CHALMERS

GÖTEBORGS UNIVERSITET

# Miljömedveten webbapplikation

## för optimering av Stena Line Freights transportrutter

Examensarbete inom Data- och Informationsteknik

Kevin Pham
Victoria Boquist

**Institutionen för Data- och Informationsteknik**

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

EXAMENSARBETE 2022

# Miljömedveten webbapplikation

för optimering av Stena Line Freights transportrutter

Kevin Pham
Victoria Boquist

GÖTEBORGS
UNIVERSITET

CHALMERS

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022

Miljömedveten webbapplikation
för optimering av Stena Line Freights transportrutter
Kevin Pham
Victoria Boquist



Uppdragsgivare: Stena Line Scandinavia AB, Freight
Handledare på Chalmers Tekniska Högskola: Gordana Dodig Crnkovic
Teknisk handledare på Stena Line Freight: Johan Lindskogen
Examinator: Lars Svensson, CSE

Examensarbete 2022
Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
SE-412 96 Göteborg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Göteborg, Sverige 2022

Miljömedveten webbapplikation
för optimering av Stena Line Freights transportrutter
Kevin Pham
Victoria Boquist
Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborgs universitet

# Abstract

Stena Line Freight is a company with one of Europe's largest freight route networks connecting ports, roads and rail. Today it is of much importance that the salespeople at Stena are able to present their customers with not only affordable but also eco-friendly solutions. The scope of this project was to implement a fundamental web application to compare carbon footprint between transporting goods by lorry on the roads compared to using Stena Lines intermodal routes, which means transporting goods by train and ferry, with the possibility of future further development. The project workflow was divided in sprints applying the Scrum framework and the MVP development strategy to reach the project goals. The results was a web application with environmental optimization of Stena Line freight routes, with an algorithm that can find the nearest intermodal route. The final product was developed in React using JavaScript and JSON to store data. Additionally Google API:s were used to retreive and calculate necessary route data and emissions.

Keywords: Web application, React, JSON, Scrum, MVP, JavaScript.

# Förord

Examensarbetet har utförts på institutionen för Data- och Informationsteknik på Chalmers tekniska högskola 2022.

Vi skulle vilja tacka vår uppdragsgivare Stena Line för all hjälp och stöd vid utförandet av projektet: Vår kontaktperson på företaget Sara Edling, tekniska handledare Johan Lindskogen, och UX-designer Niclas Jonsson.

Vi vill även tacka Gordana Dodig Crnkovic vår handledare på Chalmers tekniska högskola för all handledning under projektets gång.

Kevin Pham,
Victoria Boquist,
Göteborg, maj 2022

# Förkortningslista

| | |
|---|---|
| UI | User Interface |
| UX | User Experience |
| API | Application Programming Interface |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| NPM | Node Package Manager |
| Stena | Stena Line Freight |
| MVP | Minimum Viable Product |
| IDE | Integrated Development Environment |
| IR | Intermodal Route |
| JSON | JavaScript Object Notation |

# Innehåll

# Figurer

# 1
# Inledning

I detta kapitel presenteras bakgrund och syfte för webbapplikationen, samt diverse implementeringsmål med projektet.

## 1.1 Bakgrund

Ett av samhällets största problem idag är klimatförändringarnas konsekvenser i form av global uppvärmning. Denna uppvärmning sker på grund av utsläpp (främst koldioxid) som bildas vid förbränning av fossila bränslen. De globala koldioxidutsläppen har ökat successivt, och uppgår idag till cirka 35 miljarder ton per år. Luftens koldioxidhalt har ökat med cirka 50 procent sedan förindustriell tid, och den fortsätter att stiga med ungefär 0,4 procent per år [1]. Ett sätt att begränsa koldioxidutsläppen är att optimera transporter och motivera fler fossilsnåla transporter.

Stena Line Freights ruttnät är ett av Europas största och utgör viktiga hamnar och vägförbindelser över norra Europa [2]. Säljarna på Stena Line Freights har i många år beräknat bränslesnålast transportväg genom kalkylering med en mall i Excel. Detta tar dock tid och kräver att säljarna har komplett förståelse för beräkningarna i Excelmallen, vilket även medför risker för fel. Stena Line Freight har idag ett större behov av en webbapplikation där säljarna ska kunna mata in relevant data för en kundorder och få uträknade rutter i ett mer önskvärt format, t.ex. sortering efter sträcka, snabbhet, koldioxidutsläpp m.m. Detta möjliggör för interaktiv visualisering av transporter som kan effektivisera frakt och även minska kundernas klimatavtryck.

## 1.2 Syfte

Det grundläggande syftet med examensarbetet är att implementera en funktionell webbapplikation som jämför koldioxidutsläppet mellan att frakta gods med lastbil på vägar mot att använda Stena Lines intermodala rutter, dvs att frakta gods med tåg och färja. Applikationen ska kunna vidareutvecklas i framtiden. Intermodal transport betyder att transport av en godsenhet sker genom att flera olika transportsätt utnyttjas [3].

## 1.3 Mål

Det huvudsakliga målet för detta projekt är att implementera en webbapplikation som låter användaren välja avgång och destinationsort, samt vikt på det som skall transporteras. När detta är valt, ska applikationen simulera rutter sorterat efter önskvärd data, t.ex. sträcka, snabbaste rutt, koldioxidutsläpp m.m. Den färdiga webbapplikationen ska ha Stena Lines användargränssnitt och kunna vidareutvecklas.

## 1.4 Avgränsningar

Projektets omfattning kan bli mycket brett och resurs- och tidskrävande, vilket har lett till begränsning av omfånget med följande relevanta avgränsningar:

- Applikationen tillämpar sökning av intermodala rutter efter Stenas rutter.
- Applikationen anpassas efter de lastbilar som Stenas kunder kör.
- Applikationen anpassas efter de färjor som Stena kör.
- Applikationen tillämpar alla tågrutter efter ett medelvärde på Europas godståg i $CO_2$ ton per km.
- Data för beräkning av intermodulära rutter är givna och kommer ej kräva ytterligare undersökning.
- Applikationen ska endast utgå från städer i Europa.

# 2

# Metod

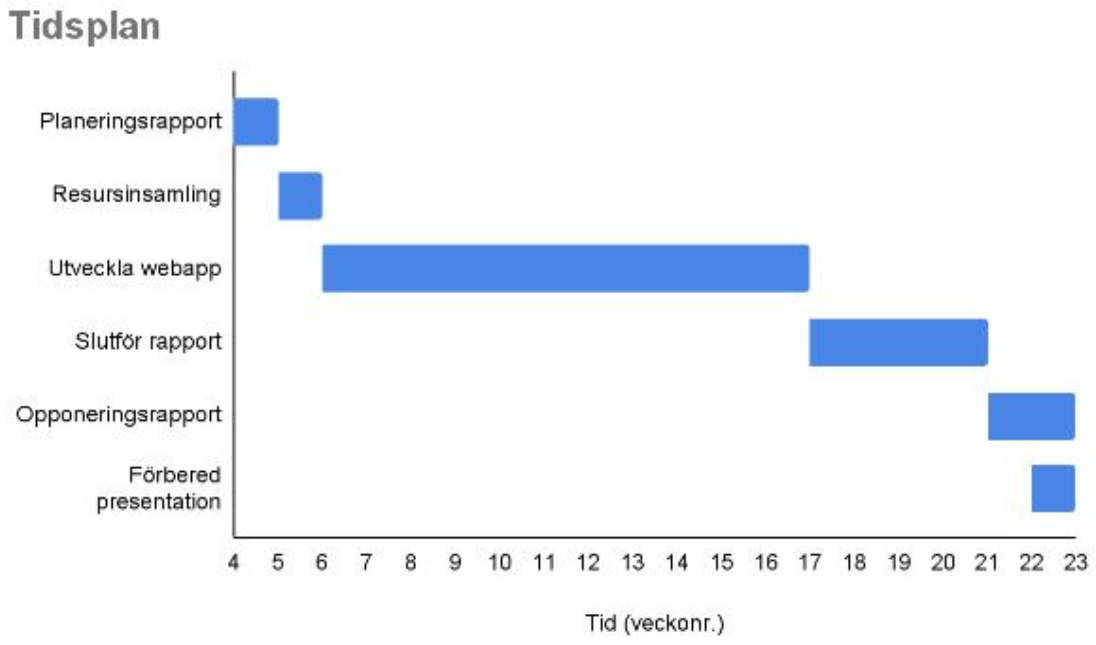
Projektets planerade tillvägagångssätt beskrivs i detta kapitel.

## 2.1 Uppdelning av arbete och tidsplan

Projektet har delats upp i 3 arbetsfaser:

1. Planering av arbetet.
2. Resursinsamling och planering av funktionalitet för simulatorn.
3. Utveckling och testning av webbapplikationen.

Enligt dessa faser har projektet planerats enligt nedan följande tidsplan. Arbetet startar igång officiellt den 24/1 som motsvarar vecka 4 i tidsplanen. Projektet pågår till 7/6 som motsvarar vecka 23.




**Figur 2.1:** Gantt-schema, planlagd tidsplan över projektets gång

## 2.2 Agilt arbetssätt

Agil betyder förmågan att förflytta sig snabbt och enkelt vars beteckning härstammar sig i engelskans agile. Likt ordets betydelse innebär ett agilt arbetssätt en strukturerad metodik för att arbeta i projekt som framhäver kommunikation och samarbete. Agil projektstyrning är idag allt vanligare inom olika branscher men härstammar sig i programmeringsprojekt som lagt grund för de 12 principer som utgör en agil projektstyrningsmetodik. Syftet är att man i sitt arbete ska värdesätta mänsklig interaktion framför verktyg och processer, anpassning till förändring av planering, resultat framför dokumentation och kundsamarbete framför kontraktsförhandling [4].

