

Service Design - C.R.API a Coffee Reviewing API

En case-studie



Studententer:

Ali Chehade

Simon Martinsson

Examinator:

Andreas Nilsson

Innehåll

Inledning.....	2
Arbetsfördelning.....	2
Resultat.....	2
Systembeskrivning.....	2
URL Routing Server.....	2
Web Scraper.....	3
Databaser.....	3
Service Design.....	3
Funktionalitet utifrån servicedesign.....	4
Diskussion.....	4
Bilagor.....	5
Bilaga A1: API - Systemöversikt.....	5
Bilaga A2: API Path översikt.....	6
Bilaga B1: Double Diamond servicedesign.....	7
Bilaga B2: Personas.....	8
Bilaga B3: Journey Maps.....	9

Inledning

Denna rapport täcker en case-studie inom Service Design, där API-struktur och programmering används som verktyg för att tillämpa teorier för att konstruera ett tekniskt serviceinriktat system. Vårt API backend-system är inriktat mot klient-servrar och applikationer som vill presentera information om svenska kaffeprodukter, samt kunna recensera dessa genom ett användarkonto. Klienter interagerar med vår service genom att skicka förfrågningar till vår URL-routing-server som hanterar och behandlar begäran.

API-servicen är främst riktad mot en kaffeälskande målgrupp som vill upptäcka nya kaffeprodukter och kunna dokumentera sina intryck av dessa. Men vi har också utvecklat andra "personas" där produktionen av kaffe är i fokus och där tillverkarna av kaffe också kan gå in och läsa vad kunderna anser om deras produkter. Vi evaluerar hur servicedesignen för vårt API är uppbyggd genom att följer de olika "personas" väg genom vårt system och undersöker hur väl vår design möter deras behov.

Arbetsfördelning

Den strukturella uppbyggnaden av projektet har diskuterats och utvecklats genom hela projektets gång, där båda har varit involverade. Det praktiska arbetet har sedan utförts främst i par men där vi koncentrerat oss på olika delar av projektet. Vi har dock alltid hjälpt varandra genom att bolla idéer och tankar om hur vi ska röra oss framåt. De områden som har haft en mer distinkt uppdelning är dels web scraping, där Ali har lagt mest fokus och varit huvudutvecklaren, samt end points där Simon varit mest drivande. Men generellt har vi haft en jämn och dynamisk arbetsprocess genom hela arbetet.

Resultat

Produkten av vårt case-arbete är en servicedesignad RESTful API-server för kaffe-recensioner. API-servern möjliggör för klienter att hämta, spara, uppdatera eller radera information om kaffeprodukter eller ett registrerat konto med tillhörande kaffe-recensioner. Nedan beskrivs systemets olika komponenter i utförligare teknisk detalj. Vi går också igenom hur vi designat servicen utifrån olika potentiella användare och beskriver strukturen för vår servicedesign.

Systembeskrivning

Systemet är uppdelat i fyra större komponenter. En URL-routing-servern som hanterar klient-förfrågningar, en SQLite databas som innehåller information om användare och recensioner, en JSON-databas som innehåller information om kaffeprodukter samt en webbscrapnings-algoritm för att hämta hem information om kaffe-produkterna. För mer detaljerad översikt om hur komponenterna hänger ihop, gå till [Bilaga A1: API - Systemöversikt](#).

URL Routing Server

Vår routing-servern som interagerar utåt mot klienten är programmerad i Flask v2.2.3 med python 3.11 i basen. Klienter kan skicka giltiga HTTP-förfrågningarna hit för att utvinna information från våra databaser. De giltiga HTTP-förfrågningarna har strukturerats och dokumenterats i en YAML-fil som klienten kommer åt genom att skicka `"/coffee/docs"` i en HTTP GET-förfrågan till servern. YAML-filen öppnas då och presenteras interaktivt med hjälp av verktyget Swagger UI. I denna rapport är en förenklad lista över de olika URL-paths och förfrågningarna som klienten kan utföra, bifogade i [Bilaga](#)

[A2: API Path översikt](#). Servern är designad enligt en RESTful arkitektur och hanterar varje förfrågan för sig, utan att spara eller spåra klientens användning.

Vid serverns uppstart initieras en SQLite-databas för datalagring av, en del användarkonto och deras information och en del kaffe-recensioner. Vid första användning initieras även databasen innehållande kaffe-produkterna, genom att vår web scraping-algoritm startas och fyller en JSON-fil med information. Efter initiering sköts all kommunikation med databaserna genom vårt API som hanterar klientens informationsflöde i bakgrunden.

Web Scraper

För att få fram olika kaffeprodukter så har web scraping använts då en API för kaffe inte har hittats. Med hjälp av Pythons bibliotek Scrapy samt Selenium har en så kallad crawler skapats för att hämta alla produkter från en svensk webbplats, <https://www.dittsvenskaskafferi.se/svensk-mat/dryck/kaffe>. Crawlern har byggts med hjälp av Selenium för att hantera den Javascript-baserade scrollfunktionen hemsidan har. Produktlistan som visas, uppdateras med nya kaffe-förslag först när man kommer till slutet av hemsidan, vilket gjort att vi inte kunnat endast använda scrapy för att hämta alla produkter på sidan.

För att få fram rätt information som ska hämtas, inspekteras webbsidan i webbläsaren för varje HTML-element som har varit relevant och korrekta HTML referenser har hittats. För detta projekt valde vi att endast hämta hem produkternas namn, deras URL. Vi har även valt att lägga till ett unikt ID för varje produkt, som stöd vid produktsökning. Varje ID är valt från produktens URLs sista 13 tecken, vilket i de flesta fall representerar artikelnummer för kaffet på sidan. För varje produkt som hämtas sparas detta i en JSON fil som en lista där varje element är en *dictionary* med nyckelorden "name", "url" och "id".

Databaser

För detta projekt skapades två databaser. Den första är en samling av alla kaffeprodukter som har skapats i en JSON fil med hjälp av web scraping, den andra är en databas skapad i SQLite. I JSON-databasen kommer vi åt information om kaffeprodukter, deras URLer samt ett unikt ID.

SQLite databasen innehåller all data för användare. En tabell för *users* har skapats för att lagra varje användares användarnamn, unikt ID samt registreringsdatum. Vi lagrar också känslig information om användare, så som e-mail och autentiseringstoken, i en separat tabell. Recensioner som skapas av användare läggs i tabellen för "reviews". En "review" har ett unikt ID, en tidsstämpel när recensionen är gjord, själva recensions-texten samt ID för användare och kaffe som recensionen är relaterad till.

Service Design

När vi designat vårt API-system har vi tillämpat olika servicedesign-teorier. Vi har följt "Double Diamond"-modellen som utgångspunkt för att designa systemet ([Bilaga B1: Double Diamond servicedesign](#)). Under *Discovery*-steget formade vi idén och bestämde oss för en API-tjänst riktad till att recensera olika produkter. Den idén utvecklades sedan och begränsades till kaffeprodukter när vi insåg att det var något som inte fanns tillgängligt (utifrån vår bakgrundsundersökning).

Därifrån definierades sedan de olika behov vi ville möta och vilka personer som kan tänkas ha dessa behov. Här karaktäriserades två "Personas" ([Bilaga B2: Personas](#)), en applikationsutvecklare som älskar att dricka kaffe och en produktutvecklare som jobbar för ett kaffeföretag. Vi insåg då att dessa personer har två olika behovsbilder. En som endast är ute efter informationen i vår tjänst och en som

faktiskt vill registrera ett konto och ha möjlighet att recensera kaffeprodukter. Under utvecklingen av vår API, grupperades därför tjänsten i två grupper. En URL-grupp som kräver en autentiseringsprocess och en del där klienten kan hämta icke-känslig data utan att skapa ett konto.

För att fylla båda *personas* behov, skapade vi sedan "*Journey Maps*" ([Bilaga B3: Journey Maps](#)) som stöd för att strukturera upp de URL-routes som krävdes i vår service. Specifikt ville vi få fram två helt skilda vägar genom vår API-tjänst. Bådas väg smälter samman en aning på några touchpoints, men ur produktutvecklarens synvinkel är åtkomsten av information den centrala delen och inte att kunna interagera med eller spara någon personlig data.

Funktionalitet utifrån servicedesign

Utifrån den resulterande funktionaliteten i vår service har vi lyckats möta de behov som vi initialt utgick ifrån. Den slutgiltiga systemöverblicken hittas i [Bilaga A2: API Path översikt](#), där end points:en för "*Authorized usage*" kräver att man registrerar ett konto och får en personlig *token* för utökad användning. Detta fyller applikationsutvecklarens behov av att kunna hålla kvar sina användares noteringar om produkterna. Vi tillåter alltså klienten att skapa och behålla kaffe-recensioner endast om de har skapat en användarprofil. Men för vår andra persona, produktutvecklare, går det att hämta hem recensioner relaterade till en specifik produkt utan att behöva registrera konto.

Diskussion

Sammanfattningsvis har vi skapat en fungerande API-service som tillåter klienter att hämta hem information om svenska kaffeprodukter. De kan också skapa ett konto kopplat till API:et för att recensera specifika produkter. Dessa recensioner går sedan att hitta genom att söka på användare eller kaffeprodukt. All klientinteraktion med service sker genom en central routingsserver som sedan i bakgrunden kommunicerar med olika tjänster för att förse klienten med data.

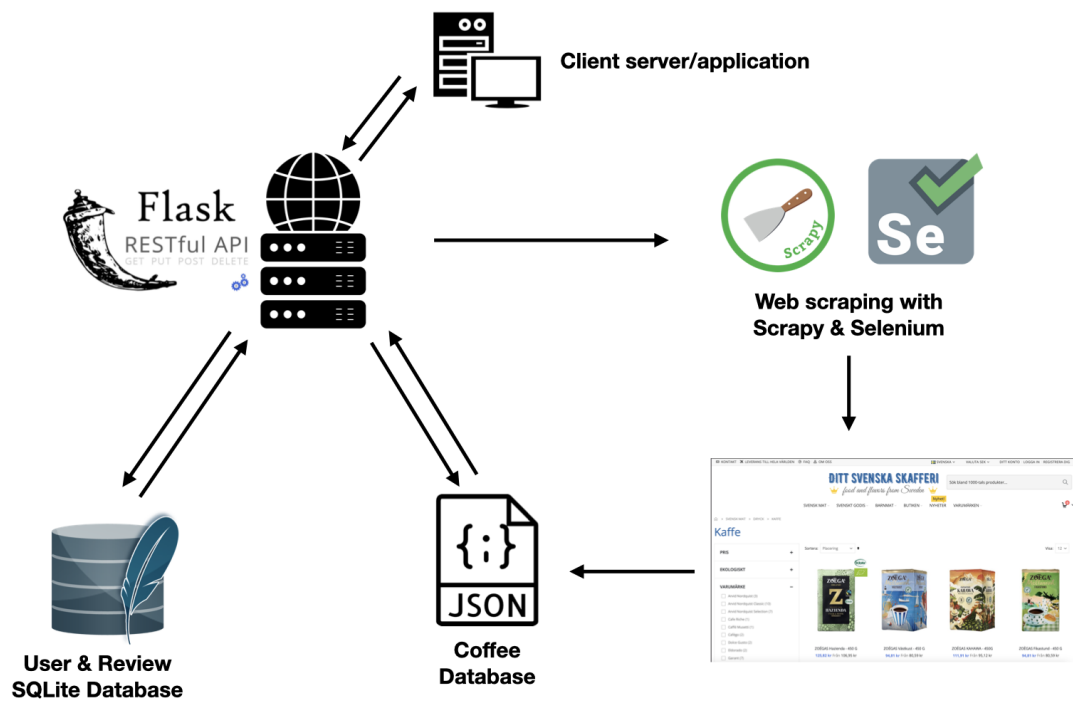
Vår främsta begränsning var att vi tyvärr inte hittade ett färdigt kaffe-API, utan fick hämta hem alla produkter genom att web scrape:a en redan existerande kaffeproduktsida. Detta var dels en utmanande uppgift, men öppnar också upp för att utökas genom att antingen scrape:a fler kaffeprodukt-sidor eller (om ett kaffe-API görs i framtiden) implementera tjänsten med ett färdigt kaffe-API.

