Examensarbete

HÖGSKOLAN I SKÖVDE

jämföra två ramverks responstid med och utan ORM

SVENSK TITEL

Jämförelse mellan Cakephp och Codeigniter
Ramverk med/utan Model mapping (ORM)

ENGLISH TITLE

Comparison between Cakephp and Codeigniter
Framework with/without Model mapping (ORM)

Examensarbete inom huvudområdet informationsteknologi

Grundnivå 30 Högskolepoäng
Vårtermin 2020

Hiwa Ahmed (a16hiwah)

Handledare: Henrik Gustavsson
Examinator: Yacine Atif

Sammanfattning

Webbapplikation med lång svarstid kan påverka negativ användaren ifall den är mer än tre sekunder, därför svarstid bliv ett intressant område att tänka på vid utveckling av webbapplikationer. Studie kommer undersöka ifall model mapping (ORM) kan påverka svarstid eftersom det finns ett extra lager mellan applikationen och databasen. Är skillnaden mycket för att välja direktkoppling mot databas eller skillnaden liten nog för att använda ORM-verktyg istället för direkt koppling om det underlättar utvecklingen av en webbapplikation. Resultatet av experimentet visade att webbapplikationer utan ORM-verktyg hade lägre responstid vid att testa funktionaliteter som inloggning /utloggning och databasinteraktivitet (CRUD), men den skiljer bara några millisekunder vilken inte värt att vi undvika fördelarna som model mapping (ORM) kan bjuda, bara för att vinna några millisekunder responstid som användaren kan inte ens observera den eller spela någon stor roll under verklighetsmiljö.

Nyckelord: [CakePHP, Codeigniter, model mapping (ORM), responstid

Innehållsförteckning

Innehållsförteckning **0**
1 Introduktion **2**
2 Bakgrund **3**
2.1 Plain PHP **3**
2.2 PHP-ramverk **3**
2.3 CakePHP **4**
2.3.1 Modell i Cakephp **5**
2.3.2 Controller i Cake php **5**
2.3.3 Vy i Cake php **6**
2.4 Codeigniter **6**
2.4.1 Model i Codeigniter **6**
2.4.2 Controller i Codeigniter **6**
2.4.3 Vy i Codeigniter **7**
2.5 Object-Relational Mapping (ORM) **8**
2.6 prestandaoverhead vid användning av ramverk **9**
3 Problemformulering **10**
3.1 Hypotes **10**
3.2 Metodbeskrivning **11**
4 Forskningsetik **13**
5 Genomförande **14**
5.1 Litteraturstudie **14**
5.2 Implementation **15**
5.2.1 webbserver **15**
5.2.2 Databaser **16**
5.2.3 Installera av ramverk **16**
5.2.4 Inledning av webbapplikationens utveckling **17**
5.3 Progression **20**
5.3.1 CakePHP 20
5.3.2 Byta av språk i gränssnitt 20
5.3.3 mätningsmetod 20
5.4 Pilotstudie **24**
5.4.1 Förberedelser inför testning 24
5.4.2 Tillvägagångssätt 24
5.4.3 Insamling av data inför testning 25
5.4.4 Resultat av pilotstudie 25
6 Utvärdering **28**
6.1 Presentation av undersökning **28**
6.2 Analys **28**
6.2.1 CakePHP Testfall 1-inloggning och utloggning 28
6.2.2 CakePHP Testfall 2-databasinteraktivitet (CRUD) 31
6.2.3 Codeigniter Testfall 3-inloggning och utloggning 37
6.2.4 Codeigniter Testfall 4-databasinteraktivitet (CRUD) 40
7 Slutsats **47**
7.1 Resultatsammanfattning **47**
7.2 Diskussion **47**
7.3 Etik och Samhälle **48**
7.4 Framtida arbete **49**
Referenser **50**

1 Introduktion

I analysen från 2000 som gjordes av DiCesare & Hoxmeier (2000)., beskrivs huvudorsaken till att användaren undviker använda webbsidor och applikation på grund av dålig prestanda och lång svarstid. DiCesare & Hoxmeier (2000). nämner att de flesta användaren skulle vara nöjda om svarstiden är mindre än tre sekunder. Enligt Kathas och Lancors (2013) spenderar användaren mycket tid på webbapplikationer, däremot, användaren kommer sluta använda webbapplikation, om applikationen har lång svarstid, därför har svarstid blivit ett intressant område för forskare att undersöka. Enligt (Majida, L. Khaoula, B. Samira, K. Mohamed. K. 2019). php anses vara ett av de mest använda skriptsspråk i webbutveckling, för att den bjuder flexibilitet, lätt att använda och lära sig. Idag php har många ramverken som underlättar utvecklingens process. Användning av ramverk hjälper att implementera komplexa applikationer, eftersom ramverken ger möjlighet att återanvända funktionalitet (Lancor & Katha, 2013; Salas-Zárate et al., 2015). Idag är många ramverk Arkitekturmönstret MVC[1] baserad, (Gorshkova, Novikov, Belov, Gurov & Spiridonov, 2005; Lancor & Katha, 2013). MVC arkitekturmönstret delar webbapplikationen i tre delar, vy renderar modell till en form som är lämpligt för interaktion, normalt ett användargränssnitt, Controller tar emot inmatning och konverterar den till kommandon för modellen eller vyn. Modell är applikationens dynamiska datastruktur, (Biermann, 2014; Lancor & Katha, 2013). Två MVC-baserad php ramverk är Cakephp och Codeigniter, båda ramverken är öppen källkod och har stöd för medel mapping (ORM). ORM är object relation mapping är en mjukvara som hjälper objektapplikation att koppla sig mot en relational databas.

Kathas och Lancors (2013) gjorde en studie för att analysera och jämföra mellan php utan ramverk, Cakephp och Codeigniter. Resultatet visade att plain php[2] ger bättre prestanda än php ramar. Anledning till att båda ramverken tog längre tid än plain php var att båda ramverken använde sig av ORM vid koppling mot databas. Frågan är om man kan få fördelarna av plain php genom att direkt koppla php ramverk mot databas utan ORM, och analysera om ORM påverkar svarstiden negativt. För att kunna besvara frågeställningen kommer en kvantitativ metod användas, metoden som ska användas är experiment som hjälper att få svar ifall ORM kommer påverka responstiden.

För att utföra experimentet kommer vi ha två webbapplikationer, en med/utan ORM för Cakephp och den en med/utan ORM för Codeigniter. Hela utvecklingsprocess från grund kommer publicera på github utom känsliga data användas.

[1] MVC (Model-View-Controller): arkitekturmönster som används inom systemutveckling
[2] Plain php är php utan ramverk

2 Bakgrund

Internet idag innehåller webbapplikationer som användas för att underlätta elektroniska transaktioner som social kommunikation, e-handling, e-post, verktygsräkningar och bank för användare, (Ushawu Ibrahim, J. B.Hayfron, Frimpong, T. 2018). I samma artikel visade (Ushawu Ibrahim, J. B.Hayfron, Frimpong, T. 2018), att det finns olika program och skriptspråk som asp.net, ruby on rails och plain php, som hjälper till att skapa dessa applikationer. Enligt James, M. (2010) användaren undviker webbsidor som tar mer än tre sekunder för att hämta webbsidas innehåll. James, M. (2010) visar på hur användaren blir nöjd med webbsidor som klarar att ladda inom två i vissa fall tre sekunder.

2.1 Plain PHP

PHP är programmeringsspråk som anses vara flexibilitet, lätt att använda och lära inom webbutveckling (Majida, L. Khaoula, B. Samira, K. Mohamed. K. 2019). Ökad efterfråga av webutveckling har lett till en ökad efterfrågan av effektivitet, pålitlighet, underhållsbarhet och skalbarhet. Huvudmål med php är att den gör lättar att återge dynamiskt innehåll på webbsidor, lätt att installation och integration php i alla operativsystem. Enligt Kathas och Lancors (2013) applikationer som skapad i php utan ramverk har bättre svarstid, eftersom php kommer inte med extra inbyggde metoder och funktioner som påverkar svarstid negativt. Php kan ansluta och manipulera många databaser, inklusive de kommersiella och öppen källkod. En av databas som är lämpligt för php är relationsdatabassystem som MySQL, PostgreSQL och Oracle.

För att kunna spara data på ett organiserad och strukturerad sätt måste finnas en databas som tar emot dess data (Abramova och Bernardino, 2013). Idag finns det olika sätt att koppla applikationer mot databas, exempelvis relationsdatabaser, object-orienteade databaser och object-relational mapping (ORM). Alla försöker interagera med varandra på något sätt (Ghosh and Rau-Chaplin, 2006).

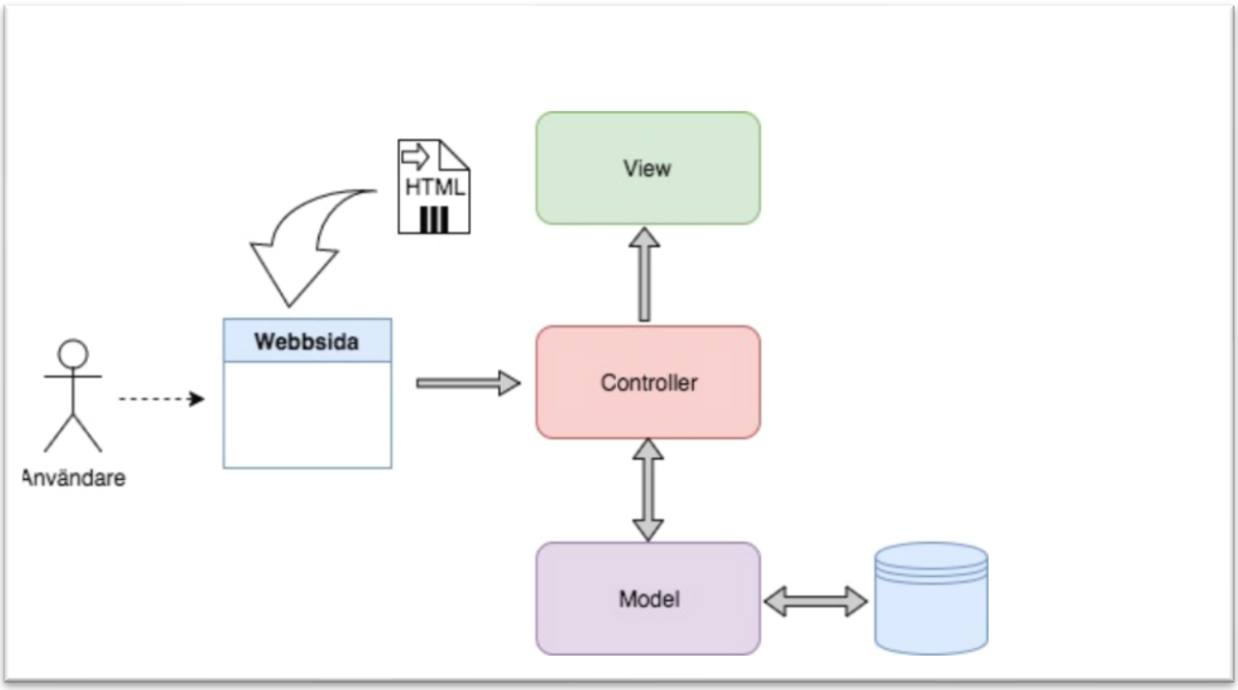
Användning av ramverk under utveckling av en applikation eller webbsida är avsedda att underlätta programvaruutveckling, påskynda utvecklingsprocessen, förkorta utvecklingstiden och ge underhållbars kod. De slutliga webbapplikationerna som skapas med hjälp av PHP-ramverk kommer att vara mer stabila och säkra (Majida Laaziriet al. 2019). Det finns många PHP-ramverk idag, var och en av dessa ramverk har sina egna fördelar jämfört med andra, men att välja det bästa PHP-ramverket vid utveckling kan vara svårt, vilket kräver en god förståelse för de olika PHP-ramverken (Alor-Hernández et al. 2015). Många PHP-ramverk har valts av utvecklarna, som Smfony, Codelgniter, Larave, Cake PHP, Yii, etc., och dessa har många fördelar, men att välja den bästa PHP-ramverk är fortfarande ett problem, och utvecklarna måste ha mer koll när de väljer det mest lämpliga ramverk med de bästa supportfunktionerna att använda i sitt projekt (Majida Laaziriet al. 2019).

PHP ramverk är ett väsentligt verktyg för webbutveckling, eftersom ett ramverk erbjuder intuitiva funktioner, öppen källa, cross-platform kompatibilitet och SQL-stöd, Majida, L. Khaoula, B. Samira, K. Mohamed. K. (2019).

2.2 PHP-ramverk

Natalya, P. (2017) har definierad ett ramverk underlättar webbprogrammering och gör det bättre organiserat på många sätt. I artikeln nämns också att ett ramverk ökar

programmerarens produktivitet, eftersom att skriva hundratals kodrader kan ta några timmar, om den skulle jämföras men inbyggda ramverksfunktioner, kan detta göras på några minuter. Enligt M. Fowler (2002) är ett sådant php-ramverk ett implementeringslager på PHP vilket innehåller arkitektoniska mönster, principer och andra implementationer som löser olika problem. Ett exempel på ett arkitekturmönster kan vara Model-View-Controller (MVC). MVC är ett mjukvaruarkitektoniskt mönster för implementering av användargränssnitt på datorer. Den delar upp en given applikation i tre sammankopplade delar. Modell används endast för databasinteraktioner. View används endast för att rendera ut data till webbläsaren. Controller används för alla kommandon, skript för input och programflödet M. Fowler (2002).



Figur 1 hur MVC fungerar

I figur 1 visar på vilken sätt MVC fungerar när användaren besöker webbapplikation MVC baserad. När användare öppnar webbapplikation för att göra en viss val exempelvis överföra pengar mellan två bankkonton via webbapplikationen går en begäran till Controller som i sin tur startar webbapplikationen, sedan Controller startar en kommunikation med Modellen för att hämta rätt information för båda bankkonton från databas. efter att modellen har hämtad information som behövs, skickas tillbaka till Controller. Vyn tar emot data från Controller och visar till användaren i HTML format.

2.3 CakePHP

Cakephp är en öppen källkod. Cakephp skriven i php men modellerade efter Ruby on Rails Natalya, P. & Victoria, B. (2017). Cakephp är ett MVC-baserat och har stöd för vanliga standardfunktioner som formulärhantering, formulärvalidering, sessionshantering, autentisering och databasinteraktion (Lancor & Katha, 2013). När det gäller kommunikation med databas, Cakephp har ett inbyggt stöd mjukvara kallar ORM, som gör databasrelationer enklare, (Kathas och Lancors, 2013).

2.3.1 Modell i Cakephp

Modell i Cakephp representerar vanligtvis en databastabell, men den kan användas för att nå andra typer av data. varje databastabell skall ha en modell som representerar den. Varje databastabell i Cakephp har sin egen modell. Modellen innehåller också kod som definierar dess relation med andra modeller. Modell klass filen i Cakephp kommunicerar med data genom databasen med två primära objekttyper delas mellan repositories/tabeller och entity objekt. Repositories visar metoder och att hämta data från databasen. Tabell objekt ger tillgång till samling av data som lagrad i en specifik tabell, medan tabellobjekt representerar och ger tillgång till en samling objekt. Entities representerar individuella rader eller domänobjekt i applikation. Entities representerar resultatet från en rad i tabellen, och innehåller metoder för att manipulera och få åtkomst till den information tabellen innehåller, (Biermann, M., (2014).

```
1 <?php
2 class Todo_model extends CI_Model {
3  public $name = ' Todo_model ';
4 }
5 ?>
```

Figur 2 Model i Cake php

Figur 2 visar kodexempel modell i Cakephp. Rad 2 är applikationers modell klass kallas **`Todo_model`**, samma namn som tabellen ska hämta data från och sedan läggas i en fil med namn **`Todo_model.php`** ligger i app/Modell mappen, på det sättet vet vi att klassen kommer automatisk att använda **`Todo_model`** tabellen.

2.3.2 Controller i Cake php

En controller applikationsflödet eller logiken för applikationen. Den skapar kommunikation mellan vy och model. Controller verifierar om korrekta modell anropades och rätt svar visas i vy vid begäran av data.

```
1 <?php
2   namespace App\Controller;
3   use App\Controller\AppController;
4   class TodosController extends AppController {
5   public function index ()
6   $todos = $this->Todo_model->find ();
7   $this->layout = 'todos';
8   $this->set (compact ('todos'));}
```

Figur 3 Controller i Cake php

Figur 3 visar ett kodexempel för controller i Cake php. klassen App\Controller är parent klass för hela applikationen. Metoden find (); användas i controller för inläsning av data från databasen, med den här kod raden väljer vi alla kategorier från tabellen och tilldelar den resulterande matrisen för att efteråt skicka den vidare till variabel $todos och visa den till

användaren vid anrop. Cakephp hämtar modellen som matchar Controller automatiskt, behöver ej utförda med händerna, (Biermann, M. (2014).

2.3.3 Vy i Cake php

Vy i Cakephp kan byggas på olika sätt. Utvecklaren kan utnyttja av metoder och funktionalitet för att bygga till exempel log in form, eller med bara HTML och plain PHP.

```
<ul>
<?php foreach ($todo as $item):?>
  <li> <?= $item ?></li>
<?php endforeach;?>
</ul>
```

Figur 4 vy i Cake php

2.4 Codeigniter

Codeigniter är en öppen källkod för att skapa fullständiga php baserade webbplatser och applikationer. Codeigniter är skriven i php och MVC-baserat, (Lancor & Katha, 2013). Codeigniter är ett lättviktigt ramverk som går enkelt att konfigurera, men det kan krävas att skriva mer rad koder än Cakephp, (Fernandes, J., Oliverira, P., Silva, C. (2012). Codeigniter använder model mapping vid koppling mot databas.

2.4.1 Model i Codeigniter

Model i Codeigniter ramverk är utformad för att arbeta med den valda databasen. Detta innebär vid kommunikation med en specifik tabell i databas måste göras manuellt. I figur 5 rad fyra visar hur kommunikationen sker.

```
1  <?php
2  class Todos_model extends CI_Model {
3  public function getData () {
4 $query =$this->db->get ('Todo_model');
5  return $query->result ();
```

Figur 5 Model i Codeigniter

I Figur 5 rad 2 skapar vi en klass. I rad 3 skapas en funktion för att hämta data från databas. I rad 4 variabel $query hämtar data från den valde tabell. Vid kommunikation med databas måste modellen laddas manuellt.

2.4.2 Controller i Codeigniter

Controller i Codeigniter används som en bro mellan vy och modellen. Controller hämtar svar för användarbegäran sparad i databas med hjälp av modellen och visar resultatet för användaren via vy.

```php
1 <?php if (! defined('BASEPATH')) exit ('No direct script access allowed');
2 class Todos extends CI_Controller {

3  public function __construct () {
4   parent::__construct ();
5      $this->load->model('Todo_model');
6    }
7  public function index ()
8  $all_todos = $this->Todo_model->getData ();
9  $data = array ();
10 $data['todos'] = $all_todos;
11 $this->load->view ('todos/index', $data);
 }
}
```

Figur 6 Controller i CodeIgniter

Figur 6 ovan visar på vilket sätt Controller fungerar i CodeIgniter. I rad två definieras klassen Todos och parent klassen CI_Controller. Rad fem ansvarar för att ladda modellen, eftersom Modell finns i Controller, betyder det att alla våra kontrollers handlingar kommer att ha tillgång till Modell med hjälp av $this->load->model. På rad åtta anropar funktioner som finns i model filen, och skickas till vy senare som ett argument. Rad nio gör informationen tillgängliga för vyn. Rad elva framställer vyn och passerar data.

2.4.3 Vy i Codeigniter

Codeigniter har sin egen metod och egna funktioner för att bygga, till exempel formulär, eller kan använda samma kod som används i Cakephp om det bara plain php och HTML som ska användas. Figur 7 visar hur vy visar data som hämtades från Controller för att visa den till användaren via webbläsaren.

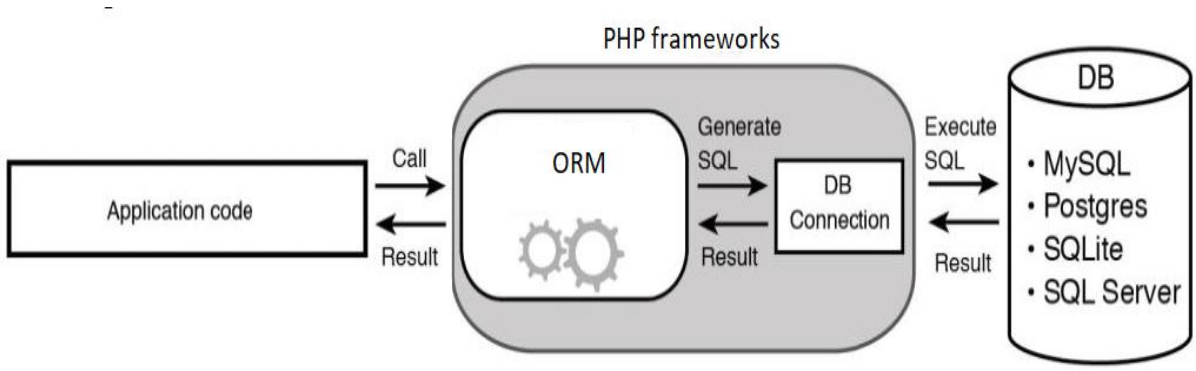
```
<ul>

        <?php foreach ($todo as $item): ?>
          <li> <?php echo $item; ?></li>
        <?php endforeach;?>
</ul>
```

Figur 7 vy i CodeIgniter

Figur 7 är ett kodexempel för foreach för att hämta data för array i controller. Koden som används i figur 7 är samma kod som används i vy i Cakephp. Anledning till att använda samma kod är för att visa likheten mellan båda ramverken vid användning av plain php och HTML i vy.

2.5 Object-Relational Mapping (ORM)

Enligt Matthew, C. Terry, H. (2010), ORM-Object relation mapping är en mjukvara som hjälper objektkodapplikation att koppla mot en relational databas. ORM omvandlar data mellan en typ system som inte kan samexistera inom relationsdatabaser och OOP programmeringsspråk, (Ireland et al., 2009). Istället för att bygga en tabell på ett traditionellt sätt som sker i SQL, ORM hjälper att skapa tabeller automatiskt, till exempel har vi en klass med fem egenskaper eller mer och vi vill skapa en tabell för klassen i databas, kan ORM skapa tabellen automatiskt utan att behöva göra den manuellt. Detta innebär att ORM skapar en tabell med samma namn som klassen och fem kolumner eller baserad på antal egenskaper i klassen. ORM hjälper php- utvecklaren att hämta data från tabellen i databas på ett enkelt sätt, genom att använda funktioner utan att behöva skriva databasfrågan manuellt.



Figur 8 ORM mappning mellan relation databas och OOP programmeringsspråk

```
<?php

   $car_list = new List ();

   $sql = "SELECT * FROM cars WHERE manufactory = 'Audi'";

   $result = mysql_query($sql); // I over simplify ...

   while ($row = mysql_fetch_assoc($result)) {

     $car = new car ();

     $car. manufactory($row.get('manufactory');

     $car_list.add($car);}

?>
```

Figur 9 SQL datahämtning från tabell Car

I figur 9 en enkel kod som visar hur data hämtning från en tabell sker utan att använda ORM. I figur 9 har vi en bilklass och vi vill hämta de bilar vars fabrik är Audi, för att hämta alla bilar behöver vi skriva lika mycket rad koder som visas i figur 9. Vid användning av ORM- Object relation mapping i figur 9 kommer vi inte skriva like mycket rad koder, utan räcker det med en rad kod som visas i figur 10 nedan.

```
$car_list = car. query (manufactory =" Audi ");
```

Figur 10 ORM datahämtning från tabell Car

2.6 prestandaoverhead vid användning av ramverk

I en artikel av Kathas och Lancors (2013), genomfördes en studie för att analysera och jämföra plain[3] php, Cakephp och Codeigniter. Båda Cakephp och Codeigniter MVC-baserade och har stöd för återanvändning av kod. Resultatet visade att det finns en betydande mängd av prestandaoverhead vid användning av ramverk. Cakephp och Codeigniter tog betydligt längre tid för att slutföra varje funktionalitet om man ska jämföra med plain php som tog mindre svarstid för att slutföra varje funktionalitet. Anledning till att båda ramverken tog längre tid än plain php är att ramverk behöver ladda sina inbyggda bibliotek innan applikationen utföra funktionalitet, och koppling mot databas går genom mjukvarans model mapping (ORM), därför ger plain php bättre prestanda än php-ramverk. Prestationsresultaten visas i figur 11.

Functionality	Plain PHP		CodeIgniter		Cake PHP	
	Time	LOC	Time	LOC	Time	LOC
Login/Logout	96	121	2291	192	1559	83
Session Mgmt	91	132	739	118	1309	102
Database	143	83	762	89	1492	52
Form Handling	487	209	1338	385	2262	191
From Validation	321	292	966	406	2411	310

Figur 11 jämförande analys av prestanda efter tid (ms.) och antal rad kod (LOC) för alla implementeringar

3 Plain php är php utan ramverk

3 Problemformulering

Kathas och Lancors (2013) har utfört ett experiment för att jämföra och analysera prestanda och svarstid för plain php och två PHP ramverk Cakephp och Codeigniter. Kathas och Lancors (2013) drar en slutsats plain php tydligt presterar bättre än Cakephp och Codeigniter, på grund av overhead vid användning av PHP-ramverk, ramverken måste ladda sina inbyggda bibliotek innan applikationen exekverar funktionalitet, vilket innebär att ramverk resulterar i mindre mängd av programkod men till kostnad av betydligt högre responstiden.

Frågan är om overhead minskar om man bygger applikation utan ORM. Kan utvecklaren få fördelarna av plain php och använda i php-ramverken genom att använda direkt koppla mot databas utan model mappning (ORM). Frågan är hur mycket responstiden skiljer mellan att använda php-ramverk med eller utan model mappning (ORM), och hur mycket overhead lägger ORM-verktyg till den relationsdatabasen i en webbmiljö? Är skillnaden mycket för att välja direktkoppling mot databas eller skillnaden liten nog för att använda ORM-verktyg istället för direkt koppling om det underlättar utvecklingen av en webbapplikation? Hur blir resultatet om man använder php-ramverken (Cakephp, Codeigniter) med model mapping (ORM) i jämför med att koppla sig till databasen på samma sätt men utan den mellanhand som ORM utgör.

Problemet med deras experiment är att fokus ligger på hur plain php, Cakephp och Codeigniter presterar gentemot varandra. De har inte heller fokuserad på kommunikation med databas, då de använder två sätt att kommunicera med databas enligt (Kathas och Lancors, 2013). Plain php använder direkt koppling mot databas, men Cakephp och Codeigniter har kopplats med model mappning (ORM) mot databas. Detta är då inget rättvist sätt att jämföra, eftersom man inte kopplar sig mot databasen på samma sätt. Det kan vara ORM som affekterar responstiden. Det skulle vara mer rättvist om de kopplar sig mot databasen på ett identiskt sätt.

En studie av Roopak et al. (2013), de jämförs relationsdatabasen mysql med objektdatabas, utan användning av model mappning (ORM) Deras slutsats var att relationsdatabasen mysql utförde select och delete operationer snabbare än objektdatabasen. Detta stämmer även överens med en undersökning av (Van Zyl et al., 2006). Enligt Van Zyl et al. (2006), model mappning (ORM) kan påverka på ett negativt sätt, när de jämförde ORM ramverk och db40. Resultatet visade att ORM ramverk kan också leda till betydande omkostnader i responstiden.

3.1 Hypotes

ORM lägger till betydande kostnader för databasinteraktion, formulärhantering, formulärvalidering och sessionshantering, eftersom det finns ett extra lager mellan applikationen och databasen (Roopak et al. 2013), och baserad på tidigare studie från Kathas och Lancors (2013). Hypotesen är då att responstiden i ramverk påverkas negativt av ORM om responstiden ska utvärderas med och utan mjukvaran model mapping (ORM). Hypotes och mothypotes ska formuleras på följande sätt;

1- PHP-ramverken CakePHP och Codeigniter kommer ha kortare responstid utan model mapping (ORM).
2- Användning av model mapping (ORM) kommer inte påverka responstiden hos CakePHP och Codeigniter ramverken.

3.2 Metodbeskrivning

För att kunna besvara frågeställning och hypotesen i studie kommer skapas två webbapplikationer med hjälp av CakePHP och Codeigniter. Både webbapplikationer kommer kopplas mot databas på två olika sätt, på det ena sättet kopplas webbapplikationen mot databas genom model mappning (ORM), och på det andra kopplas webbapplikationer direkt mot databas utan mellanhand som ORM. Efter att båda applikationer är färdiga kommer en kvantitativ metod utföras för att mäta svarstid. Under studie kommer vi mäta antal sekunder applikationen tar vid att utföra av exekveringstiden av databasinteraktivitet (CRUD[4]-funktionalitet), in och utloggning /session management, formhantering/validering.

Enligt Wohlin, C., Runeson, P (2012), när det gäller testning och analysera mjukvaruutveckling, så finns det tre typer av metoder som besvarar frågeställningen, och dessa metoder är litteraturstudie, fallstudier och experiment. Metodstrategi som ska användas för detta studie är experiment. experiment kategoriseras under kvantitativ forskning, eftersom det ger statistiskt signifikanta data somt ett objektivt och empiriskt tillvägagångssätt. Eftersom kvantitativ metod studerar det som kan avgränsas och mätas då den typ av metod passar bra till studie där vi behöver mäta och jämföra svarstid på olika sätt.

Anledning att inte använda kvalitativ metod som intervju och enkät är den typen av metoden fokuserar på specifikt mänskliga, och har brist på kontroll över prestanda och mätningar. Det andra skälet är att utesluta denna typ av metod är svårt för en person att märka resultatet när den mätas med millisekunder. Det skulle passa om resultatet var mer än fem sekunder och uppåt. Resultatet som vi får från kvalitativ metod är varierade och baserad på personer som testar, och den ska jämföras med experiment som vi får samma resultat vid upprepning och oberoende av vem som utför testet. (Hoxmeier och DiCesare, 2000).

Fallstudie kan vara ett alternativ till experiment, men ska inte användas för den studien eftersom i den studie behöver vi en kontrollerad miljö för att få ett stabilt resultat vilket inte görs vid fallstudie. Anledning till att den inte lämpligt för den typ av undersökning är fallstudie används för att undersöka händelser i deras naturliga miljöer som har verkliga saker som kanske påverkar det man undersöker i bakgrunden. Eftersom fallstudie undersöker ett specifikt fall inom ett generellt fenomen, så den mer lämpligt för levande miljöer och för verkliga situationer vilket vi saknar i det fallet i den studien, (Wohlin, C., Runeson, P 2012). Fallstudie är lämpligt bra för småskaliga experiment där ej behövas mycket kontroll över vad som sker. På ett enkelt sätt kan vi tolka att fallstudie fokuserar på utveckling mer än analysera eller mäta slutprodukt om man jämför det med experiment och kan användas för datasamling när man vill förstår ett viss socialt beteende (Denscombe, M. 2014).

[4] CRUD[1] -funktionalitet (Create, read, update and delete)

Så i princip ger experiment mer kontroll över variabler, testen och miljön än fallstudier, men det är begränsat av etik och oförmågan att kontrollera alla variabler under en lång tidsperiod. Frågan som dyker upp när man gör experiment i laboratorier om vi kan skapa samma miljö sam en riktig värld under experimentet. Och om den som undersöka beter sig på samma sätt i laborationen som i riktiga miljö.

4 Forskningsetik

Den första etiska problem under experimentet som dyker upp är data som ska användas vid mätning. För att undvika att hamna i upphovsrättslig situation måste den som gör undersökning ha rättigheter till användning av data under experimentet. Eftersom vi kommer behöva data som e-post eller användarnamn under experimentet, till exempel kan vara en loga in/loga ut form eller validering av data, så det är viktigt att dessa data ska hanteras på rätt sätt. Hämtas data utanför personer som är deltagen i studie, så det är viktigt att förstå att det kan kräva ett avtal från personer och där personer godkänner avtalet. Det måste också finnas intresse över vägning som säger att personer som är inblandade i studie behöver spara på vissa data om personer. Det är viktigt att förstå att personer har rätt att få sina sparade data raderade och den ska göras under ett kort tid.

Ett annat etiskt problem måste tänkas på är licens hantering. Det innebär att ha koll på vart man hämtar mjukvara från internet. Det är mycket viktigt att alla mjukvaror som ska användas för att utföra experimentet är korrekt licensierad. Mjukvaran som ska användas är inte öppen källkod, då måste man få tillåtelse att använda mjukvaran.

När det gäller användning av hårdvara, så det är också bra att fokusera på, eftersom vi kommer att jämföra responstid mellan två mjukvaror, så måste att båda ska analyseras i samma hårdvara. Ska man använda organisation utrustning, måste den som använder utrustning följer policy som organisationen har och inte bryta mot den. Vid användning av privata hårdvara, där undersökare måste hantera data som hämtas om personer på rätt sätt och få den raderad efter experimentet är klart.

för att skulle komma fram till transparens och replikerbarhets alla metoder som koder och data i databasen används under experimentets kommer finnas och publiceras i github. Det som inte kommer publicerar är den verkliga informationen om det använda i experimentet.

5 Genomförande

5.1 Litteraturstudie

Innan webbapplikation utvecklas är det viktigt att hitta de litteraturerna som studie baserad på, för att underlätta arbetet vid experiments replikering av andra personer. En litteratursökning är bra att ha under experiment eftersom informationen man får från litteratur är fakta och den kommer controllers av andra forskaren. Informationen som finns i litteratur förändras inte enkelt, det innebär samma information är tillgängligt även lång fram i tiden om den skulle jämföras med information som kan hittas på internet som kan försvinna eller ändras på något sätt. För detta studie kommer några litteraturer och tillgängliga information som finns på internet att användas.

En litteraturstudie skrivet av Oates (2006). Ska användas som guide under examensarbetet gång, litteraturen ska användas som grund till detta studie. Oates (2006). Hänvisar att en litteraturstudie delas upp i två faser. Den första fasen hjälper studenten att samla alla informationer som ger en bättre förståelse och hur ska studenten forma forskningsfråga och hypotesen på ett bra sätt. Den andra fasen är granska informationer som insamlades under första fasen och efter att forskningsfråga och hypotesen utformades sedan ligga de till grund för litteraturstudiens teoretiska referensram (Oates 2006). Detta hjälper för att underlätta forskningens kunskapsbidrag.

För att kunna förstår Codeigniter:s strukturell kommer en användbar dokumentation som kan hittas på Codeigniter:s hemsida användas. Dokumentation ger kodexempel, guider och förklaring som ger en full förståelse för över vad du gör och hur Codeigniter fungerar. En annan litteratur som ska också användas under studie är en bok som skrivs av Rob Foster (2013). Rob Foster (2013) förklarar i kapital 1 tillexempel, hur kan Codeigniter laddas ner och installeras, boken har också visat grundläggande konfigurationsalternativ och säkerhet inom Codeigniter ramverk och den innehåller kodexempel som kan använda under experimentet för att lösa vanliga problem. Eftersom Codeigniter är MVC-baserad så valde en litteratur skrivet av Av Eliahou Orr, Yehuda Zadik (2013), som hjälper på att förstå hur MVC i Codeigniter fungerar. ” CodeIgniter Web Application Blueprints” är en bok skrivet av Rob Foster (2015) som kan vara mycket användbarhet under studie eftersom boken ger goda råd hur olika applikationer kan byggas samt kodexempel, tillexempel kapital 8, talar om att skapa en To-do List applikation.

I CakePHP:s hemsida finns det också en snabbstartsguide som visar hur ramverken CakePHP ska installeras på webbserver. I snabbstartsguide hittar man också hur kommunikation med databas sker, och några kodexempel finns angående kopplingen mot databas som MySQL/MariaDB och PostgreSQL (Cake Software Foundation, 2020). I boken CakePHP Application Development skrivs av Ahsanul, B. & Anupom, S. (2008) finns det förklaring för utvecklaren, i kapital 9 finns en guide för att skapa en ny webbapplikation. I kapital 11 i boken innehåller den en beskrivning ifall utvecklaren vill använda AJAX och JavaScript med CakePHP. I kapital 12 finns det en guide för hur ska ORM användas i CakePHP.

Både CakePHP och Codeigniter skrivs i PHP programspråk, (Das R., Saikia L. P. 2016; & Biermann, 2014), detta innebär både ramverken är anpassade för att köras i serversida. En

webbserver behövs för att kunna installera båda ramverken, eftersom båda applikationen ska installeras lokalt iså fall kommer användas XAMPP som lokal webbserver.

I båda CakePHP och Codeigniter:s hemsida det finns en guide ångande model mapping (ORM) som ger en full förståelse för över vad du gör och hur model mapping (ORM) fungerar. (Cake Software Foundation, 2020), (Codeigniter Software Foundation, 2020). ” Handbook on Architectures of Information Systems” är en bok skrivet av Peter, B. & Kai, M. Günter, S. (2006) som förklara vad är model mapping (ORM) och hur kopplas den mot databas. Eftersom examensarbetet frukostar på ifall model mapping (ORM) kan påverka webbapplikationen kommer ett påstående skrivet så av Wegrzynowicz, P. (2013), ha under litteraturstudie. Wegrzynowicz, P. (2013), visade att model mapping (ORM) kan påverka webbapplikationen negativt, eftersom det finns ett extra lager mellan applikationen och databasen som affekterar responstiden. För att kunna visa att model mapping (ORM) kan påverka webbapplikationen negativt, kommer skaps webbapplikationer med och utan model mapping (ORM). I den applikation som ska skapas utan model mapping (ORM), kommer använda MySql för att koppla applikationer mot databas, vilken är väldigt bra att ha litteraturen ”PHP and MySQL Web Development” skriv av Luke, W. & Laura, T. (2003) under denna fasen.

En litteratur som kommer tas under denna studie skrivs av Ali Raza Fayyaz and Madhia Munir (2014), där de utvärderade prestanda för två PHP-ramverk där de använde en ram med ORM och en utan ORM, eftersom jag kommer jämföra mellan ramverken med ORM och utan ORM, så denna litteratur kommer vara mycket användbar under studie. Den mest litteraturstudie som används under studie hittades på internet genom att använda sökmotorer som Google.com, Google Scholar. Nyckelord som används för söka efter litteraturstudie som model mapping (ORM), CakePHP, Codeigniter MYSQL och PHP.

5.2 Implementation

I det kapitlet beskrivs hur webbapplikationer för experiment skapas, vilket innebär vilken webbserver samt vilken typ av databas ska användas, även beskriver vilken version av ramverken kommer användas och hur installationen sker, samt vilken mjukvara ska användas för båda ramverken.

5.2.1 webbserver

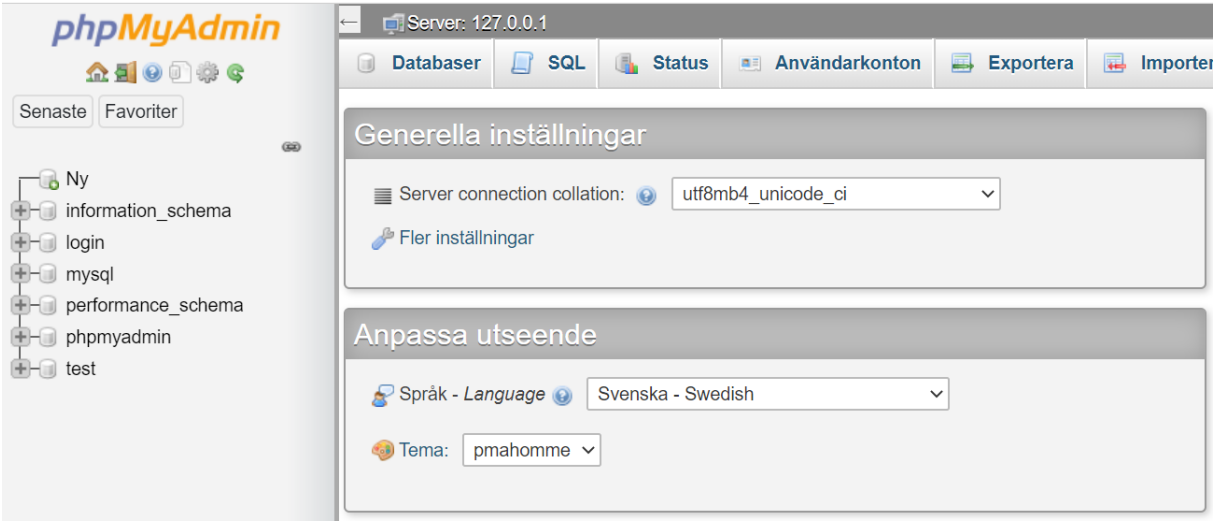
Vid skapning av webbapplikationer som är skrivna i PHP programspråk är det vanligt att kräva en webbserver, server som ska användas är lokalwebbserver kallas XAMPP. XAMPP är en mjukvara från Apache Friends som gör att man får lokalserver, Apache Friends (Sandi, 2017). Vid installation av XAMPP medföljer bland annat Apache, PHP och MySQL, dessa mjukvaror som installera med XAMPP är viktiga för att svara på våra frågeställningar.

Mjukvara	**Version**
Apache	2.4.41
MySql	10.4.11.0
PHP	7.4.2.0
phpMyAdmin	5.0.1

Tabell 1 Mjukvaruspecifikationen på XAMPP

5.2.2 Databaser

Databas som ska användas under experimentet är för både ramverken, skapas med hjälp av phpMyAdmin. PhpMyAdmin är en mjukvara som är skapats i PHP programspråk för att administrera MySQLl via gränssnitt. Via gränssnittet kan användaren skapa flera databaser och tabeller (Chowdhury, 2017). PhpMyAdmin kommer installeras automatiskt när XAMPP server installeras. Figur 12 visar hur phpMyAdmin ser ut.



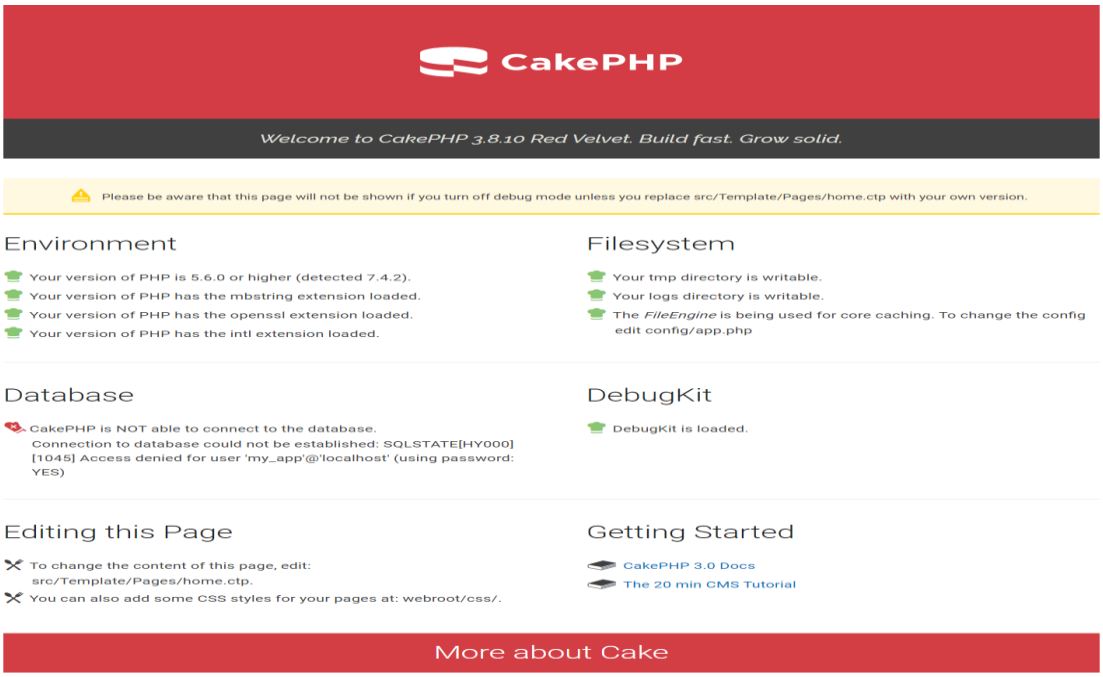
Figur 12 phpMyAdmin webbsida

5.2.3 Installera av ramverk

Installationen av CakePHP sker genom en exekveringsfil laddas ner från Composers. En Composer är ett kommandorads program som hjälper utvecklaren att installera ramverk och paket (Composer, 2020). För att installera CakePHP används kommandot i figur 13 i kommandotolken(cmd), i Windows laddas ner CakePHP paketet och flyttas till lokal webbservers fil.

```
Composer create-project –prefer-dist cakephp/app cake-auth
```

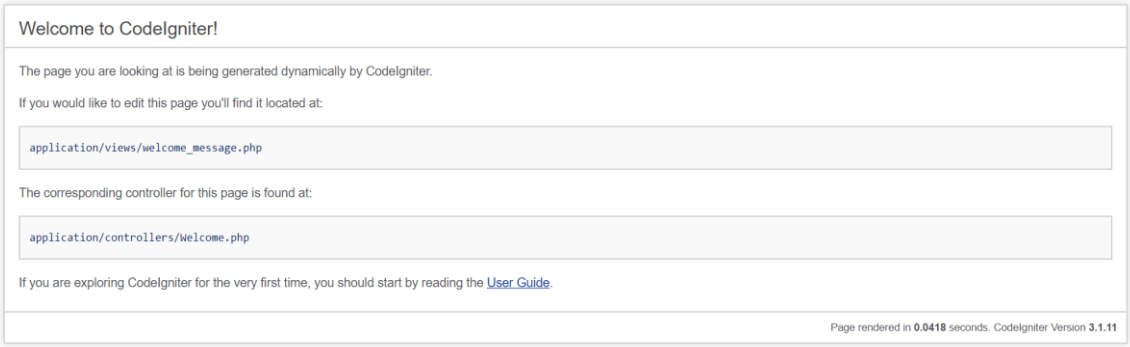
Figur 13 Kommandot för att installera CakePHP



Figur 14 CakePHP hemsida

Figur 14 visar standardsida för CakePHP efter installation.

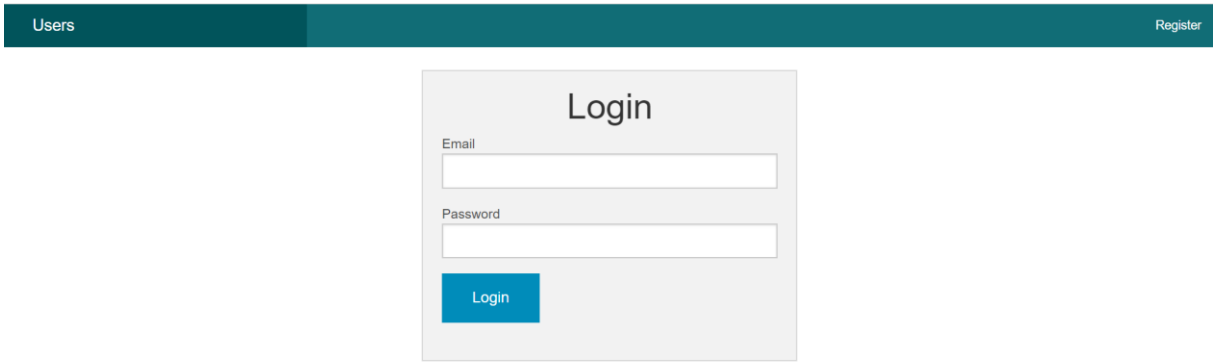
Installationen av Codeigniter skiljer sig från CakePHP. Vid installation av Codeigniter behövs inte Composer, utan bara ladda ner filen från Codeigniter webbsida och flytta den till webbserver XAMPP. Figur 15 visar standard hemsida för Codeigniter.



Figur 15 Codeigniter startsida

5.2.4 Inledning av webbapplikationens utveckling

För att kunna utföra experimentet är det nödvändigt att skapa vissa avgränsningar för båda ramverken. Det finns viktiga aspekter att peka på vid skapning av webbapplikationen att den ska innehålla funktionalitet som behövs för att kunna utföra experimentet. Båda ramverken ska innehålla funktioner som behövs för att kunna göra mätningar som besvara frågeställningar. Funktionen som ska skapas är logga in/ut form för att kunna mäta inloggning och utloggning se figur 16, registrering form, och Post form för att mäta databasförfrågan. Funktioner som ska mätas i Post form är INSERT, UPDATE, VIEW och DELETE (CRUD). se Figur 17.



Figur 16 logg in form

Figur 17 logga in form

CakePHP och Codeigniter webbapplikationer ska ha samma utseende och samma funktionalitet, därför ska CSS fil som används i CakePHP användas för Codeigniter ramverk.

Eftersom experimentet kommer att påpeka ifall ORM funktionen kommer påverka responstid hos båda ramverken, så kommer vi att skapa två versioner av varje ramverk. En version ska byggas med ORM funktionalitet och den andra ska byggas utan ORM, vilket kommer vi använda RAW SQL förfråga istället vid kommunikation med databas. I detta fall kommer vi kunna hitta svart ifall ORM påverkar responstiden hos både ramverken. I figur 18 och 19 är ett exempel som visar skillnad när CakePHP anropar funktion View på två olika sätt. I figur 18 där används ORM för att hämta data från tabell Post, och i figur 19 hämtar vi data från databas med RAW SQL utan att använda ORM.

```
/**
     * View method
     *
     * @param string|null $id Post id.
     * @return \Cake\Http\Response|null
     * @throws \Cake\Datasource\Exception\RecordNotFoundException When record
not found.
     */
    public function view($id = null)
    {
        $post = $this->Posts->get($id, [
            'contain' => ['Users'],
        ]);

        $this->set('post', $post);
    }
```

Figur 18 hämta data från tabell Posts i CakePHP med ORM

```
    /**
     * View method
     *
     * @param string|null $id Post id.
     * @return \Cake\Http\Response|null
     * @throws \Cake\Datasource\Exception\RecordNotFoundException When record
not found.
     */
    public function view($id = null)
    {
        $conn = ConnectionManager::get('default');
        $post = $conn-
>execute("SELECT a.*, b.name FROM posts a JOIN users b ON a.user_id = b.id WHE
RE a.id = $id ")->fetchAll('assoc');

        $this->set('post', $post[0]);
    }

    /**
```

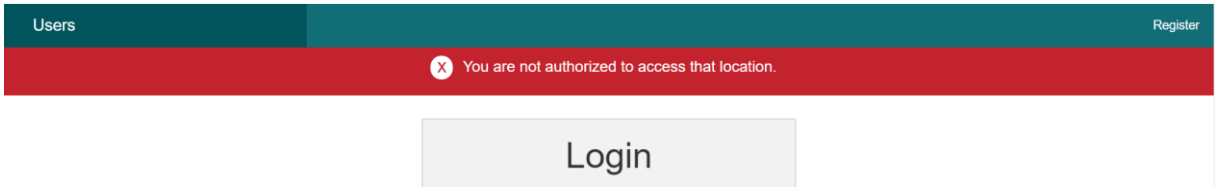
Figur 19 hämta data från tabell Posts i CakePHP utan ORM

5.3 Progression

Under progression kapitlet kommer vissa förändringar som skett under webbapplikationens utveckling att diskuteras.

5.3.1 CakePHP

Alla webbapplikationer som användas under experimentet kommer installeras i den sökvägen *C:\xampp\htdocs\ExJobb*. För att kunna utföra experimentet behövs det två versioner av CakePHP och två versioner av Codeigniter, en ska installeras med ORM funktionalitet, medan den andra versionen ska installeras utan ORM funktionalitet, vilken kommunikation med databas ska ske genom att använda RAW SQL. Som jag nämnde innan, CakePHP installeras med hjälp av Composer. Ett problem som dykte upp senare när jag lyckats med installation av båda versioner av CakePHP ramverk, fick en av dom att funka. Efter ett antal försök lösningen var att flytta en av de till en annan mapp utanför ExJobb mappen. Anledningen till detta var de använder samma kod tillskillnaden var det bara kommunikation sättet med databas, vilket gör att Composer inte kände till att det finns två skilda webbapplikationer i en gemensam mapp. Figur 20 visar fel meddelandet när jag installerade två CakePHP, en med ORM och den andra utan ORM.



Figur 20 fel meddelandet vid ladda sida CakePHP i samma huvudmapp.

5.3.2 Byta av språk i gränssnitt[5]

Eftersom har valt skriva rapporten på svenska, därför webbapplikationens användargränssnitt för både CakePHP och Codeigniter är byggda på svenska. Efter några misslyckande för att översätta några ord från engelska till svenska, så blev andra alternativet att bygga webbapplikation på engelska. Att bygga webbapplikation på svenska innebär det att mer tid krävds och även kan krävas blanda in tredjepartsbibliotek, vilket kan påverka mätning under utförande av experimentet. Att blanda in andra bibliotek kan påverka experimentet men byta av språk kommer inte påverka negativa mätningar.

5.3.3 mätningsmetod

Från början tänkte att använda XDebug extension för att mäta responstid. XDebug är för PHP för att hjälpa till med felsökning och utveckling. I början användes XDebug dokumentation (XDebug, 2020). XDebug installeras i php.int som ligger i PHP mapp under xampp mapp. Efter ett antal misslyckade försök har bestämt mig att hitta andra metod för att mäta responstid för båda applikationer. Ett val var att använda PHP microtime () Function. microtime()[6]Function returnerar den nuvarande Unix-tidsstämpeln med mikrosekunder. Figur 21 visar de koden som användas för att mäta responstid hos båda ramverken.

[5] https://github.com/a16hiwah/ExamJob/commit/d0b6bd0b3c0ac4902ba0263ca6db9103581e88d3
[6] https://github.com/a16hiwah/ExamJob/blob/master/Cakephp/src/Controller/AppController.php

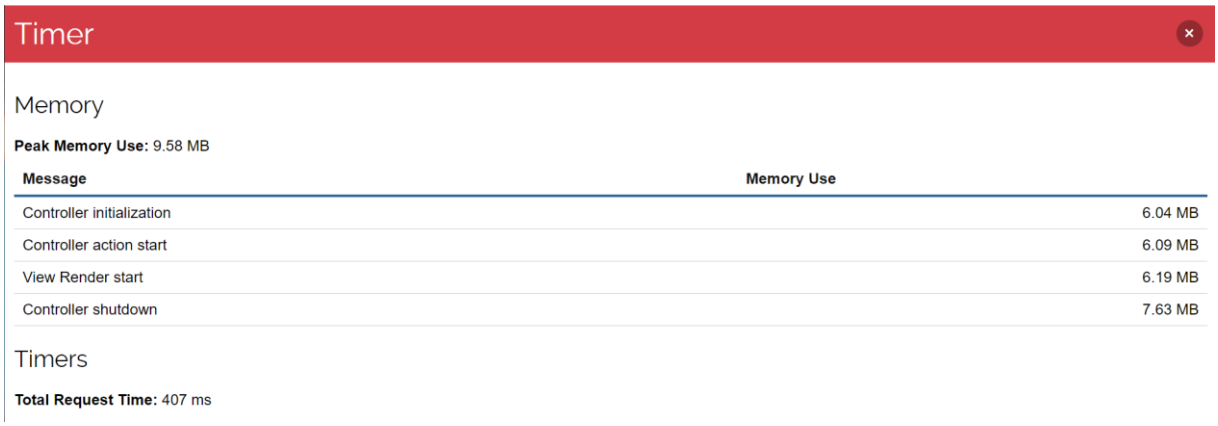
```php
<?php

$time_start = microtime(true);

// Sleep for a while

//usleep(100);

// except the PHP closing tag

$time_end = microtime(true);

// Calculate execution time in milliseconds (round to 3 decimals)

$time = $time_end - $time_start;

$time = round((($time_end - $time_start) * 1000), 3);

// Display the result on the rendered web page to make it possible to get it

// with JavaScript

echo "loddid in $time seconds\n";

// Path to csv file where the result should be saved (choose one)

$fileLocation = getenv('DOCUMENT_ROOT') . '/logs/codeigniter_measurements.csv';

// $fileLocation = getenv('DOCUMENT_ROOT') . '/logs/cakephp_measurements.csv';

// Save the result

$handle = fopen($fileLocation, 'a');

fputcsv($handle, [$time]);

fclose($handle);

?>
```

Figur 21 kod som används för att mäta responstid for cakePHP och Codeignter

PHP microtime () Function har också inte fungerat som den ska under mätningar som gjordes under pilotstudie. PHP microtime () Function gav inte den rätta siffran vid mätningar. Till exempel den gav hög outputtid vid utföring av bara log in funktionalitet i CakePHP, Och i Codeigniter fick man fler outputtid istället av en output vid utföring av bara log in funktionalitet[7].

Ett alternativ som var relativt bra hittad vid läsning av CakePHP dokumentation är Debug Kit[8]. Debug Kit är ett plugin som tillhandahåller ett antal bra felsökningsverktyg vid varje utförande. Vid utfört av funktionalitetresultat visas typ gränssnitt. se figur 22 för mer förklaring.

[7] https://github.com/a16hiwah/ExamJob/blob/master/Codeigniter/application/controllers/Login.php
[8] https://github.com/a16hiwah/ExamJob/tree/master/measurement%20method/install_debug_kit_on_cakephp



Figur 22 BENCHMARKS funktionalitet för CakePHP

Benchmark[9] kan också aktiveras i Codeigniter genom att använda Benchmark-klassen. Benchmark-klassen kan användas i kontroller, vyer eller modeller. För att kunna använda Benchmark-klassen man ska markera en startpunkt och markera en slutpunkt.

```
1-  $this->benchmark->mark('code_start');

2-  // Some code happens here

3-  $this->benchmark->mark('code_end');

4-  echo $this->benchmark->elapsed_time('code_start', 'code_end');
```

Figur 23 Benchmark-klassen för mätning av responstid i Codeigniter

Figur 23 rad 1 är startpunkt för att börja mätningen. Rad 2 funktionen som vi vill mäta exempelvis log in funktionen. Rad 3 är slutpunkt där vi vill stoppa mätningen. Rad 4 visar resultat via gränssnitt. Funktion som används i exemplen ovan, figur 23 för att beräkna tiden mellan två punkter, det innebär varje funktion ska mätas separat och man måste flytta koden manuellt ifall man har flera funktioner att mäta. För att mäta alla funktioner automatiskt utan att behöva flytta koden manuellt ska Hooks - Extending[10] aktiveras.

Som standard Hooks – Extending är inte aktiverad i Codeigniter ramverken. Funktioner för Hooks kan aktiveras / inaktiveras globalt genom att ställa in följande objekt i figur 24 i filen applikation / config / config.php:

```
$config['enable_hooks'] = TRUE;
```

Figur 24 Hooks aktiveras / inaktiveras

[9] https://github.com/a16hiwah/ExamJob/commit/57f4ef134209d1e722ac7e5ad4a3ed485452f2d1
[10] Hooks-funktionen ger ett sätt att utnyttja och ändra ramens inre funktioner utan att ändra-Main filerna.

Hook funktioner definieras i applikationen / config / hooks.php-filen. Varje Hook specificeras som en array med följande prototyp i figur 25:

```
1-  $hook[' post_controller_constructor '] = array(
2-   //'class'    => 'MyClass',
3-  'function' => 'profiler_hook',
4-  'filename' => 'profiler.php',
5-  'filepath' => 'hooks',
6-   //'params'   => array('beer', 'wine', 'snacks')
);
```

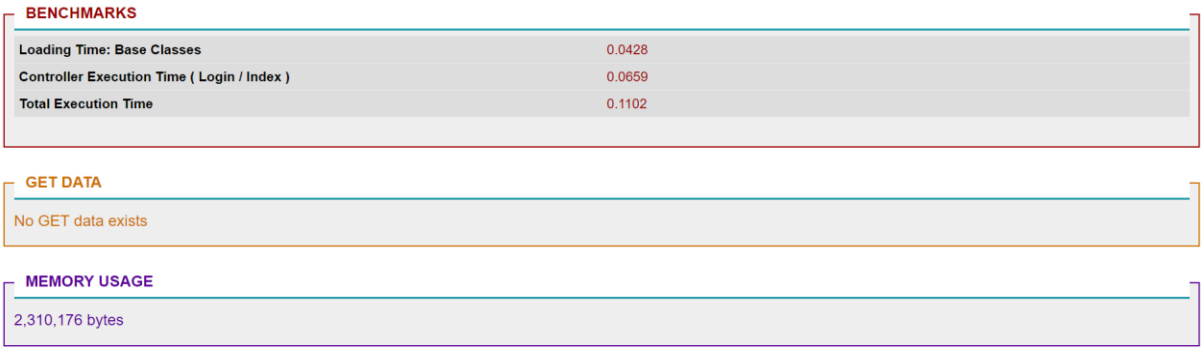
Figur 25 prototyp som användas för att aktivera Hook funktion i Codeigniter

Rad 2 i figur 25 visar vilken funktion ska anropas. Rad 4 visar var funktionen som anropas i rad 2 ska finnas. För att visa resultat via gränssnitt kommer följande funktion i figur 26 att skapas under application/hooks/ profiler.php

```
<?php

Finction profiler_hook(){

$CI =& get_instance();

$CI->output->enable_profiler(TRUE);

}
```

Figur 26 För att vissa resultatet via gränssnitt

Figur 27 visar resultatmätning via gränssnitt som skapas under Hook funktionen.



Figur 27 resultat gränssnitt för Codeigniter Benchmark-klassen.

5.4 Pilotstudie

Under det kapitlet kommer använda webbapplikationer som skapades för att se om mätningarna fungerar på ett korrekt sätt. Båda webbapplikationen byggs i PHP ramverk. Det första webbapplikationen skrivs i CakePHP och kopplas på två olika sätt, i det första sättet ORM används för att koppla mot databasen, och på det andra sättet är att direkt koppla webbapplikationen mot databasen utan ORM. Det andra webbapplikationen skrivs i Codeigniter och kopplas på två olika sätt mot databas också. Kopplingen mot databasen sker i två sätt genom att använda ORM och genom att direkt koppla applikationen mot databas utan mellan hand ORM. CakePHP och Codeigniter kräver en server. Under experimentet kommer en lokal server användas för att lokala server påverkas inte av nätverks hastighet och tidsåtgång vid mätningar. Lokala server som ska användas är XAMPP, från Apache Friend (Sandi, 2017).

5.4.1 Förberedelser inför testning

För att kunna genomföra projektet på rätt sätt och få ut rätt mättning på responstider under experimentet, måste gränssnitt innehålla de funktioner som krävs för att få de mättningar som behövas. Funktioner som ska finnas i applikationen är en login/logout forum, och databasinteraktivitet (CRUD: skapa, läsa, uppdatera och tar bort). båda applikationer kräver en databas. PhpMyAdmin är en mjukvara som hjälper med skapning av databasen för båda applikationer. Responstid för webbapplikationer som skapades med CakePHP mäts med en CakePHP extension som kallas för debugkit- extension. Debugkit hjälper till att få ut tider som funktioner tar för att utföras (Cakephp Cook Book, 2020). För att mäta responstid för webbapplikationer som skapades med Codeigniter kommer Benchmarking Class att installeras, (Codeigniter Benchmarking class, 2019).

5.4.2 Tillvägagångssätt

Experimentet kommer att genomföras lokalt för båda pilotstudier och det fullskaliga experimentet. Tabell nedan visar vilka mjukvaror ska användas och vilken version av webbserver och webbapplikation ska installeras och i vilken hårdvara ska hela experimentet utföras.

Operativsystem	Windows 10 Pro version 1909
Webbläsare	Google Chrome version 81.0.4044,113
Webbserver	Apache 2.4.41
Php-version	7.4.2.0
Databas	MySQL 10.4.11.0
CakePHP	3.8
Codeigniter	3.1.10
CPU	Intel Core i5 3570l
Lagring	500GB TOSHIBA-TR200(SATA(SSD))
Graphics	Nvidia GeForce CTX 980 Ti

5.4.3 Insamling av data inför testning

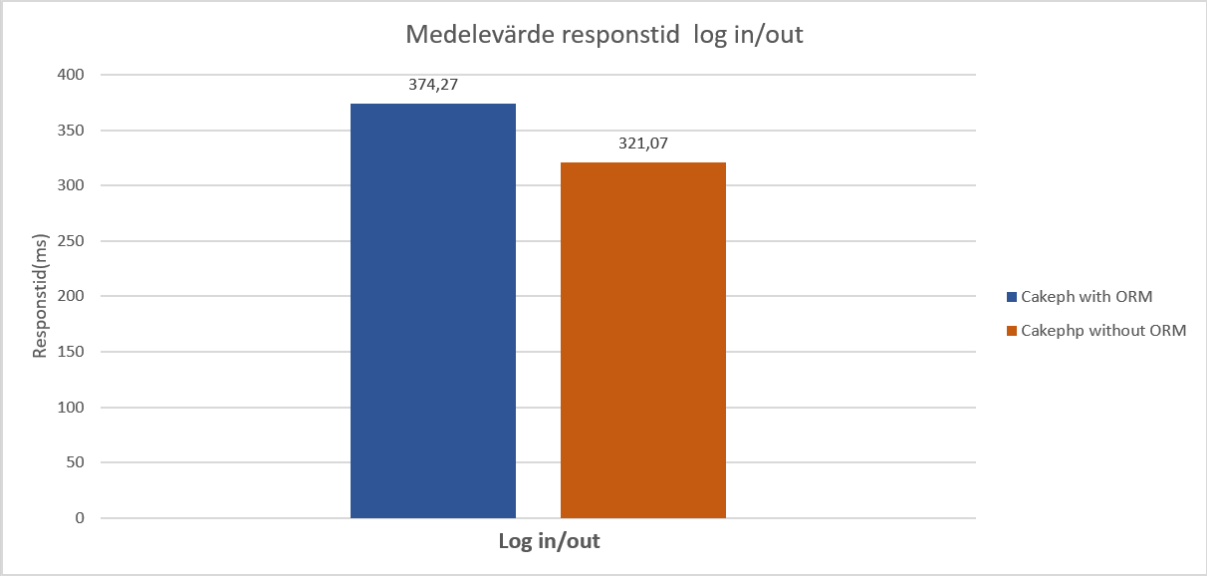
Data som ska samlas för att sammanställa ett resultat samlas genom att genomföra 30 testtillfällen för varje funktionalitet som finns i båda CakePHP med och utan ORM, och Codeigniter med och utan ORM. Funktioner som ska testas är login/logout och databasinteraktivitet (CRUD: Create, read, update och delete). vid testning av login/logout kommer 30 användare skapas, sedan räknas den tiden ramverken tog från och med användaren har tryckt på knappen logga in tills hen inloggade och tvärtom vid logga ut. Vid testning av databasinteraktivitet kommer 30 post skapas med olika långa texter, sedan räknas den tiden varje ramverk tog för att utföra databasinteraktivitet. Mätvärdena kommer sparas ner i en Excel fil. Inför testning ska användas tre olika långa texter. Texterna ska hämtas från https://www.lipsum.com/. Figur 3 visar dessa texter som ska användas under experimentet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus et.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut suscipit velit eget turpis faucibus consectetur. Sed sagittis scelerisque nibh. Fusce consectetur eu sapien rutrum gravida. Duis ut sagittis tellus. Donec dignissim tempus nisl sit amet interdum. Quisque non velit a ante molestie facilisis sed a mi. Donec feugiat laoreet libero.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec posuere, erat id eleifend ultrices, nisl dui volutpat felis, sed commodo lorem mauris ut justo. Cras ultrices bibendum quam vitae viverra. Quisque eget magna vel libero ullamcorper iaculis. Sed at porta lacus. Donec iaculis leo sed mi pharetra volutpat. Nunc vitae ex sit amet nisl pharetra gravida. Duis eleifend suscipit lacus sed semper. Cras eget sagittis mauris, in ullamcorper magna. Etiam vitae sapien ante. Pellentesque tristique leo ligula. Pellentesque efficitur ipsum non ex sollicitudin convallis. Sed lacinia lorem rhoncus quam scelerisque, vel imperdiet massa mattis. Suspendisse a ligula non elittempus.

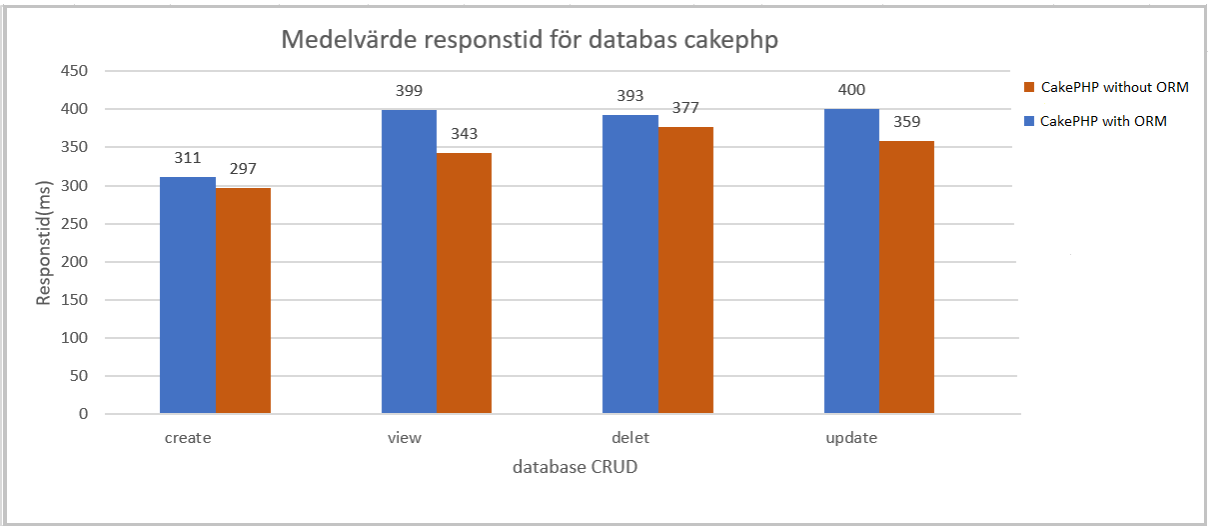
Figur 28 data användas vid databasinteraktivitet (CRUD: Create, read, update och delete).

5.4.4 Resultat av pilotstudie

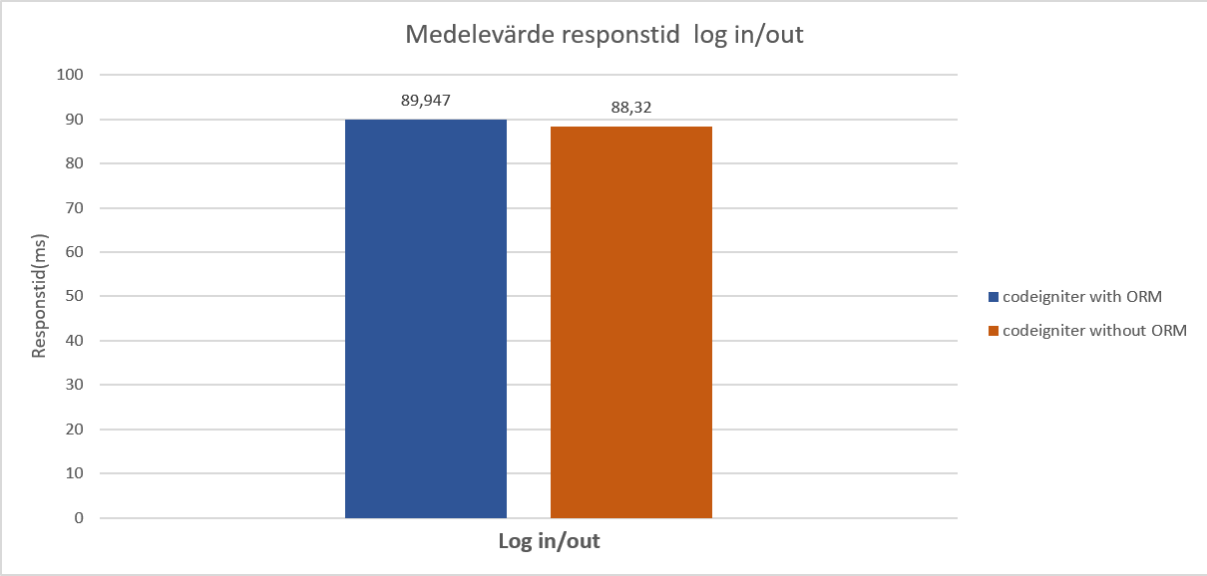
Mätningar under pilotstudie ger en indikation på att model mapping (ORM) kommer påverka negativ på båda ramverken. detta innebär att båda CakePHP och Codeigniter kommer att prestera bättra om de ska direkt kopplas mot databas utan mellan hand model mapping (ORM). Enligt en av alternativa som namnades i hypotesen PHP-ramverken CakePHP och Codeigniter kommer ha kortare responstider utan att använda model mapping (ORM).



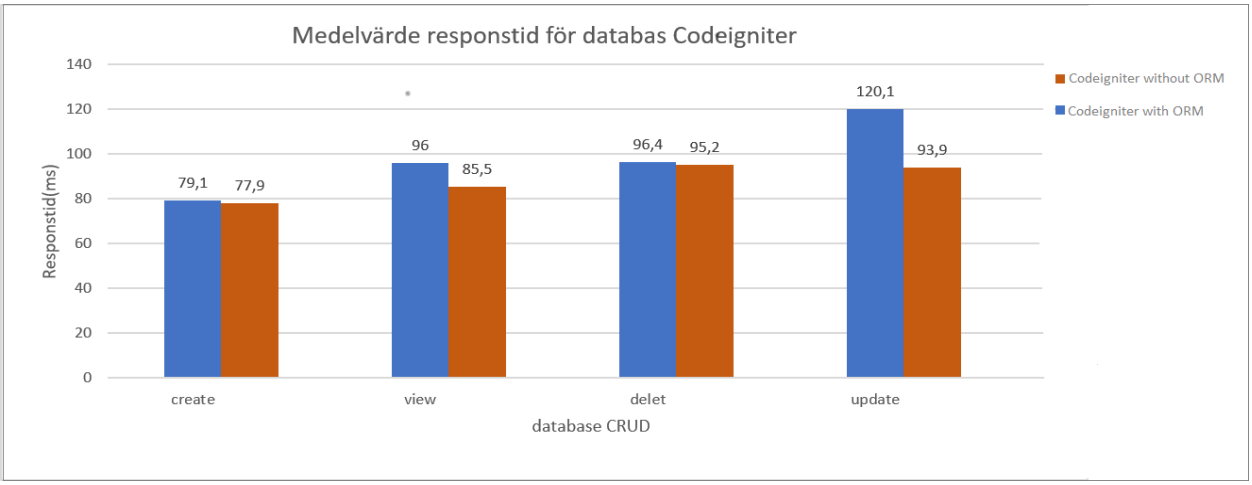
Figur 29 resultatet vid genomföring logga in/ut i CakePHP



Figur 30 resultatet vid genomföring av databasinteraktivitet i CakePHP



Figur 31 resultatet vid genomföring logga in/ut i Codeigniter



Figur 32 resultatet vid genomföring av databasinteraktivitet i Codeigniter

6 Utvärdering

Resultat från pilotstudien har visat hur kan ORM påverka responstid hos båda CakePHP och Codeigniter. Resultaten visade att CakePHP och Codeigniter utan model mapping (ORM) var någon enstaka millisekund snabbare. Under utvärderings kapitel kommer en mer omfattande undersökning att göras för båda ramverken, genom att öka antal användare vid inloggning och utloggning, och öka av antal post vid testning av databasinteraktivitet (CRUD).

6.1 Presentation av undersökning

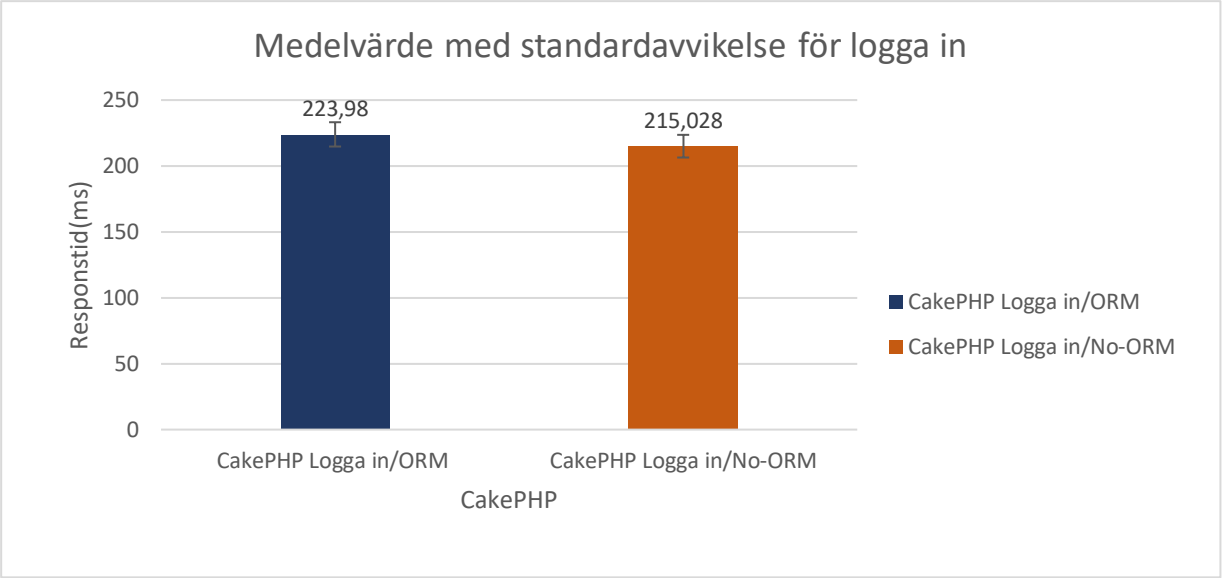
Experiment kommer att ske genom att räknas den tiden ramverken tog från och med användaren har tryckt på knappen logga in tills hen inloggade och tvärtom vid logga ut. Vid testning av databasinteraktivitet räknas den tiden varje ramverk tog för att utföra databasinteraktivitet, det innebär att vi kommer mäta tiden som ramverken tog för att utföra *INSERT, SELECT, UPDATE*, *DELETE* funktionalitet i databasen. för att få mer tillförlitliga resultat kommer mätningar ökas till 500 mätpunkter, jämfört med 30 för vid inloggning och utloggning i pilotstudie, och 500 mätpunkter för varje CRUD funktionalitet jämfört med 30 för varje i pilotstudie.

6.2 Analys

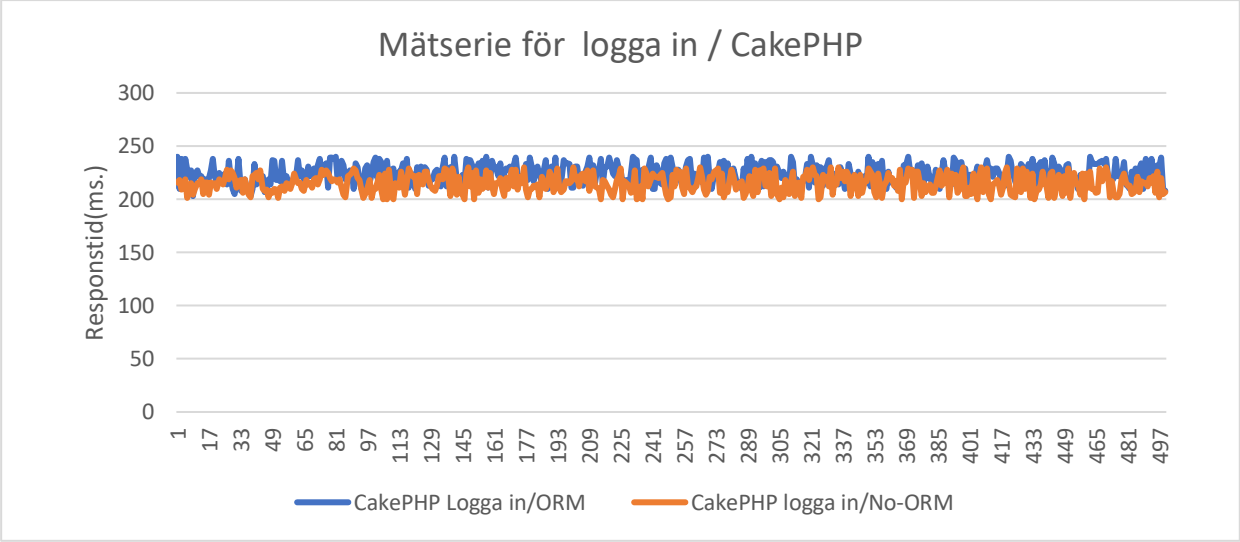
I detta studie vi kommer fokusera på att se ifall model mapping (ORM) funktionen kan påverka responstid hos båda ramverken CakePHP och Codeigniter. För utföra experimentet kommer genomföra ett antal testfall på några funktionaliteter hos båda ramverken. resultatet från detta kapitel kommer analyseras för att sedan kunna bekräfta eller avvisa hypotesen som i sin tur leder till att besvara frågeställningen.

6.2.1 CakePHP Testfall 1-inloggning och utloggning

I testfall 1 kommer vi mäta den tiden som användaren tar för att logga in och logga ut. För att utföra testfall 1 kommer skapas ett antal användarnamn och lösenord, sedan mätas tiden som varje användarnamn tog för att logga in och logga ut. Under testfallet 1 kommer genomföras 500 inloggning och utloggning för båda CakePHP webbapplikation med/utan model mapping (ORM), sedan mätas tiden varje webbapplikation tog för att utföra dess 500 inloggningar och utloggning. resultatet som har tagit fram i testfall 1 kommer visas i tabeller och grafer.



Figur 33 Medelvärde med standardavvikelse för logga in CakePHP webbapplikation



Figur 34 Mätserie logga in för både CakePHP med ORM och utan ORM

Anova Table

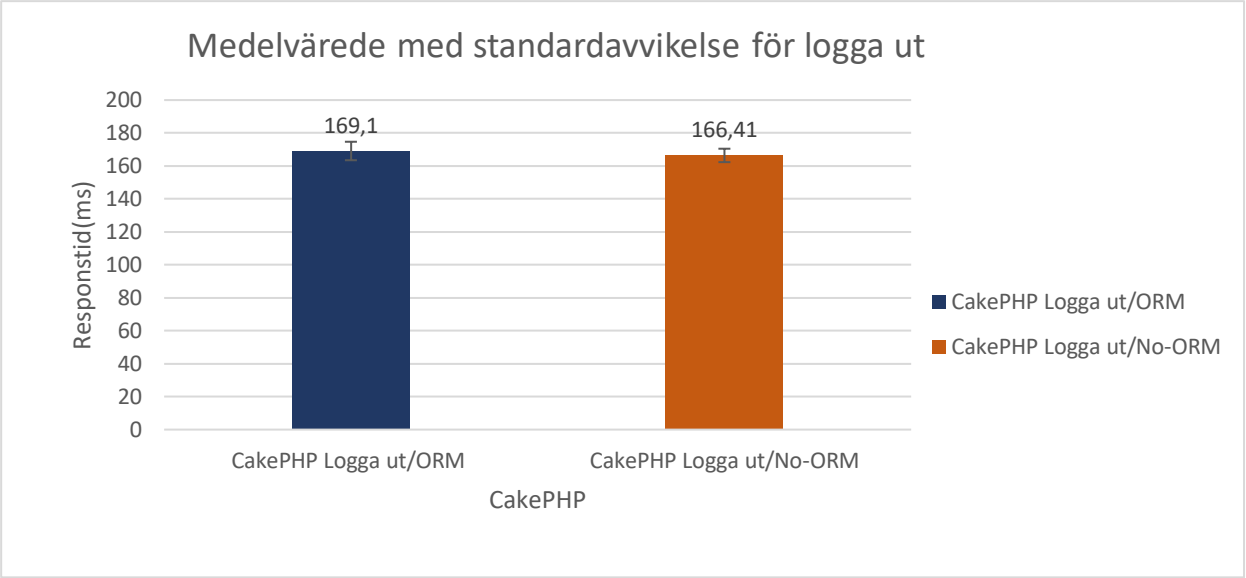
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Cakephp/ORM	500	111990	223,98	85,1018
Cakeph/NoORM	500	107514	215,028	74,01525

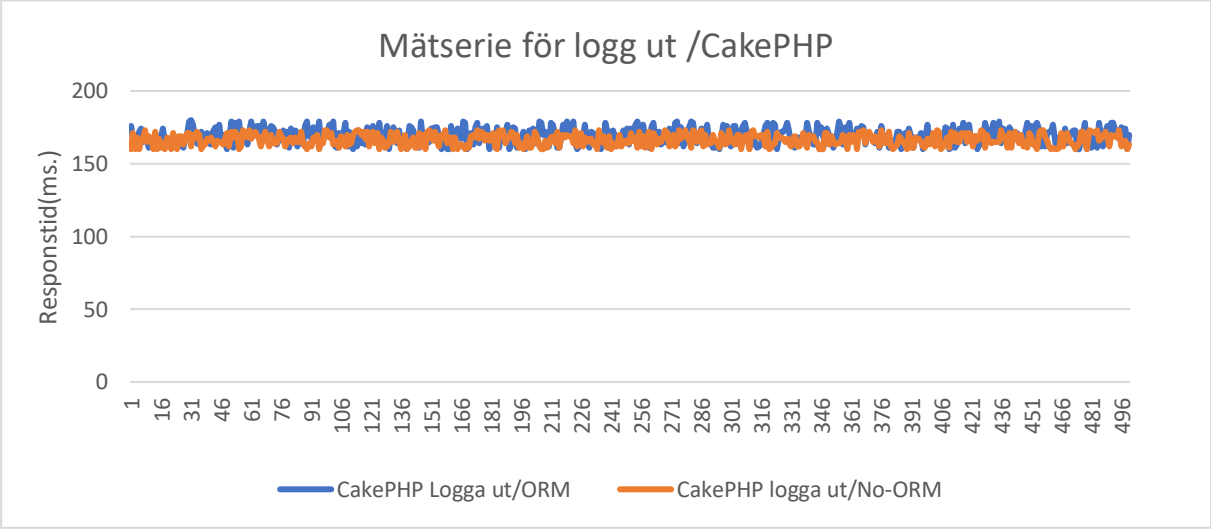
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	20034,58	1	20034,58	251,8219	9,71E-51	3,850793
Inom grupper	79399,41	998	79,55853			
Totalt	99433,98	999				

Tabell 2 ANOVAN test för logga in i CakePHP med/utan ORM



Figur 35 Medelvärde med standardavvikelse för logga ut CakePHP webbapplikation med och utan ORM.



Figur 36 Mätserie logga ut för både CakePHP med ORM och utan ORM

funktionalitet	CakePHP med ORM	CakePHP utan ORM
	Tid(ms.)	Tid(ms.)
Logga in	223,98	215,028
Logga ut	169,1	166,41

Tabell 3 Jämförande analys av prestanda efter tid

Anova Table

SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Cakephp/ORM	500	84550	169,1	31,49699
Cakeph/NoORM	500	83205	166,41	16,59108

ANOVA

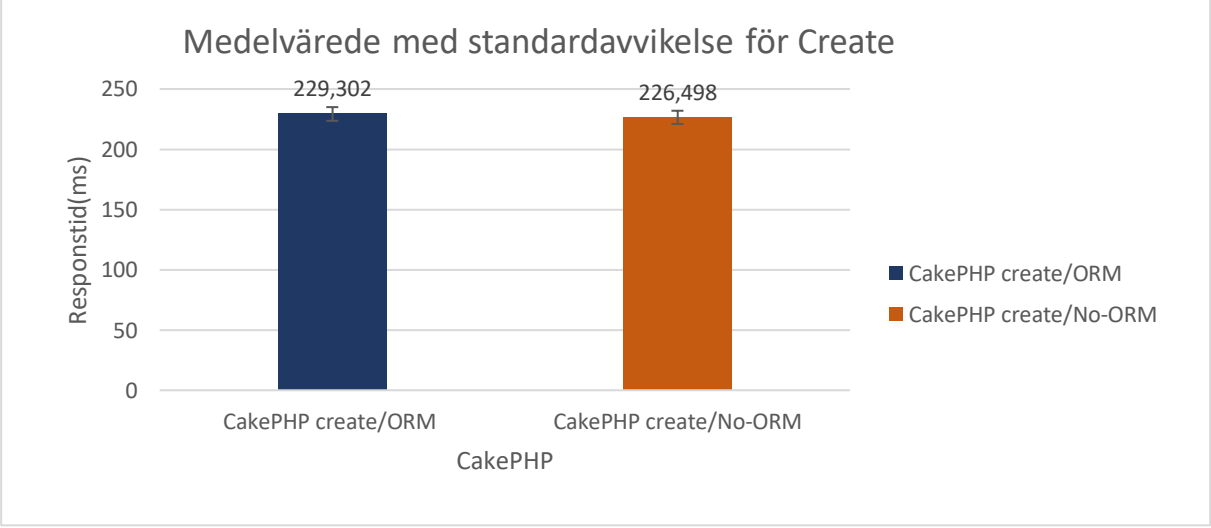
Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	1809,025	1	1809,025	75,23799	1,67E-17	3,850793
Inom grupper	23995,95	998	24,04404			
Totalt	25804,98	999				

Tabell 4 ANOVAN test för logga in i CakePHP med/utan ORM

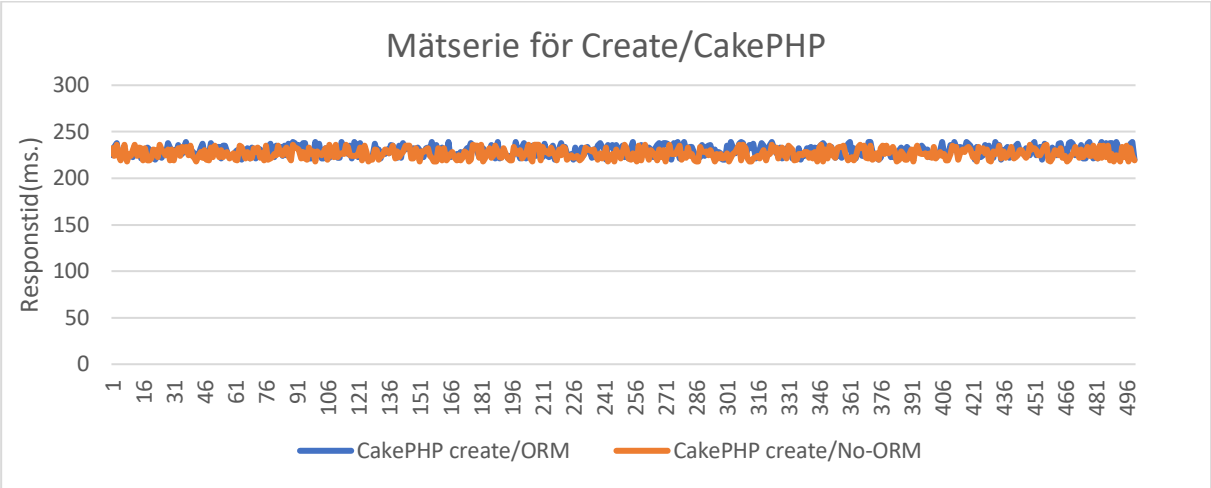
6.2.2 CakePHP Testfall 2-databasinteraktivitet (CRUD)

Under testfall 2 kommer vi mäta den tiden som CakePHP webbapplikation med model mapping (ORM) tar för att utföra databasinteraktivitet (CRUD), och sedan jämföras resultatet med tiden som CakePHP webbapplikation utan model mapping (ORM) tog för att utföra databasinteraktivitet (CRUD). Testet sker genom att genomföra 2000 testfall, dessa 2000 testfall består av 500 testfall INSERT, 500 testfall SELECT, 500 testfall UPDATE och 500 testfall DELETE. För att kunna mäta databasinteraktivitet (CRUD), skapas 500 post (kommentar). Dessa 500 poster består av tre olika långa texter, post som innehåller 10 ord, post som innehåller 50 ord och post som innehåller 100 ord. Vad som kommer mätas är den tiden som båda webbapplikationer tar för att utföra databasinteraktivitet (CRUD). Resultatet kommer visas i form av tabeller och grafer.

- För att mäta INSERT funktionen kommer skapas 500 post i databasen.
- För att mäta SELECT funktionen kommer hämtas 500 post från databasen.
- För att mäta UPDATE funktionen kommer ändras 500 post från databasen.
- För att mäta DELET funktionen kommer tas bort 500 post från databasen.



Figur 37 Medelvärde med standardavvikelse för Create i CakePHP webbapplikation med och utan ORM.



Figur 38 Mätserie Create för både CakePHP med ORM och utan ORM

Anova Table

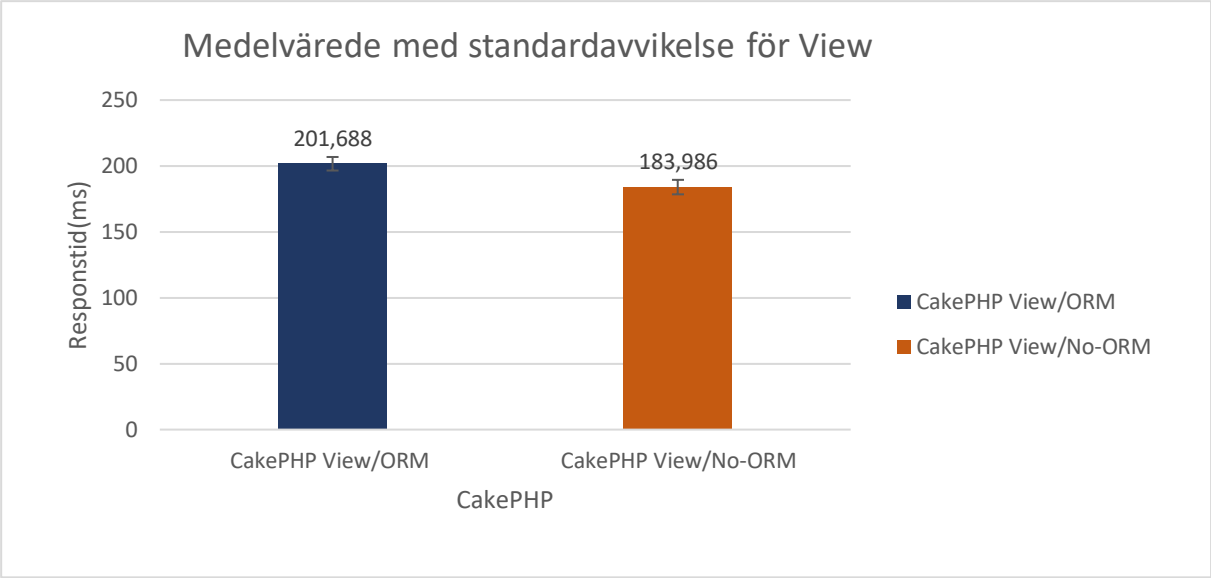
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Create/ORM	500	114651	229,302	32,41563
Create/NoORM	500	113249	226,498	30,23847

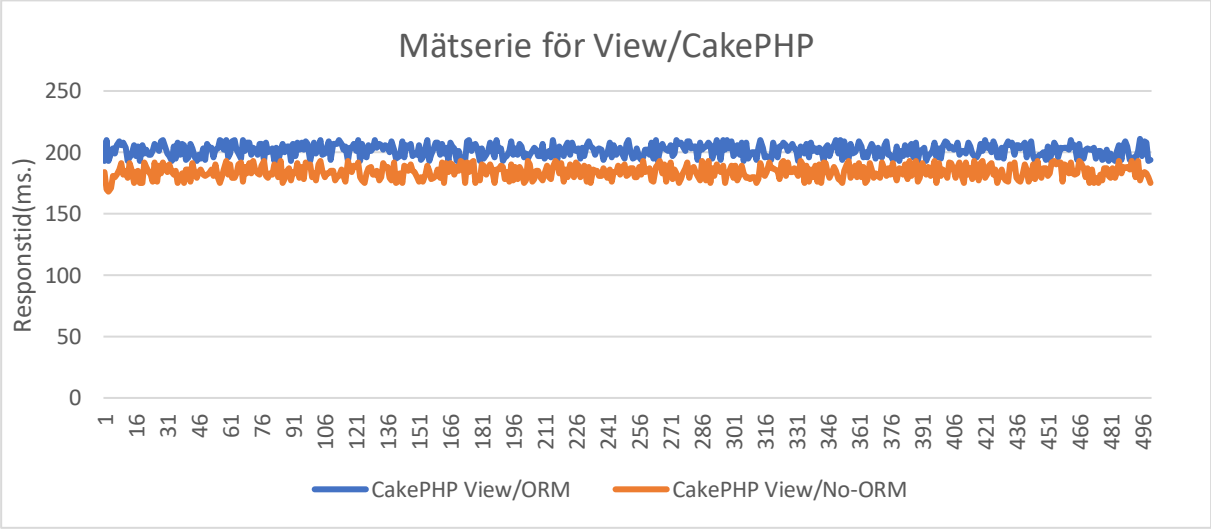
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	1965,604	1	1965,604	62,74462	6,25E-15	3,850793
Inom grupper	31264,4	998	31,32705			
Totalt	33230	999				

Tabell 5 ANOVAN test för Create i CakePHP med/utan ORM



Figur 39 Medelvärde med standardavvikelse för View i CakePHP webbapplikation med och utan ORM.



Figur 40 Mätserie View för både CakePHP med ORM och utan ORM

Anova Table

SAMMANFATTNING

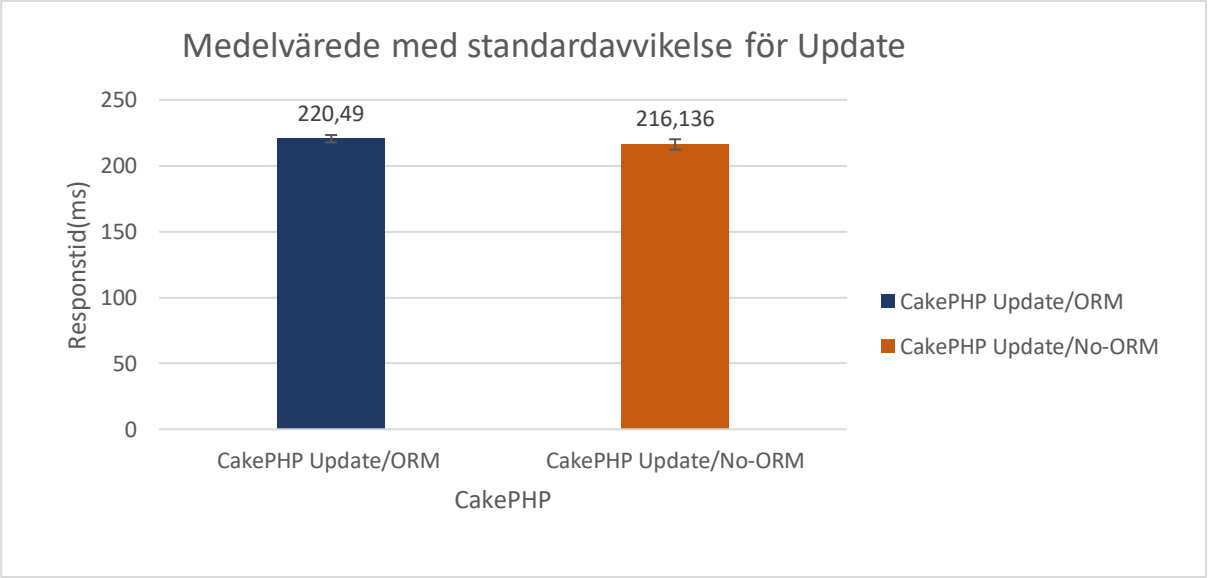
Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
View/ORM	500	100844	201,688	26,51168
View/NoORM	500	91993	183,986	30,12606

ANOVA

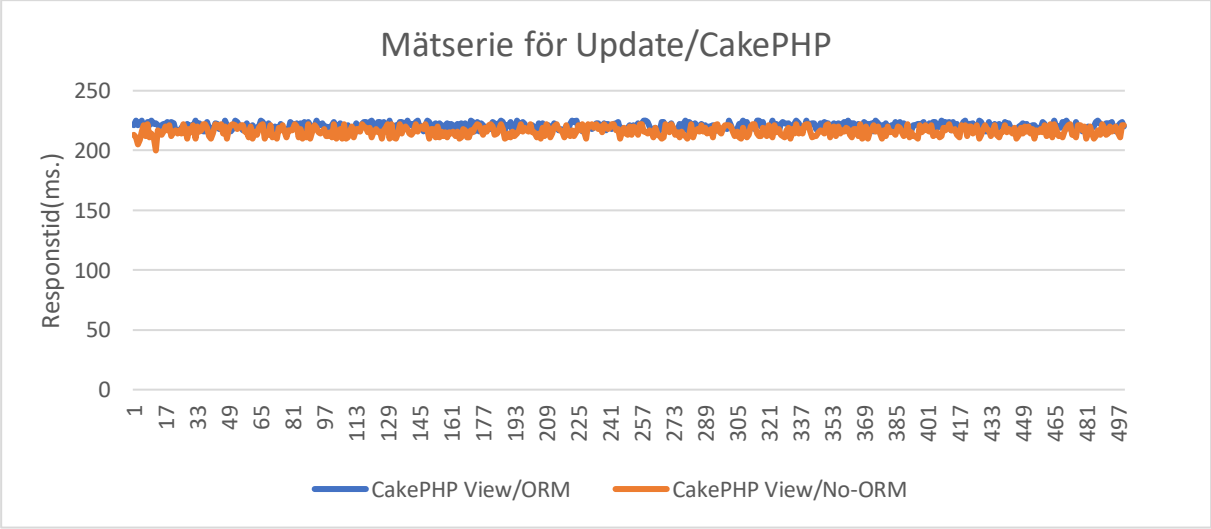
Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	78340,2	1	78340,2	2766,361	5,8E-290	3,850793
Inom grupper	28262,23	998	28,31887			

Totalt	106602,4	999	

Tabell 6 ANOVAN test för View i CakePHP med/utan ORM



Figur 41 Medelvärde med standardavvikelse för Update i CakePHP webbapplikation med och utan ORM.



Figur 42 Mätserie Update för både CakePHP med ORM och utan ORM

Anova Table

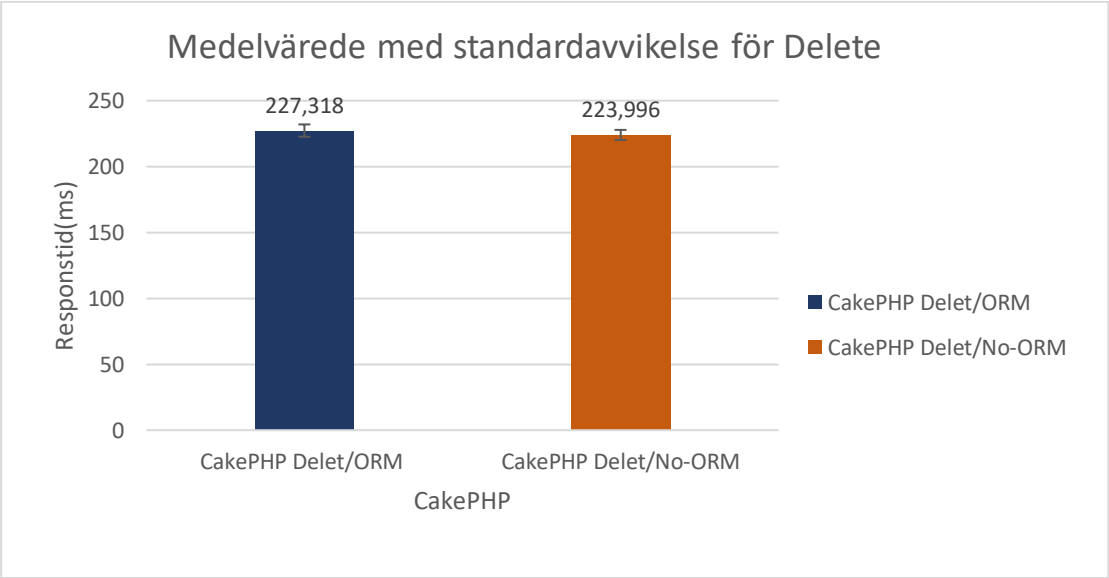
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Update/ORM	500	110245	220,49	7,829559
Update/NoORM	500	108068	216,136	15,42435

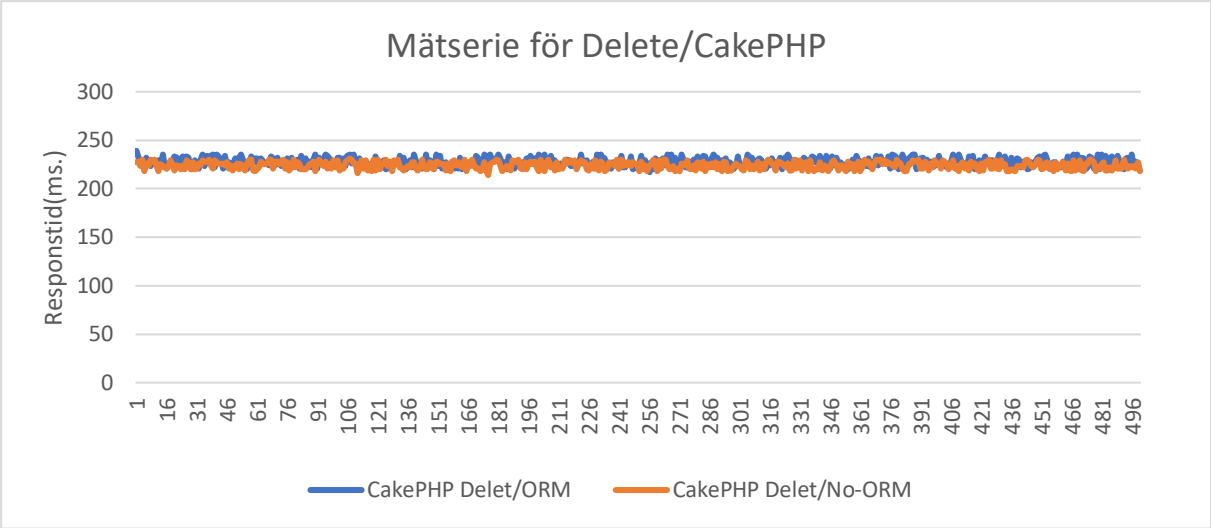
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	4739,329	1	4739,329	407,6156	2,82E-76	3,850793
Inom grupper	11603,7	998	11,62696			
Totalt	16343,03	999				

Tabell 7 ANOVAN test för Update i CakePHP med/utan ORM



Figur 41 Medelvärde med standardavvikelse för Delete i CakePHP webbapplikation med och utan ORM.



Figur 42 Mätserie Delete för både CakePHP med ORM och utan ORM

Anova Table

SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Delete/ORM	500	113659	227,318	21,97282
Delete/NoORM	500	111998	223,996	14,71341

ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	2758,921	1	2758,921	150,4063	2,63E-32	3,850793
Inom grupper	18306,43	998	18,34312			
Totalt	21065,35	999				

Tabell 8 ANOVAN test för Delete i CakePHP med/utan ORM

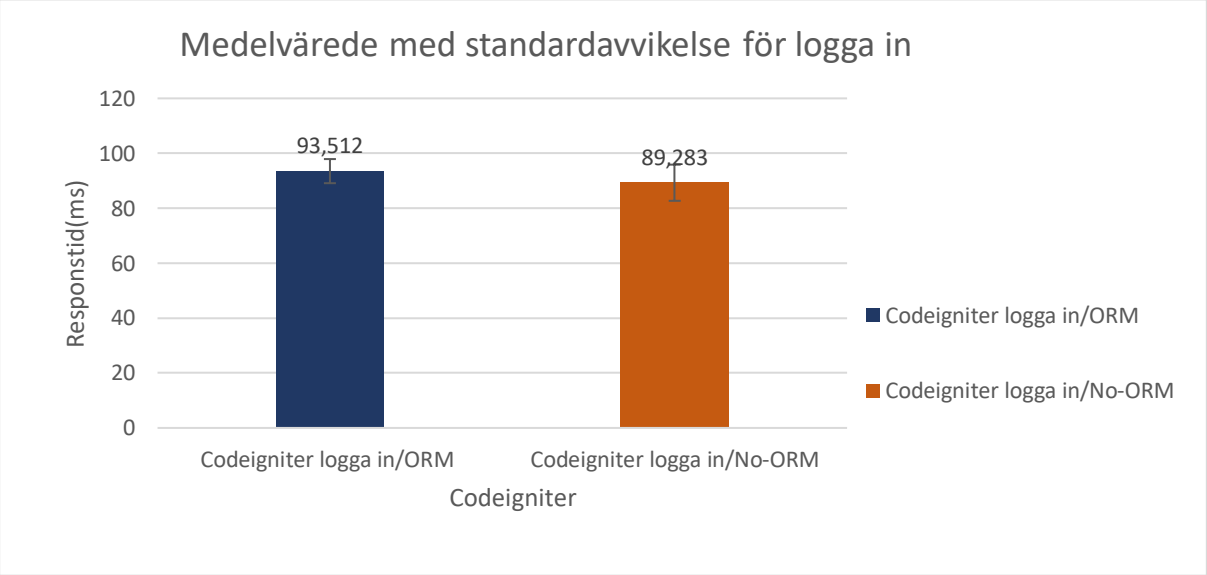
Funktionalitet	CakePHP med ORM	CakePHP utan ORM
	Tid(ms.)	Tid(ms.)
Create	229,302	226,498
View	201,688	183,986
Update	220,49	216,136
Delete	227,318	223,996

Tabell 9 Jämförande analys av prestanda efter tid

Enligt resultatet från CakePHP Testfall 1-inloggning/utloggning och Testfall 2-databasinteraktivitet (CRUD) som visar att det inte finns stor skillnad i medelvärde och det finns tydlig överlappning i standardavvikelserna, därför kan vi inte säkert säga att det finns en statistisk skillnad mellan CakePHP med aktiverad model mapping (ORM) och CakePHP inaktiverad model mapping (ORM). För att visa att det finns en statistisk skillnad mellan båda ramverken kommer en ANOVA-test att köras. Enligt resultatet från ANOVA-test visar att **P-value** är 0,0 vilket är lägre än 0.05. Eftersom **P-value** är mindre än 0.05 så vi avvisar H_0[11]. Resultat från ANOVA-test bekräftar att det finns en statistisk skillnad mellan båda ramverken. Se (tabeller 2,4,5,6,7 och 8) för mer förklaring.

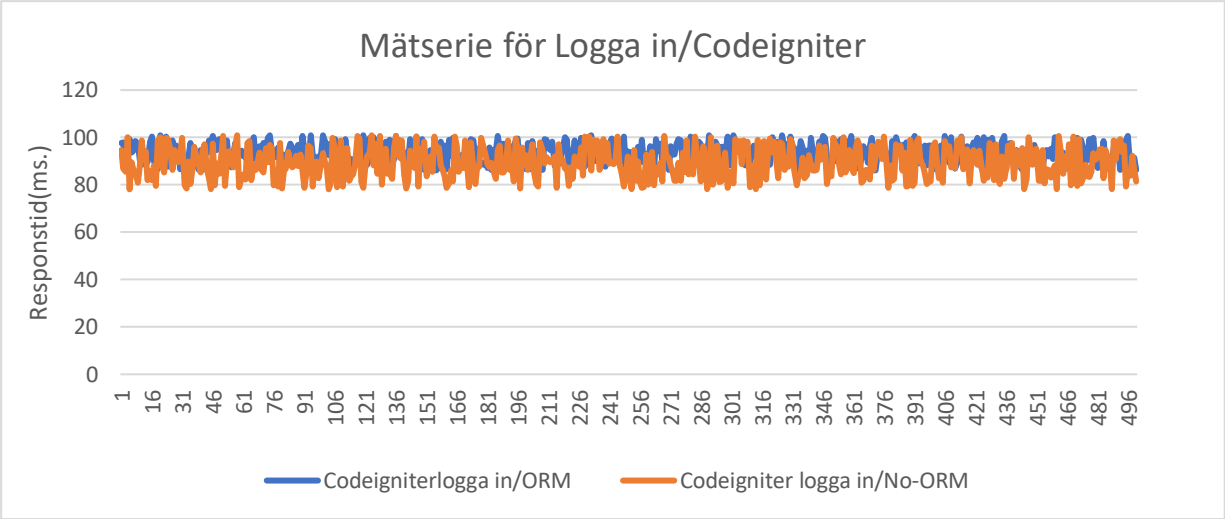
6.2.3 Codeigniter Testfall 3-inloggning och utloggning

Under testfall 3 kommer vi att upprepa testfall 1 men istället kommer vi använda webbapplikationer som skapads med Codeigniter ramverken. Här kommer vi också se i fall Codeigniter utan model mapping (ORM) kan ha mindre responstid än webbapplikationer som skapades med Codeigniter utan model mapping (ORM). Resultatet kommer visas i form av tabeller och grafer.



Figur 43 Medelvärde med standardavvikelse för logga in i Codeigniter webbapplikation med och utan ORM.

[11] Null Hypothesis(Ho): $\mu_1=\mu_2$



Figur 44 Mätserie logga in för både Codeigniter med ORM och utan ORM

Anova Table

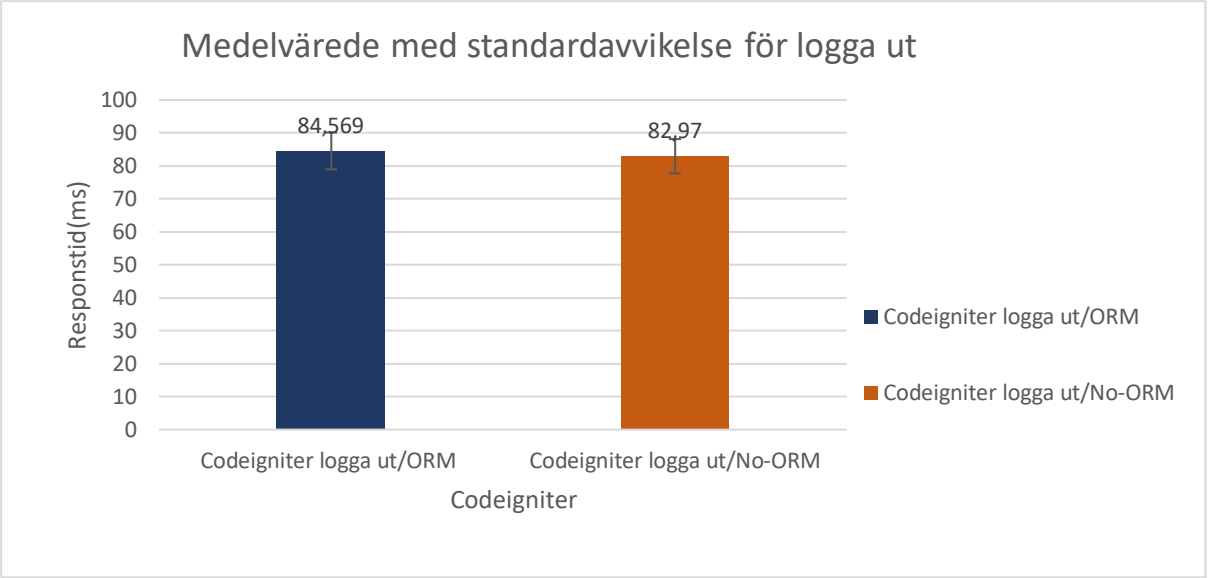
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
logga in/ ORM	501	46850	93,51297	19,30233
logga in/NoORM	501	44730,8	89,28303	43,38845

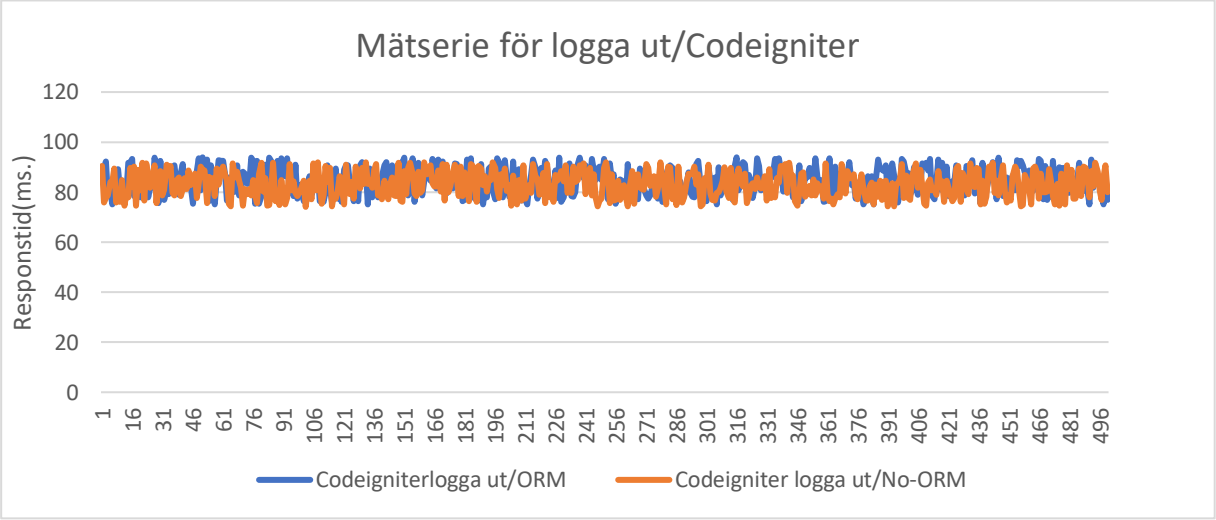
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	4482,045	1	4482,045	142,9889	6,75E-31	3,850775
Inom grupper	31345,39	1000	31,34539			
Totalt	35827,44	1001				

Tabell 10 ANOVAN test för logga in i Codeigniter med/utan ORM



Figur 45 Medelvärde med standardavvikelse för logga ut i Codeigniter webbapplikation med och utan ORM.



Figur 46 Mätserie logga ut för både Codeigniter med ORM och utan ORM

Anova Table

SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
logga ut/ ORM	501	42369,2	84,56926	31,30777
logga ut/NoORM	501	41568,4	82,97086	27,06959

ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*

Mellan grupper	640,0006	1	640,0006	21,92633	3,22E-06	3,850775
Inom grupper	29188,68	1000	29,18868			
Totalt	29828,68	1001				

Tabell 11 ANOVAN test för logga ut i Codeigniter med/utan ORM

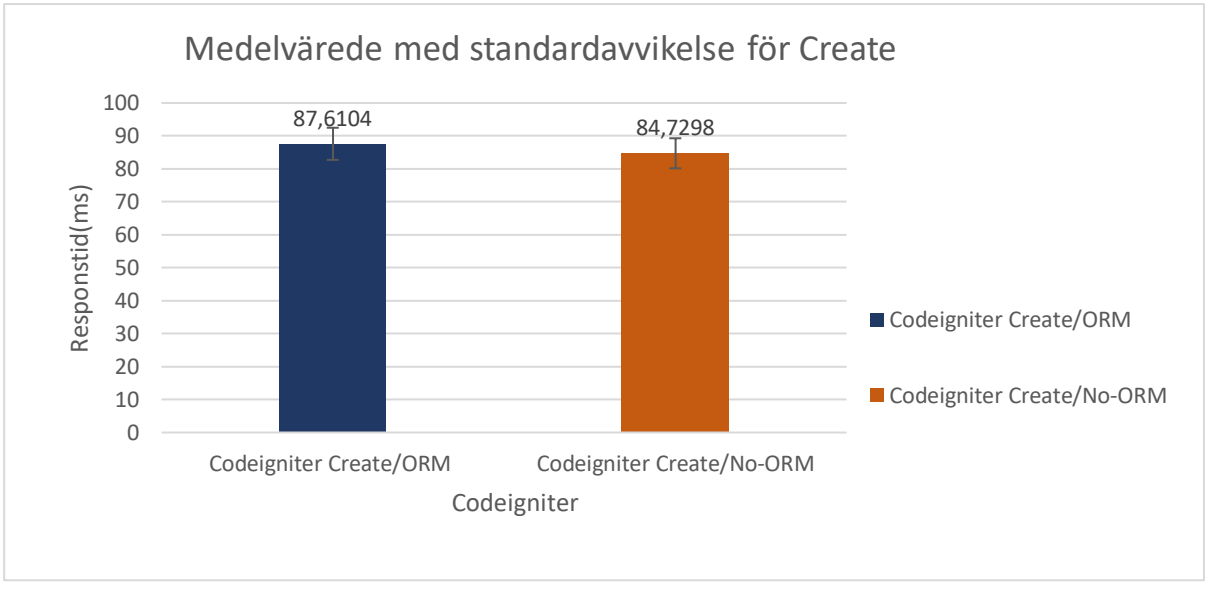
funktionalitet	Codeigniter med ORM	Codeigniter utan ORM
	Tid(ms.)	Tid(ms.)
Logga in	223,98	215,028
Logga ut	84,569	82,97

Tabell 12 Jämförande analys av prestanda efter tid

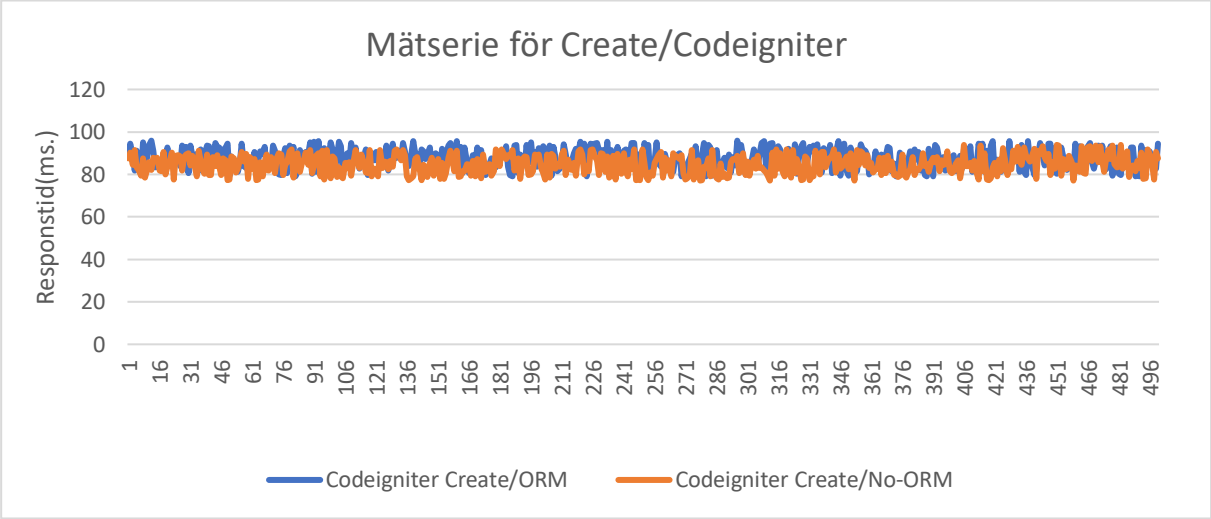
6.2.4 Codeigniter Testfall 4-databasinteraktivitet (CRUD)

För att kunna mäta databasinteraktivitet (CRUD), kommer testfall 2 att upprepas. I testfall 4 kommer jag använda samma strategi som användes i testfall 2, men istället kommer webbapplikationer som skapades med Codeigniter ramverken att användas. Resultatet kommer att visas i form av tabeller och grafer.

- För att mäta INSERT funktionen kommer skapas 500 post i databasen.
- För att mäta SELECT funktionen kommer hämtas 500 post från databasen.
- För att mäta UPDATE funktionen kommer ändras 500 post från databasen.
- För att mäta DELETE funktionen kommer tas bort 500 post från databasen.



Figur 47 Medelvärde med standardavvikelse för Create i Codeigniter webbapplikation med och utan ORM.



Figur 48 Mätserie Create för både Codeigniter med ORM och utan ORM

Anova Table

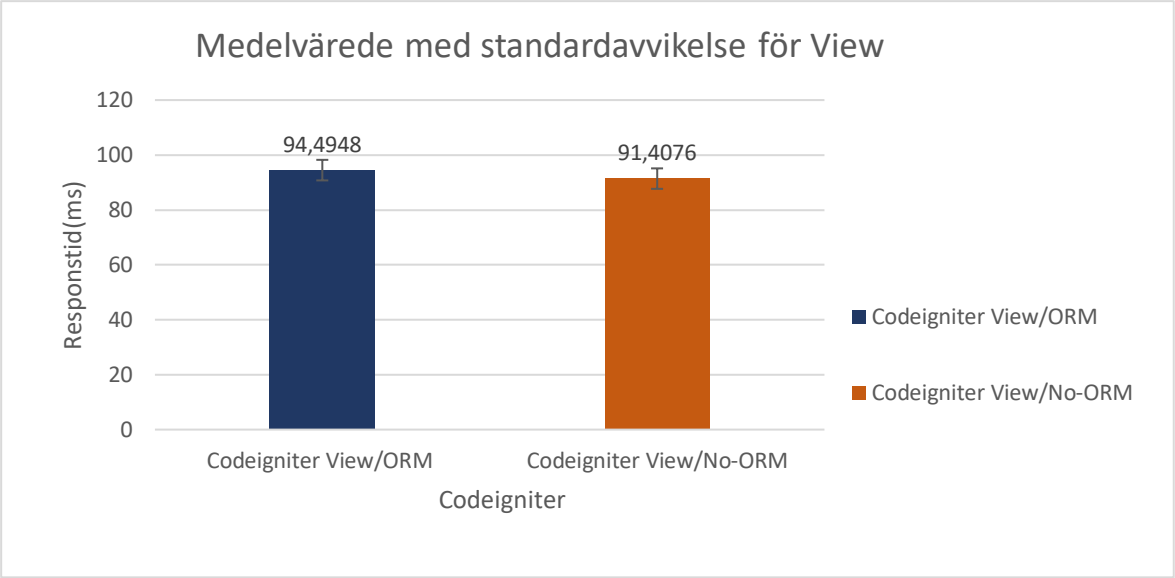
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Create/ORM	500	43805,2	87,6104	23,79256
Create/NoORM	500	42364,9	84,7298	20,79624

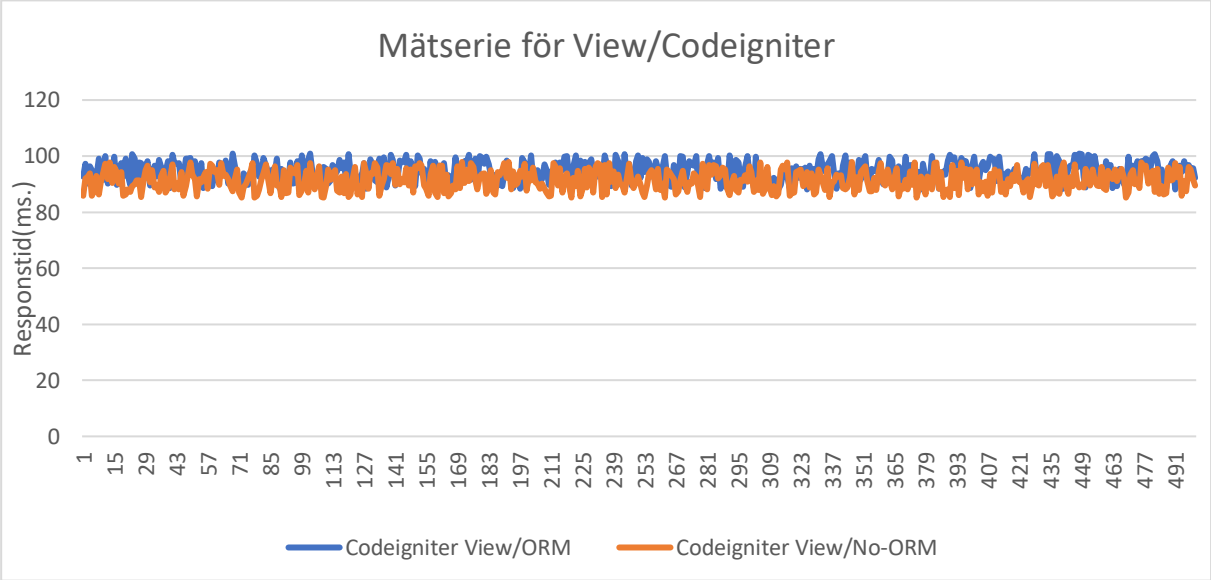
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	2074,464	1	2074,464	93,04866	4,11E-21	3,850793
Inom grupper	22249,81	998	22,2944			
Totalt	24324,28	999				

Tabell 13 ANOVAN test för Create i Codeigniter med/utan ORM



Figur 49 Medelvärde med standardavvikelse för View i Codeigniter webbapplikation med och utan ORM.



Figur 50 Mätserie View för både Codeigniter med ORM och utan ORM

Anova Table

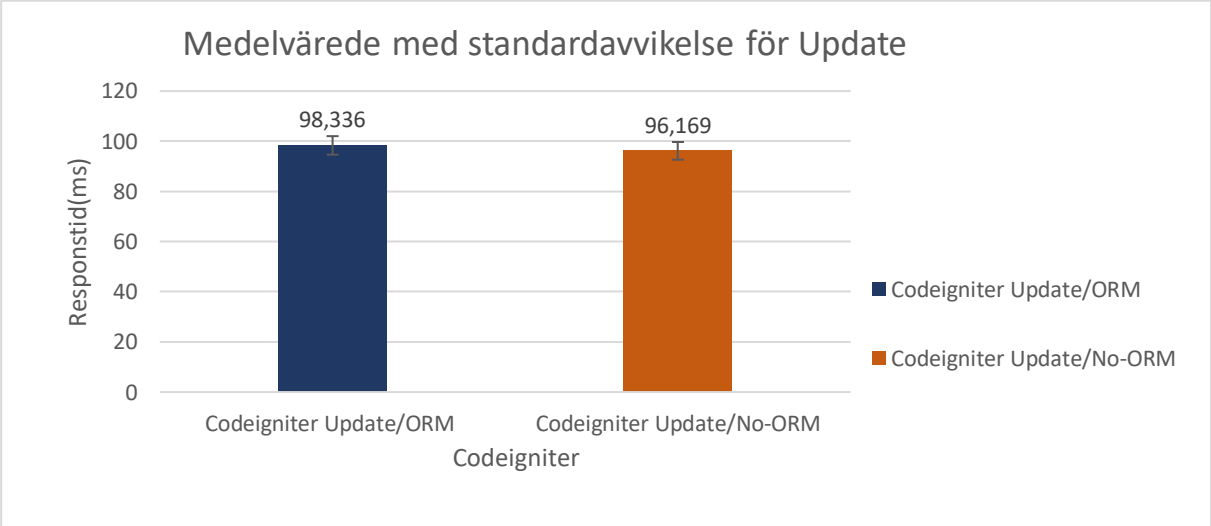
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
View/ORM	500	47247,4	94,4948	14,02739
View/NoORM	500	45703,8	91,4076	13,88567

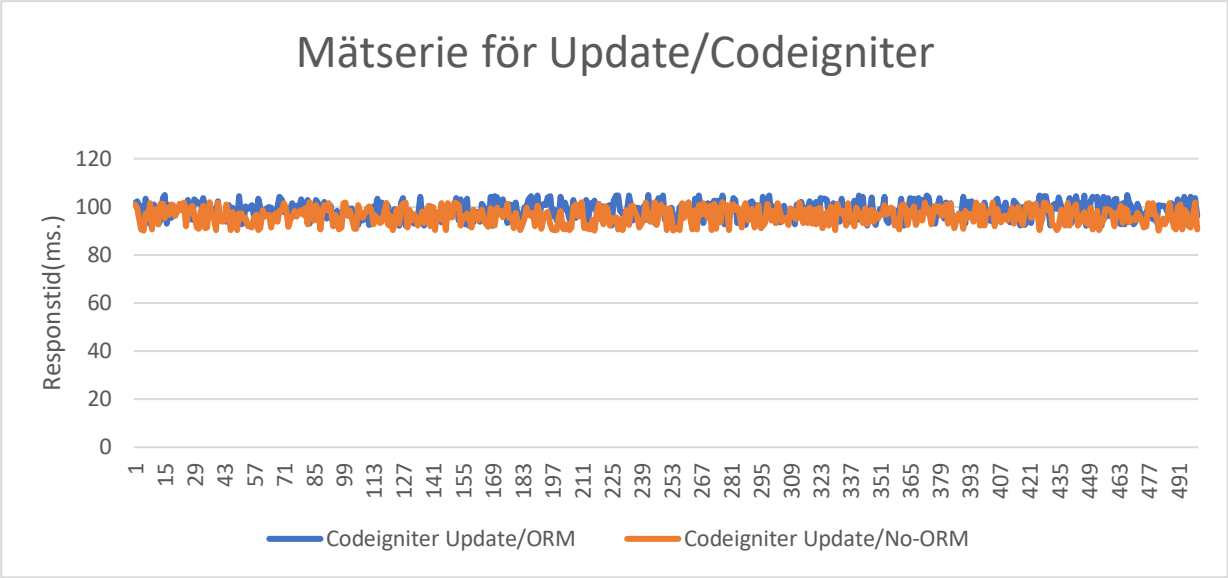
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	2382,701	1	2382,701	170,723	3,94E-36	3,850793
Inom grupper	13928,62	998	13,95653			
Totalt	16311,32	999				

Tabell 14 ANOVAN test för View i Codeigniter med/utan ORM



Figur 51 Medelvärde med standardavvikelse för Update i Codeigniter webbapplikation med och utan ORM.



Figur 52 Mätserie Update för både Codeigniter med ORM och utan ORM

Anova Table

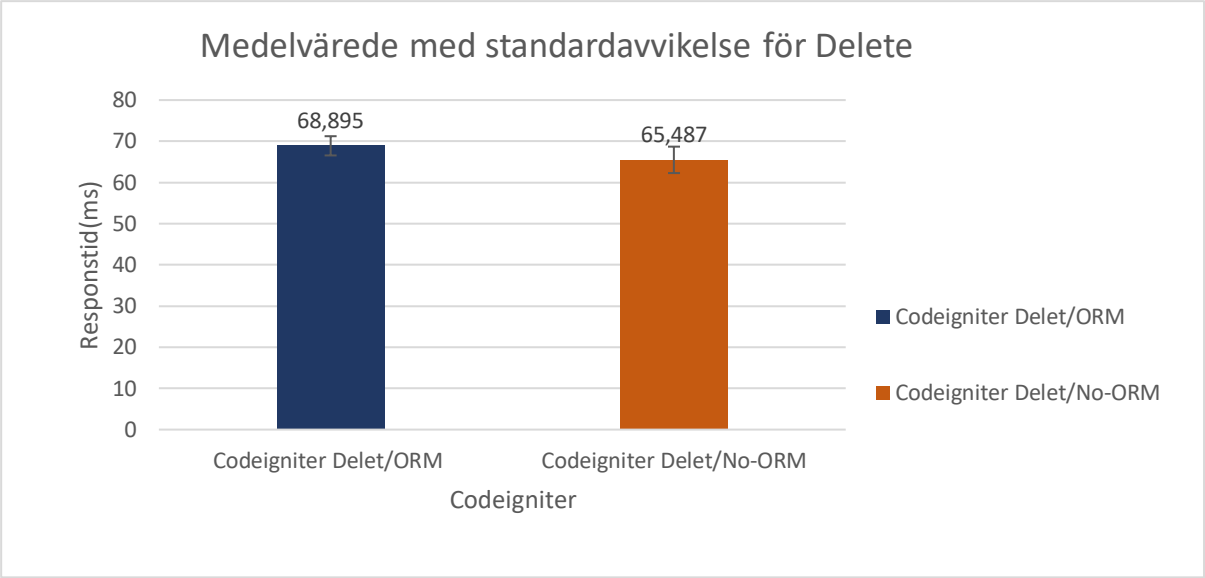
SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Update/ORM	500	52863,8	105,7276	6827,436
Update/NoORM	500	48084,5	96,169	12,44759

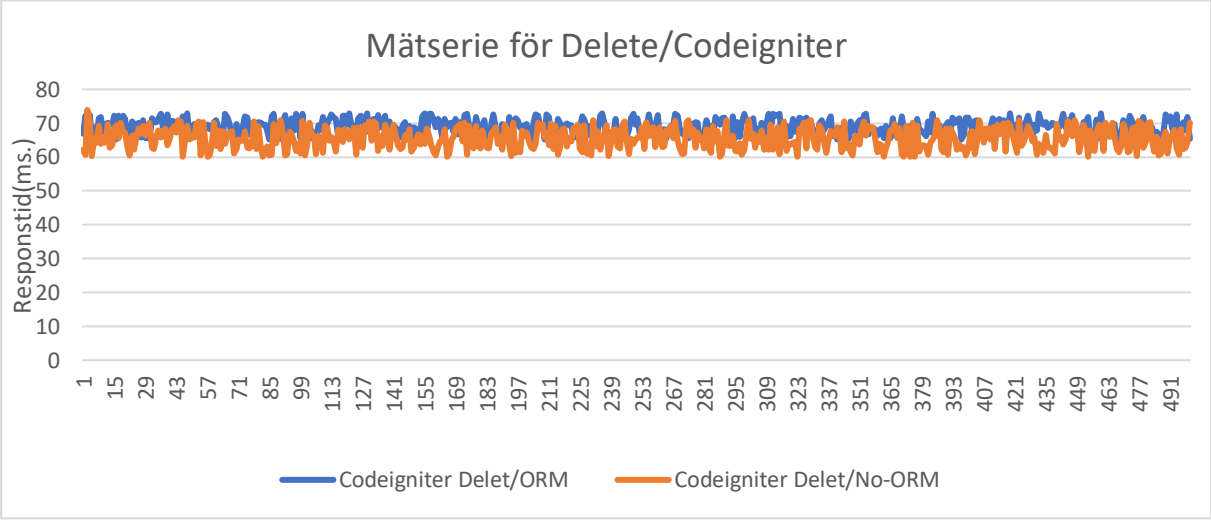
ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	22841,71	1	22841,71	6,678976	0,009897	3,850793
Inom grupper	3413102	998	3419,942			
Totalt	3435943	999				

Tabell 15 ANOVAN test för Update i Codeigniter med/utan ORM



Figur 53 Medelvärde med standardavvikelse för Delete i Codeigniter webbapplikation med och utan ORM.



Figur 54 Mätserie Delete för både Codeigniter med ORM och utan ORM

Anova Table

SAMMANFATTNING

Grupper	*Antal*	*Summa*	*Medelvärde*	*Varians*
Delete/ORM	500	34447,5	68,895	5,45751

Delete/NoORM	500	32743,5	65,487	10,29729

ANOVA

Variationsursprung	*KvS*	*fg*	*MKv*	*F*	*p-värde*	*F-krit*
Mellan grupper	2903,616	1	2903,616	368,6009	3,69E-70	3,850793
Inom grupper	7861,643	998	7,877398			
Totalt	10765,26	999				

Tabell 16 ANOVAN test för Delete i Codeigniter med/utan ORM

Funktionalitet	Codeigniter med ORM	Codeigniter utan ORM
	Tid(ms.)	Tid(ms.)
Create	229,302	226,498
View	201,688	183,986
Update	220,49	216,136
Delete	227,318	223,996

Tabell 17 Jämförande analys av prestanda efter tid

Enligt resultatet från Codeigniter Testfall 3-inloggning och utloggning och Testfall 4-databasinteraktivitet (CRUD) som visar att skillnaden är obetydligt stort i medelvärde och det finns tydlig överlappning i standardavvikelserna, därför kan man inte säga att det finns en statistisk skillnad mellan Codeigniter med aktiverad model mapping (ORM) funktionen och Codeigniter inaktiverad model mapping (ORM) funktionen. En ANOVA-test kommer att köras, resultatet från ANOVA-test bekräftar att det finns en statistisk skillnad mellan båda ramverken, enligt resultatet från ANOVA-test visar att **P-value** är 0,0 vilket är lägre än 0.05. Eftersom **P-value** är mindre än 0.05 så vi avvisar H_0[12]. se (tabeller 11,13,14,15, och 16) för mer förklaring.

[12] Null Hypothesis(Ho): $\mu_1=\mu_2$

7 Slutsats

7.1 Resultatsammanfattning

Som jag nämnde tidigare under problemformulering kommer detta studie att fokusera på ifall model mapping (ORM) påverkar responstid hos webbapplikationen som skapades med CakePHP och Codeigniter ramverken, baserat på testfallen som utfördes under pilotstudie och utvärdering, så resultatet visade att webbapplikationer som byggdes utan model mapping (ORM) har lägre responstid än webbapplikationer med model mapping (ORM) vid inloggning, utloggning och databasinteraktivitet (CRUD) funktionalitet.

Baserad på resultatet vi fått då kan vi svara på våran frågeställningar och hypoteserna. Nedan finns hypoteserna:

3- PHP-ramverken CakePHP och Codeigniter kommer ha kortare responstid utan model mapping (ORM).
4- Användning av model mapping (ORM) kommer inte påverka responstiden hos CakePHP och Codeigniter ramverken.

webbapplikation utan model mapping (ORM), grundat på mätningar, tabeller, grafer och ANOVA, kan ha kortare responstid vilket innebär hypotes nummer 1 är bekräftad. Resultatet visade också att det finns skillnad mellan ramverk med model mapping (ORM) och ramverk utan model mapping (ORM) när det gäller responstid. Därför svaret på frågeställningar blir ja, overhead minskar om man bygger applikation utan model mapping (ORM), eftersom det finns ett extra lager mellan applikationen och databasen.

När det gäller CakePHP och Codeigniter, då resultatet stämmer med tidigare experimentet från Kathas och Lancors (2013), där deras resultatet visade att CakePHP och Codeigniter tog längre tid för att ramverken har ett extra lager mellan applikationen och databasen.

7.2 Diskussion

Studien provar se ifall model mapping (ORM) kan påverka responstid hos applikationer genom att testa några funktionaliteter. Resultatet visade att webbapplikationer utan model mapping (ORM) har lägre responstid när det gäller inloggning, utloggning och databasinteraktivitet (CRUD) funktionalitet. Mätningar som tagit under experimentet visade att det finns skillnad mellan applikationer som skapas utan model mapping (ORM), men det skiljer bara några millisekunder vilket som inte kommer att märkas av användaren. En fråga kan dykas upp är det värt att vi undviker fördelarna som model mapping (ORM) kan bjuda bara för att vinna några millisekunder responstid som användaren kan inte ens observera den, eller spel den någon stor roll under verklighetsmiljö. Resultatet i denna studie kommer från en kontrollerad miljö, det innebär att vi utesluter andra faktorer som kan komma under verklighetsmiljö som påverkar mätningar på ett negativt eller positivt sätt. Det är också bra att hänvisa att data som används inför testning var relativt begränsat, därför blir det svårt att säga dessa resultat kommer gäller alltid. Det sista som kan diskuteras är ett ramverk består ofta av färdiga kodmoduler som kan bjuda lägre tid och mindre antal kod rader, vilket gör att utvecklaren inte behöva tänka på säkerhet. Genom att ändra eller ta bort funktionalitet som varje ramverk bjuder kommer vi påverka negativt på applikationen. I experimentet skiljas endast få millisekunder vilket är inte värt att vi tar risken och tar bort model mapping (ORM) från applikationen.

7.3 Etik och Samhälle

För att inte ska hamna i ett etiks problem kommer informera alla personer som deltagen för att testa webbapplikationer hur testen sker i detta studie. För att kunna testa webbapplikationer i detta experiment så måste testpersoner skapa ett konto (användarnamn och lösenord). När det gäller data som används för att testa inloggning /utloggning och databasinteraktivitet (CRUD) funktioner, den ska inte läggas under etiskt problem eftersom alla data under experimentet vi använder är på hittade data, det innebär vi kommer inte använda verkliga personers data, då dessa data är vanliga att använda och de betyder inte något speciellt och vi inte behöver upphovsrätt vid publicering. För att skydda testpersoner i fall de lämnar verkliga information vid registrering används hashing funktion för att skydda lösenord, se appendix A. Experimentera kommer inte direkt testas på människor eller djur detta innebär att inga människor eller djur skadas under denna studie.

Ett annat etiskt problem måste tänkas på är licens hantering. Det innebär att ha koll på vart man hämtar mjukvara från internet. Jag kan säkert sägas att de hård- och mjukvara som används för att utföra experimentet är korrekt licensierad. Mjukvaran som ska användas är öppen källkod, då man har tillåtelse att använda mjukvaran. Denna studie är publicerad i GitHub, så vid upprepning av andra utvecklaren behövs det ingen upphovsrätt. Studie kommer inte heller samla känsliga data om testpersoner därför Jag kan säkert säga inga personens integritet kommer att kränkas.

För att kunna upprepa experimentet då kommer publicera alla hård- och mjukvara som används under studie samt allt webbapplikationskodprogram. Enligt Cai, Nerurkar och Wu (1998), dessa informationer måste finnas tillgängliga ifall andra utvecklaren vill upprepa experimentet i framtiden. Fördelen med publicering är den kan hjälpa andra utvecklaren att få en hel syn på vad som skedde under experimentet och vad kan undvikas eller förbättras.

Eftersom tiden för detta experimentet var begränsad, så kunde inte genomföra experimentet på olika hård- och mjukvara. Resultatet i detta studie gäller bara den hård- och mjukvara som jag använde under experimentet, kan inte säga resultatet kommer bli inte samma ifall bredbandshastighet, webbserver, operativsystem eller hårdvara ska förändras.

När det gäller samhället så detta experiment kan vara användbar för andra utvecklaren som är intresserade i responstids område. Det kan också vara en vägvisare ifall någon vill undersöka model mapping (ORM). Jag kan inte nämna vilket område kan vi utnyttja av detta studie, vilket gör det svårt att nämna ett exempel. Generellt sätt studie kan vara användbar eftersom den ger översikt om vad kan påverka responstid.

7.4 Framtida arbete

Studie har många aspekter att undersöka i framtid. En av dessa aspekter är att kolla ifall cachning kan påverka responstid. Lösenord i detta experiment använder hashing vid sparning av data, det skulle vara bra att kolla ifall hashing kan också påverka responstid. I tidigare experimentet från Kathas och Lancors (2013), har de testad formulärhantering, formulärvalidering och sessionshantering, på grund av tidsram att hålla sig inom kunde inte undersöka dessa funktionaliteter under experiment ifall kommer påverkas av model mapping (ORM). Webbapplikationer i detta experimentet stöd för olika webbservrar, det skulle vara också bra om man ska undersöka ifall webbserver kan påverka responstid hos webbapplikationer. Det jag vill göra i framtiden är att installera båda webbapplikationer i en separat server där kommer fler än en klient som skickar förfrågan till server och undersöka ifall detta kan också påverka responstid.

Det hade varit väldigt intressant att undersöka säkerhet hos båda webbapplikationer med/utan model mapping (ORM) ifall aktivering eller Inaktivering model mapping (ORM) kan påverka säkerhet och prestanda, genom att använda alla säkerhetsverktyg som SQL-injektion och Cross site scripting.

Referenser

Abramova, V. & Bernardino, J. (2013) NoSQL Databases: MongoDB vs Cassandra. Proceedings of the International C* Conference on Computer Science and Software Engineering. C3S2E'13. New York, NY, USA, ACM. s. 14–22.

Ali Raza, F. Madiha, M. 2014 "Perfomance Evalution of PHP Frameworks (CakePHP and CodeIgniter) in relation to the Object-Relational-Mapping, with respect to Load testing)

Ahsanul, B. & Anupom, S. (2008). CakePHP Application Development: Packt Publishing Ltd

Alor-Hernández, G., Cuadrado, J.L.L., Rodríguez-González, A., Rodríguez-Mazahua, L., Salas-Zárate, M.d.P. & Valencia-García. R. (2015). Analyzing best practices on Web development frameworks: The lift approach. I Science of Computer Programming. Volym 102, ss.1–19. DOI: https://doi.org/10.101/j.scico.2014.12.004.

Biermann, M. (2014). A simple versatile solution for collecting multidimensional clinical data based on the CakePHP web application framework. *Computer Methods and Programs in Biomedicine*, 114 (1), ss.70–79.

British Columbia Institute of Technology. Why CodeIgniter? Tillgänglig på Internet: https://codeigniter.com/user_guide/installation/index.html].

Cai J-Y., Nerurkar A. & Wu M-Y. (1998). Making benchmarks uncheatable. *Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International*. Druham, USA 9 september 1998, ss. 216 - 226. 10.1109/IPDS.1998.707724

Cake Software Foundation (2020). Quick Start Guide. Tillgänglig på Internet: https://book.cakephp.org/4/en/quickstart.html [Hämtad 2020-03-03].

"Cakephp Cookbook. Debug kit," [Online]. Available: https://book.cakephp.org/debugkit/4/en/index.html .[Accessed 15 Mrs 2020].

Codeigniter Software Foundation (2020). Quick Start Guide. Tillgänglig på Internet: https://www.codeigniter.com/user_guide/intro/index.html [Hämtad 2020-03-03].

"Codeigniter. Benchmarking class," [Online]. Available: https://www.codeigniter.com/user_guide/intro/index.html.[Accessed 01 May 2020].

"Codeigniter. Benchmarking class," [Online]. Available: https://www.codeigniter.com/user_guide/testing/benchmark.html?highlight=benchmarking.[Accessed 01 May 2020].

Chowdhury, S. (2017). What is phpMyAdmin? Quora [blogg]. https://www.quora.com/What-is-phpMyAdmin [2020-03-07].

composer. (2018). Introduction. Gcomposer [blogg]. https://dbwebb.se/kunskap/installera-composer, hämtad [2020-03-07].

Das R., Saikia L. P. (2016). Comparison of Procedural PHP with Codeigniter and Laravel Framework. International Journal of Current Trend in Engineering & Research, vol. 2, no.6, pp. 42-48.

Denscombe, M. (2014). The Good Research Guide: For Small-scale Research Projects. 5. upp., Maidenhead, Berkshire: McGraw-Hill Education

DiCesare, C. & Hoxmeier, J.A. (2000). System Response Time and User Satisfaction: An Experimental Study of Browser-based Applications. I AMCIS 2000 Proceedings.

Dominik, L & contributors, R. (2019). Data Mapper. https://designpatternsphp.readthedocs.io/en/latest/Structural/DataMapper/README.html , hämtad [2020-05-021].

Eliahou Orr, Yehuda Zadik (2013). Programming with CodeIgniterMVC.

Fernandes, J., Oliveira, P., Silva, C. & Marcelino, L. (2012). Route Social Network. *Procedia Technology*, 5, ss.547–555.

Foster, R. (2013). CodeIgniter 2 Cookbook. Birmingham: Packt Publishing Ltd

Foster, R. (2015). CodeIgniter Web Application Blueprints. Birmingham: Packt Publishing Ltd.

Ghosh, P., Rau-Chaplin, A., (2006). Performance of Dynamic Web Page Generation for Database-driven Web Sites, in: International Conference on Next Generation Web Services Practices, 2006. NWeSP 2006. Presented at the International Conference on Next Generation Web Services Practices, 2006. NWeSP 2006, pp. 56–63. M. Fowler, ”Patterns of Enterprise Application Architecture”, Addison-Wesley Professional, 2002.

Ireland, C., Bowers, D., Newton, M., Waugh, K., 2009. A Classification of Object-Relational Impedance Mismatch, in: First International Conference on Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA’09. Presented at the First International Conference on Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA ’09, pp. 36–43.
Ishaq, J. & Hamed, H. (2016). Comparison of performance between Raw

SQL and Eloquent ORM in Laravel

James Mickens, Silo: exploiting JavaScript and DOM storage for faster page loads, Proceedings of the 2010 USENIX conference on Web application development, p.9–9, June 23–24, 2010, Boston, MA.

Katha, S. & Lancor, L. (2013). Analyzing PHP Frameworks for Use in a Project-Based Software Engineering Course. I SIGCSE '13, Proceeding of the 44th ACM technical symposium on Computer science education. Denver, Colorado, USA 6–9 mars 2013, ss.519–524. DOI: 10,1145/2 445 196 2 445 350.

Majida, L. Khaoula, B. Samira, K. Mohamed. K. (2019). A Comparative study of PHP frameworks performance. Volym 32, ss.864–871.

Matthew, C. Terry, H. (2010). The Norma Software Tool for ORM 2. (volume 72).

Natalya, P. & Victoria, B. (2017). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. Volym 104, ss.51–56.

Orr, E. & Zadik, Y. (2013). Programming with CodeIgniter MVC. Birmingham: Packt Publishing Ltd.

Oates, J. B. (2006). Researching Information Systems and Computing. London: SAGE Pulications Inc.

Prokofyeva, N. & Boltunova, V. (2017). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. I Procedia Computer Science. Volym 104, ss.51–56. Sciencedirect:

Peter, B., Kai, M., Günter, S. Handbook on Architectures of Information Systems: Springer Science & Business Media, 18 apr. 2006 - 896 sidor.

Luke Welling & Laura Thomson. 2003. PHP and MySQL Web Development.

Roopak, K.E., Rao, K.S.S., Ritesh, S., Chickerur, S., 2013. Performance Comparison of Relational Database with Object Database (DB4o), in: 2013 5th International Conference on Computational Intelligence and Communication Networks (CICN). Presented at the 2013 5th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 512–515.

Salas-Zárate, M.P., Alor-Hernández, G., Valencia-García. R., Rodríguez-Mazahua, L., Rodríguez-González, A., & López Cuadrado, J.L. (2015). Analyzing best practices on Web development frameworks: The lift approach. Science of Computer Programming, 102, ss.1–19.

Sandi. (2017). What is XAMPP and how to use it? Quora [blogg], 3 april. https://www.quora.com/What-is-XAMPP-and-how-to-use-it [2018-04-12].

Ushawu Ibrahim, J. B. Hayfron-Acquah. Frimpong, T. (2018). comparative analysis of Codeigniter and laravel in relation to object-relation mapping, load testing and stress testing. Volume 05 Issue 02.

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

Wegrzynowicz, P. (2013) Performance antipatterns of one to many association in hibernate. IEEE International. Krakoćw, Poland 11 september 2013.

Appendix A - user.php

```
<?php
namespace App\Model\Entity;

use Cake\ORM\Entity;
use Cake\Auth\DefaultPasswordHasher;

/**
 * User Entity
 *
 * @property int $id
 * @property string $name
 * @property string $email
 * @property string $password
 * @property \Cake\I18n\FrozenTime $created
 * @property \Cake\I18n\FrozenTime $modified
 *
 * @property \App\Model\Entity\Post[] $posts
 */
class User extends Entity
{
    /**
     * Fields that can be mass assigned using newEntity() or patchEntity().
     *
     * Note that when '*' is set to true, this allows all unspecified fields to
     * be mass assigned. For security purposes, it is advised to set '*' to false
     * (or remove it), and explicitly make individual fields accessible as needed.
     *
     * @var array
     */
    protected $_accessible = [
        'name' => true,
        'email' => true,
        'password' => true,
```

```
        'created' => true,
        'modified' => true,
        'posts' => true,
    ];

    /**
     * Fields that are excluded from JSON versions of the entity.
     *
     * @var array
     */
    protected $_hidden = [
        'password',
    ];

    // function _setPassword för att skydda lösen ord
    protected function _setPassword($password){
        return (new DefaultPasswordHasher)->hash($password);
    }
}
```