## 2.3 Scrum

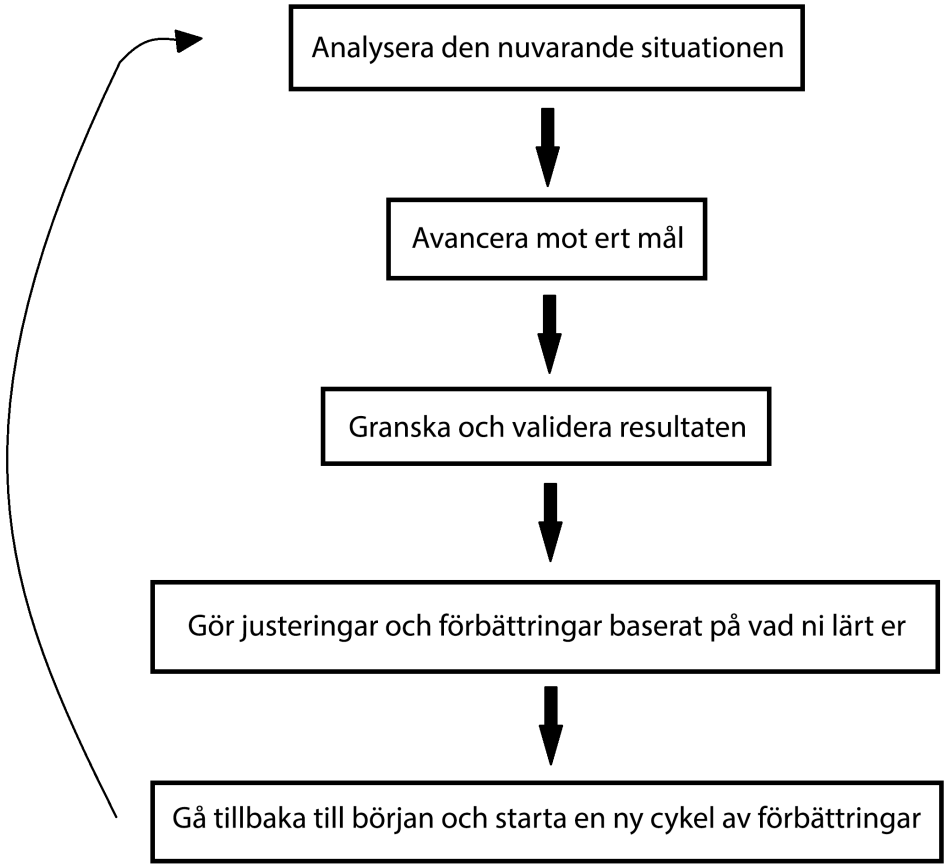
Scrum är ett ramverk inom agil projektledning som låter arbetslag att utveckla och hantera komplexa problem genom adaptiva lösningar, samtidigt som de produktivt fortsätter leverera produkter med hög kvalité [5]. Scrum innebär en arbetsmiljö som pågår i iterationer, så kallade sprintar och används mest frekvent inom mjukvaruutveckling. Det beskriver hur möten, verktyg och roller ska samverka för att hjälpa arbetslag strukturera arbete och reflektera på en sprints resultat. Ramverket är heuristiskt och bygger på kontinuerligt lärande och anpassning då oförväntade förändringar eller omprioriteringar kan ske abrupt i arbetsprocessen [6]. I sprintar [7] sker arbetet på följande sätt:

- En produktägare uttrycker behov och önskemål om vad produkten skall lösa
- Scrum teamet implementerar ett urval av dessa önskemål under en sprint
- Scrum teamet och dess intressenter inspekterar resultaten och justerar inför nästa sprint

## 2.4 MVP

Projektet kommer läggas upp utifrån ett agilt arbetssätt, där det tidigare beskrivna Scrum ramverket kombineras med MVP (Minimum Viable Product) [8]. MVP är en utvecklingsteknik och innebär att utveckla den enklaste formen av en produkt. Denna introduceras för användaren och efter respons används responsen för att förbättra den gamla versionen. Den slutgiltiltiga produkten släpps endast efter att ha fått tillräcklig med feedback från den tilltänkta användaren [9].

Likt scrum kommer projektet att byggas upp i iterationer, men istället för att implementera delar av webbapplikationen kommer fokus vara på att implementera helheten först och därefter bygga ut eller göra diverse ändringar. Flödesschemat i 2.2 visar hur ett sådant agilt arbetssätt ser ut.




**Figur 2.2:** Exempel på ett typiskt agilt arbetssätt

# 3
# Etik och hållbarhet

Detta kapitel diskuterar etiska frågeställningar för simulatorn samt hur simulatorn bidrar till en hållbar utveckling.

## 3.1 Etik

I denna sektion presenteras relevant etisk bakgrund för hur simulatorn är etiskt belagd. Etik består av tre huvudområden: metaetik (etikens teori), normativ etik, och tillämpad etik. Normativ etik studerar värden och normer och i den ingår dygdetik, utilitarismen och deontologi - pliktetik och rättighetsetik. Ingenjörsetik kombinerar dessa normativa element [10].

### 3.1.1 Dygdetik

Dygdetik är en gren inom normativ etik med betoning på en individs egenskaper och förhållningssätt. Dygdetiken är den äldsta normativa etiken, som bygger på att ett etiskt samhälle består av goda människor vars dygder utvecklas genom uppfostran och skolning i samhället. Dygdetik kännetäcknas av att utföra handlingar efter vad en dygdig människa gjort i samma situation [11].

### 3.1.2 Utilitarism

Utilitarismen är en moralteori baserad på idéen att summan av de goda konsekvenserna av ens handling ska vara så stor som möjligt. Denna moralteori kan dock inte tillämpas för många viktiga värden såsom miljövärden samt lycka och kunskap. Dessa tas dock med inom ideell utilitarism som innebär att i princip alla värden räknas in i den utilitariska summan. Enligt ideell utilitarism räknas däribland naturens egenvärde med bland den totala lyckan [12].

Simulatorn kommer ta beslut enligt utilitarismens principer där majoriteten av lyckan förhoppningsvis grundar sig i transportförändringar. Simulatorn ska vara miljömedveten och syftet är att väga in nyttan för även framtida generationer. Bortsett från resultat för simpla enstaka rutter där klimatavtrycket inte har en avgörande faktor, kommer simulatorn bidra till ett samhälle med fler bränslesnåla färdsätt. I längden kommer konsekvensen av simulatorn vara att säljarna enklare och effektivare kan sälja in mer miljövänliga rutter. I första hand betyder mer miljövänliga transporter att naturen sparas för de kommande generationerna vilket bidrar till deras bättre levnadsvillkor.

### 3.1.3 Pliktetik

En viktig gren inom etiken är pliktetik som ofta samverkar med utilitarismen. Det karaktäristiska för pliktetik är dess primära fokus på plikter som en moralisk agent har. Begreppet förekommer vanligtvis bland yrken som kräver att personer tar ansvar och ställning för handlingar. Pliktetiken byggs och baseras på normer som individer upplevt vara rätt eller spridit mycket lycka, men dessa är inte alltid tydligt definierade vilket kan resultera i förvirring [13]. Ingenjörer kan dock följa existerande hederskodexar som definierar plikter inom ingenjörsyrken. Sådana koder skrivs oftast av organisationer grundade av och är till för ingenjörer, exempelvis IEEE och ACM [14].

I detta fall är det pliktetiskt att skapa simulatorn då den i längden ska gynna mer hållbara transporter. Diverse pliktetiska konflikter kan framstå då användarna kommer vara lojala säljare hos Stena som har egna mål att sälja Stenas transportrutter. Utifrån detta perspektiv kan säljarna välja att inte berätta för kunder om Stenas rutter faktiskt är mer miljövänliga, vilket inte är pliktetiskt.

### 3.1.4 Ingenjörsetik

Simulatorn har i syfte att bidra till ett mer hållbart samhälle i form av att reducera aktuella klimatavtryck från transporter som Stena tillhandahåller. Det är i ingenjörens moraliska plikt och ansvar att värna utvecklingen av ny teknik och säkerställa att tekniken gynnar människa och samhälle [13]. I detta sammanhang krävs det att konsekvenserna av simulatorn är positiva och bidrar till bättre miljö för människor. Som skapare av simulatorn finns det ett ansvar att resultatens värden är korrekt givna, eftersom de sedan lägger till grund för beslut som i längden kan ha långsiktigt negativa konsekvenser. Exempelvis hade brister i simulatorn resulterat i rekommendation av rutter med högre klimatavtryck som då hade motverkat dess främsta syfte. Detta hade potentiellt riskerat en ökning av fossila bränslen i atmosfären som bidrar till förhöjd växthuseffekt, och i slutändan försämra livskvaliteten för människor. Det är även ingenjörens plikt att se över användandet av simulatorn för att säkerställa dess användning på längre sikt.

## 3.2 Hållbar utveckling

Den vanligaste tolkningen av hållbar utveckling uttrycks oftast som Brundtlands definitionen som beskriver hållbar utveckling enligt följande: ”En hållbar utveckling är en utveckling som tillfredsställer dagens behov utan att äventyra kommande generationers möjligheter att tillfredsställa sina behov” [15]. Detta berör ingenjörer även etiskt vars handlingar bör grunda sig i något som bidrar till mest lycka, välbefinnande, hälsa, välstånd och även andra preferenser som människor kan ha. Man ska samtidigt minimera lidanden och negativa konsekvenser av ens beslut / handling. Det kräver att ingenjören värnar om människans och samhällets bästa utifrån både miljö- och samhällsaspekt.

Detta stämmer väl in på den färdiga simulatorn, då denna kommer hjälpa Stenas säljare att motivera sina kunder att välja rutter med kortare väg och som har lägre koldioxidutsläpp. För enstaka transporter kommer inte skillnaden i koldioxidutsläpp vara stor, men vid ett större antal transporter kan det bli en avgörande skillnad i koldioxidutsläpp. Stena Line Freight som är en av världens största färjeoperatörer [16] och har hand om större antal transporter, har därmed möjlighet att med transportplaneringen till deras kunder göra skillnad i koldioxidutsläpp.