Strukturellt skulle vi också kunna utöka vår filtrering av recensionerna. En utökad recensions-struktur i vår databas som också inkluderar tex. "grade" skulle kunna hjälpa oss hitta populära kaffeprodukter. Eller också sammanfatta en utförligare produktutvecklingsrapport för våra klienter som är kaffeföretag. Vi kanske också vi hålla strukturen för våra objekt i kaffedatabasen lite mer specifika genom att lägga till fält för "brand", "image", "description" osv. Det skulle kräva en mer sofistikerad web scrape-algoritm som bara rent tidsmässigt blev begränsad i detta projekt.


En sista tanke för förbättring rör säkerhetsaspekter av API:et. Eftersom vi förser en nyligen skapad användare med ett unikt token för autentisering, vill vi kunna förmedla detta på ett säkrare sätt. Tanken från början med att ange en tillhörande e-mail vid registrering, var att kunna skicka ut den personliga token:en direkt till deras mail. En sådan funktion skulle uppgradera säkerheten i API-tjänsten och göra det svårare för avlyssnare att komma åt den personliga datan.

Bilagor

Bilaga A1: API - Systemöversikt



Bilaga A2: API Path översikt

 **Swagger**
Powered by SMARTBEAR

/static/coffeereviews.yaml

Explore

Coffee Review API ^{1.0} ^{OAS3}

/static/coffeereviews.yaml

This API allows you to create a user profile where you can write reviews of different coffees that you have tried. For now the brands and products are from the swedish selection.

Servers

http://localhost:5000

Authorized usage

API URL where authorization is required.

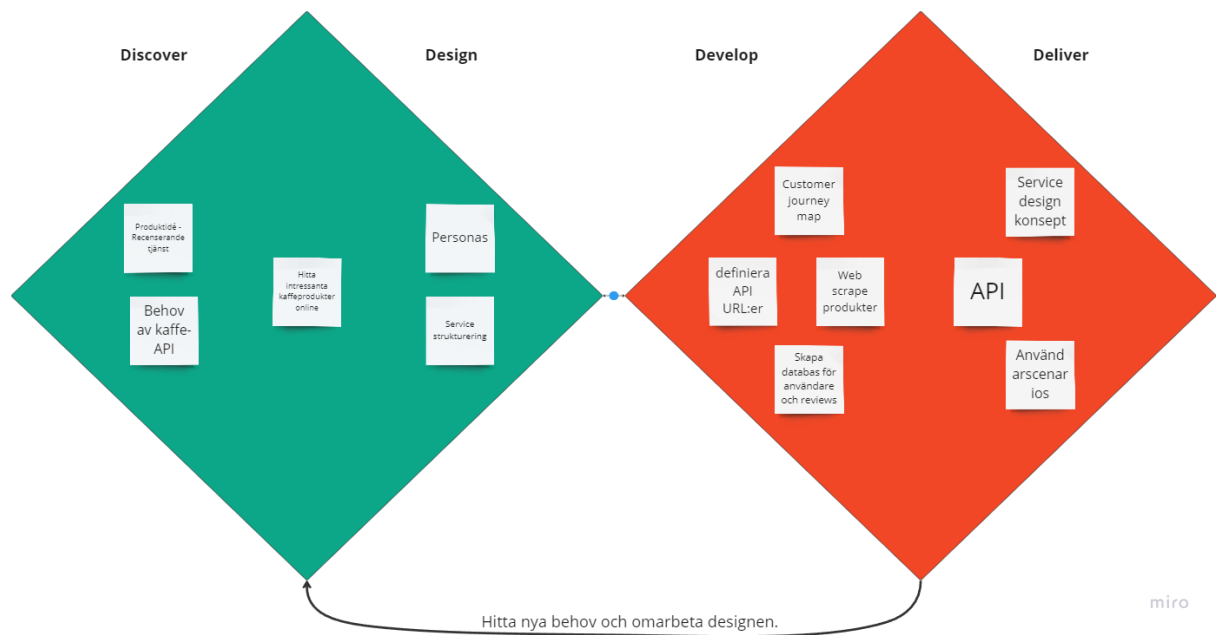
- PUT** /coffee/reviews/{review_id} Update a review
- DELETE** /coffee/reviews/{review_id} Delete a review
- POST** /coffee/{coffee_id} Post a new review
- GET** /users Return all users information
- GET** /users/{userId} Return user information
- PUT** /users/reviews/{review_id} Update a review
- DELETE** /users/reviews/{review_id} Delete a review

Standard usage

API URL available without authorization.

- GET** /coffee Return all coffee products
- GET** /coffee/reviews Return all coffee reviews
- GET** /coffee/reviews/{review_id} Review modification request
- GET** /coffee/search Search for a coffee product
- GET** /coffee/{coffee_id} Return a coffee product or post review
- GET** /coffee/{coffee_id}/reviews Return reviews for a coffee product
- POST** /users/register Create New User
- GET** /users/reviews Return all coffee reviews
- GET** /users/{user_id}/reviews Return reviews posted by a user
- GET** /users/reviews/{review_id} Review modification request

Bilaga B1: Double Diamond servicedesign.



Bilaga B2: Personas



Leslie O'Coffie

Programutvecklare

Kaffe is love, kaffe is life. Dricker all typ av kaffe och anser det som en nödvändighet för att ta sig igenom dagen. Har en egen blogg om olika kaffesorter som hon testat.

Mina Mål

Testa all kaffe i världen

Utveckla en app för kaffeälskare

Dela med sig av erfarenheter om olika kaffesorter

Personlighet

Hyper

Går inte att ta sig genom dagen utan kaffe

Delar med sig av sina tankar och åsikter

Beskrivning

Är lite nördigt lagd och gillar att betygsätta och recensera produkter hon köper och använder.

Orolig för / Jobbigt

Kaffet har tagit slut

Försova sig

Deadlines

Fastna i möte under fikastunden

miro



Kappe Chino

Produktutvecklare - Zoegas

En passion och nördigt intresse av kaffets tillverkningsprocess, i kombination med ett intresse för smaker, gör Kappe nyfiken på hur kaffets smak kan utvecklas mer.

Mina Mål

Hitta nya smaker inom kaffeindustrin.

Se till så att företaget utvecklar sina produkter enligt kundens önskemål.

Lägga upp strategier för företagets framtida kaffeutveckling.

Besöka kaffeodlingar och hitta nya leverantörer av bönor.

Personlighet

Nyfiken och frågvis

Ambitös i sin roll.

Tar alltid en ny sorts kopp kaffe på jobbet.

Vill vara effektiv i sitt arbete och hitta snabba åtgärder.

Beskrivning

Har en bakgrund som sommelier. Hans välutvecklade palett tillåter honom att definiera och upptäcka de mest distinkta variationer vid en avsmakning. Stor erfarenhet inom produktutveckling.

Orolig för / Jobbigt

Koffeinberoende.

Att inte veta kundens syn på företagets produkter.

Vill inte komma tomhänt till företagets nästa produktmöte

miro

Bilaga B3: Journey Maps

Leslie O'Coffie:



Kappe Chino:

