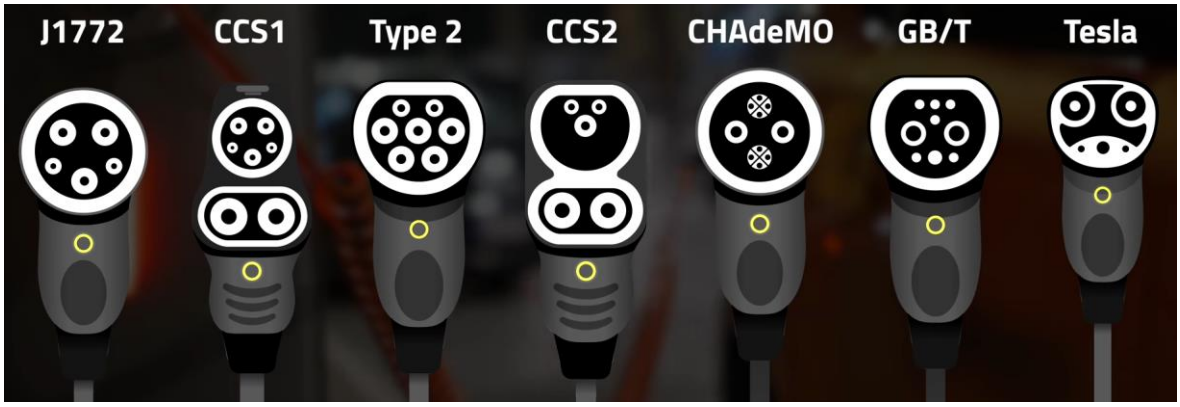
- **Mode 3** är baserad på växelström och är den rekommenderade metoden att använda vid laddning av elbilar på lång sikt. Den vanligaste kontakten för denna metod är Type 2 som blev standard enligt EU-direktiv 2014/94/EU 22 i oktober 2014. [12]

- **Mode 4** är baserad på likström och är vad som kallas snabbladdning. Denna metod använder en extern laddare med mycket högre kapacitet än vad bilens egna laddare kan leverera. I Europa var det till en början CHAdeMo-kontakten som var den vanligaste, men den har under senare år ersatts med CCS2. [13] [14]

4.1 Olika kontakter

Internationellt sett finns det väldigt många olika typer av elbilsladdkontakter. Däremot är det huvudsakligen endast fem olika typer av kontakter som har använts i Finland:

- **Type 1** användes vid de första elbilarna som dök upp på våra gator runt 2011 då Nissan lanserade Leaf i Europa. Denna kontakt använder enfas växelström.
- **Type 2** blev den europeiska standarden 2014 kan använda en- eller trefasväxelström beroende på tillämpningen. Den är i dagsläget den överlägset vanligaste kontakten.
- **CHAdeMO** var populär i Japan under en längre tid och som även den kom till Europa genom att Nissan Leaf lanserades på den europeiska marknaden. Denna kontakt används för snabbladdning och använder därför likström.
- **CCS2** är den nya europeiska standarden för snabbladdning som ersätter CHAdeMO. Även denna använder likström.
- **Tesla** kontakten används på vissa Tesla modeller medan andra av deras modeller har CCS2-kontakten. [15]



Figur 4: Olika typer av laddningskontakter för elbilar. [16]

Notera att kontakten som kallas J1772 i Figur 4 är den som kallas Typ 1 enligt IEC 62196. [13]

4.2 Comsel-laddare

Comsels laddare kommer att vara tillgänglig i ett flertal olika utföranden så den kan skräddarsys till användarens behov. Då en kund beställer en kartläggning av Comsel System kommer en kartläggare att besöka fastigheten och göra en bedömning av situationen. Sedan enligt kundens önskemål kan laddaren utrustas och konfigureras så att den passar situationen.

Laddaren kan fås som en- eller trefasmodell. Det är även möjligt att välja mellan en fast kabel med typ 2-kontakt som är 5 m, eller ett typ 2-uttag på laddaren så kunden kan använda en egen kabel. Detta kan vara bra ifall bilen inte har en typ 2-kontakt så är det möjligt att använda en annan kabel som passar, exempelvis om bilen har typ 1-kontakt. Det är även möjligt att få schuko-uttag på laddaren, antingen ett eller två. Sedan kan kunden även välja att montera laddaren mot en vägg eller montera på en traditionell motorvärmarstolpe.

Laddaren kan kommunicera över LTE, 4G, 5G, Wi-Fi 802.11 b/g/n – 2,4 GHz, Trådlös M-bus och Ethernet. I första hand används Ethernet eller Wi-Fi för att skicka data till Comsels servrar och i sin tur till användarens smarttelefon. Ifall dessa båda inte är tillgängliga kan laddaren även använda LTE/4G/5G för denna kommunikationsöverföring. I praktiken borde det inte vara någon större skillnad på hastigheten eftersom den datamängd som skickas inte är särskilt stor. M-bus används för att kommunicera med den Oculus som är kopplad till huvudmätaren för att hålla koll på förbrukningen på anslutningen.

Laddaren är utrustad med en mängd olika säkerhetsfunktioner. En av dessa är en jordfelsbrytare som slår ur vid 30 mA AC eller 6 mA DC. En annan är temperaturövervakning på flera ställen på kretskortet och i olika kontakter. Denna övervakning stänger av strömmen ifall en förhöjd temperatur detekteras. Den är även utrustad med lås för kabeln ifall laddaren är av en sådan modell som använder en typ 2-kontakt i stället för en fast kabel. Detta för att ingen utomstående ska kunna gå förbi och dra ur kabeln ur laddaren. Bilen brukar i de flesta fall vara utrustad med ett liknande lås i andra ändan av kabeln. Låset har en backup som gör att det släpper ifall ett strömavbrott skulle uppstå. Utan detta skulle kabeln vara låst i laddaren tills strömmen kommer tillbaka.

För att identifiera sig har laddaren stöd för RFID-kort och taggar. Utöver dessa kan användaren förstås använda mobilapplikationen. Det är även planerat att användaren ska kunna identifiera sig med sin smartklocka via NFC inom en snar framtid.

Laddaren uppfyller kraven för:

- IEC 61851–1:2019 Generella krav för konduktiva laddningssystem för elektriska fordon.
- IEC 62196 Kontakter, uttag och fordonskontakter för konduktiva laddningssystem för elektriska fordon.
- EU direktiv EU/2014/35 Harmonisering av lagar gällande medlemsstaters förhållande till tillgänglighet av laddningsinfrastruktur för användning inom vissa spänningsintervall.
- Den är dessutom förberedd för IEC 15118 Fordon till elnät kommunikationsgränssnitt.

Figur 6: Väggmonterad laddare med typ 2-uttag.

Figur 5: Stolpmonterad laddare med fast kabel och schuko-uttag.

4.3 Smarttelefonapplikation för elbilsladdare

Till Comsels elbilsladdare finns det även en tillbehörande applikation. Denna är publik och kommer att vara tillgänglig för alla kunder som skaffar en elbilsladdare. Applikationen är tillgänglig för både Android och iOS. Utvecklingen av denna applikation ingår inte i detta examensarbete, utan jobbet görs av Comsels mjukvaruavdelning. Den färdiga produkten är en blandning av Vasa elektriskas och Comsels överenskommelser över funktionalitet och tillgänglighet.

Då en ny användare registrerar sig och startar applikationen väljs först vilken bil användaren har. Detta är förstås viktigt då alla bilar har olika storlekar på batterier, förbrukning och kapacitet på sin inbyggda laddare.

Under en flik syns laddningshistorik och sessioner. Varje laddning loggas i historiken och användaren kan se detaljerna så som när laddningen har börjat och slutat, hur mycket ström som har laddats och hur många procent som har laddats till batteriet. Applikationen visar även månatlig statistik över hur mycket månadens laddning har kostat.

Det finns tre olika laddningslägen, det första är autostart, i detta läge tas inga externa faktorer i beaktande utan laddningen startar direkt och slutar då den angivna gränsen uppnåtts.

Det andra läget är Eco, här försöker laddaren dels ladda medan elektriciteten är som billigast, dels när det finns en överproduktion i elnätet. Då användaren ska ladda går det via en slider fritt att välja vid vilken procent laddningen önskas stängas av och vilken laddningsnivå som batteriet för tillfället har. Den beaktar även att bilen faktiskt behöver laddas innan morgonen och börjar ladda även fast priserna eller överproduktionen inte skulle hända.

Det tredje och sista läget är schemaläge. Här får användaren specificera avgångstid och hur mycket räckvidd som kommer att behövas. Detta görs på veckobasis, alltså alla veckodagar kan ha olika avgångstider och kilometerbehov. Laddningen sker då automatiskt på samma sätt som i Eco läget. I detta läge går det även att ställa in en avfärd flera dagar framåt och låta laddaren ladda lite nu och då beroende på elpriserna och belastningen på elnätet.

I användargränssnittet finns även knappar för att tvinga i gång eller tvinga avslutning av laddning. Dessa knappar ignorerar alla utomstående faktorer förutom de säkringar som är installerade. Det vill säga ifall anslutningen ligger nära sin maxkapacitet så kan inte laddningen startas för att undvika att bränna propparna. Det är specificerat i IEC 61851 att Mode 3-laddning inte får utföras med mindre än 6 A.

Oberoende vilket läge som användaren väljer så finns möjligheten att begränsa laddningseffekten via en slider. Det maximala är 22 kW och det lägsta är beroende på om bilen laddar med en- eller trefas. Eftersom 6 A är det minsta som får användas är det för enfas 1,38 kW enligt formeln:

$$P = U \cdot I = 230V \cdot 6A = 1380\text{W} \quad (9)$$

För trefas gäller samma sak, den minsta effekten är 4,14 kW enligt formeln:

$$P = (U \cdot I) \cdot antalFaser = (230V \cdot 6A) \cdot 3 = 4140W \quad (10)$$

På huvudsidan i applikationen syns en layout som är lite olika beroende på hur systemet ser ut. Ifall användaren bor i ett egnahemshus och har tillgång till huvudmätaren och producerar egen elektricitet från solpaneler fås den fulla vyn. Det som syns i vyn är hur mycket ström som går genom huvudmätaren till fastigheten, hur mycket ström som används till laddaren, hur mycket ström som används av resterande fastigheten och hur mycket ström som produceras från solpanelerna. Ifall inte någon egen produktion äger rum faller denna bit helt enkelt bort från vad som användaren ser. Ytterligare ifall användaren exempelvis bor i ett höghus kanske inte tillgång till anslutningens förbrukning finns och då fås ytterligare en avskalad vy som endast visar hur mycket ström som går till laddaren.

4.4 Smart belastningsstyrning

I nuläget kommer dessa elbilsladdare för det mesta att installeras på befintliga fastigheter, som inte nödvändigtvis har sin anslutning dimensionerad för att ha ledig effekt över till ett flertal elbilsladdare. Men med Comsels smarta laddare går det även att installera en Oculus för att mäta belastningen på anslutningen. Detta är en liten kommunikationsmodul som kopplas till fastighetens huvudmätare och kommunicerar trådlöst med den via ett standardiserat protokoll. Denna kan sedan kopplas samman med laddaren för att skicka information om hur mycket belastning som ligger på anslutningen. Denna information visas sedan i laddarapplikationens användargränssnitt. På detta vis går det exempelvis att automatiskt märka av ifall många boende i huset startar bastun samtidigt. Då systemet märker att belastningen stiger kraftigt går det samtidigt att dra ner effekten till laddaren så länge som belastningen är extra hög. Dessa gränser ställs in direkt i laddaren i installationsskedet.

Denna begränsning är även beroende på ifall det är en enfas eller trefasmodell av laddaren. Ifall det är en trefasmodell och övervakningen märker av att en av faserna är oproportionellt mer belastad än de andra blir laddaren tvungen att dra ner effekten på samtliga faser. Men ifall det däremot är frågan om en enfas laddare är det självklart endast den fasens belastning som spelar roll.

I några fall kan anslutningen ha mycket ledig effekt och behöver då möjligtvis inte göra detta utan kan låta alla laddare arbeta på full effekt hela tiden. Men i de flesta fall kan det ändå vara en bra idé att utnyttja denna funktionalitet då den är lättillgänglig och färdigt integrerad i laddaren. Det enda som behövs utöver laddaren är en Oculus. Den kommunicerar trådlöst

både med mätaren och laddaren så det innebär inte några extra kablar. Utan denna funktionalitet kunde en strömpik orsaka att någon säkring slår ur och förorsaka nya problem.

4.5 Börs-el

I dagens läge har det blivit populärare med så kallade börs-el eller Tim-el/spot-el som det även ibland kallas. Konceptet här skiljer sig från det traditionella sättet där konsumenten betalar ett fast pris för den mängd ström som förbrukats under ett dygn. Med börs-el betalar användaren olika priser för varje timme på dygnet beroende på vad priset på el börsen är för tillfället. Priset baserar sig på Nord Pool som är en organisation inom Europa. Nord Pool är en handelsplats där elproducenter och elåterförsäljare möts och gör upp om priser, beroende på tillgång och efterfrågan. [17]

Orsaken till att detta är relevant för laddning av elbilar är förstås att det går att få ett billigare pris bilen laddas vid rätt tidpunkt. Elpriserna per timme är så gott som alltid billigast mellan klockan tolv på natten och sju på morgonen. Exempelvis den 18 februari 2022 var priserna 1,88 c/kWh under natten och 12–15 c/kWh under resten av dagen. På lång sikt adderar dessa cent ihop till stora summor. Det är därför en självklar fördel med Comsel Systems laddare som kan styra laddningen enligt marknadspriserna, jämfört med en dum laddare som startar då den pluggas i.

Denna funktionalitet förutsätter förstås att ett elavtal med börs-elpriser används. Inställningen för om användaren använder ett sådant elavtal eller ej kommer då användaren parar ihop sin laddningsapplikation med laddaren. Även fast ett sådant elavtal används går det att använda autostartläget ifall användaren vill strunta i elpriserna och börja ladda direkt.

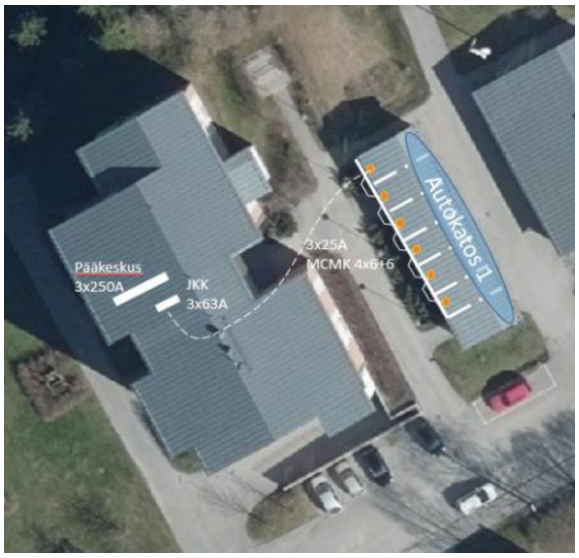
4.6 Laddbara kilometer per timme

En siffra som kunden kan vara mycket intresserad av är antalet kilometer som laddas till bilen per timme. Med hjälp av denna går det att räkna ut om det finns tillräckligt med ström för att utföra morgondagens utfärd eller ej. Även här finns det många variabler som skiljer sig åt under beräkningarna. Först och främst bilens inbyggda laddare, den kan endast ta emot en viss effekt, oberoende på om laddaren på väggen kan ge ut mer eller inte. Sedan har alla bilar olika förbrukning givet i kWh/100 km. Även denna faktor inverkar då det ska beräknas hur många kilometer som fås laddat under en timme. Sedan har vi anslutningens eventuella begränsning. Fast alla boende skulle ha en bil som kan ta emot 20 kW är det inte sagt att det

är möjligt att ta ut så mycket effekt ur anslutningen ifall alla vill ladda på samma gång. Även här kommer den smarta effektregleringen in i bilden. För att sedan i den slutliga rapporten kunna ge en uppskattning åt kunden så görs vissa antaganden vid beräkningarna.

Beräkningarna utgår ifrån att bilen använder cirka 20 kWh/100 km och att den kan ladda med 11 kW. Med dessa variabler och med det antalet laddare som kunden önskar installera kan det beräknas på ett ungefär hur många kilometer varje laddare kan ge ut per timme.

Figur 7 illustrerar hur ett förslag kan se ut. Det syns att huvudcentralen klarar av att ge ut 3x250 A. Efter huvudcentralen finns en delningscentral som klarar av 3x63 A. Från denna central kunde en ny kabel dragas som behöver klara av 3x25 A. Under biltaket kan sex elbilsladdare installeras.



Figur 7: Installationsförslag.

För att sedan göra beräkningar på antalet laddplatser, bilarnas förbrukning och antalet timmar att ladda, skapas en tabell enligt Figur 8. Det beaktas hur mycket ledig effekt de olika matningarna har och delar upp den på antalet platser. Genom att ändra på faktorerna går det att försöka hitta ett alternativ som passar för fastighetens behov.

Dessa beräkningar utgår från det tyngsta scenariot där alla boende har bilar som kan ta emot 22 kW och alla behöver ladda samtidigt. I verkligheten kommer detta troligtvis inte vara sannolikt då bilen inte behöver laddas alla dagar och vissa bilar inte kan hantera mer än 6 kW. Men i framtiden kommer troligtvis antalet elbilar att öka så detta scenario blir mer och mer sannolikt. I den sista kolumnen syns resultatet i kilometer per dygn per bil för de olika alternativen.

	Huippukulutus vuoden aikana	Käytännölinen keskikulutus	Autopaikkaa	Lataustunteja / vrk	Auton keskikulutus / 100 km	Liittymispisteen kapasiteetti	Vapaa kapasiteetti	Teho / paikka	Energia / vrk / auto	Ajomatka / vrk / auto
Liittymä										
250 A	56 kVA	40 kVA	6	13	20 kWh	*172.5 kW*	*132.5 kW*	*22.1 kW*	*287.1 kWh*	*1 435 km*
Kiinteistö-säköliittymä										
63 A	15 kVA	9 kVA	6	13	20 kWh	*43.5 kW*	*34.5 kW*	*5.7 kW*	*74.7 kWh*	*373 km*
Syöttö 1										
25 A	kVA		6	13	20 kWh	*17.3 kW*	*17.3 kW*	*2.9 kW*	*37.4 kWh*	*187 km*

Figur 8: Exempel på laddplatsberäkningar.

5 Produktion av elektricitet

I dagens läge börjar privatpersoner och husbolag producera alltmer egen energi, främst med solpaneler men också små vindkraftverk. Dessa används främst för att sälja energi tillbaka till elnätet och tjäna tillbaka en del av den månatliga energikostnaden. Med Comsels smarta laddare finns det en möjlighet att integrera egen elproduktion till laddsystemet. Detta innebär i praktiken att den egenproducerade elektriciteten kan användas för att ladda bilen.

I mobilapplikationen kommer användaren att kunna se den mängd ström som produceras. Detta tas upp i mer detalj under rubriken Smarttelefonapplikation.

6 Utförande

Det kompletta verktyget för kartläggningen kommer att delas in i tre huvudsakliga delar. Den första delen är behandling av elförbrukningshistoriken. Detta kommer att göras i Excel med hjälp av VBA-programmering. Förbrukningsdatan ska sorteras, struktureras och visualiseras för att underlätta för kartläggaren att finna de olika strömpikarna som uppstår och när dessa vanligtvis sker. Slutligen ska även grafer framställas, vilket även detta går bra att göra med VBA-programmering.

Den andra biten är insamling av grundläggande information angående fastigheten och dess ansvariga kontaktpersoner. Detta kommer att ske via ett webbformulär som skapas. Som tidigare nämnts kommer det att skrivas i React med TypeScript och använda MUI biblioteket för grafiska komponenter. Formuläret ska även kopplas samman med Comsel Service Hub för att kunna spara data i en enhet.

Den tredje biten är smarttelefonapplikationen som kartläggaren tar med sig ut på besöket till fastigheten. I den ska all befintlig information visas och kartläggaren ska ha möjlighet att anteckna all relevant information angående elbilsladdstationerna såsom var dessa ska placeras, vilken central som ska bidra med strömmen, vilka säkringar som ligger bakom, förmågan att kunna ta bilder och så vidare. Slutligen ska all denna information laddas upp till den befintliga enheten i hubben.

6.1 Excel automation

Fram tills nu har det varit väldigt mycket manuellt jobb för kartläggaren att gå genom varje fastighet som ska kartläggas en åt gången och göra samma steg om och om igen. Tack vare VBA-programmering kommer det nu att bli den mest automatiserade biten av processen. Med programmet kommer kartläggaren även att kunna se vilken veckodag mätningen är på. Detta går visserligen att göra manuellt för enskilda mätningar som intresserar, genom att kolla upp datumet i kalendern. Men det är inte något som görs för flera tusen rader, det är ett riktigt praktexempel på nyttan med VBA-programmering i Excel. Efter att en veckodag har fastslagits för varje datarad går det även att filtrera datan utifrån vissa dagar, detta kan ytterligare förbättra kartläggarens uppfattning över situationen.

6.1.1 Importering av data

Då vi får en förbrukningsdatafil från Vasa Elektriska så är den i xml-format. Filen innehåller två blad med information. Det första innehåller basinformation, såsom datum, adress, start- och slutdatum för mätningar och vilka huvudsäkringar som skyddar anslutningen.

I den andra fliken finns varje mätning på en enskild rad. Mätningen består av datum, klockslag, effekt, reaktiv effekt och skenbar effekt. Dessa mätningar i sig är det inget fel på men för att bättre kunna visualisera det hela så vill vi sen en graf över datan. Vi vill även kunna strukturera om datumformatet från *dd/mm/yyyy hh/mm/ss* och plocka ut klockslaget för att visa det i en ny kolumn och även ta reda på vilken veckodag datumet är på. Detta för att få en bättre inblick i vilka tider på dygnet och vilka veckodagar som det finns mest och minst ledig effekt på anslutningen.

För att göra processen användarvänlig och dynamisk var planen att skapa en fil som innehåller all VBA-kod. Denna fil ska importera datan från xml-filerna och göra ett antal modifieringar, för att slutligen ge användaren möjligheten att skapa en förbrukningsgraf med endast ett knapptryck.

Den första biten blir att skapa en knapp som är kopplad till en funktion som öppnar en av xml-filerna. Då användaren trycker på knappen kommer ett Windows Explorer-fönster fram, här får användaren välja vilken fil som han önskar öppna. Eftersom alla filer som innehåller förbrukningsdata heter något i stil med exempelgatanXXliittymä.xml eller exempelgatanXXkiinteistö.xml görs först en kontroll som ser på den valda filens namn. Om filens namn inte innehåller kiinteistö eller liittymä någonstans i filnamnet så stoppas importen och användaren får ett felmeddelande. Ifall en godkänd fil är vald går programmet vidare och öppnar den valda filen som en arbetsbok. Sedan öppnas den huvudsakliga filen graph-maker. Alla celler som innehåller data töms, förutom de frysta raderna högst upp som innehåller kommandoknappar och kolumnrubriker. Sedan går programmet tillbaka till xml-filen, till det första bladet som innehåller basinformationen. För att hålla funktionaliteten dynamisk, det vill säga att inte koda markera från den här cellen till den här så används en mycket kort och kraftfull kod rad.

Kodexempel 1: Räkna antalet rader som innehåller data med VBA

```
lastRow = ActiveSheet.Cells(ActiveSheet.Rows.Count, "A").End(xlUp).Row
```

Raden säger att i det aktiva arbetsbladet, räkna hur många rader som innehåller data i kolumn A, resultatet lagras sedan i en variabel. Med denna kontroll kan alla rader med data kopieras

oberoende om det är frågan om 30 eller 3000 rader. Denna metod för att ta reda på hur många rader som innehåller data kommer att användas ett flertal gånger under programmet.

När de använda raderna är kopierade klistras de sedan in i graph-maker-filen. Sedan för att ge användaren ett sätt att veta att de ser på data från rätt fil efter att importen är klar, skapas redan nu en rubrik baserat på filnamnet för den valda xml-filen. Rubriken placeras högst upp i det vänstra hörnet. Sedan går programmet vidare och tömmer det andra bladet, öppnar fliken med förbrukningsdata i xml-filen och räknar hur många rader med data som finns. Alla använda rader kopieras över till graph-maker-filen och även här placeras en rubrik i det övre vänstra hörnet.

Nu när båda sidorna med information har kopierats över från xml-filen till huvudfilen fortsätter programmet med datumformateringen. Programmet markerar hela kolumnen med datuminformation och klistrar in en kopia av den i en ny kolumn bredvid. Kolumnens rubrik sätts till Aika och kommer att innehålla klockslaget. Som tidigare nämnts innehöll den ursprungliga datan ett långt datumformat med både datum och klockslag i samma kolumn och målet är att dela upp dem i två skilda kolumner. För att ta bort klockslaget från datumkolumnerna och datumdelen från tidskolumnen körs följande simpla kodrader.

Kodexempel 2: Ändra tid & datumformatering för kolumner med VBA

```
Columns("A:B").Select
Selection.NumberFormat = "dd/mm/yyyy;@"
Columns("C:C").Select
Selection.NumberFormat = "hh:mm;@"
```

Först markeras kolumnerna A till B och sedan ändras nummerformateringen för markeringen till det angivna. Sedan markeras kolumn C och markeringens nummerformatering ändras till det angivna. För att klargöra, efter dessa ändringar finns nu startdatum för mätningen i kolumn A, slutdatum för mätningen i kolumn B och klockslaget för mätningen i kolumn C. Denna förändring tar dessutom inte bort någon information från den ursprungliga datan, så ifall en av cellerna från någon av kolumnerna markeras så fortfarande den ursprungliga datan som stod i xml-filen ses. Förändringen är endast visuell för att ge användaren en tydligare överblick över datan.

Sedan skapas ytterligare en kolumn för veckodagar. Detta kommer att vara användbart då användaren försöker hitta trender i förbrukningen. Programmet börjar med att skapa rubriken päivä och anropar sedan en funktion som döps till weekday. Funktionen börjar på samma

sätt som tidigare, genom att ta reda på hur många rader med data som finns tillgängliga. Det bör noteras att den befintliga kontrollen för antalet rader inte kan användas eftersom den nya funktionen inte befinner sig i samma vidd som det övriga programmet så variabler som använts utanför funktionen går inte att läsa från eller skriva till. Efter att funktionen vet hur många rader med data som finns tillgängliga körs en loop. I loopen kontrolleras varje cell ur den totala datan en för en. Den kontroll som görs för varje cell i loopen är:

Kodexempel 3: Ta reda på veckodag och lagra i variabel med VBA

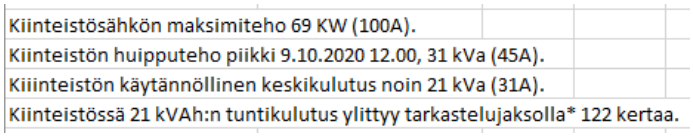
```
weekdayNumber = Application.WorksheetFunction.weekday(cell.Value)
```

Kontrollen som görs är en inbyggd funktion i Excel som tar emot ett datum och svarar med en siffra mellan 1–7, siffran motsvarar en veckodag. Då funktionen tagit redan på vilken siffra som motsvarar datumet i cellen används en select-case som är ett verktyg inom programmering. Det ger programmeraren möjligheten att bestämma vad som ska hända beroende på villkoret. I detta fall är villkoret en variabel weekdayNumber som innehåller en siffra från 1–7, om siffran är en etta så lagras söndag i en annan variabel weekdayText. Likadana instruktioner skrivs för de andra siffrorna så att alla veckodagar hamnar vid rätt siffra. Innan nästa cell i loopen påbörjas är det ännu ett steg kvar, programmet ska skriva in den veckodag som skrevs till variabeln i rätt cell i kolumn D, på samma rad som den cell vi tog datumet från. Detta görs smidigt genom att säga att den aktuella cellen i loopen, den som vi tog datumet från, ska förskjutas till höger två steg och där skrivs slutligen veckodagen in som text. Loopen kommer sedan att fortsätta med nästa rad och upprepa alla steg igen. Detta görs naturligtvis för varje rad som innehåller data.

När denna funktion når slutet hoppar programmet tillbaka till där anropade för weekday gjordes. Sedan ska programmet städa upp lite visuellt i databladet. Denna sektion markerar kolumnerna med data och centerjusterar dem. Rubrikraderna markeras och vänsterjusteras eftersom de kommer att fylla upp den totala bredden av cellen senare, de får även fetstil. De kolumner som innehåller data markeras och filtrering aktiveras. Sedan markeras alla kolumner och deras bred autojusteras för att inte lämna några celler som är onödigt breda eller smala. Slutligen sorteras G kolumnen, som innehåller data över den skenbara effekten och en färgeffekt sätts, effekten går från rött via gult till grönt. Ju högre förbrukning desto rödare blir cellen och ju lägre förbrukning desto grönare blir den. Slutligen stänger programmet xml-filen då dataimporten är färdig, så slipper användaren göra det själv.

Nu är all data importerad men programmet är inte färdigt ännu. En sak som ännu ska göras är att utföra en automatisk analys av datan. Resultatet som kartläggaren vill åstadkomma

illustreras i Figur 9. Eftersom varje bostadsbolag har två skilda uppsättningar med förbrukningsdata kommer en sådan analys att göras för båda. En för fastighetens förbrukning såsom motorvärmare och utebelysning och en för hela anslutningen där alla boendes förbrukning ingår.



Figur 9: Resultat från automatisk analys.

Programmet anropar funktionen analysisLiittymä eller analysisKiinteisto beroende på vilken av filerna som programmet hållet på och jobbar med för tillfället. Båda funktionerna fungerar på samma sätt men har lite skillnader med namngivning på datablad och är därför uppdelade i två olika funktioner.

Funktionen börjar med att skapa den första raden med den maximala effektkapaciteten. Funktionen går till fliken med basinformation och tar informationen om vilka säkringar som används, samma sak görs även för antalet faser. Utifrån dessa två variabler kan funktionen beräkna den största möjliga effekten genom formeln:

$$P = \frac{(U \cdot I) \cdot antalFaser}{1000} \tag{11}$$

Där:

P = Effekten, kW

U = Spänningen, V

I = Strömmen, A

Sedan sammanfogas texten och skrivs ut på den första raden.

För att få till den andra raden börjar programmet med att ta reda på den största förbrukningen i datasamlingen. Datan borde redan vara sorterad så att det största värdet ska vara högst upp, men för säkerhets skull utfös sorteringen igen. Efter det kan det största värdet avläsa ur cellen. Det datum som är på samma rad plockas även ut för att kunna skrivas ut när denna maximala förbrukning ägde rum. Sedan för att omvandla från skenbar effekt till ampere använder vi formeln:

$$I = \frac{\frac{(S \cdot 1000)}{230}}{antalFaser} \tag{12}$$

Där:
S = Skenbar effekt, kVA
I = Strömmen, A

Sedan sammanfogas texten och skrivs ut på den andra raden.

För att skapa den tredje raden används den nyligen sorterade datan. De 100 största värdena skippas och följande värde tas ur datan. Detta som redan tidigare förklarats är för att ta fram ett maximalt värde som stämmer under 99,99 % av tiden för anslutningen. Sedan för att omvandla från skenbar effekt till ampere används formel 12 på samma sätt som vid den förra raden. Slutligen konkateneras texten ihop och skrivs ut på den tredje raden.

Den fjärde och sista raden skapas genom att jämföra det värde som beräknades till den tredje raden med de värden som blev överhoppade. Lösningen här är inte att de 100 första raderna skippas, eftersom ett tal med datatypen Integer användes för att lagra värdet. Detta innebär att den endast använder heltal och avrundar således bort alla decimaler. Så funktionen ges ett kortare dataset som innehåller data en bit över gränsen för att vara på säkra sidan, men samtidigt ska den inte behöva kontrollera tusentals rader då användaren redan vet att det är frågan om cirka 100 rader som berörs. Varje cell jämförs helt enkelt med det beräknade avrundade värdet och ifall det är högre så läggs en etta till på räknaren. Resultatet blir en räknare som säger hur många gånger en förbrukning som är högre än det som vi tidigare beräknade har förekommit. Exempelvis i Figur 9 ser vi att detta hände 122 gånger och inte 100.

Dessa fyra rader ger kartläggaren snabbt en uppfattning över situationen utan att behöva göra något manuellt jobb. De kommer sedan att hamna i den färdiga rapporten som kunden får efter att alla övriga aspekter har tagits i beaktande.

Efter att analysen är utförd fortsätter importprogrammet ännu med att meddela användaren att allt är färdigt, med hjälp av en ruta som dyker upp. Ännu är det en liten städning kvar av arbetsbladet. De kolumner som är tomma ställs in att ha den originala bredden på 8,11 för att ge resultatet ett sammanhängande utseende. Den rubrik som står uppe i hörnet med formatet Exempelgatan 12 liittymä/kiinteistö får även en större font. Allra sist körs en rad som gör att Excel återigen uppdaterar skärmen. Denna funktionalitet stängdes nämligen av i början av programmet för att inte skärmen ska blinka och byta blad fram och tillbaka medan användaren väntar på att programmet ska köras. Det har även kunnat konstateras att programmet körs aningen snabbare om skärmen inte uppdateras på samma gång som

programmet körs, då detta använder processorkraft i onödan. Hela denna process körs på mellan 5–30 sekunder beroende på datorns prestanda och den datamängd som filen innehåller.

6.1.2 Skapa graf

För att skapa en graf med ett knapptryck behöver programmet veta vilken data som användaren vill se i grafen. Användaren är intresserade av att se den avlästa skenbara effekten på Y-axeln och förstås tidpunkten för mätningen på X-axeln. Det ritas även in en horisontell linje som visar den filtrerade maxförbrukning utan de 100 största topparna. För att göra detta på ett dynamiskt sätt görs på samma sätt som flera gånger tidigare. Programmet börjar med den användbara programkoden för att räkna hur många rader som innehåller data. Sedan kontrolleras att datan är sorterad rätt genom att göra en till sortering med datan i fallande ordning enligt den skenbara effekten. Detta steg är nödvändigt ifall användaren skulle ha sorterat om datan manuellt efter att den blev importerad, innan grafen skapas. Efter sorteringen görs på samma sätt som tidigare och tar reda på vilket värde som är på hundraförsta platsen och lagrar ett avrundat värde av detta i en variabel. För att rita in en rak linje i en graf i Excel räcker det inte med att ange värdet en gång per värde på X-axeln utan programmet hamnar att skriva in värdet i lika många celler som det finns tidsvärden. Dessa celler som innehåller den filtrerade skenbara maxeffekten lagras slutligen i en variabel lineY.

Då det filtrerade maxvärdet för den skenbara effekten har fastställts kan även den resterande datan lagras i variabler. Variabeln rngX tilldelas cellerna som innehåller datum från första till sista raden som innehåller data och rngY tilldelas cellerna som innehåller avläsningarna för den skenbara effekten. Sedan när dessa tre variabler är lagrade kan de plottas upp i en graf. Men eftersom graph-maker-filen ska vara återanvändbar och dynamisk kan det inte tas för givet att det inte redan finns en graf som ska skrivas över. Det kan inte heller alltid skapas en ny graf, då detta skapar ytterligare manuellt arbete för användaren som behöver radera den gamla grafen manuellt. Detta skulle även kunna leda till att användaren blir förvirrad över vilken graf som hör ihop med vilket data. Lösningen på detta blir att först kontrollera om det redan finns en graf för just denna data. Detta görs genom att låta programmet skapa två olika grafer, en för anslutningens data och en för fastighetens data. Båda graferna kan existera samtidigt på skilda blad, men ifall en ny graf skapas så raderas den gamla först. På detta vis kommer det inte att finnas mer än två grafer samtidigt som kan orsaka förvirring, samt kommer grafen även att få en rubrik som är länkad till den fil som datan är hämtad från, på samma sätt som den huvudsakliga filen tar sin rubrik från importeringen.

Kontrollen utför en loop som går genom alla datablad som finns i filen. Den kontrollerar om bladets namn är samma som Förbrukningsgraf liittymä eller Förbrukningsgraf kiinteistö beroende på vilken av graferna som skapas. Ifall namnet matchar så kommer detta blad att raderas.

Före programmet skapar det nya bladet och själva grafen ser det till att spara undan den befintliga rubriken från databladet så att den kan återanvändas vid ett senare skede. Sedan skapas ett nytt blad och namnges beroende på vilken av de två graferna som skapas. Det nya bladet för grafen flyttas efter det blad som har samma namn för att ytterligare undvika att bladen blandas ihop. Det nya bladet får sin titel från den variabel där det tidigare lagrade undan. Slutligen skapas själva grafobjektet och dess egenskaper populeras med värden på X- och Y-axlarna. Även vilken typ av graf det är frågan om ges som egenskap för att få den att se ut om användaren önskar.

På detta vis skapas grafen genom ett knapptryck och får en dynamisk funktionalitet där det inte spelar någon roll om det finns 35 eller 10 000 rader med data, en korrekt graf kommer att skapas i samtliga fall. Vi får även med namnet på grafen som är kopplat till den fil som datan har importerats från för att underlätta för användaren att hålla koll på vilken graf som hör ihop med vilken data. Hela processen kommer att göras i två olika funktioner, en för kiinteistö och en för liittymä detta igen för att få rätt namngivning, data och placering av grafbladet.

6.1.3 Snabbfiltrering av data

Med filen i sitt nuvarande skick går det att importera data och skapa tillhörande grafer med endast ett knapptryck. Användaren kan även filtrera datan för att se hur situationen ser ut under olika timmar på dygnet eller under olika veckodagar. Då användaren filtrerar datan kommer grafen automatiskt att uppdateras.

Med all denna funktionalitet är det ytterligare två funktionsknappar som kunde underlätta och snabba upp processen. En av dessa är att tömma alla filter. Då användaren har sorterat i de olika filtren och kryssat ur olika alternativ i en eller flera kolumner, då vore det bekvämt med en knapp som återställer alla filter till att visa all data igen. Detta kan åstadkommas väldigt enkelt genom VBA. Det enda som behövs är den användbara raden som räknar hur många rader med data som finns enligt Kodexempel 1 och en rad till enligt Kodexempel 4:

Kodexempel 4: Återställ filtrerade data till ursprungsläge med VBA.

```
ActiveSheet.Range("A3:G" & lastRow).AutoFilter Field:=1
```

Denna rad säger att på det aktiva bladet använd sektionen där datan börjar till där datan slutar. Med denna sektion sätts autofiltrering och ange fält till ett. Denna etta motsvarar direkt kolumnen A. Kodraden upprepas för de andra kolumnerna som ska återställas till att visa allt så är denna funktionalitet implementerad.

På liknande sätt kan även göras för den andra knappen. Med den är tanken att kartläggaren ska kunna filtrera så att endast de mätningar som är på nattström visas, det vill säga mellan klockan 22:00-07:00. Detta är relevant genom att de som använder börs-el kommer att ladda sina bilar under denna tidsperiod då strömmen är som billigast. Åter igen räknas hur många rader som innehåller data och sedan körs följande kodrader:

Kodexempel 5: Filtrera data för att visa mätvärden för nattström med VBA

```
ActiveSheet.Range("A3:G" & lastRow).AutoFilter Field:=3, Criteria1:=Array(
"22.00", "23.00", "00.00", "01.00", "02.00", "03.00", "04.00", "05.00",
"06.00", "07.00"), Operator:=xlFilterValues
```

Det som händer här är att all data filtreras enligt alternativen som listas inom parentesen enligt kolumn tre eller C där klockslagen finns. Sedan kan ytterligare en filtrerad maxförbrukning för den skenbara effekten tas fram genom att klippa av de största topparna. Denna gång är det inte frågan om lika många mätpunkter så det kan inte klippas lika många toppar, utan programmet klipper endast de 30 största värdena för att få en filtrerad maxförbrukning där de resterande värdena ligger under cirka 99 % av tiden. Programmet skriver sedan ut detta beräknade värde i en konkatenerad textsträng uppe i kanten på filen.

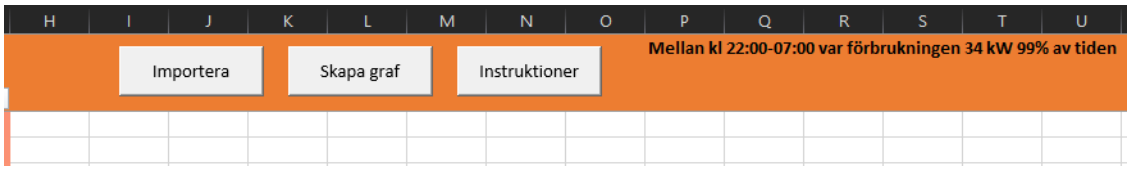
6.1.4 Resultat av dataimport

I Figur 10 syns titeln uppe i hörnet, som är skapad från den fil där datan är hämtad från. Datan som visas i raderna 4–10 har blivit sorterad efter importeringen enligt största förbrukning i kolumn G. Orsaken till att kolumn A och B visar samma sak är för att de även innehåller klockslag. Dessa klockslag är olika så därmed får båda vara kvar. Även de två snabbfiltreringsknappar som implementerades syns uppe i högra hörnet.

	A	B	C	D	E	F	G
1	Exempelgatan 12 liittymä					Visa all data	Filtrera 00:00-08:00
2							
3	Päivämäärä	Päivämäärä	Aika	Päivä	Pätöteho (kW)	Loisteho (kvar)	Näennäisteho (kVA)
4	9.1.2021	9.1.2021	20.00	Lördag	43,993	27,164	51,70364537
5	26.12.2020	26.12.2020	20.00	Lördag	43,829	27,066	51,5126159
6	23.1.2021	23.1.2021	20.00	Lördag	43,738	27,037	51,41995734
7	9.1.2021	9.1.2021	19.00	Lördag	42,024	25,959	49,3952048
8	23.1.2021	23.1.2021	21.00	Lördag	40,197	24,879	47,27328474
9	13.1.2021	13.1.2021	21.00	Onsdag	40,19	24,635	47,13936068
10	17.2.2021	17.2.2021	19.00	Onsdag	40,113	24,657	47,08524629

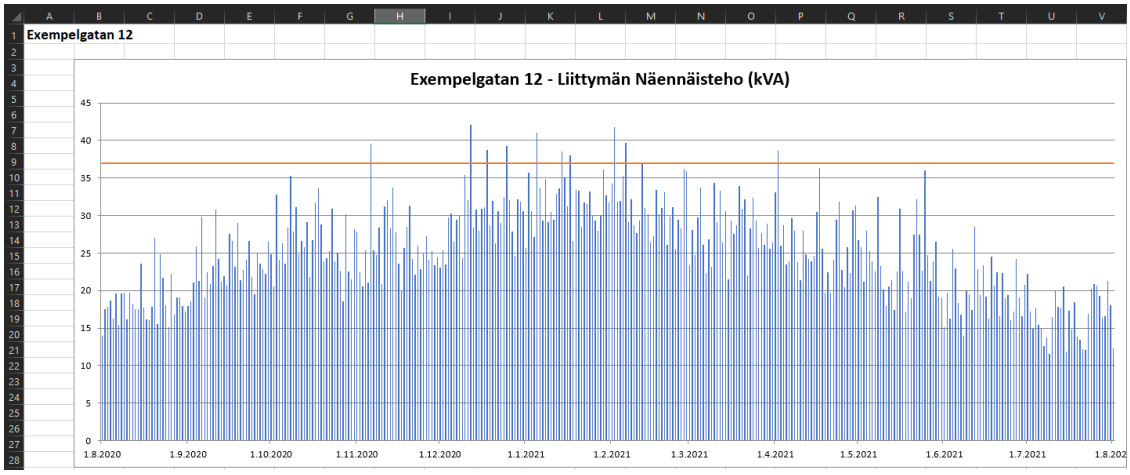
Figur 10: Graph-maker kolumn A-G exempel.

I Figur 11 syns knapparna för att importera data, skapa en graf och även en instruktionsknapp. För att sammanfatta, importeringsknappen öppnar ett Windows Explorer-fönster där användaren får välja en fil. Sedan stängs fönstret och allt sker automatiskt i bakgrunden. Skapa graf-knappen skapar en ny graf med den data som är öppen i filen och kontrollerar om det finns gamla grafer och skriver över dem i stället för att skapa en massa nya grafer. Instruktioner-knappen öppnar ett litet fönster där användare får läsa snabbt om hur programmet fungerar ifall om det är första gången programmet används eller om användaren vill friska upp minnet angående hur programmet fungerar.



Figur 11: Graph-maker kolumn H-U exempel.

I Figur 12 syns en graf som har blivit skapad automatiskt. Filen vars data den är skapad från sätter sitt namn, både uppe i hörnet och som titel på grafen. Även om grafen representerar liittymä eller kiinteistö framgår i titeln.



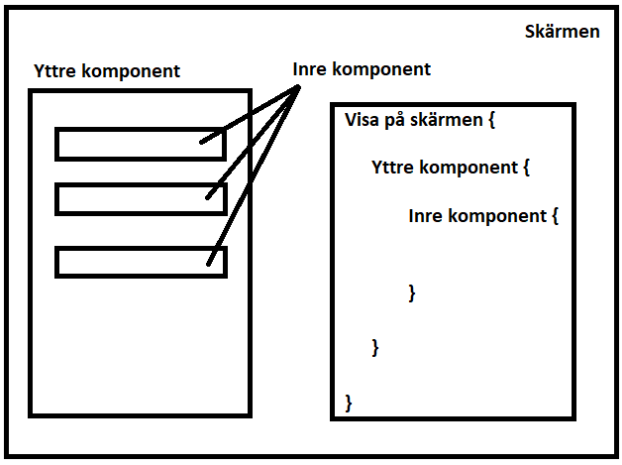
Figur 12: Exempel på graf skapad med graph-maker.

6.2 Webbsida

Då en kund initialt bestämmer sig för att kartlägga sin fastighet genom att kontakta Vasa Elektriska eller Comsel System så kommer de att bli riktade till ett webbformulär. Det är alltså detta formulär som ska skapas. Genom detta formulär ska kunderna kunna ange basuppgifter om fastigheten, hur situationen i allmänhet ser ut och förstås kontaktuppgifter till exempelvis disponent eller bostadsbolagets ordförande.

Efter att ha rett ut vilka uppgifter som ska samlas in kan själva skapandet påbörjas. Ett nytt React-projekt skapas enligt instruktionerna i den officiella dokumentationen. All information som ska samlas in, går att dela in i tre olika kategorier: kontaktpersoner, fastigheten och parkeringen. Dessa tre sektioner och en del andra funktioner kommer att kodas i varsin egen fil, dels för att det hör till god praxis, dels för att undvika att den huvudsakliga filen blir flera tusen rader lång.

