Home Energy Management Display

Examensarbete inom högskoleingenjörsprogrammet i Datateknik

Johnny Larsson

Lukas Magnusson

Institutionen för Data- och Informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022
www.chalmers.se

Home Energy management Display

Institutionen för Data- och Informationsteknik

Chalmers Tekniska Högskola / Göteborgs Universitet

412 96 Göteborg

Telefon: 031–772 1000

Tryckt av Chalmers Reproservice

Göteborg, Sverige, 2022

Abstract

For a user that has variable electricity prices, the price varies a lot during the day – there will be a big difference between the lowest and highest prices. The goal of this project is to create an application that helps the user to change their electricity consumption, with better knowledge of electricity prices and past electricity consumption. This report describes the design and implementation to create this application. The result is an easy-to-use prototype that shows the electricity prices for the current day, the next day, and the previous seven days. It also shows the weather and charging status for an electric car – this is however only implemented with mock data and not with live data in the current version due to time constraints. Having easily accessible data for the electricity prices helps with making decisions considering charging an electrical car and other usages of household electricity.

Keyword: React Native, Electricity prices, Web application, Raspberry Pi, Electric cars, Charging of electric cars, Electricity consumption, Minimizing electricity cost.

Sammanfattning

För en användare som har rörliga elpriser varierar priset mycket under dygnet – det blir stor skillnad mellan lägsta och högsta pris. Målet med detta projekt är att skapa en applikation som hjälper användaren att ändra sin elförbrukning, med bättre kunskap om elpriser och tidigare elförbrukning. Rapporten beskriver designen och implementeringen för att skapa denna applikation. Resultatet är en lättanvänd prototyp som visar elpriserna för den aktuella dagen, nästa dag och de föregående sju dagarna. Den visar även vädret och laddningsstatus för en elbil – detta är dock endast implementerat med mock data och inte live data i den nuvarande versionen på grund av tidsbrist. Att ha lättillgänglig data för elpriserna hjälper till med att fatta beslut om att ladda en elbil och annan användning av hushållsel.

Nyckelord: React Native, Elpriser, Webbapplikation, Raspberry Pi, Elbilar, Laddning av elbilar, Elförbrukning, Minimera elkostnad.

Förord

Den här rapporten är skriven av Johnny Larsson och Lukas Magnusson och gjordes som ett examensarbete på 15 högskolepoäng under vårterminen 2022.

Vi vill tacka Johan Sanneblad och Arsam Golriz på Volvo Cars Open Innovation Arena som har varit med oss under hela projektet.

Ett stort tack till Stephen White som har hjälpt oss i React och Kim Niklasson för hans fantastiska UX-design.

Vi vill också tacka Ronja Bromander, Jacob Lundberg och Per Vannas som har hjälpt till med detta projekt.

Slutligen vill vi ge ett stort tack till våran handledare Pelle Evensen för den hjälp vi har fått med denna rapport och bollandet av tankar runt examensjobbet.

Terminologi

Frontend	Det grafiska gränssnitt som användaren ser och integrerar med.
Backend	Bakomliggande mjukvara för applikationen som användaren ej kan se eller integrera med.
MAC-adress	En unik identifierare för varje nätverkskort.
Komponent	Självständig och återanvändbar kod.
API	Ett gränssnitt för kommunikation mellan program, system och applikationer.
QR-kod	Tvådimensionell kod för optisk avläsning.
Raspberry Pi	Liten enkortsdator byggd på ett kretskort.
Queries	Frågor till databas för att hämta data.
Backlog	En prioriterad och ordnad lista för vad som är planerat att utföra.
Scrumboard	Ett visuellt verktyg för att hantera och planera projekt.

Innehåll

1

Inledning

I och med att elförbrukningen i Sverige varierar under dagen så varierar även elpriserna. Detta kan ha stor påverkan för de konsumenter som har rörligt elpris. Det är till exempel mycket billigare att ladda sin elbil på natten då elkonsumtionen är lägre än på dagen. Inom Volvo Cars finns avdelningen Open Innovation Arena (OIA) som jobbar med att ta fram nya koncept. För att göra informationen om aktuellt elpris och elförbrukning mer lättillgänglig vill Volvo Cars OIA utveckla en applikation som kan visas på en display i hemmet. Denna display ska visa grafer och information om elförbrukning och det aktuella elpriset samt sin elbils batterinivå så att man kan planera och ta bättre beslut om sin elkonsumtion så att man kan välja att ladda bilen då elpriset är som lägst.

1.1 Syfte

Syftet med projektet är att utveckla en applikation som underlättar vardagen för Volvos elbilskunder. Detta med hjälp av en applikation på en digital skärm som elbilsägare kan använda för att visa information om elförbrukning, aktuellt elpris, batterinivå på elbilen och för att styra laddningen av sin elbil. Applikationens syfte är att ge användaren möjlighet att påverka sin elkonsumtion med en bättre insikt runt energiförbrukning och kostnader.

1.2 Mål

Målen för applikationen är följande:

- Använda data från Volvo Cars backend
- Visa nuvarande elpris och dagens elpris i en graf.
- Visa elbilens batteriprocent och den planerade laddningen i en graf.
- Laddning av elbilen ska kunna styras av applikationen.
- Programvaran är till en början riktad mot en Raspberry Pi 4B med en sju tums skärm men ska i framtiden även kunna köras som en applikation på Android / iOS och fungera på webbläsaren.
- Följa en minimalistisk design.

1.3 Avgränsningar

Fokuset i projektet sker endast i frontend där backend kommer skötas av Volvo. Det finns redan färdiga React Native-komponenter från Volvo som kommer att återanvändas. Applikationen kommer enbart visas i landskapsläge.

2

Teknisk bakgrund

Detta kapitel beskriver de verktyg, programmeringsspråk och bibliotek som har använts under projektet. Vi använde de verktyg Volvo Cars hade valt ut innan projektet startade, utöver React Native Carousel som vi valde för navigationssystemet. Detta valdes på grund av det erbjöd smidiga animationer och fungerade på de plattformar vi ville stödja.

2.1 React

React är ett öppenkällkod(FOSS) JavaScriptbibliotek för att utveckla interaktivt användargränssnitt inom webbutveckling. React är komponentbaserat där varje komponent ska fungera i isolation för att enkelt kunna lägga till nya funktioner i form av nya komponenter. Dessa komponenter kan sedan kombineras för att skapa flexibla och komplexa användargränssnitt. Den komponentbaserade arkitekturen kan även öka hastigheten av applikationen eftersom react bara uppdaterar de komponenter som fått en förändring [1].

2.2 React Native

React Native är ett öppenkällkod(FOSS) JavaScriptbibliotek som bygger på en variant av React för att utveckla applikationer till både iOS och Android. React Native gör det möjligt att använda samma kodbas till flera olika plattformar [2].

2.3 Expo

Expo är ett ramverk för att underlätta utvecklingen av React Native-applikationer. Det hjälper till att bygga applikationer till iOS, Android och även webbapplikationer från samma React Native-kodbas. Expo tillåter en snabbare utvecklingsprocess där man kan se uppdateringar av appen reflekteras i realtid på sin telefon eller webbläsare. Expo underlättar även arbetet med att skicka ut applikationer till olika appbutiker och tillåter uppdateringar att ske i realtid utan att användaren behöver uppdatera applikationen [3].

2.4 GraphQL

GraphQL är ett queryspråk till API:er där applikationen har full kontroll över vilka data som hämtas från servern. Applikationen specificerar vilka variabler den vill få från servern och även strukturen för datarepresentationen. Tillbaka får applikationen just den data som frågades om och inget mer, detta gör att hämtningen av data kan bli mindre då datamängden kan anpassas efter applikationens behov vilket även gör datahämtningen snabbare [4].

2.5 Apollo

Apollo Client är ett bibliotek för JavaScript som förenklar användningen av GraphQL med React(Native). Apollo hjälper även till med att cacha och lagra data så att upprepade likadana queries använder den redan cachade datan istället för att göra nya förfrågningar till servern [5].

2.6 Victory

Victory är ett modulärt diagrambibliotek för React och React Native. Det använder samma API för både web och React Native-applikationer, vilket underlättar för utveckla för flera plattformar. Victory är flexibelt med flera olika sorters diagram för att visualisera data med ytdiagram, stapeldiagram, tårtdiagram, linjediagram och andra sorters diagram. Det går att skapa egna komponenter att använda i Victory och de flesta delar i ett diagram går att anpassa. Diagrammen kan anpassas till att vara interaktiva med inforutor och händelser som sker vid interaktion [6].

