Projektrapport

WineOcular

2020-01-11

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

DA358A Datavetenskap: Storskaliga webbtjänster

Innehåll

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

1. Om tjänsten

Mat och vin går ofta hand i hand, och för att optimera smakupplevelsen gäller det att välja rätt kombination. WineOcular är en tjänst för vinälskare som söker efter recept till sitt vin. Kanske vill användaren ha ett recept som passar till favoritvinet, eller kanske har användaren en flaska vin hemma och vill ha hjälp med att hitta recept på maträtter som passar just det vinet. Det som gör WineOcular unikt är att tjänsten även anpassar receptförslagen utifrån vilka råvaror som är i säsong just nu.

Att välja recept utifrån säsong har många fördelar. Många råvaror smakar bäst när de är i säsong, de som har ätit en solmogen tomat nyplockad från kvisten kan nog stämma in i att det är en helt annan upplevelse än att äta en importerad, växthusodlad tomat under vintersäsongen. Råvaror som får mogna på egen hand innehåller dessutom mer näring. Att välja råvaror i säsong är också ekonomiskt, både för plånboken och miljön. Importerade och växthusodlade råvaror produceras och fraktas ofta på sätt som är dåliga för miljön.

Med WineOcular kan användaren få hjälp att hitta den optimala kombinationen av vin och mat, utifrån när råvarorna är som allra bäst. Vi på WineOcular har valt ut de vanligaste vita och röda druvorna i vin för dryck med mat. Användaren väljer druva och får ett antal förslag på recept som dels passar till druvan, och dels använder råvaror i säsong. Härifrån kan användaren välja ut ett recept och få fullständig beskrivning av receptet.

2. Den tekniska lösningen

I detta avsnitt presenterar vi hur WineOcular fungerar, vi tar avstamp i det användaren möter och beskriver hur detta möjliggörs. Lager för lager avslöjas och leder fram till WineOculars kärna, som inte är synlig för användaren. Användaren ska inte behöva bry sig om hur WineOcular skapar sin magi. En sak är dock säker, WineOcular hade inte varit möjligt utan externa resurser. Isaac Newton skrev "If I have seen further, it is by standing upon the shoulders of giants" [1], och detta är är sant även för WineOcular.

2.1. FrontEnd

WineOcular är en webbtjänst som görs tillgänglig för användare genom en hemsida, det är hemsidan som är WineOculars ansikte utåt. Hemsidan är utvecklad med de tre grundstenarna inom webbdesign; HTML, CSS och JavaScript. Det är med hjälp av dessa tre språk WineOculars tjänster görs tillgängliga för användaren, och det är dessa komponenter som utgör WineOculars FrontEnd.

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

WineOcular har utvecklat så kallade templates i HTML, för att skapa en struktur för hur hemsidan presenteras och med hjälp av CSS har hemsidan givits färg och form. För att ge liv till detta används JavaScript, det är det som möjliggör interaktion mellan användare och WineOcular. JavaScript fungerar som en brygga mellan WineOculars FrontEnd och kärnan. Kommunikationen mellan FrontEnd och kärnan sker med hjälp av JSON, som står för JavaScript Object Notation [2] och är ett format för JavaScript att hantera objekt. Objekten innehåller information, som FrontEnd hämtar från kärnan och presenterar på hemsidan. JavaScript hanterar de funktioner som användarens interaktion initierar genom att reagera på användarens interaktion, för att sedan hämta information och presentera denna.

2.2. Backend

WineOculars kärna är den del av tjänsten som utför logiken, och det är denna del som utgör WineOculars BackEnd. Till BackEnd hör alla de tekniker och tjänster som WineOcular använder för att leverera sin funktion till frontend. Det som kanske är mest centralt för WineOculars funktion är API. För att skapa WineOcular har vi utvecklat ett eget API som består av en mashup av två externa API. Vi har använt Säsongsmat API för att hämta information om råvaror i säsong, och spoonacular API för att matcha vin och mat, samt hämta recept.

Detta hanteras med en Java-applikation som består av tre huvudsakliga delar. Den första delen är den del som hämtar relevant information från de externa APIerna. Den andra delen omvandlar informationen till Java-objekt för att kunna skapa en anpassad struktur som är unik för WineOcular. Den tredje delen omvandlar i sin tur detta till information som kan presenteras på hemsidan, och gör det tillgängligt genom vårt eget API.

2.2.1 Externa API

Säsongsmat och klassen FoodInSeason

För att lista de råvaror som är i säsong just nu har vi använt oss av Säsongsmat API. De erbjuder data bland annat i form av recept och råvaror samt fruktträd. Anledningen till att vi valt detta API är för att de även går att få fram råvarorna baserat på när de är i säsong i Sverige och att de kan erbjuda engelsk översättning. Detta var nödvändigt för att kunna sammanfoga det med Spoonacular och att vår hemsida presenteras på engelska. Requesten som skickas till säsongsmat innehåller aktuell månad just nu samt valt språk engelska. Svaret från säsongsmat fås tillbaka i form av JSON-objekt som i sin tur består av JSON-objekt i flera lager. Dokumentationen för Säsongsmats API är inte helt enkel att förstå. De exempel som presenteras för att få fram råvaror i säsong just nu fungerar inte och sidan uppdaterades senast 2012. Adressen som hänvisas till för kontakt är inte längre i bruk. Då vi inte kunde hitta något annat API som gav oss den data vi ville

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

ha testade vi oss fram och lyckades till sist få fram det vi sökte, men det var svårarbetat och tidskrävande.

Kommunikationen med Säsongsmat sker i klassen FoodInSeason där vi använt oss av biblioteket Unirest för att enkelt kunna hantera HTTP-requests. Aktuell månad tas fram i klassen CurrentMonth, som skickar månad med som parameter i requesten till Säsongsmat. Svaret sparas i en JsonNode utifrån vilken vi gräver oss ner i lagrena av JSON-objekt och hämtar ut råvarorna och placerar dem i en ArrayList. ArrayListen ingredientsInSeason används sedan av klassen RecipeGetter som hanterar kommunikationen med Spoonacular.

Spoonacular och ReceipeGetter

Recepten som presenteras i WineOcular hämtas från API:et Spoonacular. Det är ett väldigt brett och flexibelt API där en kan komma åt recept, matförslag, vinförslag och mer på flera olika sätt, som dock har ett poängsystem för hur många förfrågningar en kan göra per dag utan att betala. Vi använder oss av Spoonacular på två sätt. Först gör vi en request för att få matförslag baserat på vilken vindruva som är vald. Efter det använder vi oss av det resultatet tillsammans med ingredienser som är i säsong, hämtade från Säsongsmat API, och hämtar en lista på recept som både passar till det valda viner och som har ingredienser som är i säsong.

All kommunikation med Spoonacular sker via ett antal metoder i klassen RecipeGetter. Vi har testat ett antal olika sätt att få fram exakt den informationen vi behöver för vår applikation, då även om Spoonacular har många sätt att presentera data så fanns det inget som passade helt perfekt för vårt syfte.

Eftersom listan av ingredienser i säsong är väldigt lång så tänkte vi först använda oss av en endpoint som kunde ta emot obegränsat antal ingredienser, och sedan ranka resultaten efter flest matchande ingredienser. Denna endpoint kostade även väldigt få poäng. Problemet med denna endpoint var att vi inte kunde lägga in matförslaget som vi fick ut tidigare i förfrågan.

Vi bestämde oss istället för att använda oss av en endpoint där vi kan lägga in en fritextsökning som till exempel “pasta” eller “stew” och sedan även lägga in en ingrediens som också måste finnas med i resultatet. Problemet med denna endpoint var att man enbart kunde lägga in en ingrediens, och även bara en fritextsökning. Detta betydde att vi var tvungna att skicka flera förfrågningar till Spoonacular, en för varje matförslag som vi fått av vindruvan och sedan en för varje ingrediens. Det var dock ohållbart när det kom till poängen vi får använda oss av. Vi kompromissade och skrev istället kod som använde sig av ett fåtal av de ingredienser som är i säsong. Det låter oss behålla kärnan av vad vår tjänst skulle göra och också hålla nere poängen som vi använder i Spoonaculars tjänst.

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

Det enda sättet som datan från Spoonacular behandlas är genom att plocka ut specifik info från JSON-filerna som vi får tillbaka, som är relevant för vår tjänst, och även ta bort HTML-taggar från textsträngar som vi inte vill ha med när vi presenterar informationen.

2.2.2 Java

För att kunna lagra den data som WineOcular använder i sina tjänster används en Java-klass som vi valt att kalla Recipe, där har vi valt ut de variabler som behövs för att leverera data till vårt API. Genom att skapa samlingar av Recipe-objekt utifrån svaren från våra externa API kan resultaten presenteras för användaren på ett enkelt och smidigt sätt. Vår applikation följer en grundläggande MVC(Model-View-Controller)-arkitektur. Recipe är modellen, som genom en Controller-klass fylls med information från Recipe Getter och FoodInSeason för att sedan leverera dessa till vårt API.