Varje enskild fil behandlas som en del av helheten. Varje fil kan ha en egen retur som ritar upp något på skärmen, men den måste inte ha det. Min kompletta webbsida kommer att ha en startfil som knyter samman alla mindre bitar. Dessa bitar kan i sin tur delas in i ännu mindre bitar och så vidare. En ny fil skapas och döps till Form, denna kommer att vara den huvudsakliga filen som alla mindre delar är länkade till. Programmet kommer att köra genom all kod i denna fil och gå in i de mindre delarna en och en för att utföra deras egen kod. I Figur 13 illustreras hur olika delar kan nästlas i varandra för att skapa en helhet.



Figur 13: Exempel på skilda komponenter.

6.2.1 Kontaktpersoner

För kontaktpersonerna skapas en egen fil PersonForm som kommer att vara en underkomponent till Form. Genom detta upplägg kan React uppdatera innehållet för kontaktpersoner skilt från det andra innehållet.

Eftersom användaren förväntas fylla i uppgifter om kontaktpersoner till fastigheten, så skapas ett gränssnitt för personer som innehåller en persons namn, roll i bostadsbolaget, e-post och telefonnummer. Det skapas även en tom variant av gränssnittet, eftersom detta kommer att behövas senare då användaren lägger till en ny person. Detta innebär att namn är tomt, roll är tomt och så vidare. För att förstå varför detta är nödvändigt behöver skillnaden mellan ett tomt värde och null förklaras. Inom programmering har även ett värde som människor uppfattar som tomt ett värde som datorn kan läsa av. Medan ifall värdet är null betyder det att det inte existerar överhuvudtaget. Det kunde jämföras med att du har ett tomt glas på bordet eller du har inte något glas på bordet.

Sedan behövs en array av personer för att hålla koll på om användaren vill skriva in fler än en person. En array är i korthet en samling data som numreras. Tänk det som en bokhylla där varje hyllplan numreras. Du kan exempelvis säga åt någon att hämta det som finns på hylla två. Lagring av personuppgifterna sker på samma sätt, varje persons namn, roll, epost och telefonnummer lagras på ett hyllplan under en viss nummer.

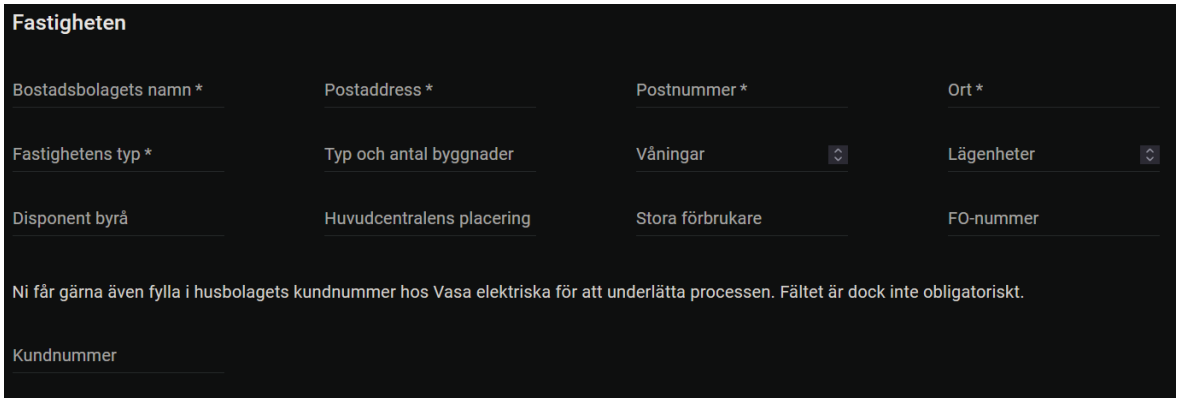
I utgångsläget kommer det finnas en tom rad med fält för användaren att fylla i. Raden kommer även att ha en radera-knapp på slutet. Om användaren trycker på denna knapp skickas det ett kommando åt programmet, att radera den person som har den numreringen. Motsvarande som att säga åt någon, ta bort det som finns på hyllplan två. Slutligen kommer det även att finnas en lägg till knapp efter alla rader för kontaktpersoner. Denna knapp lägger helt enkelt till en ny rad med tomma värden.

Figur 14: Kontaktpersoner-sektion på webbformuläret.

6.2.2 Fastigheten

På samma sätt som med kontaktpersonerna skapas en ny fil PropertyForm som ska hålla reda på allt som har med själva fastigheten att göra. I filen skapas ett gränssnitt för fastighetssektionen, detta innehåller: bostadsbolagets namn, postadress, postnummer, postort, fastighetens typ, byggnader på fastigheten, våningar, lägenheter, disponentbyrå, placering av huvudcentral, stora förbrukare, Vasa Elektriska kundnummer och FO-nummer. Även här skapas en tom variant av gränssnittet som innehåller antingen en tom text eller tomt nummer.

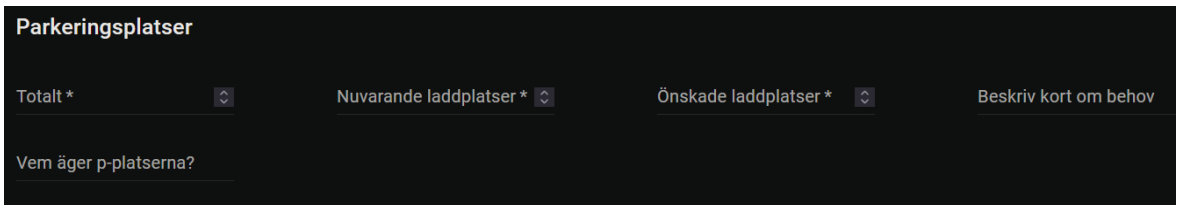
Då användaren fortsätter neråt på sidan efter kontaktpersonerna kommer fälten för fastigheten. Här finns fält för att fylla i alla de uppgifter som listades i gränssnittet ovan. I den här sektionen kan inte användaren lägga till eller ta bort några fält utan de fält som finns från början är de slutliga.



Figur 15: Fastighetssektion på webbformuläret.

6.2.3 Parkeringen

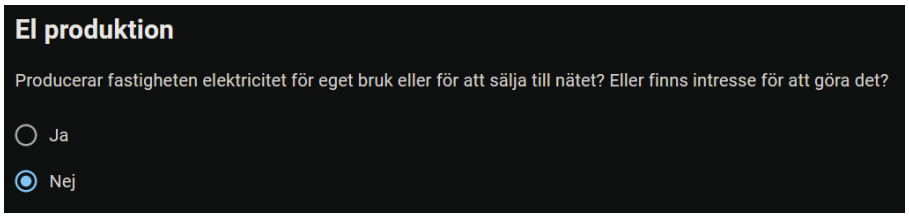
På samma sätt som för de andra komponenterna skapas en ny fil ParkingForm som har ett eget gränssnitt, som består av totala antalet parkeringsplatser, antalet parkeringsplatser där det redan går att ladda, antalet nya platser där laddningsmöjlighet önskas, en beskrivning över situationen och vem som äger parkeringsplatserna. Även här skapas en tom variant av gränssnittet för att kunna ladda in ett tomt formulär vid ett senare skede.



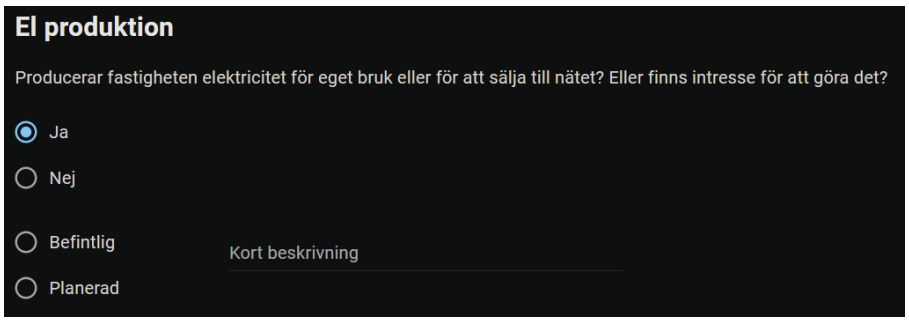
Figur 16: Parkeringssektion på webbformuläret.

6.2.4 Elproduktion

Användaren ska även kunna ange om fastigheten producerar elektricitet för eget bruk eller för att sälja till nätet. Koden för denna funktionalitet har ingen egen fil då den är relativt kort i jämförelse med de tidigare filerna. Användare ska få välja mellan alternativen ja eller nej genom radioknappar. Dessa fungerar så att endast ett alternativ kan väljas. Till skillnad från checkboxar där det går att kryssa i flera på samma gång.



Figur 17: Elproduktionssektion, alternativ dolda.



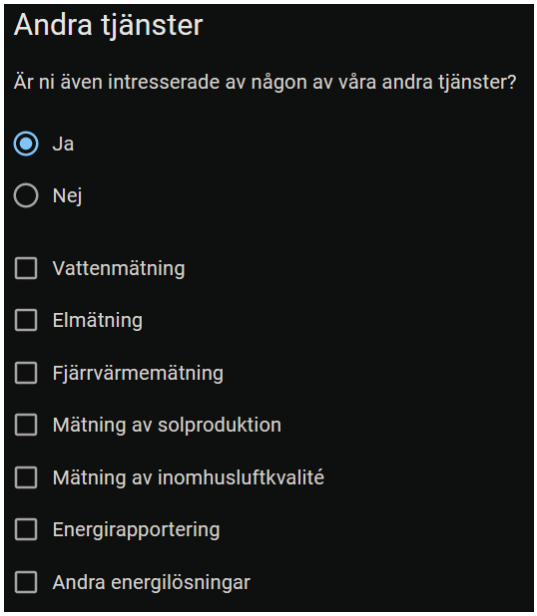
Figur 18: Elproduktionssektion, alternativ synliga.

Om användaren kryssar i ja-alternativet ska det dyka upp ytterligare en omgång radioknappar där användare får välja mellan befintlig eller planerad alltså om det redan nu produceras eller om det är planerat att i framtiden installeras sådan utrustning. Ifall användaren kryssar i nej-alternativet ska dessa ytterligare alternativ gömmas undan. Om användaren kryssar i ja ska det förutom dessa två nya alternativ komma fram ett till fält där användaren får beskriva kort om vad som är installerat just nu eller det som är planerat att installera i framtiden.

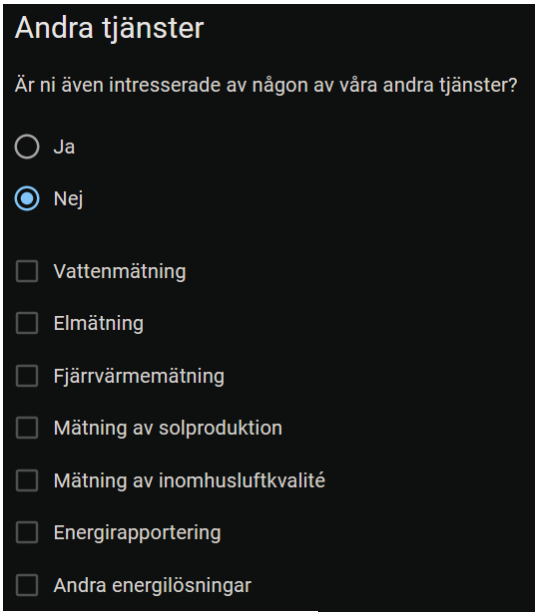
6.2.5 Andra tjänster

Webbsidan kommer på samma gång som den samlar in information om fastigheten, även att kunna samla in information kring om kunden är intresserad av att implementera någon av Comsels Systems andra lösningar. De tjänster som listas att välja mellan är vattenmätningar, effektmätningar, fjärrvärmemätningar, solproduktionsmätningar, inomhusluftmätningar och energirapportering. Tanken är endast att få en uppfattning om kunden är intresserad av någon av dessa, inte att samla mer ingående information om hur det skulle kunna genomföras i praktiken. Det skulle i så fall utföras som en skild kartläggning.

Sektionen på webbsidan börjar på samma sätt som vid elektricitetproduktionsmätningarna genom radioknappar för ja och nej. I utgångläget är alla tjänster inaktiverade, ifall kunden väljer nej så förblir alternativen inaktiva, men om kunden väljer ja aktiveras alternativen. Detta för att en kund är mer trolig att vara intresserad om de kan få en snabb uppfattning om vad som erbjuds före erbjudandet avböjs.



Figur 20: Andra tjänster, alternativ aktiva.

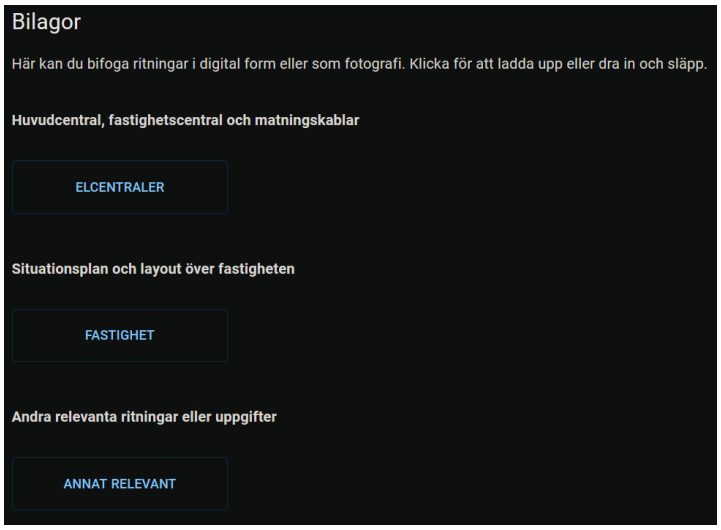


Figur 19: Andra tjänster, alternativ inaktiva.

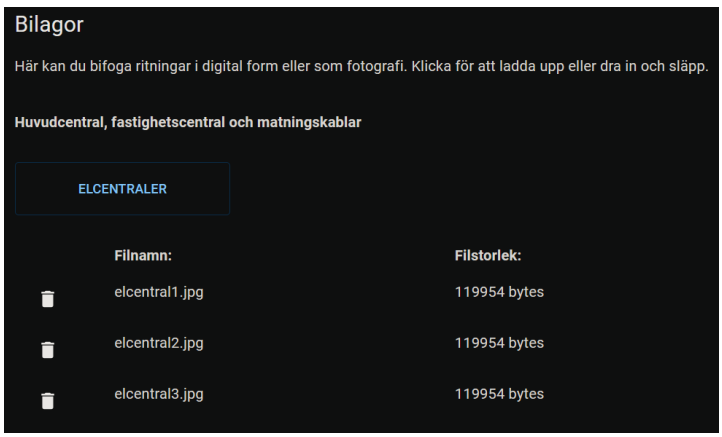
6.2.6 Bilagor

Följande sektion i webbformuläret är bilagor, till skillnad från de två föregående sektionerna är denna sektion tillbaka till att skapas i en egen fil AttachmentForm. Tanken är att kunden ska kunna ladda upp relevanta filer såsom bilder på huvudcentral, fastighetscentraler, matarkabel, situationsplan, elritningar, layout över fastigheten och annat relevant som kan tänkas användas vid kartläggningen.