2.7 Jotai

Jotai är ett React-tillståndshanteringsverktyg för globala variabler som Jotai kallar atomer. Man skapar en konfiguration med atomer som representerar en del av ett tillstånd för variabler med ett initialt värde. Sedan kan man skapa atomer med läs och eller skrivbara baserat på konfigurationen. Jotai är minimalistiskt och liknar Reacts useState, men är ej bundet till en specifik komponent som useState är [7].

2.8 React Native Carousel

React Native Carousel är ett React Native-bibliotek som används huvudsakligen för att skapa bildspel där man med svajp-rörelser kan byta till nästa eller föregående bild. Man är inte bunden till att använda bilder, utan det går använda valfri React Native-komponent och kan då t.ex. använda det som grund till ett navigationssystem för att ta sig mellan olika skärmar [8].

2.9 Figma

Figma är ett molnbaserat designverktyg för design och prototyper. Figma fungerar på alla plattformar som har en webbläsare och är ett verktyg där UX-designers kan dela sin design och prototyp med utvecklare. Man kan se CSS-kod för designen och interaktion mellan objekt och skärmar. Flera personer kan arbeta med samma design i realtid [9].

3

Metod

Detta kapitel beskriver först det förberedande arbete som har skett i projektet, sedan det agila arbetssätt som har använt och till sist den testning och utvärdering av applikationen som har skett under projektet.

3.1 Förberedande arbete

Innan utvecklingen av applikationen påbörjas, utförs ett förberedande arbete. Det förberedande arbetet innehåller följande moment:

- **Planering**: En planeringsrapport skapas för att få projektets ramar och mål nedskrivna. I planeringsrapporten skapas det ett Gant-schema för tidsplanering av projektet vilket kan ses i appendix A.
- **Val av operativsystem**: Val av operativsystem som ska köras på Raspberry Pi:n bestämdes genom att testa och forska bland de olika alternativ som finns, där prestanda och användarvänlighet var de två största faktorerna.
- **Val av programmeringsspråk och relevanta bibliotek**: Det behövdes även bestämmas om vi skulle använda React eller React Native i utvecklingen av applikationen. Den avgörande faktorn är ifall applikationen endast är menad som en webbsida eller också är tänkt att fungera som applikation på Android och iOS. Relevanta bibliotek kunde tas fram genom att fråga företaget om hur tidigare projekt är uppbyggda. Det underlättade även att vi redan hade fått designen gjort av en designer på Volvo och därefter lättare se vad som skulle behövas för att uppnå detta.

3.2 Agilt arbetssätt

Utveckling av applikationen sker med ett agilt arbetssätt. Kortare avstämningsmöten sker varje dag där de i projektet beskriver vad de arbetade med dagen innan, vad de ska göra idag och om de har några problem. Längre möten sker veckovis där gruppen går igenom en backlog och vad som ska prioriteras under veckan. Ungefär varannan vecka sker en demo för ett annat större team på Volvo Cars OIA för återkoppling. Arbetet planeras med hjälp av digital Scrumboard på planeringsverktyget Trello. Programmering sker både individuellt och parvis.

3.3 Testning av system

Under utvecklingen av applikationen testades den nya funktionalitet som lagts till under demo för ett större team på Volvo. Där kunde fel i funktionalitet eller design hittas och rättas till. Applikationen finns även som hemsida där alla i Volvo teamet kan se den nyaste versionen av applikationen. Detta användes även för att kunna köra applikationen på en Raspberry Pi i kontoret på Volvo Cars.

3.4 Utvärdering av användarvänlighet

Applikationen följer en minimalistisk designprincip. De riktlinjer som följdes under utvecklingen är följande:

- **Enkel överblick över informationen och visa endast det nödvändigaste**
  All den nödvändigaste informationen ska synas på skärmen utan att en användare ska behöva klicka runt i olika menyer. Detta görs för att applikationen är menad att köra på en dedikerad display och inte ska kräva mer interaktion än nödvändigt.
- **Konsistent design**
  Temat på applikationen ska vara konsistent på alla de olika skärmarna, detta innefattar bland annat att det ska vara samma färger, marginaler och fontstorlek på liknande komponenter som delas mellan skärmarna för att hålla applikationen enformig och lättlärd [10].

4

Design

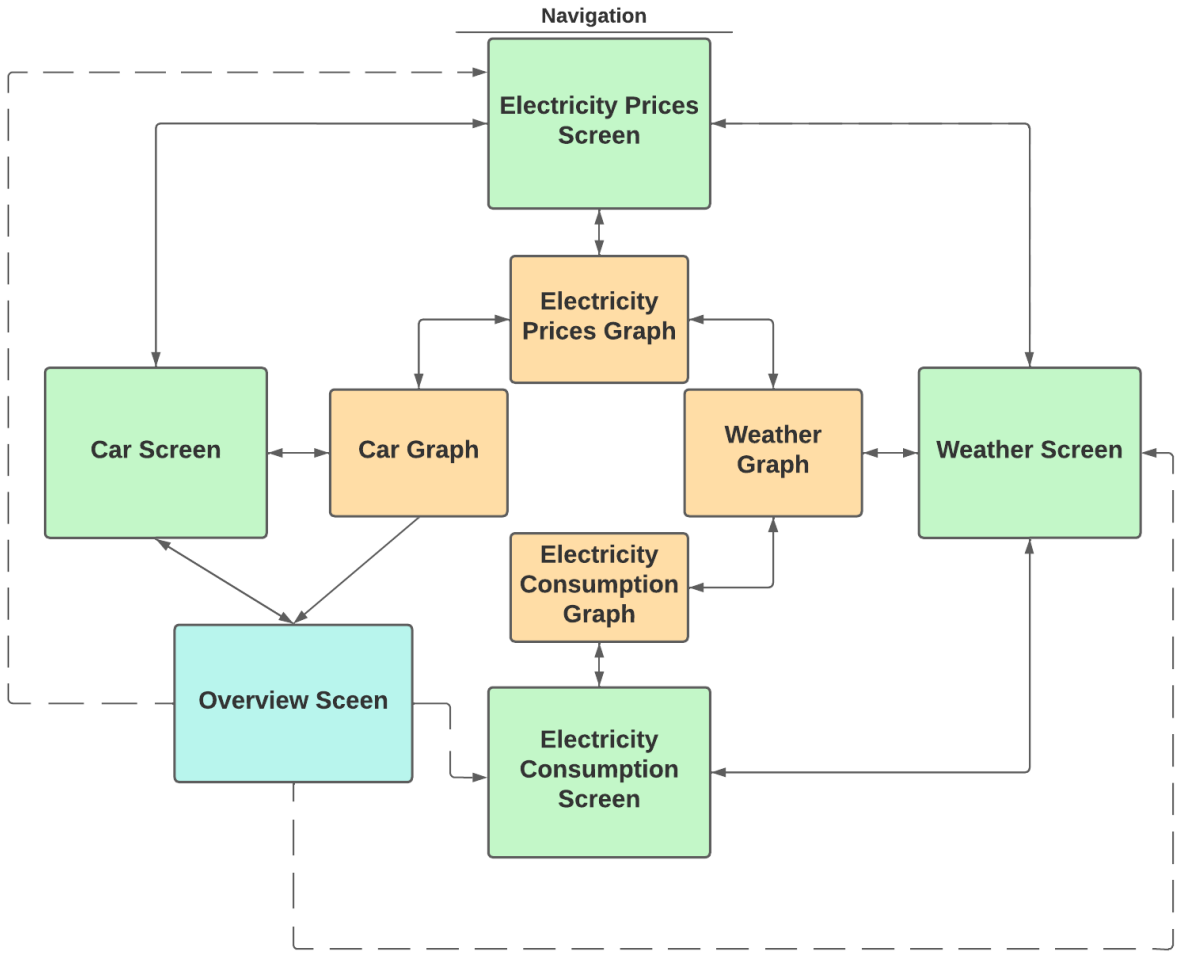
Detta kapitel beskriver först den övergripande designen och navigationen för applikationen. Sedan kommer de olika informationsskärmarna och dess tillhörande grafskärmar. Sist visas inställningsskärmar och en insiktsskärm.

4.1 Övergripande design

Under projektet har det följts en designmall gjord av en UX-designer genom Volvo. Denna mall kan ses i Figur 4.1. Applikationen är uppdelad i flera olika skärmar utlagda i ett rutnät där det ska vara möjligt att använda touch svajpgester för att navigera mellan de olika skärmarna. Högst upp i Figur 4.1 är en överblicksskärm där användaren kan få en överblick över de fyra olika områderna: Elbilsskärm, Elprisskärm, Väderskärm och Elkonsumtionskärm. De olika områderna har varsin grafskärm för att visa mer information.

Figur 4.1. Överblick över alla huvudskärmar.

Förtydligande över navigationen kan ses i Figur 4.2 där pilarna visar de möjliga övergångar som finns för att ta sig mellan de olika skärmarna. Användaren sveper horisontellt för att stanna kvar på en nivå t.ex. ta sig från *Car Screen* till *Electricity Prices Screen* och vertikal svajp för att ta sig mellan de yttre skärmarna och grafskärmarna. Från *Overview Screen* finns det knappar för att ta sig direkt till någon av de yttre skärmarna och på alla skärmar finns det en hemknapp för att ta sig tillbaka till *Overview Screen*.

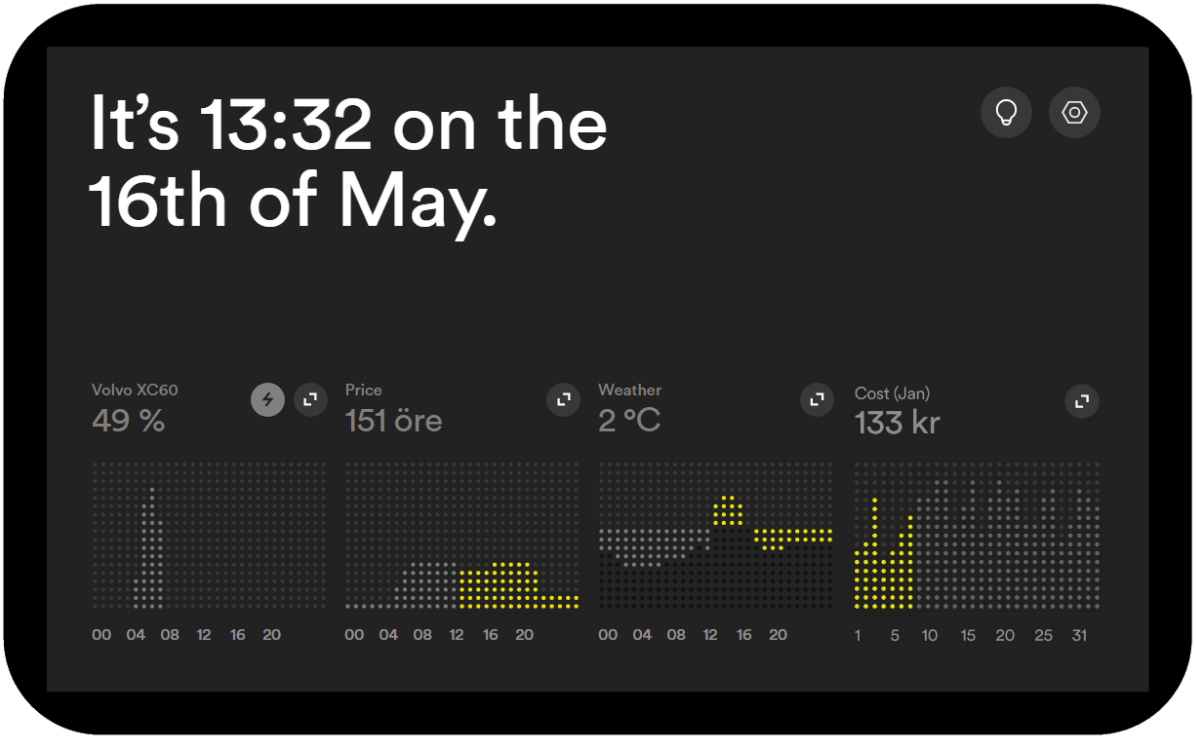


Figur 4.2. Överblick över navigationssystemet. De streckade linjerna indikerar övergångarna från *Overview screen*. De fyllda linjerna indikerar övergångar där man tar sig från närliggande skärm genom att svajpa.

För att underlätta implementationen har designen en del restriktioner. Designen utgår ifrån att applikationen alltid kommer att visas upp i ett landskapsläge på en skärm med ungefär lika höjd och bredd som basdesignen. Applikationen följer en minimalistisk designprincip, där endast den mest nödvändiga informationen visas. Alla skärmar följer samma tema och med samma marginaler på alla skärmar för att ha en konsekvent design [10].

Applikationens temafärg är också konfigurerbar efter användarens preferens. I samtliga bilder är den valda temafärgen gul.

4.2 Överblicksskärm



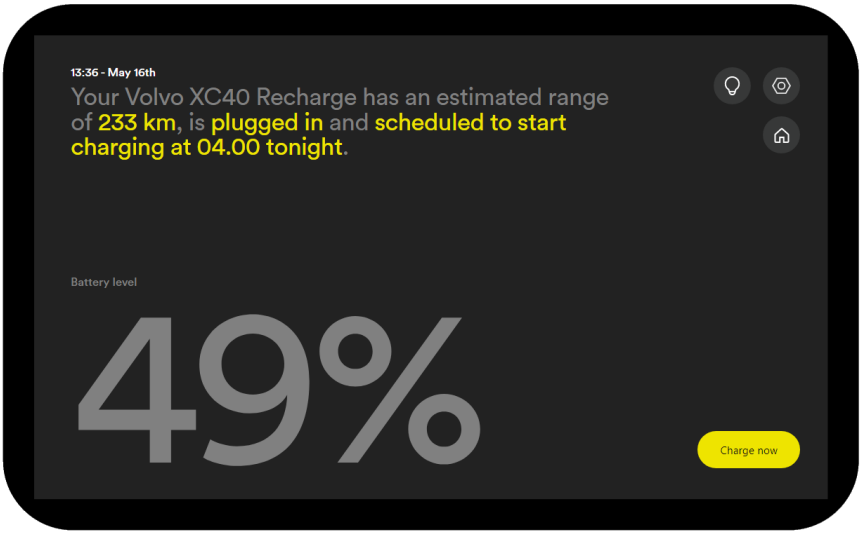
Figur 4.3 Överblicksskärm som visar mindre versioner av de fyra områdena, Elbilsskärm, Elprisskärm, Väderskärm och Elkonsumtionskärm.

Figur 4.3 visar en överblicksskärm (*Overview Screen*) där man kan se information och grafer för de fyra områdena. Varje område har en klickbar yta som tar användaren direkt till skärmen för det området, där man kan få mer information. Längst upp till höger finns det en knapp för insikter och inställningar. Klickar användaren på knappen för insikter kommer användare till en skärm med information med olika insikter om användarens elförbrukning. Klickar användaren på inställningar-knappen kommer en inställnings-modul fram. Båda dessa knappar är statiska och förekommer på alla skärmar.

4.3 Informationsskärmar

Denna sektion visar de fyra informationsskärmar som applikationen har: Elbilsskärm (*Car Screen*), Elprisskärm (*Electricity Prices Screen*), Väderskärm (*Weather Screen*) och Elkonsumtionskärm (*Electricity Consumption Screen*). Det är även dessa skärmar som finns som miniatyrer i Overview Screen där man kan klicka på dem för att ta sig direkt till vald skärm.

4.3.1 Elbilsskärm



Figur 4.4 Elbilsskärm som visar information om elbilens laddning.

Elbilsskärmen visar information om elbilen vilket kan ses i Figur 4.4. Där kan användaren se laddningsprocent, om elbilen laddar eller ej, planerad laddningstid samt en förväntad räckvidd. Man kan även starta laddningen av elbilen från denna skärm. På alla skärmar förutom översiktsskärmen finns det en knapp med en husikon för att navigera direkt till översiktsskärmen.

4.3.2 Elprisskärm

Figur 4.5 Elprisskärm som visar information om dagens elpris.