Ur ett socialt perspektiv är syftet för simulatorn att sälja fler rutter där färja eller tåg används och på så vis kommer lastbilschaufförerna ta del av fler raster och därmed få fler tillfällen att återhämta sig, vilket leder till en mer hållbar arbetsmiljö. Ytterligare är simulatorn gynnsam då kortare transportrutter medför lägre bränsleförbrukning och därmed används mindre av fossila bränslen. Då fossila bränslen som är en ändlig resurs är målet att bevara så mycket resurser för framtida generationer. Detta bidrar till att ur ekonomisk synpunkt är simulatorn etisk.

Slutligen från ett ekologisk perspektiv är det högst viktigt att koldioxidutsläppen drastiskt minskas för att uppnå UN:s klimatmål. Följs inte UN:s klimatmål, kommer den förhöjda globala temperaturen medföra skadliga konsekvenser för ekosystem [17][18]. Utifrån ett hållbarhetsperspektiv, bestående av social, ekonomisk och ekologisk perspektiv, gynnar simulatorn därmed samhället.

# 4

# Teknisk bakgrund

Detta kapitel beskriver de system och programmeringsspråk som lagt grund för webbapplikationen.

## 4.1 MVC

MVC är ett designmönster som separerar en applikation i tre huvudkomponenter bestående av Model, View och Controller. Dessa komponenter hanterar varsin aspekt av en tänkbar applikation som möjliggör skalbarhet och introducerar struktur i projektkod. De tre komponenternas generella områden är traditionellt:

**Model:** lagrar och hanterar all data relaterad logik som användaren kan tänkas använda.

**View:** levererar visuell information till användaren efter uppdatering från Model.

**Controller:** hanterar input från användaren och skickar denna data till Model.

MVC logiken används någorlunda som underlag i projektet genom att webappens komponenter är separerade och kommunicerar med Reacts inbyggda klassystem pprops"(properties) [19].

## 4.2 Node.js

Node.js är ett program som tillåter exekvering av JavaScript-kod på en server utan webbläsare. Programmets syfte är att bygga skalbara nätverksbaserade applikationer och används ofta till att skapa backend-delen av webbapplikationer [20].

## 4.3 React.js

React.js är ett effektivt och flexibelt JavaScript-bibliotek som används för att bygga användargränssnitt i webbutveckling. Det använder öppen källkod, är komponentbaserat och ansvarar för applikationens gränssnitt. Komponenterna går att återanvända med olika logik och minskar andelen kod som behöver skrivas. React använder sig av den virtuella representationen av ett gränssnitt, Virtual DOM, och uppdaterar enbart de delar som har ändrats, istället för att uppdatera alla komponenter, som vanliga konventionella webbapplikationer gör. Virtual DOM synkar med DOM (document object model) och det är när DOM ändras som uppdateringar sker på användargränssnittet [21], [22].

## 4.4 Stena komponentbibliotek

Stenas komponentbibliotek är ett designsystem och React-komponent-ramverk som har Stenas användargränssnitt [23]. Dess främsta syfte är att förenkla kommunikation mellan Stenas front-end utvecklare och UX-designer då man delar på resurser som finns tillgängligt, vilket möjliggör exakt implementation av prototypdesign. Komponentbiblioteket effektiviserar skapande och design av webbapplikationens uppbyggnad, då biblioteket innehåller många av de byggstenar som behövs för att implementera en typisk webbsida.

## 4.5 npm

Node Package Manager (npm) är pakethanterare för Node.js. Npm skapades år 2009 som öppenkällkod för att Javascript-utvecklare snabbt och enkelt skulle kunna dela och låna paketerad kod till sina projekt [24]. I detta projekt används npm för att bland annat hämta och använda Stenas komponentbibliotek.

## 4.6 Google API

Google dependencies API:s används för att effektivisera och förenkla utvecklingen av simulatorn. Dessa har installerats med hjälp av npm-paket och implementerats genom fördefinierade react komponenter från paketen för att undvika kontant upprepad användning av API länkar i källkod. Av de befintliga Google API:s har dessa använts främst:

- Distance Matrix API : Beräknar restid och distans mellan givna ursprung och destinationer med färdmedel i åtanke [25].
- Directions API: Beräknar och återger detaljerad ruttinformation mellan givna ursprung och destinationer med stöd för kollektivtrafik [26].
- Geocoding API: Konverterar adresser eller platser till geografiska koordinater i longitud och latitud och vice-versa [27].
- Places API: Återger ett utbud av platser som t.ex. städer baserat på användarens input i form av en autocomplete [28].

## 4.7 Font Awesome

Font Awesome är ett stort bibliotek av ikon och typsnittsverktyg som levererar skalbara vektorbilder. Biblioteket har över 1600 ikoner tillgängligt att använda gratis som även kan modifieras med CSS [29]. I detta projekt användes det till att förbättra och förtydliga användargränssnittet med ikoner. Font Awesome användes framförallt i samband med Stenas komponentbibliotek.

## 4.8 Github

Github är en plattform som används för att smidigt dela och arbeta med kod. Plattformen har ett inbyggt versionshanteringssystem som spårar redigeringshistorik och framför en konsistent överblick över ett projekts framsteg. Github underlättar för utvecklare att samarbeta och uppmanar till bättre struktur samt högre kodkvalité. I projekt kan Github förse med flera arbetsytor som sedan sammanfogas vilket är passande och fördelaktigt till scrum [30].

## 4.9 Microsoft Visual Studio Code

Visual studio code är en IDE (Integrated Development Environment) eller på svenska programutvecklingsmiljö, som är byggd på open source av Microsoft. Programutvecklingsmiljön har stöd för flera programspråk och inkluderar alla tänkbara verktyg en systemutvecklare kan tänkas behöva. Visual studio code används i projektet då det har bra stöd för React, Javascript och Github, samt underlättar utveckling av strukturerad kod [31].

## 4.10 Recharts

Recharts är ett grafbibliotek som erbjuder skalbara komponenter för React applikationer. Recharts komponenter följer principen separation och komposition, vilket gör att grafkomponenterna är tydligt separerade. Biblioteket är byggt i React och D3, ett Javascript bibliotek för visualisering av dynamiskt interaktiv data, samt har i projektet använts för att presentera data [32].

# 5
# Genomförande

I detta kapitel beskrivs genomförandet av projektet.

## 5.1 Planering

Arbetet började med en workshop på Stena där mål och syfte av projektet diskuterades i detalj. Det beslutades att webbapplikationen skulle vara implementerad i Javascript med React som stöd för att inte tillägna onödig tid att inhämta kunskap av nytt programmeringsspråk. Det skulle även visa sig vara effektivt i senare skede, då Stena har ett aktivt underlag och stöd för användargränssnitt med deras webui komponentramverk, baserat på Javascript. Därefter bokades kontinuerligt avstämningsmöten varje vecka med Stena för att redovisa och få återkoppling på arbetet. På dessa möten framfördes även designprototyper av Stenas UX designer som UI för webbapplikationen skulle efterlikna. För att säkerställa en användbar simulator i slutet av projektet så bestämdes det att arbetssättet skulle sträva efter MVP.

## 5.2 Resursinsamling och planerad funktionalitet

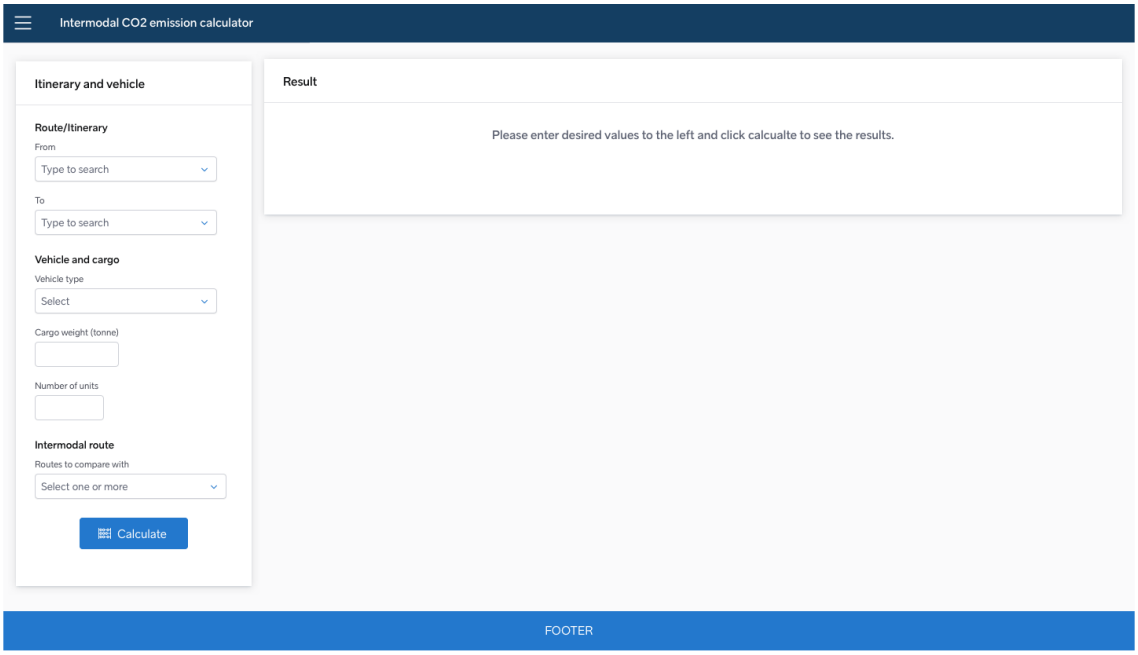
I samband med workshopen presenterades diverse dokument med nödvändig data för implementering av webbapplikationen. Simulatorn behövde matematiska beräkningar och mätvärden för att fungera, vilket överlämnades av Stena under planeringsfasen. Datan inkluderade relevanta lastbilars uthållighet och respektive klimatavtryck som skulle hårdkodas i json filer. Det planerades i början att tillhandahålla data i en databas, men eftersom att projektet förväntades bevara lite data och även brist på erfarenhet med databaser ajournerades idén. Mycket tid tillägnades till att studera tillgängliga resurser som excelmallen och samla in mer information, exempelvis tågrutter genom Europa.

Enligt MVP begränsades simulatorn till att visa de mest intressanta värdena en säljare kunde tänkas nyttja. Funktionaliteten skulle vara grundläggande med plats för framtida utveckling och för tillfället prioriterades 3 spalter med data för distans, tid och klimatavtryck. Det skulle inte heller vara för komplicerat att mata in städer för jämföring, vilket skulle lägga grund för användandet av Googles Autocomplete API i planeringen. Visionen för strukturen av koden var att organisera filerna i en liknande MVC struktur där man som utvecklare skulle kunna urskilja på var data kom från, beräknades och sedan redovisades.

## 5.3 Utveckling av webbapplikationen

Projektet startade med att skapa en gemensam och två enskilda arbetsytor på GitHub. Detta för att tillhandahålla olika versioner av programmeringskod. Den gemensamma arbetsytan skulle endast innehålla senast godkänd och testad kod, varav de enskilda arbetsytornas syfte var att kunna arbeta individuellt på olika delar av webbapplikationen. Därefter skapades en digital arbetstavla på Trello för att strukturera arbetsprocessen för varje sprint. Det var avsett att införa ett system där pågående och färdiga implementationer var dokumenterade.

Inför programmering av webbapplikationens design skapades en första prototyp. Denna baserades på tidigare nämnda krav i 1.3 Mål samt visualiserade presentering av data från mätvärden. Det var även viktigt att resultatvyn skulle presentera hur värden mellan IM och Road jämförs. Resultatet av första prototypen blev följande.




**Figur 5.1:** Prototyp 1

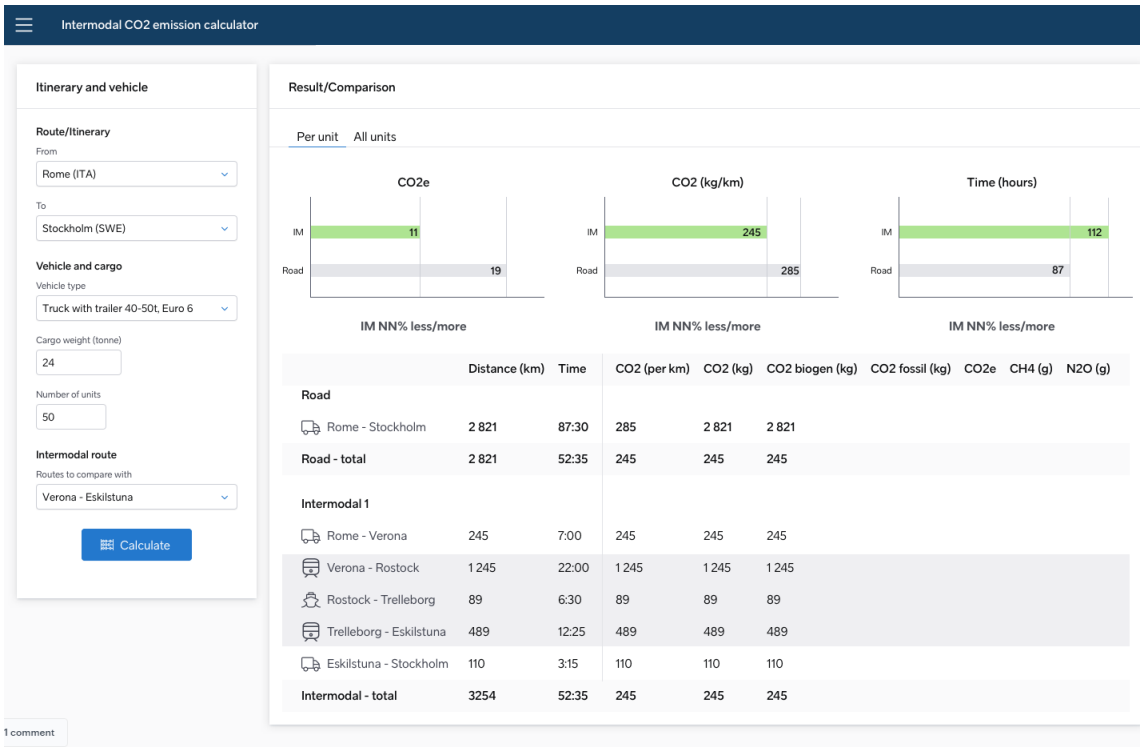
Programmeringen startade med att programmera en webbsida med de mest basala funktioner såsom flera input-fält och textfält som visade det användaren skrivit in. Detta gjordes enkelt med hjälp av ”Create React App” som automatiserar och skapar en frontend förbindelse till webbapplikationen samt upprättar utvecklingsmiljön för JavaScript [33].

Utifrån detta delades hemsidans användargränssnitt upp i 2 komponenter, Search och Result. Search innehöll endast ett input-fält och Result redovisade information som användaren hade skrivit in i Search. Detta gynnade användargränssnittet då användaren enkelt kunde särskilja på två paneler för olika syften. All grundfunktionalitet testades så att den fungerade korrekt innan beräkningar samt ytterligare design implementerades. Bortsett från programmeringen överfördes även alla resur-

ser med data till JSON filer som skulle simulera själva databasen. Efteråt började vi med att inrätta struktur för props så att ”Search” och ”Result” kunde kommunicera med varandra, däribland en check för att kolla att nödvändiga inputs var ifyllda. Inputen delades med useState variabler som låter komponenter ta del av den senaste data eller egenskaper hos en variabel [34].

När förbindelse mellan ”Search” och ”Result” upprättats kunde Google dependencies installeras. Det var nu som webbapplikationen behövde kommunicera med en extern part, vilket potentiellt kunde fördröja svarstid hos resultat beroende på särskilda faktorer som internetkoppling och användarinput. För att undvika hårdkodning av massiv data implementerardes Google maps autocomplete som möjliggjorde att man kunde skriva in städer som input istället för att välja från en lista. Funktionen tog hjälp av Googles sökmotor och begränsade inmatningsvärden till enbart städer. Vi använde oss också utav Google Maps API som returnerade begärda distanser och transporttider för olika färdmedel. Kort därefter implementerades en komponent avsedd för endast beräkning av klimatutsläpp vilken döptes till ”CO2”. Koden separerades sedan en ytterliggare gång där transporttid, distans och klimatavtryck skulle redovisa dess data i egna filer enligt planering.

Efter möte i senare sprint med Stenas säljare, korrigerades prototypen efter Stenas önskemål och behov. I den slutliga prototypen hade grafer och spaltformat lagts till samt ytterligare emissionsvärden. Resultatet efter korrigering av första prototypen presenteras nedan.




**Figur 5.2:** Prototyp 2

Eftersom att data responsen i Result behövde bearbetas strukturerades ”Result”

till att ha liknande funktion som Model i designmönstret MVC. Datan bearbetas av komponenten ”CO2” medans två nya paneler ”GraphView” och ”TabView” skulle rendera och redovisa tidigare resultat(View).

När målen för MVC var uppfyllda, påbörjades implementering av en funktion findNearby som skulle ge förslag på en rekommenderad rutt. Funktionen findNearby använder det tidigare svaret från Google directionsService för att få fram hela ruttens koordinater. Därefter skapas ett inhägnat område på 10 000 km runt koordinaterna där det utförs en sökning efter de städer som används i Stenas intermodala rutter. Denna anropar Google maps geometry poly funktion ”isLocationOnEdge” som kollar om en eller flera av städerna för en av Stenas intermodala rutter finns inom gränsen av det inhägnade området för de sparade koordinaterna. I praktiken skulle en jämförelse med varje koordinat ta alldeles för lång tid och höja kodens komplexitet. Det skulle även vara mer kostsamt med fler API kallelser inom kort intervall, då Google fakturerar efter frekvent användning av dess API tjänster. Därför implementerades i stället att koordinaterna som jämförs blir utvalda i ett så kallat stickprov. Därav minimerades antalet API anrop och koden behövde inte kolla igenom lika mycket data. Av dessa städer som finns i närheten väljs den intermodala rutten och därefter görs en jämförelse för att se ett ungefärligt totalavstånd medräknat avgång och destinationsort där den rutt som ger lägst avstånd blir den rekommenderade rutten.

# 6

# Resultat

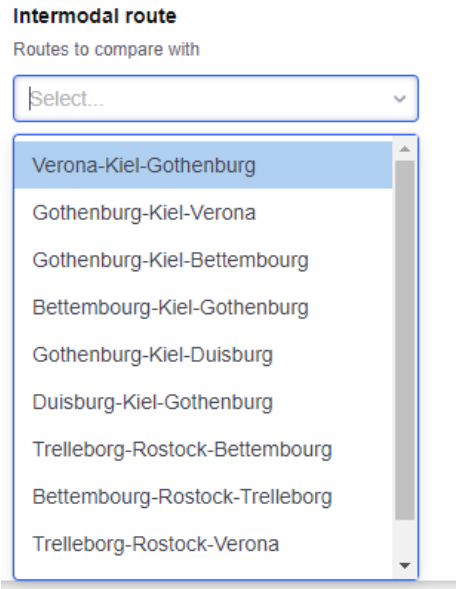
Nedan redovisas resultat av projektet uppdelat i två kategorier med kort beskrivning om vardera.

## 6.1 Design

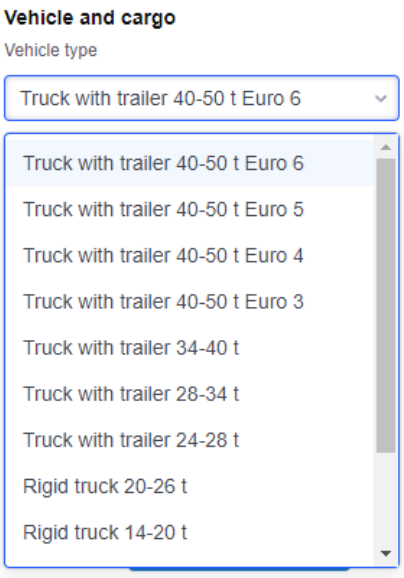
Designen för slutprodukten av webbapplikationen har tagit inspiration av och baserats på Prototyp 2 med viss skillnad. De två huvudsakliga komponenternas design beskrivs i detta kapitel.

