Utveckling av en e-kokbok med MEAN Stack

Ilkka Nyholm

Examensarbete / Degree Thesis

Informationsteknik / Information Technology

2015

Ilkka Nyholm

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	5221
Författare:	Ilkka Nyholm
Arbetets namn:	Utveckling av en e-kokbok med MEAN Stack
Handledare (Arcada):	Göran Pulkkis
Uppdragsgivare:	Arcada

Sammandrag:

Examensarbetet beskriver utvecklingen av en elektronisk kokbok, vars användargrupp är personer med kognitiva nedsättningar samt synskadade. Arbetet var en beställning av ett annat examensarbete från Arcadas Ergoterapi-linje. I arbetet genomgås olika processer från användbarheten till utvecklingen av applikationen och till slut testningen av applikationen. För att få en klar bild av hur e-kokböcker fungerar undersöktes redan existerade e-kokböcker. I enlighet med denna undersökning och beställarens krav skapades en grundläggande design för applikationen. Kokboken utvecklades som en webbapplikation. I den teoretiska delen av arbetet beskrivs teknologier som användes för att skapa e-kokboken. I den praktiska delen av arbetet gjordes en A/B-test, där två olika versioner av e-kokboken testades med två testpersoner. Resultaten av testningen analyserades för att bedömma e-kokbokens användbarhet och brister.

Nyckelord:	Webbapplikation, MEAN Stack, HTML, A/B-testning, användbarhet
Sidantal:	32
Språk:	Svenska
Datum för godkännande:	10.12.2015

DEGREE THESIS	
Arcada	
Degree Programme:	Information Technology
Identification number:	5221
Author:	Ilkka Nyholm
Title:	Development of a e-cookbook with MEAN Stack
Supervisor (Arcada):	Göran Pulkkis
Commissioned by:	Arcada

Abstract:

This thesis work describes the development of an electronic cookbook for people with cognitive and visual impairments. This work was a request from another thesis work in Arcada's Ergotherapy study programme. In this work the processes from usability to development and final testing are described. To get a clear picture how an e-cookbook is supposed to work, already existing e-cookbooks were investigated. Based of these investigations and with the help from the requesting party a basic design of the e-cookbook was created. The e-cookbook was implemented as a web application. In the theoretical part of the thesis work the technologies of the MEAN-stack on which this application is based on are described. The practical part of the work was an A/B test of the e-cookbook where two different versions were tested by two persons. The test results were analyzed to evaluate the usability of the e-cookbook and to detect deficiencies.

Keywords:	Web application, MEAN stack, HTML, A/B test, Usability
Number of pages:	32
Language:	Swedish
Date of acceptance:	10.12.2015

INNEHÅLL / CONTENTS

FIGURER

FÖRORD

Jag vill tacka min handledare Göran Pulkkis för hjälp med skrivprocessen. Ännu vill jag tacka Anna-Maria Malm och Anna Evars för innehållet till kokboken. Jag vill också tacka Förbundet Finlands Svenska Synskadade för hjälpen i testningen samt Annikki Arola som möjliggjorde testningen.

TERMINOLOGI OCH FÖRKORTNINGAR

A/B-testning = En test där två versioner är jämförda. Versionerna är till största delen lika, förutom någon skiljande sak som skall testas.

AngularJS = Ett frontend-ramverk utvecklat av Google. Har mycket unika direktiv.

BSON = (Binary JSON) Ett datautbytesformat som främst används vid datalagring och som ett nätverksöverföringsformat i en MongoDB-databas.

DOM = (Dokument Object Model) Dokumentobjektmodellen, ett gränsnitt som skapar interaktiva sidor med JavaScript

JSON = (JavaScript Object Notation) Ett JavaScript-baserat dataöverföringsformat.

SPA = (Single Page Application) En arkitektur där en enda HTML-sida används för att visa innehållet. Man kan göra det genom att dynamiskt ladda in olika modeller.

MVC = (Model View Control) En arkitektur där applikationen är indelad i modell, vy och kontroll.

1 INLEDNING

Min uppgift är att skapa en elektronisk kokbok av ett annat slutarbete från Arcadas Ergoterapi-utbildning. Kokbokens innehåll kommer från beställarna. Beställarna har ställt vissa krav som skall uppfyllas.

Jag fick fria händer angående formatet, och jag bestämde mig för att skapa en webbapplikation. I mitt projekt använder jag den så kallade ”MEAN Stack”-helheten, så att man kan skriva hela projektet i JavaScript utan att behöva syssla med PHP. Jag valde att använda den JavaScript-baserade MEAN Stack för att det är en aktuell teknik. Allt flere webbprojekt görs idag med JavaScript-baserad MEAN Stack.

1.1 Syfte och mål

Syfte med arbetet är att skapa en webbapplikation med en layout som är användarvänlig för användargruppen genom att använda MEAN Stack’s olika tekniker. Användargruppen är personer med kognitiva nedsättningar samt personer med synnedsättningar. För att säkra att slutprodukten blir så lämplig som möjligt för användargruppen utförs A/B-testning på två olika layouts och sedan analyseras testresultaten.

1.2 Avgränsning

Jag kommer inte att skriva om hur webbapplikationen installeras.

2 METODER

I detta kapitel beskrivs de olika undersökningsmetoderna i planeringen av webbapplikationens utseende samt hur webbapplikationen programmerades.

2.1 Existerande e-kokböcker

För att kunna skapa en e-kokbok måste man först undersöka att hurdana e-kokböcker redan existerar. Då får man en grunduppfattning om hur de är uppbyggda och hur de fungerar.

E-kokböcker är vanliga kokböcker men istället för att finnas i bokformat finns de i elektronisk format som läses med datorer eller andra digitala medier. Sedan finns det också vanliga webbapplikationer som innehåller recept. I denna kategori faller också detta examensarbete.

Eftersom smarttelefonerna och tabletterna är idag allt populärare finns det också flera e-kokboksappar, med vilka man lätt kan kolla recepten samtidigt man lagar mat. Speciellt Apples iPad är populär bland synskadade för tillgång till olika medier p.g.a. av att man kan med sin röst navigera igenom de olika menyerna. Dessutom kan man ha en datorröst som utläser innehållet. (Clymer 2012)

Det finns en hel del olika e-kokböcker avsedda för olika ändamål. T.ex. finns det flera, oftast engelska, e-kokböcker avsedda för personer med synskador. En populär finsk e-kokbok är (Kotikokki, 2015), där användare kan själv uppladda recept och kommentera andras recept. En annan finsk kokbok är (Valio, 2015), där man kan välja flera olika maträtter enligt ursprungsland och säsong, men också som huvudrätter. Valio har optimerat sin sida så att den kan lätt genomsökas och det finns flera kategorier för olika maträtter. På första hand kan sidan kännas lite kaotisk när det finns så mycket information att välja emellan. Båda dessa kokböcker följer en liknande stil, man har en bild av maträtten och om man musklickar på bilden öppnas själva receptet, där ingredienserna och tillverkningen beskrivs. Valios kokbok räknar dessutom kalorierna för varje maträtt.

Kotikokki i sin tur visar också olika kategorier för maträtter. Man kan också filtrera resultaten till att visa endast de recept som är populära bland användare.

Undersökningen gav inte resultat på existerande svenska eller finska e-kokböcker avsedda speciellt för personer med kognitiva nedsättningar eller synskador.

2.2 Programmering

För programmeringsdelen användes främst en litteraturstudie. Litteraturstudien innebär läsning av böcker som behandlade MEAN-stacken för att få en bild av hur det hela fungerade.

3 ANVÄNDBARHET

Användbarhet är en viktig sak att tänka på då man planerar att skapa en webbsida. Hur skall man på bästa möjliga sätt föra fram användargränssnittet så att webbsidan är lättanvänd och designen lämpar sig för sidans tema. Användbarheten byggs upp av flera olika element såsom sidans design, hur lättförståelig sidans innehåll är samt hur ”hand till ögat”-rörelsen flödar.

3.1 Krav

Kraven som ställdes på e-kokboksprojektet var till största delen visuella. Fonten skulle vara samma i alla olika delar av produkten (Rubrik: Arial, 20 punkt; ingredienser: Arial, 18 punkt; instruktionerna Arial 18. Radavstånd 1,5). Kommunikationen skulle utgå från bilder (visuell stimuli) istället för från text. Bilderna skulle ha starka kontraster som underlättar urskiljande. På webbsidan skulle det också finnas en sektion för redskapen i Arcadas Smart-kök där de visas som bilder och är lätt igenkännbara. Sidan skulle också ha en ”timer” för att ta tid då man tillreder maten. De olika tillredningsstegen skulle visas som en sekvens.

3.2 Användargränssnitt och design

Användargränssnittet är den viktigaste delen i ett datasystem. Användargränssnittet är den delen som användaren kan se, höra, röra vid och interagera med. All kod är gömd från användaren. Målet med att skapa ett smidigt användargränssnitt är att göra användningen av produkten lätt, produktiv och trevlig.

Användargränssnittet har i huvudsak två komponenter: inmatning (eng. input) och utmatning (eng. output). Med inmatning kommunicerar användaren sina önskningar till datorn. Inmatning sker oftast med mus eller tangentbord. Utmatning är hur datorn visar resultatet av sina beräkningar till användaren. Vid dagens läge är en datorskärm det mest använda sättet för att visa resultat. Mekanismer som utnyttjar användarens auditiva kapacitet, röst och ljud är det näst-mest använda sättet. (Galitz 2007 s.4)

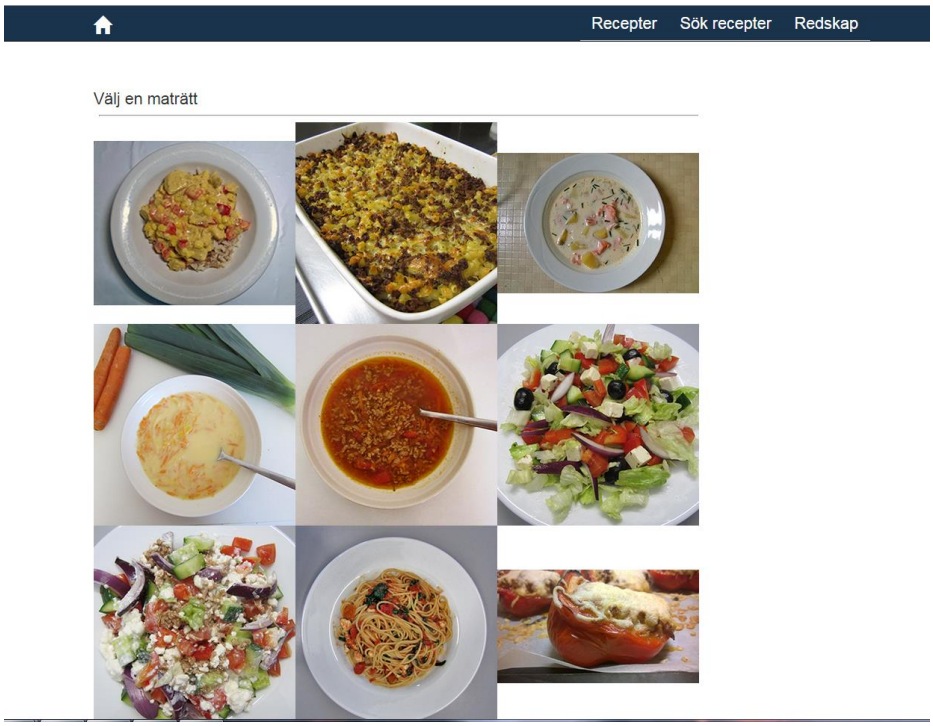
Ett välplanerat användargränssnitt använder sig av en blandning av olika in- och utmatningsmekanismer som uppfyller användarens behov och möjliga begränsningar på ett effektivt sätt. (Galitz 2007. s.4)

3.2.1 Layout 1

E-kokbokens layout är enkel och det finns inga distraktioner eller granna färger som stör användaren. Webbsidans färger är vit och mörkblå.

Figur 1. *E-kokbokens första webbsida.*

I Figur 1 visas e-kokbokens tema och navigeringsfält.



Figur 2. *Receptfliken på webbsidan*

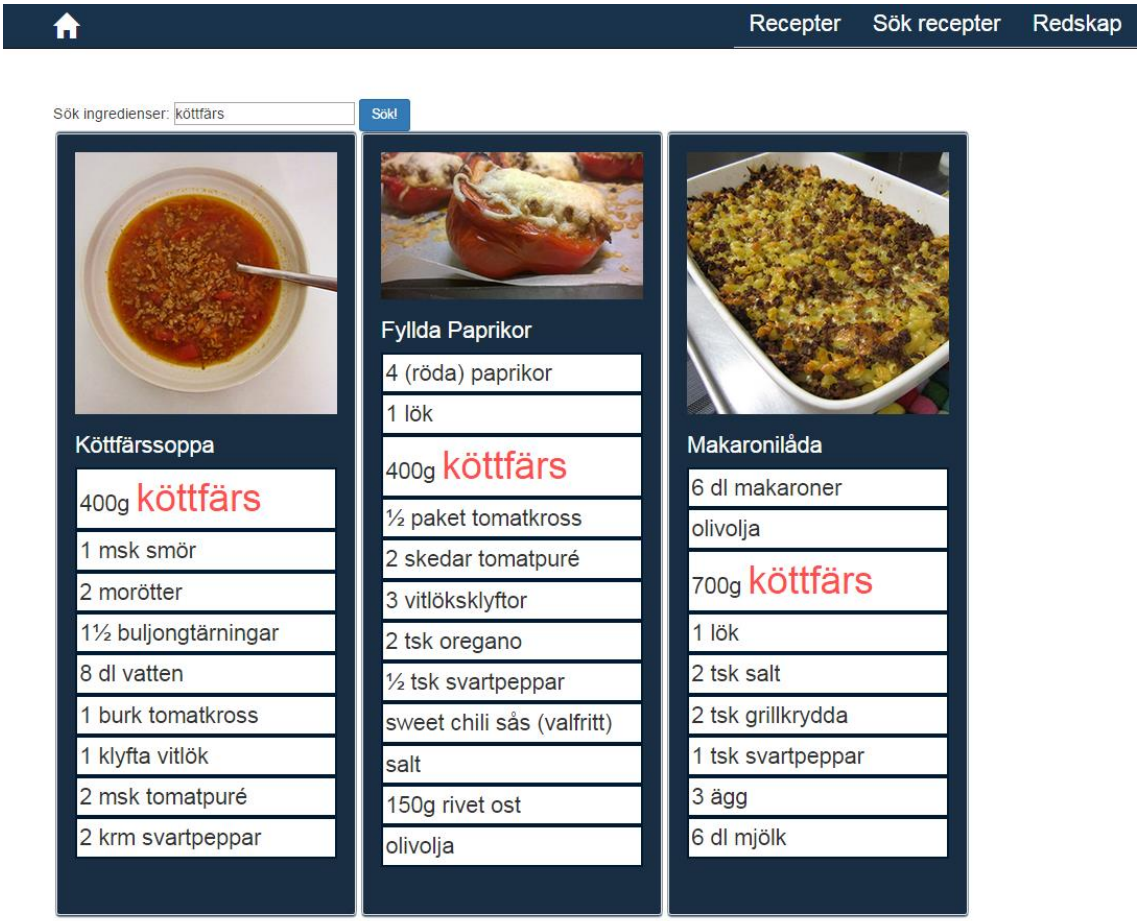
I Figur 2 visas de olika maträtterna man kan tillreda. Då användaren musklickar på en bild görs en förfrågning till databasen som söker efter recept med namnet på bilden som musklickades och hämtar resultatet till användaren på en ny webbsida (Figur 3).

Figur 3. *Webbsidan med från databasen hämtat recept.*

I Figur 3 visas databassökningens resultat. Data som hämtas från databasen är maträttens bild, receptets namn, ingredienser och tillredningsskeden. Under ”Tillredning här” visas tillredningskeden ett åt gången med pictogram som stöder användare med synskador (Figur 4).

Figur 4. *Tillredningsskedet där varje delmoment visas ett åt gången med pictogram.*

I Figur 4 visas också två pilknappar för framåt och tillbaka med vilka användaren styr tillredningen genom att musklicka framåt efter ett avklarat tillverkningsskede. Den röda pilknappen går tillbaka till föregående tillverkningsskede ifall man missat något.



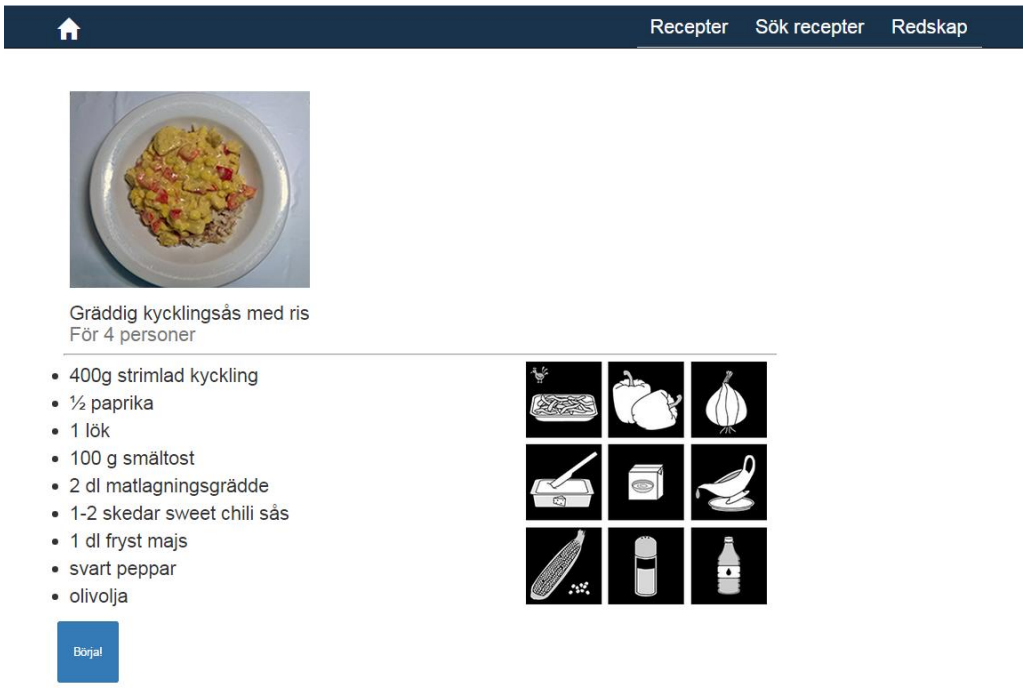
Figur 5. *Webbsidan för receptsökning.*

I Figur 5 visas att användaren kan söka recept som innehåller en viss ingrediens. Sökresultatet visas markerad med röd text och större font-storlek. Man kan visa upp till sex resultat.

3.2.2 Layout 2

Den andra layout som skall användas i A/B-testning skiljer inte sig mycket från den första layouten. Det är främst recept-webbsidan som har annan layout, p.g.a. att det är denna funktionalitet som jag vill testa och se om matlagningsprocessen är logisk att följa för användaren.

I Figur 6 kan man se receptfliken i den andra layouten. Layouten skiljer sig med att själva tillverkningsskedet inte visas samtidigt som ingredienserna. Ingredienserna visas också nu i pictogram bredvid texten, medan de i den första layouten kom som första bild i tilverkningsskedet. Med den blåa Börja-knappen går man till själva tillverkningsskedet (Figur 7).



Figur 6. *Receptfliken med en annan layout.*

Figur 7. *Tillverkningsskedet ensamstående.*

I Figur 7 visas tillverkningsskedet. Den blåa Ingredienser-knappen för användaren tillbaka till receptfliken (Figur 6). Funktionaliteten är samma som i Figur 4, två pilknappar för att gå framåt eller bakåt. I en lista visas med färger vilka skeden man gjort och vilka som är ogjorda. Gröna är gjorda, gul är aktiv och grå är ogjord. I layout 2 har det dedikerats mera utrymme för tillverkningsskedet.

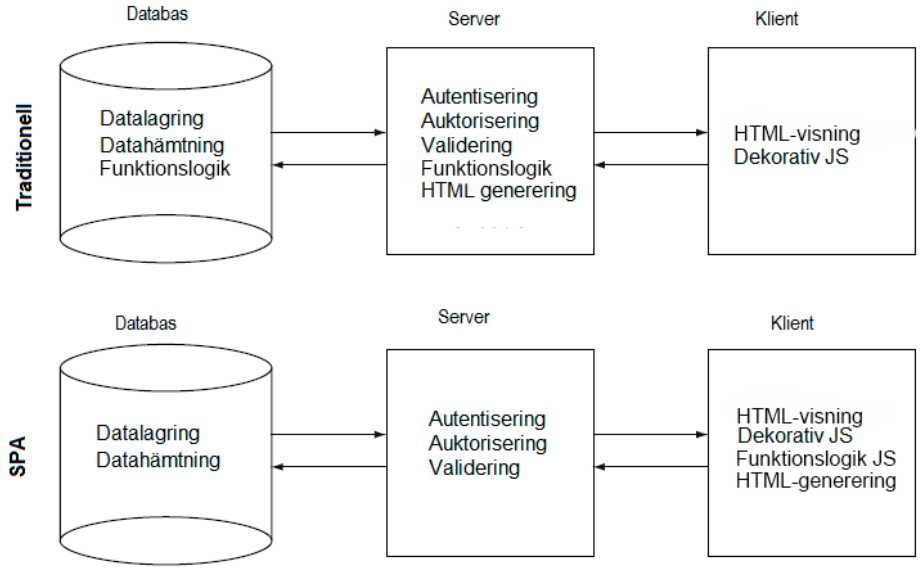
4 TEKNISK UPPBYGGNAD

Detta kapitel beskriver de olika teknikerna som använts i e-kokboksprojektet.

4.1 Single Page Application (SPA)

Single Page Application (SPA) beskriver en applikation där en HTML-fil skapar den huvudsakliga webbsidan. Då webbsidans innehåll ändras inladdas antingen allt innehåll (HTML,CSS, JavaScript) på nytt eller dynamiskt enligt behov. Inladdningen sker oftast med Ajax. Målet är att skapa en flytande användarupplevelse genom att upprepa

skrivbordsprogrammens särdrag i webbläsaren. (Mikowski & Powell 2013 s.4-5.)



Figur 8. *Jämförelse av processeringsuppgifter i databas, server och klient. (Mikowski & Powell 2013, s.8)*

I Figur 8 kan man se att servern i en SPA-arkitektur belastas mindre än i en traditionell arkitektur. Funktionslogiken och HTML-genereringen har överförts till klienten, vilket minskar på mängden förfrågningar som skickas från servern. Detta möjliggör också användningen av vilken som helst serverteknologi, på grund av att så mycket av funktionaliteten flyttas över till klienten. (Mikowski & Powell 2013, s.8.)

Det har tagit en lång tid för JavaScript SPA att bli allmänt använd. Före SPA var flash och Java Applets populära och JavaScript användes för någon funktionalitet på webbsidor. JavaScripts användning ha stigit med tiden genom att största delen av svagheterna har fixats.

4.2 MEAN Stack

MEAN Stack innebär användning av flera olika ramverk och tekniker för att skapa en fungerande utvecklingsmiljö. I en MEAN Stack fungerar Node.js som den fundamentala

utvecklingsplattformen. Node.js sköter om backenden samt olika skript på servern. MongoDB förser databasen för utvecklingsmiljön, men är tillgänglig endast genom en Node.js-modul för MongoDB. Själva webbservern definieras av Express.js, som också är en Node.js-modul. Vyn i webbläsaren sköts av det klientsidiga ramverket AngularJS. AngularJS är ett MVC-ramverk (Model-View-Control) där modellen är uppbyggd av JSON, vyn är uppbyggd av HTML/CSS och kontrollen är uppbyggd av JavaScript. (Dayley. 2014.)

4.2.1 **AngularJS**

AngularJS är ett MVC-ramverk utvecklat av Google för klientwebbsidor. AngularJS är programmerat i JavaScript med ett förminskat Jquery-bibliotek. Den grundläggande idén med AngularJS är att erbjuda ett ramverk som gör det lätt att implementera välstrukturerade och välplanerade webbsidor och webbapplikationer. AngularJS har funktionalitet för användarens inmatning, för klientsidans datamanipulering och för kontroll av hur olika element visas i webbläsaren. (Lerner 2013 s.8)

En fördel med AngularJS är ”Data binding”, med vars hjälp man kan koppla data från en kontroll till ett HTML-element genom en s.k ”scope”. En scope fungerar som lim mellan vy och kontroll. Innan webbapplikationen visar vyn till användaren länkar vyn applikationsmodellen till scopen och där införs värden som skickats från kontrollen till motsvarande variabel i HTML-vyn. AngularJS tvingar användare att skapa välskriven och logisk programkod som är återanvändbar. Det finns mycket stöd för AngularJS. Utvecklaren Google investerar mycket i projektet, vilket förutspår en bra framtid för AngularJS. (Dayley 2014 s. 399-400)

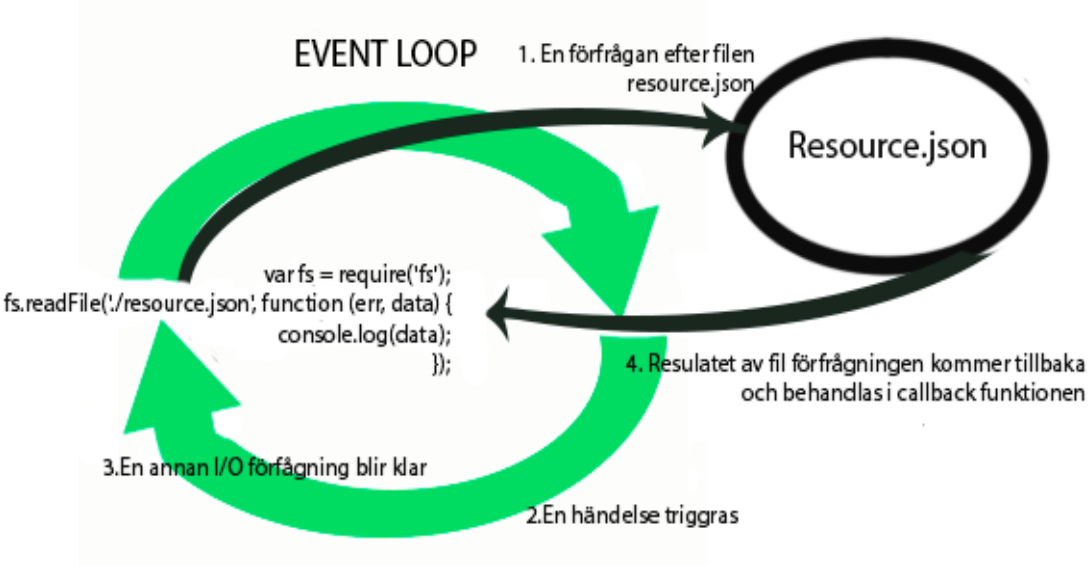
4.2.2 Node.js

Node.js är en utvecklingsramverk baserat på Google Chrome's V8 JavaScript-motor. Node.js introducerades 2009 av Ryan Dahl vid JSCONF i Berlin. Idén till Node.js fick Dahl från förloppsindikatorn för filuppladdning på Flickr (en webbsida med ett bild- och videoupptagningsarkiv). Webbläsaren visste inte hur mycket data hade överförts och

därför måste den hela tiden skicka förfrågningar till webbservern. Dahl tänkte att det måste finnas något lättare sätt och skapade därför ramverket Node.js. (Daley 2014 s.39)

Node.js-kod skrivs i JavaScript, och V8-motorn kompilerar JavaScript-koden till exekverbar maskinkod. All serverkod kan skrivas med Node,js, vilket omfattar själva webbservern, serverskripten och annan stödande funktionalitet.

Node.js använder sig av händelsestyrd skalbarhet (event-driven scalability) genom att applicera unik logik på webbförfrågningar. Detta innebär att istället för att ha flera trådar (eng. threads) som väntar på att behandla webbförfrågningar, behandlar Node.js förfrågningarna i en och samma tråd. Om det kommer nya förfrågningar skapar Node.js inte en ny tråd utan inför i samma tråd en ”callback” funktion till en ny förfrågning, och hoppar till nästa nya förfrågning för att upprepa processen. När ”callback”- funktionen har exekverats returneras svaret till webbläsaren. Node.js-webbservrar har därför en skalbarhet som traditionella webbservrar inte kan uppnå.



Figur 9.*Exempel på ”non-blocking I/O” i webbläsaren (Cantelon, 2014, s.11)*

I Figur 9 kan man se hur webbläsaren har gjort en förfrågning efter filen resource.json (1). När förfrågningens svar kommer tillbaka exekveras ”callback”-funktionen (4). Själva filhämtningen sker utanför förfrågningsslingan så att man kan exekvera nya operationer

före filhämtningen är klar (2). Operationerna exekveras i en sådan ordningsföljd att den nyaste operationen slutförs först (3). Efter alla andra förfrågningar är klara returneras den första (4).

Node.js har flera mekanismer för att ta hand om inmatning och utmatning av data från system till system. Den mest använda datatypen i Node.js är JSON (JavaScript Object Notation). JSON är en lätt metod att konvertera JavaScript-objekt till text och tvärtom. JSON fungerar bra då man serialiserar dataobjekt från klient till server.

Med Node.js kommer den mycket använda Node Package Manager (npm) med vilken man kan installera nya moduler från kommandotolken till sitt Node.js-projekt. Modulerna har blivit vardag i webbutvecklingen och det skapas hela tiden nya moduler som utvidgar Node.js-funktionaliteten. De flesta modulerna är utvecklade av en tredje part.

Node.js är en ny teknologi med en aktiv utvecklingsgemenskap och stor användarpopulation. Node.js har redan fått en viktig roll i MEAN Stack JavaScript-utvecklingen. Utan Node.js skulle det vara omöjligt att skapa end-to-end JavaScript applikationer. (Dayley 2014. s.12)

4.2.3 Express.js

Express.js är en Node.js-modul som kopplar HTTP-modulen i Node.js till ett lättanvänt gränssnitt. Express.js utökar också HTTP-modulens funktionalitet med att göra det lätt att behandla omdirigering vid webbserverbläddring, webbserversvar och HTTP-förfrågningar. (Dayley 2014. s.357)

4.2.4 MongoDB

MongoDB är en agil och skalbar NoSQL-databas. MongoDB passar bra in i ”high traffic”-webbsidor där man sparar användarkommentarer, bloggar och annan data.

MongoDB är en NoSQL-databas som sparar datamodeller som separata dokument i en kollektion istället för tabeller. NoSQL står för ”Not only SQL” och är uppbyggd av teknologier för lagring och hämtning av data från traditionella SQL-databaser utan begränsningar. En kollektion är helt enkelt en gruppering av dokument som har samma eller liknande ändamål. En kollektion påminner mycket om en tabell i en traditionell SQL-databas. Skillnaden är att en kollektion inte är bunden till ett strikt schema. Dokumenten i en kollektion kan ha olika struktur.

Ett dokument är en representation av ett data objekt i en MongoDB-databas. En kollektion byggs upp av ett eller flera dokument. Dokumenten i MongoDB skiljer sig mycket från rader i en traditionell SQL-databas. Det finns en kolumn för varje datavärde i en rad. I MongoDB kan dokumenten innehålla inbäddade subdokument, vilket ger möjlighet till att skapa bättre datamodeller för applikationer.
MongoDB-databasens dataobjekt, som representerar dokument, sparas i BSON, som är en binär form av JSON. Dokumentens värden är i JavaScript-format vilket leder till att man inte behöver konvertera MongoDB-dataobjekt till JavaScript då man anropar dem i en Node.js applikation. Största storleken för en kollektion är 16MB. Detta förhindrar sökfrågor som förorsakar en överstor använding av RAM-minnet. (Dayley 2014 s.195-197.)

4.3 Övriga tekniker

I denna kapitel beskrivs de viktiga och grundläggande teknikerna för webbdesign.

4.3.1 HTML5

HTML är en gammal teknik, som introducerades år 1993. På 90-talet fanns det en massa aktivitet kring HTML. Versionerna 2.0, 3.2 och 4.0 utkom alla under samma år. Under utvecklingen ansvarade World Wide Web Consortium (W3C) för specifikationerna. År 2004 skapade gruppen Web Hypertext Application Working Group (WHATWG) en specification för HTML5, med nya egenskaper avsedda för webbapplikationer, som var en HTML-svaghet. W3C tog över igen år 2006 och utgav sin första fungerande version

av HTML5 år 2008. Den femte och sista versionen av HTML5 utgavs i oktober 2014. (Lubbers mm 2011. s.4)

4.3.2 JavaScript

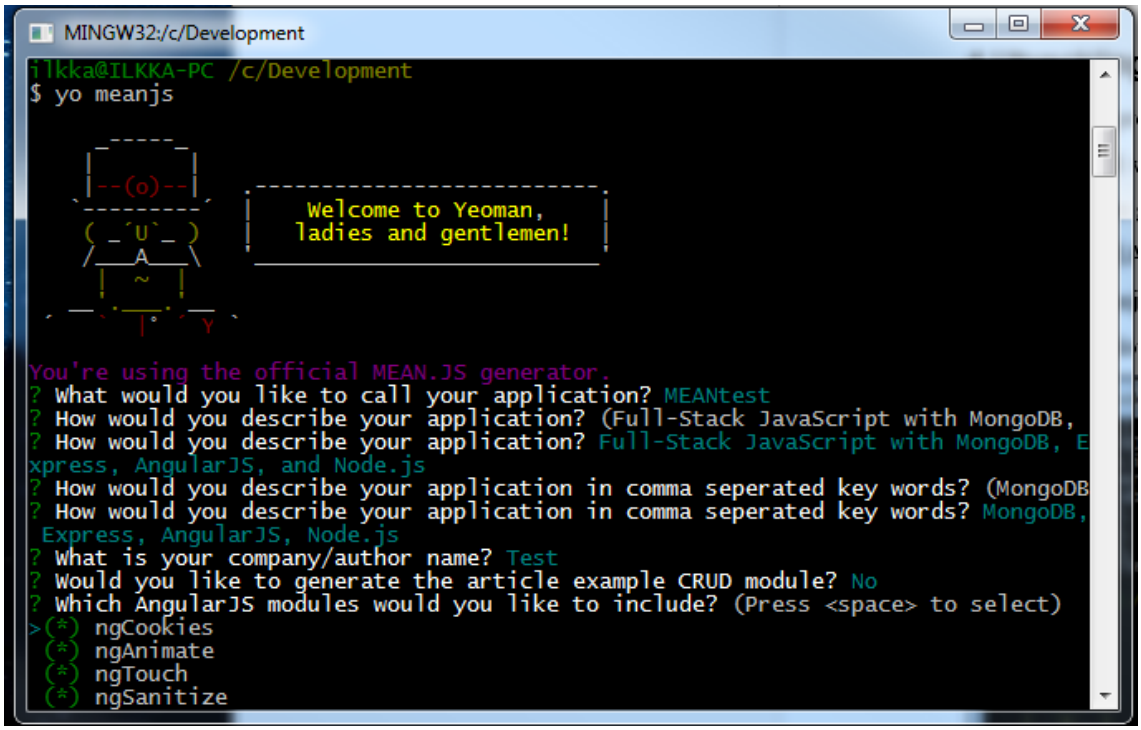
JavaScript är en programmerings språk med kapacitet för objektorientering. JavaScript-språket påminner om C, C++ och Java med programmeringsdirektiv för if-sats, while-slinga och olika operatorer. JavaScript skiljer sig dock från C, C++ och Java med att man inte behöver specificera vilken datatyps variabel man arbetar med, det räcker med att definiera en variabel. JavaScript används oftast i webbläsare. Med nya plattformer såsom Node.js kan man använda JavaScript även på webbserversidan. JavaScripts funktionalitet innebär oftast manipulering av dokumentobjektsmodellen (DOM) och skapandet av skript som interagerar med användaren. (Flanagan 2011. s.1)

4.3.3 Bootstrap 3

Bootstrap är ett frontend CSS-ramverk som erbjuder ett användarvänligt, beprövat sätt att skapa moderna användargränssnitt.

5 UTVECKLING AV E-KOKBOKSAPPEN

För att generera e-kokboksprojektet användes Yeomans MEAN.js-generator. Yeoman är en generator med vilken man kan generera projekt, som använder olika komponenter. MEAN.js är en generator i Yeoman som installerar de komponenter som behövs i ett MEAN.js-projekt. MEAN.js installeras med kommandot ”npm install –g generator-meanjs” i kommandotolken. Efter att generatorn är installerad skapar man med kommandot ”yo meanjs” i kommandotolken ett MEAN.js-projekt.

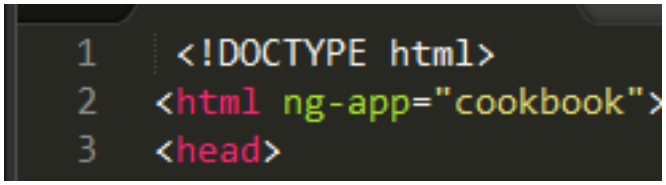


Figur 10. *Generering av ett MEAN.js-projekt med Yeoman MEAN.js-generator.*

Då Yeoman genererar ett projekt kan man med Yeoman's kommandotolkapplikation lätt namnge sitt projekt och vad man vill att skall inkluderas i projektet.

5.1 Frontend

Frontend-ramverket för e-kokboksprojektet är AngularJS. I AngularJS finns flera unika direktiv som man kan manipulera vyn med. För att skapa ett AngularJS-projekt måste man i HTML-filens HTML-tagg ta med direktivet ”ng-app”. Ett direktiv är ett sätt att inlägga en funktion till ett specifikt DOM-element. Placering av ”ng-app” på ett DOM-element anger att scope-kedjan börjar från detta element, varifrån alla andra direktiv ärver data. (Lerner 2014.)



Figur 11. *Direktivet ”ng-app” med namnet ”cookbook” i ett HTML-element.*

Efter att man specifierat ett ”ng-app”-direktiv skall man också skapa en JavaScript-fil som har en referens till direktivet man skapat.

```
angular
    .module('cookbook', [
    'ui.router',
    'ngAnimate',
    'ui.utils',
    'ngSanitize'
    ])
    .config(['$urlRouterProvider','$stateProvider',function($urlRouterProvider, $stateProvider){
        $urlRouterProvider.otherwise('/');

        $stateProvider
        .state('index', {
            url:'/home',
            templateUrl:'index.html'
        })
        .state('recipes', {
            url:'/recipes',
            templateUrl:'templates/recipe-partial.html',
            controller:'requestController'
        })
        .state('recipename', {
            url:'/recipes/view',
            templateUrl:'templates/recipe-view.html',
            controller:'recipeController'
        })
        .state('search', {
        url:'/search',
        templateUrl:'templates/search-view.html',
        controller:'searchController'
        })
        .state('tools', {
        url:'/tools',
        templateUrl:'templates/kitchentools.html',
        controller:'toolsController'

        })
    }])
```

Figur 12. *Filen app.js i e-kokboksprojektet. AngularJS-applikationen definieras här. Navigeringsdirektiven defineras i funktionen $stateProvider.*

I Figur 12 kan man se att ”ng-app”-direktivet refereras direkt på andra raden efter AngularJS-definitionen. I modulvariabeln definierar man webbapplikationen (ng-app) som skapas av HTML-filen. Modulen i AngularJS är det huvudsakliga stället där all applikationskod finns. En applikation kan innehålla flera olika moduler, där varje modul sköter om en specifik funktionalitet. Då en modul skapas behövs två parametrar. Första parametern är modulens namn. Andra parametern är en lista på applikationens beroenden (eng. dependencies). I beroenden kan man ha olika moduler såsom ui.router eller man kan injicera en kontroll vars funktionalitet behövs i en annan kontroll. Injicering gör det lätt att skapa enhetstester.

$stateProvider är en funktion i modulen ui.router. Den sköter om navigeringen inom sidan genom att kolla efter vilka tillstånd är aktiva. Ett tillstånd representerar ett ställe inom applikationen. Dess uppgift är att beskriva hur användargränsnittet skall se ut vid detta ställe, vilket görs genom samarbete av kontroll, modell och vy som är specifierade för olika tillstånd. (UI-router 2015)

```
<li>
    <a ui-sref="recipes">Recepter</a>
</li>
```

Figur 13. *En ”list item” med ui-sref-direktivet som kopplar till ett tillstånd.*

Följande variabler definieras för en tillstånd i funktionen $stateProvider:

```
.state('recipes', {
        url:'/recipes',
        templateUrl:'templates/recipe-partial.html',
        controller:'requestController'
        })
```

$stateProvider aktiveras då användaren musklickar på ett element med ett ”ui-sref”-direktiv. I Figur 13 finns ett ”ui-sref”-direktiv med värdet ”recipes”. Då användaren musklickar elementet med ”ui-sref”-direktivet aktiveras funktionen $stateProvider. Den hämtar den motsvarande modellen (specifierad i variabeln ”templateUrl” för varje tillstånd i funktionen $stateProvider) och injicerar den i index.html-filens ”ui-view”-direktiv för att byta webbsidans innehåll. Variabeln ”Controller” inställer en kontroll (JavaScript) för denna HTML-modell som laddas. I kontrollen finns funktionslogik. URL-variabeln ställer in webbsidans adress vid det aktiva tillståndet.

```
<div>

    <!-- THIS IS WEHERE ALL THE TEMPLATES WILL BE INJECTED TO THE PAGE WHEN CALLED UPON-->
        <div ui-view>
        <div class="container">
        <img class="img-responsive" src="img/ff.png"></img>
        </div>
        </div>

</div>
```

Figur 14. *Klassen Ui-view i filen index.html som alla modeller inladdas till då deras tillstånd triggas.*

Efter dessa deklarationer är sidans navigering och beroenden inställda i själva app.js-filen. Det som är kvar är att skapa HTML-filerna som motsvarar de tillstånd som skapats. På grund av att sidorna inladdas i index.html behöver man inte deklarera någon dokumenttyp såsom man gör i ensamstående HTML-filer, utan man kan direkt börja deklarera det man vill ha på sidan.

AngularJS direktivet ”ng-repeat” var till stort hjälp i e-kokboksprojektet. Ng-repeat möjliggör iteration av en kollektion av data från kontroll-filen. T.ex om man har gjort en förfråganing efter ett recept från databasen hämtas svaret till kontroll-filen och därifrån kollar direktivet ng-repeat hur många ”list items” den måste skapa. Detta innebär att man inte behöver skapa en statisk lista med t.ex 12 punkter.

5.2 Backend

Webbserver för e-kokboksprojektet är en Node.js-server. Node.js tillåter skrivning av backend-kod med JavaScript. Node.js inladdas från (Node.js 2015). I Figur 15 visas vilka moduler måste inladdas till Node.js-servern i e-kokboksprojektet för att skapa en fungerande server.

```
1  'use strict';
2  var mongoose = require('mongoose');
3  var express = require('express');
4  var app = express();
5  var bodyParser = require('body-parser');
6  var fs = require('fs');
```

Figur 15. *Server.js-filens behövliga moduler.*

Mongoose är en Node.js-modul för MongoDB-databasen. Mongoose representerar data i databasen med ett schema på servern. Schema-filen ”recipes.js” (Figur 16) innehåller samma fält som kollektionen i den egentliga databasen (Figur 10). För att öppna en förbindelse mellan servern och databasen skrivs

mongoose.connect('mongodb://localhost/recipes');

i server-filen efter moduldeklarationerna. Mongoose skapar då en förbindelse till kollektionen som finns i mappen *C:/data/db* som i detta exempel finns på localhost. Sedan skall schema-filen inladdas till servern för att kunna hämta data:

```
fs.readdirSync(__dirname + '/models').forEach(function(filename){
        if (~filename.indexOf('.js')) require(__dirname + '/models/' + filename);
});
```

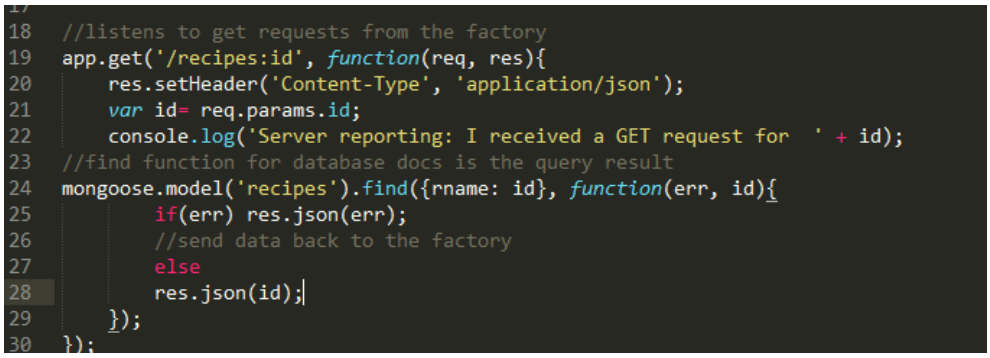
”fs” är förkortning av fil system (eng. file system) och den används i funktionen ovan för att i projektet söka från mappen ”models” efter alla filer som slutar med ”.js”. Då en motsvarande fil hittas inladdas den till servern.

```
'use strict';
/**
 * Module dependencies.
 */
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;
/**
 * Recipe Schema
 */
var RecipeSchema = new Schema({
    rimg: {
        type: String,

    },
    rname: {
        type: String,
        trim: true,
    },
    rdesc: {
        type: String,
        trim: true
    },
    ingredients: {
        type: [String],
    },
    stages: {
        type: [String],
    },
    prepimg: {
        type: [String],
    }
});
mongoose.model('recipes', RecipeSchema);
```

Figur 16. *Schema-filen ”recipes.js” för mongoose-modulen.*

Express (Figur 15) är gränssnittet för HTTP-modulen i Node.js. Denna model sköter om funktionalitet såsom exekvering av en GET-förfrågning till databasen (Figur 17).



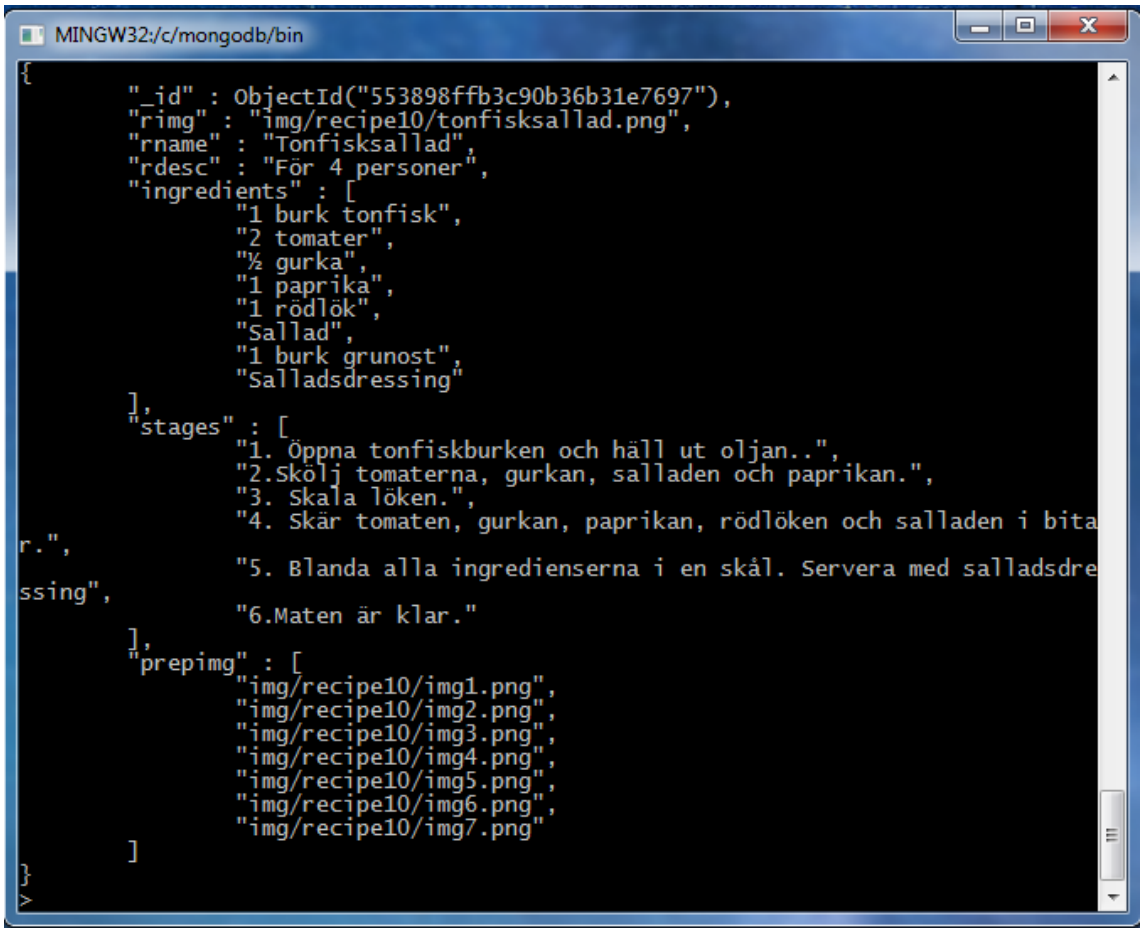
```
//listens to get requests from the factory
app.get('/recipes:id', function(req, res){
    res.setHeader('Content-Type', 'application/json');
    var id= req.params.id;
    console.log('Server reporting: I received a GET request for  ' + id);
//find function for database docs is the query result
mongoose.model('recipes').find({rname: id}, function(err, id){
        if(err) res.json(err);
        //send data back to the factory
        else
        res.json(id);
    });
});
```

Figur 17. *En funktion som söker i databasen på basen av en id som användaren vill söka med.*

5.3 Databas

Som databas i e-kokboksprojekt användes MongoDB. MongoDB är en NoSQL-databas som sparar data som separata dokument i en kollektion istället för som tabeller i en traditionell SQL-databas.

MongoDB inladdas från (MongoDB 2015.). Efter att MongoDB är installerad skall man skapa en data-mapp i C: -katalogen. I datamappen skall man ännu skapa en mapp som heter ”db”. I ”db”-mappen sparas databasen. Därefter kan man skapa och modifiera egen databas med att navigera med kommandotolken till mappen där man sparat MongoDB. När man navigerat fram skall man exekvera kommandot ”mongo” för att skapa och redigera databaser. För att skapa en ny kollektion i databasen med namnet recipes skriver man ”use recipes”. För att skapa ett nytt dokument i kollektionen skriver man ”db.recipes.insert({RecipeName:makaronilåda}) ”.



Figur 18. *Datamodell för receptet Tonfisksallad i e-kokboksprojektet.*

I Figur 18 visas den allmänna dokumentuppbyggnaden i e-kokboksprojektet. E-kokboken har tio recept och de representeras som tio dokument i kollektionen ”recipes”.
Varje dokument har samma fält som dokumentet i Figur 16.

För att tillåta sökning i databasen skall man skapa indexvärden för de element man vill genomsöka. I e-kokboksprojektet behöver man endast söka ingredienser från kollektionen recipes. Detta görs med kommandot ”db.recipes.createIndex({"ingredients": "text"})”. Indexvärden tillåter MongoDB att snabbt processera och svara på förfrågningar genom att skapa en lättanvänd och effektiv representation av dokumenten eller av fälten (i dokument) i en kollektion.

6 A/B-TESTNING

A/B-testning har blivit ett standardtestningsformat för utveckling av webbportaler med stor användarpopulation. I Silicon Valley är A/B-testning det mest använda testningssättet för att förbättra produkter på nätet. Genom A/B-testning kan man testa nya funktioner eller ändringar i realtid. (Brian 2012)

I A/B-testning har man två versioner (A & B) som skall testas av testgrupper. Grupperna får en något annorlunda version av t.ex. en webbsida. En grupp testar den nya versionen och man kan jämföra dess beteende på webbsidan med den andra gruppens beteende och dra slutsatser därifrån. Om den nya versionen visar sig vara bättre – får mera klickar, användaren spenderar längre tid på webbsidan eller det görs mera köp - kommer den nya versionen att ersätta den gamla. (Brian 2012)

Den första A/B-testet på en nätsida gjordes av Google på den 27 februari år 2000. Testet gick ut på att undersöka om den dåtida mängden av sökresultat som visas (10 sökresultat som används än idag) är optimal för användaren. Testet utfördes så att till 0.1% av användarna visades 20 sökresultat på webbsidan, till en annan 0.1% av användarna visades 25 resultat och ännu en grupp fick se 30 sökresultat. En teknisk malör gjorde dock experimentet till en katastrof. Sidorna som visades till testgrupperna laddades signifikant långsammare än till kontrollgruppen. Detta visade dock också att en fördröjning på en

tiondels sekund hade väldigt stor inverkan på användarupplevelsen. År 2011 utförde Google mera än 7000 A/B-tester på sin sökalgoritm. (Brian 2012)

Flera stora företag utför hela tiden A/B-tester på sina webbtjänster för att testa potentiella förbättringar. (Brian 2012)

6.1 Testningsplan

Testningsplanen beskriver hur testtillfället utförs på ett optimalt sätt. För detta testtillfälle behöver man 2-4 testpersoner som indelas i två grupper. Gruppstorleken är då 1-2 personer beroende på hur många testpersoner som man lyckas få med. Testpersonerna bör vara synskadade med en liknande grad av synskada. Om det finns en alltför stor skillnad i graden av synskada kommer testresultatet inte att vara giltigt eftersom testpersonerna kommer att uppleva designen av webbsidan på ett annat sätt.

Själva testet går ut på att de båda grupperna skall tillreda två olika maträtter med hjälp av e-kokboken. Grupperna skall tillverka maträtter genom att använda båda layouterna en gång. Testet kan göras så att grupperna inte gör samma recept med samma layout, dvs. ena gruppen gör recept 1 med layout 1 medan andra gruppen gör recept 1 med layout 2. När detta är gjort kan man se att hur bra grupperna klarade av matlagningen och fanns det t.ex. några problem i att förstå anvisningarna i receptet. Om det inte finns problem i att förstå layouten kan man kolla med tiden vilken av layouten som gav snabbare resultat. Testpersonerna ger också en feedback på vad som fungerade och vad som inte fungerade.

Före testtillfället skall man bestämma vilka maträtter som skall tillredas under testtillfället. Valda maträtter skall inte vara tidskrävande, sallader och dylikt. Man skall skaffa ingredienser så att de räcker för fyra recept.

Testtillfället kommer att äga rum i Arcadas Smart-kök.

6.2 Utförande

Testet utfördes 16.11 2015 klockan 10.00 i Arcadas Smart-kök. I testet deltog två testpersoner. Testpersonerna var än äldre dam och en medelålders kvinna. Båda testpersonerna är vana användare av dator/tablet.

Testet började med att testpersonerna fick lite bläddra runt på sidan. Vid detta skede märktes det redan att texten var för liten för sidan och att man måste i själva webbläsaren zooma in för att få texten större. Efter att allt var justerat var det dags att påbörja själva testet.

Grupp 1 (Äldre dam) började med att med Layout 1 skapa receptet Grekisk Sallad. Grupp 2 (medelålders kvinna) började med att skapa Vegetarisk Pasta med Layout 2. Damen i grupp 2 hade kanske lite mera datorvana och bättre syn än den äldre damen i grupp 1 så hon hade inga problem med att skapa recepten, fast hon också skulle ha sidan in zoomad. Grupp 1, den äldre damen, hade mera problem med att läsa sidans innehåll och speciellt siffror var svåra att urskilja.

Grupp 1 och Grupp 2 var nästan samtidigt färdiga fast receptet i Grupp 2 var mera krävande. Grupp 2 fortsatte med att skapa den Grekiska salladen med Layout 1, vilket gick ganska snabbt och var enkelt. Damen i grupp 1 dock försökte påbörja det andra receptet med Layout 2 men konstaterade att det inte kunde bli till något, då hon måste anstränga sig för mycket för att ordentligt se receptets innehåll.

Efter testet gick damen i grupp 1 ännu igenom det misslyckade receptet med enbart pictogram-bilderna. Hon tyckte att detta var bättre och hon kunde beskriva varje skede i tillverkningen av receptet utan att det visades någon text.

Problemet som uppkom med pictogrammen var att det är kanske svårt att urskilja vilka ingredienser som är visade i dem, p.g.a. att de kan tolkas på olika sätt.

Båda testpersoner tyckte att denna e-kokbok kanske skulle fungera bättre på en tablet än på en helt vanlig dator.

Testet tog slut klockan 12.00. Grupp 2 klarade av båda recepten med två olika layout medan Grupp 1 endast klarade av att med Layout 1 att skapa receptet Grekisk Sallad.

6.3 Resultat

Båda testpersonerna klarade av testet med Layout 1, medan Layout 2 förorsakade ett misslyckat försök och ett lyckat. Från detta kan man dra slutsatser att Layout 2 är mindre lämplig för synskadade personer.

I Layout 2 fanns det problem med att förstå receptets faser ordentligt p.g.a. texten inte finns direkt under pictogrammen utan i en lista under bilderna. D.v.s. den aktiva texten är inte direkt under bilderna och man måste scrolla ner på sidan för att se texten. Testpersonerna klagade på att i Layout 2 kunde man inte se vilka ingredienser som behövs för maträtten samtidigt som man är i receptets tillverkningsskede.

7 DISKUSSION OCH SLUTSATSER

Examensarbetet beskriver utvecklingen av en elektronisk kokbok för personer med kognitiva nedsättningar eller synskador. En produkt skapades efter undersökning av existerande e-kokböcker och efter diskussioner med beställaren om applikationen. Produkten testades för att se om den fungerade såsom den var avsedd för användargruppen. Som resultat fick man en klar skillnad mellan de två versionerna av applikationen.

För att få en mera optimerad applikation borde testpersonerna ha tagits med i ett tidigare skede av applikationsutveckling för att kunna åtgärda brister i ett tidigare utvecklingsskede. Det finns ett behov av applikationer avsedda för personer med olika slags handikapp.

Under projektets gång lärde jag mig mycket nytt om hur man skapar en webbapplikation, vilka tekniker som skall användas samt hur man först måste föreställa projektet i huvudet/på papper före man börjar utveckla själva applikationen.

Som vidareutveckling kunde man implementera för applikationen ett röstgränssnitt som skulle läsa ut receptets innehåll. För att röstgränssnittet skall fungera borde applikationen programmeras som en nativ applikation till t.ex. Android och iPhone/iPad. Då skulle man bättre kunna utnyttja pekskärmen för en användarvänligare upplevelse. Användargränssnittet borde ändras så att texten skulle vara större. Detta kan dock förvränga annat innehåll i applikationen, vilket måste tas i beaktande.

KÄLLOR

Cantelon, Mike; Harte, Marc; Holowaychuk, T.J; Rajlich, Nathan. 2014, *Node.js in Action*, 2 uppl., Shelter Island, NY, Manning Publications Co, 396 s.

Christian, Brian, 2012. Tillgänglig: http://www.wired.com/2012/04/ff_abtesting/
Hämtad 24.5.2015

Clymer, Benjamin. Tillgänglig: http://www.forbes.com/sites/booked/2010/04/12/apples-ipad-brings-easy-reading-to-the-blind/ Hämtad 17.8.2015

Dayley, Brad. 2014, *Node.js, MongoDB and AngularJS Web Development*, Upper Saddle River, NJ, Pearson Education, 696 s.

Flanagan, David. 2011, *JavaScript, the definitive guide-sixth edition*, Sebastopol, CA, O'Reilly, 994 s.

Galitz, Wilbert. 2007, *The Essential guide to user interface Design: An introduction to GUI design principles and techniques*, 3 uppl., Wiley, 888 s.

Kotikokki. 2015 Tillgänglig: http://www.kotikokki.net/reseptit
Hämtad 18.11.2015

Lerner, Ari. 2013, *ng-book – The complete book on AngularJS*, Fullstack.io, 608 s.

Lubbers, Peter; Salim, Frank; Albers, Brian. 2011, *Pro HTML 5 Programming –second edition*, Apress, 352 s.

Mikovski, Michael; Powell, Josh. 2014, *Single Page Applications: JavaScript end-to-end*, Shelter Island, NY, Manning Publications Co, 409 s.

MongoDB. 2015. Tillgänglig: https://www.mongodb.org/downloads
Hämtad 17.5.2015

Node.js. 2015. Tillgänglig: https://nodejs.org/download
Hämtad 17.5.2015

UI-router. 2015. Tillgänglig: https://github.com/angular-ui/ui-router/wiki
Hämtad 17.5.2015

Valio. 2015 Tillgänglig: http://www.valio.fi/reseptit
Hämtad 18.11.2015