För att underlätta WineOculars användning av och syfte med sitt API, innehåller Recipeklassen enbart den information som tjänsten har nytta av. Ett Recipe-objekt innehåller ett id, en titel, en url till bild, en lista ingredienser och en beskrivning. För att hantera en samling av recept används ArrayList<Recipe>. Vi använder oss sedan av klassen Gson för att kunna konvertera Java-objekten till JSON-objekt.

För att hantera druvorna som användaren kan välja mellan, har vi skapat ett Enum bestående av de druvor vi på WineOcular valt ut. Varje konstant i enumet har samma format som Spoonacular tar emot för att hämta vin och mat-matchningar.

2.2.3 WineOcular API

WineOcular serverar informationen som lagras i Java-objekten genom WineOcular API, det har vi utvecklat med hjälp av ramverket Spark [3]. Med hjälp av Spark finns vårt API tillgängligt på en server, och består främst av det Spark kallar för route. En route består av ett verb (metod), en sökväg (path) och ett callback (request och response). Vårt API består enbart av get-metoder, och genom att välja sökväg kan FrontEnd skicka ett request för att få tillbaka ett JSON-objekt med informationen från Java-objektet som response. Detta sker i vår Java-klass APIRunner, som importerar Spark.

För att initiera sökvägen till WineOculars FrontEnd så använder WineOculars BackEnd mallmotorn Pebble [4] tillsammans med Spark i APIRunner.

Pebble används även för att skapa mallar i FrontEnd som base (layout) med nav (navigation) och footer (sidfot). Mallar har även skapats för att kunna återanvända innehåll som seasonal banner

(bild med text), redgrape och whitegrape (presentation och sök med druvor), resultgrid (rutnät med sökresultat) och recipepresenter (presentation av valt recept).

3. Användarmanual

WineOcular finns på GitHub på addressen https://github.com/corneliaskold/wineocular.git, där vem som helst kan ladda ner applikationen. WineOcular har utvecklats i IntelliJ IDEA, och är därför enklast att öppna som ett projekt i IntelliJ IDEA. Detta kan en göra genom att antingen ladda ner koden från GitHub, eller genom att klona projektet direkt från GitHub. IntelliJ IDEA hittar du här: https://www.jetbrains.com/idea/.

För att kunna köra applikationen krävs även maven, instruktioner för hur det går till finns på addressen http://maven.apache.org/. Genom att sedan använda sig av projektets POM-fil kan du hämta ner alla beroenden.

När detta är gjort kan bygga och köra applikationen från IntelliJ. När applikationen är igång så kan du öppna din webbläsare och nå WineOcular med url http://localhost:2020/.

4. API-dokumentation

WineOcular API Dokumentation finns tillgänglig på den här addressen https://app.swaggerhub.com/apis-docs/WineOcular/WineOcular/2.0.0. Som nämnt ovan används WineOcular API för att servera informationen från BackEnd till FrontEnd. Vi har enbart använt oss av GET-metoder, då WineOculars tjänst går ut på att presentera ett recept för användaren.

Den första ändpunkten GET / genererar egentligen enbart index.html, och möjliggör det för WineOcular att presentera sin hemsida på http://localhost:2020/. Denna ändpunkt är inte tänkt som en del av WineOcular API, och därför finns den inte heller med i API-dokumentationen.

GET /search/{grape} används för att hämta en lista med recept som dels matchar vald druva, det vill säga parametern {grape}, och dels matchar ingredienser i säsong. Returvärdet är därför en array av JSON-objekt som representerar varje recept i listan med id, titel, och bild.

/GET /recipe/{id} används för att hämta ett recept med parametern {id}, returvärdet i detta fallet är ett JSON-objekt innehållande id, titel, beskrivning, bild, ingredienser samt instruktioner. Ingredienser består av en array med varje ingrediens.

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld

Referenser

[1] H. W. Turnbull, The correspondence of Isaac Newton / Vol. 1, 1661-1675, Cambridge University Press for the Royal Society, 1959.

[2] w3schools.com, ”What is JSON?,” [Online]. Available: https://www.w3schools.com/whatis/whatis_json.asp. [Använd 11 01 2020].

[3] Spark, ”Spark,” [Online]. Available: http://sparkjava.com/documentation. [Använd 11 01 2020].

[4] Pebble Templates, ”Pebbletemplates.io,” [Online]. Available: https://pebbletemplates.io/. [Använd 11 01 2020].

Theresa Dannberg
Josefiné Klintberg
Andreas Månsson
Cornelia Sköld