### 6.1.1 SearchView

Sökningsvyns slutdesign skiljer sig endast i att Calculate knappen saknar samma ikon som används i prototypen, samt att städerna i ”From” och ”To” inte är valbara dropdown städer. Dropdown rutorna för dessa fält är i stället en sökbar ruta som ger användaren förslag på städer beroende på de bokstäver användaren skriver in.



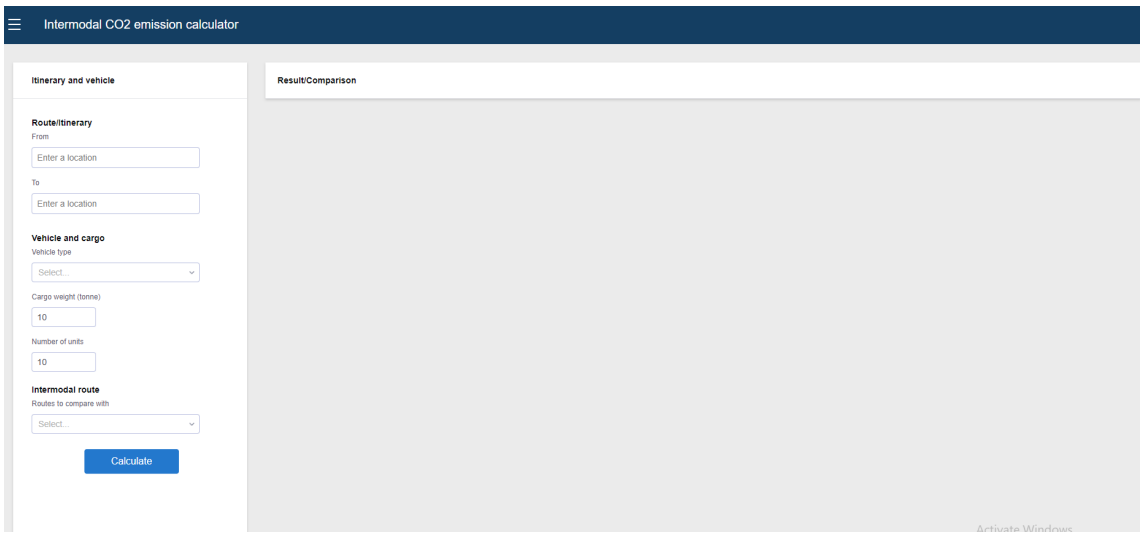

**Figur 6.1:** IR dropdown.



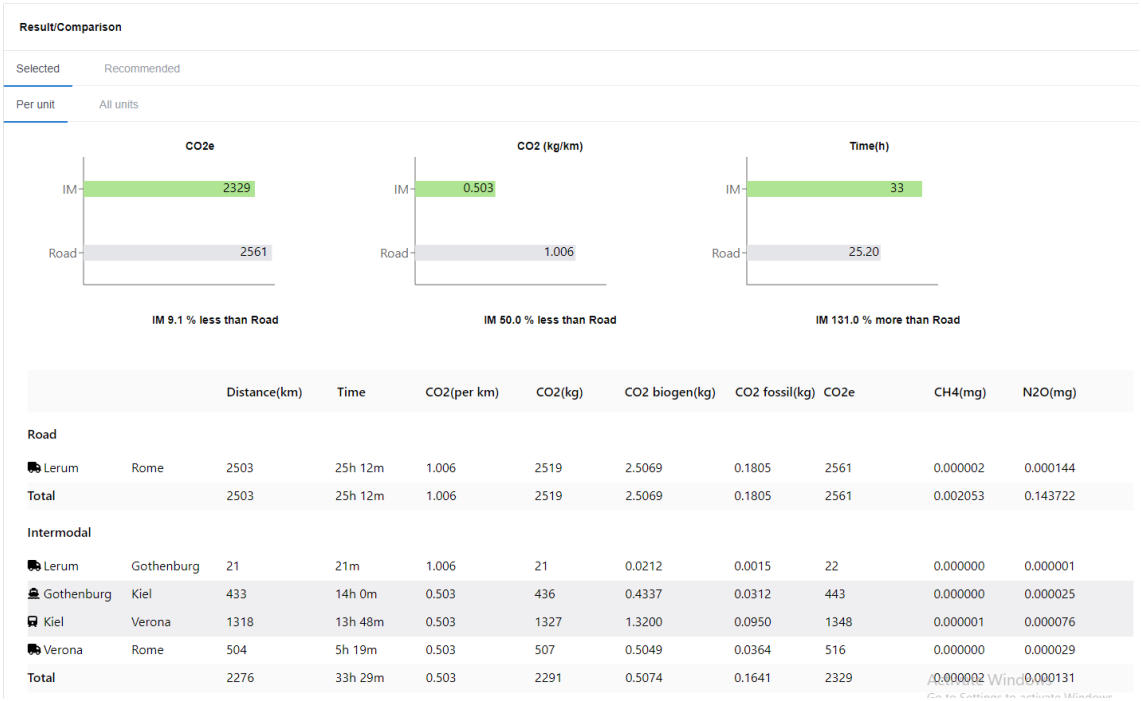

**Figur 6.2:** Vehicle dropdown

### 6.1.2 ResultView

Den färdiga applikationens resultatvy har ett utseende likt den andra prototypen. Skillnaden är att graferna som används för att presentera data inte visar stödlinjer inuti grafen. Font och textbredd skiljer sig också, webbapplikationen har använt Stena Lines komponentbibliotek och detta bibliotek har inte haft samma tillgång av fonter som det som använts i prototypen. Likt den färdiga prototypen presenteras mätvärden för ”Road” och ”Intermodal” rutt i tabellformat. För att förtydliga värdena mellan olika rutter särskiljs värdena med bredare avstånd mellan olika rutter. För varje rutt används en ikon för att visa vilket transportsätt som använts. För att synliggöra intermodala rutter har ruttens bakgrundsfärg en mörk grå ton för de rutter som är en av Stenas säljbara rutter. Därtill har klickbara flikar för ”Recommended” samt ”Selected” lagts till. Dessa är utanför projektets huvudsakliga mål och har därför inte tagits med i ovan presenterade prototyper. Flikarna använder däremot samma design som de andra klickbara flikarna ”Per unit” och ”All units” för att ge en sammanhängande design.