Bilagor sektionen delas in i tre delar, en för elcentraler, en för fastigheten och en för annat relevant. Dessa sektioner har en varsin knapp som användaren kan trycka på för att välja en eller flera filer att ladda upp. Det går även att dra och släppa filer ovanpå knappen för att ladda upp den/dessa. Varje sektion består av en filuppladdnings-komponent som har ett antal egenskaper och som innehåller den redan beskrivna knappen. Varje komponent har ett tillstånd för filerna. Detta tillstånd kan vara tillagd, laddas upp, uppladdad eller misslyckad. Detta tillstånd används på ett flertal ställen i koden för att hantera hur formuläret ska bete sig. Utöver det så kontrollerar komponenten om filen/filerna som användaren vill ladda upp godkänns eller ej. Då filer har laddats upp visas dess information under den sektion de är uppladdade till. Varje enskild fil får även en radera-knapp.



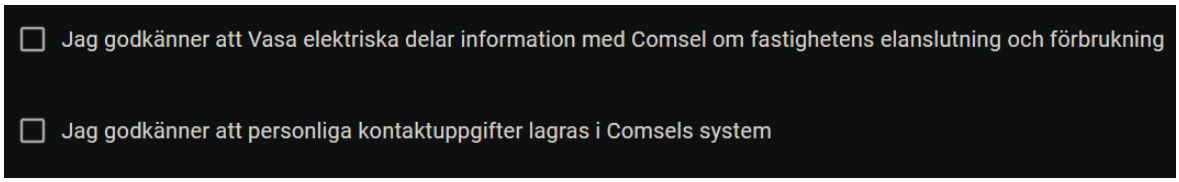
Figur 21: Bilagor sektion på webbformulär.



Figur 22: Exempel på tillagda bilagor under en sektion.

6.2.7 Lagring av personuppgifter

Eftersom vi vill lagra personers kontaktuppgifter på Comsels servrar, kräver detta att vi får personens godkännande enligt den allmänna dataskyddsförordningen GDPR. Vi har inga planer på att använda uppgifterna till marknadsföring eller andra aktiviteter, utan endast spara dem så att vi kan kontakta kunden igen då vi ska komma överens om ett besök på fastigheten för att utföra kartläggningen. I varje fall kräver detta att vi får godkännande av kunden före uppgifterna lagras i systemet. Detta löses genom att användaren måste kryssa i en godkännande ruta för att kunna skicka in sin ansökan och därmed spara sina uppgifter i Comsels system. Det är förstås ingen nytta med att låta användare skicka in en ansökan utan kontaktpersoner då det inte finns någon att kontakta för uppföljningen. Det är dock inte obligatoriskt att låta Comsel använda förbrukningshistorik från Vasa Elektriska. [18]



Figur 23: Godkännande av villkor sektion på webbformulär.

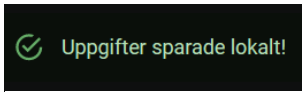
6.2.8 Kontroll av uppgifter

För att undvika att orimliga eller felaktiga uppgifter försöker lagras måste först en kontroll göras på uppgifterna. Då användaren trycker på skicka-knappen görs en kontroll på de uppgifter som behöver kontrolleras. Ifall en kontroll märker att något är felaktigt skapas ett felmeddelande som läggs till i en array med meddelanden. Den första kontrollen som görs är att kontrollera längden på arrayn med kontaktpersoner. En arrays längd kan kontrollers, alltså hur många hyllplan med data som finns. Om det inte finns något är denna längd noll, annars är den exempelvis två om det finns två rader med data. Här gäller det att inte blanda ihop längd och index. I JavaScript, och i förlängningen TypeScript är längden för en array som har två rader med data två medan indexen för dessa datan är noll och ett. Så alltså längden för arrayn med kontaktpersoner kontrolleras, om den är noll, alltså att det inte finns någon data skapas ett felmeddelande. Ifall längden är större än noll, det vill säga att det finns minst en kontaktperson görs nästa kontroll. Denna är helt enkelt att kontrollera att fälten för namn, roll, epost och telefonnummer inte är tomma. Ifall något av dessa fält är tomma skapas ett felmeddelande för detta specifika fält.

Efter att kontaktpersonerna är kontrollerade fortsätter kontrollen med informationen om fastigheten. På samma sätt kontrolleras de obligatoriska fälten för bostadsbolagets namn, postadress, postnummer, postort och fastighetstyp, genom att kontrollera att fälten inte är tomma. Ifall något av fälten är tomma skapas ett felmeddelande för just det fält.

Sedan kontrolleras även uppgifterna för parkeringen. Kontroller görs för totala antalet parkeringar, befintliga laddplatser och antalet nya platser. Ifall något av fälten är tomma skapas ett felmeddelande för detta fält specifikt.

Avslutningsvis görs kontrollen för att se till att användaren har kryssat i rutan för godkännandet av att Comsel lagrar personuppgifter i deras servrar. Efter att alla felmeddelanden har skapats och skrivits till arrayn kommer de nu att visas på skärmen. Dessa meddelanden illustreras i Figur 25. Ifall det inte finns några felmeddelanden och sparandet har lyckats visas i stället meddelandet enligt Figur 24.



Figur 24: Indikering att sparande lyckades.

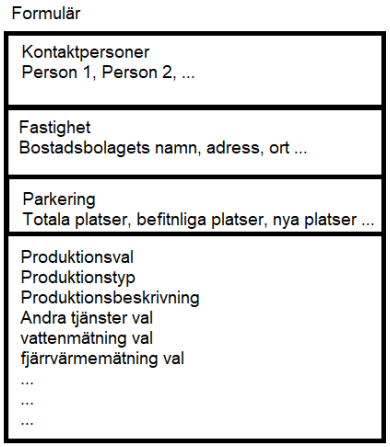
Figur 25: Felmeddelande vid sparning av formulär.

6.2.9 Spara lokalt

Användaren som fyller i formuläret kanske möjligtvis inte vet alla uppgifter som behövs till formuläret, eller av någon annan anledning önskar fortsätta vid ett senare tillfälle. Då behöver formuläret kunna spara den information som redan har fyllts i. Denna funktionalitet kan enkelt åstadkommas genom LocalForage som är ett snabbt och enkelt JavaScript bibliotek för asynkron datalagring.

Den svåraste biten är att paketera ihop alla mindre bitar av formuläret som ska lagras till en helhet. Detta löstes genom att skapa ett nytt gränssnitt som består av det befintliga gränssnittet för kontaktpersoner, fastigheten, parkering, valet för produktion av elektricitet, typen av produktion, beskrivningen över produktionen, valet om intresse för övriga tjänster och alla enskilda tjänster om de är valda eller ej.

När användaren sedan trycker på spara-knappen sköter gränssnittet om att plocka ut informationen som redan är knuten till de befintliga gränssnitten från sina respektive fält. Även förstås från varje enskild box som går att kryssa i. Datastrukturen som gränssnittet skapar är illustrerad i Figur 26.



Figur 26: Exempel på formulärdatastruktur.

När datan är strukturerad enligt exemplet sparas den lokalt i webbläsarens lagring med en enda rad. Kommandot behöver endast en unik nyckel och vilken information som ska sparas.

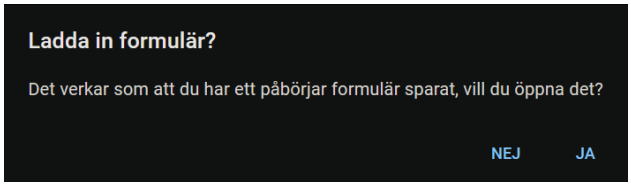
Kodexempel 6: Spara data under nyckel med Typescript.

```
localforage.setItem(LOCALFORAGE_KEY, form);
```

Nyckeln är sparad i en konstant för att undvika att den skulle ändras av misstag och datan som ska sparas är det som finns i form, som är det gränssnitt som nyss skapades.

6.2.10 Ladda från lokal lagring

Efter att användaren har sparat sitt halvfärdiga formulär i webbläsarens lokala lagring behöver programmet förstås även kunna hämta ut informationen igen. Även detta görs relativt enkelt. Varje gång som användaren ansluter till webbsidan kommer en kontroll att köras som kontrollerar om det finns information sparad i webbläsarens lagring. Om det finns data sparat kommer en ruta att dyka upp på skärmen som frågar om användaren vill ladda in det data som finns lagrat eller börja från ett tomt formulär. Programmet använder samma nyckel som datan sparades under i det föregående steget, för att hitta informationen igen.



Figur 27: Ladda formulär från webbläsarens lagring fråga.

6.2.11 Asynkron programmering

Till skillnad från vissa andra synkrona programmeringsmiljöer så är denna tillämpning med TypeScript i React asynkron. Att programkod körs synkront betyder att varje rad med kod körs en åt gången. Nästa rad väntar alltså på att köras tills den föregående är färdig. Så är inte fallet vid asynkron programmering. Då startar raderna visserligen en efter en men det är inte sagt att de kommer att bli färdiga i den ordning som förväntas. Detta kan ställa till med massvis med problem. Exempelvis om resultat ska lagras från en funktion i en variabel, så kan det hända att raden som försöker lagra värdet körs redan före funktionen har hunnit returnera ett värde.

Det finns ett antal olika sätt att tackla detta på. Ett sätt är att använda en timeout som väntar en viss tid före den fortsätter att köra koden. Ibland kan detta tillvägagångssätt vara lämpligt men för detta projekt valdes oftare att använda effekter. Dessa fungerar genom att sätta tydliga gränser för var koden som ska köras börjar och slutar. Sedan sägs vad som ska göra så att denna kod ska köras.

Kodexempel 7: Syntax för en effekt med Typesript.

```
useEffect(() => {
   console.log(”Nu körs koden”)
}, [variabel])
```

I exemplet demonstreras att koden som ska köras börjar och slutar med de mjuka klamrarna. Inom de hårda klamrarna på slutet anges vad som ska orsaka att effekten körs. I exemplet används en variabel som ska trigga effekten. Det spelar ingen roll vad som händer med variabeln, utan så fort något gör att den ändras kommer effekten att köras. Klammarna kan även lämnas tomma för att göra så att effekten körs endast en gång då programmet först startar.

Dessa effekter har varit väldigt användbara för att få helheten i programmet att fungera som det ska. De har använts till bland annat, att kontrollera om användaren har data sparad i webbläsaren, om användaren har öppnat sidan genom en länk, köra funktioner då filuppladdningen ändrar läge och för att uppdatera listor.

6.2.12 Spara till Comsel Service Hub

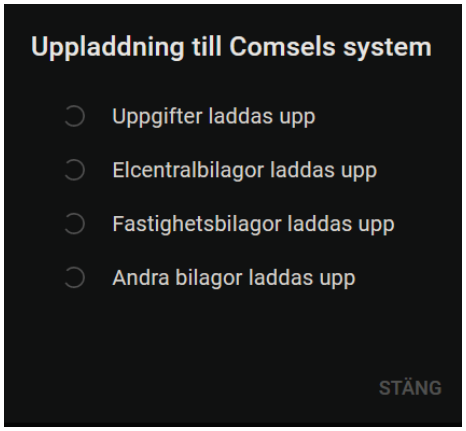
När användaren är färdig med att fylla i formuläret och har laddat upp sina bilagor börjar det vara dags att slutligen ladda upp all information till Comsels Service Hub. Hubben är ett befintligt system som Comsel använder för att spara information och avläsningar från sina andra produkter. Historiskt sett har den inte varit planerad att användas för denna typ av datalagring specifikt. Detta innebär att datastrukturen behöver paketeras om för att passa in med systemet.

Sparandet börjar med att skapa en ny enhet i hubben som det internt kallas. Varje enhet har möjlighet att lagar viss information angående var, när och hur givare har blivit installerade och hur de arbetar. Varje enhet får en Universally Unique Identifier, som förkortas UUID, som är en 128-bitars sträng bestående av siffror och bokstäver. Denna används för att referera till just denna enhet och är garanterad att vara unik.

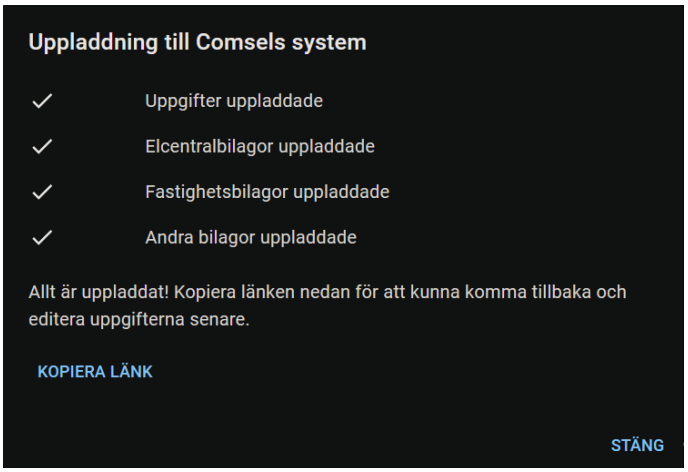
Några av fälten såsom adress och bostadsbolagets namn går att spara direkt till motsvarande fält i enheten, medan andra fält inte passar in någonstans. Detta löses genom att spara den informationen som metadata. Tack vare denna metadata kan information som annars inte passar in någonstans även lagras. Efter att all information är strukturerad och tilldelad en lämplig plats i datastrukturen kan slutligen allt skrivas till den nya enheten.

Efter att användaren trycker på skicka-knappen kommer en ruta upp på skärmen där användaren kan se hur processen fortlöper. Rutan innehåller en status för de angivna uppgifterna, status för elcentrals bilagor, fastighetsbilagorna och status för övriga bilagor.

Eftersom programmet är asynkront kommer uppladdningen av alla sektioner att påbörjas samtidigt, men kan bli färdiga i vilken ordning som helst.



Figur 28: Uppladdning till hub pågår.

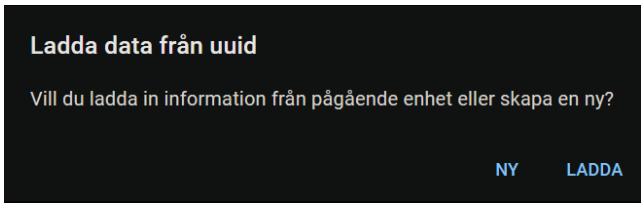


Figur 29: Uppladdning till hub färdig.

Vartefter som en sektion blir färdig kommer den att byta status i rutan från en snurrande indikator till ett färdigtecken. När alla sektioner är färdiga kommer ett nytt meddelande fram om att uppladdningen är färdig. Samt dyker en till knapp upp som användaren kan trycka på för att kopiera länken till formuläret. Denna länk innehåller den nyligen skapade enhetens UUID och kommer låta användaren återvända vid ett senare tillfälle för att ytterligare editera informationen.

6.2.13 Ladda från Comsel Service Hub

Efter att användaren har sparat sin information till hubben fås en länk som kan användas för att komma tillbaka till formuläret för att editera i efterhand. Detta görs genom att kontrollera navigationsfältet då användaren ansluter till sidan. Om de har öppnat sidan via länken och har loggat in med matchande användarkonto kommer sidan att fråga om de vill öppna sin befintliga data eller börja på en ny.



Figur 30: Ladda från länk fråga.

Kontrollen söker efter ett UUID i navigationsfältet. Detta sparas sedan undan i en variabel för att bland annat kunna användas ifall användaren önskar spara undan sin uppdaterade information igen utan att skapa en ny enhet.

Då användaren väljer att ladda in data från sin UUID så görs en efterfrågan till hubben som skickar tillbaka all information som hör till enheten. Den informationen som blev sparad i riktiga fält såsom adress och bostadsbolagets namn hittar direkt till sina rätta platser. Men allt som är sparat som metadata måste sorteras och placeras var för sig för att hamna i rätt fält. En array med metadata skapas och loopas genom, alla värden gås genom ett åt gången. För varje värde jämförs dess nyckel med de fält som finns tillgängliga. Blir det en match så placeras det värdet i fältet.

