

MÄTNING AV SVARSTIDER PÅ GRAFISKT TUNGA WEBBUTIKER

Med användning av AngularJS, Twitter
Bootstrap och Foundation.

MEASURING RESPONSE TIME OF HEAVY GRAPHIC E-COMMERCE

Using AngularJS, Twitter Bootstrap and
Foundation.

Examensarbete inom huvudområdet Datalogi
Grundnivå 30 högskolepoäng
Vårtermin 2015

Jango Saleh

Handledare: Henrik Gustavsson
Examinator: Mikael Berndtsson

Sammanfattning

Utvecklingen av e-handelsbutiker har utvecklats enormt mycket sedan början av World Wide Web. Olika ramverk och programmeringsspråk används idag för att bygga webbsidor och webbapplikationer. I och med att hemsidors komplexitet och mängden kod blir större, är det viktigt att det redan från början väljs ett rätt ramverk när en utvecklare skall ge sig in i utvecklingsfasen. Detta arbete undersöker ifall ramverket AngularJS har lägre svarstider med egenskriven stilmall än de stilmallar som används i Twitter Bootstrap och Foundation när en e-handelsbutik skall tillämpas för webben. Svarstider kommer att mätas på tre olika webbplatser som efterliknar en e-handelsbutik.

Nyckelord: AngularJS, Twitter Bootstrap, Foundation JavaScript, Html5, CSS, Mysql och svarstider.

Innehållsförteckning

1. Introduktion	1
2. Bakgrund	2
2.1. HTML och HTML5	2
2.2. Vad är ett ramverk?	2
2.3. Model-View-Controller	3
2.3.1. JavaScript	3
2.3.2. AngularJS	4
2.4. Responsiva Ramverk.....	4
2.4.1. CSS och CSS3	5
2.4.2. Twitter Bootstrap.....	5
2.4.3. Foundation	6
3. Problemformulering	7
3.1. Hypotes	7
3.2. Metodbeskrivning.....	7
3.2.1. Avgränsningar i metoden.....	8
3.2.2. Begränsningar i metoden.....	8
3.2.3. Alternativa metoder.....	8
3.2.4. Forskningsetik.....	8
4. Genomförande.....	10
4.1. Förstudie.....	10
4.2. Progression.....	10
4.3. Implementation av e-handelsbutik	11
4.3.1. Mysql	11
4.3.2. PHP Hypertext Preprocessor.....	12
4.3.3. AngularJs med Twitter Bootstrap.....	13
4.4. Pilotstudie	15
4.4.1. Hårdvara	15
4.4.2. Mjukvara	15
4.4.3. Ramverk	15
4.4.4. AngularJs med olika stilmallar	15
4.4.5. Sammanställning av pilottest.....	16
5. Utvärdering	17
5.1. Presentation av undersökning	17
5.1.1. Utrustning	17
5.2. Analys	18
5.2.1. Mätning 1: AngularJS med olika stilmallar - nivå 1	18
5.2.2. Mätning 2: AngularJS med olika stilmallar - nivå 2	19
5.2.3. Mätning 3: AngularJS med olika stilmallar - nivå 3	20
5.2.4. Mätning 4: AngularJS med olika stilmallar - nivå 4	21
5.3. Slutsatser.....	22
6. Avslutande diskussion	23
6.1. Sammanfattning.....	23
6.2. Diskussion	23
6.3. Framtida arbete	24
Referenser.....	25

1. Introduktion

Webbsidors komplexitet och mängden kod blir allt större, därför är det viktigt att det redan från början väljs ett rätt ramverk när en utvecklare skall ge sig in i utvecklingsfasen. En webbutvecklare skall inte välja ett ramverk som endast stävar efter projektets behov. Istället bör valet av ett ramverk eftersträva hög kvalitet hos programkoden samt låga svarstider (Gizas, et al., 2012). Användarna förväntar sig att webbsidor ska vara smidiga och att navigationen på sidan ska ha så låga svarstider som möjligt, därför är svarstider en viktig fråga inom webbutveckling (Richard-Foy, et al., 2013).

I en artikel nämns det att en undersökning gjordes på e-handelsbutiken Amazon där det visade sig att verksamheten förlorade 1% av försäljning på grund av ökad svarstid på 100 ms (Wei & Xu 2011). Här framgår det tydligt att användarna stävar efter snabba svarstider. Uppnås inte dessa krav så är risken att användaren besöker någon annan webbsida.

”Early experiments at Amazon even showed a one percent sales loss for an additional 100 ms delay. Therefore, obtaining quantitative measures of the response times in a timely and cost-effective manner is fundamental to manage e-commerce services across a diverse and rapidly changing client population”

Wei och Xu, 2011, s.773

Frågeställningen som skall undersökas i arbetet är ifall ramverket AngularJS har lägre svarstider med egenskriven stilmall än de stilmallar som används i Twitter Bootstrap och Foundation när en e-handelsbutik skall skapas. Dessutom kommer det att undersökas hur AngularJS påverkas när responsiva ramverk som Twitter Bootstrap och Foundation implementeras. Både dessa ramverken använder stilmallar som innehåller en mängd rader kod. Stilmallarna är skriven för att anpassa sig till olika enheter och skärmstorlekar.

Hypotesen är att AngularJS kommer ha en kortare svarstid än Twitter Bootstrap och Foundation eftersom handskrivna stilmallar brukar inte innehålla mycket kod samt att de endast innehåller kod som är nödvändig. Att använda färdiga stilmallar från Twitter Bootstrap och Foundation kan leda till att mycket av koden inte kommer till användning och att felsökningar blir allt svårare då det inte finns kontroll över koden. Att ha för mycket kod i sina stilmallar som inte har någon funktion kan leda till högre svarstid (Vernica, et al., 2015).

Undersökningsmetoden experiment kommer att tillämpas för att undersöka frågeställningen. Svarstider kommer att mätas på en webbplats som efterliknar en e-handelsbutik där webbplatsen innehåller produkter samt en sökfunktion. Produkterna kommer att lagras i en databas så att webbplatsen kan hämta data när användaren använder sökfunktionen. De ramverk som kommer att implementeras är AngularJS, Twitter Bootstrap och Foundation. En stilmall kommer implementeras med AngularJS och sedan kommer svarstider att mätas. Liknande mätningar kommer att utföras med stilmallar från Twitter Bootstrap och Foundation i AngularJS för att se skillnaderna på svarstiderna.

2. Bakgrund

I detta kapitel beskrivs grunderna inom de ramverk och programmeringsspråk som kommer att användas för att genomföra detta experiment. Sedan kommer det en mer genomgripande beskrivning på hur experimentet utförs.

2.1. HTML och HTML5

HyperText Markup Language (HTML) är det enda programmeringsspråket för att skapa webbsidor. Programmeringsspråket HTML använder sig av element som består av taggar som skrivs i så kallade vinkelparenteser, <tagg></tagg>. Fördel med att använda sig av HTML är ifall en utvecklare vill strukturera sin webbsida genom att använda paragrafer, tabeller, rubriker och länkar. Dock saknar den element för bildspel, footer eller menyer (Shelly, et al., 2007).

Den senaste versionen av HTML är HTML5. HTML5 har infört en mängd nya element som gör webbsidorna mer logiska. I tidigare versioner har det använts id eller class. Med HTML5 går det att använda t.ex <article>, <footer>, <time> eller <header>. Det infördes enorma förändringar med HTML5 och ett av dem största ändringen var att spela upp video och ljud i webbläsaren utan att behöva installera flash plug-in. Formulärshanteringen förbättrades genom att inför nya funktioner så som placeholder, validering och obligatoriska inmatningsfält. I de tidigare versionen användes programmeringsspråket, JavaScript för att skapa dessa funktioner. Se figur 1 för exempel på HTML5 kod.

```
<form>
  <input class="form-control
    type="text" name="name" placeholder="Ange ditt namn">

  <input class="form-control
    type="email" name="email" placeholder="Ange din epost">

  <textarea class="form-control animated
    name="message" placeholder="Meddelanden..."
    cols="30" rows="10"></textarea>

  <button type="submit">
    <i class="button"></i>Skicka</button>
</form>
```

Figur 1 Exempel på HTML5 kod

2.2. Vad är ett ramverk?

Utvecklingen av webbapplikationer har utvecklats enormt mycket sedan början av World Wide Web. En mängd olika ramverk och programmeringsspråk används idag för att bygga webbsidor och webbapplikationer (Pop, et al., 2014). Som webbutvecklare kan det vara enormt frustrerande att utveckla webbsidor där en mängd samma rader kod skrivs om flera gånger och att samma problem uppstår under utvecklingen. För att undvika dessa problem så används ramverk.

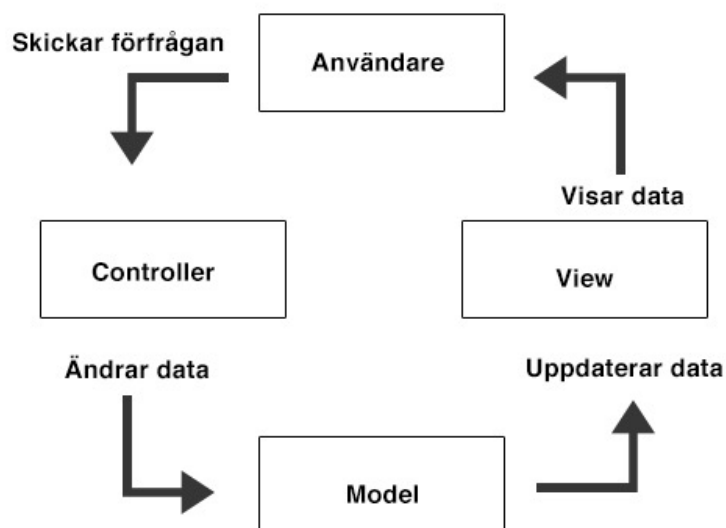
En ramverk är ett färdigutvecklat bibliotek som innehåller funktioner, design och element för att underlätta utvecklingen under ett projekt. I en artikel nämner Balasubramanee, et al.,

(2013) att fördelen med ett ramverk är att en utvecklare kan fokusera på viktigare aspekter så som användbarhet på en webbsida.

Det finns hundratals olika ramverk ute på Internet och som webbutvecklare kan det vara svårt att hitta rätt ramverk för ett specifikt projekt. Risken med att välja fel ramverk kan leda till att en stor mängd av kod blir oanvändbar och filstorleken blir större än vad som är idealiskt.

2.3. Model-View-Controller

Model-View-Controller (MVC), är en teknik som används för att dela upp koden. Tekniken används när en utvecklare vill strukturera i sin webbsida. När det skapas en webbsida med en MVC-ramverk så vill en utvecklare separera på model, view och controller vilket kan vara till en stor nytta när det används olika programmeringsspråk för att skilja på logik och presentation. Att skilja dessa åt kan leda till fler olika vyer för att presentera data samt att återanvändningen av kod blir lättare att använda för framtida projekt. Model, view och controller har olika uppgifter och den första komponenten, vilket är controller, sköter interaktioner som utförs av användaren så som inmatningar på webbsidan eller knapptryck. Model ser till att presentera all data och innehåll. View har i uppgift att skapa en gränssnitt för användaren (Pop, et al., 2014). Se figur 2 för exempel på MVC.



