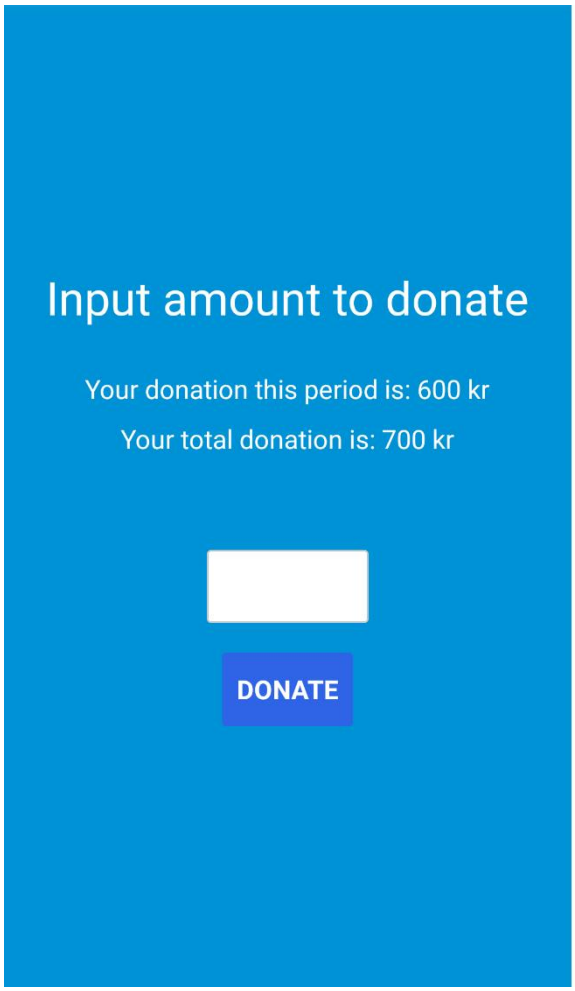
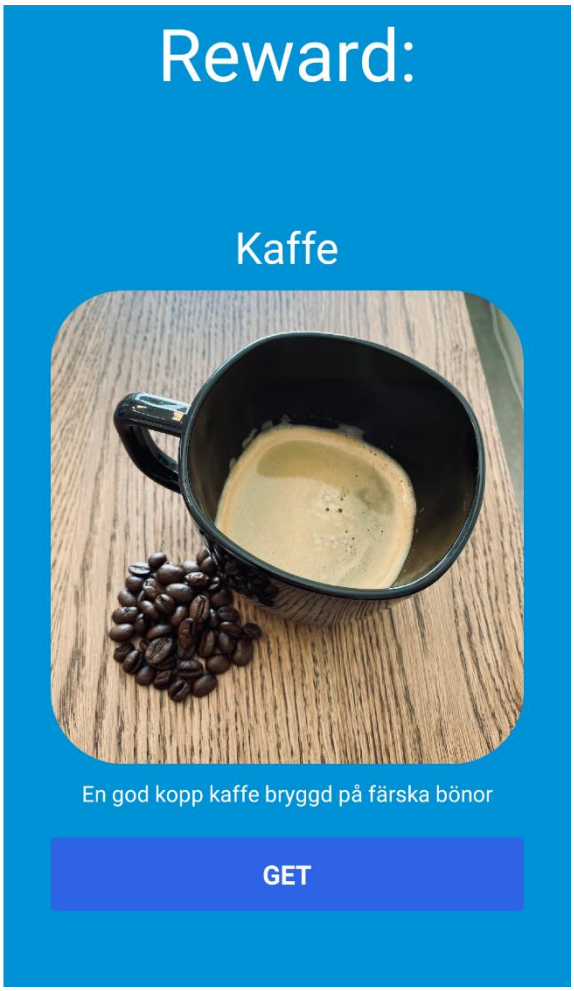
CHALMERS





Konceptest för donation och belöningsapplikation

Ett sätt att stödja kafébranschen under coronapandemin

Examensarbete inom Data- och Informationsteknik

Carl Andersson
Joakim Lönnerstedt

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

EXAMENSARBETE

Koncepttest för donation och belöningsapplikation

Ett sätt att stödja kafébranschen under coronapandemin

Carl Andersson
Joakim Lönnerstedt

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2021

Koncepttest för donation och belöningsapplikation
Ett sätt att stödja kafébranschen under coronapandemin
Carl Andersson
Joakim Lönnerstedt

Examinator: Peter Lundin

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

Omslag:
Egen bild tagen från donations och belöningssidan i mobilapplikationen. Foto av kaffe tagen av Carl Andersson.

Institutionen för Data- och Informationsteknik
Göteborg 2021

Förord

Denna rapport och det projekt den handlar om är resultatet av ett examensarbete vid institutionen för data- och informationsteknik våren 2021 på Chalmers tekniska högskola. Ett stort tack till Annelie Svensson och Johan Schyberg som inte bara varit våra handledare under projektet utan kom på idén för projektet från första början. Ni har varit ett bra bollplank för idéer och kommit med vinklar vi kanske inte tänkt på tidigare. Vi vill även tacka Sakib Sistek som varit vår handledare från Chalmers och som alltid ställer upp. Du har alltid bra feedback med din enorma mängd kunskap. Slutligen vill vi tacka Katarina Gustavsson, en tidigare kurskamrat som gav oss en stor mängd feedback på vår rapport. Feedbacken var guld värd och är en del av den resulterade rapport du nu läser.

Carl Andersson, Joakim Lönnerstedt, Göteborg, 2021

Sammanfattning

Under coronapandemin drabbades kafébranschen hårt när gästerna minskade. Ett sätt att ge stöd till branschen är för privatpersoner att donera pengar till ett kafé men det saknas idag tjänster som inriktar sig på just denna branschen. Projektet mål var att utveckla en prototyp till en mobilapplikation för kunder som gör det möjligt att donera pengar till ett kafé, och en webbapplikation där kaféägare i gengäld kan ge ut gåvor beroende på vilken summa som donerats. Med hjälp av JavaScript-verktygen ReactJS och React-Native har applikationer utvecklats för kund och kaféägare. Dessa två applikationer kommunicerar genom Firebase som gett projektet tillgång till en stor mängd verktyg. Arbetet resulterade i ett system där en användare kan donera pengar och köpa varor från ett kafé. Kunden identifieras genom att deras QR-kod skannas i den webbapplikation som kaféägaren har tillgång till. Genom webbapplikationen kan ägaren lägga in olika gåvor beroende på vilken summa som donerats, identifiera kunder och lägga ut produkter som kan köpas. Projektet uppnåde det satta målet och det finns stor potential för vidareutveckling av projektet.

Nyckelord: Koncepttest, QR, ReactJS, React-Native, Firebase

Abstract

The café industry got hit hard by the corona pandemic. One way to support the industry is for individuals to donate money to cafés. However there does not exist such a service today that focuses on this industry. The goal of the project was to develop a proof of concept for such a service that allow individuals to make donations to a café. Café owners can in return give rewards depending on the sum an individual donated. By using the JavaScript tools ReactJS and React-Native, two applications has been developed, one for the customer and the other for the café owner. These two applications communicate through Firebase which provides the project with a large number of tools. The project resulted in a system where a user can donate money and buy wares from a café. The café owner identifies a customer by scanning their QR-code in a web-application which they have access to. In the web-application the owner can add different rewards depending on the amount donated, identify customers and add products that can be bought. The project achieved the set goals and there is big potential for further development of the project.

Keywords: Proof of Concept, QR, ReactJS, React-Native, Firebase

Ordlista

API - Applikationsprogrammeringsgränssnitt
BLE - Bluetooth Low Energy
CSS - Cascading Style Sheets
GDPR - Dataskyddsförordningen (en. General Data Protection Regulation)
HTTP - HyperText Transfer Protocol
QR - Quick Response

Figurer

Innehåll

1 Inledning

I kapitlet nedan redogörs bakgrundsinformation som ligger till grund för projektet. Målet med projektet, syftet med att utföra projektet samt frågeställningar och avgränsningar formuleras.

1.1 Bakgrund

I början av år 2020 drabbades världen av coronapandemin. För att minska smittspridningen i landet infördes restriktioner och med det, minskade hushållskonsumtionen[1] vilket drabbade kafébranschen hårt[2]. För att hjälpa branschen kan människor donera pengar men det finns idag ingen smidig tjänst för detta som riktar sig till kafébranschen. Som tack för donationer vill kaféer ha möjligheten att ge ut gåvor till lojala medlemmar. Detta projekt handlar om att utveckla en mobil och webbaserad prototyp för en sådan lösning.

1.2 Mål

Målet med projektet är att utveckla en mobilapplikation och en webbapplikation. Mobilapplikationen utvecklas för kafékunder till Android och iOS där användaren kan donera en summa varje månad mot någon belöning beroende på summan. Applikationen ska också kunna identifiera användaren mot webbapplikationen. Webbapplikationen utvecklas åt kaféägaren och används för administration av gåvor samt identifiering av användare. I applikationen kan ägaren även se vilken gåva användaren är berättigad till och om den är uthämtad. Ägaren ska själv ha möjlighet att lägga in hur många belöningsnivåer som finns och vid vilken summa belöningen låses upp.

1.3 Syfte

Syftet med projektet är att utveckla ett koncepttest bestående av databas, mobilapplikation och webbapplikation. Användare ska kunna stödja ett kafé med en donation mot en belöning och kaféet ska kunna identifiera dessa donatorer och ge ut gåvan om användaren önskar. Detta för att människor ska kunna stödja en bransch som haft det svårt ekonomiskt under pandemin.

1.4 Frågeställningar

- Vad är en bra lösning för att identifiera användare som donerat?
- Hur kan projektet utformas för att kräva minimalt med hårdvara för kaféägaren?

1.5 Avgränsningar

Nedan följer de avgränsningar som gjorts för projektet.

Mobilapplikation

- Utvecklas för Android och iOS.
- Implementerar endast en betaltjänst.

Webbapplikation

- Ingen möjlighet att administrera databasen från applikationen.

Databas

- Ingen egen server kommer utvecklas att hysa databasen på.

2 Teknisk bakgrund

I detta kapitel beskrivs teori för de tekniker och mjukvaror som använts vid genomförandet av projektet. Kapitlet innehåller dessutom bakgrund om de tjänster som använts.

2.1 Quick Response

För att identifiera användare mot kaféet genereras en kod kallad Quick Response (QR). Ett exempel på hur en QR-kod ser ut visas i figur 1. En QR-kod är en tvådimensionell matris bestående av data[3]. Informationen från en QR-kod kan enkelt läsas från ett foto eller med en kamera. QR-standarden innehåller även felkorrigering och kan skannas från vilket håll som helst då koden innehåller information om ordningen datan ska läsas. Koden är portabel och kan enkelt skrivas ut på papper eller genereras i en telefon. Det gör den smidig till identifiering eftersom QR-koden enkelt kan genereras i en mobiltelefon och skannas direkt från skärmen.

Figur 1: QR-kod till Chalmers Data & IT institution

2.2 3D Secure

3D Secure är en teknik som används vid betalningar online[4]. Tekniken kräver extra verifieringssteg för att minska risken för bedrägerier online. Vanligtvis fungerar verifieringssteget så att användaren dirigeras till en sida på användarens bankhemsida där någon form av verifiering sköts av banken. För betalningar inom Europa måste 3D Secure användas enligt lag.

2.3 Firebase

Utgångspunkten för projektet var att hitta en lösning där kaféägaren själv inte behövde hysa någon egen server. Valet blev Googles lösning Firebase. Firebase är en utvecklingsplattform för applikationer som erbjuder flertalet olika tjänster. Firebase kan användas gratis upp till en viss nivå av användning, därefter tillkommer en kostnad som är proportionerlig mot användandet. Nedan följer bakgrund till de olika tjänsterna från Firebase som har använts i projektet.

2.3.1 Firebase Authentication

Firebase Authentication är ett ramverk för att hantera autentisering. Det tar hand om registrering, inloggning och sparandet av lösenord och inloggningsmetod[5].

2.3.2 Firebase Firestore Database

Firestore Database är en skalbar NoSQL-databas[6] för liten data som textsträngar och nummertal. Strukturen på Firebase Firestore är uppbyggd på att skapa olika samlingar, kallade *collections*, där dokument kan läggas till. I dessa dokument kan fält med nyckel-värde par och nya samlingar skapas. Firestore fungerar i realtid och har en offline-lösning som sparar uppdateringar till dess att en internetanslutning upptas igen. Säkerheten i Firestore hanteras av olika regler som sätts och styr vilka kriterier som måste uppfyllas för att få tillgång till dokument i databasen. Typen av tillgång, som skrivning eller läsning, är även styrd av reglerna.

2.3.3 Firebase Storage

Som komplement till Firebase Firestore har Firebase Storage använts. Firebase Storage är en servicetjänst för molnlagring som kan lagra större objekt som till exempel bilder[7]. Säkerheten för Firebase Storage är av samma typ som för Firebase Functions.

2.3.4 Firebase Functions

Firebase Functions erbjuder en servermiljö där kod kan köras utan någon möjlighet för användare att påverka körningen. Firebase Functions erbjuder både funktioner som kan anropas genom HTTP anrop och funktioner som aktiveras av händelser[8]. Dessa händelser kan till exempel vara en uppdatering på ett specifikt ställe i databasen på Firebase Firestore eller att en ny användare skapas via Firebase Authentication. Firebase Functions skalas dessutom till användandet så att ju oftare en funktion används, desto fortare kommer den att köras. Serverfunktioner i Firebase Functions som hanterar

databasen står över de regler som sätts för databasen då de anses vara i en säker miljö.

2.3.5 Firebase Hosting

För att hysa webbapplikationen har Firebase Hosting använts. Firebase Hosting är en internettjänst som erbjuder värdservrar för hemsidor[9]. Tjänsten är dessutom skalbar till användandet likt Firebase Functions.

2.4 Stripe

För funktionaliteten av betalningar i projektet har Stripe använts. Stripe är en finansiell tjänst som erbjuder applikationsprogrammeringsgränssnitt (API) för betalningar på hemsidor och i mobilapplikationer[10]. De olika objekt som använts från Stripe i detta projekt är Token, Source och PaymentIntent. Dessa kommer att redogöras för i de kommande undersektionerna.

2.4.1 Token

Token är ett objekt som Stripe använder för att på ett säkert sätt samla in känslig information, som till exempel en användares kortuppgifter[11]. Skapandet av ett Token objekt kan göras genom direkt kontakt med Stripe och behöver därför inte behandlas av någon mellanliggande server som kan utgöra en säkerhetsrisk. Token objektet kan endast användas en gång och kan därför inte användas som en kontinuerlig betalningsmetod. För detta används i stället ett Source objekt.

2.4.2 Source

Ett Source objekt är ett objekt som fungerar som en betalningskälla som kan användas för kommande betalningar[12]. För att skapa ett Source objekt kan ett Token objekt som innehåller en användares kortuppgifter användas.

2.4.3 PaymentIntent

PaymentIntent är ett objekt som skapas av Stripe där information om en betalning finns sparad[13]. Objektet kan även genomgå olika tillstånd som beskriver status på betalningen, som till exempel *Succeeded* och *Incomplete* efter ett betalningsförsök. PaymentIntent stödjer användningen av 3D Secure som krävs för att genomföra betalningar online inom de europeiska länderna.

2.5 Programspråk

Det primära programmeringsspråk som har använts under arbetet är JavaScript. Både mobil- och webbapplikationen har till stor del använt och varit beroende av olika bibliotek och ramverk för JavaScript. CSS kod har dessutom använts för att kontrollera utseendet på delar av webbapplikationen.

2.6 React

På grund av att JavaScript är ett programmeringsspråk främst menat för logik och inte grafik var ett komplement för det grafiska nödvändigt. Detta hittades i ReactJS och React-Native som kommer att redogöras för nedan.

2.6.1 ReactJS

ReactJS är ett JavaScript-bibliotek med öppen källkod skapat av Facebook [14]. Biblioteket är till för att skapa användargränssnitt och grafiska komponenter för webbutveckling. I projektet används ReactJS till hjälp för att utveckla webbapplikationen.

2.6.2 React-Native

React-Native är ett utvecklingsramverk skapat att efterlikna utvecklingsformen hos ReactJS[15]. React-Native tillåter kod skriven i JavaScript att kopplas till plattform specifika komponenter, vilket betyder att plattformsberoende API:er kan användas för flera plattformar. Med hjälp av detta har en kodbas utvecklats för både iOS och Android. Källkoden för React-Native är precis som ReactJS öppen och skapad av Facebook.

2.7 Ytterligare bibliotek

För att få tillgång till ytterligare funktionalitet som ReactJS och React-Native inte erbjöd, har ett antal fler bibliotek importerats och använts.

2.7.1 Webbapplikationen

För webbapplikationen har nio bibliotek använts. Dessa är react-bootstrap, react-firebase, react-qr-reader, i18next, react-i18next, i18next-http-backend, js-cookie, react-router-dom och react-export-excel.

react-bootstrap
react-bootstrap är ett bibliotek som erbjuder flertalet färdiga grafiska komponenter[16]. Biblioteket användes i webbapplikationen som ett komplement till de grafiska komponenter som ReactJS erbjuder för att skapa en mer stilfull webbapplikation.

firebase-js-sdk
firebase-js-sdk är ett JavaScript bibliotek skapat och underhållet av Firebase[17]. Biblioteket erbjuder funktion för användning av de tjänster från Firebase som används i webbapplikationen.

react-qr-reader
För att kunna läsa av de QR-koder som genereras på mobilapplikationen användes biblioteket react-qr-reader[18]. Biblioteket använder webbkameran på användarens dator och letar samt läser av eventuella QR-koder.

i18next
För att få funktionalitet för översättning på webbapplikationen användes i18next[19], react-i18next[20] och i18next-http-backend[21]. i18next är ett internationaliserings-ramverk som kan användas för att få översättning av hemsidor till flera språk. För att få i18next att fungera med ReactJS behövdes även react-i18next som används för att göra i18next tillgängligt för alla ReactJS komponenter. För att ladda översättningsresurserna genom HTTP från en server används även i18next-http-backend.

js-cookie
js-cookie är ett bibliotek som används för att hantera webbläsarkakor[22]. Biblioteket används i webbapplikationen endast för att ställa in det förvalda språket vid initieringen av i18next.

react-router-dom
react-router-dom är ett bibliotek för ReactJS som erbjuder navigering[23]. Detta bibliotek har använts för att navigera mellan de olika sidorna på webbapplikationen och har dessutom använts för att neka åtkomst om inte någon användare är inloggad.

react-export-excel
För att få funktionalitet för exportering av användardata till nedladdningsbart Excel format har rect-export-excel använts[24].

2.7.2 Mobilapplikationen

För mobilapplikationen har fyra bibliotek använts. Dessa är react-native-firebase, react-navigation, react-native-qrcode-svg och tipsi-stripe.

react-native-firebase
react-native-firebase är ett bibliotek som används för att kunna få tillgång till de Firebase funktioner som behövts i mobilapplikationen[25]. Detta bibliotek har använts på grund av att Firebase själva inte erbjuder en egen lösning för att använda deras tjänster tillsammans med React-Native. Biblioteket är

rekommenderat av Firebase själva genom deras hemsida men är inte officiellt utgivet av Firebase.

react-navigation
Ett annat bibliotek som använts i mobilapplikationen är react-navigation[26]. Det har använts för att kunna navigera mellan de olika sidorna i applikationen.

react-native-qrcode-svg
För att skapa olika QR-koder har biblioteket react-native-qrcode-svg använts[27]. Biblioteket erbjuder funktionalitet för att skapa QR-koder av textsträngar.

tipsi-stripe
Vid tidpunkten av arbetet tillhandahöll Stripe inget eget bibliotek för användningen av deras funktioner för React-Native. I stället användes biblioteket tipsi-stripe som erbjuder en länk för React-Native till de plattformsberoende biblioteken för Stripe som finns tillgängliga för iOS och Android[28]. Efter det att arbetet slutförts har Stripe utvecklat ett bibliotek för funktionalitet för React-Native.

3 Metod

Detta kapitlet innehåller den metod som använts för att genomföra projektet.

3.1 Arbetsmetod

Projektet har mestadels använt sig av vattenfallsmetoden för arbetsprocessen. Mobil och webbapplikation utvecklades parallellt med varandra och konfigurationen för databasen implementerades tidigt i processen för båda applikationerna.

3.2 Initiera projekt

För att starta ett React-Native projekt krävs NodeJS, Java-utvecklingskit mjukvara och en utvecklingsmiljö. När detta hämtats körs kommandot *npx react-native init MobilAppNamn* i den mapp där projektfilerna ska ligga (se figur 2). Detta genererar ett React-Native projekt och alla filer som behövs med namnet MobilAppNamn. En mer detaljerad information om installationen hittas här: https://reactnative.dev/docs/environment-setup. Utöver det har Android Studio och Xcode använts för emulering av telefoner. Detta möjliggör testning av den kod som skrivs. För att testa koden på en emulator körs kommandot *npx react-native run-android* för Android och *npx react-native run-ios* för iOS från mappen där projektet ligger.

```
C:\Users\Joakim> npx react-native init MobilAppNamn
```

Figur 2: Initiering av React-Native projekt

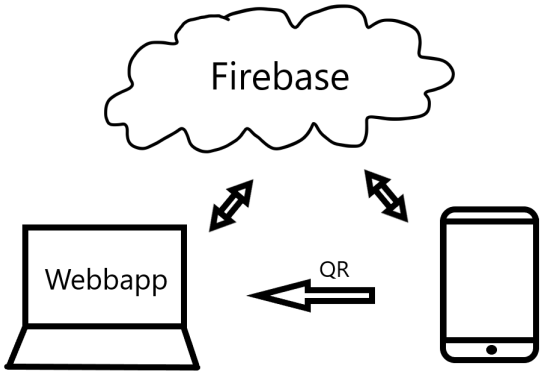
Att starta ett ReactJS projekt är liknande som att starta ett React-Native projekt. Precis som för React-Native krävs NodeJS för att kunna starta ett projekt. Kör kommandot *npx create-react-app WebbAppNamn* där projektfilerna ska ligga (se figur 3). Testning av koden körs i en webbläsare efter de att projektet startats med kommandot *npm start*.

```
C:\Users\Joakim> npx create-react-app WebbAppNamn
```

Figur 3: Initiering av ReactJS projekt

4 Systemkonstruktion

Projektet har bestått av fyra olika delar som tillsammans bildar ett system. Dessa fyra delar är mobilapplikation, webbapplikation, databas och serverfunktioner. Applikationerna kommunicerar främst över internet med varandra genom Firebase, men mobilapplikationen kan även kommunicera med webbapplikationen med hjälp av QR-kod (se figur 4). Nedan redogörs hur varje system fungerar och kommunicerar.



Figur 4: Illustration över systemkommunikation

4.1 Mobilapplikation

Applikationen är uppbyggd av tre huvudsakliga kategorier, komponenter, navigation och skärmar. Skärmar innehåller koden för varje enskild sida applikationen visar, allt från inloggningssidan till sidan där QR-koden visas. Komponenter är mindre delar som återanvänds i skärmarna. De två komponenter som skapats är ett inmatningsfält och en knapp. Inmatningsfältet används vid inloggning och registrering. Den knapp som skapats används i applikationen för att skapa knappar som användaren kan trycka på. De filer som tillhör navigation hanterar navigationen mellan skärmarna som är strukturerade i en hierarki. Filerna har kod för vilka skärmar som kan nås från olika skärmar men också navigation av data mellan inmatningsfält och databas.

4.2 Webbapplikation

Webbapplikationens uppbyggnad liknar på många sätt uppbyggnaden av mobilprojektet. Webbapplikationen består till stor del av enskilt uppbyggda skärmar där funktionalitet som QR-skanning och hämtande av information från databasen är direkt inbyggda. Det finns även enskilda komponenter som återanvänds i olika delar av projektet som till exempel ett navigeringsfält och

en modul för att kontrollera autentiseringen med Firebase Authentication. Det finns dock en stor skillnad mot mobilprojektet i hur navigeringen fungerar. I webbapplikationen behövs ingen hierarki mellan skärmarna utan man kan enkelt navigera mellan de olika skärmarna via länkar. Länkar finns framförallt i navigeringsfältet som visas på alla sidorna när man väl har loggat in. Webbapplikationen har efter utveckling laddats upp till internet med hjälp av Firebase Hosting.

4.3 Databas

Databasen på Firebase Firestore är uppbyggd för att innehålla all information som är nödvändig för projektet.

4.3.1 Struktur

Databasen är strukturerad med fyra samlingar. Den första kallas *users* och innehåller grundläggande information som namn och donerat belopp om alla användarna. Två andra samlingar är *products* och *rewards* som respektive innehåller information om de produkter som kaféet säljer och de gåvor som användare kan kvalificera sig för. Den fjärde samling är *stripe_customers* som består av fler samlingar som tillsammans innehåller all information som har med Stripe att göra. Detta inkluderar betalning- och donationshistorik samt uppgifter om användarnas betalningsmedel som är genererat av Stripe.

4.3.2 Säkerhet

För att skydda känslig information har även regler för databasen implementerats. Generellt krävs det för databasen att en användare är inloggad för att läsa eller skriva till databasen. Det finns dock fler regler för specifika dokument. Användaruppgifter kan till exempel endast läsas av användarna själva och kaféägarna.

4.4 Serverfunktioner

Av skäl som säkerhet och i vissa fall enkelhet har det varit nödvändigt att flytta en del funktionalitet från användarnas egna hårdvara till en central server. Funktioner har därför skrivits i JavaScript och laddats upp till Firebase Functions där de kan exekveras i en säker exekveringsmiljö. Kommunikationen med Stripe har varit den viktigaste anledningen till att flytta funktioner till en central server då detta kräver högre säkerhet.

5 Resultat

Det samlade resultatet av projektet består av de fyra delar som är angivna i systemkonstruktion, nämligen en mobilapplikation, en webbapplikation, en databas och serverfunktioner.

5.1 Mobilapplikation

Utvecklingen av mobilapplikationen resulterade i en applikation med följande funktioner. För att se det grafiska resultatet av mobilapplikationen se bilagorna B.1 till B.8.

5.1.1 Autentisering

Den mest grundläggande funktionaliteten av mobilapplikationen är möjligheten att kunna autentisera sig som användare. Detta fungerar med hjälp av Firebase Authentication där man kan registrera sig med en e-postadress och ett lösenord. Användare kan sedan logga in och ut med de valda autentiseringsuppgifterna. Vid registrering kräver applikationen även att användaren uppger sitt för- och efternamn som sparas på databasen som ligger på Firebase Firestore.

5.1.2 Kortuppgifter

När man väl är registrerad som användare har man möjlighet att lägga till sina kortuppgifter i applikationen. Då de flesta av applikationens funktioner bygger på betalningar är även detta en grundläggande funktionalitet. När användarens kortuppgifter har lagts in i applikationen kan de återanvändas vid betalningar utan att behöva skriva in dem igen.

5.1.3 Donering

Det är möjligt för användare att använda de kortuppgifter som tidigare lagts in och göra donationer av valfritt belopp, mellan 10 och 9999 kr till kaféet. Det går även att se hur mycket som redan har donerats under både den aktuella perioden och totalt.

5.1.4 Köp

Det är möjligt att i mobilapplikationen se alla de varor som kaféet erbjuder och välja att köpa en vara. Precis som för donationer, sköts betalningen av de kortuppgifter som tidigare lagts in. För att hämta ut de varor som köpts kan användaren via en QR-kod identifiera sig hos kaféet där kaféägaren kan se vilka varor som har betalats. Det är även möjligt att se sin köphistorik i applikationen och vilka varor som har hämtats ut från kaféet samt vilka som inte hämtats ut.

5.1.5 Gåvor

Efter att man har gjort en donation är det möjligt att man har gjort sig berättigad till en gåva från kaféet. Om så är fallet kan man i applikationen se vilken gåva och, likt hur man hämtar ut köpta varor, via en QR-kod identifiera att man är berättigad till gåvan på kaféet hämta ut den.

5.2 Webbapplikation

Resultatet av utvecklingen av webbapplikationen är ett fungerande webb-interface för kaféägare att kunna hantera de funktioner som i de kommande underkapitel kommer beskrivas. För att se det grafiska resultatet av webbapplikationen se bilagorna B.9 till B.15.

5.2.1 Autentisering

Precis som för mobilapplikationen är autentisering den mest grundläggande funktionen och fungerar här på samma sätt. Det går att registrera sig med e-postadress, namn och lösenord.

5.2.2 Skanning

En av de viktigaste funktionerna för webbapplikationen är skanningen av QR-koder. Med skanningen kan kunder både identifiera sig för att visa vilka varor som köpts och visa vilken gåva som kunden eventuellt är berättigad till.

5.2.3 Kundinformation

För att kaféägarna ska kunna få en tydlig bild av användandet av tjänsten var det även viktigt att ge dem en möjlighet att se viss kundinformation. Det har därför skapats en sida där kaféägarna kan se information om alla kunder som har registrerat sig genom mobilapplikationen. Där finns information om användarnas e-postadress, namn, totala donation till kaféet och donation till kaféet under pågående period. Det går även att sortera användarna på namn, total donation och donation under period, samt exportera informationen till Excel-format för att kaféägarna själva ska kunna hantera informationen externt.

5.2.4 Donationsinformation

En sida lades till där kaféägarna kan se den totala donationsumman från alla användare, både under pågående period och totalt. Där går det även att nollställa perioden för att ge kaféägarna själva möjligheten att styra längden på perioder.

5.2.5 Produkter

Likt funktionen i mobilapplikationen går det för kaféägarna att se de produkter som kaféet erbjuder tillsammans med information om dessa produkter. Information består av produktnummer, namn på produkten, bild på produkten, en beskrivning av produkten och pris. Det går även för kaféägarna att ta bort befintliga produkter samt lägga till fler.

5.2.6 Gåvor

De gåvor som användare för tillfället kan berättiga sig till kan ses på webbapplikationen av kaféägare. Där går det att se vilken produkt som gåvan består av och vilket belopp som en användare måste donera för att få gåvan. Kaféägarna kan här även ta bort och lägga till nya gåvor.

5.2.7 Byta språk

För att tjänsten inte enbart ska kunna användas av personer som talar svenska, lades funktionalitet för att byta språk till i navigeringsfältet. De språk som stöds är svenska och engelska men funktionaliteten för att lägga till fler språk är möjlig.

5.2.8 Kontoinformation

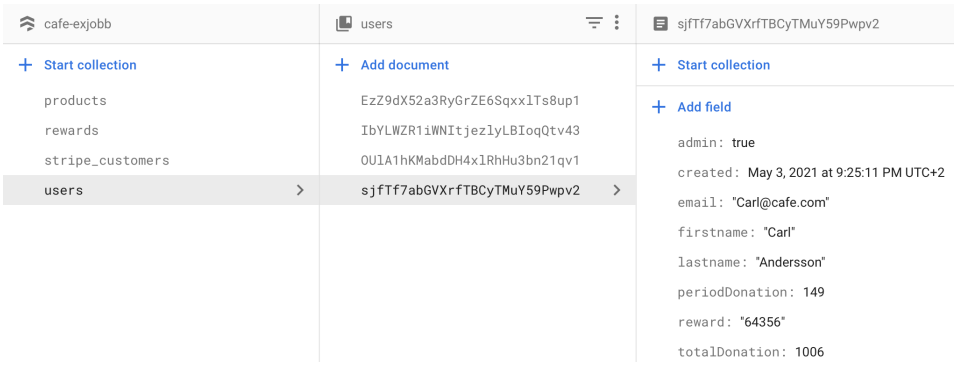
En sida består av grundläggande kontoinformation som namn och e-postadress. Möjligheten att byta lösenord och att logga ut finns också tillgänglig här.

5.3 Databas

Resultatet av utvecklingen av databasen är en fulltäckande databas som innehåller all nödvändig information för projektet. Databasen är strukturerad in i 4 samlingar som berör information gällande användare, produkter, gåvor och Stripe. Denna sektion kommer att redogöra för samtliga delar, samt vilka regler som gäller för respektive samling. För att se de fullständiga reglerna satta för databasen se bilaga A.1.

5.3.1 Användardata

Användardatan som finns sparad på databasen består av 8 fält med information. Dessa är förnamn, efternamn, e-postadress, datumet användaren registrerades, värdet på den totala och den periodiska donation som användaren gjort till kaféet, den gåva som användaren eventuellt är berättigad till och ett fält som berättar om användaren är admin, vilket i detta fallet betyder kaféägare. Ett urklipp från databasen med information om användardatan finns att se i figur 4 Användardatan skyddas av regler som säger att datan endast kan läsas och skrivas till av användaren själv samt kaféägare.



Figur 5: Urklipp från Firebase Firestore med användardata

5.3.2 Produkter

Den sparade informationen om de produkter som kaféet säljer består av fem fält. Dessa är produktnamnet, priset på produkten, en beskrivning av produkten, en länk till en bild av produkten som finns sparad på Firebase Storage samt namnet på bilden av produkten. Namnet på bilden sparas endast av de skälet att de gör det enklare att ta bort bilden från Firebase Storage om produkten skulle behöva tas bort. På grund av att informationen om kaféets produkter inte behöver samma skydd som till exempel användardata är det möjligt för alla användare att läsa produktinformationen. Att skriva data till produktinformationen är dock skyddat så att endast kaféägaren kan göra detta.

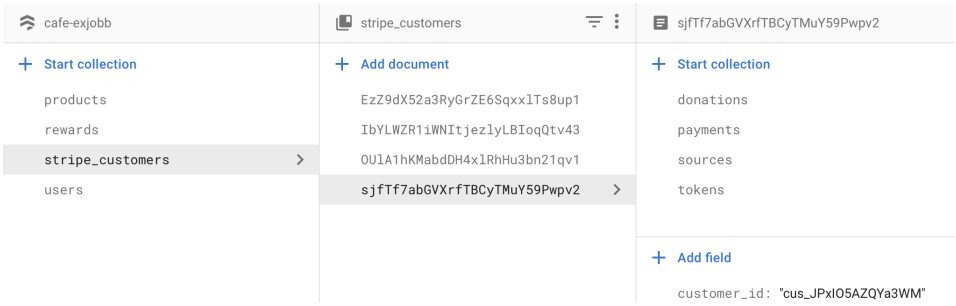
5.3.3 Gåvor

Informationen om gåvor består av tre fält samt den samlade informationen om produkten som gåvan består av. De tre fälten består av det minimala och maximala donations beloppet som en kund ska ha donerat för att göra sig berättigad till gåvan, samt produktnumret till produkten. Reglerna är samma som för produkter, nämligen att endast kaféägare kan skriva till databasen men alla användare kan läsa den.

5.3.4 Stripe-data

I samlingen gällande Stripe finns all den samlade informationen som på något sätt berör Stripe. Det finns ett fält med kundnummret som varje användare tilldelas av Stripe när de registrerar sig. Dessutom finns det fyra samlingar med mer specifik information. Dessa behandlar information om betalningar, donationer, Token objekt och Source objekt. Ett urklipp från databasen med information om Stripe finns att se i figur 5 Informationen gällande betalningar och donationer är endast tillgänglig för användarna själva och kaféägarna. Token objekten är endast tillgänglig för användaren och Source objekten är

reglerade för att ingen ska kunna läsa eller skriva till dem. Informationen om Source objekten behandlas endast av de serverfunktioner som sköter kommunikationen med Stripe och de står över de regler som sätts för databasen. Det behöver därför inte finnas några regler som ger åtkomst till Source objekten för någon.



Figur 6: Urklipp från Firebase Firestore med data rörande Stripe

5.4 Serverfunktioner

Av de serverfunktioner som utvecklades under projektets gång finns de två typer. Den första typen av funktioner är de som rör betalningar och därmed kommunikationen med Stripe. Den andra typen av funktioner hanterar borttagningen av gåvor och produkter från kaféets utbud.

5.4.1 Stripe-funktioner

Den viktigaste anledningen till att separera exekveringen av kod är för att i en säker exekveringsmiljö hantera betalningar genom Stripe.

Skapa Stripe kund
Den första funktionen som anropas när en användare registrerar sig är en funktion som registrerar användaren som en kund hos Stripe med den angivna e-postadressen. Detta görs för att eventuellt kommande betalningar ska kunna kopplas ihop mellan Stripe och Firestore för kunna behålla databasen konsekvent.

Skapa betalningskälla
När en användare lägger till sina kortuppgifter i mobilapplikationen skapas ett Token objekt med kortinformationen som uppdateras på databasen. När databasen uppdateras med Token objektet anropas en serverfunktion. Serverfunktionen skapar med hjälp av Token objektet ett Source objekt som fungerar som en betalningskälla. Source objektet sparas därefter på databasen för att användas vid kommande betalningar.

Skapa PaymentIntent
För att skapa ett PaymentIntent objekt för betalningar av köp och donationer används två funktioner. Dessa kallas på från mobilapplikationen genom HTTP anrop och returnerar en hemlighet som kan användas för att slutföra en betalning genom att ändra tillståndet på det skapade PaymentIntent objektet. Anledningen till att ha två olika funktioner för betalningar av varor och donationer är att de olika funktionerna sparar annorlunda information i de olika PaymentIntent objekten.

Slutförd betalning
När en betalning slutförs av Stripe anropas en serverfunktion av Stripe genom en ett HTTP anrop som sparar informationen om den slutförda betalningen på databasen. Funktionen kollar även om det är ett köp av en vara eller en donation och sparar informationen därefter på rätt ställe i databasen.

5.4.2 Ta bort gåvor och produkter

Förutom de funktioner som hanterar kommunikation med Stripe finns även två funktioner för att ta bort gåvor och produkter. Det räcker inte med att endast ta bort produkten eller gåvan från databasen då andra delar av databasen kan vara beroende av informationen som tas bort. För att ta bort en produkt måste varje gåva som kan ge produkten också tas bort för att inte det ska finnas kvar gåvor som lovar produkter som inte längre finns. För att ta bort gåvor måste även gåvorna tas bort från de personer som är berättigad till gåvan men som ännu inte hämtat ut den.

6 Diskussion

Projektet lyckades uppnå både det mål och syfte som sattes upp i inledningen. Det har funnits många idéer och tankar på resultatet och vad som kan utvecklas. Generellt för både mobil och webbapplikation är att en designer för användarupplevelse kan snygga till och förbättra stora delar av applikationerna. En neutral blå färg valdes som bakgrund med vit text. Detta valdes då ingen tidigare erfarenhet fanns av användardesign och det kändes neutralt. En funktion att köpa direkt från mobilapplikationen tillkom under utvecklingen. Detta var inte planerat från början men kändes som något naturligt att implementera eftersom funktionaliteten att betala fanns där för donationer. Något som tillkom under utvecklingen i webbapplikationen är möjligheten att exportera databasen till ett Excel dokument. Detta blev en efterfrågad funktion då det kan finnas intresse och behov av att på ett enkelt sätt få ut information ur databasen.

Under projektets gång har många idéer funnits som det inte fanns tid att implementera. För mobilapplikationen borde fler språk implementeras, just nu är engelska det enda språket. Andra identifieringsmetoder än QR har det även diskuterats om, som till exempel Bluetooth low energy (BLE) beacon men ansågs vara komplicerat för både kaféägare och kund. Utöver det borde fler betalningsmetoder implementeras så som Apple och Google Pay. Premunerationer genom Stripe är även en funktion som skulle kunna implementeras. För webbapplikationen skulle designen behöva ses över. Då vissa kaféer kanske skulle vilja använda sig av en mobiltelefon eller surfplatta för att använda applikationen kan det uppstå problem då den är designad för ett liggande format.

6.1 Frågeställningar

I inledningen utav rapporten presenterades två frågeställningar. Här kommer dessa frågeställningar diskuteras.

- Vad är en bra lösning för att identifiera användare som donerat?

För projektet anses QR vara en bra lösning att identifiera kunder med. Två alternativa lösningar är BLE beacon samt NFC men till skillnad från QR kräver båda alternativen specifik hårdvara för att fungera. Det krävs antingen en Bluetooth eller NFC enhet i mobilen för att använda dessa tekniker. Där har QR en fördel som endast kräver en skärm att visa upp koden på. BLE beacon och NFC kräver också att kaféägaren har specifik hårdvara så kunder kan identifiera sig. Det innebär mer hårdvara för kaféägaren och leder till den andra frågeställningen.

- Hur kan projektet utformas för att kräva minimalt med hårdvara för kaféägaren?

Som redogjorts för ovan är QR ett bra alternativ för identifering men minimerar också hårdvara som kaféägaren behöver. Firebase är en annan del av projektet som löser mycket utav hårdvaran då databas, server och webbapplikation ligger hos Firebase. Detta gör att den enda hårdvara som krävs är någon form utav dator, mobiltelefon eller surfplatta med kamera för identifiering.

6.2 Metodkritik

De teknologier och lösningar som använts i projektet har alla sina för och nackdelar. Denna sektionen tar upp de val som gjort och diskutera deras för och nackdelar med alternativa lösningar som skulle kunna användas.

6.2.1 React-Native

React-Native valdes eftersom en kod skrivs till både Android och iOS. Fördelen att använda React-Native är att tid sparas då endast en kodbas behöver utvecklas. Inlärningen blir även lättare med endast ett språk. Nackdelen är att många funktioner i applikationen som kommer från tjänster har färdig kod eller SDK:er till direkta programspråk, detta existerar inte alltid för React-Native. I stället blir projektet beroende av importerade bibliotek som ger samma funktionalitet. Dessa bibliotek kan vara utvecklade av vem som helst och det är olika hur bra underhållningen är av biblioteken. Ett exempel är Stripe som har SDK för Java och Swift men som inte existerade för React-Native vid projektets genomförande. I stället användes biblioteket tipsi-stripe. Det fanns ett problem att senaste versionen inte fungerade för iOS, så en gammal version fick användas i stället. Alternativet är att utveckla applikationen med direkta programspråk, Java eller Kotlin för Android och Swift för iOS. Fördelen med dessa är färdiga SDK:er och officiella bibliotek som kan implementeras direkt i koden. Nackdelen är att två kodbaser behöver underhållas.

6.2.2 Databas

För att inte behöva utveckla en egen server valdes Google Firebase som lösning för databas. Detta sparar utvecklingstid och kaféägare behöver inte hysa någon egen server. Nackdelen med Firebase är kostnaden som är oförutsägbar. Kostnaden varierar beroende på hur mycket trafik tjänsten använder. Det innebär att kostnaden ökar när fler vill använda applikationerna. En annan nackdel är att databasen har begränsad funktionalitet vad gäller hämtning av efterfrågad data. Ett alternativ är att utveckla en egen SQL databas som MySQL. Fördelen är kostnaden för mjukvaran och ett större stöd för

olika programspråk. Nackdelen är att databasen måste köras på en server som antingen hyrs eller köps in. Utöver kostnaden för hårdvaran måste även en server konfigureras på hårdvaran. SQL databaser kräver även mer planering eftersom data i en SQL databas har relationer till varandra. Det gör det svårt att göra större förändringar i databasen efter den fyllts med data.

6.3 Etik

Applikationerna hanterar och lagrar persondata vilket innebär att den europeiska dataskyddsförordningen (GDPR) måste följas[29]. Den data som lagras är e-post, för och efternamn men även kortuppgifter. Kortuppgifterna sparas inte direkt i databasen, i stället skickas kortinformationen till Stripe som validerar kortet och skickar tillbaka en Token. Det är denna Token som sparas och den innehåller ingen kritisk information om kortuppgifterna. Det är viktigt att följa GDPR och implementera system som följer de regler som finns. Ett sådant system är inte implementerat i det framtagna koncepttestet och är något som behöver implementeras. Det är även viktigt att tydligt redogöra för medlemmar hur deras data används och sparas. Detta då det är enkelt att missbruka den sparade informationen i databasen.

En annan viktig aspekt är den information applikationen kan samla in direkt från användarens telefon. I dagsläget samlas ingen information in från mobilen men det går att göra. Det är viktigt att användaren känner sig trygg i användandet av applikationen och att ingen data samlas in i ont syfte.

6.4 Hållbarhet och miljö

Inom hållbarhet finns tre dimensioner, den ekologiska, ekonomiska och den sociala[30]. Projektet berör delar av den ekonomiska och sociala dimensionen. Med en produktifiering av konceptet kan kaféer överleva med det extra ekonomiska stödet från kunder. Detta har en positiv effekt för ägare och medarbetare som kan fortsätta försörja sig med lön. Det påverkar även det sociala då många människor besöker kaféer för socialt umgänge. I dagsläget vidrör inte projektet den ekologiska dimensionen. En möjlig funktion som kan utvecklas i framtiden för miljön är att skänka överblivna bakelser för att minska matsvinnet.

7 Slutsats

Projektet lyckades utveckla ett fungerande koncepttest utefter det mål som sattes i inledningen. Firebase och QR gör att kaféägaren behöver minimalt med hårdvara medan QR också fungerar som en bra identifieringsmetod för kunder. Under utvecklingens gång lades fler funktioner till än vad som planerades från början. Dessa inkluderar bland annat möjligheten att köpa varor direkt från mobilapplikationen och visar att det finns stor potential i fortsatt utveckling utav konceptet.

7.1 Rekommendationer till fortsatt utveckling

Det första som bör prioriteras är att vidareutveckla databaslösningen för att skala med fler kaféer som vill ansluta sig. Detta kan göras genom att utöka den nuvarande lösningen eller byta databas helt till något annat. I framtiden kan mobilapplikationen utvecklas till att stödja alla kaféer som vill ansluta sig och sedan används samma applikation för alla kaféer. Användaren får då efter inloggning välja vilket kafé de vill stödja. Om projektet ska vidareutvecklas till en produkt behöver GDPR följas.

Referenser

[1] SCB. (2021). ”Handel och konsumtion,” [Online]. Tillgänglig: https://www.scb.se/hitta-statistik/temaomraden/sveriges-ekonomi/konjunkturindikatorer/handel-och-konsumtion/ (hämtad 2021-05-07).

[2] ——, (2021). ”Restaurangindex, utveckling totalt och per restaurangkategori,” [Online]. Tillgänglig: https://www.scb.se/hitta-statistik/statistik-efter-amne/handel-med-varor-och-tjanster/inrikeshandel/omsattning-inom-tjanstesektorn/pong/tabell-och-diagram/restaurangindex/restaurangindex-utveckling-totalt-och-per-restaurangkategori/ (hämtad 2021-05-04).

[3] Keyence. (2021). ”What is a QR code?” [Online]. Tillgänglig: https://www.keyence.com/ss/products/auto_id/barcode_lecture/basic_2d/qr/ (hämtad 2021-05-11).

[4] Stripe. (2021). ”Card authentication and 3D Secure,” [Online]. Tillgänglig: https://stripe.com/docs/payments/3d-secure (hämtad 2021-05-20).

[5] Firebase. (2021). ”Firebase Authentication,” [Online]. Tillgänglig: https://firebase.google.com/docs/auth (hämtad 2021-05-20).

[6] ——, (2021). ”Cloud Firestore,” [Online]. Tillgänglig: https://firebase.google.com/docs/firestore (hämtad 2021-05-20).

[7] ——, (2021). ”Cloud Storage for Firebase,” [Online]. Tillgänglig: https://firebase.google.com/docs/storage (hämtad 2021-05-20).

[8] ——, (2021). ”Cloud Functions for Firebase,” [Online]. Tillgänglig: https://firebase.google.com/docs/functions (hämtad 2021-05-20).

[9] ——, (2021). ”Firebase Hosting,” [Online]. Tillgänglig: https://firebase.google.com/docs/hosting (hämtad 2021-05-20).

[10] Stripe. (2021). ”Online payment processing for internet businesses,” [Online]. Tillgänglig: https://stripe.com/en-se (hämtad 2021-05-20).

[11] ——, (2021). ”Tokens,” [Online]. Tillgänglig: https://stripe.com/docs/api/tokens (hämtad 2021-05-20).

[12] ——, (2021). ”Sources,” [Online]. Tillgänglig: https://stripe.com/docs/api/sources (hämtad 2021-05-20).

[13] ——, (2021). ”PaymentIntents,” [Online]. Tillgänglig: https://stripe.com/docs/api/payment_intents (hämtad 2021-05-20).

[14] Facebook Inc. (2021). ”React,” [Online]. Tillgänglig: https://reactjs.org/ (hämtad 2021-04-27).

[15] ——, (2021). ”React Native,” [Online]. Tillgänglig: https://reactnative.dev/ (hämtad 2021-04-27).

[16] react-bootstrap. (2021). ”react-bootstrap,” [Online]. Tillgänglig: https://github.com/react-bootstrap/react-bootstrap (hämtad 2021-05-23).

[17] firebase. (2021). ”firebase-js-sdk,” [Online]. Tillgänglig: https://github.com/firebase/firebase-js-sdk (hämtad 2021-05-23).

[18] JodusNodus. (2021). ”react-qr-reader,” [Online]. Tillgänglig: https://github.com/JodusNodus/react-qr-reader (hämtad 2021-05-23).

[19] i18next. (2021). ”i18next,” [Online]. Tillgänglig: https://github.com/i18next/i18next (hämtad 2021-05-23).

[20] ——, (2021). ”react-i18next,” [Online]. Tillgänglig: https://github.com/i18next/react-i18next (hämtad 2021-05-23).

[21] ——, (2021). ”i18next-http-backend,” [Online]. Tillgänglig: https://github.com/i18next/i18next-http-backend (hämtad 2021-05-23).

[22] js-cookie. (2021). ”js-cookie,” [Online]. Tillgänglig: https://github.com/js-cookie/js-cookie (hämtad 2021-05-23).

[23] ReactTraining. (2021). ”react-router,” [Online]. Tillgänglig: https://github.com/ReactTraining/react-router (hämtad 2021-05-23).

[24] rdcalle. (2021). ”react-export-excel,” [Online]. Tillgänglig: https://github.com/rdcalle/react-export-excel (hämtad 2021-05-23).

[25] Invertase. (2021). ”react-native-firebase,” [Online]. Tillgänglig: https://github.com/invertase/react-native-firebase (hämtad 2021-05-20).

[26] react-navigation. (2021). ”react-navigation,” [Online]. Tillgänglig: https://github.com/react-navigation/react-navigation (hämtad 2021-05-20).

[27] awesomejerry. (2021). ”react-native-qrcode-svg,” [Online]. Tillgänglig: https://github.com/awesomejerry/react-native-qrcode-svg (hämtad 2021-05-23).

[28] tipsi. (2021). ”tipsi-stripe,” [Online]. Tillgänglig: https://github.com/tipsi/tipsi-stripe (hämtad 2021-05-23).

[29] Europeiska unionen. (2021). ”Uppgiftsskydd enligt GDPR,” [Online]. Tillgänglig: https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_sv.htm (hämtad 2021-05-20).

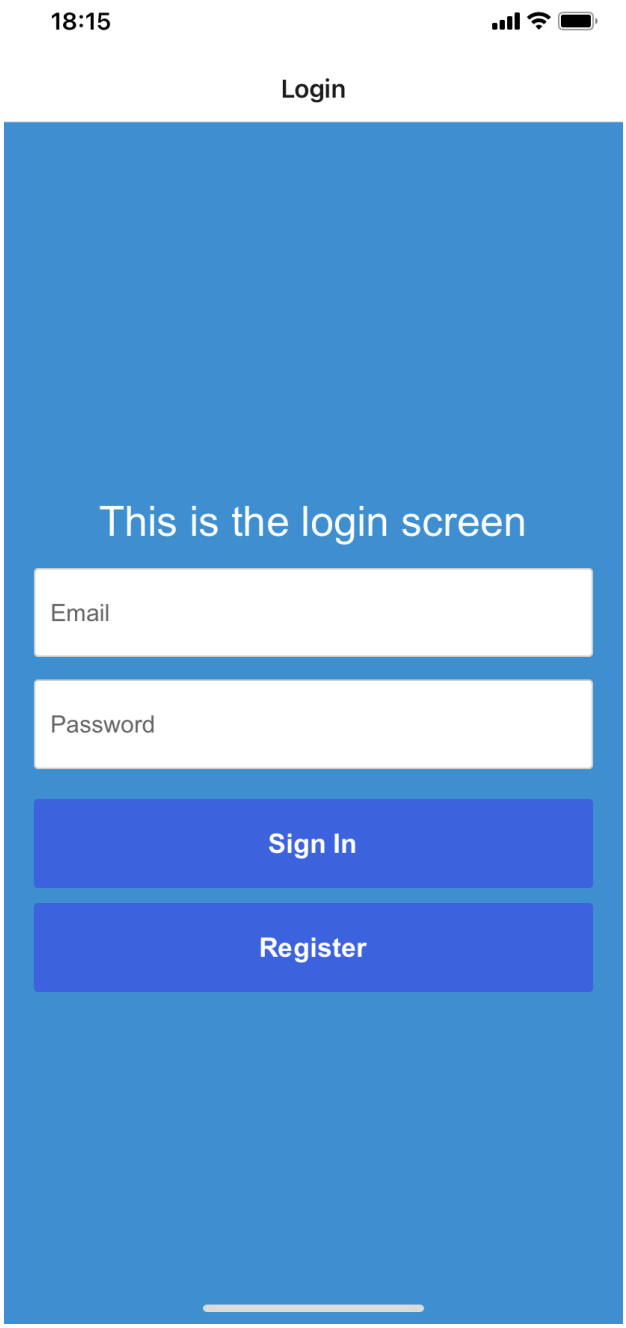
[30] F. Hedenus, M. Persson, F. Sprei, *Hållbar utveckling - nyanser och tolkningar*, 1:4 uppl. Lund. Sverige: Studentlitteratur AB, 2018, s. 31–43.

A Kod

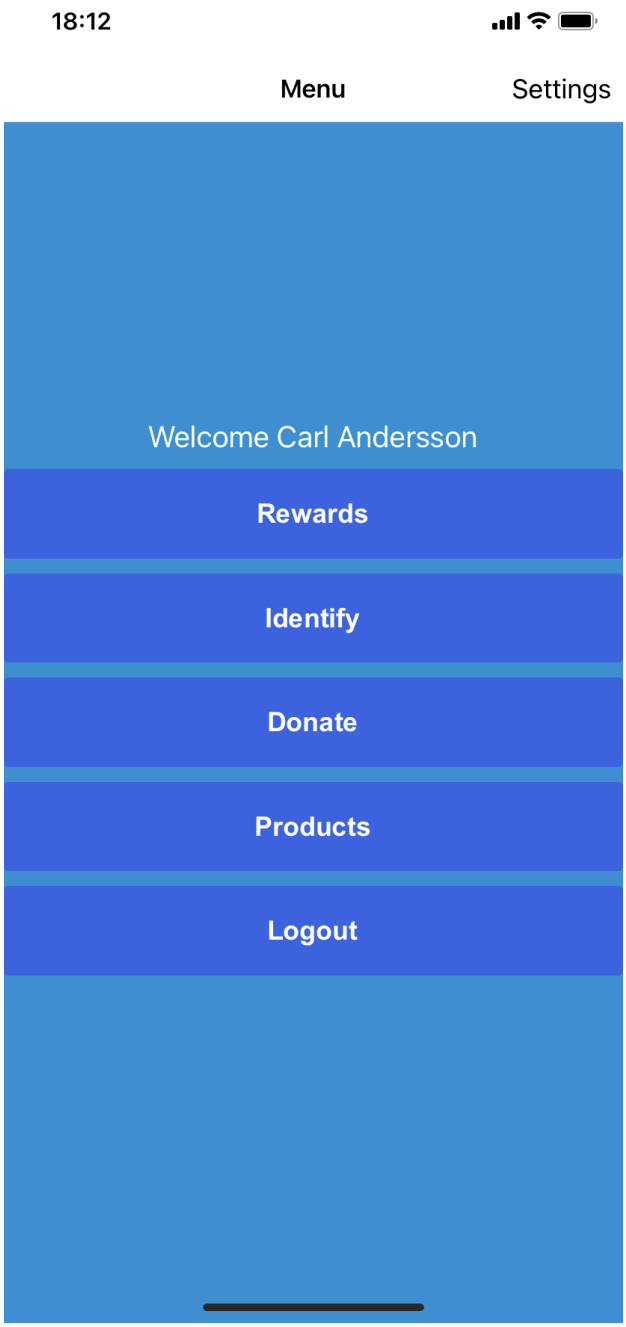
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId || get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
    }
    match /products/{productId} {
      allow read: if request.auth != null;
      allow write: if request.auth != null && get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
    }
    match /rewards/{userId} {
      allow read: if request.auth != null;
      allow write: if request.auth != null && get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
    }
    match /stripe_customers/{customerId} {
      allow read: if request.auth != null && request.auth.uid == customerId || get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
      allow write: if false;

      match /tokens/{tokenId} {
        allow read, write: if request.auth != null && request.auth.uid == tokenId;
      }
      match /sources/{sourceId} {
        allow read, write: if false;
      }
      match /donations/{donationId} {
        allow read: if request.auth != null && get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
        allow write: if false;
      }
      match /payments/{paymentId} {
        allow read: if request.auth != null && get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
        allow write: if get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;
      }
    }
  }
}
```

Figur A.1: Regler för databasen på Firebase Firestore

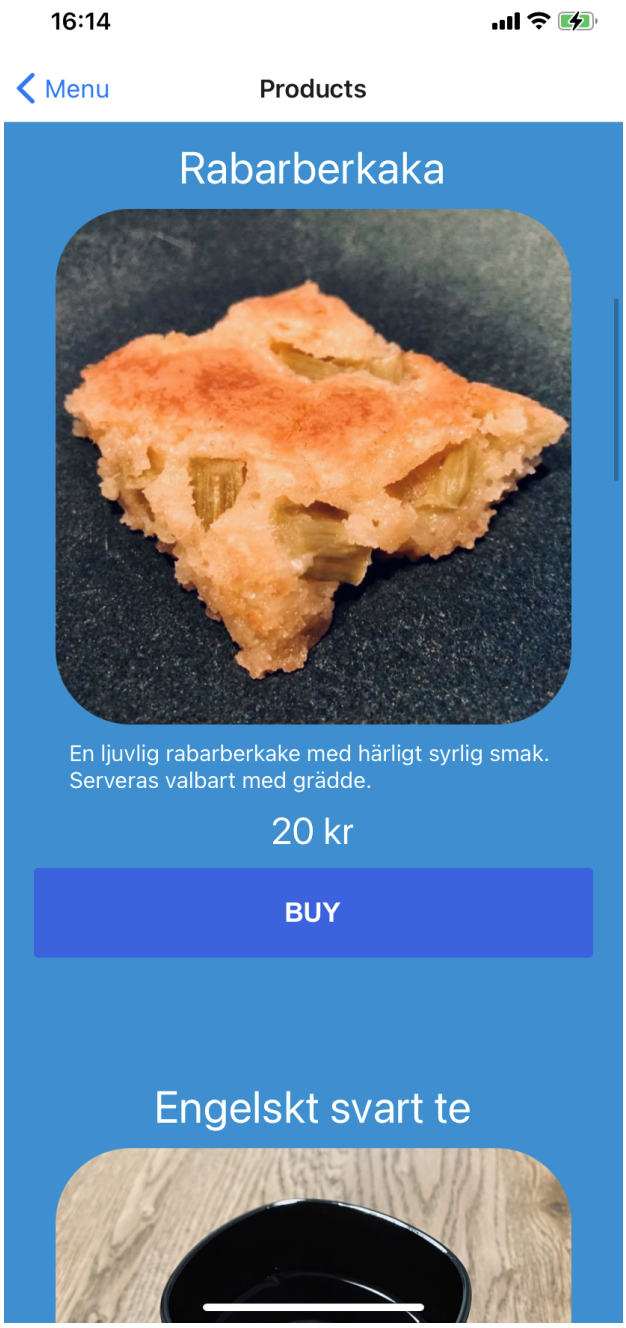
B Bilder



Figur B.1: Inloggningsskärm för mobilapplikationen

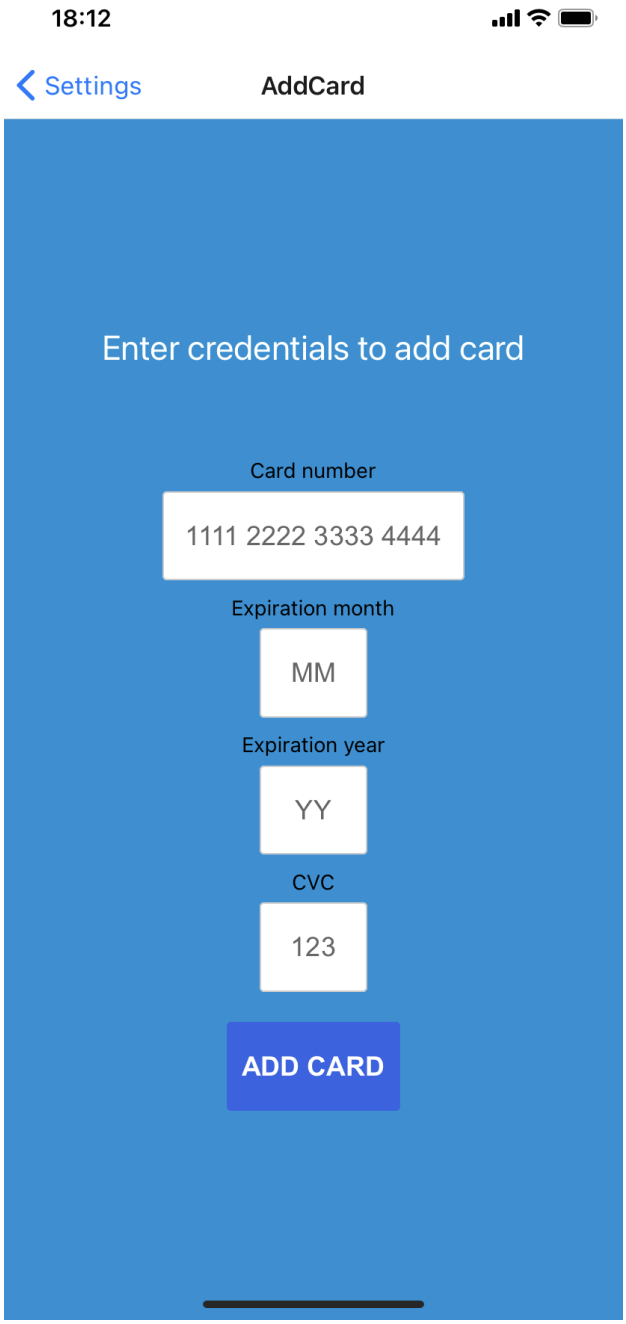


Figur B.2: Hemskärm för mobilapplikationen

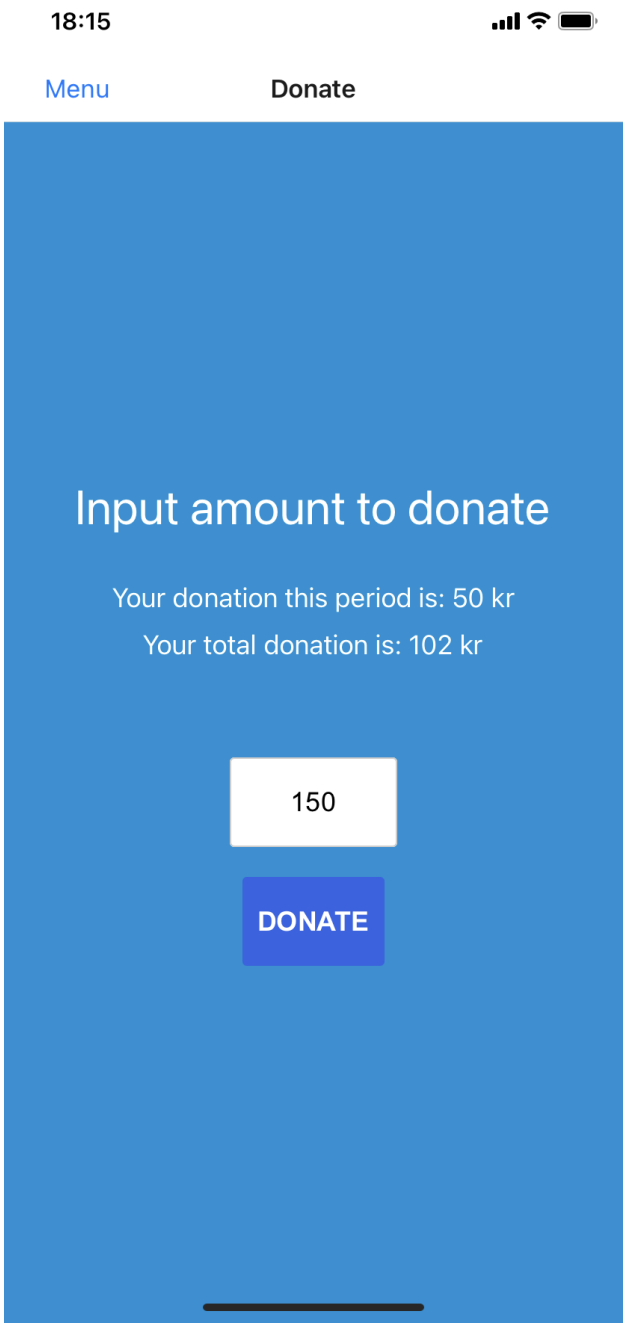


Figur B.3: Produktskärm för mobilapplikationen

Figur B.4: Gåvoskärm för mobilapplikationen



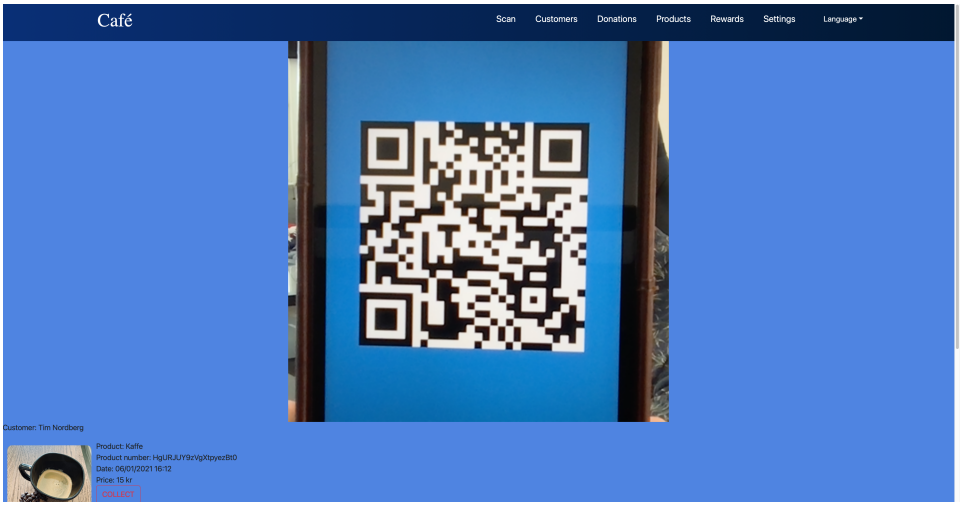
Figur B.5: Skärm för att lägga till kortuppgifter i mobilapplikationen



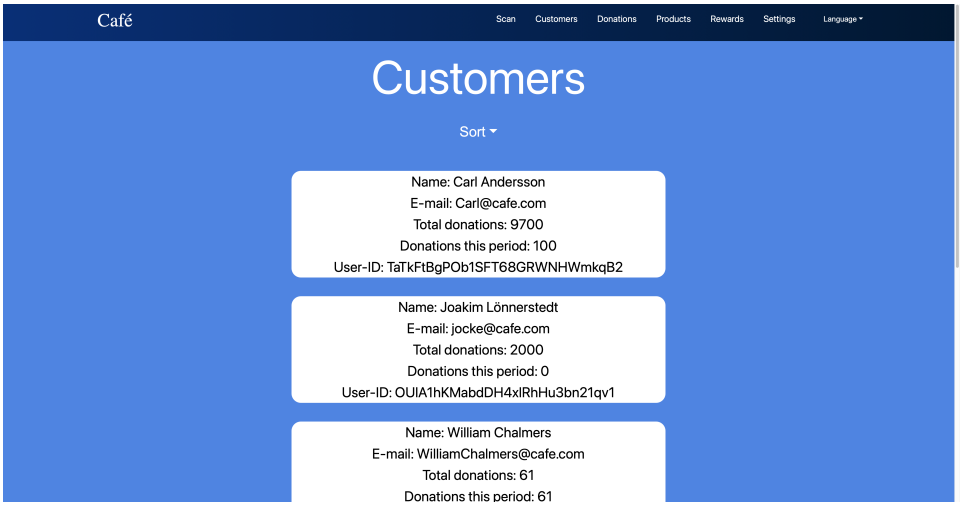
Figur B.6: Donationsskärm för mobilapplikationen

Figur B.7: Köphistoriksskärm för mobilapplikationen

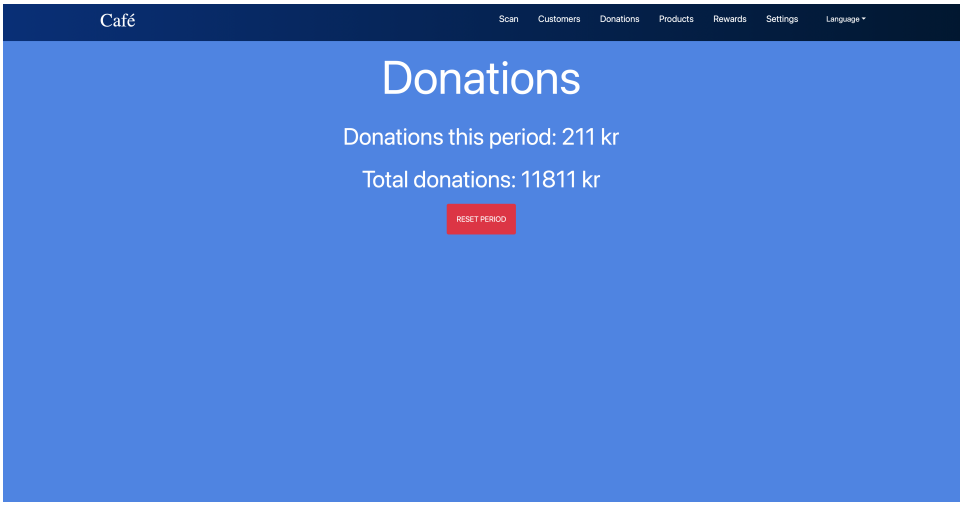
Figur B.8: Identifiering med QR-kod på mobilapplikationen



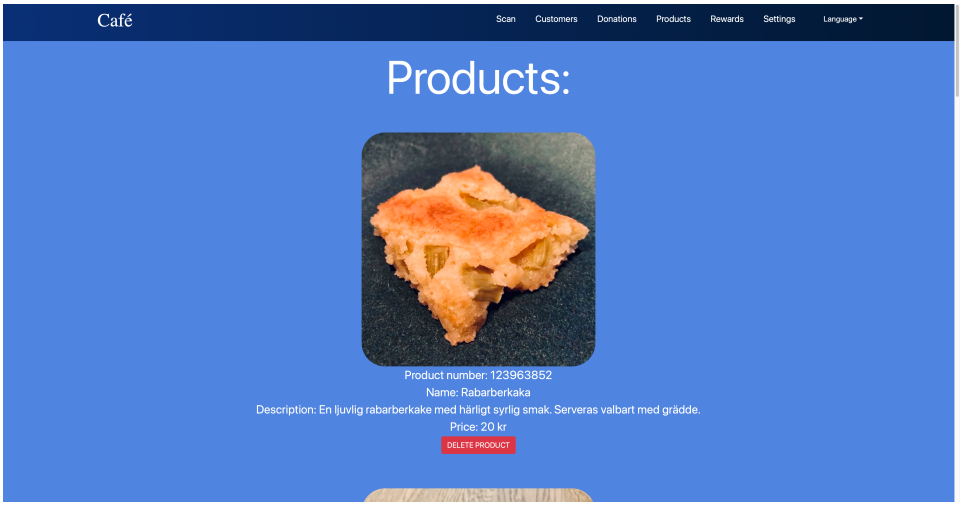
Figur B.9: Skanningsskärm på webbapplikationen



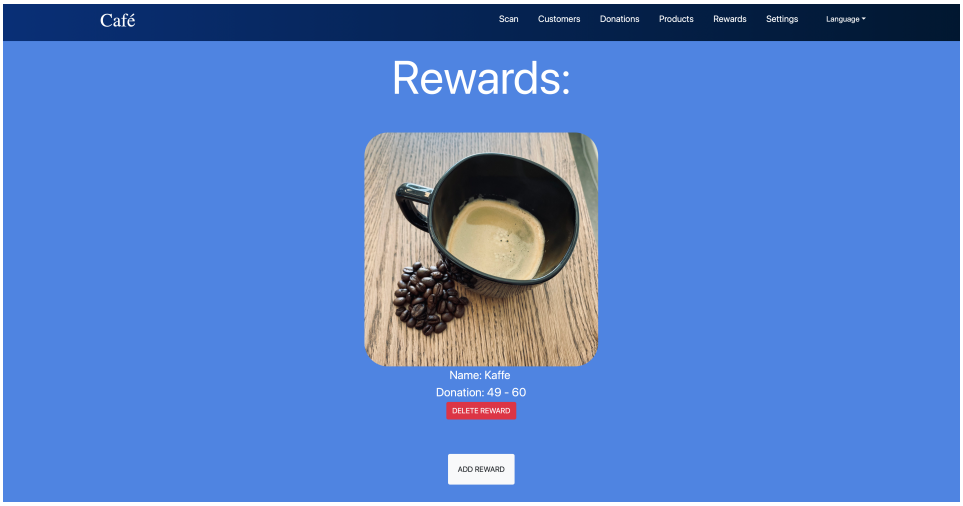
Figur B.10: Skärm för kunduppgifter på webbapplikationen



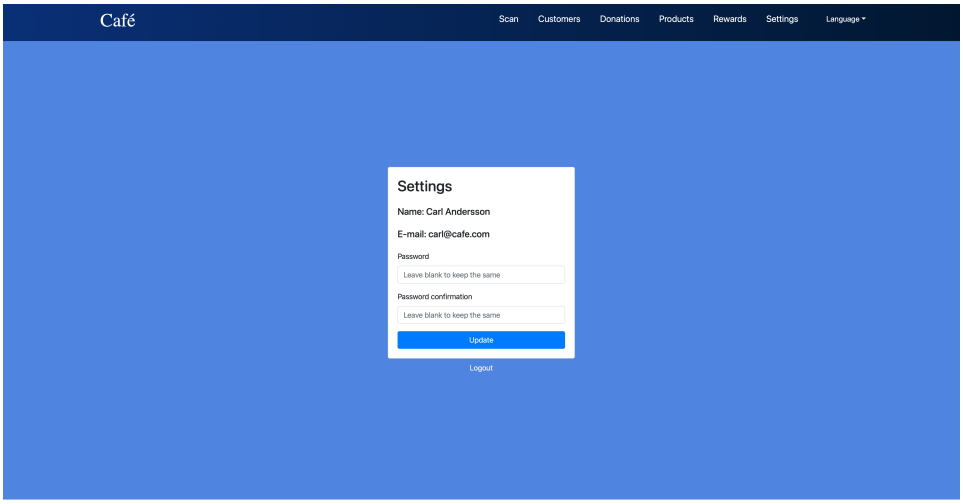
Figur B.11: Skärm för information om donationer på webbapplikationen



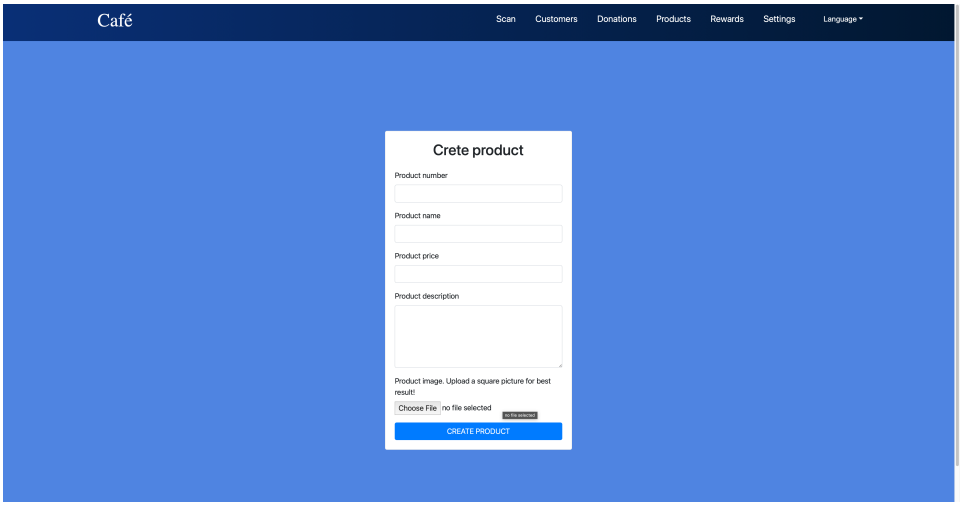
Figur B.12: Skärm för produktinformation på webbapplikationen



Figur B.13: Skärm för gåvoinformation på webbapplikationen



Figur B.14: Skärm för kontoinformation på webbapplikationen



Figur B.15: Skärm för skapandet av produkt på webbapplikationen