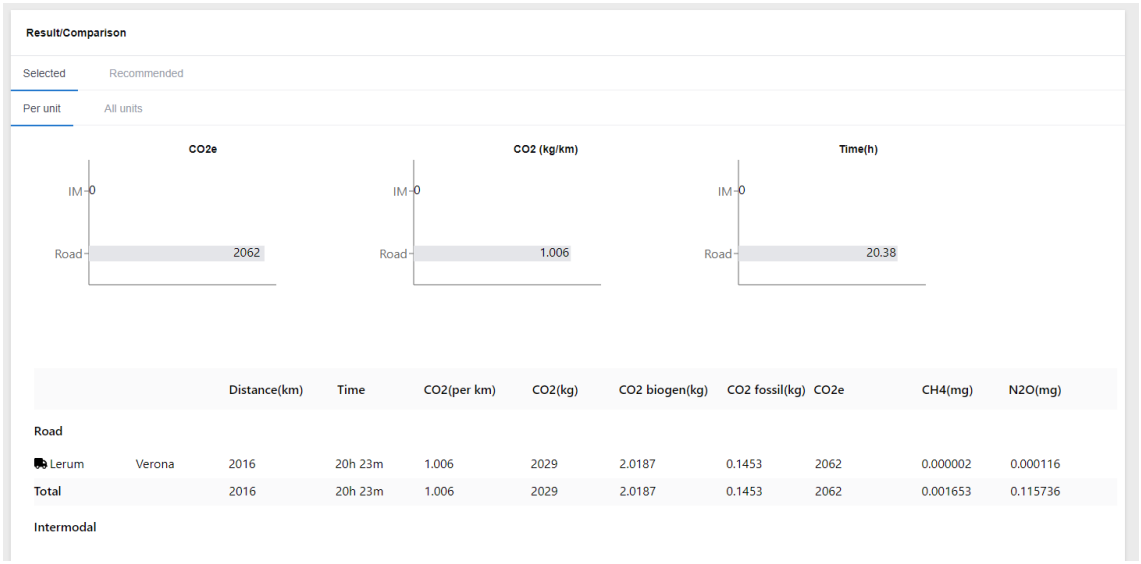



**Figur 6.3:** Hemsidan vid start



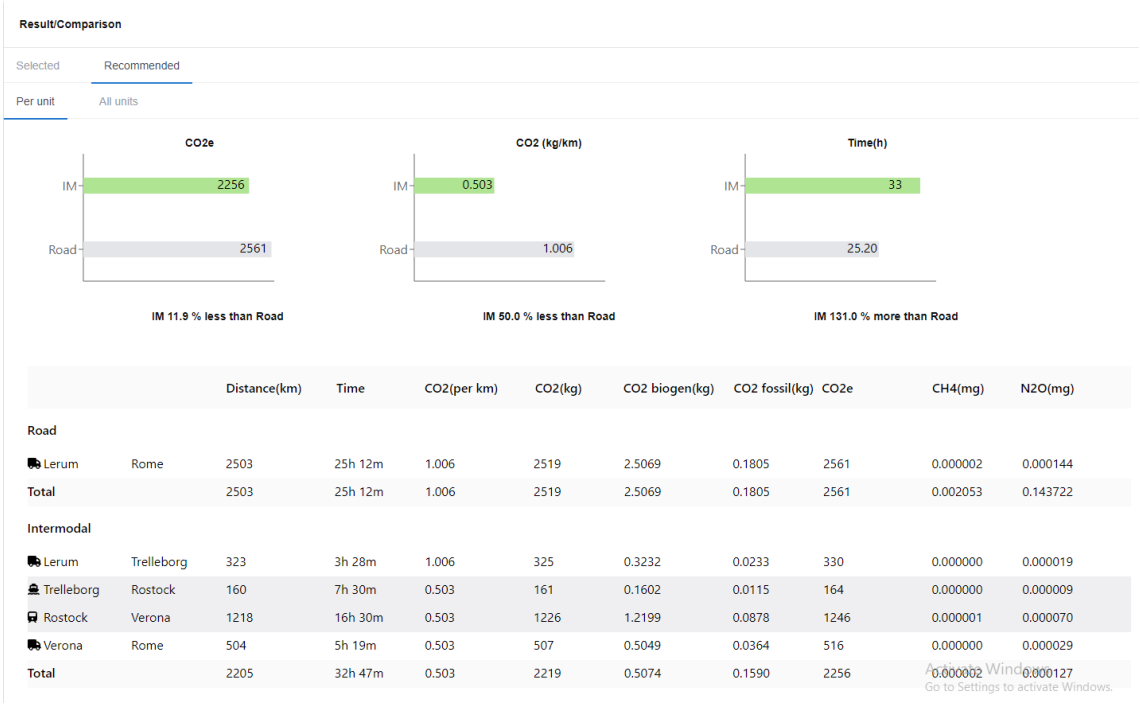

| | | Distance(km) | Time | CO2(per km) | CO2(kg) | CO2 biogen(kg) | CO2 fossil(kg) | CO2e | CH4(mg) | N2O(mg) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Road** | | | | | | | | | | |
| Lerum | Rome | 2503 | 25h 12m | 1.006 | 2519 | 2.5069 | 0.1805 | 2561 | 0.000002 | 0.000144 |
| **Total** | | 2503 | 25h 12m | 1.006 | 2519 | 2.5069 | 0.1805 | 2561 | 0.002053 | 0.143722 |
| **Intermodal** | | | | | | | | | | |
| Lerum | Gothenburg | 21 | 21m | 1.006 | 21 | 0.0212 | 0.0015 | 22 | 0.000000 | 0.000001 |
| Gothenburg | Kiel | 433 | 14h 0m | 0.503 | 436 | 0.4337 | 0.0312 | 443 | 0.000000 | 0.000025 |
| Kiel | Verona | 1318 | 13h 48m | 0.503 | 1327 | 1.3200 | 0.0950 | 1348 | 0.000001 | 0.000076 |
| Verona | Rome | 504 | 5h 19m | 0.503 | 507 | 0.5049 | 0.0364 | 516 | 0.000000 | 0.000029 |
| **Total** | | 2276 | 33h 29m | 0.503 | 2291 | 0.5074 | 0.1641 | 2329 | 0.000002 | 0.000131 |

**Figur 6.4:** Jämförelse mellan vald rutt och intermodal




| | | Distance(km) | Time | CO2(per km) | CO2(kg) | CO2 biogen(kg) | CO2 fossil(kg) | CO2e | CH4(mg) | N2O(mg) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Road** | | | | | | | | | | |
| Lerum | Verona | 2016 | 20h 23m | 1.006 | 2029 | 2.0187 | 0.1453 | 2062 | 0.000002 | 0.000116 |
| **Total** | | 2016 | 20h 23m | 1.006 | 2029 | 2.0187 | 0.1453 | 2062 | 0.001653 | 0.115736 |
| **Intermodal** | | | | | | | | | | |

**Figur 6.5:** Visa enbart vald rutt



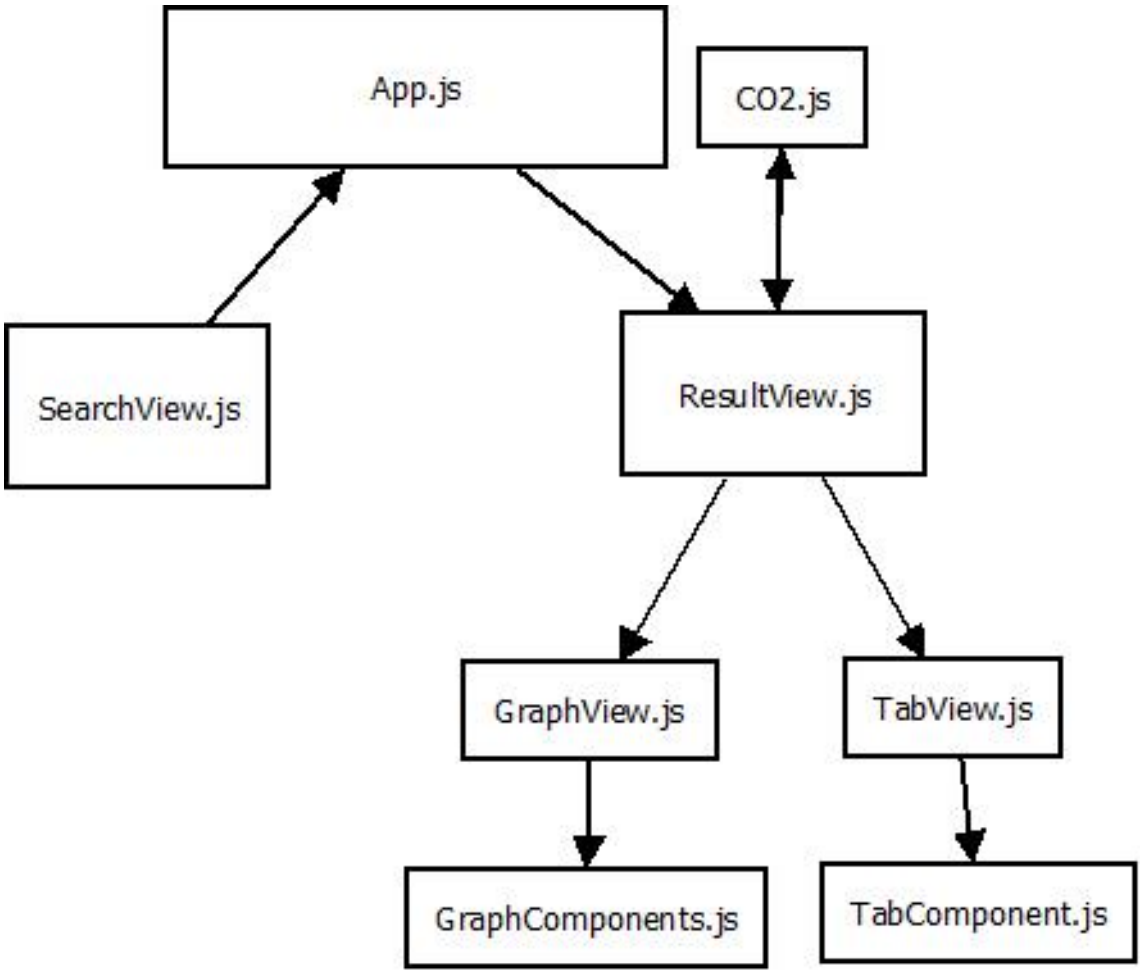

| | | Distance(km) | Time | CO2(per km) | CO2(kg) | CO2 biogen(kg) | CO2 fossil(kg) | CO2e | CH4(mg) | N2O(mg) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Road** | | | | | | | | | | |
| Lerum | Rome | 2503 | 25h 12m | 1.006 | 2519 | 2.5069 | 0.1805 | 2561 | 0.000002 | 0.000144 |
| **Total** | | 2503 | 25h 12m | 1.006 | 2519 | 2.5069 | 0.1805 | 2561 | 0.002053 | 0.143722 |
| **Intermodal** | | | | | | | | | | |
| Lerum | Trelleborg | 323 | 3h 28m | 1.006 | 325 | 0.3232 | 0.0233 | 330 | 0.000000 | 0.000019 |
| Trelleborg | Rostock | 160 | 7h 30m | 0.503 | 161 | 0.1602 | 0.0115 | 164 | 0.000000 | 0.000009 |
| Rostock | Verona | 1218 | 16h 30m | 0.503 | 1226 | 1.2199 | 0.0878 | 1246 | 0.000001 | 0.000070 |
| Verona | Rome | 504 | 5h 19m | 0.503 | 507 | 0.5049 | 0.0364 | 516 | 0.000000 | 0.000029 |
| **Total** | | 2205 | 32h 47m | 0.503 | 2219 | 0.5074 | 0.1590 | 2256 | 0.000002 | 0.000127 |

**Figur 6.6:** Visa rekommenderad rutt vald

## 6.2 Kod

Applikationen är uppbyggd med huvudkomponenterna SearchView, ResultView och CO2. Användaren interagerar med SearchView och väljer nödvändig information om rutten som skall sökas, vilken typ av lastbil som används samt lastens vikt. Användaren kan också här välja att jämföra rutten med en av Stena Lines intermodala ruttlösningar. Komponenten ”ResultView” tar emot all information som användaren valt och använder dessa för att sedan kalla på Googles API:s med denna information. Den använder svaren från Googles API:s för att kalla på ”CO2” med distans värden för den valda rutten, varpå ”CO2” kalkylerar den valda ruttens utsläpp och returnerar utsläppsdata som sedan visas med ResultViews graf och tabulär vy (GraphView och TabView).