Figur 2 Exempel på MVC struktur

2.3.1. JavaScript

Ett av de populäraste programmeringsspråket för en webbläsare är Javascript (Gizas, et al., 2012). För att skapa en attraktiv och responsiv hemsida är det JavaScript som bör användas. Stora webbapplikationer som, Google, Yahoo och Facebook använder sig av JavaScript på större delar av sina webbplatser. Anledningen till att den ha blivit så populär är för att koden körs direkt, till skillnad från Java eller C++ där koden måste kompileras innan den körs. En fördel med att använda JavaScript på sin webbsida är att aktiviteter hanteras direkt i användarens webbläsaren genom att använda HTML-taggen, `<script></script>`. Se figur 3 för exempel på Javascript kod.

```

window.onload = function () {
    document.getElementsByTagName("RADERA").item(0).onclick = function ()
    {
        document.getElementsByTagName("RADERA").item(0).innerHTML = "";
    };
};

```

Figur 3 Exempel på en Javascript funktion som talar om att radera t.ex. en bild när användaren klickar på bilden

2.3.2. AngularJS

AngularJS är utvecklat av Google och är ett JavaScript ramverk med Model-View-Controller (MVC), där gräsnittet genereras direkt i webbläsaren till skillnad från en traditionell applikationsarkitektur, SpringMVC, där gräsnittet genererades från servern (Balasubramanee, et al., 2013). AngularJS används för att utveckla det grafiska gränssnittet på en webbsida. En stor fördel med att använda sig av AngularJS är att den har en väldigt enkel syntax med en simpel struktur och det går väldigt fort att bygga funktioner. En annan fördel är deras "two-way-data-binding". Med data-bindning så menas det att användarens vy uppdateras så fort datan ändras i kodningen. Se figur 4 för exempel på AngularJS kod.

```

<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"
></script>
</head>
<body>
    <div ng-app="">
        <p>Personnummer: <input type="text" ng-model="pnr" /></p>
        <h3 ng-bind="pnr"></h3>
    </div>
</body>
</html>

```

Figur 4 Exempel på AngularJS kod som tillåter användaren att ange ett personnummer som genereras direkt i webbläsaren

2.4. Responsiva Ramverk

Mobiler och surfplattor har blivit allt vanligare i vår vardag. Idag används det olika enheter för att snabbt söka efter information eller handla på nätet, därför är det viktigt med en webbsida som är anpassad efter olika enheter. För att anpassa en webbsida efter olika enheter så bör det användas responsiv design vilket betyder att det grafiska delen av webbsidan tillämpar sig efter skärmstorleken (Vernica, et al., 2015). Många webbsidor på Internet kräver en attraktiv och responsiv webbsida. Idag finns det många ramverk som stödjer responsiv design och två av de vanligaste ramverkan är Twitter Bootstrap och Foundation som använder media queries i sina CSS-filer. Media queries talar om hur gränssnittet ska se ut efter en viss skärmstorlek. Att använda sig av ramverk kan leda till sämre svarstider då mängden kod utökas när en e-handelsbutik ska anpassas efter olika

enheter. I en artikel nämner Wei & Xu (2011) att företaget Amazon förlorade 1 % av försäljning på grund av ökad svarstid på 100 ms. Se figur 5 för exempel på Media queries.

```
@media all and (min-width: 600px) {
  .bilder {
    text-align: left;
  }
}

@media all and (min-width: 400px) {
  .bilder {
    text-align: center;
  }
}
```

Figur 5 Exempel på hur bilder ska visas på olika skärmstorlekar

2.4.1. CSS och CSS3

Cascading Style Sheets (CSS) används för att presentera text och bilder från en HTML dokument. CSS används för att formge olika typsnitt, ändra textstorlek eller justera placering av bild och text. Även CSS har en nyare version, CSS3 som erbjuder en mängd nya funktioner så som skuggor bakom text, rotation, responsiv design eller runda hörn. CSS3 medför även responsiva lösningar för olika skärmstorlekar genom att använda tekniken, media queries i stilmallarna.

2.4.2. Twitter Bootstrap

Det populäraste ramverket för responsiv design är Twitter Bootstrap (Vernica, et al.). Twitter Bootstrap är ett ramverk med färdiga element och funktioner som är sammansvetsade vilket underlättar för utvecklaren när den tar fram en webbsida. Genom att använda dess bibliotek så är det med all säkerhet att dessa element och funktioner kommer att fungera tillsammans. Twitter Bootstrap är känd för sitt grid system som består av rader och kolumner vilket är själva strukturen för en webbsida. Varje rad kan innehålla högst 12 kolumner där det anges ett värde från 1-12. Se figur 6 för exempel på Twitter Bootstrap grid system.

```
<div class="row">
  <div class="col-sm-4"></div>
  <div class="col-sm-4"></div>
  <div class="col-sm-4"></div>
</div>

// Här kommer vi få 3 kolumner som täcker 100 % av raden. 4 x 3 = 12.

<div class="row">
  <div class="col-sm-12"></div>
</div>

// Här kommer vi få 1 kolumn som täcker 100 % av raden. 12 x 1 = 12.
```

Figur 6 Exempel på Twitter Bootstrap grid system

2.4.3. Foundation

Foundation är skapad av företaget ZURB och är identisk med ramverket, Twitter Bootstrap. Det som skiljer dem åt är kodnings semantiken där Foundation är något tydligare. Med Foundation går det att välja vilka funktioner som skall användas till en webbsida. Ett 12 kolumners grid system används även i detta ramverk men här har dem valt att förtydliga koden. Se figur 7 för exempel på Foundation grid system.

```
<div class="row">
  <div class="small-4 columns"></div>
  <div class="small-4 columns"></div>
  <div class="small-4 columns"></div>
</div>

// Här kommer vi få 3 kolumner som täcker 100 % av raden. 4 x 3 = 12.

<div class="row">
  <div class="small-12 columns"></div>
</div>

// Här kommer vi få 1 kolumn som täcker 100 % av raden. 12 x 1 = 12.
```

Figur 7 Exempel på Foundation grid system

3. Problemformulering

Att förutse svarstider på webbapplikationer kan ibland vara en stor utmaning. Richard-Foy, et al., (2013) menar att stora grafiska webbsidor måste förlita sig på mycket mer kod vilket kan leda till sämre svarstider. Som Richard-Foy, et al., (2013) vidare nämner så förväntar sig användarna att en webbsida ska ha så låga svarstider som möjligt, därför är låga svarstider en viktig fråga inom webbutveckling.

När AngularJS används för att ta fram en e-handelsbutik så är inte alla bitarna på plats. Webbsidan är inte responsiv och kräver en del kodning. I en artikel nämner Kucukcay & Benyoucef (2014) att 40% av Internetanvändningen sker via mobila enheter. Genom att använda sig av responsiv design för att tillämpa webbsidan efter olika skärmstorleken så ökar mängden kod, vilket kan leda till att svarstiderna försämras (Vernica, et al., 2015).

I en artikel nämner Chittaro & Ranon (2002) att utmaningen med att skapa en e-handelsbutik är att utforma designen på webbsidan för att på ett effektivt sätt presentera data. Författarna framhäver i artikeln att stor mängd av grafisk kodning kan leda till att besökarna får det svårt för att hitta de produkter dem söker efter. Enligt Chittaro & Ranon (2002) finns det en risk att kunden avbryter köpet även om produkten är funnen.

Att använda grafiskt tunga webbutiker påverkar svarstider och ligger till grund för kundernas beslut vid köp av en produkt. I en artikel av Chow, D. (2001) nämns det att svarstider är en viktig faktor när det gäller e-handelsbutiker, detta visades även i en studie där 48% av 12000 besökare avbröt sitt köp på grund av höga svarstider på webbsidan.

Frågeställningen som detta arbete skall fokusera på är om grafiskt tunga webbutiker får högre eller lägre svarstid när det används responsiv ramverk tillsammans med AngularJS eller om det är bättre att använda handskriften stilmall.

3.3.1. Hypotes

I vår undersökning kommer arbetet att försöka få fram svarstider genom att implementerar responsiva ramvek samt handskriften stilmall med AngularJS. Förmodligen så kommer den handskrivna stilmallen ha lägre svarstid än de responsiva ramverken eftersom större mängd kod kan försämra svarstider på webbsidan.

3.2. Metodbeskrivning

Den metod som har valt är experiment. Enligt Wohlin, et al., (2012) är huvudsyftet med ett experiment att utvärdera en hypotes eller relation. Experimentet tar fram en samband på ett problem mellan olika orsaker och påverkan. För att få en bättre bild på om hypotesen stämmer eller inte så kommer mätning av svarstider ske genom experiment. I detta experiment kommer arbetet att använda sig av ett JavaScript ramverk, AngularJS som en grund för vår webbsida. Implementation av två responsiva ramverk kommer att ske samt ett handskriften stilmall.

Mätningen kommer baseras på en liknande experiment som gjordes av Gizas, et al., (2012), där mätning av svarstider utfördes på olika JavaScript ramverk. Deras mätning grundades på antalet rader kod samt filstorlek på de olika ramverken. Arbetet kommer att utföra ett liknande experiment där mätningen av svarstider kommer att ske via en sökfunktion som kommer att implementeras på webbsidan. Inmatning av en viss data kommer att sökas och det som kommer att mätas är exekveringstiden från att sökningen blivit skickat till

databasen och från att webbsidan renderat fram de data som efterfrågas i sökfunktionen. Det här arbetet kommer att fokusera sig på svarstider med olika ramverk samt handskrivna stilmall.

3.2.1. Avgränsningar i metoden

Valet av dessa ramverk baserar på olika kriterier. Ramverken var tvungen att vara JavaScript baserat, Open-Source samt att de skulle vara responsiva. AngularJS, Twitter Bootstrap och Foundation är bland dem största ramverken ute på marknaden med en detaljerad dokumentation på hur installationen av ramverken utförs.

3.2.2. Begränsningar i metoden

En nackdel med att utföra vårt experiment i en online-miljö är det finns risk att en webbsida havererar när ändringar sker i källkoden eller om webbsidan blir allt för belastad under mätningen vilken kan leda till vilseledande resultat (Wohlin, et al., 2012). Som Wohlin, et al., (2012) vidare nämner, så blir kontrollen över experimentet större ifall det sker i en offline-miljö, därför kommer det att skapas en webbsida som liknar en verklig webbplats och dess funktioner. Webbplatsen blir mer kontrollerad och inga verkliga webbsidor riskeras stängas ner på grund av felaktigheter.

Genom att använda en metod som är tillräckligt kontrollerad så kan mätning av svarstider, där även små detaljer kommer till användning i analysen. Mätmetoden som används kommer vara identiskt under varje mätning eftersom mätningarna är teknikbaserat till skillnad från att använda sig av riktiga användare som i sin tur ska analysera skillnaderna i svarstiderna.

Enligt Wohlin, et al., (2012) så kan en användare fånga upp olika objekt under en mätningen, vilket kan leda till att trovärdigheten på resultatet minimeras. Det nämns även att det blir svårare för en användare att utföra ett kvantitativt mätning då man strävar efter precisa mätningar.

3.2.3. Alternativa metoder

Att skapa en prototyp av en riktig webbsida och använda sig av användarstudier är en annan metod som hade kunnat användas i detta arbete. Att fördjupa sig i fallstudier genom att analysera användarnas beteende när de använder sig av olika ramverk. Genom att använda användarstudier så kan man analysera skillnaderna på olika beteenden utifrån en annan mätning. Att mäta svarstider baserat på användarnas beteenden hade inte blivit lika precisa som i ett experiment eftersom fokuset ligger på den aktuella situationen till skillnad från experiment, där det kan få fram flera faktorer som kan vara avgörande för olika resultat som sedan kan vara jämförbara (Wohlin, et al., 2012). Det nämns även att experiment hänger samman till de variabler som påverkas medan fallstudier utser de variabler som ska företräda representativa situationer. Om arbetet hade fokuserat sig på användbarhet så hade användarstudier varit ett alternativ för att mäta användarens beteende när de försöker hitta olika produkter på webbplatsen.

3.2.4. Forskningsetik

Arbetet kommer att endast ske på en dator så att resultat blir så rättvis som möjligt. Det kommer att användas ett operativsystem och ramverken kommer att testas med samma internetuppkoppling. Arbetet kommer att mätas på tre olika webbläsare, Chrome, Firefox och Safari, ifall problem skulle uppstå med någon av webbläsarna eller om det skulle finnas någon typ av felaktigheter på en viss webbläsare. För att garantera

opartiskheten när det gäller valet av webbläsare så kommer mätning av svarstider ske på olika webbläsare. Det skulle vara oetiskt att endast mäta på en webbläsare.

Inga andra ramverk eller programmeringsspråk kommer att användas för att påverka de mätningarna som genomförs. Inga testpersoner kommer att användas i detta arbetet vilket betyder att etiska problem, så som hantering av personlig data inte kommer att publiceras. För att arbetet skall kunna bidra med ett pålitligt resultat så kommer all kod, alla verktyg och den hårdvaran som används i detta arbete att publiceras så att andra personer kan återskapa det vi har gjort i detta arbete (Cai, et al., 1998). Vi kan även poängtera att resultatet kanske inte blir samma eftersom det kan vara olika förutsättningarna. Dock kommer vi ha samma tillgångar då vi publicerar all kod och alla verktyg.

4. Genomförande

Denna del beskriver hur e-handelsbutikerna framställts, vilka problem som uppstått under utvecklingens gång, vilka inspirationskällor som används för detta arbete samt resultat från pilotstudien. För att förverkliga syftet i detta mål, behöver mjukvaruprogramet MAMP installeras. MAMP används för att skapa en lokal webbserver samt en databas som i sin tur kan kopplas till olika webbapplikationer. Tre identiska e-handelsbutiker kommer att implementeras på denna server och alla webbsidor kommer använda sig av en och samma databas.

4.1. Förstudie

För att ta till sig inspiration och idéer på hur e-handelsbutiker skall implementeras på den lokala webbservern användes en förstudie. *AngularJS: Up and Running (2014)* har varit till en stor hjälp för att förstå hur ramverket fungerar. Boken ger även en bred kunskap om hur olika funktioner skall implementeras för att appliceras på webben.

En annan inspirationskälla som har varit till stor nytta är *Bootstrap: Responsive Web Development (2013)*. Boken ger en helhetsbild av ramverket och dess funktioner samt att boken ger en detaljerad beskrivning på hur implementationen ska gå till och hur funktionerna skall tillämpas på en webbsida.

Även *Learning Zurb Foundation (2014)* har varit en inspirationskälla. Boken var väldigt detaljrik och innehöll stor mängd av kodexempel med stegvisa förklaringar på hur implementationen skall utföras samt vilka filer som skall ändras.

Utöver dessa tre böcker har *AngularJS*, *Twitter Bootstrap* och *Foundation* detaljerade beskrivningar på respektive hemsidor, där det visas hur strukturen samt funktionaliteten skall implementeras. Dessa guider har varit en stor inspirationskälla för att förstå hur ramverken fungerar och vad som krävs för att utveckla en e-handelsbutik som är integrerad med dagens Internet. Informationen som är tillgänglig ger en stor uppfattning på hur funktionerna är uppbyggda och hur de fungerar. De beskriver även när funktionerna är användbara vid implementering av e-handelsbutik.

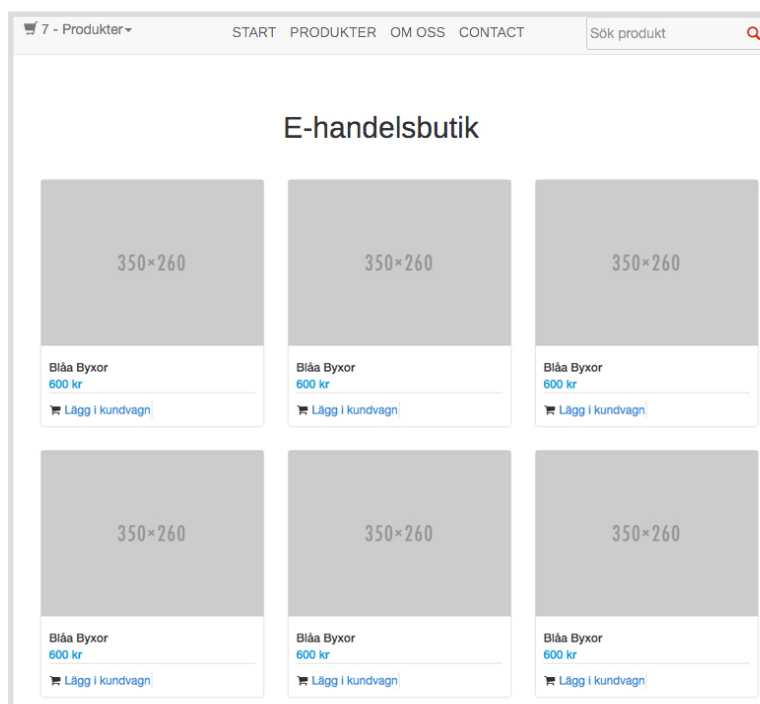
En ytterligare inspirationskälla som har varit till stor hjälp är *Bootsnipp*, som används för att implementera grafiska gränssnittet för e-handelsbutikerna. Hemsidan har varit väldigt användbar och ligger till grund för valet av design för alla tre e-handelsbutikerna.

Även en artikel av *Gizas, et al., (2012)* har varit en inspirationskälla där de visar i en studie på hur större mängd rader kod påverkade svarstider med olika ramverk. Denna artikeln har givit idéer kring hur mätningen skall gå till.

4.2. Progression

Ett gränssnitt skapades i *Adobe Photoshop (figur 8)* för de tre olika ramverken. Gränssnittet kommer vara identisk för varje ramverk där *AngularJS* kommer vara grunden för vår webbapplikation. Prototypen speglar en verklig e-handelsbutik som innehåller en kundvagn, meny, sökfunktion samt produkter som är lagrade i en databas. De funktioner som kommer vara tillgängliga på hemsidorna är sökfunktionen, där användaren kan söka på olika produkter och startsidan för att visa alla produkter. Det var tänkt från början att sökfunktionen skulle skapas med *AngularJS* men eftersom deras sökfunktion är data-

bindning skulle det bli svårt att mäta svarstider genom den typen av sökfunktion. Att använda sig av data-bindning innebär att data genereras direkt i webbläsaren, vilket betyder om en användare söker på en produkt, skulle data hämtas från databasen så fort användaren anger en bokstav i sökfunktionen. Se figur 8 för prototyp av gränssnitt.



Figur 8 Prototyp av gränssnitt

4.3. Implementation av e-handelsbutik

Denna del beskriver hur implementationen utfördes på de olika e-handelsbutikerna, men dock kommer det endast att beskriva hur AngularJS implementerades med Twitter Bootstrap.

Liknande tekniker användes för AngularJS med Foundation och AngularJS med handskrivna stilmallen, men det som skiljer sig åt är sökvägarna till stilmallarna och JavaScript filerna samt klassernas funktionalitet.

4.3.1. Mysql

Det första steget som utfördes var att skapa en enkel databas som innehöll en tabell för produkter. Tabellen döptes till varor och innehöll fyra element, ett id nummer, namnet på varorna, pris för varje vara samt bild för enskild produkt. Sedan skapades 54 olika produkter i databasen. Se figur 9 för databas implementation.

```
CREATE TABLE `varor` (
  `id` int NOT NULL AUTO_INCREMENT,
  `namn` varchar(64) NOT NULL,
  `pris` int NOT NULL,
  `img` varchar(64) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_swedish_ci;

INSERT INTO `varor` (`namn`, `pris`, `img`) VALUES('Röda byxor', 600, 'rodbyx.png');
INSERT INTO `varor` (`namn`, `pris`, `img`) VALUES('Blåa byxor', 600, 'blabyx.png');
INSERT INTO `varor` (`namn`, `pris`, `img`) VALUES('Svarta byxor', 600, 'svartbyx.png');
INSERT INTO `varor` (`namn`, `pris`, `img`) VALUES('Gröna byxor', 600, 'gronbyx.png');
```

Figur 9 Databas implementation och dess struktur

4.3.2. PHP Hypertext Preprocessor

För att generera data från databasen skapades en PHP fil för att koppla databasen till vår e-handelsbutik samt de funktioner som skulle användas för att hämta ut data via sökfunktionen på respektive webbsida. Se figur 10 för databaskoppling.

```
$conn = new mysqli("localhost:8889", "root", "root", "xjobb");
```

Figur 10 Databaskoppling

Efter att ha skapat en koppling från databasen till e-handelsbutikerna behövdes det en funktion som omvandlar resultatet från databasen till JSON data (figur 11) för att det skall kunna gå att parse ut data med hjälp av ramverket AngularJS. JSON (JavaScript Object Notation) är ett format som används när data ska skickas från en server till webbläsaren på ett strukturerat sätt.

```
$output = "";  
while($result = $rs->fetch_array(MYSQLI_ASSOC))  
{  
  if($output != "")  
  {  
    $output .= ",";  
  }  
  
  $output .= '{"id":' . $result["id"] . ',';  
  $output .= "namn":"' . $result["namn"] . ',';  
  $output .= "pris":"' . $result["pris"] . ',';  
  $output .= "img":"' . $result["img"] . '"}';  
}  
  
$output = '{"records":[' . $output . ']'}';
```

Figur 11 While loop som omvandlar resultatet till JSON data

För att det ska vara möjligt för AngularJS att börja parse behövs det en output funktion som skriver ut JSON data. Se figur 12 för output funktion.

```
echo($output);
```

Figur 12 Output funktion som skriver ut JSON data

Som det har nämnts tidigare sker mätningar efter sökta produkter och därför behövs det en funktion som hämtar produkter från databasen. För att funktionen skall vara genomförbar, skapades en query (figur 13) som kollar om användaren har sökt på en specifik produkt eller en sträng. För att hämta ut ett värde (figur 14) som användaren söker behöver det implementeras en _GET funktion som hämtar ett värde från sökfunktionen.

```
Srs = $conn->query("SELECT id, namn, pris, img FROM varor WHERE namn LIKE '%$search%' OR pris LIKE '%$search%'");
```

Figur 13 Sökfunktionen tillåter användare att söka på produktnamn och pris

```
$search = utf8_decode($_GET["search"]);
```

Figur 14 Hämtar ett värde som användaren har sökt på

4.3.3. AngularJs med Twitter Bootstrap

För att påbörja implementationen med ramverket AngularJS och Twitter Bootstrap behöver det finnas JavaScript filer på servern. Dessa filer kan de hämta funktioner som används i själva koden. Figur 15 visar vilka filer som hämtas från servern för att kunna användas på e-handelsbutiken.

```
<script src="js/jquery.min.js"></script>
<!-- Include all compiled plugins (below), or include individual files as needed -->
<script src="js/bootstrap.min.js"></script>
<script src="js/bootstrap.js"></script>
<!-- Include AngularJS -->
<script src="js/angular.min.js"></script>
```

Figur 15 JavaScript filer för att skapa funktioner till e-handelsbutiken

Nästa steg var att skapa en applikation som döptes till xjobbApp med AngularJS. Denna funktion kommer att användas i HTML koden för att kunna skriva ut data från databasen. Sedan skapades en controller som döptes till varorCtrl. Denna funktion kontrollerar data samt att den kontrollerar att rätt data hämtas beroende på vilka attribut man deklarerar i HTML koden. För att en controller skall fungera behövs det en \$scope, vilket är ett objekt som kopplar ihop controller med HTML. Sedan skapades en \$http.get funktion som gör det möjligt att skicka en förfrågan till databasen. I figur 17 har det skapats en metod, \$scope.names som senare kommer att användas i HTML koden för att hämta ut data från databasen.

Applikationen implementerades ihop med sökfunktionen (figur 16) samt mätningsklockan för att kunna få ut de svarstider som efterfrågades i arbetet. I figur 17 ser vi att starttid och sluttid har deklarerats där den först gör en mätning på att läsa in alla produkter på startsidan. Sedan tar den en ny tid efter att användaren har sökt på en viss produkt och därefter subtraherar den sluttiden med starttiden.

```
<script type="text/javascript">
  window.search_value = '<?=$_GET['search']?>';
</script>
```

Figur 16 Sökfunktion

```
<script type="text/javascript">
// deklarerar starttid och sluttid
var startTid = Date.now();
var slutTid;

var app = angular.module('xjobbApp', []);
app.controller('varorCtrl', function($scope, $http)
{
  if(search_value)
  {
    $http.get("http://localhost:8888/boot/db.php?search=" + search_value).then(function(response) {
      $scope.names = response.data.records;

      slutTid = Date.now();
      console.log("Mätningen för söket tog %d ms", slutTid - startTid);
    });
  }
  else
  {
    $http.get("http://localhost:8888/boot/db.php").then(function(response) {
      $scope.names = response.data.records;

      slutTid = Date.now();
      console.log("Mätningen för att ladda in samtliga produkter tog %d ms", slutTid - startTid);
    });
  }
});
</script>
```

Figur 17 AngularJS applikation, mätningsfunktion och sökfunktion

Efter att ha skapat en applikation samt funktioner, påbörjades implementationen av strukturen på e-handelsbutiken. Första steget var att implementera olika stilmallar från Twitter Bootstrap. Se figur 18 för användning av stilmallar från Bootstrap.

```
<link href="font-awesome-4.5.0/css/font-awesome.min.css" rel="stylesheet"
integrity="sha384-XdYbMnZ/QjLh6il4ogqCTAJrFk87ip+ekljefZch0Y+PvJ8CDYtEs1ipDmPorQ+"
crossorigin="anonymous">

<!-- Bootstrap -->
<link href="css/bootstrap.css" rel="stylesheet">
```

Figur 18 Stilmall från Bootstrap och font-awesome

Här används det två stilmallar där den översta ser till att hämta ikoner från den angivna sökvägen. Dessa ikoner används genom att ange HTML koden `text här`. Den understa sökvägen, bootstrap.css är själva stilmallen för hela e-handelsbutiken. Den innehåller en färdig responsiv design och hämtar endast de funktioner som används i HTML koden.

För att få ut data från databasen (figur 19) så anges AngularJS applikationen i en Div-tag tillsammans med controller. Eftersom controller redan vet vilken databas som används behövs det endast anges de attribut som finns i databasen.

Genom att använda koden `ng-repeat="x in names">` tillsammans med `"col-sm-3"` kommer en tabell att skapas där den visar tre produkter på varje rad. Denna tabell kommer att repetera sig tills den hämtat all data från databasen. För att tala om för AngularJS vilken typ av data den skall hämta skall det anges x. innan varje attribut som finns i databasen, dvs `{{x.img}}`, `{{x.namn}}` och `{{x.pris}}`. Koden i figur 19 kommer att parse ut all data som finns i databasen och det som kommer att skrivas ut är namnen på produkterna, priset samt produktbilderna.

```
<div ng-app="xjobbApp" ng-controller="varorCtrl">
  <div class="item active">
    <div class="row">
      <div class="col-sm-3" ng-repeat="x in names">
        <div class="col-item">
          <div class="photo">
            
          </div>
          <div class="info">
            <div class="row">
              <div class="price col-md-6">
                <h5>{{ x.namn }}</h5>
                <h5 class="price-text-color">{{ x.pris }} SEK</h5>
              </div>
            </div>
            <div class="separator clear-left">
              <p class="btn-knapp">
                <a href="#" class="btn btn-sm btn-success"><span class="glyphicon glyphicon-shopping-cart">
                </span> Lägg i kundvagn</a>
              </p>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Figur 19 AngularJS applikation i HTML kod

4.4. Pilotstudie

En pilotstudie har utförts på tre olika webbläsare för att undersöka om e-handelsbutikerna kan utföra de mätningar som är väsentlig för detta arbete. Det kontrollerades även om det gick att mäta på de tilltänkta webbläsarna, vilket är Chrome, Firefox och Safari. Pilotstudien utfördes genom att söka på en specifik produkt 10 gånger per webbläsare med varje e-handelsbutik och sedan redogöra alla mätningar i en graf.

Testfallen som utfördes visade ett fåtal spikar. Dessa spikar kan ha berott på hårdvarans specifikation och kommer därför att inte visas i nedan nämnda grafer. Det kommer att visa de sammanställda medelvärden för varje e-handelsbutik och även vilka webbläsare e-handelsbutiken testades på. Pilottestet har konstaterat att det går att utföra mätningar av svarstider för e-handelsbutikerna vilket var pilottestets ändamål. Svarstiderna är beräknade i ms (millisekunder).

4.4.1. Hårdvara

Mätningarna har utförts på en MacBook Pro (figur 20) med operativsystemet OS X El Capitan som är Apple's senaste operativsystem.

MackBook Pro	OS	Processor	Minne	Grafik
15 tum, sent 2011	El Capitan Version 10.11.3	2,2 GHz Intel core i7	4 GB 1333 MHz DDR3	AMD Radeon HD 6750M 512 MB

Figur 20 Hårdvarans specifikation

4.4.2. Mjukvara

Chrome	Firefox	Safari
Version 50.0.2661.86	Version 45.0.1	Version 9.0.3 (11601.4.4)

Figur 21 Mjukvarans specifikation

4.4.3. Ramverk

AngularJS	Twitter Bootstrap	Foundation
Version 1.5.3	Version 3.3.6	Version 6.2.1

Figur 22 Ramverk som används i detta arbete

4.4.4. AngularJs med olika stilmallar

Det första pilottestet utfördes på en e-handelsbutik som är utvecklad med AngularJS samt Twitter Bootstrap (figur 23) där det visade sig att Firefox hade snabbare svarstider än Chrome och Safari. Mätningarna baserar endast på en specifik produkt och därför kan resultat inte uteslutas än, då det är tänkt att utföra tester där det skall sökas på flera produkter samtidigt.

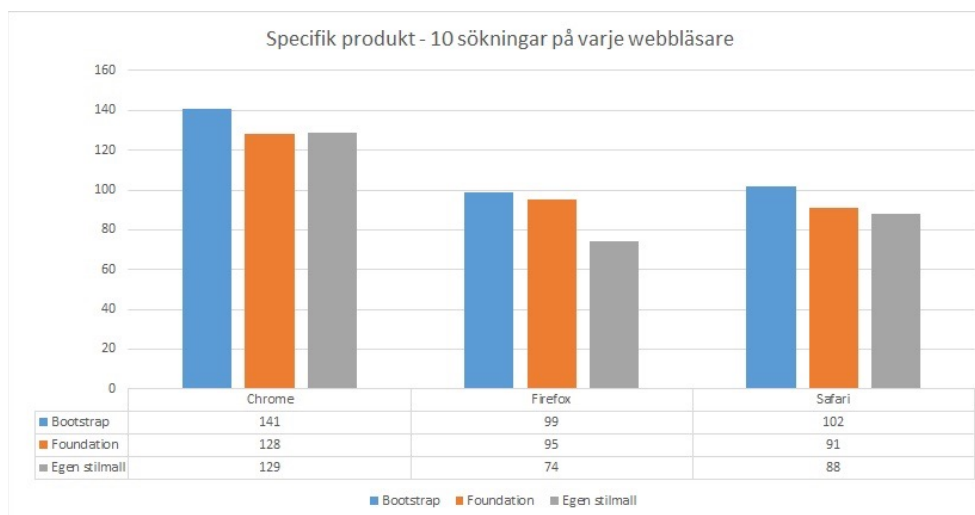
Databasen kommer att utökas med flera hundra produkter för att undersöka om datamängden kan påverka svarstiderna från databasen. Utöver kommer det även att ske mätningar av svarstider, där e-handelsbutikerna renderar ett tiotal produkter under samma sökning.

Det andra pilottestet utfördes på e-handelsbutiken som är skapad med AngularJS och Foundation där det visade sig att Safari hade snabbaste svarstider (figur 23). En jämförelse med föregående graf visar att AngularJS och Foundation har snabbare svarstider på alla tre webbläsare men dock fortsätter Chrome att ha högst svarstid.

Det tredje pilottestet utfördes på en e-handelsbutik som är utvecklad med AngularJs och en handskrivna stilmall. Mätningen visade att Firefox hade snabbast svarstid (figur 23) och även i denna mätning fick Chrome högst svarstid och vad det kan bero på kan vi inte svara på.

4.4.5. Sammanställning av pilottest

En övergripande graf visar resultaten för alla e-handelsbutikerna samt de webbläsare som mätningarna utfördes på. Grafen visar (figur 23) att Firefox och Safari hade väldigt jämna svarstider medan Chrome var den webbläsare som hade högst svarstider när mätningarna utfördes på alla e-handelsbutikerna.



Figur 23 Sammanställning av pilottest

5. Utvärdering

I detta kapitel kommer resultaten från mätningarna att presenteras och analys från de genomförda mätningarna. Det väsentliga utvärderingen i detta arbete är att fokusera på svarstider mellan olika ramverk samt handskrivna stillmall med användning av AngularJs.

Experimentet utförs i två separata mätningar för att försöka fastställa om hypotesen stämmer eller inte. De första mätningen kommer ske ett med grafiskt innehåll, där varje produkt har en pixelstorlek på 512x512 och en filstorlek mellan 600-1100 kilobyte för enskild produktbild. Filstorleken varierar beroende antalet pixlar i varje bild, därför går det inte att fastställa det exakta storleken för varje produktbild. De andra mätningen kommer ske med ett grafiskt innehåll, där varje produkt har en pixelstorlek på 256x256 och en filstorlek mellan 50-110 kilobyte.

- Pixelstorlek på 256x256
 - Filstorlek från 50-110 Kilobyte för enskild produktbild
- Pixelstorlek på 512x512
 - Filstorlek från 600-1100 Kilobyte för enskild produktbild

Mätningar av varje e-handelsbutik kommer att utföras i fyra olika nivåer där varje nivå ökar databasen med produkter. De tre olika e-handelsbutikerna kommer att mätas på lika villkor, dvs om en mätning sker på nivå 2 där databasen innehåller mellan 200-300 produkter kommer alla tre e-handelsbutikerna att mätas med exakt lika många produkter. Databasen kommer att vara fördefinierad inför varje mätning och kommer inte att ändras i efterhand. Varje sökning kommer att ske 10 gånger för att resultatet av svarstiderna skall kunna vara pålitliga för analysen.

- Nivå 1. 50 produkter.
- Nivå 2. 200-300 produkter.
- Nivå 3. 400-600 produkter.
- Nivå 4. 800-1200 produkter.

5.1. Presentation av undersökning

En mer genomgripande mätning på e-handelsbutikerna med en resolut hårdvara samt plattform så kunde data analyseras för att skapa två olika diagram med standardavvikelser för att ge svar på arbetets syfte. Varje mätning kommer att visa två olika diagram, ett sammanställt diagram som visar svarstider för varje sökning i de respektive webbläsare samt ett diagram som visar medelvärdet av svarstiderna för varje webbläsare.

5.1.1. Utrustning

Alla mätningar genomfördes på följande plattform:

Hårdvara		Mjukvara	
Macbook Pro	15 tum, sent 2011	Chrome	Version 50.0.2661.86
Operativsystem	El Capitan Version 10.11.3	Firefox	Version 45.0.1
Processor	2,2 GHz Intel core i7	Safari	Version 9.0.3 (11601.4.4)
Grafik	AMD Radeon HD 6750M 512 MB		
Minne	4 GB 1333 MHz DDR3		

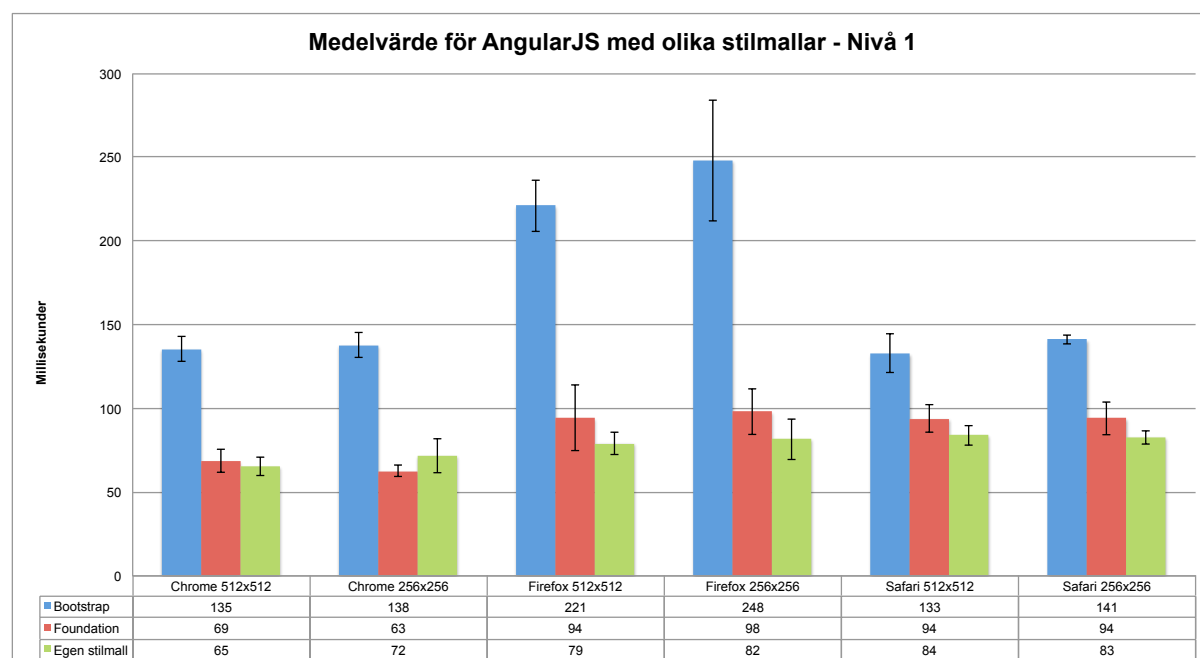
5.2. Analys

5.2.1. Mätning 1: AngularJS med olika stilmallar - nivå 1

Första mätningen använder sig av samma teckning som i pilotstudien. Varje e-handelsbutik körs på tre webbläsare med två olika körningar. Den första körningen mäter svarstider av grafiskt innehåller på 50 produkter, där varje produkt har en storlek på 50-110 Kb och en bildpunkt på 256x256 pixlar.

Den andra körningen mäter svarstider av grafiskt innehåller på 50 produkter, där varje produkt har en storlek på 600-1100 Kb och en bildpunkt på 512x512 pixlar.

Figur 24 visar att standardavvikelserna korsar varandra när de e-handelsbutikerna jämförs på de olika webbläsarna. Dock visar grafen att det är en stor skillnad mellan Twitter Bootstrap och de övriga stilmallarna. Den visar även att svarstiderna för Firefox har en högre resultat i jämförelse med de olika webbläsarna där det högsta medelvärdet är på 248 millisekunder.



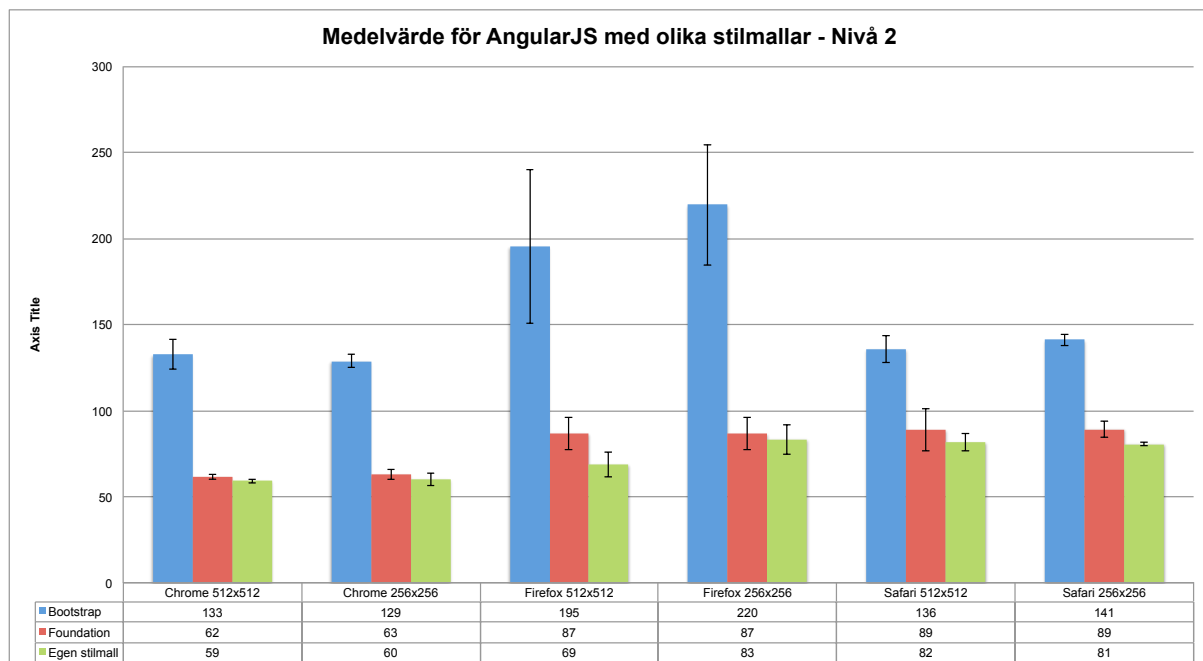
Figur 24 Medelvärde för de tre olika e-handelsbutikerna med användning av AngularJS

5.2.2. Mätning 2: AngularJS med olika stilmallar - nivå 2

Den här mätningen har en utökad databas med 200-300 produkter för att mäta om antalet lagrade produkter påverkar svarstiderna på e-handelsbutikerna. Även denna mätning innehåller två olika körningar för varje webbläsare där den första körningen mäter svarstider av grafiskt innehåller på 200-300 produkter, där varje produkt har en storlek på 50-110 Kb och en bildpunkt på 256x256 pixlar. Den andra körningen mäter svarstider av grafiskt innehåller på 200-300 produkter, där varje produkt har en storlek på 600-1100 Kb och en bildpunkt på 512x512 pixlar.

Diagrammet i figur 25 visar att Bootstrap har högst svarstider oavsett webbläsare men dock har den lägre medelvärde än föregående mätning. Det mest intressanta med denna graf är att den handskrivna stilmallen har ett lägra medelvärde än sina konkurrenter.

Eftersom standardavvikelserna korsar varandra går det inte att utesluta om den handskrivna stilmallen är snabbare eller inte. Det högsta medelvärdet är på 220 millisekunder som tidigare låg på 248 millisekunder, (se figur 24) vilket betyder att antalet produkter mellan nivå 1 och nivå 2 inte påverkar svarstiderna för e-handelsbutikerna.



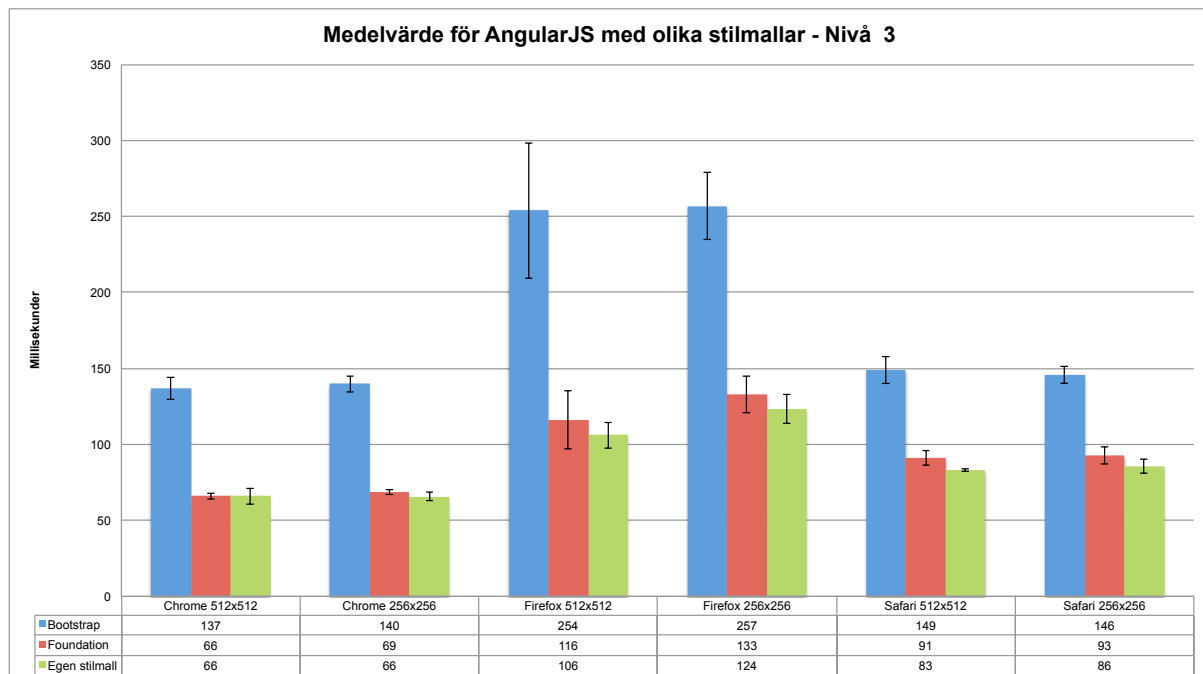
Figur 25 Medelvärde för de tre olika e-handelsbutikerna med användning av AngularJS

5.2.3. Mätning 3: AngularJS med olika stilmallar - nivå 3

Den tredje mätningen har en utökad databas med 400-600 produkter för att mäta om antalet lagrade produkter påverkar svarstiderna på e-handelsbutikerna. Likt de andra, innehåller mätningarna två olika körningar för varje webbläsare.

Den första körningen mäter svarstider av grafiskt innehåller på 400-600 produkter, där varje produkt har en storlek på 50-110 Kb och en bildpunkt på 256x256 pixlar. Den andra körningen mäter svarstider av grafiskt innehåller på 400-600 produkter, där varje produkt har en storlek på 600-1100 Kb och en bildpunkt på 512x512 pixlar.

Figur 26 visar att medelvärdet för varje graf har ökat jämfört med grafen i mätning 2 (se figur 25). Även denna mätning visar att Bootstrap har högst svarstider oavsett webbläsare. Det intressanta med denna mätning är att stilmallen Foundation och den handskrivna stilmallen har ett betydligt högre medelvärden i Firefox än de tidigare mätningar där medelvärdet har varit under 100 millisekunder, men med ett utökad databas är medelvärdet över 110 millisekunder.



Figur 26 Medelvärde för de tre olika e-handelsbutikerna med användning av AngularJS

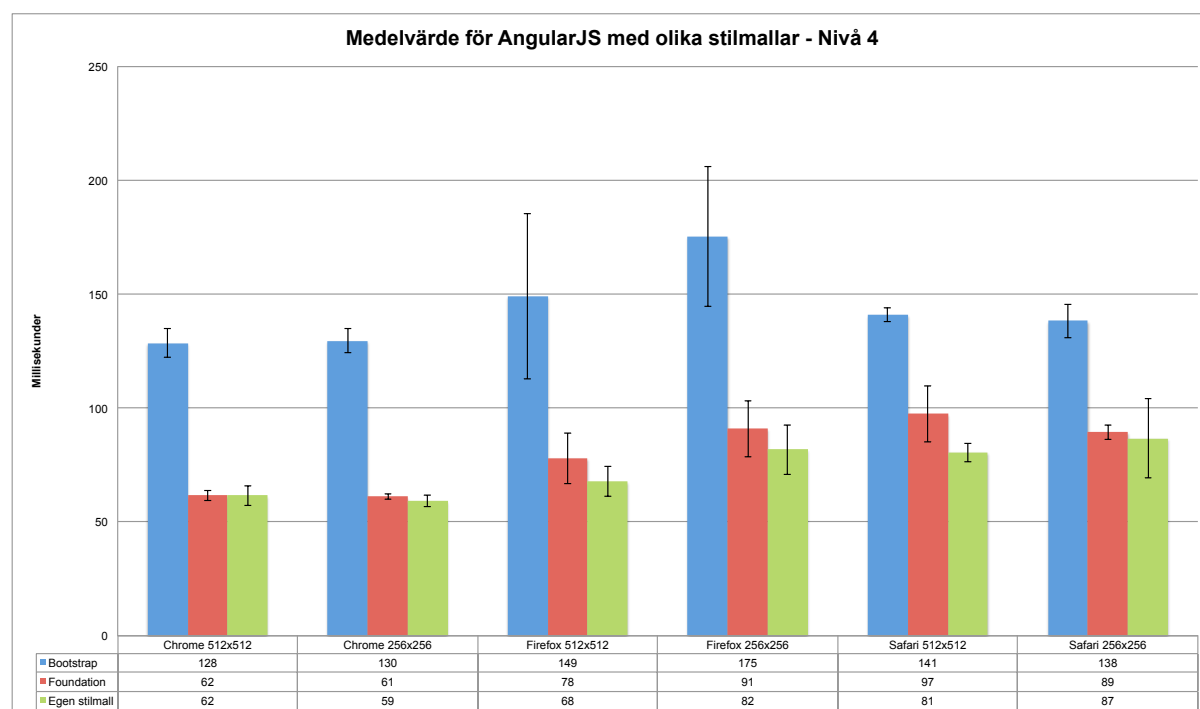
5.2.4. Mätning 4: AngularJS med olika stilmallar - nivå 4

Denna mätningen har en utökad databas med 800-1200 produkter för att mäta om antalet lagrade produkter påverkar svarstiderna på e-handelsbutikerna. Mätningarna innehåller två olika körningar för varje webbläsare.

Den första körningen mäter svarstider av grafiskt innehåller på 800-1200 produkter, där varje produkt har en storlek på 50-110 Kb och en bildpunkt på 256x256 pixlar. Den andra körningen mäter svarstider av grafiskt innehåller på 800-1200 produkter, där varje produkt har en storlek på 600-1100 Kb och en bildpunkt på 512x512 pixlar.

Medelvärdet i figur 27 visar att varje graf har ett lägre värde jämfört med grafen i mätning 3 men dock är Bootstrap de ramverk som har högst medelvärde oavsett webbläsare. Stilmallen Foundation och den handskrivna stilmallen har relativt jämna medelvärden där den handskrivna stilmallen är lite snabbare än konkurrenten.

Eftersom standardavvikelserna korsar vandra går det inte att fastställa om handskrivna stilmallen är snabbare eller inte. Det högsta medelvärdet är på 175 millisekunder som tidigare låg på 257 millisekunder, (se figur 26) vilket betyder att antalet produkter mellan nivå 3 och nivå 4 inte påverkar svarstiderna för e-handelsbutikerna.



Figur 27 Medelvärde för de tre olika e-handelsbutikerna med användning av AngularJS

5.3. Slutsatser

Hypotesen säger att AngularJS med användning av handskrivna stilmall kommer ha lägre svarstider på grafiskt tunga e-handelsbutiker jämfört med responsiva ramverken eftersom större mängd kod kan försämra svarstider på en webbsidan. Med de mätningar som har utförts i detta arbete har det fastställts att större mängd rader kod inte påverkar svarstiderna.

Det är en stor skillnad på Twitter Bootstrap och den handskrivna stilmallen därför går det inte att utesluta något resultat eftersom stilmallen Foundation hade nästan lika låga värden som den handskrivna stilmallen samt att standardavvikelserna korsar varandra i samtliga mätningar. Resultatet av mätningarna visade även att majoriteten av medelvärden för svarstiderna var som högst när e-handelsbutikerna använde sig av produkter där bildpunkterna var 256x256 pixlar samt en storlek på 50-110 Kb.

Utifrån de mätningar som utförts på e-handelsbutikerna kan en övergripande slutsats tas att valet av webbläsare har en viss betydelse när grafiskt tunga webbutiker skall tillämpas med responsiva ramverk. Samtliga mätningar visade att Twitter Bootstrap hade högst svarstider oavsett webbläsare men att svarstiderna var som högst i webbläsaren Firefox. De andra ramverken varierade beroende på vilken webbläsare som användes samt hur många produkter som var lagrade i databasen.

Det som var lite förvånande var att webbläsaren Safari hade höga svarstider när mätningarna gjordes med Twitter Bootstrap. Mätningarna utfördes på en MacBook Pro vilket betyder att webbläsaren Safari är prestandamässigt optimerad för datorn.

E-handelsbutikerna innehöll som mest 1200 produkter och var uppdelad i fyra olika nivåer. Det som skulle vara intressant är att utföra flera mätningar med fler nivåer med ett ökat databas mellan 5000-10000 produkter för att se hur svarstiderna skiljer sig åt jämfört med 500-1000 produkter. I detta arbete gick det inte att öka antalet produkter då e-handelsbutikerna slutade fungera när databasen innehöll mer än 1300 produkter och vad det kan bero på går det inte att svara på.

6. Avslutande diskussion

6.1. Sammanfattning

Arbetet visar hur AngularJS förhåller sig till olika stilmallar när grafiskt tunga webbapplikationer skall tillämpas på webben. Syftet med detta arbetet var att undersöka hur e-handelsbutikernas svarstider påverkades när det användes responsiva ramverk tillsammans med AngularJs jämfört med en handskrivna stilmall tillsammans med AngularJS. Chow (2001) nämner i en artikel att svarstider har en viktig roll när det gäller e-handelsbutiker, detta visades även i en studie där 48% av 12000 besökare avbröt sitt köp på grund av höga svarstider på webbsidan.

Hypotesen är att den handskrivna stilmallen kommer ha lägre svarstid än de responsiva ramverken eftersom större mängd kod kan försämma svarstider på en webbsida. Richard-Foy, et al., (2013) nämner i sin artikel att tunga grafiska webbsidor måste förlita sig på programkoden, vilket kan leda till att högre svarstider.

Undersökningen visar att det finns en skillnad på Twitter Bootstrap och den handskrivna stilmallen där Twitter Bootstrap har högre svarstider oavsett webbläsare. I en artikel av Vernica, et al., (2015) nämns det att mängden kod ökar svarstiderna när responsiv design tillämpas på webbsidorna. Resultatet i detta arbete visar att Foundation och den handskrivna stilmallen har jämlika svarstider men dock kan standardavvikelseerna inte tala om vilken av e-handelsbutikerna som är snabbast.

I en artikel av Gizas, et al., (2012) visade de i en studie på hur större mängd rader kod påverkade svarstider med olika Javascript ramverk. Skillnaden på deras studie och detta arbete var att de testade mätningarna på olika operativsystem samt fler webbläsare och att deras studie baserades endast på Javascript ramverk jämför med detta arbete där vi inkluderade responsiva ramverk med AngularJS. Båda studierna visade att ramverken påverkades till en viss del av webbläsaren och att större mängd rader kod inte påverkade svarstiderna.

6.2. Diskussion

Det går inte att säga att resultatet är statistiskt säkerställt eftersom t-testet varierar beroende på vilken webbläsare som används samt att det endas baserar på 10 st sökningar. Förbättringar som kan utföras är att antalet mätningar utökas i flera nivåer för att få en bättre överblick på hur svarstiderna förändras när datamängden ökas. Fler sökningar bör utföras, dvs att en sökning på produkter sker kanske 30 gånger under en mätning.

Chittaro & Ranon (2002) nämner i en artikel att utmaningen med att skapa en e-handelsbutik är att utforma designen på webbsidan för att på ett effektivt sätt presentera data. En förbättringar som kan utföras är att e-handelsbutikerna utvecklas med flera funktioner som till exempel en kundvagn som fungerar eller produktvisning i flera olika vinklar för att se hur mycket data det krävs innan svarstiderna försämmas.

Ett etiskt problem är att mätningarna på e-handelsbutikerna utfördes endast på tre webbläsare i detta arbete samt ett operativsystem, där resultatet visade att ramverken påverkades beroende på vilken webbläsare som användes, därför kan det vara intressant att se hur e-handelsbutikernas svarstider påverkas när mätningar utförs på flera olika webbläsare samt flera operativsystem.

Samhällsnyttan i detta arbete är att den skall kunna agera som ett verktyg för valet av ramverk. Det nämns tidigt i arbetet att valet av ramverk är en viktig del av utvecklingen när en e-handelsbutik skall tillämpas för webben. En webb utvecklare skall inte välja ett ramverk som endast stävar efter projektets behov. Istället bör valet av ett ramverk eftersträva hög kvalitet hos programkoden samt låga svarstider (Gizas, et al., 2012).

Det bör även diskuteras att valet av ramverk borde varieras beroende på vilken webbapplikation som tas fram. Mätningarna visade att Twitter Bootstrap inte var det snabbaste ramverket när en e-handelsbutik skulle tillämpas jämfört med den handskrivna stilmallen, medan Foundation och den handskrivna stilmallen hade nästan lika låga svarstider.

En annan samhällsnytta kan vara att någon utvecklar ett eget ramverk som efterliknar Twitter Bootstrap eller Foundation men istället använder sig av stilmallar som innehåller grundläggande funktioner.

Forskningsetiskt garanteras återupprepning genom att programkoden publiceras i arbetet samt vilka webbläsare som används och vilka versioner som användes under mätningarna. Även datorns hårdvaruspecifikationer kommer att publiceras så att andra personer kan återskapa det vi har gjort i detta arbete (Cai, et al., 1998). All kod kommer att finnas under Appendix så att utvecklare med grundläggande kunskaper inom programmering kan återskapa det som utförts i detta arbete.

6.3. Framtida arbete

Detta arbete har många förutsättningar för framtida arbete eftersom det finns mycket intressanta faktorer som bör lyftas fram men som inte har gjorts i detta arbete. Möjliga faktorer kan till exempel vara att mätningar utförs på datorer med olika operativsystem för att se om datorernas prestanda påverkar svarstider på grafiskt tunga e-handelsbutiker.

Wohlin, et al., (2012) nämner i en artikel mätningar av svarstider bör ske i en offline-miljö, så att kontrollen över experimentet blir större. Det de menar är att mätningarna kan leda till ett vilseledande resultat när e-handelsbutikerna har verkliga besökare under mätningarna. Det skulle vara intressant att mäta e-handelsbutikerna via ett globalt nätverk, dvs att e-handelsbutikerna är publicerade på World Wide Web och sedan jämföra skillnaderna på svarstiderna med ett lokalt nätverk.

Arbetet har fokuserat på AngularJS, vilket ledde till att andra Javascript ramverk inte tagits upp i detta arbete. Det som skulle kunna utföras i framtida arbete är att ställa olika Javascript ramverk med responsiva ramverk mot varandra för att se hur de olika ramverken förhåller sig till stilmallar som innehåller större mängd rader kod.

Ett intressant fortsättning på forskningsarbete skulle vara att utveckla ett nytt Javascript ramverk med responsiv design anpassad för större webbapplikationer där ramverket lägger till funktioner i stilmallarna beroende på vilken kod som används i webbapplikationen. Det skulle då vara intressant att mäta svarstider på ett sådan ramverk för att sedan göra en jämförelse med de ramverk som finns idag.

Andra intressanta fortsättningar skulle vara att e-handelsbutikerna tillämpas på ett projekt för en verksamhet där utvecklingen sker efter företagets behov. Eftersom detta arbete ger ett någorlunda svar på vilka ramverk som är tillämpad för tungt grafiskt innehåll kan företag dra nytta av projektets analys och resultat.

Referenser

Balasubramanee, V., Wimalasena, C., Singh, R. & Pierce, M. (2013). Twitter Bootstrap and AngularJS. Frontend Frameworks to expedite Science Gateway development. *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, ss. 23–37.

Bootsnipp (2016-03-13). *Bootsnipp*. Tillgänglig: <http://bootsnipp.com/> [2016-04-05]

Cai, J-Y., Nerurkar, A. & Wu, M-Y. (1998) Making benchmarks uncheatable. *Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International*, ss, 216 - 226.

Chittaro, L. & Ranon, R. (2002). New directions for the design of virtual reality interfaces to e-commerce sites. *Proceeding AVI '02 Proceedings of the Working Conference on Advanced Visual Interfaces*, ss. 308-315.

Chow, D. (2001). The effects of time delay in electronic commerce. *Proceeding CHI EA '01 CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ss. 387-388.

Gizas, A., Christodoulou, S. & Papatheodorou, T. (2012). Comparative evaluation of javascript frameworks. *WWW '12 Companion Proceedings of the 21st International Conference on World Wide Web*, ss. 513-514.

Google (2016-03-30). *Angular Material*. Tillgänglig: <https://material.angularjs.org/latest/> [2016-04-02]

Green, B. & Seshadri, S. (2014). *AngularJS: Up and Running. Enhanced Productivity with Structured Web Apps*. 1. uppl. Publisher: O'Reilly Media. Tillgänglig: <https://library.oreilly.com/book/0636920033486/angularjs-up-and-running/toc> [2016-04-02]

Horek, K. (2014). *Learning Zurb Foundation*. 1. uppl. Publisher: Packt Publishing. Tillgänglig: <https://library.oreilly.com/book/9781782164265/learning-zurb-foundation/toc> [2016-04-05]

Kucukcay, I.E & Benyoucef, M. (2014). Mobile Social Commerce Implementation. *MEDES '14 Proceedings of the 6th International Conference on Management of Emergent Digital EcoSystems*, ss. 1-8 .

Pop, D.P. & Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*, vol. 69, ss. 1172–1179.

Richard-Foy, J., Barais, O. & Jézéquel, J-M. (2013). Efficient high-level abstractions for web programming. *GPCE '13 Proceedings of the 12th international conference on Generative programming: concepts & experiences*, ss. 53-60.

Shelly, C.C & Young, G. (2007). Accessibility for simple to moderate-complexity DHTML web sites. *W4A '07 Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, ss. 65-73.

Spurlock, J. (2013). *Bootstrap: Responsive Web Development*. 1. uppl. Publisher: O'Reilly Media. Tillgänglig: <https://library.oreilly.com/book/0636920027867/bootstrap/toc> [2016-04-04]

Twitter Bootstrap (2015-12-08). CSS. Tillgänglig: <http://getbootstrap.com/css/> [2016-04-04]

Vernica, R. & Venkata, N.D. (2015). AERO: An Extensible Framework for Adaptive Web Layout Synthesis. *DocEng '15 Proceedings of the 2015 ACM Symposium on Document Engineering*, ss. 187-190.

Wei, J. & Xu, C-Z. (2010). Measuring Client-Perceived Pageview Response Time of Internet Services. *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, ss. 773–785.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. & Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin Heidelberg: Springer-Verlag. ISBN 978-3-642-29043-5.

ZURB Foundation (2016-04-01). *Foundation for Sites*. Tillgänglig: <http://foundation.zurb.com/sites/docs> [2016-04-05]

A. Appendix - index.php AngularJS & Bootstrap

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="font-awesome-4.5.0/css/font-awesome.min.css" rel="stylesheet" integrity="sha384-XdYbMnZ/QjLh6ii4ogqCTaIjrFk87ip
+ekIjefZchoY+PvJ8CDYtEs1pDmPorQ+" crossorigin="anonymous">

    <link href="css/bootstrap.css" rel="stylesheet">

    <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">

    <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these tags -->

    <title>Bootstrap</title>

    <!-- Bootstrap -->

    <!--[if lt IE 9]>

      <script src="js/html5shiv.min.js"></script>

      <script src="js/respond.min.js"></script>

    <![endif]-->

  </head>

  <body>

    <nav class="navbar navbar-default" role="navigation">

      <div class="navbar-header">

        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

          </button>

        <a class="navbar-brand" href="#">E-handelsbutik</a>

      </div>

      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

        <ul class="nav navbar-nav">
```

```

<li class="active"><a href="index.php">Start</a></li>

<li><a href="#">Om oss</a></li>

<li><a href="#">Produkter</a></li>

<li><a href="#">Kontakt</a></li>

</ul>

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav navbar-right">

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-expanded="false"> <span class="glyphicon glyphicon-shopping-cart"></span> 0 - Produkter<span class="caret"></span></a>

</li>

</ul>

</div>

</div>

</div>

</div>

</nav>

<div class="container">

<div class="row">

<h2></h2>

<div id="custom-search-input">

<form action="index.php" method="get" name="search">

<div class="input-group pull-right col-md-4">

<input type="text" name="search" id="sok" class="search-query form-control" placeholder="Sök produkt" />

<span class="input-group-btn">

<button class="btn btn-danger" type="submit">

<span class="glyphicon glyphicon-search"></span>

</button>

</span>

</div>

</div>

</form>

</div>

</div>

```

```

<div class="container">
<div class="row">
  <div class="row">
    <div class="col-md-9">
      <h3></h3>
    </div>

  </div>

<div ng-app="xjobbApp" ng-controller="varorCtrl">
  <!-- Trigger the modal with a button -->
  <div class="item active">
    <div class="row">
      <div class="col-sm-3" ng-repeat="x in names">
        <div class="col-item">
          <div class="photo">
            </div>
          <div class="info">
            <div class="row">
              <div class="price col-md-6">
                <h5>{{ x.namn }}</h5>
                <h5 class="price-text-color">{{ x.pris }} SEK</h5>
              </div>
            </div>
            <div class="separator clear-left">
              <p class="btn-knapp">
                <a href="#" class="btn btn-sm btn-success"><span class="glyphicon glyphicon-shopping-cart"></span> Lägg i kundvagn</
a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

```

```
<script src="js/jquery.min.js"></script>
```

```
<!-- Include all compiled plugins (below), or include individual files as needed -->
```

```
<script src="js/respond.min.js"></script>
```

```
<script src="js/html5shiv.min.js"></script>
```

```
<!-- Include AngularJS -->
```

```
<script src="js/angular.min.js"></script>
```

```
<script type="text/javascript">
```

```
    window.search_value = '<?=$_GET[search]?>';
```

```
</script>
```

```
<script type="text/javascript">
```

```
    // deklarera starttid och sluttid
```

```
    var startTid = Date.now();
```

```
    var slutTid;
```

```
    var app = angular.module('xjobbApp', []);
```

```
    app.controller('varorCtrl', function($scope, $http)
```

```
    {
```

```
        if(search_value)
```

```
        {
```

```
            $http.get("http://localhost:8888/boot/db.php?search=" + search_value).then(function(response) {
```

```
                $scope.names = response.data.records;
```

```
                slutTid = Date.now();
```

```
                console.log("Mätningen för söknet tog %d ms", slutTid - startTid);
```

```
            });
```

```
        }
```

```
    else
```

```
    {
```

```
        $http.get("http://localhost:8888/boot/db.php").then(function(response) {
```

```
            $scope.names = response.data.records;
```

```
            slutTid = Date.now();
```

```
            console.log("Mätningen för att ladda in samtliga produkter tog %d ms", slutTid - startTid);
```

```
    });  
  }  
});  
</script>  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>  
<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>  
</body>  
</html>
```

B. Appendix - index.php AngularJS & Foundation

```
<!doctype html>
<html class="no-js" lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Foundation</title>
    <link rel="stylesheet" href="css/foundation.css">
    <link rel="stylesheet" href="css/app.css">
    <link rel="stylesheet" href="foundation-icons/foundation-icons.css">

  </head>
  <body>

    <div class="row">
      <div class="small-11 small-centered columns"></div>
      <div class="top-bar">
        <div class="top-bar-title">
          <span data-responsive-toggle="responsive-menu" data-hide-for="medium">
            <span class="menu-icon dark" data-toggle></span>
          </span>
        </div>
        <div id="responsive-menu">
          <div class="top-bar-left">
            <ul class="dropdown menu" data-dropdown-menu>

              <li> <strong>E-handelsbutik</strong>
                <li><a href="index.php">Start</a></li>
                <li><a href="#">Om oss</a></li>
                <li><a href="#">Produkter</a></li>
                <li><a href="#">Kontakt</a></li>
              </ul>
            </div>
            <i class="top-bar-right fi-shopping-cart"> 0 - Produkter</i>
          </div>
        </div>
      </div>
    </div>
```

```

</div>
<h1></h1>

<div class="top-bar-right">
  <div class="row collapse">
    <div class="large-8 columns">
      <form action="index.php" method="get" name="search">
        <input type="text" name="search" class="search-query form-control" placeholder="Sök produkt" />
      </div>
      <div class="small-4 columns">
        <span class="alert button"><i class="fi-magnifying-glass"></i></span>
      </div>
    </div>
  </div>
</form>
</div>

```

```

</div>
<h1></h1>
<div ng-app="xjobbApp" ng-controller="varorCtrl">
  <div class="row">
    <div class="medium-3 columns" ng-repeat="x in names">
      
    </div>
    <div class="row">
      <div class="small-11 small-centered columns">
        <h5>{{ x.namn }}</h5>
        <h5 class="price-text-color">{{ x.pris }} SEK</h5>
      </div>
    </div>
    <a href="#" class="button small success">Lägg i kundvagn</a>
  </div>
</div>
</div>

```

```

<script src="js/angular.min.js"></script>

```

```

<script src="js/jquery.min.js"></script>

<script src="js/vendor/jquery.js"></script>

<script src="js/vendor/foundation.min.js"></script>

<!-- Other JS plugins can be included here -->

<script>

$(document).foundation();

</script>

<script type="text/javascript">

    window.search_value = '<?=$_GET[search]?>';

</script>

<script type="text/javascript">

    // deklarerera starttid och sluttid

    var startTid = Date.now();

    var slutTid;

    var app = angular.module('xjobbApp', []);

    app.controller('varorCtrl', function($scope, $http)

    {

        if(search_value)

        {

            $http.get("http://localhost:8888/found/db.php?search=" + search_value).then(function(response) {

                $scope.names = response.data.records;

                slutTid = Date.now();

                console.log("Mätningen för söket tog %d ms", slutTid - startTid);

            });

        }

        else

        {

            $http.get("http://localhost:8888/found/db.php").then(function(response) {

                $scope.names = response.data.records;

                slutTid = Date.now();

                console.log("Mätningen för att ladda in samtliga produkter tog %d ms", slutTid - startTid);

            });

        }

    }

}

```

```
    }  
  });  
</script>  
</body>  
</html>
```

C. Appendix - index.php AngularJS & Handskriven

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Egen</title>

    <link href="css/style.css" rel="stylesheet">

  </head>
  <body>
    <div>
      <ul>
        <li><a class="active"><strong>E-handelsbutik</strong></a></li>
        <li><a href="index.php">Start</a></li>
        <li><a href="#Om oss">Produkter</a></li>
        <li><a href="#Produkter">Om oss</a></li>
        <li><a href="#Kontakt">Kontakt</a></li>
      <div class="ikon"
        
        <a href="#" class="produker"> o - Produkter</a>
      </ul>
    </div>

    <div class="sok">
      <li id="sok-funktion">
        <form action="index.php" method="get" name="search">
          <input type="text" name="search" id="sok_text" class="search-query form-control" placeholder="Sök produkt" />
          <input type="button" name="sok_knapp" id="sok_knapp"></a>
```

```

    </form>

</li>

</div>

<div ng-app="xjobbApp" ng-controller="varorCtrl">

<div class="rad">

<div class="rad-3 ng-scope" ng-repeat="x in names">

    <div class="rad-produkt">

        <div class="bild">

        </div>

        <div class="info">

            <div class="rad">

                <div class="pirs rad-3">

                    <h5 class="ng-binding">{{ x.namn }}</h5>

                    <h5 class="pris-text ng-binding">{{ x.pris }} SEK</h5>

                </div>

            </div>

            <div class="rad">

                <p class="knapp">

                    <button type="button" class="knapp">Lägg i kundvagn</button>

                </p></div>

            </div>

        </div>

    </div>

</div>

</div>

</div>

</div>

<script src="js/angular.min.js"></script>

<script src="js/jquery.min.js"></script>

<script type="text/javascript">

    window.search_value = '<?=$_GET[search]?>';

</script>

<script type="text/javascript">

    // deklarerera starttid och sluttid

    var startTid = Date.now();

    var slutTid;

```

```
var app = angular.module('xjobbApp', []);

app.controller('varorCtrl', function($scope, $http)
{
  if(search_value)
  {
    $http.get("http://localhost:8888/egen/db.php?search=" + search_value).then(function(response) {
      $scope.names = response.data.records;

      slutTid = Date.now();
      console.log("Mätningen för söket tog %d ms", slutTid - startTid);
    });
  }
  else
  {
    $http.get("http://localhost:8888/egen/db.php").then(function(response) {
      $scope.names = response.data.records;

      slutTid = Date.now();
      console.log("Mätningen för att ladda in samtliga produkter tog %d ms", slutTid - startTid);
    });
  }
});
</script>
</body>
</html>
```

D. Appendix - style.css Handskriven stilmall

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    background-color: #f8f8f8;  
    border: 1px solid #e7e7e7;  
}
```

```
li {  
    float: left;  
    display: inherit;  
}
```

```
li a {  
    display: block;  
    color: #555;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}
```

```
a.produker {  
    float: right;  
    margin-right: -146px;  
    margin-top: 4px;  
    color: #555;  
    text-decoration: none;  
    width: 130px;  
}
```

```
.ikon {  
    float: right;  
    margin-right: 140px;  
    background-image: url("../commerce.png");
```

```

margin-top: 12px;

width: 21px;

height: 21px;

background-size: contain;
}

form {

margin-top: 20px;

margin-right: 80px;
}

li#sok-funktion {

float: right;
}

#sok_text{

width: 297px;

padding: 15px 0 15px 20px;

font-size: 16px;

font-family: Montserrat, sans-serif;

border: 1px solid #ccc;

border-radius: 2px;

height: 34px;

margin-right: 0;

color: #222;

outline: none;

background: #fff;

float: left;

box-sizing: border-box;
}

::-webkit-input-placeholder { /* WebKit browsers */

color: #222;
}

:-moz-placeholder { /* Mozilla Firefox 4 to 18 */

color: #222;
}

```

```

::-moz-placeholder { /* Mozilla Firefox 19+ */
    color: #222;
}

:-ms-input-placeholder { /* Internet Explorer 10+ */
    color: #222;
}

#sok_knapp {
    border: 0 none;
    background: #d9534f url(..../tool.png) center no-repeat;
    width: 60px;
    float: left;
    padding: 0;
    text-align: center;
    height: 34px;
    cursor: pointer;
}

.rad {
    margin-right: -15px;
    margin-left: -15px;
}

.rad-3.ng-scope {margin-left: 100px;margin-top: 80px;}

.pirs.rad-3 {
    font-size: 18px;
    color: #333;
    margin: 17px;
    line-height: oem;
}

button.knapp {
    background-color: #5cb85c;
    color: #fff;
}

```

```
border: 1px solid #e7e7e7;
border-radius: 5px;
height: 30px;
margin-left: 12px;
}
```

```
img.img-responsive {
width: 100%;
}
```

```
.rad-produkt {
float: left;
margin-left: 106px;
margin-right: -117px;
padding: 10px;
height: 230px;
}
```

```
.bild {
width: 128px;
}
```

```
.info {
width: 285px;
}
```

E. Appendix - db.php Databaskoppling

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=ISO-8859-1");

// hämta värdet användaren har sökt på
$search = utf8_decode($_GET["search"]);

// anslut till MySQL databasen
$conn = new mysqli("localhost:8889", "root", "root", "xjobb");

// en query som hämtar produkter från databasen, om användaren har sökt på en sträng, kommer detta även att kolla på.
// det är möjligt att söka på både namn och pris.
$rs = $conn->query("SELECT id, namn, pris, img FROM varor WHERE namn LIKE '%$search%' OR pris LIKE '%$search%'");

// en while loop som omvandlar databasens resultat till JSON data för att det skall kunna gå att parsea den med hjälp av AngularJS
$output = "";
while($result = $rs->fetch_array(MYSQLI_ASSOC))
{
    if($output != "")
    {
        $output .= ",";
    }

    $output .= '{"id":"' . $result["id"] . "'";
    $output .= "namn":"' . $result["namn"] . "'";
    $output .= "pris":"' . $result["pris"] . "'";
    $output .= "img":"' . $result["img"] . "'}';
}
$output = '{"records":[' . $output . ']'}';

// stäng MySQL databasanslutningen
$conn->close();

// skriv ut JSON data för att AngularJS ska börja parsea.
echo($output);
?>
```

F. Appendix - xjobb.sql Databas

```
CREATE TABLE `varor` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `namn` varchar(64) NOT NULL,  
  `pris` int NOT NULL,  
  `img` varchar(64) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_swedish_ci;  
  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röda byxor', 600, 'rodbyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blåa byxor', 600, 'blabyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svarta byxor', 600, 'svartbyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gröna byxor', 600, 'gronbyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gråa byxor', 600, 'grabyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa byxor', 600, 'rosabyx2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röd tröja', 300, 'rodtroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blå tröja', 300, 'blatroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svart tröja', 300, 'svarttroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grön tröja', 300, 'grontroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grå tröja', 300, 'gratroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa tröja', 300, 'rosatroja2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röd jacka', 750, 'rodjacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blå jacka', 750, 'blajacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svart jacka', 750, 'svartjacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grön jacka', 750, 'gronjacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grå jacka', 750, 'grajacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa jacka', 750, 'rosajacka2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röda strumpor', 100, 'rodstrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blåa strumpor', 100, 'blastrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svarta strumpor', 100, 'svartstrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gröna strumpor', 100, 'gronstrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gråa strumpor', 100, 'grastrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa strumpor', 100, 'rosastrump2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röd mössa', 300, 'rodmoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blå mössa', 300, 'blamoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svart mössa', 300, 'svartmoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grön mössa', 300, 'gronmoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grå mössa', 300, 'gramoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa mössa', 300, 'rosamoss2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röda handskar', 300, 'rodhand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blåa handskar', 300, 'blahand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svarta handskar', 300, 'svarthand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gröna handskar', 300, 'gronhand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gråa handskar', 300, 'grahand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa handskar', 300, 'rosahand2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röd halsduk', 50, 'rodhals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blå halsduk', 50, 'blahals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svart halsduk', 50, 'svarthals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grön halsduk', 50, 'gronhals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grå halsduk', 50, 'grahals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa halsduk', 50, 'rosahals2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röda skor', 600, 'rodsko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blåa skor', 600, 'blasko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svarta skor', 600, 'svartsko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gröna skor', 600, 'gronsko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Gråa skor', 600, 'grasko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa skor', 600, 'rosasko2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Röd väska', 750, 'rodvask2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Blå väska', 750, 'blavask2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Svart väska', 750, 'svartvask2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grön väska', 750, 'gronvask2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Grå väska', 750, 'gravask2.jpg');  
INSERT INTO `varor`(`namn`, `pris`, `img`) VALUES('Rosa väska', 750, 'rosavask2.jpg');
```