6.2.14 Mobil-layout

En webbsida i dagens läge brukar antingen ha en skild sida för större och mindre skärmar eller så använder de samma sida som skalas om för att passa olika skärmstorlekar. Ofta kan den första metoden använda två olika varianter av sidan som har lite olika funktionalitet. I detta examensarbete kommer endast en variant av webbsidan att kodas, därmed behöver den skalas om beroende på skärmstorleken. För att göra webbsidan dynamisk så att den ser bra ut i alla olika storlekar av skärmar finns en smidig metod att göra just det i React.

```
<Grid item xs={6} sm={6} md={4} lg={3}>
    <TextField
    label="Lägenheter"
    type="number"
    variant="standard"
    value={props.property.apartments}
    onChange={handleApartmentsChange}/>
</Grid>
```

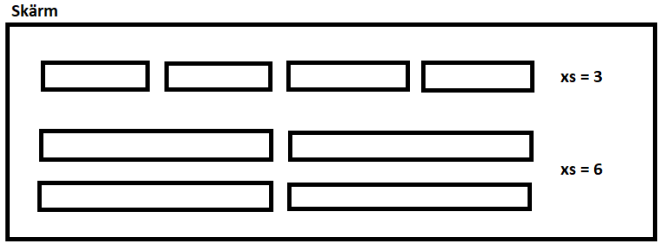
Figur 31: Exempel på olika brytpunkter för samma komponent.

I Figur 31 ser vi på första raden några tecken xs, sm, md och lg. Dessa är brytpunkter i layouten. De står för olika skärmstorlekar och talar om för React hur stor komponenten ska vara i relation till skärmstorleken. Dessa gränser går även att ändra på ifall behovet finns men i min tillämpning fick de vara standard som är:

- Xs, extra liten, området 0–599 pixlar
- Sm, liten, gräns 600–899 pixlar
- Md medium, gräns 900–1199 pixlar

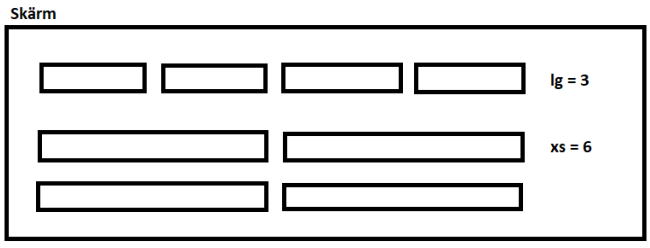
- Lg stor, gräns 1200–1535 pixlar
- Xl extra stor 1536– pixlar [19]

Xs täcker området 0–599 pixlar på bredden av skärmen. Det vill säga att majoriteten av alla smarttelefoner använder den här brytpunkten. Renderaren ritar alltså den maximala skärmbredden, per rad och bryter sedan. Då layouten designas kan värdena väljas fritt för att säga hur stora elementen ska vara i förhållande till skärmstorleken. Den totala bredden för skärmen räknas som 12. I exemplet i Figur 32 visas på den första raden att alla rutor är tre breda. Detta gör att alla fyra ryms på samma rad eftersom totalen blir 12. Sedan på de nedre raderna illustreras att komponenterna är sex breda. Efter att de två första komponenterna är utritade kommer renderaren att bryta raden för att fortsätta på följande.



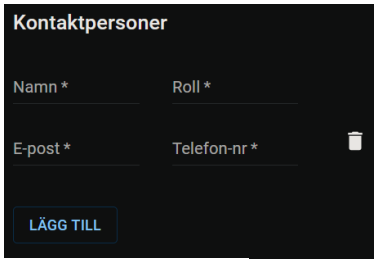
Figur 32: Layoutexempel.

Sedan kan olika bredder användas för olika skärmstorlekar på samma gång enligt exemplet i Figur 33. Om skärmen är stor så kommer alla fyra element att hamna på samma rad medan om vi har en extra liten skärm så kommer dessa fyra att delas in i två rader genom att ge xs och lg olika värden.

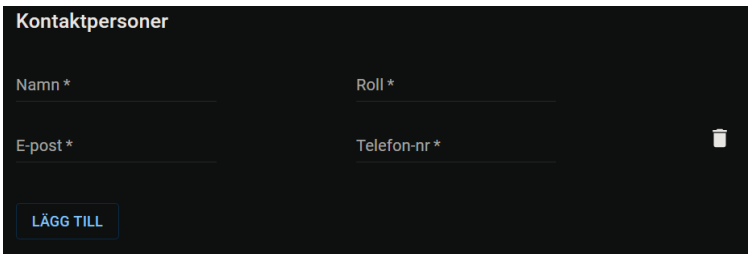


Figur 33: Layoutexempel för olika skärmstorlekar.

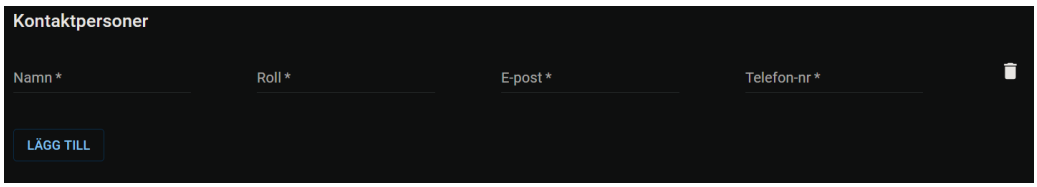
Då koden körs på webbsidan kan användaren jämföra hur layouten för samma komponent kommer att se ut då den visas i en smarttelefon, en surfplatta och en vanlig datorskärm. För större skärmar eller tv-skärmar är det ingen större skillnad från den vanliga datorskärmen, det blir bara mer tomrum på sidorna. Detta är förstås nödvändigt för att inte få en väldigt inzoomad vy då webbsidan besöks från exempelvis en smart-tv.



Figur 35: Exempel på layout i mobiltelefon.



Figur 34: Exempel på layout i surfplatta.



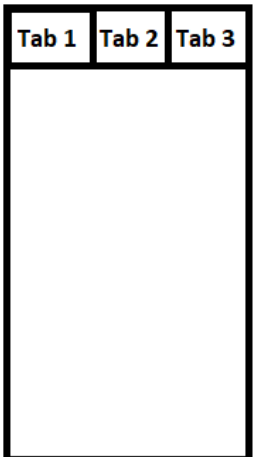
Figur 36: Exempel på layout i dator- och tv-skärm.

6.3 Android-applikation

Comsel har redan en annan applikation sedan tidigare som används i liknande ändamål. I den applikationen ligger fokusen på att installera deras produkter vid en fastighet. Exempelvis om en Oculus som ska kommunicera med anslutningens elmätare, för att sedan sända tillbaka mätdata till hubben ska installeras används denna applikation. Det är naturligtvis frågan om helt andra uppgifter som ska fyllas i, men det finns en mängd användbar funktionalitet och strukturer redo att användas i Comsels bibliotek, som har blivit skapad åt denna tidigare applikation.

Utgångsläget blir således att skapa ett nytt React Native projekt och kopiera över en del från den befintliga applikationen. Den funktionalitet som kan återanvändas är att kunna välja mellan olika enheter som finns i hubben. Efter att kartläggaren först har loggat in i applikationen möts han av ett fönster där det går att söka efter enheter enligt ort. Om användaren har gett applikationen platsåtkomst kommer den även automatiskt att lista de tio närmaste enheterna till den aktuella positionen. Tanken är alltså att kartläggaren ska kunna åka till fastigheten, ta fram applikationen och välja rätt enhet ur listan.

Med denna funktionalitet på plats kan skapandet av ny kod påbörjas. En grundlayout för applikationen påbörjas som första åtgärd. Denna sektion av applikationen kommer att visas efter att användaren har valt en enhet från den tidigare sidan. Layouten består av tre flikar. Dessa tre flikar kommer att innehålla sektionerna, Fastighet, Elcentral och Parkering. Användaren ska kunna svepa mellan flikarna eller trycka på flikarnas ikoner för att byta vilken flik som visas på skärmen. Detta åstadkoms med en React Native-komponent som heter Tab-navigator. Varje flik i komponenten får ett namn som kommer att stå på fliken, ett innehåll som i detta fall kommer att vara ett funktionsanrop till en skild fil, och slutligen lite stilegenskaper för att få flikarna att följa Comsels designtema. Förutom dessa tre flikar kommer grundlayouten även att ha bostadsbolagets namn högst upp i fönstret, detta värde ändrar inte fastän användaren byter flik.

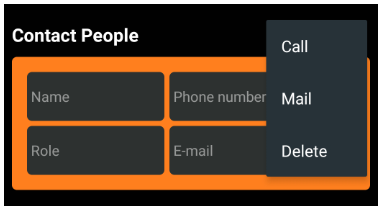


Figur 37: Grundlayout för smarttelefon-applikation.

6.3.1 Fastighetsfliken

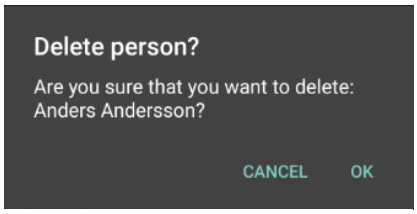
Under fastighetsfliken kommer information om kontaktpersoner och fastigheten. Fliken är indelad i två mindre sektioner, fastighet och kontaktpersoner. Den första sektionen består av en rubrik och textfält. De fält som är listade är, bostadsbolagets namn, disponentbyrå, postadress, Vasa Elektriska kundnummer, fastighetstyp, fastighetens byggnader, våningar, lägenheter och stora förbrukare. Samtliga av dessa fält får sin information från hubben, utgående ifrån den information som kunden har angett i webbformuläret. Kartläggaren kan editera dessa värden genom att ändra värde i fälten där de visas. Fälten görs med React Native-komponenten som heter TextInput.

Den andra sektionen kontaktpersoner, består av de orangefärgade rutorna som syns i Figur 38. När användaren trycker på lägg till-knappen kommer en till sådan orange ruta med nya tomma fält. Alla dessa fält hör till en enskild person. Varje sådan ruta har dessutom en liten meny som kommer fram då användaren trycker på menyikonen. I menyn finns alternativ för att, ringa, mejla eller radera personen. Dessa funktioner implementerades med React Natives Linking som fungerar för både Android och iOS.



Figur 38: Kontaktpersonmeny.

Då användaren trycker på radera-knappen kommer en ruta upp som frågar om användaren är säker på att radera personen. Detta för att förhindra att av misstag ta bort viktig information. Varningen använder det namn som står i fältet för att användaren ska kunna vara säker på vilken person som kommer att raderas. Varningen skapas med React Native-komponenten som heter Alert.



Figur 39: Radera kontaktperson-kontroll.

Ett gränssnitt skapas för kontaktpersonerna på samma sätt som i webbformuläret, för att hålla koll på namn, roll, epost och telefonnummer för varje person. Personen lagras sedan i en array på samma sätt som i webbformuläret. Under sektionen för personer i fastighetsfliken skapas en person-komponent för varje person som läggs till. Komponenten håller koll på

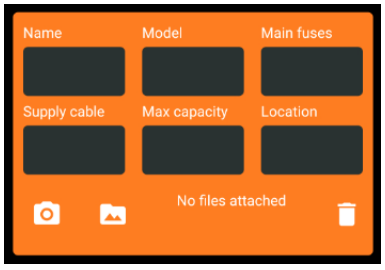
varje persons enskilda uppgifter och att rätt person blir raderad ifall användaren vill radera en person. En helhet över fastighetsfliken kan ses i bilaga 1.

6.3.2 Elcentralsfliken

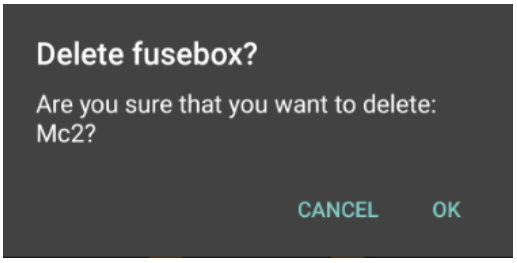
Elcentralsfliken är den som innehåller mest information. Den har skilda sektioner för huvudcentralen, undercentraler, förbrukningsmätare, kabeldragning och elritningar. Ett flertal av dessa sektioner använder kamerafunktionalitet för att kartläggaren ska kunna ta minnesbilder för att sedan i lugn och ro tillbaka på kontoret kunna se över alla detaljer en gång till.

Huvudcentralen är den sektion som kommer först på sidan. Här kan användaren samla information om centralens modell, anslutningens säkringar, matarkabelns typ, matarkabelns maximala kapacitet och var själva centralen är placerad. Det enda av dessa fält som kommer från den förifyllda informationen från kunden är centralens placering. Användaren kan även använda komponentens kamerafunktion för att ta bilder med kameran eller välja befintliga bilder från galleriet.

Sedan kommer sektionen med undercentraler. Det går inte att veta på förhand hur en fastighets elsystem ser ut. Ibland kan det vara bästa lösningen att ta matning till elbilsladdarna direkt från huvudcentralen, medan det ibland kan vara en annan mindre central närmare parkeringen som lämpar sig bäst. På samma sätt som för kontaktpersonerna har varje enskild undercentral en egen komponent. Denna består av ett namn, modell, huvudsäkringar, matarkabel, maximal kapacitet för matarkabel och var den är placerad. Varje enskild undercentral har dessutom kamerafunktionalitet så kartläggaren får en skild uppsättning bilder för varje central. Slutligen en radera-knapp som på samma vis som kontaktpersonerna har en kontroll före den raderas för att förhindra att användaren av misstag råkar raderar en central.

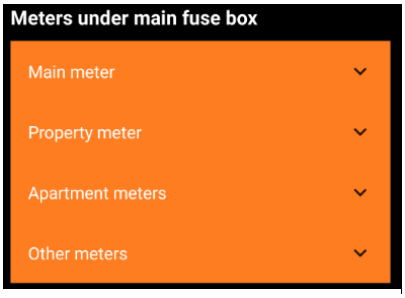


Figur 40: Parkerings-komponent.

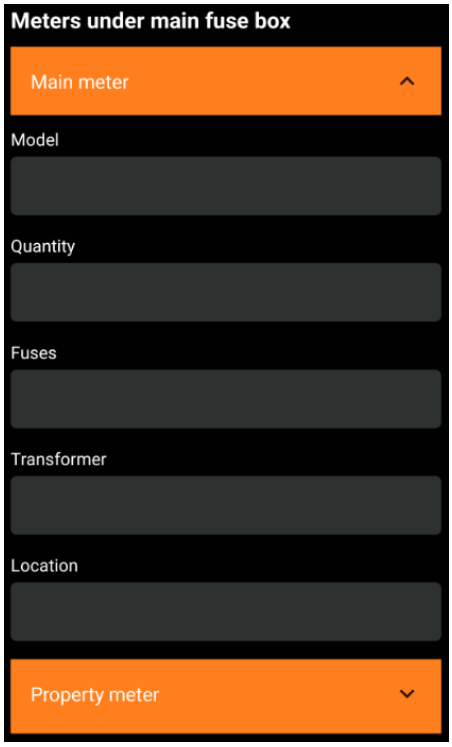


Figur 41: Radera parkering-varning.

Sedan följer sektionen med mätare. Här får kartläggaren fylla i information om de mätare som kan tänkas finnas vid fastigheten. De olika typerna är huvudmätare för anslutningen, fastighetsmätare för den ström som går till fastigheten exempelvis motorvärmarstolpar och gårdsbelysning, lägenhetsmätare för de individuella bostäderna och eventuella andra mätare. Varje kategori av mätare har fälten modell, antal, säkringar, transformator och placering. För att undvika att skriva ut 20 fält efter varandra i en svårtolkad röra används i stället en React Native-komponent som heter List.accordion. Denna komponent skapar små sektioner som går att öppna och stänga en åt gången. På detta vis behöver inte applikationen visa fler än fem fält åt gången. Användaren får öppna sektionerna en åt gången och fylla i de fält som anses vara nödvändiga vid den aktuella fastigheten.