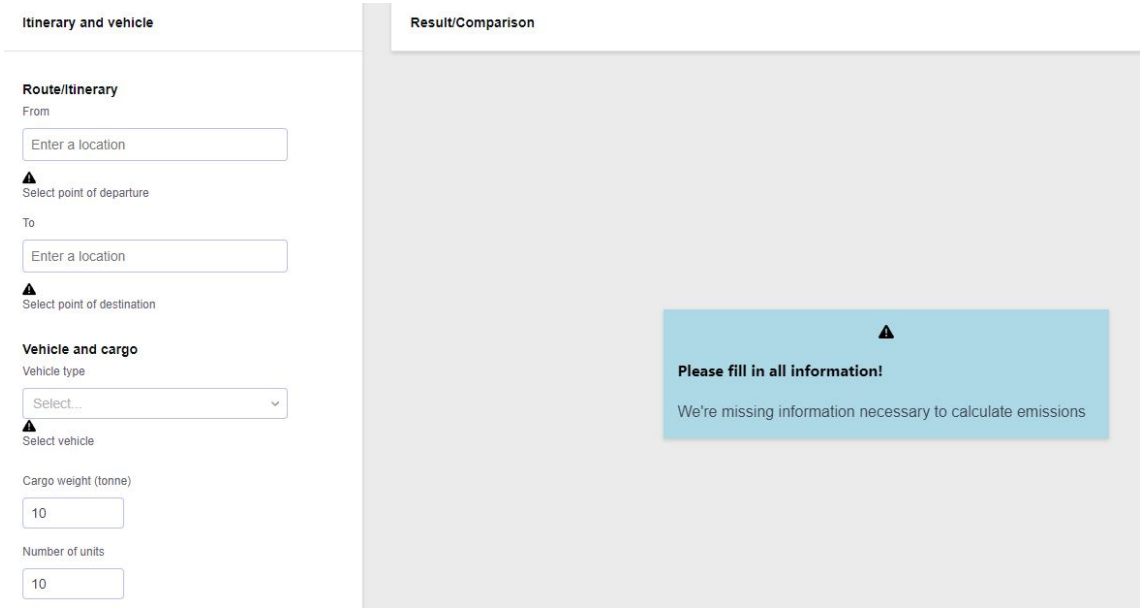
Denna arkitektur illustreras i figur 6.7.




**Figur 6.7:** Diagram över kodstruktur

### 6.2.1 SearchView

SearchView består av textinputfält där användaren skriver in avgång och destionationsstad, lastvikt, antal transport enheter, samt dropdown-fält för att välja vilken typ av lastbil som kommer köras. Textinputfälten för avgång och destinationsstad använder Google Autocomplete som ger platsförutsägelser baserat på användarens textinmatning. Ytterligare kan användaren även välja en av Stena Lines intermodala rutter att jämföra med. För att förhindra undefined error kollas även att alla obligatoriska värden är ifyllda innan SearchView interagerar med ResultView. Om enstaka information saknas uppmärksammas användaren med varningsikon och felmeddelande under det saknade värdet. Om däremot alla värden saknas får användaren en pop up notis med ytterligare felmeddelande.




**Figur 6.8:** Felnotifikation vid inmatning av noll värden

### 6.2.2 ResultView

ResultView anropar först funktionen getRoute så fort alla nödvändiga paramterar från SearchView fyllts i. Funktionen getRoute anropar i sin tur Google Maps API directionsService, som returnerar den snabbaste körvägen mellan utgångs- och destinationsort och som undviker färja som transport. Vissa rutter måste däremot använda färje som transporträtt, och i dessa fall kommer dom rutterna tas med. Efter att ha fått ett svar från directionService, sparas svaret och alla koordinater för denna rutt i form av useState variabler. Koordinaterna sparas undan för att användas ifall användaren väljer att söka en rekommenderad rutt.

Därefter kallas funktionen getIntermodal som används för att jämföra rutten med en intermodal rutt. Denna använder google maps distanceMatrix för att få fram tid och distans mellan städer. Till exempel om användaren vill jämföra rutten Göteborg-Rom med den intermodala rutten Göteborg-Kiel-Verona, kommer distanceMatrix att ta fram distans och tid mellan Göteborg och Kiel samt Verona och Rom. Dessa värden sparas sedan i useState variabler.

Om användaren väljer att jämföra med rekommenderad rutt anropas funktionen findNearby, som använder de ovan nämnda sparade koordinaterna och skapar ett inhägnat område på 10 000 km runt dessa koordinater. Sedan utförs en sökning med städer som används i Stenas intermodala rutter, och därefter anropar findNearby Google Maps Geometry Poly funktion isLocationOnEdge, som kollar om en eller flera av städerna för en intermodal rutt finns inom eller på gränsen av det inhägnade området för de sparade koordinaterna. Även en koll görs för att se till att färdriktningen stämmer. Om alla dessa parametrar stämmer sparas rutten undan. Av dessa rutter görs därefter en jämförelse efter vilken rutt som har minst totala reseavstånd mellan avgångsstad och första staden i den valda rutten, samt sista staden i den

valda rutten och destination. Detta är för att dra ner tidskomplexiteten, den rutt som väljs blir också den rekommenderade rutten. Efter detta kallas getIntermodal, dock i stället för att jämföra origalrutten med intermodal rutt, gör den detta med den rekommenderade rutten.

#### 6.2.2.1 CO2

CO2 tar emot information om bland annat färdmedel för rutten. Denna används för att CO2 ska veta vilken av beräkningarna som skall utföras då CO2 innehåller funktioner för att beräkna utsläpp för lastbil, färja och tåg. För tågberäkningarna valde vi att använda genomsnittligt utsläppsvärde för tåg, då utsläppsinformation om transporttågen saknades. Vi anser att ett genomsnittligt utsläppsvärde inte kommer påverka resultat avsevärt och därför gör det inte stor skillnad.

#### 6.2.2.2 GraphView

GraphView presenterar den intermodala och den vanliga ruttens $CO_2$ utsläpp i kg, kg/km och tid för den totala resan i grafer för vardera värde. Den består av tre GraphCompontents som vardera returnerar varsin graf. Varje graphComponent tar emot värde på intermodal och den vanliga ruttens data och på detta vis blir Resultatvyn skalbar, då det är enkelt att bygga ut eller ta bort grafer för diverse värden.

#### 6.2.2.3 TabView

TabView presenterar den intermodala och den vanliga ruttens utsläppsdata, tid och distans i spaltformat. Denna komponent tar emot beräknad utsläppsdata för rutterna samt tid, distans och transportmedel. Dessa visas för varje del av rutten i kronologisk ordning. TabView består av två TabComponents, en för intermodal rutt och den andra för vanlig rutt. Likt GraphView är denna komponent också skalbart implementerad. TabComponent returnerar rader med varje rutts resetid, distans och utsläppsdata samt en sista rad med summerade värden för den totala rutten.

### 6.2.3 Tester

Utförlig testning har skett särskilt på getIntermodal funktionen för att algoritmen ska söka efter rätt avstånd. Algoritmen är programmerad för att detektera om den intermodala rutten ska räknas från början, slutet eller som mellanliggande rutt. Exempelvis mellan rutten Lerum-Verona om användaren valt att jämföra med rutten Göteborg-Kiel-Verona kommer algoritmen detektera detta och söka avstånd och tid för rutten Lerum-Göteborg-Kiel-Verona. Medan rutten Lerum-Rom detekterar för Lerum-Göteborg-Kiel-Verona-Rom.
För att inte ge missvisande resultat kommer webbapplikationen visa felmeddelandet ”route not possible” för Road och/eller Intermodal spalten om directionService inte hittar en körväg.

# 7
# Diskussion

Detta kapitel presenterar en reflektion över projektet samt diskuterar möjliga förbättringar för applikationen i framtiden.

## 7.1 Reflektion på arbetet

De mål som skulle uppfyllas var att implementera en webbapplikation som simulerar intermodala rutter och låter användaren välja avgång och destinationsort, samt vikt på det som skall transporteras. Dessa mål har uppfyllts då applikationen simulerar rutternas utsläppsvärden i både graf och tabellform, där graferna visar koldioxidutsläpp per km, tid och totalt koldioxidutsläpp i kg. Då applikationen använder Googles distancematrix och directionservice API tillkommer det fördröjningar, varav dessa har uppmätts max 1-2 sek, och påverkar inte användarupplevelsen negativt. Den färdiga webbapplikationen använder Stena Lines användargränssnitt och är byggd för att kunna vidareutvecklas.

Utanför projektets MVP har en rekommenderad funktion också implementerats. Denna är väldigt grundläggande och behöver dock förbättras om den ska tillämpas av Stena Line i framtiden.

Enligt tidsplanen skulle utvecklingen av simulatorn ta 8 veckor, vilket har hållits. Däremot hade den planerade tiden för att implementera funktionen som returnerar rekommenderad rutt behövt minskas, eftersom att problem hos Googles directionservice API tillkom mot slutet av projektet som medförde svårlösta programmeringsfel. Problemet spårades till API-anropet av DistanceService som tidigare fungerade men nu returnerade felkod i stället. Dessa problem kunde dock redas ut med hjälp från vår tekniska handledare på Stena.

## 7.2 Etik och Hållbarhet

Utifrån ett ingenjörsperspektiv ser vi ett ansvar att utföra ett arbete som förhoppningsvis bidrar till en hållbar utveckling för både miljö och samhälle. Projektet utgår ifrån plikter vi har som utvecklare att resultatet har en utilitaristisk gynning inkluderat framtida generationer. Vi ser gärna ett minskat klimatavtryck från lastbilar där det finns möjlighet att transportera sitt gods på tåg eller båtar. Den generella tanken är att man ska förlita sig mer på fossilfri energi än fossila bränslen då detta är något vi aktivt kämpar med som globalt mål idag.

Vår arbetsprocess bygger på instruktioner från Chalmers som är baserade på hederskodex och har haft det i åtanke vid strukturering av arbetsflödet. Det ligger i säljarnas ansvar att på ett etiskt sätt framföra omanipulerad data till kunder, som inte står i konflikt med företagets intresse för vinster.

## 7.3 Framtida arbete

Som tidigare benämnt var ett utav de främsta målen att webbapplikationen skulle ha möjlighet för vidareutveckling i framtiden. Nedan beskrivs tankar och idéer kring hur webbapplikationen kan förbättras.