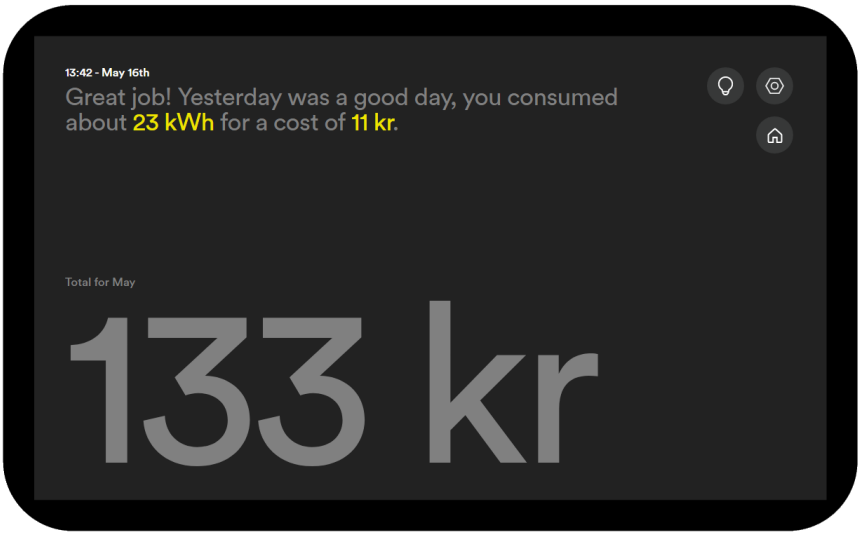
Figur 4.5 visar lägsta och högsta elpriset för dagen och nuvarande elpris för den aktuella timmen. Skärmen beskriver även om priset för dagen är lågt, normalt eller högt. Om sex timmar under en dag har ett pris som är lägre än en standardavvikelse för de senaste 30 dagarna beskrivs priset som lågt, är sex timmar högre än en standardavvikelse beskrivs det som högt, annars som normalt.

4.3.3 Väderskärm

Figur 4.6 Väderskärm som visar information om dagens väder.

Väderskärmen visar information om vädret vilket kan ses i Figur 4.6. Där kan man se aktuell temperatur för nuvarande timmen och en beskrivning av vädret, samt maximala vindhastigheten för dagen.

4.3.4 Elkonsumtionskärm



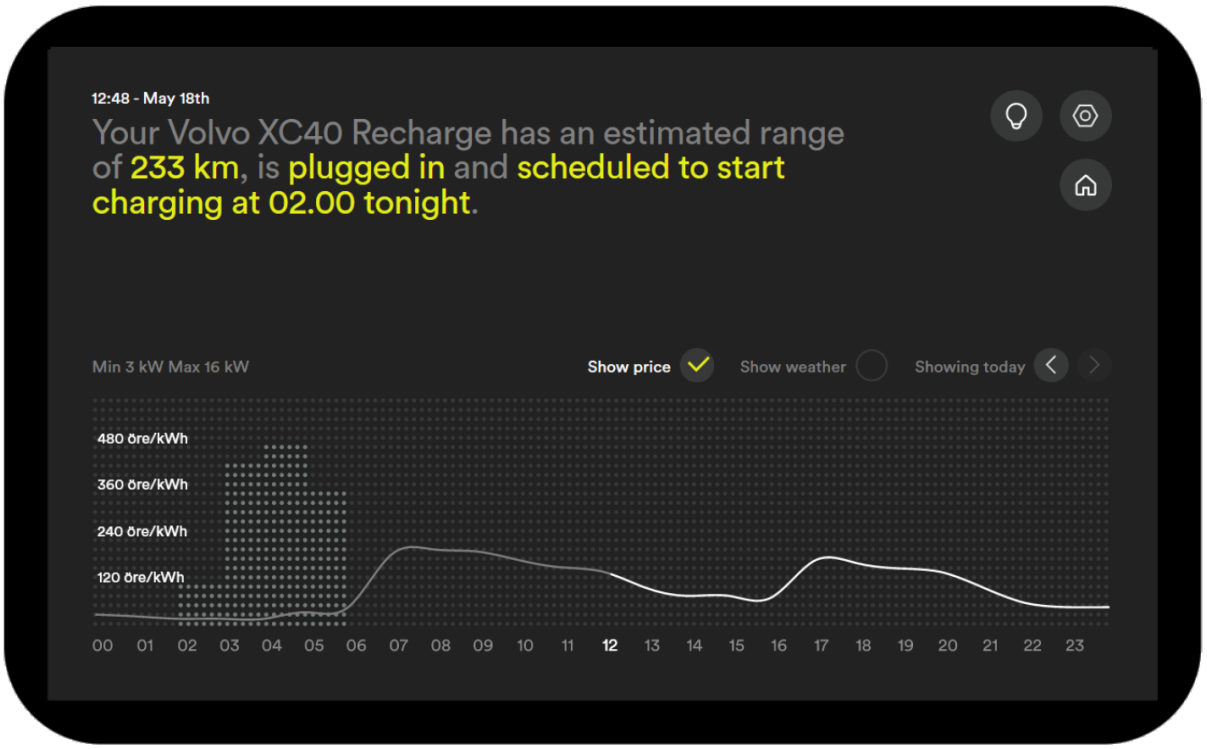
Figur 4.7. Skärm som visar information om hemmets elförbrukning.

Figur 4.7 visar den totala kostnaden för hemmet under den aktuella månaden, gårdagens elkonsumtion i kWh och kronor. Skärmen beskriver även gårdagens elkonsumtion.

4.4 Grafskärmar

Denna sektion visar de tre grafskärmar som implementerades, samt en skärm där interaktion med grafen visas. Dessa tre grafskärmarna är för: Elbilenskärm, Elprisskärm, Väderskärm. Det var även meningen att Elkonsumtionskärmen skulle ha en grafskärm men den implementerades aldrig.

4.4.1 Grafskärm för elbilen



Figur 4.8 Skärm som visar en graf och information om laddning för elbilen.

Grafskärmen visar information om användarens elbil vilket kan ses i Figur 4.8. Där kan användaren se förväntad räckvidd, om bilen laddar eller ej och planerad start av laddning. Hur mycket bilen laddas och vilka timmar visas med små fyllda cirklar, där cirklarna är i ljusgrått om det är före den aktuella timmen och i temafärgen efter den aktuella timmen. Siffran för nuvarande klockslag visas i vitt.

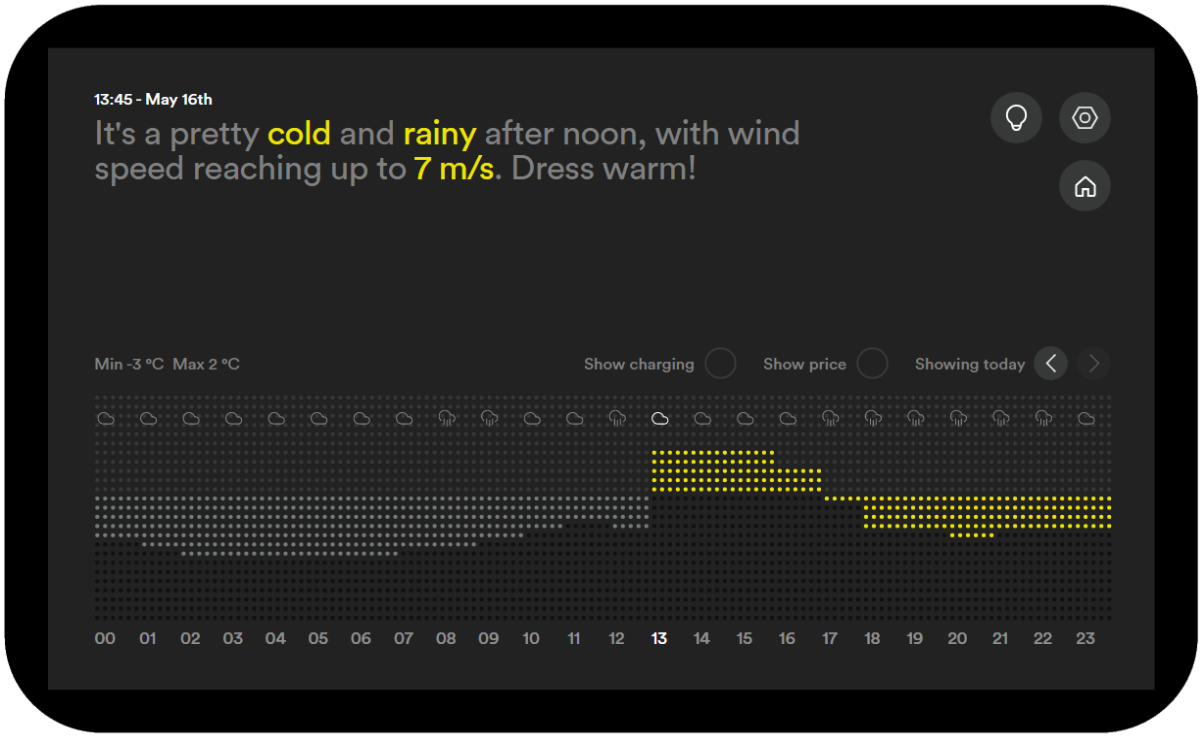
Genom att klicka i en radioknapp för *Show price*, så visas en linjegraf med dagens elpris så att det går att se den planerade laddningen av elbilen tydligare i jämförhet med elpriset. Det går även att visa vädret vilket fungerar på samma vis.

4.4.2 Grafskärm för elpriset

Figur 4.9. Skärm som visar information och en graf om elpriser.

Figur 4.9 visar information om elpriset. Lägsta och högsta elpriset för dagen visas och beskriver om priset för dagen är lågt, normalt eller högt. Grafen är i en mörkare färg innan den aktuella timmen och i temafärgen under och efter nuvarande klockslag för den aktuella dagen. Efter kl. 13 när det har kommit in prognos för morgondagens elpriser kan man byta till nästa dag. Man kan även gå bakåt upp till sju dagar för att se tidigare elpriser. Tidigare dagar är alltid i en mörkare färg medan morgondagens graf alltid är i temafärgen. Är man sju dagar bakåt vilket är det maximala det finns data för, så är bakåtknappen inaktiverad och i en mörkare färg. Samma gäller för framåtknappen om klockan är innan kl. 13 under nuvarande dag, eller om man visar nästkommande dag.

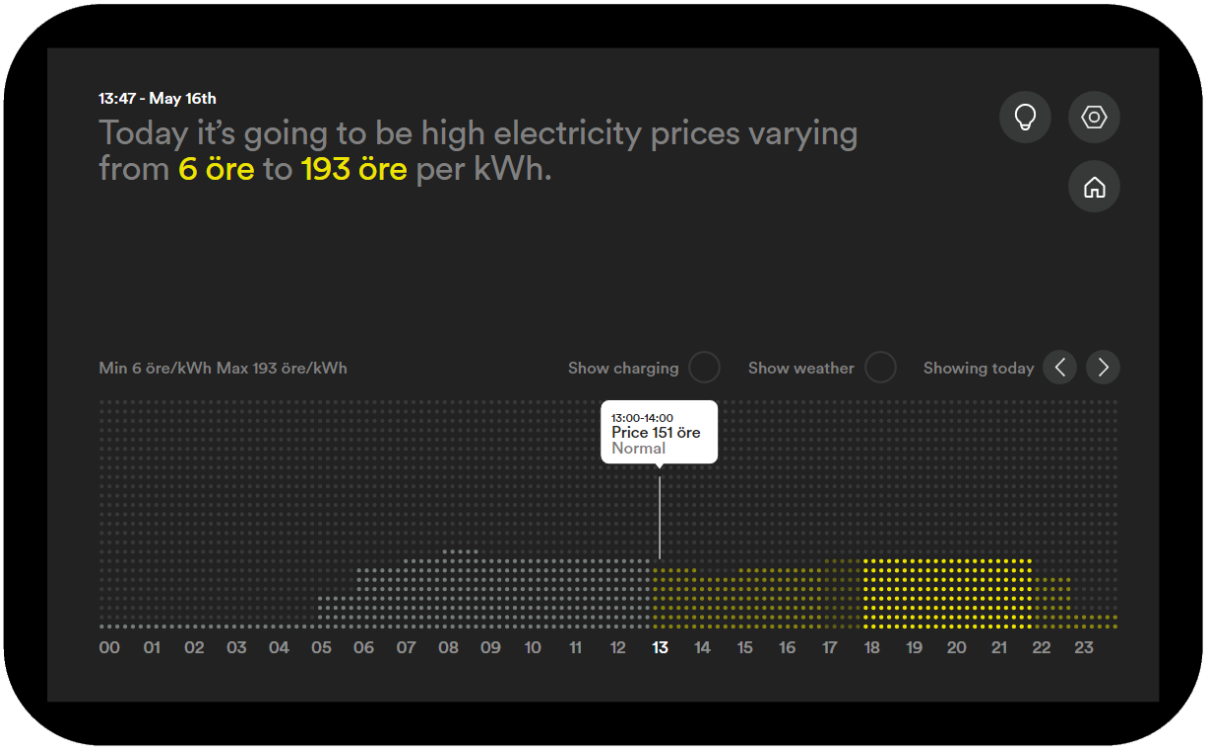
4.4.3 Grafskärm för vädret



Figur 4.10. Skärm som visar information och en graf om vädret.

Vädergrafen, vilket kan ses i Figur 4.10 fungerar utseendemässigt på liknande sätt som elprisgrafen, men skiljer sig på så sätt att den även visar ikoner högst upp som indikerar väderinformation som t.ex. om det är soligt, regnigt eller åskar. Grafens y-axel indikerar temperaturen. Cirklarna i bakgrunden till grafen färgas ljusgrå vid plusgrader och mörkgrå vid minusgrader. Grafen utgår sedan från denna nollpunkt och negativa värden går nedåt och positiva uppåt.

4.4.4 Elprisgraf interaktion



Figur 4.11. Skärm som visar interaktion med graf för elpris.

Skärmen visar vad som sker när man trycker på grafen vilket kan ses i Figur 4.11. En ruta kommer upp för det klockslag man tryckte på och visar mer information. Man får se priset för den timmen och om det anses vara lågt, normalt eller högt. Färgen på inforutan och färgen på grafen ändras också baserat på priset.

4.5 Inställningar

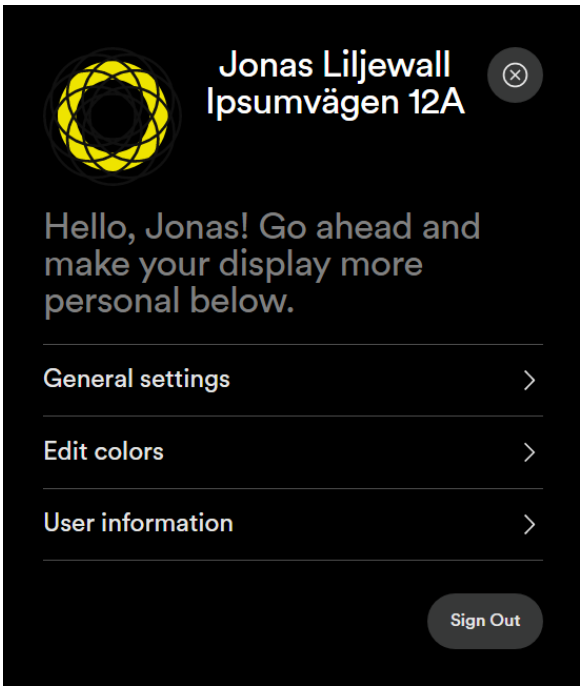
Här visas de olika skärmarna för inställningsmodulen.

4.5.1 Inställningsmodul

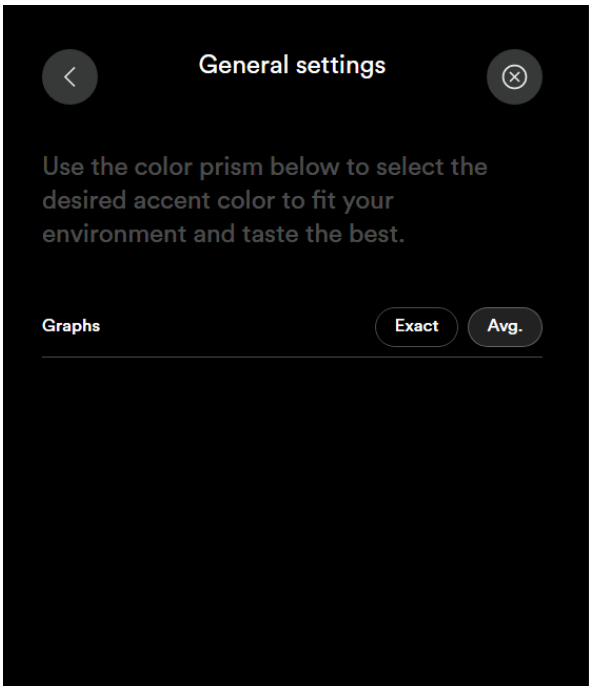
Figur 4.12 visar en inställningsmodul där man kan logga ut eller öppna någon av tre olika inställningsområden. Modulen går att stänga genom att klicka på krysset eller utanför modulen.

4.5.2 Generella inställningar

I generella inställningar som man kan se i figur 4.13 kan man ställa in om graferna ska målas upp exakt efter de data de har, eller om de ska avrundas efter närliggande värden för att få en mer jämn graf.



Figur 4.12. Inställningsmodul.



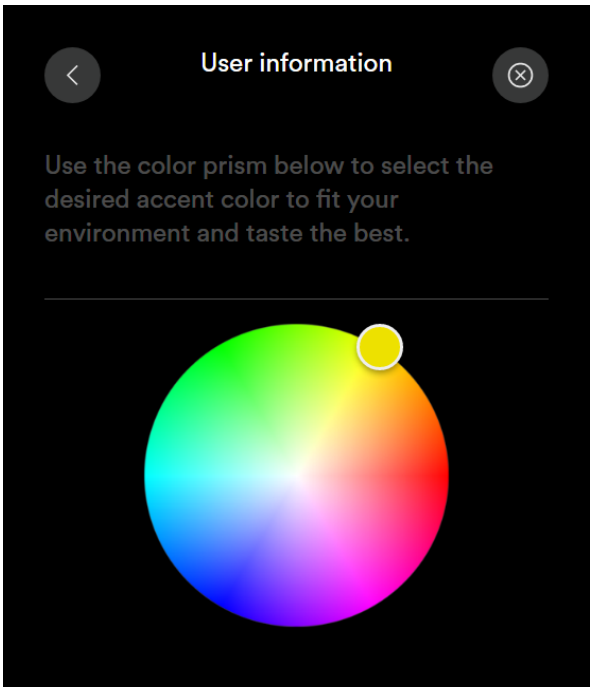
Figur 4.13. Generella inställningar.

4.5.3 Ändring av temafärg

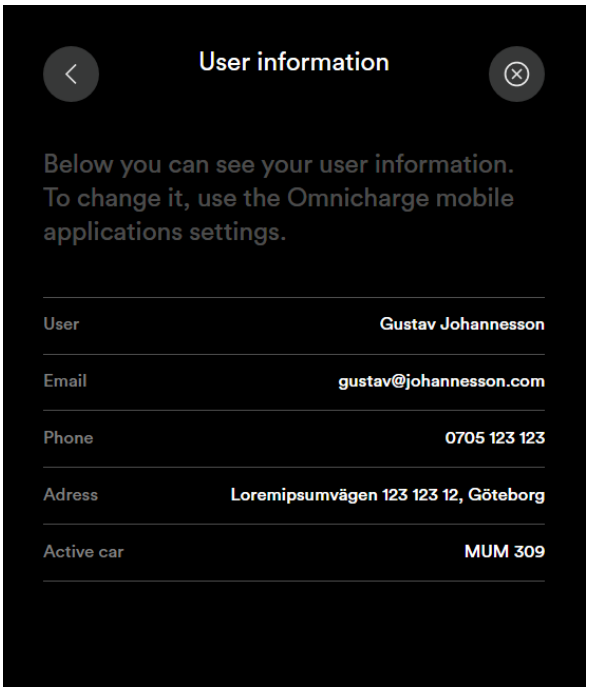
Temafärgen går att ändra i inställningar som visas Figur 4.14. Detta påverkar färger på grafer, laddningsknappen, insiktsskärmen samt en del av texten som är extra viktig att framhäva. Standardfärgen som är aktiv i dessa bilder är gul.

4.5.4 Användarinformation

Användarinformation som man kan se i Figur 4.15 visar information om användaren.

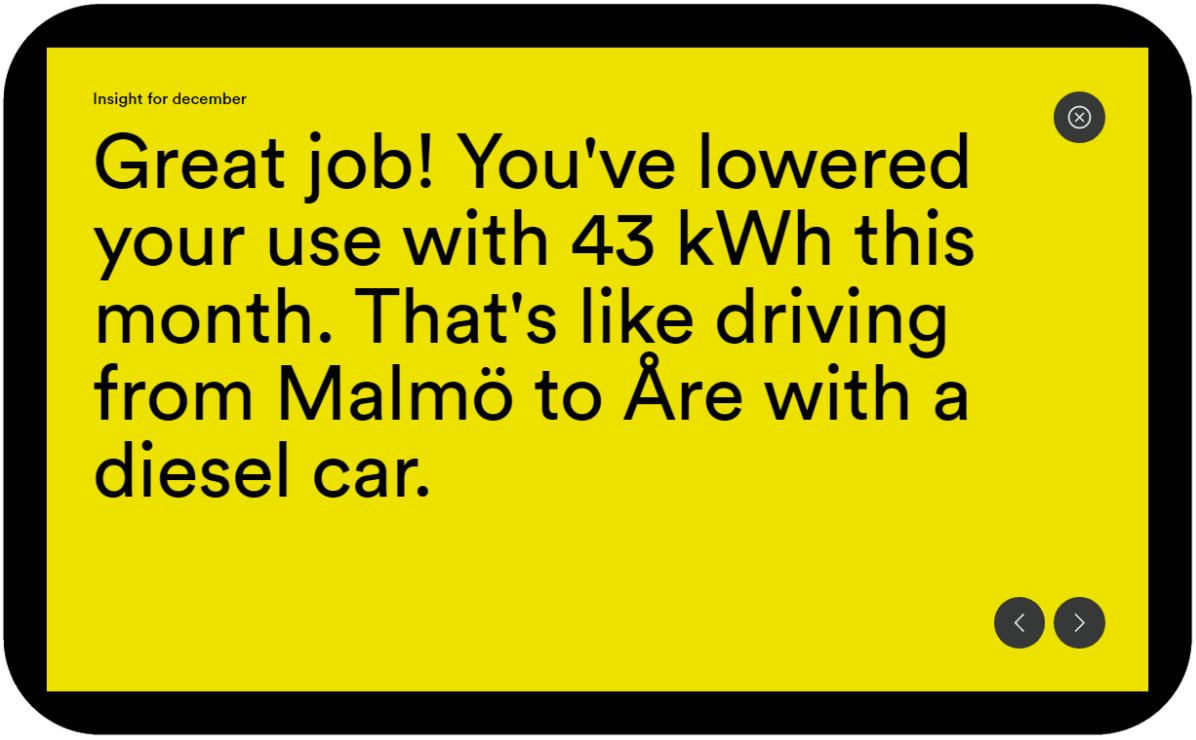


Figur 4.14. Inställning för att ändra temafärg.



Figur 4.15. Användarinformation.

4.6 Insiktsskärm



Figur 4.16. Skärm som visar insikter för användaren.

Insiktsskärmen som kan ses i Figur 4.16 visar insikter om användarens elförbrukning.

5

Analys

I detta kapitel beskrivs den analys som har skett för val av operativsystem samt programmeringsspråk för applikationen som utvecklades.

5.1 Val av operativsystem

Ett antal olika operativsystem testades på Raspberry Pi:n och vi la upp för- och nackdelar med de olika systemen. De viktigaste kraven på operativsystem för applikationen var följande:

- Prestanda
- Användarvänlighet
- Stöd för hårdvaruacceleration

Prestanda och hårdvaruacceleration var ett krav eftersom graferna i applikationen är grafiskt krävande och i framtiden ska applikationen kunna hantera fler animationer.

Vi testade både Android och Windows 10 men vi hade problem med att få det att fungera då det inte fanns någon officiell lösning för det. Tillslut bestämdes det för det officiella Raspberry Pi operativsystemet: Raspberry Pi OS. Raspberry Pi OS bygger på Debian och är optimerad för hårdvaran för en Raspberry Pi samt tillåter den hårdvaruacceleration vilket ger hög prestanda. Det är även smidigt att använda då det kommer med en lättanvändlig skrivbordsmiljö [11].

5.2 Val av programmeringsspråk

I början av projektet var det planerat av Volvo Cars att React skulle användas som programmeringsspråk där applikationen körs som en webbapplikation på Raspberry Pi. Men då applikationen utvecklades för en touchskärm så övervägdes även React Native som ett alternativ. React är större och har fler bibliotek än React Native. Applikationer i React Native går att porta över till iOS och Android med samma kodbas, med några mindre förändringar. React Native valdes därför baserat på att det fanns tankar om att även släppa denna applikation på iPads senare [2].

6

Implementation

Detta kapitel beskriver implementationen av olika områden för applikationen. Först beskrivs navigationen, sen inställningar och grafer och till sist hämtning av data från backend.

6.1 Navigation

Navigationen använder sig av biblioteket React Native Reanimated Carousel som ger stöd för svajprörelser med hänsyn till svajphastighet för att byta skärm och funkar på både webb och som applikation. Det är möjligt att både svajpa vertikalt och horisontellt för att ta sig till närliggande skärmar i ett rutnätsmönster. Det implementerades även funktionalitet för att ta sig till olika skärmar genom att trycka på knappar för att göra navigationen smidigare.

6.2 Inställningar

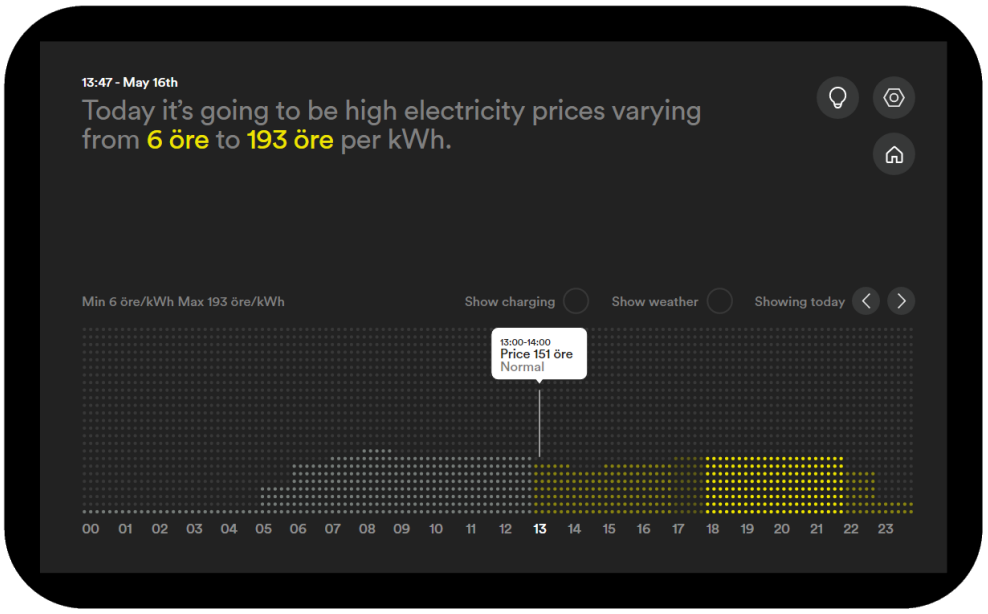
För att skapa en meny för inställningar har biblioteket React Native Modal använts [12]. Detta har implementerats så att ett knapptryck gör så att en modul kommer fram från vänster sida med flera undersidor. Där kan man ändra i temafärgen för applikationen vilket ändrar en grundfärg som används på alla sidor. Det finns även möjlighet för att ändra hur grafen målas upp, se användarinformation och logga ut. Modulen går att stänga med ett knapptryck men också genom att höger svajpa på modulen.

6.3 Grafer

För att skapa skräddarsydda och interaktiva grafer användes grafbiblioteket Victory. För att få en skräddarsydd design där grafen är uppbyggd av små fyllda cirklar skapades en egen komponent. För att enklare överblick så är cirklarna i grafen i en mörkare färg före den aktuella timmen och i temafärgen under och efter den aktuella timmen. När man klickar på grafen kommer en inforuta upp som visar information för den aktuella timmen.

Den planerade designen, vilken kan ses i Figur 6.1, har inforutan ovanför grafen och framför radioknapparna, men radioknapparna gick ej att interagera med när vi implementerade denna lösning. Istället fick inforutan placeras i översta delen av grafen som man kan se i Figur 6.2 och skala om y-axeln, så att grafen sällan är i högsta delen av dess utrymme.

Figur 6.1. Figma-prototyp som visar interaktion med elprisgraf.



Figur 6.2. Slutgiltig version av interaktion med elprisgraf.

6.4 Hämtning av data från backend

För att hämta data för elpriset och vädret från backend användes GraphQL med hjälp av Apollo. Med GraphQL specificerar man de fält man vill hämta från backend och får tillbaka data i samma struktur som man frågade efter. För att underlätta hämtningen av data användes även Apollo. Apollo lagrar responsen från backend lokalt så fler av samma förfrågningar kan använda lokal data istället för att skicka onödiga förfrågningar till servern.

```
fragment SpotPriceField on SpotPriceOutput {
  confirmationStatus
  currency
  deliveryArea
  endTimeUtc
  isPreliminary
  lastUpdatedUtc
  pricePerKwh
  startTimeUtc
  status
}
```

Figur 6.3. Fragment för elprisdata.

I figur 6.3 visas fragmenten för att få data för elpriset i applikationen. En fragment talar om vilka fält man vill få data för.

```
query SpotPrices($deliveryArea: String!, $startTime: DateTime!, $endTime: DateTime) {
  spotPrices(deliveryArea: $deliveryArea, startTime: $startTime, endTime: $endTime) {
    ...SpotPriceField
  }
}
```

Figur 6.4. Query för elprisdata.

I figur 6.4 visar en query för att hämta data för elpriset. Query:n använder fragmenten som skapades ovan men har även extra argument för att tala om vilket elområde data ska hämtas för samt start och sluttid för data.

7

Resultat

Projektet resulterade i en webbapplikation i React Native anpassad för en sju-tums skärm med upplösningen 1024 x 600 för Raspberry Pi. Vi nådde alla våra designmål där applikationen följer en enhetlig design, med samma marginal på alla skärmar. Alla skärmar följer samma tema där användaren kan anpassa hur applikationen ser ut genom att ändra temafärgen. Applikationen är även minimalistisk och visar endast den mest nödvändiga informationen.

Applikationen hämtar data från Volvo Cars backend som sedan visar information och graf för elpriset och vädret. Det är möjligt att se nuvarande elpris och även en graf över dagens elpriser, samt elpriset sju dagar bakåt och även morgondagens elpris när den datan väl finns. Användaren kan interagera med grafen och få en indikation om elpriset är lågt, normalt eller högt. Applikationen visar information och graf för vädrets temperatur med ikoner för varje klockslag som ger en beskrivning av vädrets status för lättare överblick.

Applikation visar även data för laddningen av en Volvo-elbil där man kan se information och graf för den inplanerade laddningstiden. Det går att visa elprisgrafen över grafen för bilens laddning för att underlätta att planera sin laddning till när elpriset är som lägst. En knapp finns för att starta och stoppa laddning av elbilen, vilket visas visuellt men kopplingen till elbilen implementerades ej. All information finns även i mindre miniatyrgrafer på en övergripande skärm där man kan navigera direkt till det området genom att klicka på tillhörande miniatyrgraf.

Data för elbilen samt vädret är inte implementerad med aktuell data då detta ej hann med att implementeras i backend innan projektets slut.

8

Diskussion

I detta kapitel diskuteras först en utvärdering av resultatet baserat på de mål som har funnits för projektet. Efter det diskuteras applikationens påverkan på samhället och framtida utvecklingsmöjligheter.

8.1 Utvärdering av resultat

Applikationen når upp till de mål som fanns för elpriserna där displayen visar upp det aktuella elpriset på ett enkelt och stilrent sätt. Elprisinformationen finns både som värden och i en interaktiv graf. Detta gäller också för elbilens laddning och vädret, men grund av yttre omständigheter så har detta projekt nedprioriteras och det är ej aktuella data som visas. Det visuella ser dock likadant ut, vilket gör att nuvarande version av applikationen fungerar bra som en prototyp. När aktuella data finns från backend, är det enkelt att implementera så att allt fungerar som det var tänkt från början. Applikationen är anpassad för Raspberry Pi med upplösningen 1024 x 600.

8.2 Påverkan på samhället

Denna sektion tar upp samhälleliga, etiska och ekologiska aspekter för projektet.

8.2.1 Samhälleliga aspekter

Genom en större insikt om när elpriserna är lägre och en ökad laddning under dessa tider, så blir det en jämnare belastning på elnätet. Detta gör att elnätet kan användas bättre och ej behöver byggas ut för att klara av toppar med hög belastning.

8.2.2 Etiska aspekter

En etisk aspekt skulle kunna vara insamlingen av data från användaren. Detta betyder att minst en aktör utöver ägaren har tillgång till ägarens data. Dessa data skulle kunna innehålla beteendemönster som t.ex. när personen är hemma och laddar bilen men även information om elkonsumtion.

8.2.3 Ekologiska aspekter

Ökad insikt om elpriser och användning leder förhoppningsvis till en lägre elkonsumtion. En lägre elkonsumtion gör att elnätet ej behövs byggas ut lika mycket och sparar på naturens resurser.

8.3 Framtida utvecklingsmöjligheter

Denna sektion kommer beskriva olika områden som kan utvecklas framöver.

8.3.1 Utveckla till iPad

Nuvarande applikation är enbart anpassad till upplösningen 1024 x 600 och webb-versionen av React Native. Det fanns ett tag planer på att även anpassa den till iPads med varierande upplösningar och iOS versionen av React Native. För att genomföra detta behöver man anpassa teckensnitt, storlekar på olika element och marginaler så att de skalar om baserat på upplösning. Det finns även en del kod som måste skrivas specifikt för iOS och man måste utveckla applikationen på en dator med Mac OS, alternativt testa koden på en iPad.

8.3.2 QR-kod för inloggning

För att logga in så skriver man in sin emailadress och lösenord. Det fanns även tankar ett tag på att implementera en lösning med QR-kod som kopplas till ett konto för en annan Volvo Cars-applikation. När man scannar QR-koden skickas man till en hemsida för att logga in, som inkluderar MAC-adressen för enheten. När man har loggat in så kopplas MAC-adressen till kontot för identifikation.

8.3.3 Elkonsumtion hemmet

Det var planerat att även implementera en skärm och graf med hemmets elkonsumtion. Vi hann aldrig få data för detta så det implementerades ej. Tanken var att man kopplar samman en elmätare för sitt hem till kontot, där man i applikationen kan se elkostnaden och elkonsumtionen. Detta visas som de andra graferna men med dagar istället för timmar och en hel månad istället för en dag.

8.3.4 Vidareutveckling av vädret och elbilen

Data från backend för vädret är i nuläget samma dag som återupprepas då aktuell data ej implementerades. För elbilens laddning finns det ingen integration med backend utan det som visas är enbart testdata. För att få en applikation som visar aktuell information behöver det implementeras korrekt data från backend och sedan integrera detta med frontend.

9

Slutsats

Detta projekt har resulterat i en prototypapplikation som ger användaren information om elpriser, laddning av en elbil och vädret. Designen är minimalistisk och det är lätt för användaren att få en överblick om elpriset och styra sin laddning av elbilen till när elpriset är som lägst. En bra överblick över dagens och morgondagens elpriser underlättar för att styra sin elkonsumtion och laddning till när priserna är som lägst. Detta gör att belastningen av elnätet kan jämnas ut vilket gör att elnätet ej behöver byggas ut för att hantera stora toppar av elkonsumtion.

Vi har kommit en bra bit på vägen men för att få en fungerande konsumentprodukt behöver det implementeras aktuell data för elbilen och vädret, vilket saknas i nuvarande version. Applikationen är anpassad som webbapplikation till en sju-tums skärm för Raspberry Pi och för att nå en större marknad bör applikationen skrivas om så att den anpassas till olika storlekar och upplösningar.

Referenser

[1] React, ”React. A JavaScript library for building user interfaces”, 2022. [Online] Tillgänglig: https://reactjs.org/ (hämtad: 2022-05-17).

[2] React Native, ”React Native. Learn once, write anywhere”, 2022. [Online] Tillgänglig: https://reactnative.dev/ (hämtad: 2022-05-17).

[3] Expo ”Create amazing apps that run everywhere”, u.å. [Online] Tillgänglig: https://docs.expo.dev/ (hämtad: 2022-05-17).

[4] GraphQL, ”A query language for your API”, 2022. [Online] Tillgänglig: https://graphql.org/ (hämtad: 2022-05-17).

[5] Apollo docs, ”Introduction to Apollo Client”, u.å. [Online] Tillgänglig: https://www.apollographql.com/docs/react/ (hämtad: 2022-05-17).

[6] Victory, ”Charting for React and React Native”, u.å. [Online] Tillgänglig: https://formidable.com/open-source/victory/about/ (hämtad: 2022-05-17).

[7] Jotai, ”Primitive and flexible state management for React”, u.å. [Online] Tillgänglig: https://jotai.org/ (hämtad: 2022-05-17).

[8] npm, ”react-native-reanimated-carousel”, u.å. [Online] Tillgänglig: https://www.npmjs.com/package/react-native-reanimated-carousel (hämtad: 2022-05-17).

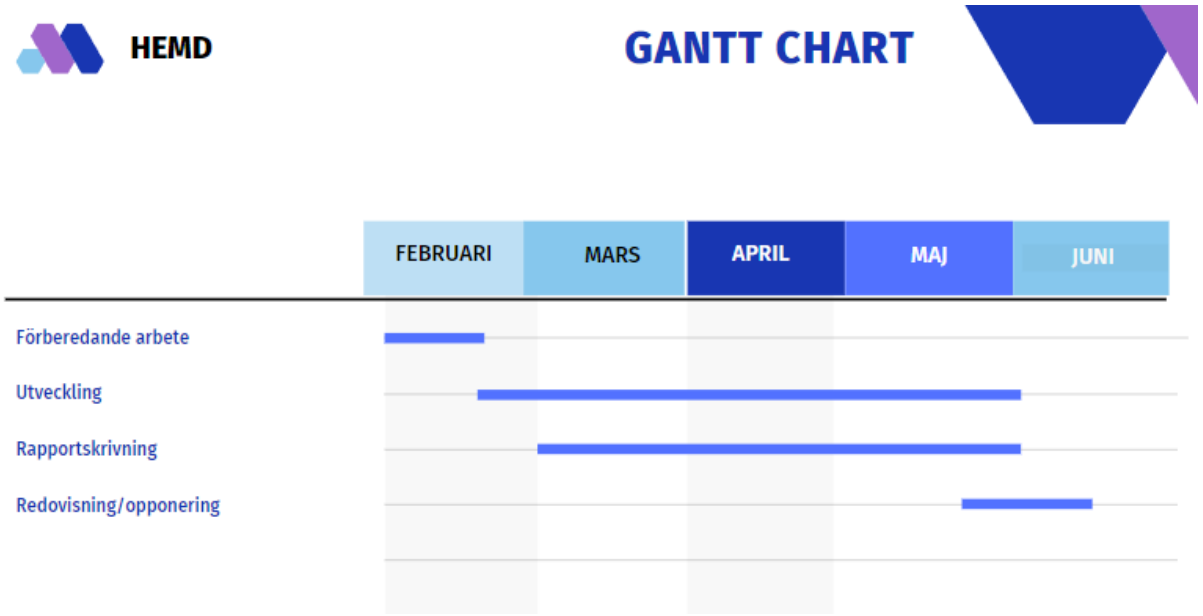
[9] Figma, ”Nothing great is made alone”, u.å. [Online] Tillgänglig: https://www.figma.com/ (hämtad: 2022-05-17).

[10] K. Y. Zarmi, N. N. A. Subhi , ”10 user interface elements for mobile learning application development”, IEE. Dec. 2015. [Online] Tillgänglig:https://ieeexplore.ieee.org/document/7359551

[11] Raspberry Pi Documentation, “Raspberry Pi OS”, u.å. [Online] Tillgänglig: https://www.raspberrypi.com/documentation/computers/os.html#introduction (hämtad: 2022-06-19).

[12] npm, react-native-modal”, 2022. [Online] Tillgänglig: https://www.npmjs.com/package/react-native-modal/ (hämtad: 2022-05-25).

Appendix A – Gantt-schema



INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020
www.chalmers.se

CHALMERS