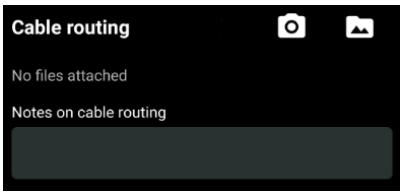


Figur 42: Mätare sektion, stängd.



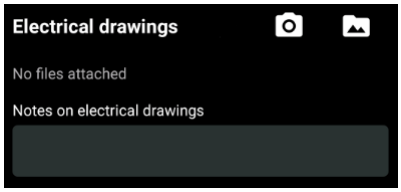
Figur 43: Huvudmätarsektion öppen.

Sedan följer sektionen om kabeldragning. Denna sektion är relativt kort i jämförelse med de tidigare sektionerna. Den innehåller endast ett fält för att fritt formulera anteckningar angående situationen. Utöver fältet har den även kamerafunktionaliteten och allt som hör därtill.



Figur 44: Kabeldragningssektion.

Den sista sektionen på fliken är elritningar. Den är nästan identisk till kabeldragningssektionen med att endast ha ett fält för fri formulering och kamerafunktionaliteten för att ta bilder av ritningarna. Elritningar kan behövas för att planera hur nya kablar ska dras ifall det blir aktuellt. En helhet över elcentralsfliken kan ses i bilaga 1.

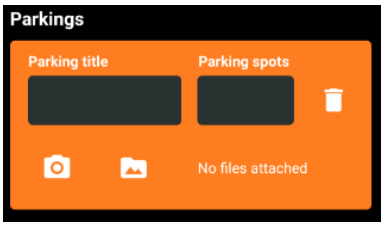


Figur 45: Elritningssektion.

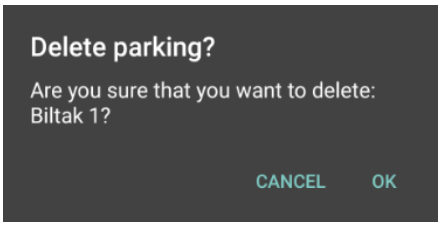
6.3.3 Parkeringsfliken

Parkeringsfliken är indelad i två sektioner, en med information om parkeringsplatser och en som listar parkeringarna som komponenter. I den första sektionen finns fälten, totala antalet parkeringsplatser, antalet befintliga platser med laddmöjlighet, önskat antal nya platser med laddmöjlighet, vilken central som strömmen ska tas från, kort beskrivning över situationen vilket fås som metadata från kundens beskrivning i webbformuläret, och ett fält för fri formulering och anteckningar.

I den andra sektionen listas parkeringskomponenterna. Dessa har liknande struktur som kontaktpersoner och undercentraler. De fält som finns i en parkerings-komponent är titel och antalet platser. Detta för att det exempelvis kan finnas ett biltak, ett antal garage och en sandparkering i samma bostadsbolag. Då är tanken att kartläggaren ska kunna få dem listade som skilda parkeringar och ta bilder på dem skilt. Således har parkerings-komponenten även kamerafunktionalitet. Slutligen även en varning före parkeringen raderas som använder parkerings namn i varningen för att användaren ska kunna vara säker att rätt parkering är på väg att raderas.



Figur 46: Parkerings-komponent.



Figur 47: Radera parkering-varning.

6.3.4 Kamerafunktionalitet

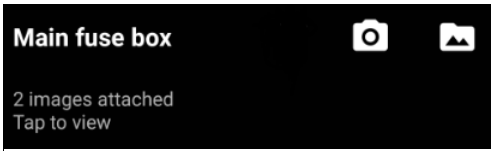
Denna funktionalitet implementerades med en komponent som heter Imagepicker som är från ett externt bibliotek. Varje sektion i applikationen som har kameramöjlighet är uppbyggd på samma sätt. Ikonerna för att ta bild eller välja från galleriet är placerade på ett sådant sätt att det är tydligt vilken sektion de hör till. När användaren trycker på någon av knapparna kontrollerar programmet om användaren har gett applikationen rättighet att använda kamera och tillgång till lagring på enheten. Ifall användaren inte har gjort det så poppar det upp en ruta som frågar användaren om godkännande av rättigheterna. När programmet har fått rättigheterna kan kameran eller galleriet öppnas.

Biblioteket som används för Imagepicker-komponenten har ett antal inställningar som kan ändras i programkoden. En av dessa är att inkludera en base64 textsträng. Detta innebär att bilden som tas med kameran eller väljs från galleriet även sparas som rådata i base64 format. När bilden slutligen ska laddas upp till hubben kommer det inte att gå att ladda upp den som en fil på samma sätt som på webbsidan genom att bara välja filen och säga att den här filen ska laddas upp, utan filen som ska laddas upp behöver ha en viss struktur. Denna är bredd, höjd, mapp, länk till var den är lagrad på enheten, numrering och slutligen innehållet i rådata formatet base64.

För varje bild som tas med kameran eller väljs från galleriet skapas filen med de egenskaper som hör ihop med den aktuella bilden. Den nya filen hamnar i rätt array som hör till den aktuella sektionen och arrayn sparas i en tillståndsvariabel tills att den ska laddas upp till hubben.

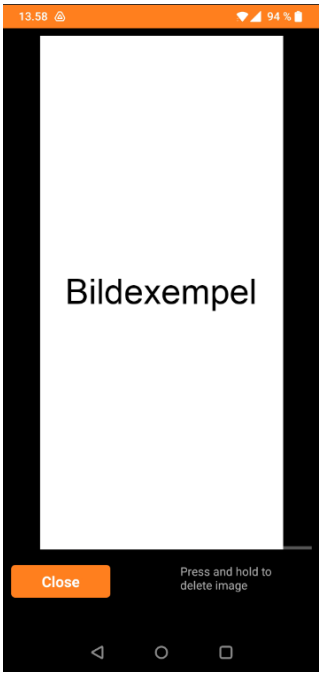
För att sedan kunna se de bilder som är tagna kan användaren trycka på texten som fungerar som en länk. Detta öppnar en React Native-komponent som heter modal denna komponent öppnar ett fönster ovanpå det aktuella fönstret för att visa något annat innehåll tillfälligt. Tanken är att den ska stängas för att fortsätta vidare i applikationen. Denna komponent lämpar sig bra för att visa bilderna i fullskärm.

Figur 49: Inga bilder tagna

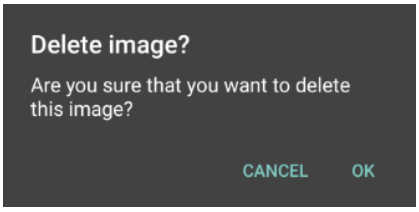


Figur 48: Bilder tagna

Den modal-komponent som skapas då användaren trycker på texten öppnar ett fönster ovanpå den aktuella fliken. Denna komponent är i sin tur uppdelad i mindre delar. Komponenten som visar bilderna är ingen vanlig bildvisare utan en komponent som heter GallerySwiper som kommer från ett externt bibliotek. Denna komponent tar in en array av bilder och visar dem på samma sätt som det inbyggda bildgalleriet i smarttelefoner som vi är vana att använda gör. Det ger även användaren möjlighet att nypa för att zooma in, vilket inte den bildvisare som är inbyggd i React Native gör. En liten indikator i form av ett streck syns även nere i kanten på bilden, den indikerar vilken bild som är synlig. Då användaren sveper mellan bilderna flyttar indikatorn på sig för att ge feedback till var i bildsamlingen som den aktuella bilden befinner sig. Ifall användaren vill radera en bild går det även bra att göra genom att trycka och hålla på bilden för att ta fram en ruta som frågar om användaren är säker på att de vill radera bilden. För att få till denna radera funktionalitet används Galleryswiper-komponentes egenskap onLongPress, den anropar i sin tur en funktion som skapar Alert-komponenten som dyker upp. Då användaren är färdig på modalsidan så stängs den genom att trycka på close-knappen.



Figur 50: Modal-komponent.



Figur 51: Radera bild-varning.

6.3.5 Kontext

Eftersom applikationen är uppdelad i ett antal mindre komponenter som kan behöva ha tillgång till samma information behöver en kontext skapas för varje information som behöver delas mellan filerna. En sådan är inloggningsuppgifterna. De fylls i av användaren då de först startar applikationen och kommer sedan att sparas i kontexten. Efter att detta är gjort så har alla andra filer även tillgång till att läsa av den informationen som är sparad där. Ett annat exempel är information om enheten. Efter att en site har valts ur listan efter inloggningen så kan dess information sparas i kontexten för att de andra filerna sedan ska kunna komma åt uppgifter som är lagrade direkt i enheten och inte i metadata.

Ännu en sak som sparas i en kontext är en lista över filer som ska laddas upp till hubben när det begärs. Då behöver alla sektioner av applikationen kunna se om det finns befintliga filer i listan före de lägger till sina egna, för att undvika att de inte skriver över det som redan finns i listan. Ännu en sak som sparas i en kontext är temat för applikationen. Det är mycket lättare för kodaren att ändra färgkod på ett ställe och hela applikationen byter utseende, än att gå in på ett okänt antal ställen och ändra på färgkoden. Metadata är också en sak som sparas i kontext för att samtliga flikar ska ha tillgång till det som har hämtats från hubben, och sortera ut det som den fliken är intresserad av. Dessutom slipper programmet hämta metadata från hubben ett flertal gånger om denna teknik används.

6.3.6 Läsa metadata

Efter att användaren har valt en enhet ur listan hämtar kontexten all metadata och spara allt i en och samma lista. De enskilda flikarna i applikationen får i sin tur utföra en egen sortering, för att plocka ut den datan de är intresserade av. De letar efter den metadatan som har de nycklar som hör till de fält som finns i den fliken. Att använda en kontext i kombination med effekter för denna funktionalitet visar sig fungera mycket väl, då metadatan sorteras och skrivs in i fälten under alla flikar så fort som metadatan är tillgänglig. Eftersom effekten för varje flik som ska utföra sorteringen ställs in att triggas då metadatan i kontexten ändras, som bara händer en gång då enheten laddas från hubben.

Efter att användaren har valt en enhet ur listan tar det cirka två sekunder och sedan finns metadatan i alla fält i hela applikationen utan att användaren behöver vänta någon fler gång. Detta ger användaren en känsla av att applikationen känns snabb, då den bara behöver ladda en gång.

6.3.7 Spara till hubben

Då användaren är färdig med att fylla i all information i de olika flikarna bör applikationen även kunna spara undan all information tillbaka till hubben. Detta kan göras på lite olika sätt beroende på vilken typ av information det är frågan om. För textfälten gäller en liknande procedur som för att spara till hubben från webbsidan. De fält som finns direkt under enheten som bostadsbolagets namn och adress exempelvis, kan skrivas direkt till motsvarande fält i enheten. Medan all annan information som inte har egna fält i datastrukturen i hubben sparas som metadata.

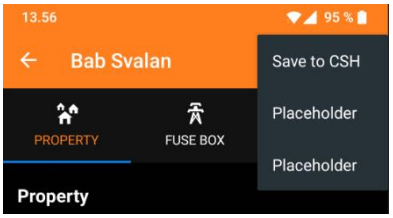
React Native har en egenskap för TextInput-komponenten som heter onEndEditing där det kan specificeras vad som ska hända då användaren slutar skriva i fältet. Alltså varje gång användaren byter till nästa fält eller stänger ner tangentbordet så triggas denna egenskap. Det specificeras att programmet ska kalla på en funktion, som lägger till fältets värde i en array över information som ska laddas upp till hubben. Alla värden som sparas i listan får en viss datastruktur som hubben förväntar sig, bestående av fältets namn, en nyckel, det nya värdet och det gamla värdet. Fältets namn är något som kommer att visas i hubben då kartläggaren ser över de värden som finns lagrade i enheten, medan nyckeln är det som binder värdet till rätt ställe i datastrukturen. Det nya värdet är förstås det som vi är intresserade av och det gamla värdet lämnas tomt då det inte är något som vi behöver ta i beaktande. Ingen uppladdning sker förstås ännu utan datan läggs endast till i listan. Denna lista är alltså en array i kontexten som alla filer har tillgång till så samtliga flikar i applikationen kan lägga till ändringar eller nya värden utan att någon data förloras.

Genom detta tillvägagångssätt kan användaren ändra på värden helt fritt i alla flikar kors och tvärs utan att det skapar problem. Det går även att ändra på värdet i samma fält ett flertal gånger då det nyaste värdet helt enkelt skriver över det gamla. Eftersom uppladdningen inte sker före användaren trycker på ladda upp-knappen så kommer inte varje editering att behöva göra en egen uppladdning till hubben och skapa en massa trafik.

Sedan för bilderna behövs lite mer kontroller för att få allt på rätt ställe. Eftersom applikationen är uppdelad i flera flikar och flikarna har flera sektioner som är helt skilda från varandra, är det återigen en kontext som gäller för att spara alla bilder. Till skillnad från textfälten så behöver bilderna sorteras i mappar så att det senare går att organisera dem tillbaka till rätt sektioner.

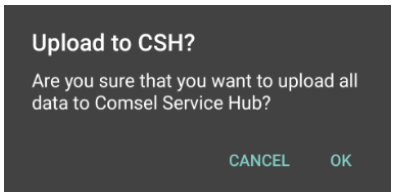
Applikationen har redan listor över de bilder som har blivit tagna i respektive sektion. Under samtliga flikar skapas en effekt som triggas om någon av sektionernas bildlistor ändrar. Alltså så fort användaren lägger till eller tar bort bilder i någon av sektionerna så triggas effekten. Det som effekten ska göra är att lägga till alla filer som finns i de olika sektionernas listor i en ny lista. Alla filer som finns i dessa listor har redan rätt datastruktur sedan tidigare i form av bredd, höjd, mapp, sökväg till filen på den lokala enheten, numrering och innehåll i base64 format. Detta gjordes redan då filen lades till efter att den blivit tagen med kameran eller vald från galleriet. Effekten skapar således en komplett lista över alla bilder som användare har tagit i hela applikationen. Denna lista skrivs till kontexten för att kunna vara tillgänglig över hela applikationen.

Efter att både textinformation och bildinformation är sparad i respektive kontext är allt förberett för att spara till hubben. Sedan för att faktiskt spara finns det en knapp i menyn uppe till höger, som är tillgänglig från samtliga flikar. Då användaren trycker på menyikonen kommer menyn fram som illustreras i Figur 52.



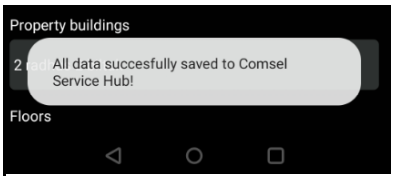
Figur 52: Huvudmeny.

Då användaren trycker på save to CSH kommer en varningsruta fram, för att undvika att användaren ska komma mot knappen i misstag. De övriga alternativen i menyn har för tillfället ingen mission utan är endast där i felsökningssyfte.



Figur 53: Uppladdning-varning.

Sedan när uppladdningen är färdig kommer ett meddelande om att uppladdningen lyckades. Denna komponent är gjord med en Toast som är en Android-komponent. Skulle en iOS version av applikationen i detta skede försöka kompileras skulle det inte fungera då denna komponent inte finns i iOS. I så fall skulle den behöva bytas ut mot en motsvarande React Native-komponent som fungerar på båda operativsystemen. Detta kommer troligtvis att åtgärdas vid en senare uppdatering av applikationen.