### 7.3.1 Mätvärden

Alla resetider mellan Stenas intermodala rutter är inte helt korrekta, vilket kan ge missvisande resultat. Prioritet låg på att utveckla fungerande funktionalitet, i stället för att tillämpa korrekta mätvärden. Detta då det var tidskrävande att hitta denna data, och den även saknades hos Stena Line. Detta är däremot i framtiden inte svårt att lägga in i filen intermodalroutes.json, då det är manuell inmatning av data som kan utföras av någon som inte är insatt i programkoden. Det finns även stor potential för att implementera en faktisk databas som lättgör hantering av större data. Det hade tillåtit användarna att spara och distribuera rutter för att jämföra mätvärden vid olika tillfällen.

### 7.3.2 Stöd för mobil och surfplatta

Redan i början av planeringen valde vi att inte lägga ner stort fokus på UI och design. Applikationen använder Stenas användargränsnitt och därtill finns det stöd för att kunna anpassas för de flesta skärmstorlekar. Den saknar dock stöd för mobil och surfplatta, vilket kan vara önskvärt för säljarna i framtiden då det innebär att applikationen får mobilitet. Det öppnar upp för flexibilitet med en responsiv webbapplikation för säljarna att nå simulatorn där datorer ej finns tillgängligt.

### 7.3.3 Rekommenderad algoritm

Funktionen findNearby ger inte den bästa rutten. Detta är på grund att distansen den räknar ut använder för få städer av den totala rutten och räknar approximalt ut distansen. Om denna funktion ska tillämpas i framtiden behöver en algoritm implementeras som tar fram korrekt distans mellan alla potentiella städer som rutten kan passera. Detta hade varit enklast att implementera med Google distancematrix genom att göra anrop till API:t mellan varje rutt, men funkar inte i praktiken då datan fås asynkront. Därmed blir tidskomplexiteten för varje anrop för stor om vi vill få fram avståndet för flera rutter. Vi valde därför att räkna ut avståndet med hjälp av de koordinater som används i rutten. I nuläget används Eulers formel för ett approximalt värde, men denna formel skulle kunna bytas ut till Haversine som även anpassar koordinaters avstånd efter jordens kurvning. Exempel på algoritm

hade varit Djikstras algoritm som tar fram den kortaste vägen mellan noder. För att använda Djikstras algoritm hade en nodstruktur däremot behövt implementeras. Därtill hade algoritmen förbättrats om den även tog hänsyn till andra värden än distans, exempelvis kunna välja en rekommenderad rutt baserad på lägst tid eller metangas utsläpp.

### 7.3.4 Grafbibliotek

I projektet användes grafbiblioteket Recharts, men detta bibliotek har gett problem med rerendering så att grafvärden inte uppdateras korrekt. Det har därmed varit svårare att anpassa graferna efter prototypens design. Exempelvis har text inuti grafen som ska presentera värden gett ovanstående problem med rerendering.

# 8

# Slutsats

Syftet med projektet var att implementera en webbapplikation som kan jämföra och simulera koldioxidavtrycket mellan att frakta gods via Stena Lines intermodala rutter, mot att frakta gods via lastbil på vägar och genom optimering föreslå miljövänliga alternativ. Simulatorn var även planerad att ha möjlighet till framtida vidareutveckling och har programmerats i React med Javascript i fokus. Arbetet har utförts efter de planerade fasernas tidsplan och under dessa har projektets mål uppfyllts och påbörjan på ytterligare funktionalitet utöver MVP, en rekommenderad ruttförslag har implementerats. Därtill har även testning genomförts både under implementeringsfas och på den färdiga applikationen. Resultatet är en grundläggande simulator som med hjälp av varierande Google API:n redovisar mätvärden för geografiska rutter baserat på användarens inmatningsvärden. Målgruppen för simulatorn är Stena Lines säljare som förhoppningsvis kan rekommendera ett mer miljövänligt alternativ till sina kunder.

# Litteraturförteckning


[1] Naturvårdsverket, “Därför blir det varmare”. [Online] Tillgänglig: https://www.naturvardsverket.se/amnesomraden/klimatfakta/darfor-blir-det-varmare/ (hämtad: 2022-01-27).

[2] Stena Line, “Our Company,” [Online]. Tillgänglig: https://www.stenaline.com/about-us/our-company/ (hämtad: 2022-01-27).

[3] Logtrade, “INTERMODALA TRANSPORTER,” [Online]. Tillgänglig: https://www.logtrade.se/ordlista/intermodala-transporter/ (hämtad: 2022-01-31).

[4] P. Tegborg, ”Vad innebär det att jobba agilt?,” *Kntnt*, 2018. [Online]. Tillgänglig: https://www.kntnt.se/vad-innebar-det-att-jobba-agilt/18725 (hämtad: 2022-03-05).

[5] Scrum.org, ”WHAT IS SCRUM?” u.å. [Online]. Tillgänglig: https://www.scrum.org/resources/what-is-scrum (hämtad: 2022-03-08).

[6] Atlassian, ”What is Scrum?” u.å. [Online]. Tillgänglig: https://www.atlassian.com/agile/scrum (hämtad: 2022-03-08).

[7] H. Kniberg, Scrum and XP from the Trenches. 2. uppl., 2015.

[8] Heflo, “Can Scrum and Minimum Viable Product methodologies work together?” [Online]. Tillgänglig: https://www.heflo.com/blog/agile/minimum-viable-product-scrum/ (hämtad: 2022-01-27).

[9] The Economic Times, ”What is ’Minimum Viable Product’,” [Online] Tillgänglig: https://economictimes.indiatimes.com/definition/minimum-viable-product (hämtad: 2022-03-08).

[10] Kenneth Einar Himma and Herman T. Tavani (Ed.) (2008) The Handbook of Information and Computer Ethics. John Wiley & Sons. Tillgänglig: https://www.academia.edu/24892978/The_Handbook_of_Information_and_Computer_Ethics

[11] Internet Encyclopedia of Philosophy IEP, ”Virtue Ethics,” u.å. [Online]. Tillgänglig: https://iep.utm.edu/virtue/ (hämtad: 2022-05-27).

[12] Sarah Spiekermann (2016) Ethical IT Innovation: A Value-Based System Design Approach, CRC Press

[13] S.O. Hansson, ”Teknik och etik” Avdelningen för Filosofi, Institutionen för Filosofi och Teknikhistoria, KTH, Stockholm, Sverige, 2009. [Online]. Tillgänglig: https://people.kth.se/~soh/tekniketik.pdf

[14] IEEE, ”IEEE Code of Ethics,” u.å. [Online]. Tillgänglig: https://www.ieee.org/about/corporate/governance/p7-8.html (hämtad: 2022-05-13).

[15] F. Hedenus, M. Persson och F. Sprei, Hållbar utveckling - nyanser och tolkningar. 1. uppl., Lund, Sverige: Studentlitteratur AB, 2018.

[16] Freightlink ”Stena Line,” u.å. [Online]. Tillgänglig: https://www.freightlink.co.uk/ferry-operator/stena-line (hämtad: 2022-05-15).

[17] 10 New Insights in Climate Science 2021, “A year of climate-related science in review”. [Online] Tillgänglig: https://10insightsclimate.science/ (hämtad: 2022-01-27).

[18] Charles J. Kibert, Leslie Thiele, Anna Peterson and Martha Monroe (2018) The Ethics of Sustainability.

[19] Tutorialspoint, ”MVC Framework - Introduction,” u.å. [Online]. Tillgänglig: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (hämtad 2022-03-09).

[20] OpenJS Foundation, ”About Node.js,” u.å. [Online]. Tillgänglig: https://nodejs.org/en/about/ (hämtad: 2022-03-08).

[21] Simplilearn, “What is React: Definition, Why ReactJS, its Features and Installation,” 2021. [Online]. Tillgänglig: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (hämtad: 2021-11-22).

[22] React, “Tutorial: Intro to React,” u.å. [Online]. Tillgänglig: https://reactjs.org/tutorial/tutorial.html (hämtad: 2021-11-22).

[23] Stena Github. [Online]. Tillgänglig: https://github.com/StenaIT/stenajs-webui (hämtad: 2022-03-05).

[24] npm Docs, ”About npm,” u.å. [Online]. Tillgänglig: https://docs.npmjs.com/about-npm (hämtad: 2022-03-05).

[25] Google Maps Platform,”Distance Matrix Service,” u.å. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/javascript/distancematrix (hämtad: 2022-03-08).

[26] Google Maps Platform,”Directions Service,” u.å. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/javascript/directions (hämtad: 2022-03-08).

[27] Google Maps Platform,”Geocoding API,” u.å. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/geocoding (hämtad: 2022-03-08).

[28] Google Maps Platform,”Places API,” u.å. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/places/web-service (hämtad: 2022-03-08).

[29] S. Bass, ”How to use Font Awesome on your site ,” *UCI Sites*, 2021 [Online]. Tillgänglig: https://sites.uci.edu/blog/tips-tricks/how-to-use-font-awesome-on-your-site/ (hämtad 08-03-22).

[30] Github, ”About Git,” u.å. [Online]. Tillgänglig: https://docs.github.com/en/get-started/using-git/about-git#how-github-works (hämtad 2022-03-09).

[31] Visual Studio Code, ”Visual Studio Code FAQ ,” u.å. [Online]. Tillgänglig: https://code.visualstudio.com/docs/supporting/FAQ (hämtad: 2022-03-09).

[32] Recharts, ”Redefined chart library built with React and D3,” u.å. [Online]. Tillgänglig: https://github.com/recharts/recharts (hämtad: 2022-05-27).
[33] React, “Create a New React App” u.å. [Online]. Tillgänglig: https://reactjs.org/docs/create-a-new-react-app.html#create-react-app (hämtad: 2022-05-19).
[34] React, ”Using the State Hook,” u.å. [Online]. Tillgänglig: https://reactjs.org/docs/hooks-state.html (hämtad: 2021-11-22).
[35] Office of Rail and Road”Rail Emissions 2020-21,” 2021. [Online]. Tillgänglig: https://dataportal.orr.gov.uk/media/1993/rail-emissions-2020-21.pdf (hämtad: 2022-03-09).

**Institutionen för Data- och Informationsteknik**
**Chalmers tekniska högskola**
**Göteborgs universitet**
Göteborg, Sverige
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS