Inrapporteringssystem av resultat från skriptbaserade mjukvarutest

CHRISTIAN SHEHADEH
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Inrapporteringssystem av resultat från skriptbaserade mjukvarutest

Av

Christian Shehadeh

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Sammanfattning

Syftet med examensarbetet har varit att utveckla en prototyp åt Region Kalmar läns IT-avdelning som ska möjliggöra inrapportering av testresultat från tester på mjukvara som utförts automatiskt med hjälp av skript (automatiska test). Med hjälp av skripten kan Region Kalmar läns IT-avdelning testa olika delar av den mjukvara som används i regionen utan att en testare manuellt behöver utföra testet.

Resultatet av examensarbetet är ett system som kan användas för att rapportera in resultat av utförda mjukvarutest, och förhoppningsvis underlätta arbetet för mjukvarutestare på Region Kalmar län. Prototypen består av tre delar. En databas, ett Rest API och ett webbgränssnitt.

Databasen är utvecklad med databashanteraren Microsoft SQL Server med hjälp av ramverket Laravel. Databasen används för att lagra resultaten av de utförda testen.

REST API:et är utvecklat i PHP med hjälp av Laravel. API:et tillhandahåller funktioner för att rapportera in testresultat till databasen.

Webbgränssnittet är utvecklat i PHP och JavaScript med Laravel. Gränssnittet använder sig av REST API:et för att visa och hantera inrapporterade testresultat.

Nyckelord: Test av mjukvara, API, Rest, PHP, SQL server, JavaScript, Laravel.

Abstract

The aim of the thesis has been to develop a prototype for Region Kalmar läns IT department that would allow reporting of results from tests that have been performed automatically using scripts. These scripts allow the IT department of Region Kalmar län to test different parts of the software that is used by the whole region without the need for a tester to manually perform the tests. The developed prototype developed during the thesis consists of three parts: A database, a REST API, and a web-based interface.

The result of the thesis is a system that can be used to report the results of testing of software, which would hopefully ease the workload on the testers of Region Kalmar län. The prototype consists of three parts. A database, a REST API and a web interface.

The database is developed with the database management system Microsoft SQL Server using the framework Laravel. The database is used to store the results of the test that has been performed.

The REST API is developed with PHP using the framework Laravel. The API allows results from tests to be stored in the database by providing the necessary functions to do so.

The web interface is developed with PHP and JavaScript using the framework Laravel. The interface uses the REST API to display and manage the results from the tests.

Keywords: Testing of software, API, Rest, PHP, SQL Server, Laravel.

Förord

Jag vill tacka Region Kalmar läns IT-avdelning för att jag fick möjligheten att utföra examensarbetet. Ett extra stort tack vill jag rikta till min handledare Magnus Karlsson som har varit till stor hjälp under examensarbetet.

Jag vill även tacka min handledare Christin Lindholm och min examinator Christian Nyberg på LTH.

Examensarbetet har varit väldigt givande och jag har lärt mig väldigt mycket under tiden det har pågått, speciellt inom webbutveckling både inom backend och frontend.

Christian Shehadeh

Innehållsförteckning

1. Inledning

I det här kapitlet kommer examensarbetet att introduceras. Bakgrunden, syftet, målen, problemformuleringar, motiveringen och avgränsningar kommer tas upp.

1.1. Bakgrund

Examensarbetet har utförts i samarbete med Region Kalmar läns IT-avdelning. IT-avdelningen ansvarar för att driva IT för hälso- och sjukvård, kollektivtrafik, bildningsverksamhet och regional utveckling inom Region Kalmar län. Region Kalmar län använder ett stort antal program i sin verksamhet, till exempel journalsystemet Cosmic. För många av de här programmen krävs det att tester utförs av IT-avdelningen inför varje ny utgåva av mjukvaran för att säkerhetsställa att programmen går att använda på ett korrekt sätt av de som använder sig av programmen i sitt arbete. För närvarande utförs testerna manuellt av testare på IT-avdelningen på Region Kalmar län. Den manuella testningen utförs för hand av en testare som rapporterar in resultatet i Region Kalmars inrapporteringssystem för tester. Ett flödesschema över hur ett vanligt manuellt test utförs illustreras i Fig. 1.

Sedan en tid tillbaka så har ett arbete pågått hos Region Kalmar läns IT-avdelning där en del av testningen skall utföras automatiskt. Detta arbete har resulterat i att kommandon/skript skapats som kan användas för att utföra test av mjukvara automatiskt utan att en testare manuellt utför testet. Dessa test kallas internt på Region Kalmar läns IT-avdelning för automatiska test. För närvarande finns det inget system hos Region Kalmar län för att rapportera in resultaten av de automatiska testen samtidigt som de utförs.

Examensarbetets syfte har varit att utveckla en prototyp bestående av en databas som kan lagra testresultat från utförda automatiska test, ett API som kan användas för att lagra testresultat i databasen samtidigt som testen utförs genom att kalla på API:et under exekvering av de

automatiska test skripten. Ett webbgränssnitt som kan visa resultaten av de utförda testen ska också ingå i prototypen.

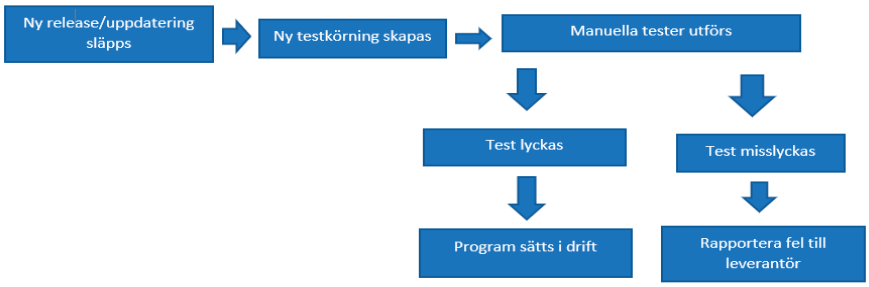


Fig. 1. Ett flödesschema över hur ett manuellt testfall vanligtvis utförs

1.2. Syfte

Syftet med examensarbetet är att utveckla en prototyp som ger testare på Region Kalmar län möjligheten att se resultatet av de automatiska tester som har utförts. Prototypen ska bestå av ett API för inrapportering av testresultat, en databas som lagrar information om testerna samt resultat av utförda tester och ett webbgränssnitt som visar information om testerna samt resultat av utförda automatiska tester.

1.3. Målformulering

Målen som examensarbetet ska uppnå består av:

Utveckling av en databas. I utvecklingen ingår en utvärdering av vilken information databasen ska lagra för att prototypen ska kunna hantera information om inrapporterade test på ett korrekt sätt.

Utveckling av ett API kopplat till databasen som möjliggör inrapportering av testfallsresultat. Man ska genom API:et kunna rapportera in resultat om de test man har utfört.

Utveckling av ett webbaserat gränssnitt som ska kunna visa resultat av utförda testerna. Gränssnittet ska använda sig av den utvecklade

databasen för att visa resultat av tester som utförts. I utvecklingen av prototypen ingår en utvärdering av vad gränssnittet ska innehålla.

Den utvecklade mjukvaran ska vara väl dokumenterad. Detta innebär en manual över API:et samt webbgränssnittet. Koden som utvecklats ska vara kommenterad så att det går att förstå den för någon som redan har programmeringserfarenhet. Användartest ska utföras för att säkerställa att mjukvaran är väl dokumenterad.

1.4. Problemformulering

Under examensarbetet skulle nedanstående frågor besvaras:

1. Vilken information om de automatiska testerna behöver lagras i databasen?

2. Vilka funktioner ska API: et tillhandahålla för att testare ska kunna rapportera in sina resultat?

3. Vilken funktionalitet ska webbgränssnittet ha för att möta de krav som ställs på den?

4. Vilka programspråk ska användas vid utvecklingen av prototypen?

5. Vilken typ av server ska användas för prototypen?

6. Vad ska manualen för API:et samt manualen för webbgränssnittet innehålla? Hur ska det avgöras om manualerna och kommentarerna i kod är tydliga nog för att kunna användas av testare samt vid en eventuell utveckling av prototypen efter examensarbetets slut?

1.5. Motivering av examensarbetet

Examenarbetet valdes eftersom examensarbetaren tyckte att det är ett examensarbete där många av de kunskaper examensarbetaren har fått under sin utbildning kommer till användning i praktiken. Det är också ett examensarbete där examensarbetaren kan få nya kunskaper inom utveckling av mjukvara, vilket examensarbetaren kommer ha användning av i sitt yrkesliv.

För Region Kalmar län kommer det slutförda examensarbetet innebära att testare kommer få en möjlighet att se resultaten av de automatiska test som har utförts. Detta kommer underlätta för testare på Region

Kalmar län att få en överblick över vilka automatiska tester som utförts och resultatet av dem. Detta kan därmed spara tid och underlätta testarnas arbete.

Examensarbetet skulle potentiellt kunna användas för utvecklingen av liknande system av andra personer. Till exempel skulle lärdomar som nås under utvecklingen av prototypen kunna användas i utveckling av ett annat system för att spara tid.

1.6. Avgränsningar

Examensarbetet innefattar inte att skapa de automatiserade testen. API:et skapas så att det går att skicka lämplig information till databasen, men hur detta API används är inte en del av examensarbetet.

Den utvecklade prototypen ska kunna användas på operativsystemet Windows 10 och på webbläsaren Microsoft Edge.

2. Teknisk Bakgrund

Detta kapitel beskriver de tekniker och mjukvara som har använts under examensarbetet.

2.1. Automatiska tester

Automatiska tester är vad de tester som har utvecklats av Region Kalmar läns IT-avdelning för att testa mjukvara med hjälp av skript kallas.

För nuvarande består testerna av små skript som kan köras i till exempel ett kommandotolkprogram. Vad testen testar är olika beroende på vilken mjukvara och vilken del av mjukvaran som ska testas, för en sorts mjukvara kan ett skript till exempel bestå i att en inloggningsknapp klickas genom att muspekaren rör sig till knappen för att sedan utföra ett klick medan i annat kan testet bestå av att ett program startas. Vanligtvis utgörs ett automatiskt test av flera av dessa skript, om ett automatiskt test ska till exempel logga in en användare så skulle det kunna ha följande skript i sig.

1. Skriv in inloggningsinformation
2. Klicka på knappen ”Logga in”

Skripten brukar därför kallas teststeg av Region Kalmar och samlingen med skript kallas testfall.

De automatiska testerna är utvecklade för att underlätta och spara tid för testare på Region Kalmar län genom att till exempel utföra repetitiva test med skript i stället för att testaren utför testet manuellt.

Examensarbetets mål har varit att utveckla ett rapporteringssystem för resultaten av de här testen som kan användas under exekveringen av de automatiska testen, ett skript som använder sig av examensarbetets prototyp skulle exempelvis kunna försöka logga in i en mjukvara genom att.

1. Skriva in inloggningsinformation.
2. Kontrollera om inloggningsinformation är korrekt, och rapportera in resultat (t.ex. Lyckat) till prototypen.
3. Klicka på knappen ”Logga in”.

4. Kontrollera om en lyckad inloggning har skett och rapportera in resultatet till prototypen.

2.2. REST (Representational State Transfer) och REST API

REST är en arkitekturstil inom it som kan användas för att genom HTTP (URL-adresser) modifiera resurser, resurser innebär någon typ av information till exempel en persons förnamn eller ett dokument.

En viktig del inom REST är tillståndslöshet vid HTTP-anrop, detta betyder att ett anrop från en klient till en server måste innehålla all data för att servern ska förstå anropet [9].

Genom http-metoder kan man ange vad man vill göra med en resurs vid ett anrop. GET hämtar resursen, POST skapar en resurs, PUT/PATCH uppdaterar en resurs och DELETE tar bort en resurs.

En annan del av REST arkitekturen är att vid svar på anrop så skickas HTTP-statuskoder för att meddela om huruvida anropet var lyckat eller ej [12].

Ett REST API är ett API som använder sig av REST-arkitekturen för att hantera resurser. Ett REST API skulle till exempel kunna användas för att i en databastabell som innehåller data om olika personer ändra en persons förnamn eller ta bort ett dokument från ett filsystem.

Examensarbetets prototyp består av ett REST API som lagrar data i prototypens databas. I prototypen representerats därför varje databastabell som en resurs.

2.3. JSON (JavaScript Object Notation)

JSON är ett textbaserat format som används för att representera data. JSON är utvecklat med tanken att vara enkel för människor att läsa och skriva samt enkelt för maskiner att tolka. Ett JSON-objekt består av en nyckel och ett värde. Nyckeln består alltid av en textsträng medan värdet kan vara av typen textsträng, nummer, bool, null, vektor eller JSON-objekt [2].

I examensarbetet används JSON när prototypens API returnerar ett svar vid ett anrop.

Ett exempel på hur ett JSON-format kan se ut visas i Fig.2.

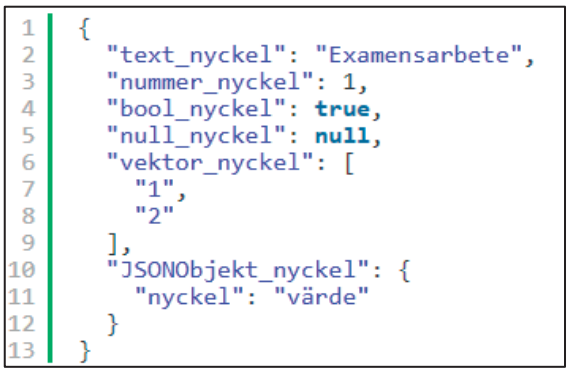


Fig. 2. Ett exempel på hur ett JSON objekt kan se ut. I figuren är nycklarna namngivna enligt formatet ”x_nyckel” där x anger typen av värdet.

2.4. SQL Server

SQL Server är en databashanterare som är utvecklad av Microsoft.

SQL Server gör det möjligt att lagra och hantera data i en databas genom användning av SQL. Microsoft erbjuder olika versioner av SQL Server [15].

Under examensarbetet har SQL Server använts som databashanterare till prototypens databas. Den kostnadsfria versionen SQL Server Express har använts under examensarbetet.

2.5. PHP (Hypertext Preprocessor)

PHP är ett programmeringsspråk som skapades av Rasmus Lerdorf 1994 och släpptes som öppen källkod 1995. PHP är ett programmeringsspråk som exekveras på en server och skickas därefter till klienten till skillnad från JavaScript som vanligtvis exekveras på klientens maskin. En av funktionerna som PHP tillhandahåller är att det går att inbädda PHP kod direkt i HTML-kod [1].

Under examensarbetet har prototypen utvecklats med PHP som språk på serversidan.

2.6. Node.js

Node.js är en exekveringsmiljö som används för att exekvera JavaScript på serversidan. Node.js behandlar anrop asynkront till skillnad från PHP som behandlar anrop synkront. Det innebär att Node.js inte behöver använda sig av trådar på servern för att hantera anrop vilket är det mer traditionella sättet som andra språk på serversidan använder sig av (t.ex. PHP) [16].

Under examensarbetet var Node.js ett av de två programspråk som examensarbetaren studerade närmare under examensarbetet som ett potentiellt språk att använda för prototypen på serversidan.

2.7. Laravel

Laravel är ett ramverk med öppen källkod baserat på PHP som är utvecklat med målet att underlätta utvecklandet av webbaserade applikationer [3].

Ramverket tillhandahåller flera klasser och funktioner som är inriktade på både backend och frontend utveckling. I examensarbetet har ramverket använts för att utveckla både API:et och webbgränssnittet. I delkapitlen 2.7.1 - 2.7.4 kommer de Laravel-klasser som examensarbetaren har använt under utvecklingen av prototypen att beskrivas.

Laravel innehåller ett objekt-rationellt mappningssystem (ORM-system). Det innebär att tabeller i en databas konverteras till klasser. En rad i en databastabell kan därmed representeras av en instans av en klass, detta system möjliggör hantering av tabeller i en databas utan att användaren behöver skriva egen SQL-kod [8].

Laravel innehåller ett databasmigreringssytem som tillåter utvecklare att bygga upp databastabeller. Systemet består av klasser som har två metoder:

- ”up” där vilka ändringarna som ska göras i databasen definieras
- ”down” som definierar vad som ska göras för att återgå till databasens tillstånd innan ”up” kördes.

Genom kommandon i Laravel kan man gå tillbaka till tidigare versioner av databasen genom kommandon som utför ”down” och man kan också använda kommandon för att köra ”up”. Detta gör att man kan versionshantera en databas [6].

Laravel har funktioner för att förenkla hanteringen av routing. Routing innebär att när ett HTTP-anrop kommer in till en applikation så hanteras anropet och en åtgärd utförs. Laravel använder sig av routing-filer för att hantera routing. I routing-filerna definieras URL- adresser som ska kunna nås och vad som ska hända vid anrop med olika sorters HTTP-metoder till samma URL-adress [7].

2.7.1. Request-klasser

Request-klasser är klasser som kan användas för att validera inkommande data vid HTTP-anrop. Vid ett HTTP-anrop så valideras den inkommande datan genom att det i funktionen ”rules” anges vilka fält som ska valideras och hur dem ska valideras. Laravel har fördefinierade regler som kan användas för att ange hur fält ska valideras men det går även att utveckla egna regler. I prototypens API används request-klasser för att validera anropsdata (se 5.3.1).

2.7.2. Model-klasser

Model-klasser använder sig av Laravels ORM-system för att hämta/modifiera data ifrån databaser.

Om ett fel uppstår när datan hämtas eller modifieras så genererar Laravel ett felsvar som skickas till användaren.

I prototypens API används model-klasser för att hämta och hantera tabellerna i prototypens databas (se 5.3.2).

2.7.3. Resource-klasser

Resource-klasser används för att generera ett JSON-format (se 2.3) utifrån data från databaser. För att hämta datan från databaser används Laravels model-klasser (2.7.2).

I prototypens API används resource-klasser för att generera JSON-format, de genererade JSON-formaten används för att visa data ifrån prototypens databas när API:et returnerar ett svar till användaren (se 5.3.3).

2.7.4. Controller-klasser

Controller-klasser används för att utföra åtgärder vid inkommande anrop till Laravel applikationer. Dem kan exempelvis användas för att vid ett anrop modifiera data i en databas genom en model-klass (2.7.2) och returnera ett svar bestående av ett JSON-format som genererats av en resource-klass.

I prototypens API används controller-klasser för att hantera model-klasser och returnera svar till användaren vid ett anrop (se 5.3.4).

Vid anrop till prototypens webbgränssnitt används en controller-klass för att returnera sidor till användaren (se 5.4.2).

2.8. CodeIgniter

CodeIgniter är ett PHP-ramverk som under examenarbetet var ett alternativ i valet av ett ramverk som kunde användas under utvecklingen av prototypen. CodeIgniter är ett ramverk som är till för att skapa webbaserade applikationer. CodeIgniter följer en MVC-struktur och har funktioner för routing, validering av anropsdata och felhanteringsklasser. Till skillnad från Laravel har inte CodeIgniter något inbyggt ORM system [17].

2.9. jQuery

jQuery är ett JavaScript-bibliotek med öppen källkod som är skapat med avsikten att tillhandahålla metoder som förenklar användningen av JavaScript [11] (se Fig.2).

jQuery erbjuder metoder för att:

- Hantera HTML, DOM och CSS (se Fig.3).
- Hantering av händelser och animering (se Fig.4).
- Utföra AJAX-operationer (se Fig.5).

```
01. //Jämförelse mellan Javascript och jQuery vid utförandet av nedanstående operation
02.
03. //Hittar html-elementet med id:et test.
04. //Ändrar elementets text till Ok
05. //Ändrar elementets färg till grön
06.
07. //JavaScript
08. let element = document.getElementById('test');
09. element.innerHTML = 'Ok';
10. element.style.color = 'green';
11.
12. //jQuery
13. $('#test').html('Ok').css("color","green")
```

Fig. 3. Jämförelse mellan JavaScript och jQuery vid hantering av HTML, DOM och CSS.

```
01. //Vid klickande (händelse) på elementet med id:et test
02. //Flytta elementet 250 pixlar åt vänster (animation)
03. $('#test').on('click',function(){
04.     $(this).animate({left: '250px'});
05. });
```

Fig. 4. Exempel på hur jQuery kan användas för att hantera händelser och animering av HTML-element.

```
01. //Gör ett anrop till /api/testText
02. //Om anropet lyckades lägg in resultatet i elementet med id:et test
03.
04. $.ajax({
05.   method: "GET",
06.   url: "/api/testText",
07.   success: function( result ) {
08.     $( "#test" ).html(result);
09.   }
10. });
```

Fig. 5. Exempel på ett AJAX-anrop med jQuery.

I examensarbetet har jQuery använts i prototypens webbgränssnitt för att till exempel ändra klass på olika html-element, utföra AJAX-anrop till API:et och för att hantera händelser vid exempelvis knapptryck.

2.10. Bootstrap 4

Bootstrap 4 är ett ramverk som använder sig av CSS och JavaScript för att designa responsiva websidor, det innebär att sidorna anpassar sig till olika skärmas storlek. Ramverket består av CSS- och JavaScript-filer som gör det möjligt att genom olika klasser på HTML-element tillämpa en stil på elementet som skiljer sig från elementets standardstil. Bootstrap kan även använda HTML-klasser för att skapa element som får utökad funktionalitet genom JavaScript [10].

Den utvecklade prototypen i examensarbetet använder sig av de CSS-filer och JavaScriptkod som Bootstrap 4 tillhandahåller för att i webbgränssnittet lägga till stilar och funktionalitet till HTML-element.

2.11. DataTables

DataTables är ett tillägg till jQuery som används för att lägga till ytterligare funktioner till HTML-tabeller. DataTables är utvecklat med målet att förbättra tillgängligheten av data i HTML-tabeller [14].

En del av de funktioner som DataTables kan lägga till i HTML-tabeller listas nedan:

- Paginering av tabeller.
- Sökning av data i tabeller.
- Utföra AJAX-anrop för att fylla tabeller med data.
- Sortering av kolumner i tabeller.

I examensarbetets webbgränssnitt används DataTables för att fylla HTML-tabeller med data om testresultat genom AJAX-anrop till prototypens API. DataTables används även för att lägga till övrig funktionalitet som nämns i listan.

2.12. Git och Github

Git är ett versionshanteringssystem som gör det möjligt att hålla reda på ändringar som har gjorts i filer. Ändringarna sparas i en mapp som kallas för ett repository. Detta görs genom att utföra så kallade commits, en commit sparar versionhanterade filers tillstånd i ett repository.

En av fördelarna med att använda Git vid utveckling av kod är möjligheten att gå tillbaka till en tidigare version (commit) av koden [12].

Github är en websida som tillåter utvecklare att ladda upp ett Git repository och därmed göra det tillgängligt online. Ett uppladdat repository kan vara publikt så att vem som helst kan ladda ner koden eller privat så att endast personer som blivit inbjudna får tillgång.

I examensarbetet har Git använts för att versionskontrollera prototypens kod. Github har använts för att ladda upp koden till två privata repositiories, ett för API:et och ett för webbgränssnittet.

3. Metod

I detta kapitel kommer den arbetsprocess som har följts under examensarbetets gång att beskrivas. Därefter kommer en beskrivning av de faser som examensarbetet varit uppdelad i. Till sist kommer en analys av de källor som har använts under examensarbetet.

3.1. Arbetsprocess

Detta examensarbete utfördes under våren 2020, samtidigt som coronapandemin. Detta medförde vissa förändringar av hur arbetsprocessen gick till.

Under examensarbetet har all utveckling skett hemifrån, detta arbetssätt var inte på grund av coronapandemin utan var tänkt från uppstart av examensarbetet eftersom all utveckling kunde göras själv. Möten har hållits mellan examensarbetaren och handledaren på Region Kalmar med cirka 3 veckors mellanrum. På mötena har exjobbaren presenterat det arbete som har gjorts sedan förra mötet och en diskussion har därefter hållits med handledaren om vad som borde göras i nästa steg. Mötena med handledaren skedde först på plats i Kalmar men kom att senare utföras via videosamtal på grund av coronapandemin som härjade samtidigt som examensarbetet utfördes.

Varje fredag har examensarbetaren mejlat en statuppdatering till handledaren på Region Kalmar län med information om vad som har gjorts under veckan. Handledaren har då haft möjlighet att komma med synpunkter på arbetet.

All kod som har skrivits under examensarbetet har versionhanterats med Git och lagts upp på ett repository på Github.

3.2. Faser

Under examenarbetet har arbetet delats in i faser där olika delar av examenarbetet har tilldelats ett större fokus, under faserna kunde det ibland behövas göra ändringar till delar av prototypen som haft ett större fokus i en annan fas (se Fig.6.). I detta delkapitel presenteras faserna i kronologisk ordning av när de inleddes. Examensarbetaren har inte följt någon utvecklingsmetod men arbetsprocessen under utvecklingsfaserna (Fas. 2-4) har varit iterativ och under faserna har ändringar gjorts till prototypens olika delar som inte har varit planerade vid början av utvecklingsfaserna.

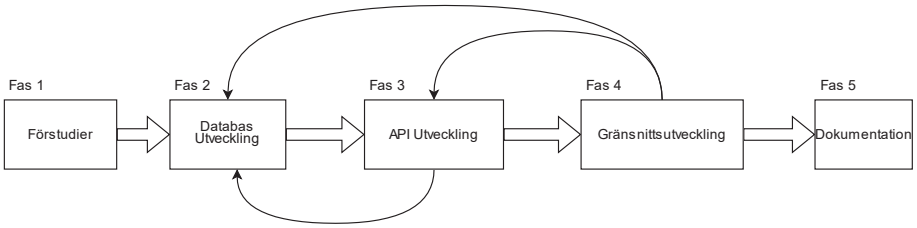


Fig. 6. Bilden visar hur examensarbetens faser har genomförts. De stora pilarna demonstrerar ett byte från en fas till en annan medan de mindre pilarna visar hur det i olika faser har behövts göra ändringar till delar av protypen som tillhört en annan fas.

3.2.1. Fas 1. Förstudier

I början av examensarbetet ägnades två veckor till att läsa på om tekniker och designprinciper som möjligtvis kunde användas under utvecklandet av prototypen. Studierna innefattade sökning efter information på internet och litteraturstudier. All information som hittades som kunde vara till hjälp antecknades ner i en Excel-fil (länkar till artiklar, länkar till dokumentation och namn på böcker). För att analysera informationen och dra slutsatser om vad som kunde användas under prototypens utveckling jämfördes de olika informationskällorna med varandra.

I denna fas studerades också hur resultaten från de manuella testerna på Region Kalmar län rapporterades in. Detta gjordes genom att examensarbetaren undersökte det verktyg som används för manuell

rapportering av tester, examensarbetaren undersökte verktyget genom att använda sig av verktyget för att själv rapportera in låtsasresultat. Detta hjälpte examensarbetaren att få en klarare bild av hur tester utförs och vilken data som kan vara viktigt att ha med vid inrapportering av resultat av tester.

Under fasen gjordes en sökning på internet efter information om olika programmeringsspråk som skulle kunna användas vid utvecklingen av API:et och webbgränssnittet. I studien lästes artiklar där olika programmeringsspråk för webbutveckling jämfördes, artiklarna kunde inte ses som en trovärdig källa eftersom de är baserade på åsikter, men de kunde ge en antydan om vad andra utvecklare anser om olika programmeringsspråk. Efter studien av artiklarna så lästes en del av dokumentationen av de programmeringsspråk som skulle kunna vara aktuella att använda sig av vid utvecklandet av prototypen, När dokumentationen för de olika programspråken hade läst gjordes ett val av vilket språk som skulle användas (se 4.1.1).

När valet av programmeringspråk hade gjorts studerades även ett antal artiklar där olika PHP-ramverk jämfördes med varandra. Artiklarna hjälpte examensarbetaren att få en fingervisning om vilka ramverk inom PHP som kunde användas vi utveckling av webbaserade applikationer, utifrån artiklarna valde examensarbetaren att studera CodeIgniter och Laravels dokumentationer för att kunna göra ett val av vilket ramverk som skulle väljas (se 4.1.2).

3.2.2. Fas 2. Databasutveckling

Designen av databasen inleddes med diskussioner tillsammans med handledaren på plats i Region Kalmar om hur deras nuvarande inrapportering av manuella tester ser ut och hur handledaren tänkt sig att de automatiska testernas inrapportering skulle skilja sig ifrån dem. Utifrån diskussionerna så definierades den struktur som de inrapporterade testresultaten skulle bestå av.

Efter diskussionerna hade hållits och därmed en struktur för testresultaten hade definierats så designades en initial version av databasen av examensarbetaren, den initiala versionen var tänkt att i

stora drag representera hur den slutgiltiga databasen skulle vara uppbyggd.

Den initiala versionen av databasen presenterades för handledaren på Region Kalmar län. När den initiala versionen var godkänd av handledaren kunde nästa fas påbörjas.

3.2.3. Fas 3. Utveckling av API

API-utvecklingsfasen inleddes med att utveckla kod som kunde användas för att demonstrera hur ett testresultat skulle rapporteras in. Koden presenterades för handledaren på Region Kalmar län först genom e-mail där en skärmdump togs av hur anropen till API:et skulle kunna se ut, som handledaren kunde komma med synpunkter på. En demonstration hölls även för handledaren vid ett möte i Kalmar, även här kunde examensarbetaren och handledaren diskutera ändringar av API:et.

För att säkerhetsställa att API:ets funktioner fungerade som det var tänkt utfördes tester för varje funktion som API:et skulle tillhandahålla. Testerna bestod av anrop till API:et och en kontroll av att svaren stämde med vad examensarbetaren förväntade att svaret från API:et skulle innehålla. Om ett svar inte var korrekt så analyserades och korrigerades API:ets kod för att uppnå ett svar från API:et där svaret stämde överens med vad examensarbetaren förväntade sig som svar.

Under utvecklingen av API:et och i mindre utsträckning webgränssnittet (3.2.4) upptäcktes möjliga ändringar i designen av databasen som examensarbetaren ansåg skulle innebära en förbättring av inrapporteringen av tester i databasen. Vid dessa upptäckter infördes ändringarna och presenterades därefter till handledaren på Region Kalmar som kunde ge sin feedback på ändringen.

3.2.4. Fas 4. Utveckling av webbgränssnitt

Utvecklingen av webbgränssnittet inleddes efter att API:ets funktioner hade blivit utvecklade till den grad att det gick att använda dess funktioner för att skapa gränssnittet. Eftersom gränssnittet använde sig av API:et blev utvecklingen av gränssnittet även ett sätt för

examensarbetaren att upptäcka brister i API:ets funktioner utöver de test som redan hade genomförts under API:ets utvecklingsfas.

Under utvecklingen av gränssnittet så utfördes tester genom att examensarbetaren testade nya funktioner och kod efter dem blivit utvecklade för att se om dem fungerade som det var tänkt. Testerna gjordes genom att examensarbetaren använde sig av gränssnittet och utförde den funktion som hade utvecklats till gränssnittet eller testade den del av gränssnittet där ändringarna hade gjorts. Om en bugg upptäcktes så påbörjades arbete med att åtgärda buggen.

Fasen tog längre tid att än vad som var planerat, de största anledningarna till att fasen tog längre tid beskrivs i delkapitel 4.1.4 och 4.4.

3.2.5. Fas 5. Dokumentation

När en slutgiltig prototyp hade blivit utvecklad började examensarbetaren skriva manualer för API:et och Webbgränssnittet. Manualerna som skrevs är tänkta att kunna användas av testare på Region Kalmar vid användning av prototypen och består av två PDF-filer som förklarar olika delar av prototypen, en för webbgränssnittet och en för API:et. Till API:et skapades även en Excel-fil som beskriver vilken data som kan skickas in till API:et vid anrop samt vilken data ett svar från API:et kan innehålla. Manualerna skickades till handledaren när de var klara. Tanken var att det skulle finnas tid till att utföra användartest av manualerna men på grund av tidsbrist kunde inte detta utföras (se 6.2).

3.3. Källkritik

I detta delkapitel kommer källorna som använts i detta examensarbete att beskrivas, en kort förklaring om varför examensarbetaren anser dem vara trovärdiga kommer även ges.

[1] Är dokumentation över programspråket PHP skriven av utvecklarna av språket, eftersom dokumentationen är skriven av utvecklarna anses den vara trovärdig.

[2] Är den officiella sidan för JSON, utöver detta finns det information om JSON på andra sidor som bekräftar att informationen stämmer, källan anses därför vara en trovärdig källa.

[3],[4],[5],[6],[7],[8] Är officiell dokumentation för Laravel. Information på andra sidor bekräftar informationen som anges i källorna och eftersom dokumentation har använts under utvecklandet av prototypen det kunnat bekräftas av examensarbetaren själv att dokumentationen är korrekt. Därför anses källorna som trovärdiga.

[9] Är en avhandling som beskrev REST för första gången, flera olika hemsidor refererar till avhandlingen, därför anses det vara en trovärdig källa.

[10] Är officiell dokumentation av utvecklarna av Bootstrap. Dokumentationen har använts under utvecklingen av prototypen och har därmed kunnat bekräftas av examensarbetaren, därför anses det vara en trovärdig källa.

[11] Är jQuerys officiella sida informationen på sidan stämmer överens med information på andra sidor, informationen anses därför vara trovärdig.

[12] Är en bok om Git som är publicerad av förlaget Apress. Boken finns tillgänglig att läsa på Gits officiella sida, den anses därför vara trovärdig.

[13] Är en bok som är publicerad av förlaget O'Reilly som tar upp REST, boken finns tillgänglig på LTH:s bibliotek vilket används i utbildningssyfte och anses därför vara trovärdig.

[14] Är dokumentation från DataTables egna hemsida dokumentation har kunnat bekräftas av examensarbetaren under utvecklandet av prototypen, den anses därför vara en trovärdig källa.

[15] Är Microsofts Officiella sida för SQL Server, informationen har bekräftats genom andra sidor som innehåller samma information. Källan anses därför vara pålitlig.

[16] Källan är Node.Js officiella sida informationen på sidan stämmer överens med information om Node.Js på andra sidor. Källan anses därför vara trovärdig.

[17] Källan är dokumentation för CodeIgniter informationen i dokumentationen stämmer överens med andra hemsidors information om ramverket. Därför anses källan vara trovärdig.

4. Analys

I detta kapitel kommer de viktigaste val som gjordes under examenarbetet att beskrivas.

4.1. Val av mjukvara att använda i utvecklingen

Under förstudiernas undersökning av programmeringspråk (3.2.1) nämndes det i en stor del av de artiklar som examensarbetaren läste att användning av ramverk kan skynda på processen vid utveckling av webbaserade applikationer. Efter att examensarbetaren undersökt detta närmare genom att läsa olika ramverks dokumentationer ansåg examensarbetaren att användning av ett ramverk skulle komma att förenkla utvecklingen av prototypen avsevärt eftersom examensarbetaren då har tillgång till funktioner och klasser redan vid början av utvecklingen och slipper utveckla dem själv. Examensarbetaren ansåg också att användningen av ett ramverk skulle hjälpa till vid en eventuell vidareutveckling av prototypen i framtiden.

Region Kalmar har som krav att all mjukvara som användes för utvecklingen av prototypen ska ha en gratislicens men utöver detta hade examensarbetaren fria händer i valet av mjukvara. Alla beslut angående vilken mjukvara som skulle användas vid utvecklingen av prototypen har därför utgått från detta krav.

4.1.1. Val av programmeringsspråk på serversidan

Vid valet av vilket programmeringsspråk som skulle användas som språk på serversidan vid utvecklandet av API/Webbgränssnitt så valdes PHP. Valet gjordes utifrån punkterna nedan.

- PHP är ett språk som är specialiserat för webbutveckling.
- Eftersom PHP har funnits sedan 1995 finns det mycket litteratur angående språket utöver den dokumentation som utvecklarna av PHP själva tillhandahåller. Detta ansågs av examensarbetaren att vara en fördel vid utvecklandet av protypen eftersom det innebär att det finns mer information om språket som kan användas under utvecklandet av prototypen.
- Examensarbetaren hade redan erfarenhet av utvecklande med PHP innan examensarbetet utfördes.

Andra programmeringsspråk som undersöktes var Node.js. Node.js valdes inte eftersom examensarbetaren inte hade någon erfarenhet av språket sedan tidigare och onödig tid skulle läggas på att lära sig språket, vilket med stor sannolikhet skulle påverka kvalitén hos prototypen negativt

När valet att använda PHP var gjort beslöts även att använda en Windowsbaserad server för prototypen, valet gjordes eftersom Region Kalmar har erfarenhet av att använda sig av Windowsservrar.

4.1.2. Val av PHP-ramverk

När valet av PHP som språk på serversidan hade gjorts kunde ett val av ett ramverk därmed också göras. I alla artiklar som examensarbetaren läste så nämndes ramverket Laravel och CodeIgniter som de mest använda ramverken för webbutveckling i PHP och därför valde examensarbetaren att studera ramverkens dokumentation närmare för att se om ramverket kunde användas för att utveckla prototypen.

Vid jämförelse med dokumentationen av de båda ramverken så fanns det vissa delar som gjorde att Laravel valdes. Eftersom CodeIgniter är ett lättviktigt ramverk så var det många funktioner som Laravel har inbyggt som krävde nerladdning av andra bibliotek i CodeIgniter. Till exempel har Laravel ett inbyggt ORM-system, det hade inte CodeIgniter utan det kräver användning av externa bibliotek. Utifrån detta gjorde examensarbetaren valet att använda sig av Laravel under utvecklandet av prototypen.

De fördelar examensarbetaren ansåg att Laravel tillhandahöll i utvecklingen av prototypen listas nedan.

- Laravel kommer med ett inbyggt ORM-system som skulle tillåta prototypen att hantera tabeller i databasen på ett objektorineterat vis. Det innebär att en utvecklare kan slipper skriva egen SQL-kod vilket examensarbetaren ansåg skulle spara tid under utvecklandet av prototypen.
- Laravel har inbyggda funktioner för routing av API:er vilket gör det enklare att hantera API:ets resurser vid ett anrop.
- Laravel kan användas både i utvecklingen av API:et och Webbgränssnittet. Ramverket kommer både med funktioner för backend och frontend utveckling.
- Laravels databasmigrationssystem gör det möjligt att skapa databastabeller utan att skriva SQL-kod. Migrationssystemet ger också utvecklaren kommandon som kan utföras för att återgå till tidigare versioner av databasen via kommandotolken. Detta ansåg examensarbetaren skulle underlätta om ändringar behövde göras i databasen under examensarbetets gång.

De här fördelarna tillsammans med de generella fördelarna med ett ramverk nämnda i 4.1 gjorde att Laravel valdes som ramverk.

4.1.3. Bootstrap 4

Eftersom examensarbetaren inte hade någon utbildning inom webbdesign så gjordes ett val att utveckla webbgränssnittet med Bootstrap 4. Examensarbetaren ansåg att detta skulle resultera i ett webbgränssnitt med en design som var konsekvent i hela gränssnittet och spara tid eftersom examensarbetaren istället för att skriva egen CSS-kod kunde använda sig av den CSS som Bootstrap 4 tillhandahåller.

4.1.4. DataTables

Under utvecklingen av webbgränssnittet (se 3.2.4) insåg examensarbetaren att de HTML-tabeller som visade testrapporteringsdatan riskerade att innehålla mycket data. Examensarbetaren ansåg därför att det fanns en risk att det skulle bli svårt för en användare att hitta den information i tabellen som användaren sökte efter.

För att lösa detta problem tog examensarbetaren först ett beslut att skriva egen JavaScript-kod för att ge användaren möjlighet att söka efter data i tabellerna, för att implementera funktionen gjorde examensarbetaren en internetsökning på hur man på bästa sätt kunde implementera denna funktion och hittade ett diskussionsforum som nämnde att DataTables möjliggjorde denna funktion. Vid en studie av DataTables dokumentation upptäckte examensarbetaren ett antal funktioner som ansågs förbättra webbgränssnittets tabeller, förbättringar listas nedan.

- DataTables kan göra AJAX-anrop till ett API och utifrån svarsdata bygga upp en tabell.
- DataTables har funktioner för att söka efter data i tabeller utifrån ett sökord.
- DataTables kan paginera en tabell och låta användaren kontrollera hur många rader i tabellen som ska visas per sida.
- DataTables lägger till funktioner så att användaren kan sortera en tabell utifrån kolumner.

En konsekvens av detta val var att en del kod i webbgränssnittet behövde skrivas om, detta ledde till att det tog längre tid än planerat att utveckla webbgränssnittet.

4.2. Databas

Valet av Microsoft SQL Server som databashanterare gjordes enbart av den anledning att Region Kalmar framförallt använder sig av den som databashanterare. Det ansågs därför vara lämpligt att prototypen blev utvecklad med SQL Server eftersom det skulle underlätta vid en potentiell vidareutveckling av prototypen.

Examensarbetaren hade erfarenhet av att använda SQL sedan tidigare, men då med databashanteraren MySQL. Skillnaden mellan MySQL och SQL Server ansågs av examensarbetaren att inte vara tillräckligt stor för att det skulle uppstå problem vid utvecklandet av prototypen.

4.3. API

4.3.1. Val av API teknik

Vid valet av vilken teknik som skulle användas vid utvecklandet av API:et så valdes REST API arkitekturen. Valet gjordes till största del för att Laravel tillhandahåller klasser och funktioner som använder sig av RESTful arkitekturen [5]. Efter att ha studerat RESTful arkitekturen ansåg examensarbetaren att prototypens API kunde använda sig av tekniken.

4.3.2. Val av JSON som svar på anrop

För att API:et skulle vara kompatibelt med så många programmeringsspråk som möjligt valde examensarbetararen att skicka svar vid anrop till API:et i ett JSON-format om det var möjligt.

4.4. Webbgränssnitt

När webbgränssnittet skulle utvecklas (se 3.2.4) gjordes ett val av examensarbetaren att enbart använda anrop till API:et för att hantera den inrapporterade data. Detta var inte ett krav som ställdes av Region Kalmar och det hade i praktiken varit möjligt att använda sig av samma klasser och funktioner som API:et använder sig av för att skapa webbgränssnittet. Examensarbetaren ansåg dock att en separation av API från webbgränssnittet där all data som visas i gränssnittet hämtas med anrop till API:et skulle vara till en fördel för utvecklandet av protypen och handledaren höll med om att det var en bra idé. Fördelarna som examensarbetaren ansåg att användning av API:et vid utvecklingen av gränssnittet listas nedan.

- Genom att använda sig enbart av anrop till API:et kan brister i API:et upptäckas samtidigt som utvecklingen av webbgränssnittet sker. Detta skulle innebära att utvecklingen av gränssnittet blir en del av testningen av API:et, vilket skulle potentiellt kunna förbättra API:et.
- De resurser webbgränssnittet kan hantera går även att hantera genom API:et. Webbgränssnittet blir med andra ord en demonstration av de funktioner som API:et tillhandahåller.

Valet av att använda sig enbart av API:et för att visa/hantera testresultat i webbgränssnittet innebar att examensarbetaren även gjorde ett val att dela upp API:et och webbgränssnittet i två olika utvecklingsprojekt. Uppdelningen gjordes för att det under utvecklandet av webgränsnittet skulle säkerhetsställas att det i webbgränssnittet inte fanns några beroenden till kod som skrivits för API:et, därmed är examensarbetaren säker på att endast anrop till API:et används av gränssnittet.

Valet ledde också till att utvärderingen av vad gränssnittet skulle innehålla utökades till att försöka få med så mycket funktioner som API:et tillhandahåller som möjligt i gränssnittet, detta innebar att examensarbetaren utvecklade funktioner i webgränsnittet som tillät en användare att utföra CRUD-operationer på all data som visas i gränssnittet. En konsekvens av detta val blev att tiden det tog att utveckla gränssnittet blev längre än vad som var planerat.

5. Resultat

I följande kapitel kommer resultatet av examensarbetet att presenteras.

5.1. Översikt av prototypen

Prototypen som utvecklats under examenarbetets gång består av tre delar.

1. En databas.
2. Ett REST API.
3. Ett webbgränssnitt.

Prototypen är utvecklad för att ge Region Kalmar möjlighet att rapportera in resultat vid utförandet av automatiska tester.

Databasen används för att lagra testresultaten från de utförda automatiska testen, API:et används för att hantera testresultaten i databasen och webbgränssnittet används för att visa och hantera testresultaten genom användning av API:et. I Fig.7 visas ett scenario där alla tre delar av prototypen används.

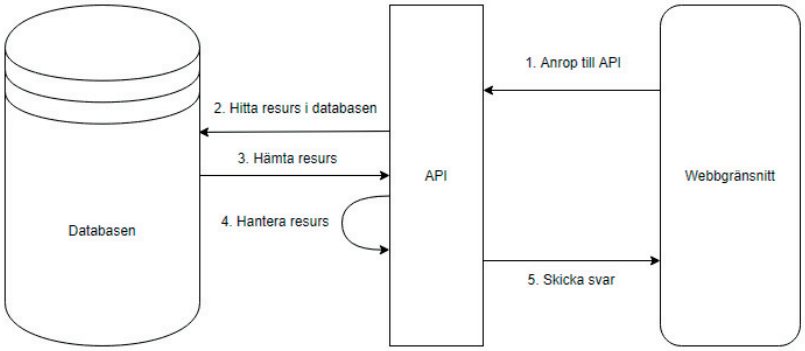


Fig. 7. Ett förenklat exempel på hur ett anrop som använder sig av hela prototypen ser ut.

5.1.1. Testresultatens struktur

De rapporterade testresultat som prototypen hanterar består av en hierarki av fyra resurser:

1. Ett projekt
2. En testkörning
3. En testfallsinstans
4. Ett teststeg.

Projektet anger vilken mjukvara som ska testas (t.ex. Cosmic 8.2).

Testkörningar anger vad det är som ska testas (t.ex. Logga in i mjukvaran).

Testfallsinstanser anger mer specifikt vad som ska göras (t.ex. Logga in med Bank id).

Ett teststeg anger vad som ska göras i testfallsinstanserna (t.ex. Klicka på inloggnings knappen). Det är i teststegen som resultatet av testerna rapporteras in (t.ex. lyckat eller misslyckat). I Fig. 8 visas hur testresultatens hierarki är uppbyggd.

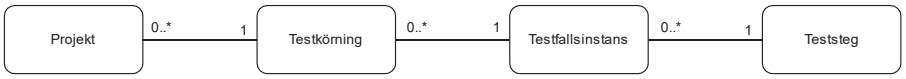


Fig. 8. Testresultatens struktur

Testfallsinstanserna är instanser av testfall. Testfall är en separat del av testen och tillhör inte hierarkin av testresultat.

När en testfallsinstans skapas så anges vilket testfall som instansen ska utgå ifrån, en testfallsinstans är med andra ord en kopia av ett testfall. Denna struktur underlättar att utföra samma testfall men med olika teststeg (se Fig. 9).

Till teststegen och testfallsinstanserna är det även möjligt att skriva textkommentarer. Detta gör att ytterligare information om testresultaten kan läggas till om det skulle behövas.

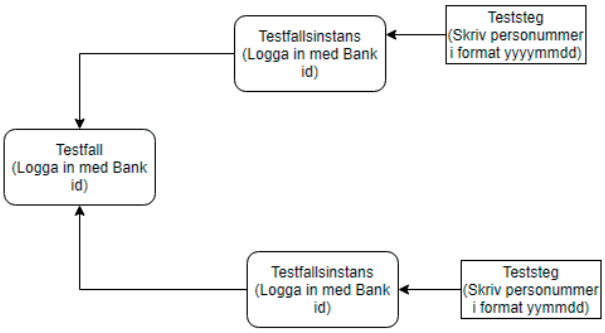


Fig. 9. Exempel på hur olika testfallsinstanser används för att underlätta utförandet av ett testfall men med olika teststeg (notera att teststegen anger annorlunda format på personnummer).

5.2. Databasen

Prototypens databas består av en relationsdatabas som hanteras av Microsoft SQL server. Databasen har utvecklats för att kunna lagra inrapporteringsdata från automatiska tester utförda av Region Kalmar. Databasen består av 11 databastabeller. Ett diagram över databasen kan ses i Appendix 9.1.

Databasen utvecklades med hjälp av Laravels databasmigreringssytem (se Fig.10.).

```
01. class CreateProjectsTable extends Migration
02. {
03. 
04.     //Skapa project tabellen i databasen
05.     public function up()
06.     {
07. 
08.         Schema::create('project', function (Blueprint $table) {
09.             $table->bigIncrements('id');
10.             $table->string('title')->unique();
11.             $table->string('description')->nullable();
12.             $table->string('owned_by')->nullable(false);
13.             $table->timestamp('ending_date')->nullable();
14.             $table->softDeletes('archived');
15.             $table->string('created_by')->nullable(false);
16.             $table->string('updated_by')->nullable();
17.             //Skapar två datum-kolumner (created_at, updated_at)
18.             $table->timestamps();
19.         });
20.     }
21.     //Ta bort project-tabellen från databasen
22.     public function down()
23.     {
24. 
25.         Schema::dropIfExists('project');
26. 
27.     }
28. }
```

Fig. 10. Kod för att skapa en tabell med hjälp av Laravels databasmigrationssystem. Koden skapar databastabellen med namn project (se Appendix 9.1).

5.3. API

Prototypen består av ett REST API som via HTTP-anrop hanterar resurser (data) i prototypens databas. API:et är utvecklat med hjälp av ramverket Laravel.

API:et kan användas för att utföra CRUD-operationer i databasens tabeller.

API:et använder sig av Laravels request-, model-, controller- och resource-klasser samt en routing-fil för att hantera inkommande HTTP-anrop. I Fig. 11 visas hur ett anrop från en användare av API:et hanteras av klasserna och filen.

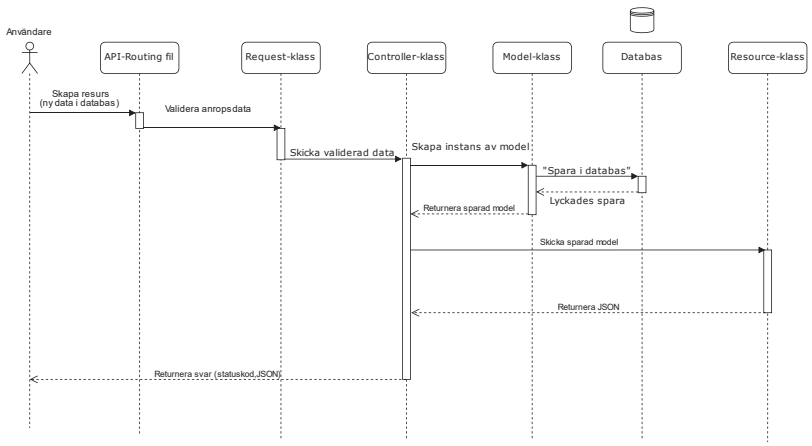


Fig. 11. Ett förenklat sekvensdiagram över ett anrop till API:et vid skapande av en resurs.

I kommande delkapitel kommer klasserna och filen att beskrivas mer djupgående.

5.3.1. Validering av inkommande data

För varje resurs som hanteras av API:et har en request-klass skapats för att validera inkommande data vid HTTP-anrop som försöker skapa eller uppdatera en resurs (se Fig. 12). Förutom de regler som redan är definierade av Laravel har examensarbetaren även skapat regler som används för att inte tillåta att ett fält anges vid skapande eller uppdaterande av en resurs.

```
01. class ProjectRequest extends FormRequest
02. {
03.  public function rules()
04.     {
05.         //Switch sats som kontrollerar vilken metod som har använts i anropet
06.         switch ($this->method()) {
07.             //Vid skapande av resurs
08.             case "POST":
09.                 return [
10.                 'title' => 'required|string|unique:App\Project,title|max:255',
11.                 'description' => 'sometimes|string|filled',
12.                 'owned_by' => 'sometimes|string|max:255',
13.                 'created_by' => 'required|string|max:255',
14.                 'updated_by'=>new OnCreationNotAllowedField(),
15.                 'ending_date' => 'date'];
16.
17.             //Vid uppdatering av resurs
18.             case "PUT":
19.             case "PATCH":
20.                 return [
21.                     'title'=>'sometimes|string|unique:App\Project,title|max:255',
22.                     'description' => 'sometimes|string|filled',
23.                     'owned_by' => 'sometimes|string|max:255',
24.                     'created_by' => new OnUpdateNotAllowedField(),
25.                     'updated_by'=>'required|string|max:255',
26.                     'ending_date' => 'date'];
27.         }
28.     }
29.
30. }
```

Fig. 12. Kod från klassen ProjectRequest som validerar anrop som försöker skapa/ändra projektresurser. I koden kan man se hur Laravels egna definierade regler används, exempelvis på rad 15 och rad 26 där fältet ”ending_date” måste vara i ett datumformat. Rad 27 och 35 är exempel på regler som har utvecklats av examensarbetaren.

5.3.2. Hantering av prototypens databas

I API:et används Laravels model-klasser för att hämta/modifiera data ifrån prototypens databas. En model-klass har skapats för varje databastabell i prototypens databas. I Fig. 13 visas kod från den model-klass som används av API:et för att hantera data i tabellen med namn ”project” (se Appendix 9.1) och i Fig. 14 visas hur klassen används för att uppdatera data i databasen.

```php
class Project extends Model
{
    //Anger vilken tabell i databasen som model-klassen tillhör
    protected $table = 'project';

    //Anger relationer som model-klassen har till andra model-klassen (foreign keys)
    //Ett projekt kan ha många model-klasser av typ TestRun (teskörningar)
    public function testRuns()
    {
        return $this->hasMany('App\TestRun');
    }
}
```

Fig. 13. Model-klassen Project som används för att hämta projekt-resurser ifrån databasen.

```php
//Hitta ett projekt i databasen som har id 1 (SELECT * FROM project WHERE id = 1)
//Skapa en instans av en model-klass från datan
$project = Project::find(1);
//Ändra titeln på instansen
$project->title = "Examensarbete";
//Spara modelen i databasen
$project->save();
```

Fig. 14. Exempel på hur man med hjälp av Laravels ORM system uppdaterar data i databasens project-tabell (Exemplet kräver att det finns ett projekt som har id 1 i databasen).

5.3.3. Generering av JSON-format

I API:et används Laravels resource-klasser för att generera ett JSON-format utifrån API:ets model-klasser.

I Fig. 15 visas koden för den resource-klass som används för att visa projekt-resurser och i Fig. 16 visas hur ett svar som använder sig av koden ser ut.

```php
class ProjectResource extends JsonResource
{
   //Returnera en array utifrån en model-klass ($this)
   public function toArray($request)
    {
            $array = array();
            $array['Project id'] = $this->id;
            $array['Title'] = $this->title;
            $array['Description'] = $this->description;
            $array['Owned by'] = $this->owned_by;
            $array['Creation date'] = \Carbon\Carbon::parse($this->created_at)->format('Y-m-d H:i:s');
            $array['Creator'] = $this->created_by;
            $array['Updated by'] = $this->updated_by;
            $array['Updated date'] =$this->updated_at === null ? null:\Carbon\Carbon::parse($this->updated_at)->format('Y-m-d H:i:s');
            //Om resursen är arkiverad visa när den arkiverades
            if($this->archived !==null){
                $array['Archived date'] = \Carbon\Carbon::parse($this->archived)->format('Y-m-d H:i:s');
            }
            $array['Ending date'] = $this->ending_date === null ? null:\Carbon\Carbon::parse($this->ending_date)->format('Y-m-d');
            $array['Percentage done'] = $this->percentage_done;
            $array['Testrun count'] = $this->testrun_count;
            return $array;
    }
}
```

Fig. 15. Kod för resource-klassen ”ProjectResource” som används för att presentera information till användaren om projekt-resurser.


```json
{
    "Project id": 1,
    "Title": "Exjobb",
    "Description": "Test projekt för exjobbare",
    "Owned by": "Christian",
    "Creation date": "2020-05-05 20:47:34",
    "Creator": "Christian",
    "Updated by": null,
    "Updated date": "2020-05-05 20:47:35",
    "Ending date": null,
    "Percentage done": 0.6875,
    "Testrun count": 1
}
```


Fig. 16. Exempel på hur ett JSON-svar från API:et ser ut när användaren vill få information om projekt-resursen som har titeln Exjobb. API:et använder resource-klassen som visas i Fig. 16 för att generera svaret.

5.3.4. Åtgärder vid anrop till API:et

För varje model-klass i API:et har en controller-klass skapats. Controller-klasser i API:et utför två steg:

1. En CRUD-operation utförs på en eller flera model-klass instanser.
2. Ett svar returneras till användaren.

Svaret innehåller en HTTP-statuskod som talar om för användaren om anropet har lyckats eller ej. Laravel har funktioner för att på egen hand bestämma vilken statuskod som ska skickas vid ett eventuellt misslyckat anrop så examensarbetaren har endast behövt avgöra vilken statuskod som ska skickas vid ett lyckat anrop.
Svaren ifrån API:et innehåller för det även ett data i form av ett JSON-format genererat av API:ets resource-klasser (se Fig.16.).

Koden för den controller-klass som hanterar projekt-resurser visas i Fig.17.

```
01. class ProjectController extends Controller
02. {
03.     //Skapa en projekt-resurs
04.     //ProjectRequest blir automatiskt validerad när den används som parameter
05.     public function store(ProjectRequest $request)
06.     {
07.         return response()->json(new ProjectResource(Project::create($request->all())),200);
08.     }
09.     //Visa en projekt-resurs
10.     // När en model-klass används som parameter görs sökningen i databastabellen automatiskt
11.     public function show(Project $project)
12.     {
13.
14.         return response()->json(new ProjectResource($project),200);
15.     }
16.
17.     //Uppdatera en projekt-resurs
18.     public function update(ProjectRequest $request, Project $project)
19.     {
20.
21.         $project->update($request->all());
22.         return response()->json(new ProjectResource($project->fresh()), 200);
23.     }
24.    //ta bort en projekt resurs
25.    public function destroy(Project $project)
26.     {
27.         $project->forceDelete();
28.         return response()->json(new ProjectResource($project),200);
29.     }
30. }
```

Fig. 17. Koden för controller-klassen ProjectController, som hanterar projekt-resurser vid anrop till API:et.

5.3.5. API-Routing

API:et använder sig av en routing-fil för att avgöra vilken controller-klass och funktion i controller-klassen som ska användas för att hantera anropet (se Fig. 18). Med andra ord är det routing-filen som exponerar API:ets funktioner till användare.

```
01. //Skapa projekt
02. Route::post('/projects', 'API\ProjectController@store');
03. //Hämta information om ett projekt
04. Route::get('/projects/{project}','API\ProjectController@show');
05. //Uppdatera ett projekt
06. Route::patch('/projects/{project}','API\ProjectController@update');
07. //Ta bort ett projekt
08. Route::delete('/projects/{title}', 'API\ProjectController@destroy');
```

Fig. 18. Exempel av kod ifrån API:ets routing-fil. Exemplet visar hur URL adresser för att hantera projekt-resurser anges. Måsvingarna (t.ex. {project}) anger en parameter som skickas till controller-klassen. Controller-klassfunktionerna som används är samma funktioner som visas i Fig.17.

5.4. Webbgränssnitt

Webbgränssnittet har utvecklats i PHP och JavaScript. PHP har använts som kod på serversidan och JavaScript som kod på klientsidan. Gränssnittet är utvecklat med hjälp av Laravel. Bootstrap 4 används för att skapa en responsiv sida, tillämpa stilar till gränsnittets HTML-element med hjälp av CSS och för att lägga till mer funktionalitet till elementen med hjälp av JavaScript.

Gränssnittet använder sig av anrop till prototypens API för att visa/hantera inrapporterade resultat från automatiska tester. Webbgränssnittet har funktionalitet för att använda sig av alla API:ets funktioner. I Fig.19 kan den sida som visar alla projekt ses.

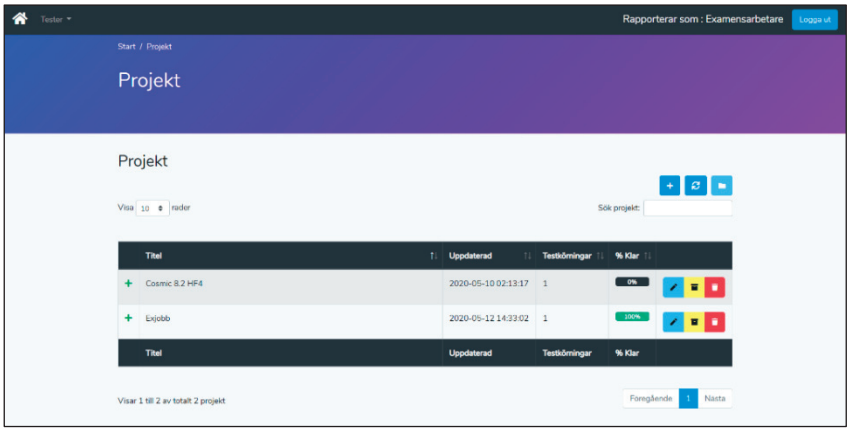


Fig. 19. Projektsidan av webbgränssnittet, sidan visar alla projekt som finns i prototypens databas. För resterande sidor i gränssnittet se Appendix 10.

5.4.1. API-Anrop

Gränssnittet kan anropa API:et på två olika sätt:

1. När en sida laddas initialt (anrop från serversidan).
2. På en redan nedladdad sida med hjälp av AJAX-anrop (anrop från klientsidan). Anropen utförs tillexempel när en ny resurs skapas genom gränssnittet.

Anropen som utförs på serversidan görs av klassen RestApiRequester. Klassen är en wrapper-klass runt Laravels ”Http” klass, som används för att göra http anrop. Koden för RestApiRequester visas i Fig. 20.

```php
class RestApiRequester implements RestApiRequestInterface
{
    public static function request(string $apiRoute, string $method = self::METHOD_GET, FormRequest $request = null): \Illuminate\Http\Client\Response
    {
        //Hämta domänen till api:et från laravels app-config fil
        $baseUrl = config('app.api_base_adress');
        $data = [];
        if ($request !== null) {
            $data = $request->toArray();
        }
        //Lägg till den specifika adressen i api som vill nås till domän adressen
        $fullAdress = $baseUrl . $apiRoute;
        switch ($method) {
            case 'GET':
                return HTTP::get($fullAdress);
            case 'POST':
                return Http::post($fullAdress, $data);
            case 'PUT':
                return Http::put($fullAdress, $data);
            case 'PATCH':
                return Http::patch($fullAdress, $data);
            case 'DELETE':
                return Http::delete($fullAdress, $data);
            default:
                return null;
        }
    }
}
```

Fig. 20. Kod för klassen RestApiRequester.

För att utföra anropen på klientsidan används de AJAX-funktioner som jQuery samt DataTables tillhandahåller.

5.4.2. Anrop till gränssnittet

När en användare gör ett anrop för att nå en sida i gränssnittet så används en routing-fil (liknande den i Fig. 18) för att kalla på en funktion i klassen ViewController (se Fig. 21.).

ViewController är en klass som anropar API:et med hjälp av klassen RestApiRequester, om anropet var lyckat returneras en sida tillsammans med datan som svaret från API:et innehåller.

```php
class ViewController extends Controller
{
//Visa sidan för alla projekt
  public function projektView(){
        $projectUri = '/projects';
        //Gör ett anrop till api:et
        $projectsResponse = RestApiRequester::request($projectUri);
        //om api-anropet var lyckat
        if($projectsResponse->successful()){
            //Visa  projektsidan med medföjande data om projekten
            return $this->generalView('layouts.project.MainProjectView')
                ->with('projects',$projectsResponse->json()['Projects']);
        }
        //om anropet inte var lyckat visa felsida
        return self::abortview($projectsResponse);
    }
    //Visa felsida (abort() är en laravel funktion för att visa en felsida)
    private static function abortview(Response $response){
       return abort($response->status(),$response->body());
    }
    //returnera en sida med api:ets url som medföljande data (api:ets url används av JavaScript)
    private function generalView(string $view){
        return view($view)->with('apiUrl',config('app.api_base_adress'));
    }
}
```

Fig. 21. Kod i ViewController som används för att visa projekt-sidan.

5.4.3. Resurs-tabeller

Webbgränssnittet består till stor del utav sidor där resurserna från API:et ska hanteras. För att hantera resurserna innehåller gränssnittets sidor tabeller där resurserna listas (se Fig. 22.). Tabellerna använder sig av DataTables för att göra AJAX-anrop till API:et och fylla i tabellerna med svarsdata. DataTables används även i tabellerna för att sortera kolumner, söka i tabellerna, dela upp tabellen i sidor och bestämma hur många rader som ska visas per sida. Alla tabeller har en möjlighet att expanderas för att visa mer information om resursen (se Fig. 23).

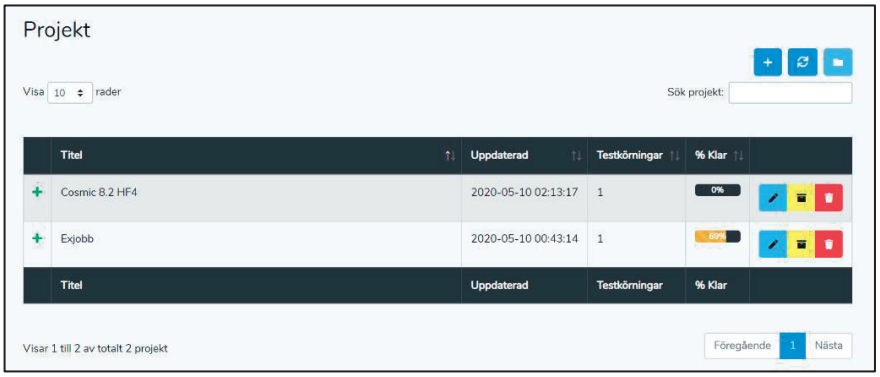


Fig. 22. Resurs-tabell som visar alla projekt. Sista kolumnen innehåller knappar för att hantera resurserna, i detta fall för att uppdatera, arkivera och ta bort projektet. Knapparna ovanför sökfältet skapar ett nytt projekt, uppdaterar tabellen och visar arkiverade projekt.

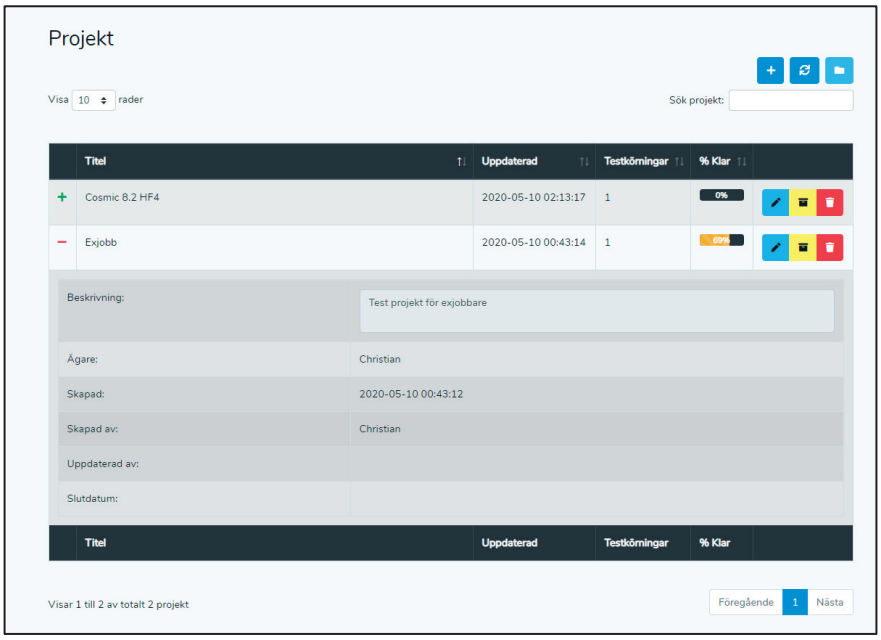


Fig. 23. Exempel på en expanderad resurs-tabell.

5.4.4. Hanterings-modaler

I webbgränssnittet kan resurser skapas eller uppdateras genom så kallade modaler (se Fig.24). Modaler är ett fönster som visas inuti den nuvarande sidan. Modalerna innehåller fält där användaren kan mata in data som används för att skapa/uppdatera resurserna.

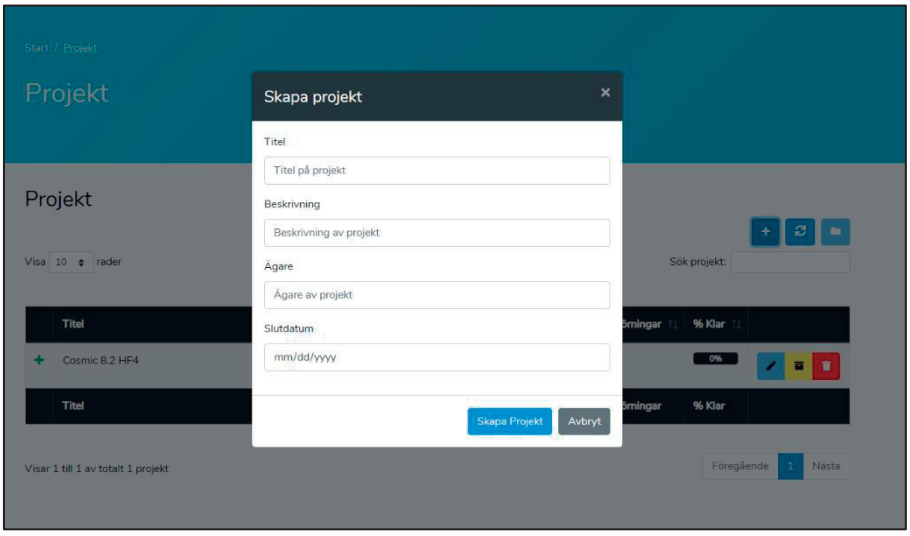


Fig. 24. En modal som används för att skapa ett projekt

5.5. Dokumentation av prototypen

Till den utvecklade prototypen skrevs två manualer, en för API:et och en för Webbgränssnittet. Manualerna är tänkta att användas för de på Region Kalmar län som ska använda webbgränsnittet och av de som ska implementera API anrop i de skript som utför automatiska tester.

5.5.1. Manual till API:et

Manualen till API:et består av två filer.

1. En Excel-fil som beskriver de olika resurserna i API:et. För varje resurs visas vilken data som kan skickas vid ett anrop och vad API:et visar när resursen skickas som ett svar.
2. En PDF-fil som beskriver API:ets URI-Adresser och vad ett lyckat anrop till adresserna utför, en beskrivning av hur man tolkar Excel-filen och en kort beskrivning av vad ett REST API är och hur anrop fungerar.

5.5.2. Manual till webbgränssnittet

Webbgränssnittets manual består av en PDF-fil. Filen beskriver de olika delarna av gränssnittet genom att beskriva generella element i gränssnittet (t.ex. resurs-tabellerna i delkapitel 5.4.3) och en beskrivning av varje websida i gränssnittet, vad de visar och vilken adress man kan nå dem på.

6. Slutsats

Syftet med examensarbetet var att utveckla en prototyp som tillät Region Kalmar län att rapportera in testresultat när automatiska tester utförs.

Examensarbetet resulterade i en databas utvecklad med SQL Server som kan lagra testresultaten, ett REST API som kan hantera testresultaten i databasen och ett webbgränssnitt som använder sig av REST API:et för att visa och hantera testresultatsdatan.

6.1. Problemformuleringar

I detta delkapitel kommer de problemformuleringar som ingick i delkapitel 1.4 att besvaras.

Vilken information om de automatiska testerna behöver lagras i databasen?

Informationen som har behövt lagras i databasen kan ses i Appendix 9.1. Databasen är uppbyggd för att lagra den testresultats struktur som beskrivs i delkapitel 5.1.1. Databasen lagrar bland annat data om projekt, resultat av utförda teststeg och kommentarer till testfallsinstanser.
Testresultatens struktur har under examensarbetets gång genomgått små förändringar. Men i stora drag har strukturen på databasen varit bestämd sedan databasutvecklingsfasen (se delkapitel 3.2.2).

Vilka funktioner ska API: et tillhandahålla för att testare ska kunna rapportera in sina resultat?

För att testare ska kunna rapportera in resultaten har API:et behövt kunna tillhandahålla testare med funktioner för att utföra CRUD-operationer på testresultatsdatan i databasen. För exempelvis projekt-resurser så kan API:et skapa ett projekt, visa ett eller flera projekt, uppdatera ett projekt och ta bort ett projekt.

Vilken funktionalitet ska webbgränssnittet ha för att möta de krav som ställs på den?

Det krav som webbgränsnittet skulle uppnå var att kunna visa resultaten av de automatiskt utförda testen. Detta har uppnåtts i gränssnittet genom att använda sig av API:ets funktioner för att visa testresultat. Utöver detta har funktionalitet utvecklats för att kunna skapa, uppdatera eller ta bort testresultat, för att uppnå denna funktionalitet har API:et funktioner använts. Webgränssnittet kan bland annat visa resultaten av teststegen i en specifik testfallsinstans och ändra resultatet för ett specifikt teststeg.

Vilka programspråk ska användas vid utvecklingen av prototypen?

PHP har valts för att utveckla både prototypens API och webbgränssnitt på serversidan. Prototypen hade kunnat använda sig av andra språk på serversidan, men av de anledningar som nämns i delkapitel 4.1.1 så valdes PHP. Utöver PHP har även JavaScript använts för att uppnå utökad funktionalitet på klientsidan.

Vilken typ av server ska användas för prototypen?

När programmeringspråk på serversidan hade valts så valdes att prototypen skulle använda sig av en Windowsbaserad server. Vid skrivandet av denna rapport har den slutgiltiga prototypen inte installerats på någon server hos Region Kalmar och ett val av vilken version av Windowsserver som ska användas för den slutgiltiga prototypen har därför inte gjorts under examensarbetet.
Valet att använda en Windowsbaserad server gjordes för att Region Kalmar län har erfarenhet av att använda sig av Windowsservrar.

Vad ska manualen för API:et samt manualen för webbgränssnittet innehålla? Hur ska det avgöras om manualerna och kommentarerna i kod är tydliga nog för att kunna användas av testare samt vid en eventuell utveckling av prototypen efter examensarbetets slut?

I delkapitel 5.5 beskrivs vad manualerna innehåller. För att avgöra om manualerna är tillräckligt borde manualerna ha blivit testade av testare på Region Kalmar län. För att säkerhetsställa att koden är tillräckligt kommenterad så borde analys av koden utföras av någon annan än examensarbetaren.

6.2. Lärdomar

Ett av målen i detta examensarbete var att utföra användartester på de manualer och den kod som skrevs till API:et och webbgränssnittet. På grund av tidsbrist har detta mål inte uppnåtts. Eftersom dokumentation skrevs sent under examensarbetet fanns det inte tillräckligt med tid och utvecklandet av framförallt gränssnittet (se 3.2.4) tog längre tid än vad som var planerat. Det är möjligt att en dokumentation för API:et hade kunnat ha skrivits i ett tidigare skede och därmed kunnat användarteststats, men eftersom API:et även förändrades under gränssnittsutvecklingsfasen så skrevs inte någon dokumentation förrän både API:et och gränssnittet ansågs vara klara. Denna brist i resultatet har accepterats av handledaren på Region Kalmar län.

6.3. Etiska aspekter

I det här delkapitlet kommer etiska aspekter i examensarbetet som examensarbetaren har identifierat under examensarbetets gång att beskrivas.

6.3.1. Samhällsnytta

Examensarbetet har bidragit med att ta fram en prototyp för att kunna rapportera och visa resultat av de automatiska test som har utvecklats av Region Kalmar län. Prototypen eller en vidareutveckling av

prototypen kommer förmodligen hjälpa Region Kalmar läns testare att få en översikt av testens resultat. Därmed kan fel i mjukvara detekteras vilket bidrar till bättre mjukvara för hela Regionen.

6.4. Framtida utvecklingsmöjligheter

I detta delkapitel kommer de utvecklingsmöjligheter av prototypen som examensarbetaren har identifierat beskrivas.

6.4.1. Säkerhet

Prototypen är utvecklad utan någon implementerad säkerhet. Detta var ett val som gjordes av Region Kalmar eftersom prototypen ska installeras på en server som inte är kopplad till internet och det finns olika tankar om hur säkerheten ska implementeras. Examensarbetaren har trots detta utvecklat prototypen med tankar om hur en framtida implementation av säkerhet skulle kunna se ut, exempel på detta är databasen där nästintill varje tabell innehåller en kolumn för vem som har skapat samt uppdaterat datan, ett annat exempel är webbgränssnittet där man måste skriva in vilket namn man ska ha när man rapporterar in data innan man får tillgång till några sidor.

Laravel tillhandahåller ett antal olika alternativ för att implementera autentisering och auktorisering i projekt utvecklade med ramverket och det borde därför vara fullt möjligt att implementera säkerhet i prototypen i framtiden utan en allt för stor omskrivning av koden.

6.4.2. Loggning av databasändringar

Prototypens databas innehåller vid skrivandet av denna rapport ingen typ av logg över vilken data som har ändrats i databasen. En implementation av detta skulle kunna utvecklas i framtiden för att göra det möjligt att se historiken över den inrapporterade datan samt återställa datan till ett tidigare stadie. Detta skulle kunna vara en fördelaktig funktion om till exempel ett felaktigt resultat har rapporterats in eller om någon data av misstag har modifierats.

7. Ordlista

Automatiska tester

Tester utvecklade av Region Kalmar läns IT-avdelning som kan utföras genom skript av en dator

AJAX

Anrop som kan utföras för att hämta data utan att behöva ladda om en hel sida.

Serversida kod

Kod som exekveras av servern.

Klientsida kod

Kod som exekveras av klienten (anroparen till servern)

API

Ett gränssnitt som tillhandahåller specifika funktioner som kan användas av andra program.

CRUD (Create, Read, Update, Delete)

Operationer som antingen skapar (create), läser (read), ändrar (update) eller tar bort (delete) data.

Användare

Maskin/Person som använder sig av systemet.

CSS (Cascading Style Sheets)

Ett språk som används för att ändra stilen på dokument, tillexempel HTML dokument.

DOM (Document Object Model)

En representation av ett HTML dokument som en trädstruktur.

Region Kalmar län

Region Kalmar läns IT-avdelning

Paginera

Att dela upp något i sidor

Exekveringsmiljö

Tillåter program att exekveras på ett visst sätt.

8. Källor

[1] What is PHP? (Hämtad: 2020-01-28)
https://www.php.net/manual/en/intro-whatis.php

[2] Introducing JSON (Hämtad: 2020-05-10)
https://www.json.org/json-en.html

[3] Laravels hemsida (Hämtad: 2020-02-03)
https://laravel.com/

[4] Laravels dokumentation (Hämtad: 2020-02-03)
https://laravel.com/docs/6.x/

[5] Laravel dokumentation: Restful controllers (Hämtad: 2020-02-03)
https://laravel.com/docs/5.1/controllers#restful-resource-controllers

[6] Laravel dokumentation: Migrations (Hämtad: 2020-02-03)
https://laravel.com/docs/7.x/migrations

[7] Laravel dokumentation: Routing (Hämtad: 2020-02-03)
https://laravel.com/docs/7.x/routing

[8] Laravel dokumentation: ORM-system (Hämtad: 2020-02-03)
https://laravel.com/docs/master/eloquent

[9] Representational State Transfer (REST) (Hämtad: 2020-05-10)
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[10] Bootstrap 4 dokumentation (Hämtad: 2020-03-10)
https://getbootstrap.com/docs/4.4/getting-started/introduction/

[11] jQuerys hemsida (Hämtad: 2020-03-10)
https://jquery.com/

[12] B. Straub, S. Chacon, *Pro Git*, Berkeley, CA, Apress; 2nd ed, 9781484200773, 2014

[13] L. Richardson, S. Ruby, *Restful Web Services*, Farnham, O'Reilly, 9780596529260 ,2007

[14] DataTables manual (Hämtad: 2020-03-27)
https://datatables.net/manual/

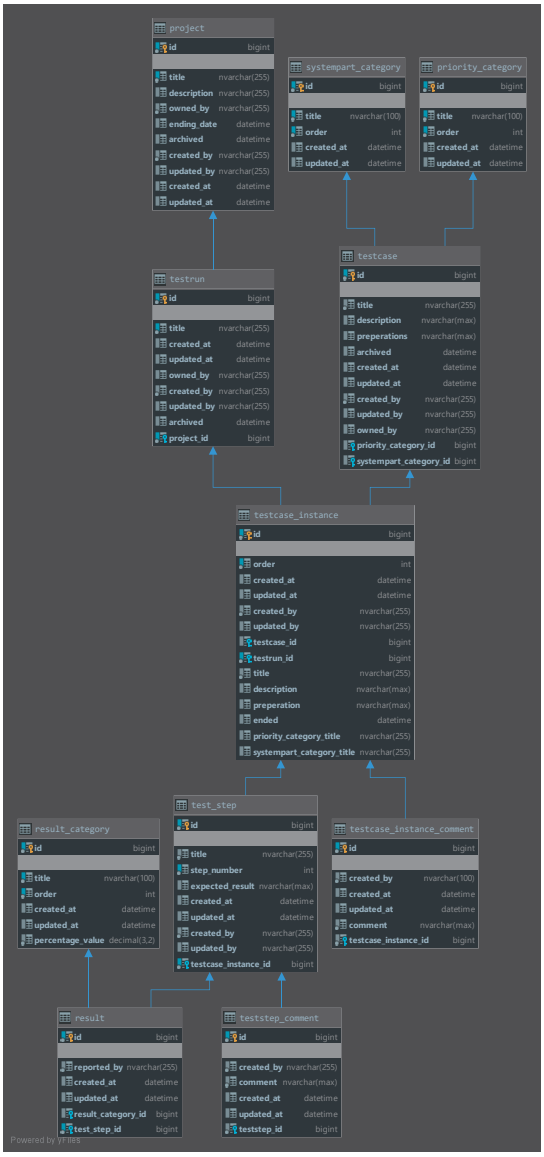
[15] Microsoft SQL Server hemsida (Hämtad: 2020-02-01)
https://www.microsoft.com/sv-se/sql-server/sql-server-2019

[16] About Node.js (Hämtad: 2020-01-29)
https://nodejs.org/en/about/

[17] CodeIngniters hemsida (Hämtad: 2020-02-03)
https://codeigniter.com/

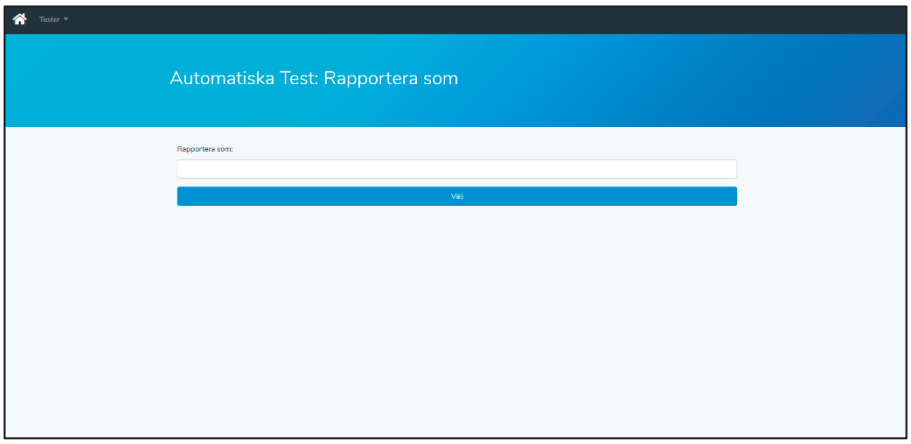
9. Appendix Databas

9.1. Databas diagram

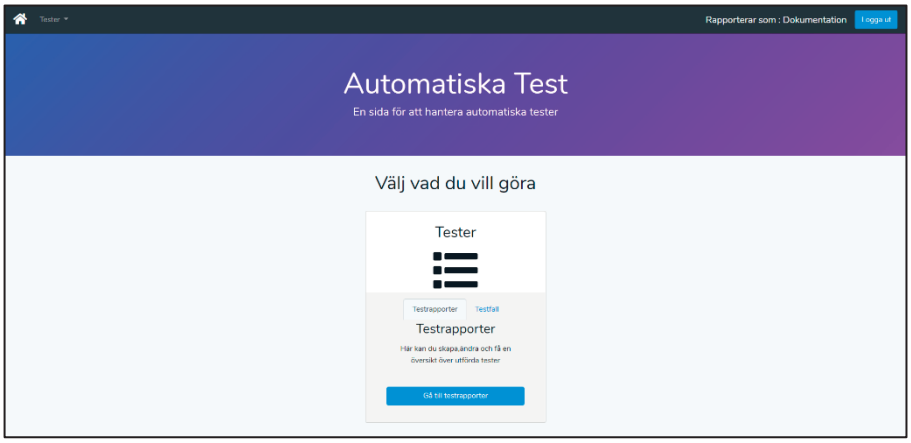


10. Appendix Webbgränssnitt

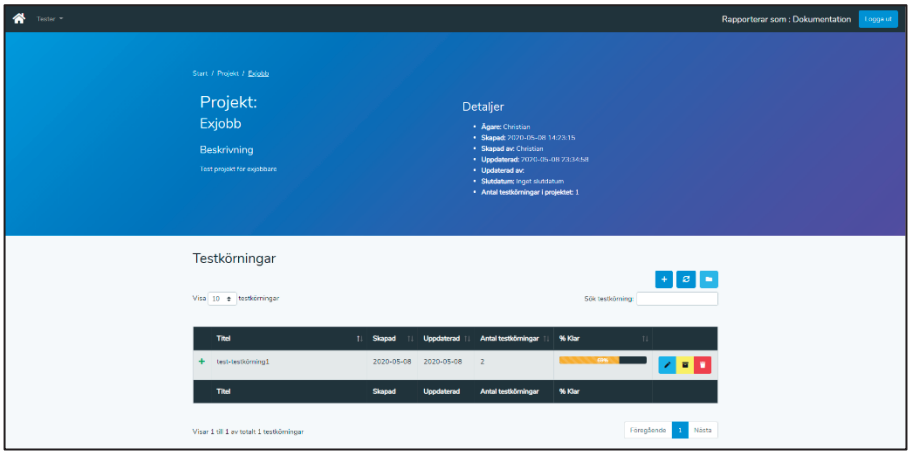
10.1. Inloggningsida



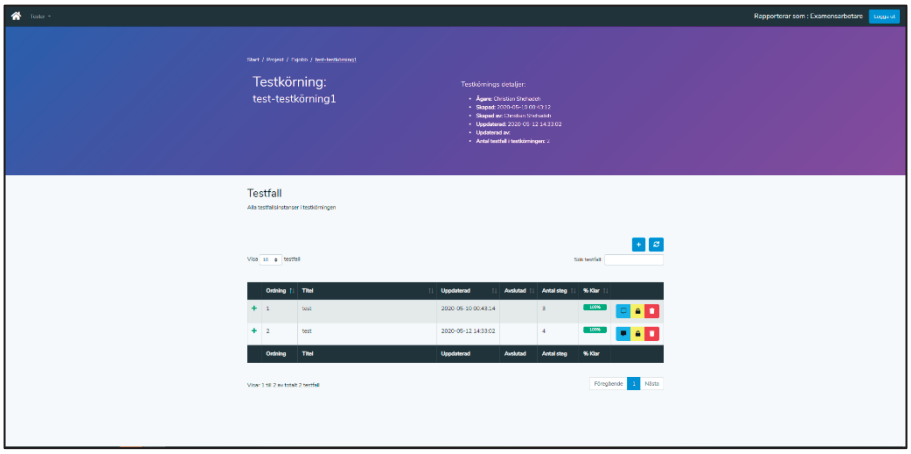
10.2. Startsida



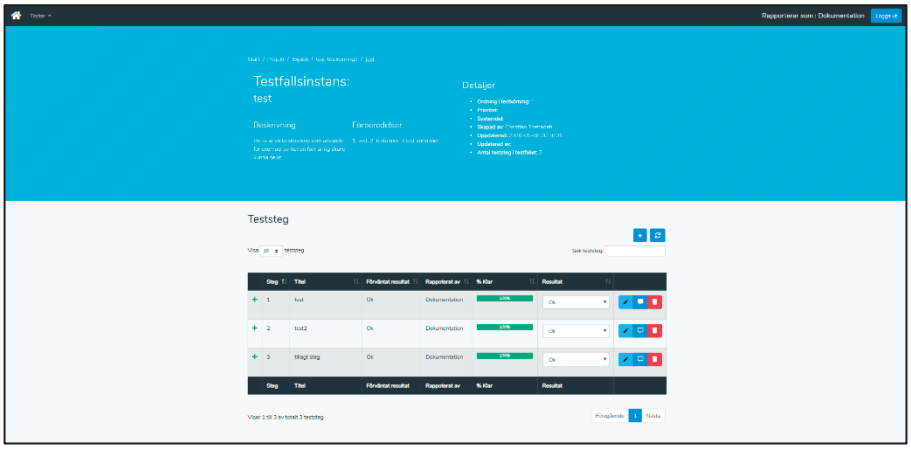
10.3. Testkörningsida



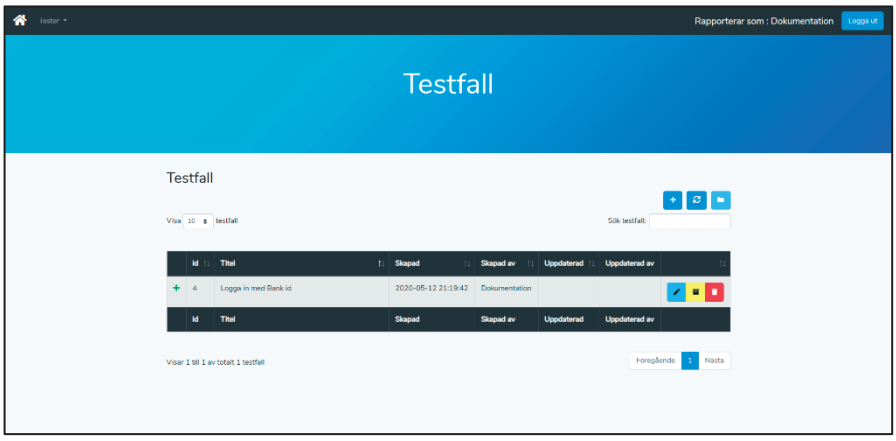
10.4. Testfallsinstanssida



10.5. Teststegsida



10.6. Testfallsida



Printed by Tryckeriet i E-huset, Lund 2020

LUND
UNIVERSITY

Series of Bachelor's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2020-759
http://www.eit.lth.se