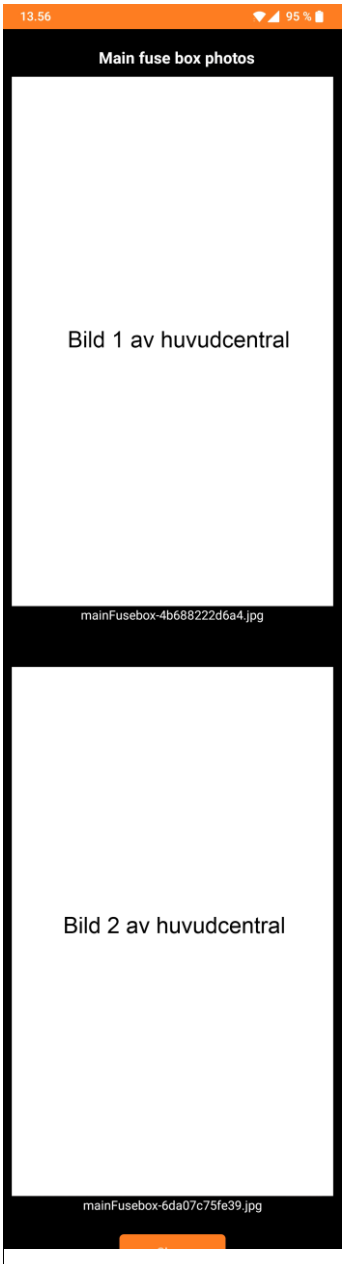


Figur 54: Uppladdning lyckades.

Eftersom applikationen ska användas av Comsels anställda eller utvalda elektriker som har blivit skolade i hur applikationen fungerar så kommer det inte att behövas en lika strikt kontroll på det som skrivs in i fälten.

6.3.8 Ladda från hubben

Varje gång som en enhet väljs ur listan då applikationen startas kommer applikationen att ladda ner den information som hör till enheten från hubben. Som redan tidigare beskrivits fås informationen direkt ur enhetens datastruktur och metadata då enheten laddas in. Första gången som kartläggaren öppnar enheten kommer endast den information som kunden har fyllt i webbformuläret att vara tillgänglig. Då kartläggaren har fyllt i mer information under sitt besök och tagit bilder laddas de sedan upp till hubben. Ifall kartläggaren av någon orsak skulle vara tvungen att göra ett återbesök till fastigheten, behöver även nerladdningen av information inkludera de bilder som nu finns sparade i hubben. Detta kan åstadkommas genom att använda Link funktionen som finns i File-klassen i Comsels bibliotek. Då den körs svarar hubben med en länk till filen så att den kan öppnas genom denna länk i exempelvis webbläsaren. Funktionen kontrollerar även användarens rättigheter för att inte ge ut länkar åt någon som inte har rättigheter till dem. En funktion skapas som går genom alla filer som fås från hubben och kör denna länk funktion på varje fil. Den länk som fås från hubben sparas sedan som en till egenskap för filen och slutligen sorteras alla filer in i rätt sektion av applikationen igen.



Figur 55: Arkiv-exempel.

Layoutmässigt har det valts att lägga in en till ikon bredvid de befintliga kamera- och galleriikonerna i varje sektion. Den nya ikonen är tänkt att representera ett arkiv. Då användaren trycker på den öppnas arkivet som är en Modal-komponent. I Modal-komponenten görs en lista över alla filer som finns för den sektionens array med filer och visar dem i Image-komponenter efter varandra. Komponenten behöver ett innehåll att visa, detta blir således länken som nyss lades till. Sedan under bilden kommer filnamnet som är i formatet mapp, bindestreck och en tredjedel av det UUID som genereras för bilden före den laddas upp till hubben.

För andra sektioner såsom parkering och undercentraler kan det finnas ett flertal underkomponenter som alla kommer att visas i samma arkiv. Här skiljer sig filnamnet från de övriga sektionerna för att lättare kunna se vilka filer som hör ihop med

vilken parkering eller vilken central. Skillnaden är att namnet även inkluderar den numrerade underkomponenten och det namn som underkomponenten har, alltså garage1, biltak1 eller HC2 eller liknande som parkeringar eller centraler kan tänkas döpas till. Ett exempel på ett sådant namn kan vara parking0-garage1-ad5as2d1gd.jpg. Filnamnet börjar med numreringen på parkeringen eller centralen följt av det namn som användaren har get den och slutligen en tredjedel av UUID. Orsaken till denna struktur är dels att bilderna sorteras enligt mapp i hubben dels för att sortera dem i detta arkiv.

I de olika sektionerna dyker arkivikonen upp bredvid de befintliga kamera- och galleriikonerna. Denna ikon kommer endast att vara synlig ifall det finns bilder i arkivet. Det vill säga endast ifall kartläggaren gör ett återbesök och det finns bilder sparade i hubben från förra besöket. Om det inte finns några bilder kommer helt enkelt den befintliga layouten utan arkivikon att användas i stället.

Figur 56: Exempel på arkiv ikon.

7 Resultat

Resultatet blir en fungerande webbsida där kunderna får fylla i ett antal uppgifter angående fastigheten som ska kartläggas. Webbsidan låter kunden även ladda upp bilagor såsom elritningar och situationsplan för att underlätta för kartläggaren inför besöket vid fastigheten. Webbsidan låter även användaren spara ett halvfärdigt formulär för att fortsätta med det vid ett senare tillfälle. Då kunden är nöjd med sina uppgifter kan de skicka in helheten till Comsels datasystem där det lagras i en förbestämd datastruktur. Kunden kan även frivilligt välja att uppge sin Vasa Elektriska kundnummer för att underlätta processen ytterligare då kartläggaren får tillgång till förbrukningshistorik.

Denna historik kan sedan automatiskt importeras och analyseras med hjälp av ett knapptryck i en Excel fil. Filen kan även med ett knapptryck skapa en graf över den datan som importerades. Analysen och graferna som skapas med filen används sedan i den slutliga kartläggningsrapporten över fastigheten.

Med all denna information samlad på ett ställe kan kartläggaren sedan bege sig till fastigheten för att utföra kartläggningen. Då han anländer på plats används kartläggningsapplikationen för att ytterligare samla in information som behövs. Applikationen låter även kartläggaren ta bilder på elcentraler och parkeringsplatser för att smidigt kunna reda ut hur situationen ska hanteras, utan att missa några detaljer.

8 Diskussion

Detta projekt var från början mycket löst definierat eftersom Comsel System som är uppdragsgivaren inte var helt säkra på vad de ville ha eller hur det skulle utföras. Tanken var att digitalisera och automatisera processen att kartlägga en fastighet. Jag fick mycket frihet med hur jag ville strukturera projektet och hur det skulle utföras. Jag hade dessutom innan detta inte alls hört talas om React eller TypeScript. Själva ämnet elbilsladdning hade jag endast skrapat ytan på genom att jag själv skaffat en elbil cirka två månader före jag började på med arbetet.

Jag började på med projektet i november 2021 och blev färdig med en första version av webbsidan och applikationen i februari 2022. Jag kommer att fortsätta jobba vid Comsel system efter att utexaminerats från skolan. Då kommer jag att fortsätta utvecklingen av denna applikation och troligtvis implementera mer funktionalitet. Det är värt att nämna att webbsidan som kommer att vara en direktkontakt för kunderna till Comsel högst sannolikt kommer att få en grafisk omdesign för att passa in med företagets befintliga image. Det kommer att bli intressant att se om de lösningar som jag har försökt skapa kommer att fungera som det är tänkt ute på fältet med alla olika fastigheter och elbehov som finns.

Vad det gäller funktionaliteten för applikationen åt Comsels elbilsladdare är det ännu inte fastslaget vilka funktioner som kommer att ändras, läggas till eller tas bort då både laddaren och applikationen ännu inte har lanserats officiellt på marknaden.

9 Källförteckning

[1] Meta Platforms, Inc, “React,” 2022. [Online]. Available: https://reactjs.org/. [Accessed 18 02 2022].

[2] Pluralsight, ”JavaScript,” 2022. [Online]. Available: https://www.javascript.com/. [Använd 18 02 2022].

[3] Microsoft, ”Typescriptlang,” 2022. [Online]. Available: https://www.typescriptlang.org/. [Använd 18 02 2022].

[4] Material-UI SAS, ”MUI,” 2022. [Online]. Available: https://mui.com. [Använd 18 02 2022].

[5] Meta Platforms, Inc, ”React Native,” 2022. [Online]. Available: https://reactnative.dev/. [Använd 18 02 2022].

[6] Microsoft, ”Getting started with VBA in Office,” Microsoft, 22 01 2022. [Online]. Available: https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office. [Använd 22 02 2022].

[7] Comsel System, ”Oculus,” [Online]. Available: https://comsel.fi/oculus.html. [Använd 20 03 2022].

[8] Steven McFayden, ”Current Capacity Of Cables - An Introduction,” My Electrical Engineering, 31 01 2013. [Online]. Available: https://myelectrical.com/notes/entryid/207/iec-60287-current-capacity-of-cables-an-introduction. [Använd 21 02 2022].

[9] Steven McFayden, ”Current Capacity Of Cables - Rated Current,” My Electrical Engineering, 18 02 2013. [Online]. Available: https://myelectrical.com/notes/entryid/210/iec-60287-current-capacity-of-cables-capacity-equations. [Använd 21 02 2022].

[10] Kiran Daware, ”electricaleasy.com,” [Online]. Available: https://www.electricaleasy.com/2017/03/underground-power-cables.html. [Använd 24 02 2022].

[11] Draka, ”Kabelstickan,” [Online]. Available: http://kabelstickan.draka.se/k/56cc5bcb0c72b80003834d22. [Använd 18 02 2022].

[12] European Union, ”EU direktiv,” 28 10 2014. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014L0094&from=EN. [Använd 14 02 2022].

[13] Virta, ”EV charging plug standards,” 05 12 2017. [Online]. Available: https://www.virta.global/blog/ev-charging-plug-standards. [Använd 18 02 2022].

[14] Virta, ”Electric vehicle charging modes,” 01 06 2021. [Online]. Available: https://www.virta.global/blog/ev-charging-modes. [Använd 18 02 2022].

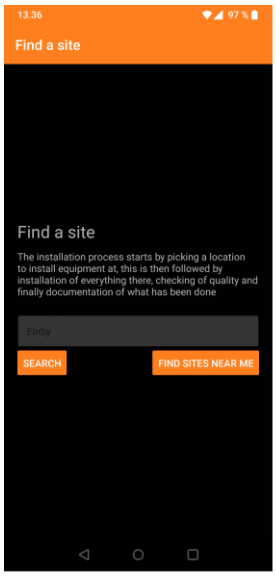
[15] Tesla, ”Tesla supercharging,” 2022. [Online]. Available: https://www.tesla.com/sv_SE/support/SUPERCHARGING#ccs. [Använd 18 02 2022].

[16] M. Ferrell, ”Youtube,” 07 04 2020. [Online]. Available: https://www.youtube.com/watch?v=3FSMG5KbQkM. [Använd 18 02 2022].

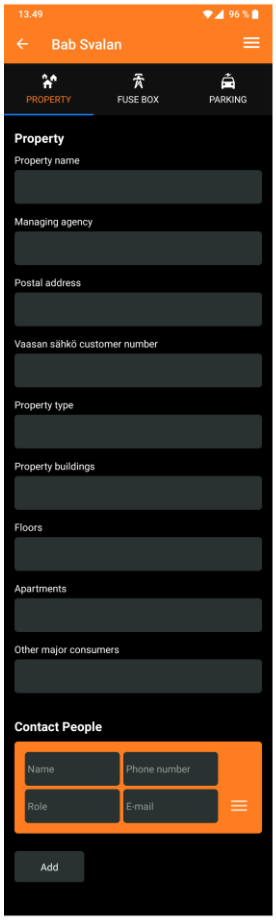
[17] Vattenfall, ”Vattenfall börsel,” 2022. [Online]. Available: https://www.vattenfall.fi/sv/elavtal/borsel/. [Använd 18 02 2022].

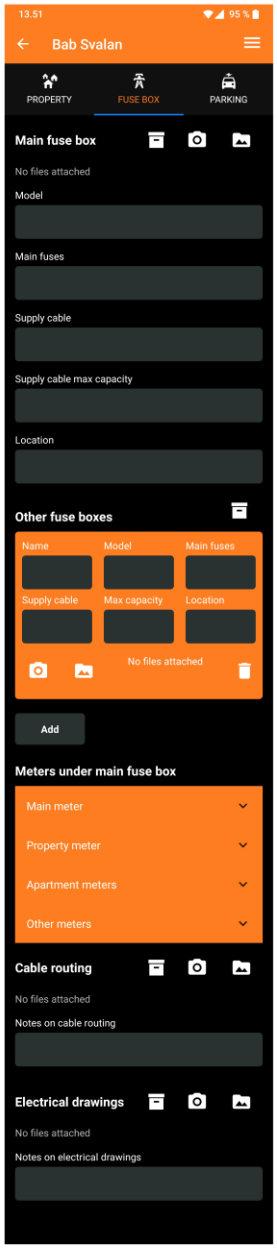
[18] European Union, ”General Data Protection Regulation,” 27 04 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679. [Använd 01 03 2022].

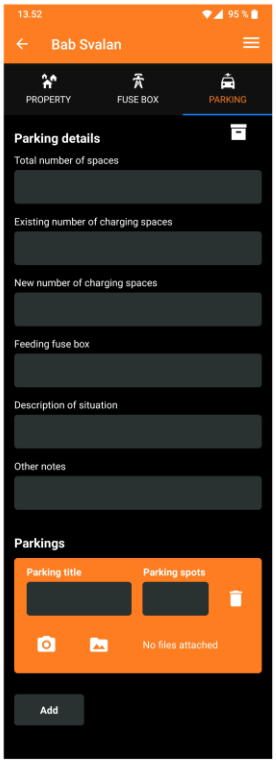
[19] MUI, ”Breakpoints,” [Online]. Available: https://mui.com/customization/breakpoints/#main-content. [Använd 01 03 2022].


13.36
97 %
Find a site
Find a site
The installation process starts by picking a location to install equipment at, this is then followed by installation of everything there, checking of quality and finally documentation of what has been done
Finby
SEARCH
FIND SITES NEAR ME


14.49
78 %
Find a site
Find a site
Finby
SEARCH
FIND SITES NEAR ME
123456 Bab Svalan
väg 12, 65100 vasa
USE THIS SITE
123456 Bab Svalan
Kyrkgatan 2, 65100 Vasa
USE THIS SITE
6575864 Bab Svalan
holmvägen 12, 65100 vasa
USE THIS SITE
123456 Bab Svalan
holmvägen 12, 6500 Vasa
USE THIS SITE
6575864 Bab Svalan
holmvägen 12, 77000 Vasa
Google


13.49
96 %
Bab Svalan
PROPERTY
FUSE BOX
PARKING
Property
Property name
Managing agency
Postal address
Vaasan sähkö customer number
Property type
Property buildings
Floors
Apartments
Other major consumers
Contact People
Name
Phone number
Role
E-mail
Add


13.51
95 %
Bab Svalan
PROPERTY
FUSE BOX
PARKING
Main fuse box
No files attached
Model
Main fuses
Supply cable
Supply cable max capacity
Location
Other fuse boxes
Name
Model
Main fuses
Supply cable
Max capacity
Location
No files attached
Add
Meters under main fuse box
Main meter
Property meter
Apartment meters
Other meters
Cable routing
No files attached
Notes on cable routing
Electrical drawings
No files attached
Notes on electrical drawings


13.52
95 %
Bab Svalan
PROPERTY
FUSE BOX
PARKING
Parking details
Total number of spaces
Existing number of charging spaces
New number of charging spaces
Feeding fuse box
Description of situation
Other notes
Parkings
Parking title
Parking spots
No files attached
Add