CHALMERS

ShareConnect

En webbapplikation med tillämpning av spelifiering för att främja en cirkulär ekonomi

Examensarbete inom Data- och Informationsteknik

NIKLAS PAASONEN
LUCAS ROSVALL

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

EXAMENSARBETE

ShareConnect

Vidareutvecklingen av en webbapplikation som, genom att tillåta användare annonsera ut begagnade produkter och tillämpning av spelifiering, ska främja en cirkulär ekonomi och hållbar utveckling

NIKLAS PAASONEN
LUCAS ROSVALL

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2020

ShareConnect
Vidareutvecklingen av en webbapplikation som, genom att tillåta användare annonsera ut begagnade produkter och tillämpning av spelifiering, ska främja en cirkulär ekonomi och hållbar utveckling

NIKLAS PAASONEN
LUCAS ROSVALL

Examinator: Peter Lundin, Data- och Informationsteknik
Handledare: Sakib Sistek, Data- och Informationsteknik

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2020

Sammanfattning

Produkter och tjänster följer vanligen en linjär ekonomi i dagens samhälle. Produktens livscykel innebär att den köps in, förbrukas och sedan slängs därefter. Den här rapporten behandlar utvecklingen av en webbapplikation vid namn ShareConnect, vars avsikt är att utmana den linjära ekonomin. Med en markanadsplats, som tilllåter användare annonsera ut och skänka begagnade varor och tjänster, främjar applikationen en cirkulär ekonomi och noll-avfall. Utvecklingen har skett tillsammans med företaget Ericsson. Syftet med projektet var att vidareutveckla en befintlig plattform genom att tillföra mer funktionalitet, anpassa plattformen till mobila enheter samt att tillämpa konceptet spelfiering. Tillämpningen av spelifiering innebar att införa spelelement vars syfte var att motivera och uppmana till användande av webbapplikationen. Projektet inleddes med genomförandet av en analys för att ta reda på relevanta spelelement i den aktuella kontexten, som sedan implementerades. Projektet resulterade i en webbapplikation vars användargränssnitt utvecklades med JavaScript-biblioteket React samt med en serversida utvecklad med JavaScript-biblioteket Express.js, vars databashanteringsystem hanteras av MongoDB. Resultatet medförde även ett flertal nya funktioner till webbapplikationen samt att gränssnittet anpassades för flera sorters enheter.

Nyckelord: Cirkulär ekonomi, noll-avfall, spelifiering, React, Express.js, MongoDB

Abstract

Today, products and services commonly follow a linear economy. The basic life cycle of a product starts with it being purchased, then consumed only to finally be thrown away. This thesis details the development of a web application called ShareConnect, which purpose is to challenge the linear economy. With the implementation of a market place and allowing users to advertise and donate used products, the application will encourage the idea of a circular economy and zero waste. The web application was developed together with Ericsson. The purpose of the project was to further develop an existing system by adding more functionality, adapt the application for mobile devices and apply the concept called gamification. The implementation of gamification implies adding game elements with the purpose to motivate and exhort usage of the Web application. The project was initiated by performing an analysis to figure out relevant game elements that potentially could be implemented. The project resulted in a Web application with a user interface developed using React and a server side developed with the JavaScript-framework ExpressJS, whose database system is managed by MongoDB. The result also brought a number of new features to the Web application and the user interface was adapted to multiple devices.

Keywords: Circular economy, zero-waste, gamification, React, Express.js, MongoDB

Förord

Examensarbetet har utförts vid företaget Ericsson via Chalmers Tekniska Högskola och utgörs av 15 högskolepoäng. Arbetet är genomfört under dataingenjörsprogrammet som omfattar 180 högskolepoäng.

Vi vill tacka Rikard Mattsson, vår handledare på Ericsson, för den tid han ägnat oss inför och under examensarbetet. Rikard har assisterat med teknisk expertis och väglätt vårt arbete. Vi tackar även företaget Ericsson för utlåning av hårdvara och ett varmt mottagande under arbetet.

Slutligen vill vi dessutom tacka vår handledare Sakib Sistek från Chalmers Tekniska Högskola. Sakib har motiverat och stöttat oss genom hela vårt examensarbete. Hans hjälpsamma personlighet är en ständig inspiration.

Niklas Paasonen & Lucas Rosvall, Göteborg, maj 2020

Terminologi

- **Spelifiering** - Användandet av spelelement och spelprinciper i en icke traditionell spelmiljö. Tekniken används för att öka en användares engagemang genom att utnyttja människans behov att vilja uppnå mål, socialisera och tävla mot andra.

- **API** - Förkortning för Application Programming Interface och är ett gränssnitt för att kommunicera mellan program.

- **Ramverk** - Ett ramverk är en form av hjälpbibliotek med funktionalitet som förbättrar och underlättar användningen av programspråk.

- **NoSQL** - En databasstruktur avsedd för dokument, exempelvis på JSON-format.

- **Agil systemutveckling** - Ett arbetssätt som bedrivs iterativt där fungerande delleveranser av funktionalitet sker löpande enligt ett schema. Regelbundna möten sker mellan utvecklare och mottagare.

- **CSS** - En förkortning för Cascade Styling Sheet och används inom webbutveckling för att forma utseendet för en webbsida.

- **HTTP** - En förkortning för Hypertext Transfer Protocol.

- **IDE** - Integrerad utvecklingsmiljö för mjukvaruutveckling.

- **JSON** - Förkortning för JavaScript Object Notation, och är ett kompakt och textbaserat format som används vid datautbyte.

- **Backend** - Innefattar bakomliggande system såsom servrar och databaser, det vill säga det som sker bakom kulisserna för en applikation.

- **Frontend** - Innefattar det gränssnitt som användaren kommunicerar och interagerar med på webbplatsen, det vill säga den kod som exekveras på klientsidan i webbläsaren.

Innehåll

Kapitel 1

Inledning

1.1 Introduktion

Idag, i vårt moderna samhälle, följer en stor del av alla produkter en linjär ekonomi, där varje produkt har ett start- såväl som ett slutdatum. En produkt köps, används under en period, och sedan slängs. Det finns däremot tjänster som gör att ekonomin blir något mer cirkulär där produkter kan nå flera konsumenter innan de kasseras eller slängs. Detta är tjänster såsom Tradera, Blocket och Hygglo.

Det som de här tjänster dock saknar är en positiv respons när man utfört en god handling när man säljer eller köper en produkt. Dessutom saknar de här digitala marknadsplatserna valet att skänka bort alldagliga produkter gratis, alltifrån kläder till frukt som annars skulle slängas. Företaget Ericsson anser att detta kan ändras på genom införandet av ny funktionalitet och spelifiering till ett projekt som kallas ShareConnect. Ericsson önskar att konceptet spelifiering studeras och att olika spelelement implementeras till en existerande prototyp av webbapplikationen. Genom ShareConnect skall användare få positiv respons för sina goda handlingar och det skall var ett enkelt sätt att hitta en annan individ som är villig att ta emot produkter som inte längre behövs.

1.2 Syfte

Projektets syfte är att vidareutveckla en befintlig prototyp av en webbapplikation för att främja hållbarhet med inriktningen noll-avfall (zero-waste) och cirkulär ekonomi. Webbapplikationen skall förenkla och tilltala personer att dela och låna ut produkter som kan komma någon annan till nytta. Detta skall dessutom göras genom att implementera spelifiering för motivera användare till användande av plattformen.

1.3 Mål

Målet med projektet är att utveckla den befintliga webbapplikationen med en webbanpassad användarsida och en serversida, samt att implementera konceptet spelifiering. Webbapplikationen skall tillämpa spelelement såsom poäng, prestationer och poängtavlor. Applikationen skall även möjliggöra publicering av annonser på en marknadsplats och låta användare skapa egna användarkonton. Marknadsplatsen skall tillhandahålla alla tillgängliga annonser samt låta användaren söka bland dem och filtrera resultaten efter distans, kategori och publiceringsdatum. Ytterligare funktionalitet som systemet skall tillhandahålla är verifikationsmejl vid skapande av ett nytt konto samt möjlighet att återställa ett glömt lösenord. Till sist skall applikationens användargränssnitt vara fungerande och anpassat för mobilenheter.

1.4 Frågeställningar

Nedan beskrivs de frågeställningar som skall besvaras i rapporten.

- Hur bör plattformen utformas för att främja en cirkulär ekonomi och noll-avfall?
- Hur bör spelifiering implementeras för att främja användande?

1.5 Avgränsningar

Vårt projekt behandlar spelifiering och hur konceptet bör implementeras för att att främja användande av plattformen. Dessutom beskrivs vidareutvecklingen av plattformens användargränssnitt och serversida, där ny funktionalitet även tillkommer. Vi frånsäger oss därför ansvar gällande plattformens säkerhet, då detta ej innefattar projektets syfte. Vi frånsäger oss också ansvaret gällande värderingen av etiska frågor och hur plattformen kan komma att missbrukas av andra användare. Vi frånsäger oss även ansvaret att utföra användartester och övriga undersökningar angående webbplattformens användarvänlighet och utseende på grund av tidsbrist. Vi kommer i stället koncentrera oss på utvecklingen av funktionalitet och tillämpningen av spelifiering.

Webbplattformen ShareConnect är ett projekt framtaget av Ericsson där flera olika delprojekt ingår. Rapporten beskriver endast delprojektet som inkluderar utvecklingen av marknadsplatsen, sidor anknutna till användarkontot samt sidor berörda av tillämpningen av spelifiering, som instruktionssidor. En funktion som redan finns tillgänglig i prototypen för plattformen är en meddelandefunktion. Utvecklingen av den här funktionen kommer ej beröras av rapporten.

Kapitel 2

Teori och Bakgrund

Det här kapitlet beskriver den inledande teorin där koncept såsom cirkulär ekonomi, spelifiering och ramverk tas upp.

2.1 Cirkulär ekonomi

Den cirkulära ekonomin är ett koncept som utmanar dagens ekonomiska struktur, den linjära ekonomin. I en linjär ekonomi så utvecklas produkter i syftet att de ska brukas för att sedan kasseras. En cirkulär ekonomi handlar däremot om att maximera en produkts livscykel, från inköp i leveranskedjan till produktens åtgång och liv därefter. För att sluta cirkeln i produktens livscykel bör den antingen återvinnas vid förbrukat tillstånd eller vara utvecklad för att hålla länge och brukas på många olika sätt. Ett exempel på en sådan produkt kan vara ett par jeans som tillverkats av miljövänliga material, använts under en längre period för att sedan omvandlas till exempelvis ett par shorts för att fylla ett annat syfte, genom att benen på jeansen kapas. Målet med den cirkulära ekonomin är att minska avfallen samt att vara mer sparsam om jordens resurser.[1]

2.2 Spelifiering

Spelifiering kan användas för att främja användande av en produkt eller tjänst genom att öka motivationen hos användaren. Genom att utnyttja den naturliga instinkten hos människan att vilja tävla, uppnå prestationer samt uppleva spänning och lycka kan spelelement implementeras i sammanhang där de i vanliga fall inte brukar förekomma. Det skulle i sin tur uppmuntra och öka användarens intresse. Att belöna fortsatt användning av en tjänst med exempelvis poäng och prestationsutmärkelser är vanliga spelelement som används för att höja intresset hos användaren. Att vidare låta användare jämföra dessa utmärkelser sinsemellan är också en vanlig teknik inom spelifiering, vilket knyter an till viljan att tävla och prestera bra och därmed ökar motivationen till tjänstens användande ytterligare.[2]

Idag kan implementation av spelifiering ses i flera olika sammanhang, inom allt från utbildning till affärslivet. Företag och andra aktörer använder spelifiering för att driva ett specifikt användarbeetende och därmed gynna deras varumärke eller tjänster. Inom utbildning för yngre elever har spelifiering börjat användas för att öka motivationen till lärande i till exempel matematik, där användning av datorer och spelsystem vanligtvis inte förekommer, vilket utmanar det traditionella sättet att lära ut.[2]

2.2.1 Motivation

Motivation är en viktig faktor inom spelifiering. Det beskrivs som önskan om att uträtta något och är grundpelaren för att driva fram det beteende som önskas av användare i ett spelifierat system. Motivation kan delas upp i två delar, inre och yttre motivation. Inre motivation definieras som den inre önskan att uträtta saker för att finna njutning och harmoni, som att exempelvis lägga ett pussel för att uppnå självtillfredsställelsen av att lösa det. Den yttre motivationen innebär att utföra en handling endast i hopp om att vinna en belöning. Exempel på sådana belöningar kan vara materiella, som priser och pengar, eller abstrakta, som beröm och socialt erkännande. Inom spelifiering är det användning av yttre motivation som ofta är tydligast, såsom utdelande av poäng och utmärkelser vid ett visst efterfrågat beteende. Att finna en balans mellan de två typerna av motivation medför däremot den effektivaste implementationen. Målet är att kunna motivera en användares igångsättning av en aktivitet med en yttre motivation, i hoppet om att användaren sedan finner ett inre värde i att uträtta aktiviteten. Då blir motivationen för användaren istället det inre välbehaget att utföra och delta i aktiviteten, vilket kan innebära ett längre deltagande i aktiviteten än om motivationen hela tiden endast varit extern. En individ som ska börja träna kan vara ett exempel på en sådan situation. Individen kan till en början vara motiverad till att uppnå de positiva resultaten som träningen i slutändan medför, men kan lätt förlora motivationen under tiden ifall aktiviteten inte driver individens inre motivation starkt nog. Då kan kan motivationen till att träna upphöra och individens personliga utveckling på det planet går helt förlorad.[3]

2.2.2 Spelartyper

Utöver definitionerna för inre och yttre motivation har motivation inom videospel delats upp i fyra modeller representerat som fyra olika spelartyper, där spelarens motivation kan anknytas till någon av dem, vilket beskrivs i artikeln *Game on: Engaging customers and employees through gamification*. Vanligt är dock att spelaren representerar fler än endast en spelartyp och att under lärandets och spelets gång kan skifta mellan typerna. Vid implementation av spelifiering kan de användare som deltar i den spelifierade upplevelsen ses som spelare. Det visar sig att även de här användarna vanligtvis förhåller sig till en viss spelartyp, med varierande nivåer av motivation och konkurrenskraft. De olika typerna av spelare motiveras även av olika kontexter och spelelement. Vid tillämpning av spelifiering bör spelemenent därför väljas utefter vilken typ av spelare som efterfrågas använda tjänsten eller produkten. De olika spelartyperna samt vad de motiveras av är:

- **Kämpar (Strivers)** - spelar för att vara engagerade i sin personliga utveckling. Att slå sitt personliga rekord är viktigt.
- **Mördare (Slayers)** - spelar för att bli bättre än andra. Ställning relativt till andra spelare och att vinna är viktigt.
- **Lärjungar (Scholars)** - spelar för att lära sig om spelet och den spelifierade upplevelsen. Att förstå och studera upplevelsen i sig är viktigt.
- **Societetslejon (Socialites)** - spelar för att nätverka, samarbeta och knyta vänskapsband med andra spelare. Att känna empati med och lära sig om andra spelare är viktigt. [4]

2.2.3 Spelelement i vardagen

I vardagen hittas spelelement i många olika kontexter. Några alldagliga exempel där vi hittar spelelement är i tjänsterna Habitica och KhanAcademy. Det som dessa två exempel har gemensamt är att de använder spelelement för att uppmuntra användande och öka aktiviteten.

Habitica är en applikation som uppmuntrar användare till att bygga nya vanor, såsom att börja träna varje dag. För att användaren skall kunna bygga nya vanor på ett roligt sätt har Habitica implementerat spelelement i flera olika kontexter. I Habitica har du en karaktär som du kan träna och förbättra genom att utföra dina vanor. Genom att fortsätta genomföra dina vanor kan du på så sätt förbättra din karaktär och köpa olika tillbehör till din spelare. Exempelvis kan du köpa olika rustningar, drakägg, sadlar m.m. som du sedan kan använda till olika saker. I Habitica kan du också skapa grupper så att du kan samarbeta med dina vänner för att kunna förbättra din karaktär snabbare.[5]

KhanAcademy är en utbildningstjänst med videoklipp och beskrivningar. Spelelement som KhanAcademy har implementerat är att användaren får poäng när han/hon svarar rätt på frågor som sedan uppmärksammas med glada ansikten eller symboler. Medan om användaren svarar fel inträffar en motsatt effekt. Användaren får därmed direkt respons på vad han/hon gjorde fel eller rätt, vilket skiljer sig från normal undervisning. Användaren kan också nå olika milstolpar som användaren uppmuntras att exempelvis dela med sig av på olika sociala medier.[6]

2.2.4 Spelifiering och hållbar utveckling

Spelifiering används också i flera olika kontexter för att främja en hållbar utveckling. Några exempel där spelifiering används för att främja hållbar utveckling är Recyclebank, WeSpire och OPower.

Recyclebank är ett företag som använder sig av spelifieringselement för att främja återvinning och andra miljövänliga vanor. Spelelement som Recyclebank använder sig av är framförallt poäng och belöningar. Poängen intjänas genom att göra miljövänliga val gällande exempelvis transport och välja mer miljövänliga produkter, men poängen kan också intjänas genom att göra quiz för att lära sig mer om hur man skall återvinna olika produkter. Användare kan till och med få poäng genom att läsa eller konsumera annan information gällande hållbar utveckling. De intjänade poängen kan därefter användas för att få rabatter på olika produkter och tjänster i det verkliga livet.[7]

WeSpire är en plattform utvecklad för medarbetarengagemang som skall stödja större organisationer att utforma, leverera och mäta fördelarna med positiva effektivitetsprogram inom områden såsom hållbarhet, företagets sociala ansvar och kulturinitiativ. Plattformen använder sig av spelifiering för att öka aktiviteten hos medarbetarna där användarna får poäng för varje aktivitet de deltar i. Tjänsten möjliggör också för medarbetare att samarbeta med övriga medarbetare och de kan även se grafer över sin egna personliga utveckling.[8]

OPower har som syfte att minska energiförbrukningen genom att förändra sättet som människor förbrukar energin. Detta gör OPower genom att arbeta med energiföretag som bidrar med data om hushållens förbrukning och som i sin tur används för att förändra hushållens beteende. Datan används till att jämföra ett specifikt hushålls förbrukning med grannar, vänner

och förbrukningsmål som hushållet satt upp. Detta skapar en form av tävling om vem som kan förbruka minst energi över en given tidsperiod och där ingen vill vara sämre än någon annan. OPower är ett exempel hur man kan kombinera spelifiering och beteendevetenskap för att främja ett mer hållbart beteende.[9]

2.3 Ramverk

Ett ramverk är ett form av hjälpbibliotek med funktionalitet som förbättrar och underlättar användning av programspråk. Det finns många alternativ av ramverk för utveckling av både server- och användarsida. I projektet används Express på serversidan och React på användarsida.

2.3.1 Express

Express är ett minimalt ramverk för serversidan baserat på Node.js http-modul och Connect-komponenter. Express tillhandahåller sätt att enkelt återanvända kod och tillhandahåller också en model-view-controller (MVC) struktur för webbapplikationer. Däremot behöver modellen (M) kompletteras med ytterligare ett databas-drivet bibliotek såsom MongoDB.

Express tillåter så kallade mellanprogram (middlewares) som möjliggör återanvändningen av kod. Den väsentliga definition av mellanprogram är att det är en funktion som är tre parametrar, förfrågan, respons och nästa. Ett mellanprogram är kod som körs innan en HTTP-förfrågan når sin önskade rutt. Det går att göra egna mellanprogram såväl som att använda mellanprogram framtagna av andra utvecklare. Ett exempel på en tredjeparts mellanprogram är "bodyparser" som formaterar om en förfrågans nyttolast till JSON. Resultat av formateringen läggs sedan i req.body-objektet som lagrar parametrar vid en förfrågan. Förfrågan skickas skickas sedan vidare till nästa mellanprogram eller rutt (route).

En rutt är en så kallad ändpunkt i REST-API:t. När en rutt definieras används syntaxen app.VERB() där VERB() är en HTTP-metod såsom GET, POST eller PUT. Förfrågningar från användargränssnittet skickas sedan till dessa ändpunkter för att spara och hämta information från serversidan.[10]

2.3.2 React

React är ett ramverk utvecklat av Facebook som togs fram för att lösa det problem vid utveckling av storskaliga användargränssnitt där data förändras under en livscykel. React löser problemet genom att ha en struktur uppbyggd av flera komponenter som sedan sammankopplas. Det som gör React unikt är att när data förändras i React, så kallade DOM-mutationer, beräknar React den minsta mängden uppdateringar som behövs genomföras och uppdaterar sedan dessa i applikationens faktiska okumentobjektmodell (DOM).

En komponent i React kan antingen vara definierad som en funktion eller som en klass. React-komponenter har i regel ett antal metoder som exempelvis beskriver hur en komponent blir insatt i dokumentobjektmodellen eller hur en komponent uppdateras. Några av dessa metoder är render(), constructor(), static getDerivedStateFromProps(), componentDidMount() och componentWillUnmount(). Däremot är det endast render-metoden som är ett måste för att det skall klassas som en komponent. I Figur 2.1 syns en bild på hur en

enkel React-komponent kan definieras.

En komponent i React har dessutom ett antal variabler som beskriver en komponents inställningar och tillstånd. Två av dessa variabler är props som skickas in från en förälderkomponent och state som är en komponents tillstånd och något som förändras under en komponents livscykel. Hur props-variabeln kan användas i en komponent kan även ses i Figur 2.1.[11]

```
1  import React from "react";
2
3  export default class SimpleComponent extends React.Component {
4    render() {
5      return <h1>Hello, {this.props.name}</h1>;
6    }
7  }
8
```

Figur 2.1: En enkel React-komponent

2.4 MongoDB

MongoDB är en dokumentorienterad NoSQL-databas som är framtaget i C++ vilket gör att applikationer med MongoDB kan köras i princip överallt. NoSQL-databaser skiljer sig från SQL-databaser på flera sätt. För det första så måste en relationsdatabas i SQL ha en unik identifierare som identifierar ett objekt, eftersom det annars blir omöjligt att referera till ett specifikt objekt. I relationsdatabaser kallas denna identifierare för primärnyckel (primary key). MongoDB har en liknande implementation där varje objekt måste ha en unik identifierare som kallas ”_id”. En annan skillnad är dessutom att i SQL använder tabeller, medan MongoDB använder sig av samlingar (collections). Skillnaden mellan dessa två är att samlingar i MongoDB inte kräver att de lagrade objekten skall vara lika. Fördelen med detta är att det leder till flexibilitet när man arbetar med data, nackdelen är däremot att det försvårar något vid frågor (queries) till databasen. Till sist så använder sig även MongoDB av BSON, vilket står för binary JSON. BSON är binärt (binary) och på grund av detta blir BSON lättare för en dator att behandla, men samtidigt tillhandahålls samma funktionalitet som JSON.[12]

2.4.1 BSON

Skillnaden mellan BSON och JSON är att BSON erbjuder bättre frågor och indexering genom att ta något mer diskutrymme. Nackdelen är däremot att mer diskutrymme används vilket kan ta plats från något annat, men detta blir sällan ett problem eftersom diskar är relativt billiga och kan därför lätt kompletteras med ytterligare hårdvara.

BSON kan även lagra alla typer av JSON-dokument, men det betyder inte att ett giltigt

BSON-dokument är ett giltigt JSON-dokument. Det här har dock ingen negativ påverkan med att använda BSON eftersom varje språk har sin egna driver som kan konvertera datan till programmeringspråkets egna dataformat. [12]

Kapitel 3

Metod

I det här kapitlet kommer det arbetssätt som använts under projektet att tas upp och förklaras. Kapitlet kommer också ta upp den kontinuerliga integrationen av kod, dessutom så diskuteras val av plattform och verktyg.

3.1 Arbetssätt

För att organisera arbetet i projektet har ett agilt arbetssätt använts. Varje vecka började med ett avstämningsmöte, via verktyget Microsoft Teams, där framsteg och eventuella problem från föregående veckan, samt mål för kommande vecka diskuterades och togs upp med produktägaren på Ericsson. I samband med dessa möten gavs en tydlig bild var projektet befann sig i planeringen och vad för slags åtgärder som krävdes för att justera för eventuella oförutsedda händelser. Det gav också möjlighet till simpel justering av avancemang från föregående vecka, eftersom nya uppdateringar endast rörde sig i takten av en vecka åt gången.

3.2 Kontinuerlig integration

Inför varje avstämningsmöte genomfördes en sammanslagning av kod med hjälp av versionshanteringsprogrammet Git. Det här gjordes för att minimera framtida konflikter i koden samt uppmärksamma buggar och liknande som kan uppstå vid en sammanslagning av kod. Vid behov genomfördes också sammanslagningar av kod under veckans gång. I och med att plattformen består av både en serversida såväl som klientsida har två olika Git-respositories använts. Alla Git-respositories lagrades på Ericsson Forge, som är Ericssons interna versionshanteringsplattform.

3.3 Val av plattform

Vid utveckling av webbapplikationer finns det flera olika valmöjligheter för val av plattform, där flera olika ramverk kan användas på såväl serversida som för klientsidan. För projektet valdes dock React och ExpressJS eftersom det redan fanns ett befintligt system som använde sig av dessa två ramverk. På så sätt kan vi maximera återanvändningen av kod, men ramverken har även andra fördelar. Med ExpressJS kan en applikation enkelt byggas ut genom att ytterligare funktionaliteter läggs till. Medan React är utmärkt för att utveckla komplexa applikationer och har dessutom en mycket stor gemenskap. React används också av stora företag såsom Facebook, Netflix och Dropbox.[13]

3.4 Val av verktyg

I det här avsnittet beskrivs de digitala verktyg som vi har valt att använda oss av under projektet.

3.4.1 Figma

Figma är ett digitalt verktyg för att designa idéer. Figma valdes eftersom det ger oss möjligheten att skissa prototyper hur det grafiska användargränssnittet skall se ut innan utvecklingen är påbörjad. Figma möjliggör också samarbete mellan olika utvecklare där flera individer kan arbeta på samma fil.

3.4.2 MongoDB

MongoDB är ett dokumentorienterat databasprogram och även ett av de största dataprogrammen inom NoSQL. MongoDB valdes dels på grund av att NoSQL är väldigt skalbart, vilket kan vara en fördel när man arbetar iterativt. Databasen kan därför enkelt byggas på under projektet gång. MongoDB valdes också på grund av att det befintliga systemet använde sig MongoDB.

3.4.3 MongoDB Compass

MongoDB Compass är ett verktyg för att hantera data som är lagrad i MongoDB. Verktyget tillhandahåller ett enkelt användargränssnitt och tillhandahåller flera verktyg som gör att man kan filtera, redigera, lägga till och ta bort data från databasen. Verktyget valdes för att underlätta användningen av MongoDB.

3.4.4 Visual Studio Code

Vid utveckling av applikationer behövs någon typ av utvecklingsmiljö. Visual Studio Code är en utvecklingsmiljö med stöd för flera olika programspråk och valdes eftersom det har mycket stöd för utveckling av Node js applikationer, samt verktyg för webbtekniker såsom React, HTML, SCSS, JSON med flera.

3.4.5 Postman

Postman är ett verktyg för testning av RESTful API:er. Postman valdes att användas eftersom det är ett enkelt verktyg för att konstruera förfrågning för att säkerställa serversidans funktionalitet. I Postman går det att skicka olika typer av förfrågningar såsom GET, POST, PUT och DELETE och därefter få ett svar tillbaka som kan användas för att verifiera funktionaliteten hos API:et.

3.4.6 Jest och Supertest

Jest är ett testningsramverk medan Supertest är en modul som underlättar testningen av HTTP-förfrågningar. Fördelen med Jest är att det är ett stort testningsramverk som används av flera stora företag, medan Supertest underlättar testningen eftersom du som testare bland annat inte behöver hålla reda på serverns portar eftersom en flyktig port används.

Kapitel 4

Analys

Det här kapitlet beskriver det befintliga systemet som fanns sedan tidigare vid projektstart. Kapitlet diskuterar även val av spelelement.

4.1 Analys av befintliga system

Vid projektstart finns en tidig prototyp av plattformen. Sedan tidigare kan en användare skapa ett konto, publicera och ta bort annonser, se utlagda annonser, lägga till profilbild och skapa en chat via sockets. Det befintliga systemet är utvecklat genom React för användargränssnittet och ExpressJS på serversidan. Följande brister anser vi finns med det befintliga systemet:

- Plattformen är inte anpassad för mobila enheter, utan endast lämplig för specifika datorskärmstorlekar.
- Marknadsplatsens omfång är endast inriktat mot bortskänkning av egenodlade äpplen, inga övriga tjänster eller produkter.
- Trots att plattformens användargränssnitt utvecklats i React är de olika webbsidorna inte bestående av komponenter och delkomponenter. Vi anser att det här försummar de fördelar som React erbjuder.

Vissa resurser från det befintliga systemet har vi möjligheten att använda oss av. Detta innefattar bland annat av en fungerande serversida baserat på det befintliga systemet. I det befintliga systemet är dessutom serversidan ansluten till databasprogrammet MongoDB och innehar flera olika metoder med förfrågningar för att kommunicera med databasen. Flera av dessa resurser kommer användas senare i projektet, däremot kommer majoriteten av dem kräva vidareutveckling för att tillhandahålla önskad funktionalitet.

4.2 Val av spelelement

ShareConnect skall använda sig av spelifiering för att skapa en underliggande motivation att fortsätta använda sig plattformen. Spelifiering är däremot något som kan implementeras på många olika sätt, där tillvägagångsättet i regel beror på i vilken kontext spelelemententen skall befinna sig i. Dessutom passar sällan ett spelelement för alla vilket ställer krav på att det finns kännedom om vem som kommer använda plattformen, vilket kunde ses i avsnitt 2.2.2, Spelartyper. Till sist är det viktigt att vara medveten om vilka beteenden som uppmuntras av spelelementen vid tillämpningen i olika sammanhang. Spelifieringen bör endast förekomma i de sammanhang där man vill uppmuntra till ett visst beteende eller en skapa en viss motivation.

I projektet bestämde vi oss för att implementera tre olika spelelement; poäng, poängtavlor och prestationer. Dessa spelelement har valts ut eftersom vi tror att de kommer passa bra in på plattformen och de är spelelement som vanligtvis är förekommande i de flesta spelsammanhang. De är därför lätta att förstå sig på vilket gör dem effektiva i ett sammanhang där användaren inte förväntar sig dess tillämpning. Dessutom baseras valet av spelelement på de förhandsstudier som gjorts om de olika spelartyperna. Se förklaring nedan:

- **Poäng** - Spelelementet poäng har valts på grund av sin mångsidighet och typiska närvaro i spelsammanhang. Det är väldigt sällan en spelifierad upplevelse tillämpas utan någon form av poäng. Vidare riktas samlandet av poäng mot de särskilda spelartyperna *Kämpar* och *Mördare*. Motivationen hos kämparna grundar sig i deras vilja att slå sina egna rekord och se personlig utveckling. Poäng är effektivt i detta sammanhang då det är ett konkret mått på hur bra användaren presterat, beroende på vad poängen mäter, samt att det är lätt att se förändring och framgång över tid. För spelartypen Mördare, som motiveras av att vinna och prestera bättre än andra spelare, är poängräkning i liknande syfte även effektivt då det är väldigt enkelt att jämföra olika spelares poäng sinsemellan.

- **Poängtavlor** - Valet att implementera poängtavlor riktar sig ytterligare mot spelartypen *Mördare*. Eftersom det är viktigt för de här spelarna att jämföra sin bedrift gentemot andra, med ändamålet att prestera bäst, behövs det en plats på plattformen där detta kan ske. Poängtavlor är även de vanligen förekommande i spelsammanhang som inkluderar fler än en spelare samt nyttjandet av poäng. Genom att ställa upp alla användares samlade typer av poäng i en nedåtgående lista kan det enkelt synas vem som har presterat bäst högst upp. Det här kommer motivera typen Mördare att sträva efter placering i toppen av poängtavlan, vilket i sin tur innebär ökad användning av de funktioner som belönas med poäng.

- **Prestationer** - Slutligen har prestationer valts som det sista spelelementet. Detta element innebär ett särskilt mål eller uppgift som skall utföras och klaras av för att uppnå prestationen. Likt poäng riktar sig detta element mot spelartypen *Kämpar*. Poäng som endast samlas utan ett konkret mål, annat än i syftet att få fler poäng än andra användare, kan för Kämpar upplevas grundlöst. Det är som tidigare nämnt enkelt att se sina poäng öka med tiden, vilket kan bringa tillfredsställelse hos Kämpar, men genom att införa delmål som prestationer kommer det att öka chansen till motivation. Uppnådda prestationer kommer därmed även de vara ett mått på personlig utveckling för den här spelartypen.

Valet av spelelement har främst riktats åt spelartyperna *Kämpar* och *Mördare* på grund av att dessa speltyper ger ett bättre gensvar på olika typer av spelelement. Däremot anser vi att webbapplikationens utformning och syfte även kan komma att locka även de andra spelartyperna. *Societetslejon* motiveras, som tidigare nämnt, av att nätverka, samarbeta och känna empati med andra spelare. De ovan nämnda spelelementen är väldigt konkurrensbetonade och riktas därför inte direkt mot den här spelartypen. Däremot är webbapplikationens syfte att främja hållbarhet och cirkulär ekonomi, vilket innebär en viss medmänsklig omtanke. Att dela med sig av begagnade saker och egna tjänster innebär ett samarbete mellan användarna, vilket även uppmanar kommunikation och bekantskap. Därför kan även *Societetslejon* komma att motiveras till användning av webbapplikationen anser vi. Dessutom

ser vi att även *Lärjungar* kan lockas av att använda plattformen eftersom spelelementen befinner sig i en kontext som de normalt inte hittas. På så sätt kan *Lärjungar* lockas till plattformen av ren nyfikenhet för att studera själv upplevelsen.

Kapitel 5

Genomförande av backend-applikation

Det här kapitlet beskriver utvecklingen av webbapplikationens backend-del. Kapitlet tar upp vidareutveckling av det befintliga systemet, samt implementationen av ny funktionalitet.

5.1 API-förfrågningar

Användargränssnittet använder sig av HTTP-förfrågningar för att kommunicera med backend-applikationen med syftet att antingen hämta, lägga till, ändra eller ta bort data från databasen.

När en förfrågan skickas till API:et tas den först emot av ett mellanprogram som validerar förfrågan och kollar om HTTP-förfrågans URL kräver autentisering eller inte. Om autentisering inte krävs godkänns förfrågan och går vidare till nästa mellanprogram eller rutt. Om autentisering däremot krävs kontrollerar mellanprogrammet om HTTP-förfrågan innehar en giltig autentiseringsnyckel. Om nyckeln är giltig skickas HTTP-förfrågan vidare till nästa mellanprogram eller rutt. Om nyckeln däremot är ogiltig nekas förfrågan och returnerar ett meddelande att förfrågan har blivit nekad. När en förfrågan når en ändpunkt eller rutt kan backend-applikationen börja utföra det efterfrågade arbetet. En rutt i Express kan defineras enligt nedan.

```
42  app.get("/", function (req, res) {
43    res.send("hello world");
44  });
45
```

Figur 5.1: En rutt som returnerar hello world

I projektet fanns det ett antal förfrågningar som kunde användas från det befintliga systemet. Majoriteten av dem behövdes däremot vidareutvecklas för att målen skulle kunna uppnås. Ett antal buggar fanns även i de befintliga HTTP-förfrågningarna. Exempelvis saknades en hel del validering vilket bland annat gjorde att flera användarkonton kunde skapas med samma mail-address, vilket skapade problem senare på plattformen.

5.2 Vidareutveckling av databasen

Den ursprungliga databasen bestod av de tre samlingar (collections) Users, MarketObjects och Conversations. För att uppnå vårt mål behövde dokumenten i dessa samlingar utvecklas med fler attribut, såsom sparade annonser, tidsstämplar, olika typer av poäng och så vidare. Ett antal nya samlingar behövdes också läggas till, såsom Locations och Achievements. Där Locations är en samling bestående av olika orter med tillhörande postnummer och koordinater och används för att kunna länka angivna orter till geografiska positioner. Detta använts i sin tur för att beräkna avståndet mellan två koordinater med hjälp av Haversine-formel (Se Bilaga L). Medan Achievements är en databas bestående av olika prestationer.

Det var även nödvändigt att implementera fler förfrågningar (queries) till MongoDB. Exempelvis för att kunna söka efter annonsobjekt som innehöll ett visst ord i dess beskrivning. För att kunna göra detta behövdes även flera text-index skapas för de olika attributen för att kunna stödja textfrågor efter ett visst stränginnehåll. Nedan syns ett exempel på textfråga till MongoDB.

```
db.marketobjects.find( { $text: { $search: "apples fruit green" } } )
```

5.3 Testning och verifikation

För att säkerställa funktionaliteten på backenden har tester genomförts med hjälp av applikationen Postman som är ett verktyg för testning av RESTful API:er. Ett test genomförs genom att skicka en begäran till en specifik ändpunkt, som sedan returnerar ett svar som används för verifikation. Det finns flera olika förfrågningsmetoder som går att använda där några av dem är POST, GET, DELETE och PUT. POST används för att lägga till information på servern. GET används för att hämta information. DELETE används för att ta bort information och PUT används till att uppdatera information på servern.

Testningsramverket Jest har också använts för testning av förfrågningar på backenden. Jest är ett mycket populärt ramverk för testning och det används bland annat av företag såsom Facebook, Twitter och Spotify. Tillsammans med Jest används som tidigare nämnt Supertest som är en modul som underlättar testning av HTTP-förfrågningar.

5.4 Nodemailer

För att kunna skicka information till användare när de inte är aktiva på plattformen används Nodemailer som är en modul för Node JS för att skicka mail. Genom Nodemailer kan ett transporter-objekt skapas som sedan kan användas för att skicka mail. För att kunna transportera meddelanden använder sig Nodemailer av SMTP (Simple Mail Transfer Protocol). Mail skickas till användaren vid flera tillfällen, bland annat när användaren registrerar sig och tar bort sitt konto.

5.5 Verifiering av användarkonton

Ett av målen var att nya användare skulle behöva verifiera sitt konto innan användaren skulle få tillgång till plattformen. Målet löstes genom användningen av verifieringsnycklar och att den nya ändpunkten (‘/confirmation’) sattes upp för att signera nycklar. Verifieringsnycklarna genereras och signeras med hjälp av JSON Web Token (JWT). För att överlämna verifieringsnyckelarna används i sin tur Nodemailer, som tidigare nämnt är en Node JS modul för att skicka mail.

Figur 5.2: Ett verifieringsmail som skickas vid registering

5.6 Geokodning och platsinformation

För att möjliggöra platssökningarna och funktionalitet som att visa avståndet till olika marknadsobjektet givet en adress eller orter behövs geokodning för länka en adress eller ort till en koordinat. För att lösa detta problem har två lösningar implementerats. För det första så har platsdatabasen blivit initierad med data bestående av lite mer än 1600 av de större orterna i Sverige med tillhörande koordinater och postnummer. Den här datan är ursprungligen hämtad från Geonames.org där den kan hämtas i ett textfil-format. För att sedan infoga datan till MongoDB fick därför ett skript skrivas för att föra över informationen. För att sedan också kunna bygga ut platsdatabasen med mer platsinformation har det externa API:et Nominatim som är framtaget av OpenStreetMap använts. Med hjälp av API:et kan man givet en adress eller ort hämta koordinater för en specifik plats. För att använda API:et behövdes dock först vissa krav tillgodoses. API:et hade bland annat ett krav att endast en förfrågan fick skickas per sekund. För att uppfylla detta krav implementerades en kö för att schemalägga hämtningen så att endast en förfrågan skickas per sekund. För att minimera antalet hämtningar sparas all hämtad platsinformation sedan i MongoDB. På så sätt kommer platsdatabasen växa med fler och fler orter i takt med att plattformen används.

Kapitel 6

Genomförande av frontend-applikation

I detta kapitel beskrivs utvecklingen av applikationens frontend-del. Kapitlet inleds med beskrivningen av hur arbetet inleddes, för att sedan beskriva hur JavaSctipt-biblioteket React tillämpats och hur användargränssnittet har utformats.

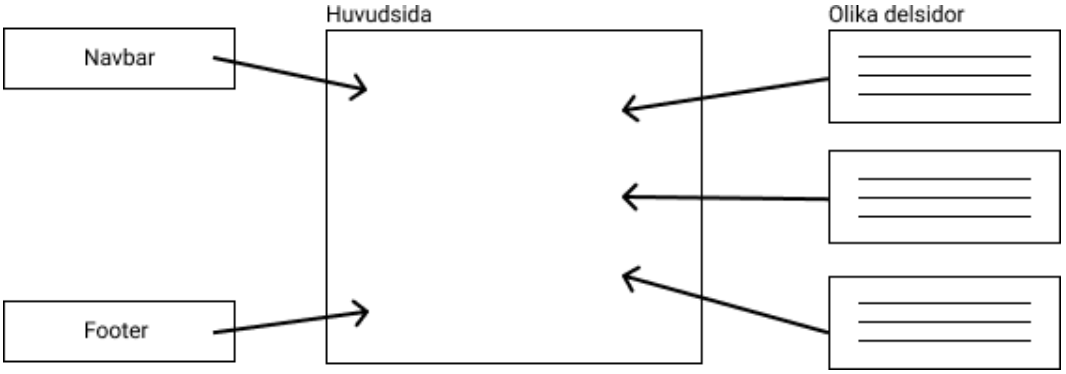
6.1 Skisser

I projektets första stadie undersöktes och skissades det på olika varianter av plattformens utseende. Dessa skisser gjordes digitalt i designverktyget Figma. Skisserna över plattformen eftersträvades att hålla en hög kvalitet, då det är väldigt användbart att ha en färdig och noggrann skiss över webbapplikationens grafiska upplägg och utseende vid framställning i programkod. Minst en skiss för varje delsida togs fram. Till en början togs en utformning fram för applikationens allomfattande, grafiska gränssnitt, så som ett färgtema och en generell struktur för varje delsida, vilket gjorde det enklare att färdigställa skisser på kvarstående delsidor. Två exempel på en skisser som gjordes kan ses i Bilaga A och B.

6.2 React

Vid utveckling av React-applikationer används som tidigare nämnt komponenter. Den här komponentbaserade uppbyggnaden av webbsidor gör att React skiljer sig från traditionell webbutveckling. Istället för att ett HTML-dokument representerar varje enskild delsida på en webbplats så möjliggör React tillämpningen av endast en enda HTML-sida. Den här enskilda sidan blir huvudsidan som istället innehåller en mängd delkomponenter som representerar webbplatsens delsidor. Dessa komponenter utvecklas separat och sammanfogas därefter till huvudsidan, som då representerar hela applikationens användargränssnitt. Den här huvudsidan kallas även för förälderkomponent, eftersom den är ursprungsföräldern till alla delkomponenter. Det här innebär att varje delsida i webbapplikationen representeras som en egen komponent, men varje delsida i sig kan även delas upp i komponenter om nämnd delsida innefattar mer komplex funktionalitet. Varje komponent utgörs av en separat JavaScript-fil. Den här typen av uppdelning av varje delsida medför enkelt underhåll och utvidgande av kodplattformen. Det är även lätt att redigera varje enskild komponent för sig istället för att behöva leta igenom en massiv fil innehållande alla komponenter samtidigt. Se figur nedan för ett exempel på den fil- och komponentstruktur som använts i projektet. Ett exempel på delsidor som använder sig av flertalet, uppdelade komponenter är marknadssidan och profilsidan. De här sidorna innehåller funktioner som behöver arbeta oberoende varandra. Genom att dela upp de här funktionerna i separata komponenter innebär även

det en enklare överblick och struktur över var de här funktionerna hanteras och är implementerade i kodbasen.



Figur 6.1: Huvudsida tillsammans med olika delsidor

Vidare så medför den komponentaserade utvecklingen av webbapplikationen, utöver att skapa egna komponenter, att det enkelt går att implementera redan färdiga komponenter som finns att implementera från olika komponentbibliotek. Bootstrap-React är ett bibliotek med många olika komponenter som utan kostnad går att implementera.[14] Exempel på komponenter som tillämpats från Bootstrap-React är inmatningsfält, knappar, rullgardinslistor och flikmenyer. Ytterligare en komponent som implementerats från ett externt bibliotek är meddelandemodulen som dyker upp vid olika tillfällen i applikationen. Den här komponenten heter Swal och har finns i biblioteket Sweet Alert 2.[15]

För att sedan tillåta applikationen vara mer än en enkelsidig webbsida baserat på endast ett HTML-dokument används Reacts rutt-bibliotek, React Router, som gör det möjligt att navigera mellan huvudsidans alla delkomponenter. Resultatet upplevs därmed som en vanlig webbplats, med fördelen att webbläsaren inte behöver ladda om varje ny sida i applikationen utan bara växlar mellan redan hämtade komponenter vid navigering.

6.3 Användargränssnittets utseende

För att ge webbapplikationens användargränssnitt ett lockande utseende, utöver dess funktionalitet, behövde webbsidorna ges en anpassad stil, såsom teckensnitt, färger, avstånd och upplägg. Detta gjordes med hjälp av Cascading Style Sheets (CSS), en mekanism som tillåter personlig stilgivning och arrangemang av HTML-element på en webbsida. Redan färdiga komponenter, som nämnt ovan i avsnitt 6.2, importerades och användes även för att ge webbapplikationen ett stärkt utförande. De skisser som fastställts under projektets inledande fas efterföljdes vid utformningen av användargränssnittets utseende och stil.

6.4 Bildhantering och lokal lagring

Annonsobjekten i webbapplikationen innehåller i de flesta fall bilder som behöver hämtas från serversidan, där de lagras i ett filsystem, för att sedan visas upp på klientsidan. För att maximera prestandan och minimera väntetider bestämdes det att dessa bildfiler skulle sparas

lokalt i användarens webbsession. Detta görs genom att bilden konverteras med Base64 till en teckensträng. Base64 innebär en översättning från bildfilens originalformat, en binär datasträng, till en sträng av ASCII-tecken. Det är fördelaktigt vid sändning av bildfilen via datapaket mellan klientsidan och serversidan, samt vid lagring i databasen.[16] Den nya, konverterade bilddatan kan därefter hämtas igen vid behov och konverteras tillbaka till en bild igen. Andra variabler som även sparas lokalt i webbsessionen är användarens användarkonto och angiven platsinformation. Användarkontot sparas för att användaren skall fortsättas vara inloggad när en webbsida laddas om på nytt, medan platsinfomationen sparas för öka användarupplevelelsen.

Kapitel 7

Systemkonstruktion

Det här kapitlet kommer ta upp hur spelfiering har implementerats på plattformen. Dessutom kommer plattformens olika delar att tas upp i detta kapitlet. Följande sektioner har plattformen delats upp i; marknad, profilsida, inställningssida, poängtavlor och övriga sidor.

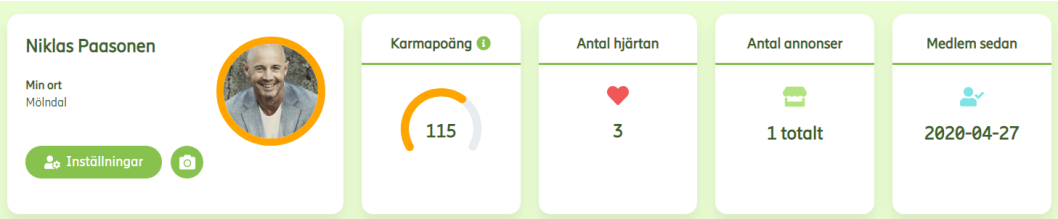
7.1 Spelifiering

Konceptet spelifiering har tillämpats i webbapplikationen baserat på den analys som gjordes i början av projektet gällande spelifiering, spelartyper och motivation. Spelelementen som valts att tillämpa är som ovan nämnt poäng, prestationer och poängtavlor. Dessa spelelement har valts ut eftersom de anses passa till plattformens sammanhang utan att försvåra det huvudsakliga användningsområdet, vilket är att interagera användare sinsemellan samt med annonser på marknadsplatsen.

7.1.1 Poäng

En användare kan samla två olika typer av poäng på plattformen, hjärtan och karmapoäng. Dessa två typer av poäng skiljer sig från varandra eftersom de intjänas på olika sätt.

- **Hjärtan** intjänas genom att att en annan användare uppmärksammar och gillar en av dina annonser. Då finns möjligheten att skicka ett hjärta samt lämna en hälsning som uppskattning för annonsen och ett hjärta adderas till sammanställningen på profilsidan (se figur nedan). På så sätt skapas ytterligare en interaktion mellan olika användare, samtidigt som det finns en uppmuntran att lägga ut bra annonser. Varför namnet hjärtan gavs till poängtypen härstammar från att ett hjärta associeras med kärlek och omtanke, vilket även är det som funktionen symboliserar. Hjärtan är även en poängtyp som vi tror kan motivera spelartypen *Societetslejon* eftersom denna poängtyp skapar mer interaktion mellan olika användare.
- **Karmapoäng** är den andra typen av poäng och intjänas genom att användaren utför en uppsättning av handlingar. Dessa handlingar inkluderar att lägga ut en annons, uppnå en prestation, skänka ett hjärta till en användare, motta ett hjärta och att bli kontaktad av en användare angående en utlagd annons. Därmed blir karmapoängen ett mått på hur flitigt användaren utfört det olika handlingarna och utforskat plattformens funktioner. Dessa funktioner är tänkta att förknippas med hållbar utveckling och miljö, vilket medför att karmapoängen även kan ses som ett mått på hur mycket användaren bidrar med för att främja just detta. Därav härstammar även namnet karmapoäng, då handlingarna som medför dessa poäng skulle bidra till en ökad personlig karma. Användarens insamlade Karmapoäng visas även de upp på profilsidan enligt figur nedan.

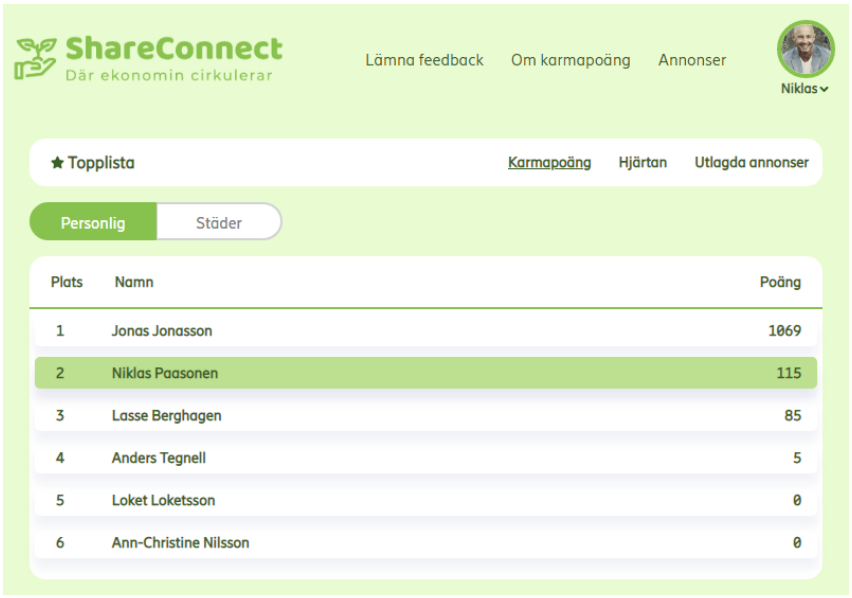


Figur 7.1: Vy över poängtyperna Karmapoäng och Hjärtan på användarens profilsida.

7.1.2 Poängtavlor

Poängtavlor har också implementerats för att kunna visa upp de olika typerna av poäng. Varje poängtavla har två olika tillstånd där jämförelsen antingen sker via olika spelare eller orter. Utgångstanken var först att poängtavlan endast skulle utgöras av användare, men en poängtavla med orter utvecklas senare för att tillföra ett annat spelelement där användare också kan samarbeta med varandra för att ta sin ort högre upp på poängtavlan.

Det finns tre olika poängtävlor som implementerats som visar olika typer av poäng; karmapoäng, hjärtan och antalet utlagda annonser. Poängtävlorna kan även väljas att jämföra individer eller orter med varandra. På så sätt kan användare tävla med varandra såväl som att samarbeta och tävla med varandra mot andra orter. I figuren nedan syns den implementerade sidan för topplistorna.

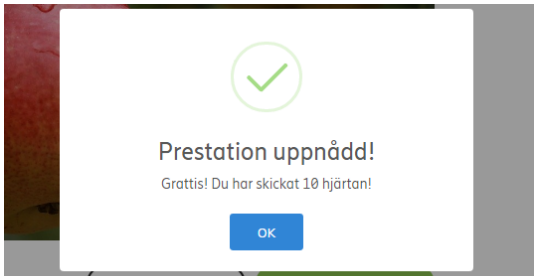


Figur 7.2: Vy över sidan för de olika topplistorna. I figurens syns den aktiva topplistan för Karmapoäng, där den inloggade användaren är markerad i listan.

7.1.3 Prestationer

Användaren kan även uppnå prestationer eller utmärkelser som mäter hur flitigt och konsekvent en viss funktion har använts. En sådan funktion kan till exempel vara att lägga ut en annons på marknaden. Prestationerna delas in i kategorier för vardera funktion och uppnås när användaren använt funktionen ett visst antal gånger. Det finns tre nivåer för varje kategori, brons, silver och guld, därmed tre prestationer med stigande antal gånger som funktionen behöver ha utförts för att nå målet. Bronsnivån har lättast mål att uppnå och guldnivån det svåraste. Som exempel är prestationerna som kan uppnås i kategorin *utlagda annonser på marknaden* är tre stycken för brons, tio stycken för silver och slutligen 50 stycken annonser för att uppnå guldprestationen. Prestationer implementerades eftersom de innebär ett delmål för användaren på plattformen och ska driva motivationen och viljan att använda funktionerna som är efterfrågade. De prestationer som finns att uppnå finns listade på användarens profilsida under fliken märkt *Prestationer*. Se Figur 7.3 nedan. När användaren utför en handling som leder till att en prestation uppnås visas textrutan, som beskrivs i Figur 7.4, upp för att gratulera och uppmuntra användaren. Därefter kommer den uppnådda prestationen att färgläggas i prestationslistan på användarens profilsida, vilket därmed markerar att den är vunnen.

Figur 7.3: Vy på profilsidan över de olika prestationerna som går att uppnå samt kriterierna för dem. De prestationer som användaren har uppnått är färglagda.



Figur 7.4: Figuren beskriver den textruta som dyker upp när användaren utfört en handling som leder till en uppnådd prestation. I det här fallet har användaren åstadkommit prestationen att skicka tio hjärtan.

7.2 Marknaden

Marknaden utgörs av tre delsidor. En sida för att skapa en ny annons, en sida för att visa upp alla annonser på plattformen och en sida för att visa upp en enskild annons, även kallad marknadsobjektsida.

7.2.1 Marknadsplatsen

Marknadsplatsen visar upp alla annonser och tillhandahåller sökfunktionalitet såsom sökning via text efter annonser som innehåller specifika ord i antingen titeln eller beskrivningen. Det finns också möjligheten att filtrera efter kategorier, distans och annonstyper. Om en användare exempelvis bara vill se annonser inom kategorin "Frukt och grönt" kan detta filtreras.

Marknadsplatsen tillhandahåller även sökning baserat på platsinformation. För att söka annonser baserat på plats finns ett antal alternativ. Användaren kan välja att söka efter annonser baserat på sin angivna ort i inställningssidan, nuvarande plats som hämtas via HTML Geolocation API eller ange en specifik ort genom att söka efter den. Om användaren söker efter en specifik plats skickas en förfrågan till serversidan som försöker matcha den angivna platsen med en plats i platsdatabasen, om en plats hittas returneras den med tillhörande koordinater. Haversines formel (Se Bilaga L) används sedan för att beräkna avståndet till de olika annonserna. En bild över gränssnittet för att söka efter en plats kan ses i Bilaga F.

När en sökning efter annonser har genomförts visas de aktuella annonserna upp på marknadsplatsen. Användaren har därefter möjligheten till ytterligare sortering genom att sortera efter distans eller nyligen upplagda annonser.

För att hämta alla annonser behöver marknadsplatsen skickas många förfrågningar för att hämta relaterade bilder, vilket gör att renderingen kan ta lång tid. För att förhindra detta har paginering (pagination) implementerats så att endast ett specifikt antal objekt hämtas åt gången. Detta förbättrar prestandan på plattformen. Dessutom sparas objekt i webbläsarens session efter att de hämtas från serversidan för att minska antalet förfrågningar som skickas under en session, vilket ökar prestandan ytterligare. En bild på marknadsplatsen kan ses i Bilaga C.

7.2.2 Marknadsobjekt

Marknadsobjektssidan visar upp all information om en specifik annonsen. Sidan innehåller också en karta där avhämtningsorten markeras ut. Dessutom tillhandahåller även denna sidan en rad funktionalitet, såsom att spara annonsen, kontakta annonsören och lämna ett hjärta.

Om en användare sparar annonsen kommer användaren kunna hitta denna i sin profilsida under Sparade annonser, så att användaren lätt kan komma tillbaka till en specifik annons igen. Användaren kan också kontakta annonsören och då skapas en chat där de två parterna kan diskutera om avhämtning eller liknande. Till sist, så kan användare också lämna ett hjärta och hälsning till annonsören som är ett sätt att visa uppskattning. Om det önskas kan användaren också att välja att lämna ytterligare en kommentar tillsammans

med hjärtat. En bild på marknadsobjektsidan kan ses i Bilaga E.

7.2.3 Skapa annons

Sidan för att skapa en annons är uppbyggd av ett enkelt HTML-formulär där användaren fyller i information om sitt annonsobjekt. Informationen som efterfrågas är titel, beskrivning, kategori, typ av annons och plats. Det finns även en möjligheten att ladda upp en bild för annonsen. Däremot är en bild inget krav, eftersom vi vill göra det så lätt som möjligt för användaren att skapa en annons. Det här är speciellt för dem som funderar på att skänka bort en produkt, eftersom vi då vill minska tröskeln så mycket som möjligt. Skapa annons sidan kan ses i Bilaga D.

7.3 Profilsida

Profilsidan på plattformen kan jämföras med en dashboard, där användaren får en sammanställning över sitt konto. På profilsidan kan en användare se sin kontoinformation, mottagna hälsningar, sina prestationer, aktiva- och inaktiva annonser, samt sparade annonser. Användaren kan dessutom se och lägga upp en profilbild på profilsidan. Profilsidan kan ses i Bilaga I.

- **Mottagna hälsningar** innefattar de gillningar eller hjärtan som användaren tagit emot från sina annonser.
- **Prestationer** En användare kan uppnå flera prestationer på plattformen när de utför olika handlingar.
- **Inaktiva och aktiva annonser** inkluderar de annonser som användaren har lagt ut på marknadsplatsen, men det är endast de aktiva annonserna som en annan användare kan se. Annonser kan göras inaktiva eller aktiva oberoende vilket som är dess nuvarande tillstånd. Såväl inaktiva och aktiva annonser kan även tas bort helt från plattformen om detta önskas.
- **Sparade annonser** är annonser som användare sparat från marknadsplatsen och denna funktionalitet skall fungera som ett sätt att snabbt komma tillbaka till annonser som användaren gillade. Det är dock endast aktiva annonser som kommer visas under denna komponent. Om en annons är inaktiva kommer den först visas igen när den blir aktiv.

7.4 Inställningssida

På inställningssidan kan användaren hantera sin konto. Funktioner som tillhandahålls är att byta kontoinformation såsom ort och postnummer. Användaren har också möjligheten att byta lösenord och ta bort sitt konto.

7.5 Poängtavlor

Det finns tre olika poängtavlor på plattformen som visar upp de olika poängen som en användare kan samla in. En poängtavla visar upp de 10 användarna med mest poäng,

samt den enskilda inloggade spelaren med sin placering om hen inte ingår i topp 10. Varje poängtavla har två tillstånd där antingen enskilda individer eller orter kan jämföras.

7.6 Övriga sidor

Det finns ett antal andra sidor som också utvecklats. Det här är sidor för såsom skapa konto, logga in, glömt lösenord och informationssidor. Skapa konto och logga in utgörs av två moduler. När en användare skapar ett konto skickas ett verifieringsmejl till användaren som måste bekräftas. Användaren kan sedan logga in på sidan. Glömt lösenord sidan kan användaren komma åt från logga in modulen, den består av ett enkelt formulär där användaren anger sin mejladress. Ett nytt lösenord skickas därefter till den angivna mejladressen. Se Bilaga H för en vy över gränssnittet vid återställning av lösenord. Slutligen har det även implementerats en sida som beskriver hur karmapoäng fungerar och hur en användare kan tjäna in dem, se Bilaga K, samt en sida där användare kan skicka in återkoppling och förbättringsförslag angående applikationens gränssnitt och funktioner. Se Bilaga J.

Kapitel 8

Resultat

Projektet resulterade i en webbapplikation som tillhandahåller en grafisk, webbanpassad användarsida som kommunicerar med en serversida. Användarsidan och gränssnittet utvecklades med Javascript-biblioteket React och serversidan med Express.js. Konceptet spelifiering har även tillämpats och spelelement som används i tillämpningen är poäng, prestationer och poängtavlor. Spelelementen nås och kan upplevas på användarens profilsida. Dessutom implementerades en marknadsplats i applikationen där användare kan visa och lägga ut annonser. För övriga webbsidor som implementerades i projektet se kapitel 7, Systemkonstruktion. Resultatet innebär således att målen för projektet möttes.

- Spelifiering har implementerats med spelelementen såsom poäng, prestationer och poängtavlor.
- En fungerande marknadsplats har utvecklats med filtrering- och sorteringsfunktionalitet.
- Efterfrågade sidor med tillhörande funktioner har framtagits.
- Plattformen är fungerande och anpassad för mobila enheter.

Kapitel 9

Diskussion

9.1 Kritisk diskussion

När projektet startade tog vi över ett befintligt system som redan hade en fungerande frontend- och backend-applikation. Detta innebar i sin tur att mycket tid i början av projektet fick läggas ner på att förstå den nuvarande koden och det befintliga systemets uppbyggnad. Den befintliga backendapplikationen var relativt lätt att förstå och där gick inlärningsprocessen relativt snabbt, vilket delvis berodde på att vi hade tidigare erfarenheter att arbeta med ExpressJS. Frontend-applikationen var däremot svårare på flera sätt. För det första var React ett ramverk som vi inte hade använt tidigare. Dessutom var det väldigt svårt att förstå vilka CSS-klasser som resulterade i vilket utseende eftersom varje element hade mellan 5-9 olika CSS-klasser. Detta ansåg vi i sin tur gjorde koden relativt svårläst. I början av projektet försökte vi använda oss av denna tidigare struktur som de hade använts sig av på frontend, men sedan bestämde vi oss för att göra om all CSS. Detta visade sig senare vara ett bra val och nu i efterhand borde vi nog gjort om all CSS redan från start. Inom vissa aspekter hade det till och med varit en bra idé att börja om helt från början när det gäller frontend-applikationen, eftersom vi fick lägga ner en hel del tid att förstå strukturen och rätta till fel i koden. Däremot skulle det innebära att vi skulle gått miste om funktionalitet som vi hade kunnat använda oss av, så båda alternativen har sina fördelar och nackdelar.

Då projektet omfattats av både backend- och frontend-applikation har detta också medfört vissa följder. Det har dels varit ett sätt att lätt kunna separera delar av funktionaliteten och användargränssnitt men ibland har det också varit svårt att veta vilken del som skall göra vad såsom vid filtrering, där vi vid flera tillfällen fått göra en avvägning om man bör minimera antalet förfrågningar och sköta mer på frontenden eller skicka fler förfrågningar och endast sköta renderingen på frontend-applikationen. Att systemet även utgjordes av två applikationer (en backend och en frontend) har även försvårat något vid testning av hela systemet. Ett antal automatiserade tester skrevs i början av projektet på backenden, men i takt med att projektet växte kände vi att vi behövde fokusera mer på plattformens funktionalitet. Vi gick då över mer till att använda oss av Postman, som gjorde testning mindre tidskrävande. I projektets början fanns även tanken att skriva automatiserade Seleniumtester för applikationen men eftersom vi kände att funktionaliteten skulle behöva begränsas ytterligare om testerna hade genomförts bestämde vi oss för att det inte skulle genomföras.

Syftet med tillämpningen av spelfiering var att öka engagemanget på plattformen för att främja en cirkulär ekonomi. Genom att försöka ge användaren tillfredsställelse och en känsla av att en bra gärning har utförts skulle motivationen vara större för att göra om denna handling. Om spelelementen verkligen ger denna effekt är däremot svårt att svara på eftersom spelelement kan fungera olika bra i olika typer av kontexter. Utan tester på riktiga användare går det inte att fastställa bevis för att de valda spelelementen var passande i

just den här tillämpningen. Däremot var, som tidigare nämnt i avsnitt 1.5 Avgränsningar, sådana användartester uteslutna ur projektets omfång med anledning av tidsbrist.

9.2 Användarvänlighet

Plattformen har utformats för att utseendet och funktioner skall vara enkla att förstå och lära sig att använda. Alla sidor och delsidor på plattformen har också blivit anpassade för att fungera väl på mobila enheter. Varje sida har tagits fram efter ”3 click rulesom lyder att allt innehåll på sidan alltid skall vara minst 3 klick ifrån. Allt innehåll på sidan som även har behövt beskrivningar har även fått detta, ett exempel på detta är karmapoängen som beskrivs på en enskild sida. Bilder och symboler har även lagts till för att förstärka meningen med olika funktionaliteter. Till sist, så har marknadsplatsen optimerats för att innehållet skall laddas snabbare. Detta har gjorts genom att införa paginering av annonsobjektet, så att endast ett specifikt antal objekt hämtas åt gången. Detta var tidigare ett problem på den tidigare plattformen eftersom förfrågningar för väldigt många objekt skickades samtidigt.

9.3 Hållbar utveckling

Plattformens huvudsyfte är att främja en cirkulär ekonomi, vilket gör att det finns ett starkt samband mellan plattform och hållbar utveckling ur ett ekologiskt perspektiv. Genom att möjliggöra att fler individer väljer begagnat före att köpa nytt kan användningen av miljöns resurser reduceras. Vi ser också att det finns ett starkt koppling mellan hållbar utveckling ur ett ekonomiskt perspektiv på grund av samma anledning.

9.4 Etisk diskussion

Ur ett etiskt perspektiv finns det ett par punkter som bör diskuteras. Till att börja med är tillämpningen av spelifiering i webbapplikationen menat att uppmuntra användande i syftet att främja hållbar utveckling. Däremot är det inte omöjligt att tillämpningen kan framkalla ett beroende hos användaren på grund av spelelementen, något som i sin tur hade kunnat medföra ett skadligt beteende hos den egna individen. Vi ser även en risk i att anonyma användarkonton kan skapas för att annonsera ut bluffannonser eller ta kontakt med annonsörer i syftet att utföra icke etiska handlingar. Som tidigare nämnt i avsnitt 1.5, Avgränsningar, är däremot detta något som inte tagits någon som helst hänsyn till vid utvecklingen av applikationen. I projektet har vi endast fokuserat på spelfieringens positiva aspekter samt implementationen av applikationens funktionalitet.

9.5 Vidareutveckling

I nuläget är ShareConnect endast en marknadsplats för annonser men vi ser också potentialen till att vidareutveckla plattformen. Tidigare under avsnittet 2.2.4 Spelifiering och hållbar utveckling nämndes bland annat plattformen WeSpire, som är en plattform med en mängd olika effektivitetsprogram inom områden såsom hållbarhet. Vi ser att det finns en möjlighet att implementera något liknande för ShareConnect där användare hade kunnat skapa och delta i olika projekt för att främja hållbarhet och det övriga samhället. Det här är även något som diskuterats med handledaren på Ericsson. Om man även fortsätter på

hållbarhetsspåret så finns även tankar om hur man skulle kunna använda de poäng som användaren kan intjäna på olika sätt. En tanke är att ShareConnect hade kunnat utvecklas till en plattform för personlig utveckling i området hållbar utveckling, där poängen skulle kunna vara resultatet av dina hållbara handlingar. Likt RecycleBank som togs upp i avsnitt 2.2.4, finns det även idéer om att införa rabatter på olika produkter i utbyte mot poäng som intjänas i tjänsten. Däremot hade detta medfört att det behövt finnas striktare regler på när poäng delas ut till användaren för att förhindra olika typer av olämpligt missbrukande. Till sist, finns även tankar om att införa hållbarhetsquiz och andra funktioner för att stärka detta syfte. Gällande marknadsplatsen ser vi också en möjlighet att man hade kunnat ta kontakt med matbutiker för samarbete, där skadad mat hade kunnat skänkas bort eller säljas till ett billigt pris i stället för att slängas, förutsatt att det följer diverse restriktioner.

Många av dessa funktionaliteter som hade kunnat utvecklats och som diskuterades i den tidigare paragrafen bygger dock på att plattformen redan blivit publicerad. Innan projektet dock kommer till ett sådant stadie anser vi att det också hade varit av intresse att utföra användartester för att ta reda på om spelifieringen uppfyllt de förhoppningar om att det skulle ge en ökad aktivitet på plattformen. Användartesterna hade exempelvis kunnat utföras genom att låta en testgrupp använda plattformen och ställa frågor gällande vilket incitament de har för att utföra handlingar i olika situationer. Grundtanken med dessa användartester är till största del att det skulle kunna ge värdefull feedback om hur spelifieringen hade kunnat förbättras eller byggas ut för att öka aktiviteten ytterligare.

Litteratur

[1] Khaled Soufani Mark Esposito Terence Tse. "Is the Circular Economy a New Fast-Expanding Market?" I: *Thunderbird International Business Review* 59.1 (2015), s. 9–14. DOI: https://doi.org/10.1002/tie.21764.

[2] Ole Goethe. *Gamification Mindset*. Cham, Switzerland: Springer, 2019. URL: https://link.springer.com/book/10.1007%2F978-3-030-11078-9.

[3] Susanne Robra-BissantzRüdiger Zarnekow Tobias Brockmann Stefan Stieglitz Christoph Lattemann. *Gamification: Using Game Elements in Serious Contexts*. Cham, Switzerland: Springer, 2015. URL: https://link-springer-com.proxy.lib.chalmers.se/book/10.1007%2F978-3-319-45557-0.

[4] Jan H. Kietzmann Ian McCarthy Leyland Pitt Karen Robson Kirk Plangger. "Game on: Engaging customers and employees through gamification". I: *Business Horizons* 59.1 (2015), s. 29–36. DOI: https://doi.org/10.1016/j.bushor.2015.08.002.

[5] *Habitca*. https://habitica.com/. Hämtad: 2020-03-19.

[6] *Khan Academy*. https://www.khanacademy.org/. Hämtad: 2020-03-19.

[7] Recyclebank. *How Recyclebank Works*. https://www.recyclebank.com/about-us/. Hämtad: 2020-04-01.

[8] WeSpire. *Product*. https://www.wespire.com/product/. Hämtad: 2020-04-01.

[9] Energypost. *OPower*. https://energypost.eu/opower/. Hämtad: 2020-04-02.

[10] Azat Mardan. *Pro Express.js*. Berkeley, CA: Apress, 2014.

[11] React. *React.Component*. https://reactjs.org/docs/react-component.html. Hämtad: 2020-03-28.

[12] David HowsPeter MembreyEelco Plugge. *MongoDB Basics*. Berkeley, CA: Apress, 2014. URL: https://link-springer-com.proxy.lib.chalmers.se/content/pdf/10.1007%2F978-1-4842-0895-3.pdf.

[13] Matt Warcholinski. *10 Famous Apps Using ReactJS Nowadays*. https://brainhub.eu/blog/10-famous-apps-using-reactjs-nowadays/. Hämtad: 2020-04-22.

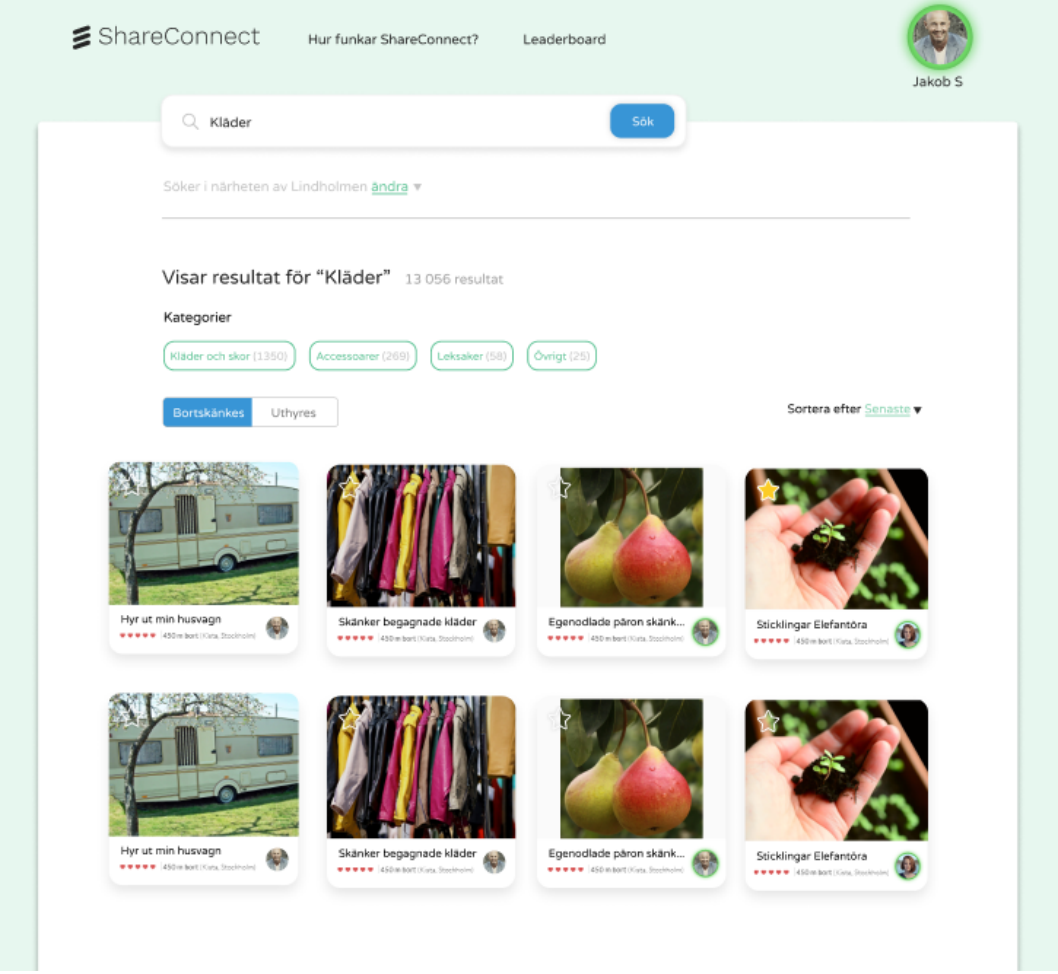
[14] *Bootstrap-React*. https://react-bootstrap.github.io/. Hämtad: 2020-05-19.

[15] *Sweet Alert 2*. https://sweetalert2.github.io/. Hämtad: 2020-05-19.

[16] *Base64*. https://developer.mozilla.org/en-US/docs/Glossary/Base64. Hämtad: 2020-05-21.

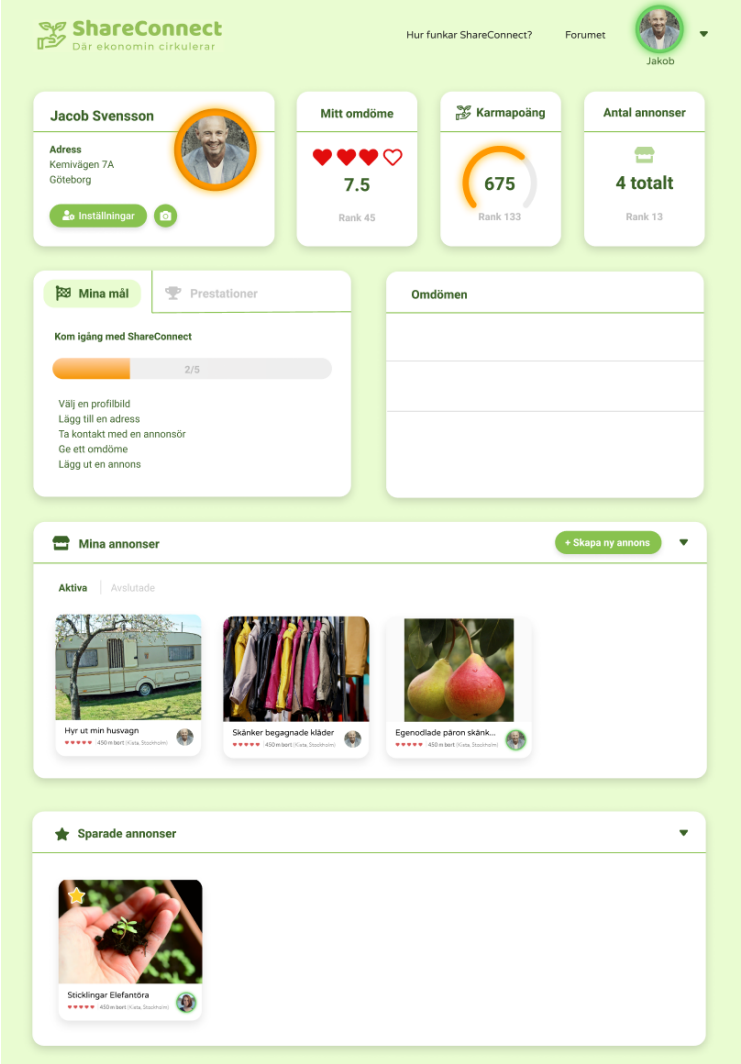
Bilaga A

Skiss över marknadssidan



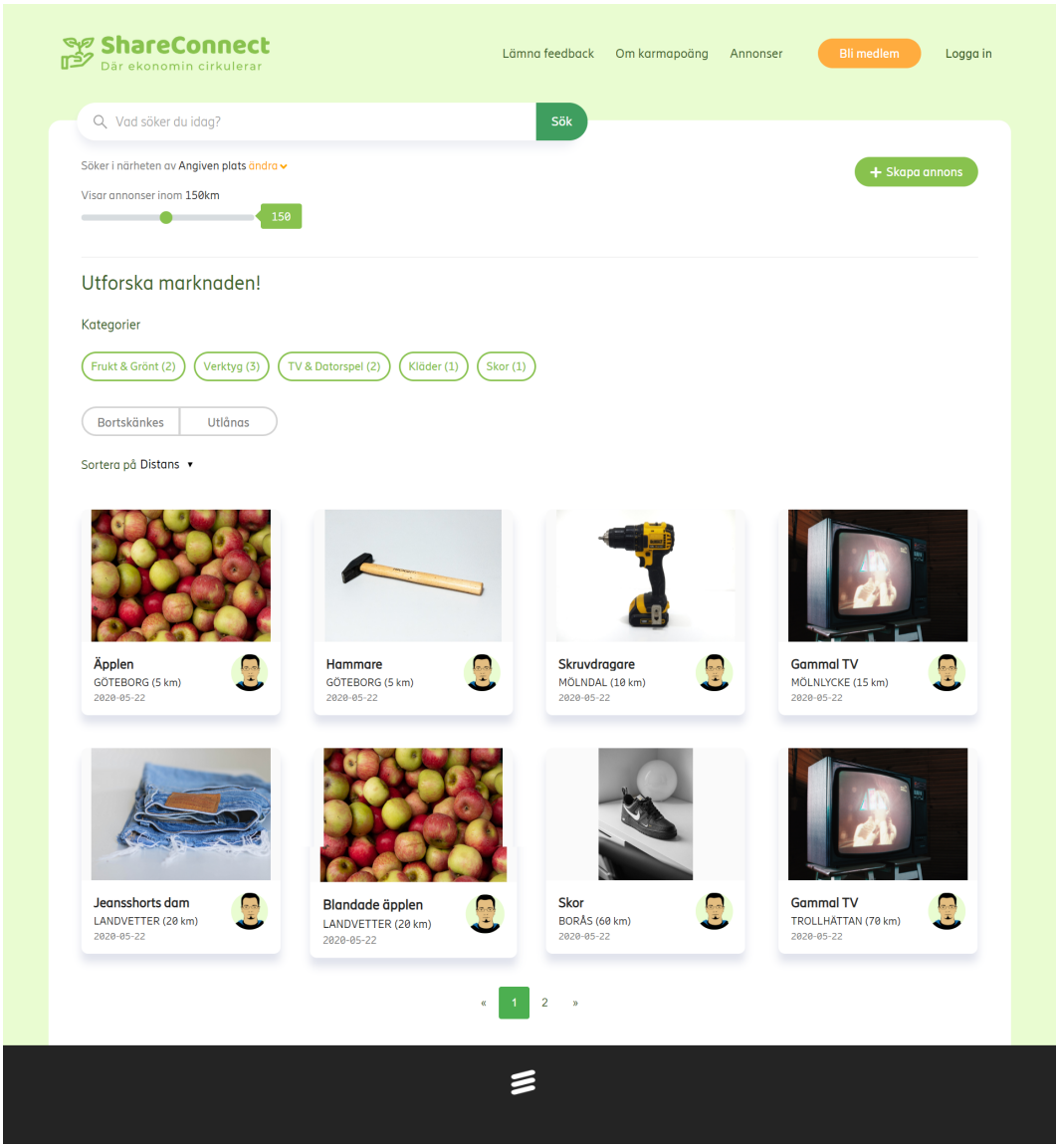
Bilaga B

Skiss över profilsidan



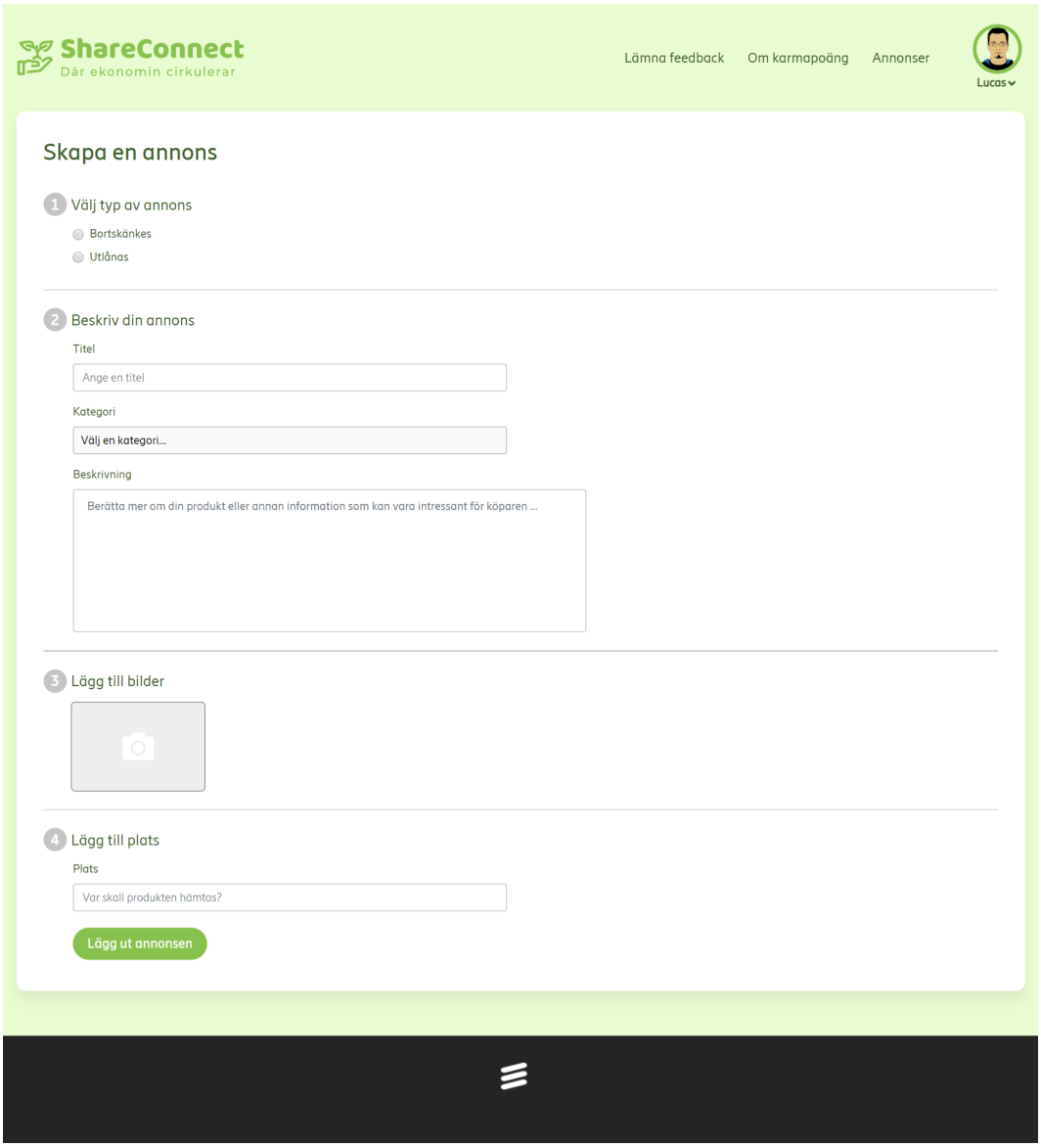
Bilaga C

Marknadsplats



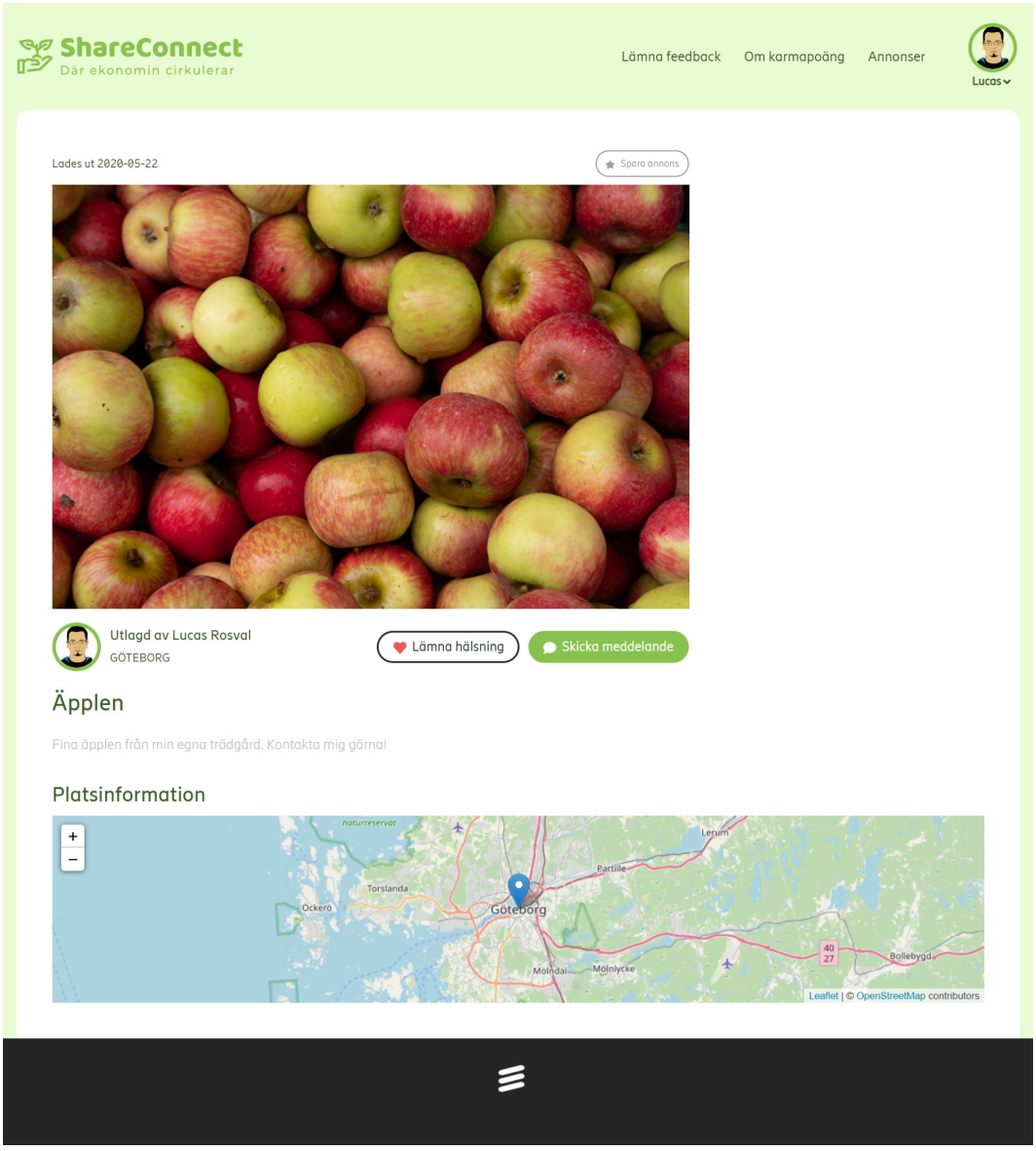
Bilaga D

Skapa annons sida



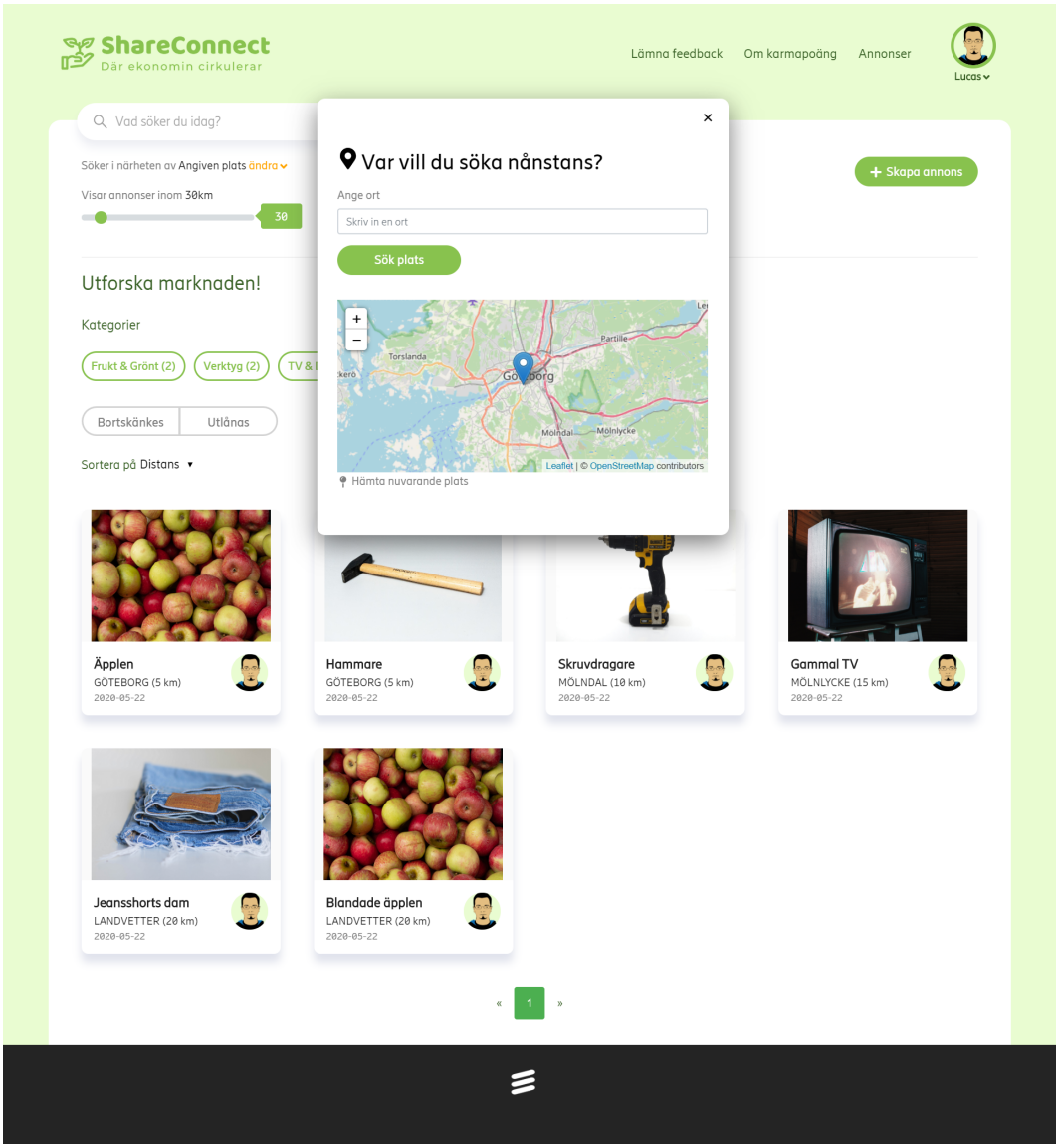
Bilaga E

Annonssida



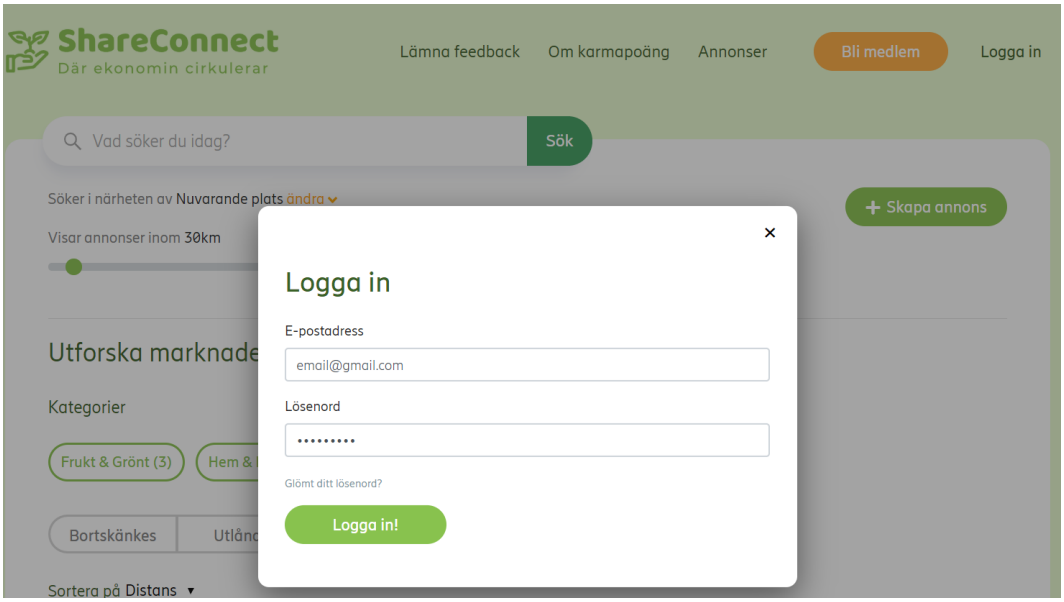
Bilaga F

Platssökningsmodul



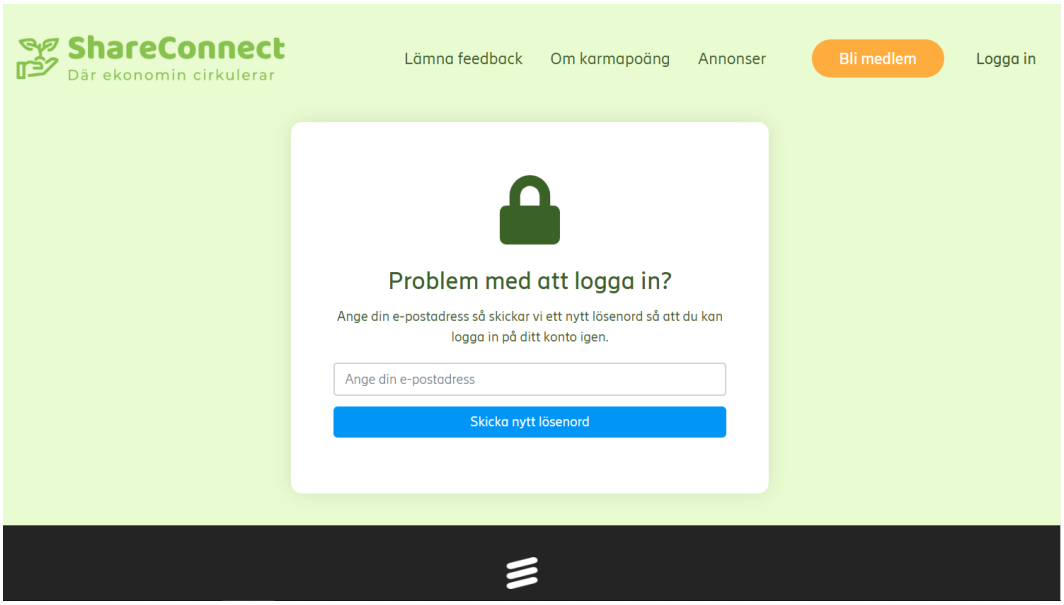
Bilaga G

Login-modul



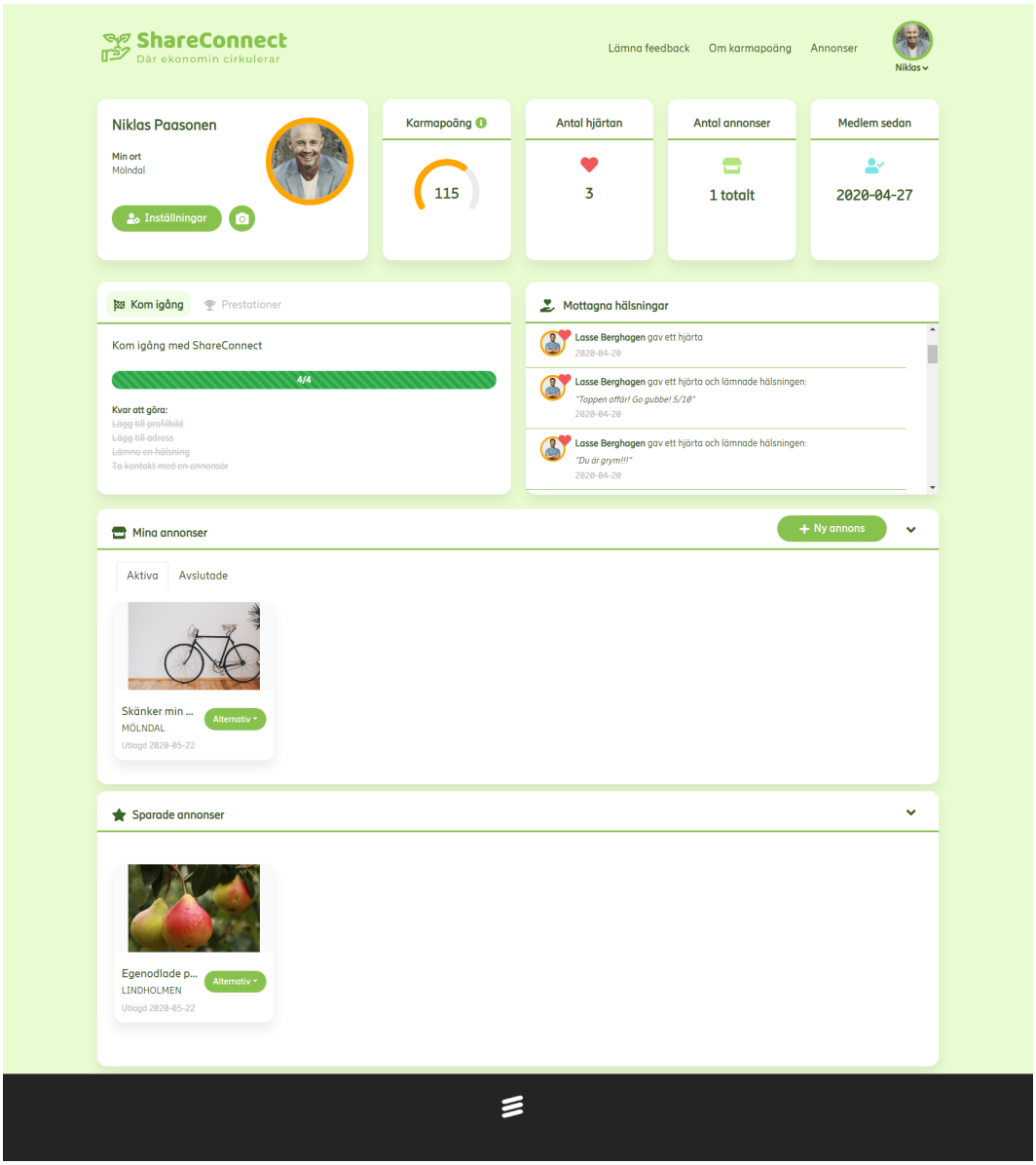
Bilaga H

Glömt lösenord



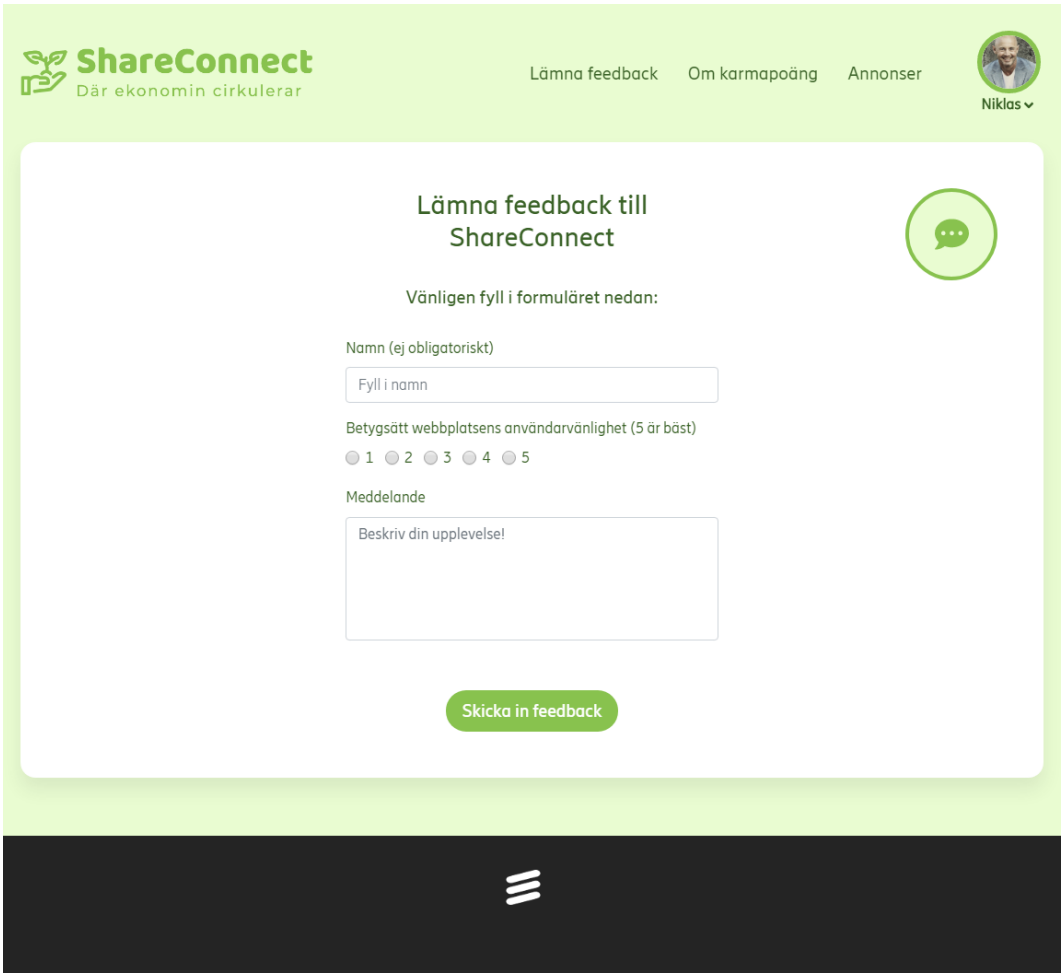
Bilaga I

Användarens profilsida



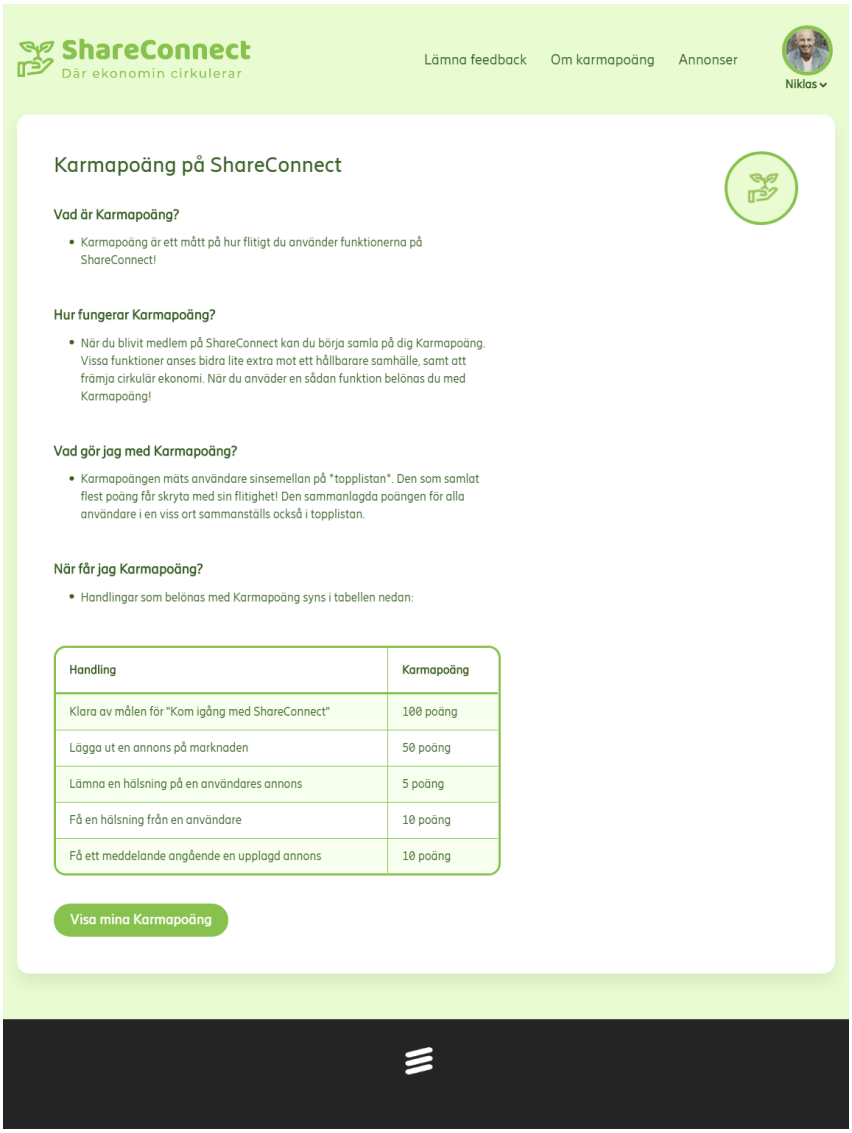
Bilaga J

Lämna synpunkter på hemsidan



Bilaga K

Information om Karmapoäng



Bilaga L

Haversine formel

Formeln används för att beräkna avståndet givet två koordinater. φ är latituden, λ är longituden medan R är jordens radie (6 371 km).

$\mathrm{a} = \sin^2(\Delta\varphi/2) + \cos(\varphi_1) \times \cos(\varphi_2) \times \sin^2(\Delta\lambda/2)$

$c = 2 \times atan2(\sqrt{a}, \sqrt{(1\text{-}a)})$

$d = R \times c$