Utveckling av system för hantering av arbetsbeskrivningar

Christer Sund

Examensarbete för ingenjörs (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2016

EXAMENSARBETE

Författare: Christer Sund

Utbildningsprogram och ort: Informationsteknik Vasa

Handledare: Susanne Österholm

Titel: *Utveckling av system för hantering av arbetsbeskrivningar*

Datum: 26.4.2016 Sidantal: 26 Bilagor: 1

Detta examensarbete har utförts på begäran av Sparal Oy Ab. Uppgiften var att utveckla en mobil applikation samt ett webbgränssnitt för hantering av arbetsbeskrivningar.

Applikationen utvecklades med HTML, JavaScript och CSS i ramverket Apache Cordova och webbgränssnittet utvecklades med HTML, JavaScript, PHP och CSS i ramverket CodeIgniter.

Resultatet blev ett system som ersätter företagets arbetsbeskrivningar på papper.

Språk: svenska Nyckelord: webbdesign, mobil applikationsutveckling

---

BACHELOR'S THESIS

Author:	Christer Sund
Degree Programme:	Information Technology, Vaasa
Supervisor:	Susanne Österholm

Title: *Development of System for the Handling of Work Descriptions*

Date: 26.4.2016 Number of pages: 26 Appendices: 1

This thesis was made on the request of Sparal Oy Ab. The task was to develop a mobile application together with a web interface for the handling of work descriptions.

The application was developed with HTML, JavaScript and CSS in the Apache Cordova framework and the web interface was developed with HTML, JavaScript, PHP and CSS in the CodeIgniter framework.

The result is a system replacing the company's work descriptions on paper.

Language: Swedish Key words: web design, mobile app development

---

OPINNÄYTETYÖ

Tekijä:	Christer Sund
Koulutusohjelma ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaaja:	Susanne Österholm

Nimike: *Työselostehallintajärjestelmän kehittäminen*

Päivämäärä: 26.4.2016 Sivumäärä: 26 Liitteet: 1

Tämän opinnäytetyön toimeksiantaja on Sparal Oy Ab. Tehtävä oli kehittää mobiiliapplikaatio sekä käyttöliittymä työselosteiden käsittelyä varten.

Applikaatio kehitettiin HTML-, JavaScript- ja CSS-ohjelmointikielillä Apache Cordovassa. Rajapinta kehitettiin HTML-, JavaScript-, PHP- ja CSS-ohjelmointikielillä CodeIgniterissa.

Tulos on järjestelmä, joka korvaa yhtiön paperiset työselosteet.

Kieli: ruotsi Avainsanat: verkkosivujen suunnittelu, mobiiliohjelmisto

---

Innehållsförteckning

Figurförteckning

Kodexempelsförteckning

Bilagor

1. Inledning

Uppgiften gick ut på att utveckla ett system för hantering av arbetsbeskrivningar. För att svara på företagets behov behövde systemet bestå av en mobil applikation och ett webbgränssnitt.

1.1. Uppdragsgivare

Sparal Oy Ab är ett privat företag som utför rivnings-, diamantborrnings- och diamantsågningsarbeten. Företaget är grundat år 1986 och är beläget i Vasa, men utför arbeten runtom i Finland och har även arbetat i Sverige. År 2005 köptes företaget upp av grundarens söner. Företaget har sedan dess expanderat mycket och antalet anställda har ökat från 4 till ca 20 personer. Det betyder att mera arbeten utförs och behovet av ett nytt system ökar.

Figur 1: Exempel på rivningsarbeten.

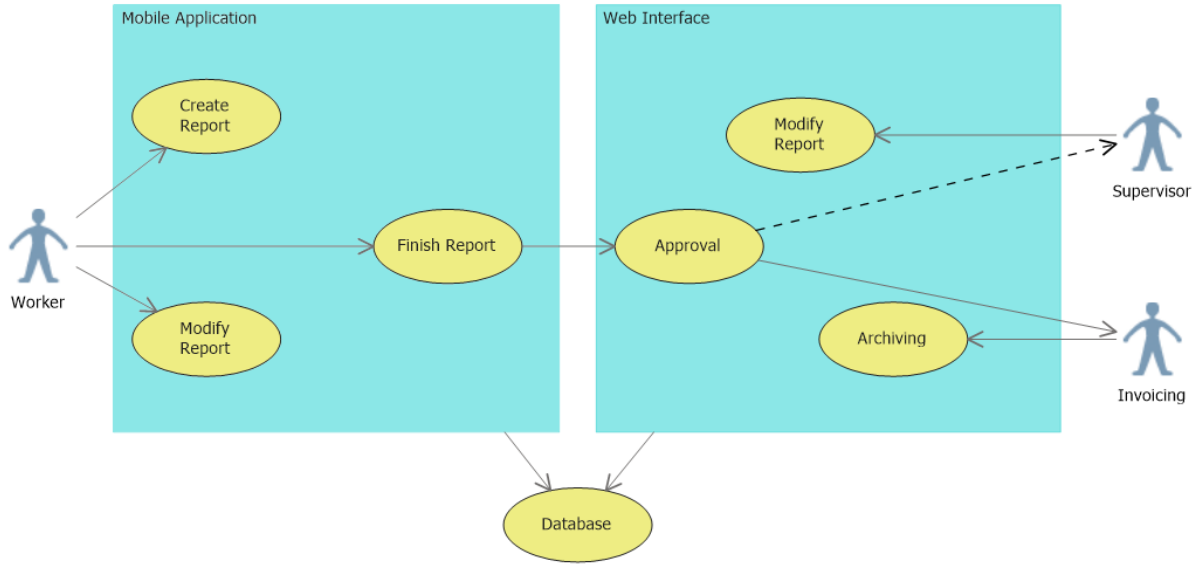
1.2. Bakgrund

För varje arbete och projekt som de anställda arbetar med ska en arbetsbeskrivning fyllas i. På denna blankett ska det framkomma information om kunden, dvs. namn, faktureringsadress, plats för utförande, kontaktperson och eventuella beställningskoder. Beskrivningen ska dessutom innehålla de arbetsmoment som utförts. Med borrning som exempel ska det framkomma typen av borrning (vägg, golv mm.), antal borrningar, borrens tjocklek, längden eller djupet på borrningen samt eventuella kommentarer. Ett exempel på en sådan blankett kan ses i *bilaga 1*. Allt detta har hittills gjorts på papper. När ett arbete eller projekt är slutfört och godkänt lämnas blanketten in för fakturering.

1.3. Uppgift

Jag har jobbat med fakturering på Sparal i ett år. Under denna tid insåg jag att det fanns ett stort behov av ett bättre och snabbare system för hantering av arbetsbeskrivningarna. Jag framförde ett förslag till arbetsgivaren om att låta mig skapa detta system som mitt examensarbete, vilket ansågs vara en bra idé.

Syftet med detta examensarbete är att utveckla ett system som ersätter den gamla processen för arbetsbeskrivningar. Istället för att skriva ner uppgifterna på papper, ska man med en mobil applikation kunna knappa in utförda arbetsmoment. Eftersom inga blanketter lämnas in till faktureringen, behövs dessutom ett gränssnitt för att kunna avläsa samt vid behov skriva ut arbetsbeskrivningarna. Gränssnittet ska vara tillgängligt även om man inte har åtkomst till en arbetsdator. Alla anställda använder sig av smarttelefoner i arbetet, men eftersom inte alla arbetare har telefoner med samma operativsystem så behövdes en applikation som är kompatibel med flera plattformar.



Figur 2: Use Case diagram.

I *Figur 2* ser man ett Use Case diagram över processen för arbetsbeskrivningar och fakturering. Arbetaren, som då använder den mobila applikationen, skapar en ny arbetsbeskrivning och vid behov ändrar innehållet. Varje ändring uppdaterar databasen. När arbetet är slutfört och har skickats vidare måste en förman godkänna det förrän det är redo att faktureras. Då arbetsbeskrivningarna har fakturerats arkiveras de i databasen.

2. Tekniker

För att det ska vara möjligt att använda den mobila applikationen med både iOS- och Android-telefoner användes ramverket Apache Cordova, vilket tillåter applikationen att utvecklas med HTML, CSS och JavaScript. Webbgränssnittet utvecklades med dessa tre språk samt PHP för att man ska kunna komma åt det varifrån som helst, även från en telefon om så behövs.

2.1. HTML

HTML (*HyperText Markup Language*) är en standard för att strukturera innehållet i webbsidor. Tillsammans med HTML kan man använda andra språk som t.ex. CSS och JavaScript för att förbättra utseende och funktionalitet på webbsidan. För att webbgränssnittet för hantering av arbetsbeskrivningar ska vara åtkomligt från vilken plats som helst och från vilken enhet som helst, t.ex. mobiltelefon och dator, valdes HTML som programmeringsspråk. I *Kodexempel 1* ser man kod för den mobila applikationen. [1]

Kodexempel 1: HTML.

```
<div data-role="page" id="mainPage" data-theme="a" class="page">
    <div data-role="header" class="header" data-position="fixed" data-tap-toggle="false">
        <h1>Työselosteet</h1>
        <button id="settingsButton" data-theme="a" class="ui-btn-left" data-icon="gear"></button>
        <button id="newButton" data-theme="b" class="ui-btn-right">Uusi</button>
    </div>
    <div data-role="content">
        <input data-type="search" id="rFilter" placeholder="Etsi"/>
        <ul id="reportList" data-role="listview" data-inset="true" data-filter="true" data-input="#rFilter"></ul>
    </div>
    <div data-role="footer" class="footer" data-position="fixed" data-tap-toggle="false">
        <div data-role="navbar">
            <ul>
                <li><a id="ongoing" class="statusListing ui-btn ui-btn-active" href="#" value=1>AVOIMIA</a></li>
                <li><a id="finished" class="statusListing ui-btn" href="#" value=2>VALMIIT</a></li>
            </ul>
        </div>
    </div>
</div>
```

I *Kodexempel 1* ses HTML-koden för startsidan, där olika arbetsbeskrivningar listas. Koden är skriven för den mobila applikationen, men fungerar också i en vanlig webbläsare.

2.2. CSS

CSS (*Cascading Style Sheets*) används i samband med HTML för att definiera hur en hemsida ska presenteras som t.ex. färg och storlek på text. Webbgränssnittet använder sig av Bootstrap CSS. Bootstrap är ett HTML, CSS och JavaScript ramverk som kan användas som grund för responsiva webbsidor. Med responsiv webbdesign anpassar sig webbsidan till den enhet man surfar på, på olika sätt som t.ex. fontstorlek och placering av element. Den mobila applikationen använder sig av jQuery Mobiles CSS (se kapitel 2.4.1). *Kodexempel 2* är ett exempel på CSS för webbgränssnittets menyknappar. [1][2][3]

Kodexempel 2: CSS.

```
.nav-tabs li a
{
    display: block;
    float: left;
    height: 37px;
    font-weight: bold;
    line-height: 37px;
    text-decoration: none;
    color: #646464;
    padding: 0 35px;
}

.nav-tabs li a:hover
{
    background: linear-gradient(#67c8ff, #e3e3ff);
    cursor:pointer;
}

.nav-tabs li a.active
{
    color: #454545;
    background: linear-gradient(#67c8ff, #e3e3ff);
    border-bottom:solid 1px #ddd;
}
```

Den CSS som syns i *Kodexempel 2* beskriver storlek, placering och färg för menyknapparna samt hur de ser ut när man håller musen över dem och när de är aktiva dvs. när de blivit valda.

2.3. PHP

PHP: Hypertext Preprocessor är ett skriptspråk som exekveras på serversidan. Med PHP kan man t.ex. hämta data från en databas för att sedan ladda datan i webbläsaren eller från ett formulär skicka data till databasen. För att kunna använda PHP behöver det vara installerat på servern där det används. Det finns flera alternativ till PHP som t.ex. Ruby eller Python, men PHP är mer än tillräckligt för den funktionalitet som behövs för gränssnittet. PHP-koden kan skrivas i en egen fil eller mixas in med HTML-kod. I *Kodexempel 3* finns ett exempel på det senare. [4]

Kodexempel 3: PHP.

```
<form id="reportf" method="post" action="">
    <input type="hidden" name="id" id="formId" value="<?php echo $form['Id']; ?>">
    <label>Tilaaja :</label>
    <input type="text" name="contact" value="<?php echo $form['Contact']; ?>">
    <label>Tilaus no :</label>
    <input type="text" name="code" value="<?php echo $form['Code']; ?>">
    <label>Työmaa/Osoite :</label>
    <input type="text" name="workplace" value="<?php echo $form['WorkPlace']; ?>">
    <label>Asiakas :</label>
    <input type="text" name="customer" value="<?php echo $form['Customer']; ?>">
    <label>Laskutusosoite :</label>
    <input type="text" name="invoiceaddress" value="<?php echo $form['InvoiceAddress']; ?>">
    <label>Työnsuorittaja :</label>
    <input type="text" name="workers" value="<?php echo $form['Workers']; ?>">
</form>
```

I *Kodexempel 3* syns koden för ett enkelt formulär i HTML. När sidan laddas används PHP för att fylla textfälten med värden, som är hämtade ur en matris med data från databasen.

2.4. JavaScript

JavaScript är ett scriptspråk som används för att inkludera dynamisk funktionalitet i en webbsida. Med JavaScript kan man lägga till händelser för t.ex. när man trycker på en knapp. Medan PHP exekveras på servern så körs JavaScript främst i klientens webbläsare. En stor del av webbsidor på internet använder sig av JavaScript och alla moderna webbläsare stöder språket. [5]

2.4.1. JQuery/jQuery Mobile

jQuery är ett JavaScript-bibliotek som gör det möjligt att med mindre kod uppnå samma funktionalitet som man gör med enbart JavaScript. JQuery Mobile är ett ramverk för utveckling av responsiva webbsidor och mobila applikationer. Ramverket använder HTML5, CSS3 och jQuery, vilka de flesta smarttelefoner stöder. I *Kodexempel 4* visas jQuery-funktioner för den mobila applikationens startsida. [6]

Kodexempel 4: jQuery.

```
$(document).on( "pagecontainerbeforeshow", function() {
    activePageId = $('body').pagecontainer('getActivePage').prop('id');

    switch(activePageId) {
        case 'mainPage':
            $("#reportList").empty();
            var status = 1;
            loadList(status);
            break;

        case ...
    }
});

$("#newButton").on("tap", function () {
   $("body").pagecontainer("change","#newPage");
});

$('#reportList').on("tap", 'li a', function() {
    rId = $(this).attr('value');

    if($("#ongoing").hasClass('ui-btn-active'))
    {
        $("body").pagecontainer("change","#reportPage");
        statId = 1;
    }
    else
    {
        $("body").pagecontainer("change","#finishedPage");
        statId = 2;
    }
});

$(".statusListing").on("tap", function () {
    $("#reportList").empty();
    var status = $(this).attr('value');
    loadList(status);
});
```

Med koden i *Kodexempel 4* hämtas data från databasen, med hjälp av AJAX, när startsidan öppnas och listar sedan datan på sidan. Det finns funktioner som kör när användaren trycker på knappar eller listobjekt.

2.4.2. AJAX

Med AJAX (*Asynchronous JavaScript and XML*) kan webbsidor och applikationer hämta data i bakgrunden och sedan visa denna data utan att behöva ladda om sidan. AJAX-funktionen gör en förfrågan till servern som då skickar tillbaka önskad data. Med denna lösning kan man t.ex. uppdatera listor i realtid. [7]

Kodexempel 5: AJAX

```
function loadList(status)
{
    var output = $('#reportList');
    $.ajax({
        url: 'http://localhost/ajax/test.php/?id=' + status,
        dataType: 'jsonp',
        jsonp: 'jsoncallback',
        timeout: 5000,
        success: function(data){
            $.each(data, function(i,item){
                var forms = "<li><a href='#' value='"+ item.Id +"'>"+ item.Customer + " - " +
                 item.WorkPlace + " - " + item.Workers + "</a></li>";
                output.append(forms).listview('refresh');
            });
        },
        error: function(){
            output.text('There was an error loading the data.');
        }
    });
}
```

I *Kodexempel 5* ser man funktionen som används för att ladda in listan i den mobila applikationens startsida. Funktionen skickar en förfrågan till servern där PHP hanterar datan och skickar tillbaka en lista på arbetsbeskrivningar.

2.5. MySQL

MySQL (*My Structured Query Language*) är en databashanterare som använder sig av språket SQL. SQL är ett kommandospråk som används främst för att hantera data i en databas. MySQL är en av de populäraste databashanterarna. Som nämnts i kapitel 2.3 används PHP för att hämta data till webbgränssnittet, men PHP kan inte hämta data direkt ur databasen utan hjälp av SQL och en databashanterare som t.ex. MySQL. [4][8]

Kodexempel 6: MySQL.

```
$conn = new mysqli($server, $username, $password, $database);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$stmt = $conn->prepare("INSERT INTO forms (Contact, Code, WorkPlace,
Customer, InvoiceAddress, Workers, StartDate) VALUES (?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param("sssssss", $contact, $code, $workplace, $customer,
$invoiceAddress, $workers, $startDate);

$contact = $_POST["contact"];
$code = $_POST["code"];
$workplace = $_POST["workPlace"];
$customer = $_POST["customer"];
$invoiceAddress = $_POST["invoiceAddress"];
$workers = $_POST["workers"];
$startDate = date("Y-m-d");
$stmt->execute();

echo "Report added successfully";

$stmt->close();
$conn->close();
```

I *Kodexempel 6* ser man exempel på kod var data från ett formulär i den mobila applikationen, har skickats med hjälp av en AJAX-funktion till servern och sparas i databasen med hjälp av PHP och MySQL.

2.6. Apache Cordova

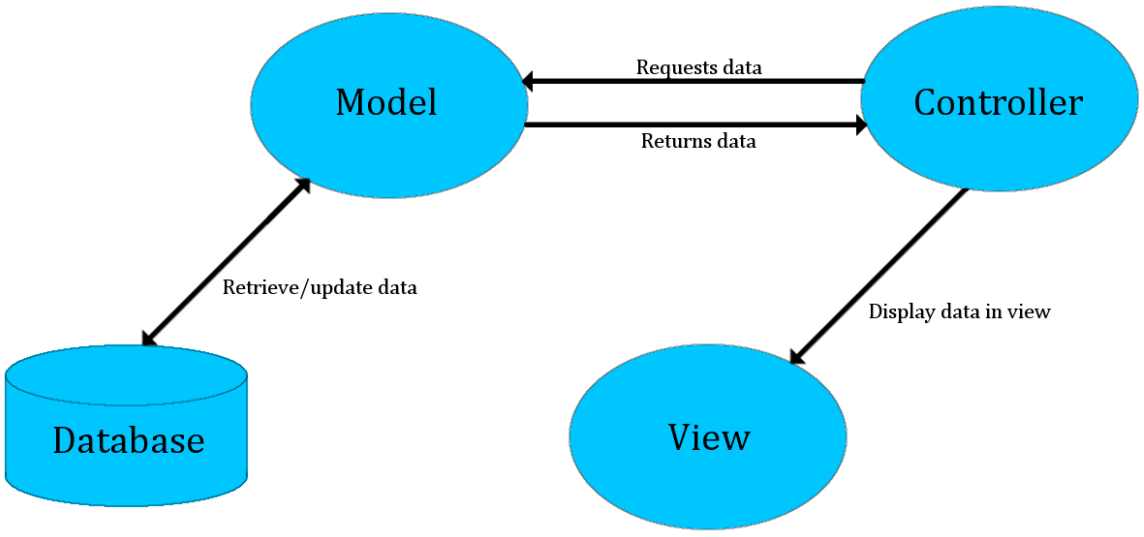
Apache Cordova är ett ramverk för utveckling av mobila applikationer. Ramverket tillåter applikationerna att använda HTML, CSS och JavaScript tillsammans med insticksmoduler. Detta behövs för att kunna använda funktioner för diverse hårdvara i enheten, som t.ex. kamera, GPS eller accelerometer, istället för att behöva förlita sig på de plattformsspecifika APIn som t.ex. Android och iOS har. API (*Application Programming Interface*) är ett bibliotek som definierar hur program kommunicerar med varandra. [9][10]

Med Apache Cordova kan man genom att mata in kommandon i t.ex. Windows kommandotolk kompilera koden och köra applikationen på en telefon som är

kopplad till datorn. Man kan också köra applikationen i en emulator på samma sätt, ifall en sådan är installerad.
Applikationens gränssnitt beter sig som en responsiv webbsida, men istället för att surfa till en sida på internet visas det med telefonens inbyggda webbläsare. Med jQuery Mobile kan man implementera alla nödvändiga funktioner och data kan sparas lokalt på telefonen eller med AJAX hämtas från och sparas till en server.

2.7. CodeIgniter Web Framework

CodeIgniter är ett PHP ramverk som är baserat på MVC-modellen. MVC är ett designmönster som separerar koden i delarna *Model, View* och *Controller*. På detta sätt kan man ändra i presentationsdelen utan att det uppstår fel i datahanteringen. [11][12]



Figur 3: MVC-modellen.

I F*igur 3* ser man hur de olika delarna i modellen kommunicerar. Controller gör förfrågningar till Model, som hämtar data från databasen och vidarebefordrar resultatet till Controller, som sedan skickar datan till View, som är den visuella delen.

Med ett ramverk som CodeIgniter behöver man inte skriva all kod från grunden utan man kan använda sig av färdiga funktioner och hjälpmedel som förenklar kommunikationen mellan sidorna och databasen. CodeIgniter är ett litet, snabbt och lättkonfigurerat ramverk som har bra dokumentering. Detta gör att man bara efter en snabb introduktion kan komma igång med projekt. Ramverket är utmärkt för mindre projekt, medan ett ramverk som t.ex. Zend PHP Framework kan vara mer passande för större projekt.

3. Utförande

I detta kapitel tas det upp hur arbetet planerats, vilka olika verktyg som använts för att utveckla den mobila applikationen och webbgränssnittet, hur de är uppbyggda och fungerar samt information om testningsfasen.

3.1. Planering

Innan det praktiska arbetet inleddes planerades systemets nödvändiga funktionalitet tillsammans med Sparals arbetsledning och fakturering. Eftersom ett system för arbetsbeskrivningar redan existerade behövdes själva innehållet inte planeras från grunden.

Från början var det bestämt att en mobil applikation för både Android och iOS skulle utvecklas. Det behövdes även ett användargränssnitt, som faktureringen kan använda för att snabbt och enkelt fakturera projekt vartefter de är slutförda och godkända.

Den mobila applikationen utvecklades i ramverket Apache Cordova för att kunna skriva programmet i HTML och jQuery, vilket gör att samma kod kan användas för att skapa applikationer till både iOS och Android. Med Cordova finns också möjligheten att enkelt överföra applikationen till Windows Phone, om det i framtiden skulle finnas behov.

Kravet på användargränssnittet var att förmännen ska kunna gå igenom och godkänna färdiga arbetsbeskrivningar och att faktureringen kan hantera dem så fort de är redo att faktureras. Det finns många möjligheter för att utveckla ett användargränssnitt, t.ex. i ett av C-språken, men man valde att skapa ett webbgränssnitt. Med ett webbgränssnitt har användaren åtkomst från vilken enhet som helst med internetuppkoppling, och ingen installation krävs.

3.2. Verktyg

Projektet har kodats i Notepad++ som är ett gratis textediteringsprogram utvecklat av Don Ho. Skillnaden mellan Microsofts eget textediteringsprogram Notepad och Notepad++ är att den senare har många funktioner och stöd för flera olika

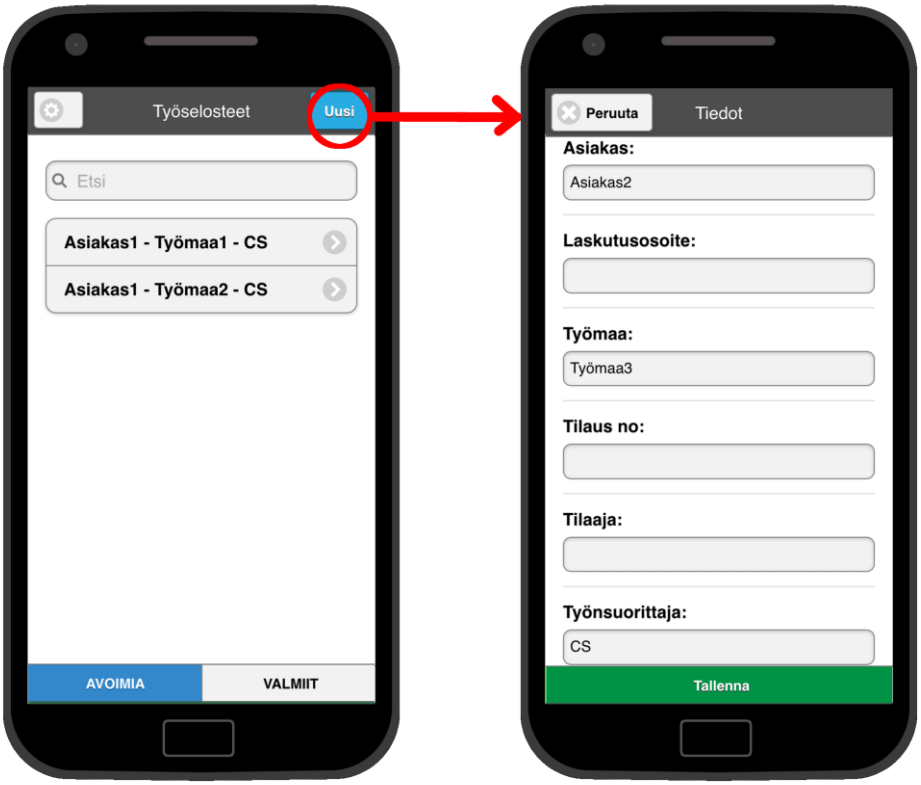
programmerings- och skriptspråk. Som många andra editeringsprogram har Notepad++ också färgad kodtext vilket gör koden mera lättläslig. [13]

XAMPP (*Cross-platform, Apache, MariaDB, PHP and Perl)* är en lösning som tillåter att man kör en lokal, virtuell webserver på sin egen dator. Med denna lösning kan man testa sitt program snabbt och enkelt utan att behöva en äkta webserver. XAMPP använder sig av Apache HTTP Server vilket är ett webbserverprogram som hanterar förfrågningar via HTTP. Som databas kan man använda MariaDB eller MySQL. XAMPP inkluderar också phpMyAdmin som är ett administrationsgränssnitt för databasen. Med en lokal server kan man också testa kommunikationen mellan applikationen i telefonen och webbservern bara genom att vara på samma nätverk. [14]

För testning av projektet användes en gratis server från *Amazon Web Services* (AWS). En central del av AWS är *Elastic Compute Cloud* (EC2) som ger möjligheten att konfigurera virtuella maskiner, eller s.k. instanser, för att använda som servrar och installera operativsystem och nödvändigheter som t.ex. MySQL. För detta slutarbetes testning installerades ett Linux operativsystem, Apache HTTP Server och MySQL. För administration av databasen användes HeidiSQL, som är ett verktyg för hantering av databaser baserade på SQL, MySQL och PostgreSQL. För webbgränssnittets filhantering användes WinSCP som är ett FTP verktyg (*File Transfer Protocol*). [15][16][17]

3.3. Mobil applikation

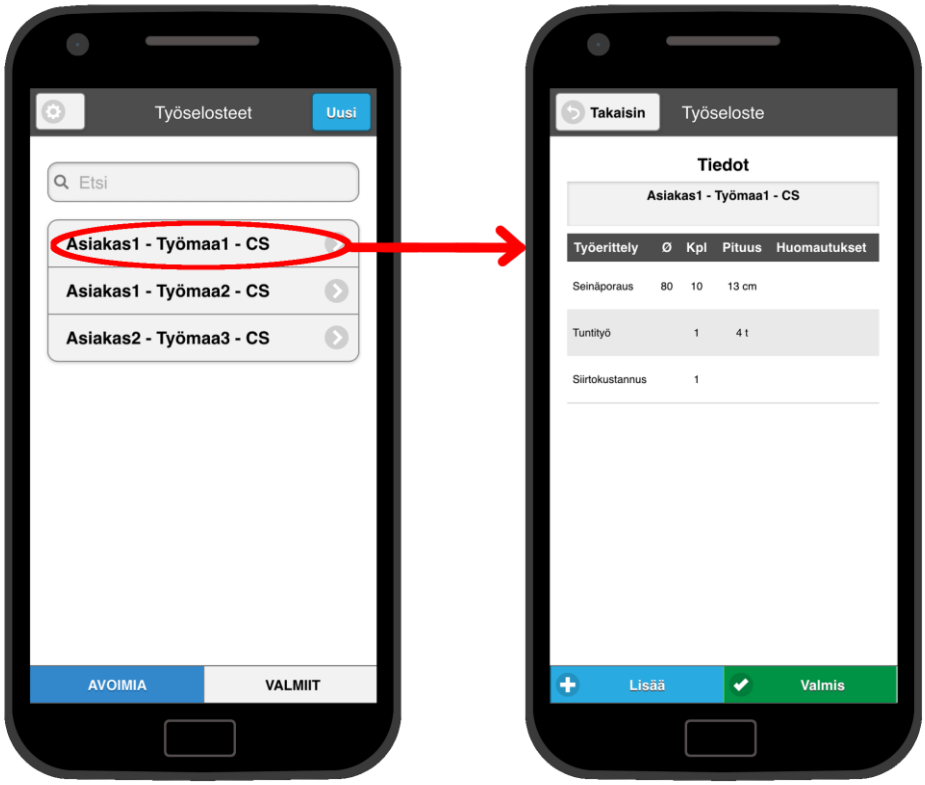
Den huvudsakliga produkten i detta projekt är en mobil applikation som används av personalen för att skapa nya arbetsbeskrivningar och mata in arbetsmoment som utförts i projektet. Mer än en arbetare kan jobba på ett projekt och tidigare har det varit en person som skriver ner alla arbetsmoment på pappersblankett, efter att de är utförda. Med hjälp av applikationen kan arbetarna vid behov mata in det egna arbetsmomentet, genast efter utförandet, på en och samma arbetsbeskrivning.



Figur 4: Applikation, startsida.

I *Figur 4* ser man sidan som visas när applikationen öppnas. När sidan visas hämtas det in en lista över alla öppna arbetsbeskrivningar. Med hjälp av knapparna på botten av skärmen kan man välja om listan ska visa öppna (Avoimia) arbetsbeskrivningar eller slutförda (Valmiit). För varje arbetsbeskrivning visas det i listan kundens namn, var arbetet utförs och vem eller vilka som utför arbetet. Listan kan filtreras genom att skriva in sökord som motsvarar dessa.

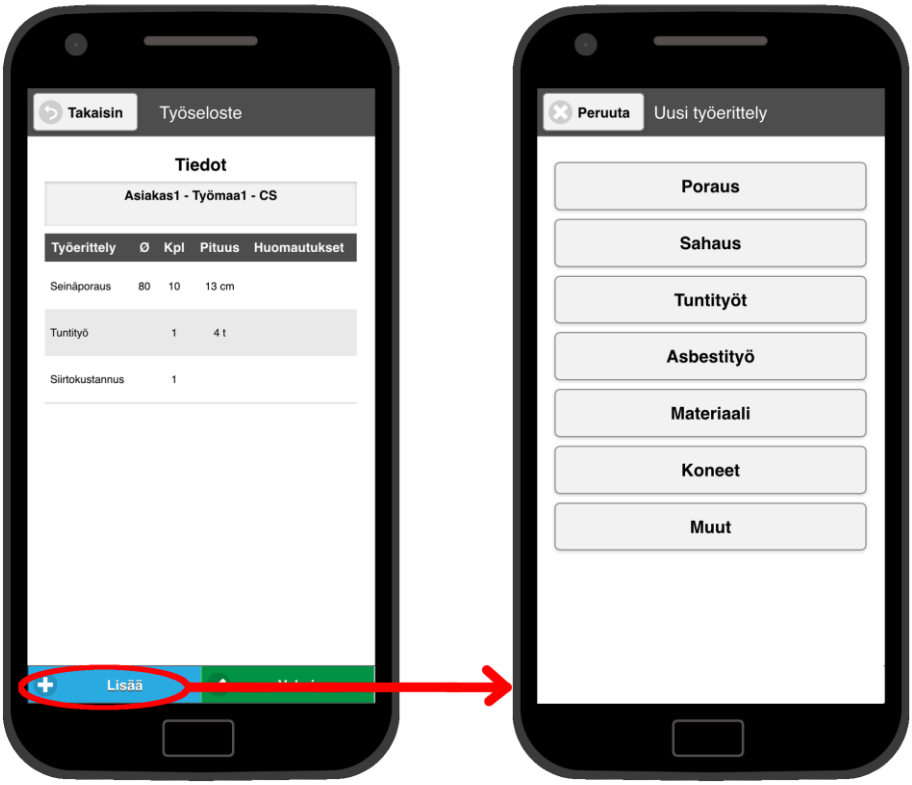
När användaren trycker på knappen ”Uusi” öppnas en ny sida med ett formulär för kundens och arbetets information. Användaren matar in information som t.ex. faktureringsadress, beställningsnummer mm. I vissa fall behövs inte all information, t.ex. om kunden fakturerats förut behövs inte en faktureringsadress matas in, men kundens namn och platsen för arbetet krävs alltid. Om användaren matat in sitt namn vid ibruktagning av applikationen fylls namnet automatiskt i fältet för vem som utför arbetet när formuläret visas. När användaren matat in all nödvändig information och sparar den skickas informationen till servern som sparar det i databasen, varefter huvudsidan öppnas och listan uppdateras med den nya arbetsbeskrivningen.



Figur 5: Applikation, arbetsbeskrivningens sida.

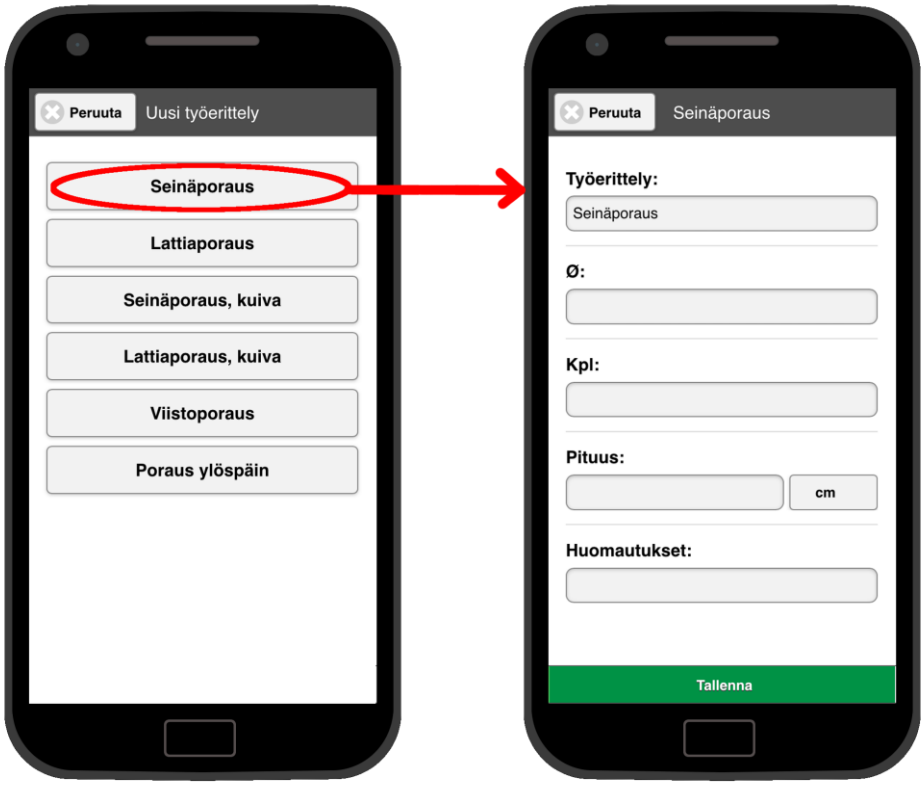
I *Figur 5* ser man vad som händer då användaren väljer en arbetsbeskrivning i listan. En ny sida öppnas för den valda arbetsbeskrivningen och på denna sida finns informationen som användaren fyllt i vid skapandet av beskrivningen och en tabell som listar alla arbetsmoment för vald arbetsbeskrivning, samt två knappar som är för att lägga till nya arbetsmoment och att markera projektet som slutfört.

När användaren trycker på rutan med kundens information öppnas en ny sida där man kan redigera informationen vid behov. Ifall användaren väljer ett arbetsmoment i tabellen öppnas en ny sida där arbetsmomentet kan redigeras eller tas bort.



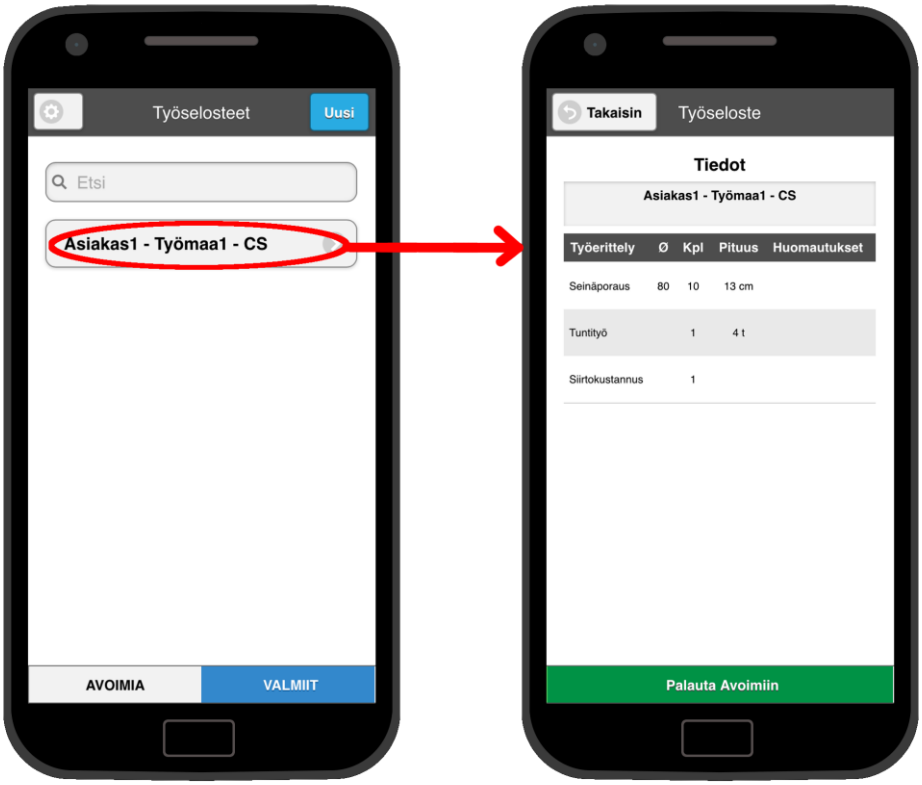
Figur 6: Applikation, nytt arbetsmoment.

I *Figur 6* ser man att när användaren trycker på knappen ”Lisää” för att lägga till ett arbetsmoment öppnas en sida med knappar som representerar kategorier för de olika arbetsmomenten. När man väljer en kategori byts knapparna ut med nya knappar som representerar arbetsmoment för den valda kategorin. Sammanlagt finns det ca 25 förinställda arbetsmoment att välja från, men om ett arbetsmoment ska matas in som inte har en knapp kan användaren välja en knapp för att skapa ett eget arbetsmoment.



Figur 7: Applikation, formulär för arbetsmoment.

I *Figur 7* ser man att vid val av arbetsmoment öppnas en sida där användaren matar in informationen för arbetsmomentet. Beroende på vilken kategori som valts visas endast nödvändiga fält, t.ex. för borrningar krävs endast diameter, stycke och längd medan timarbete behöver endast antal arbetare och timmar. Bredvid fältet för längd finns en knapp som öppnar en lista med olika enheter som metrar, timmar mm. I denna lista finns endast enheter som är tillåtna för det aktuella arbetsmomentet t.ex. för maskinhyror finns det endast timmar och dagar. Alla arbetsmoment har ett fält för eventuella anmärkningar. När användaren matat in all nödvändig information och sparar arbetsmomentet sparas det i databasen och projektsidan öppnas igen.



Figur 8: Applikation, slutförd.

När alla arbeten är genomförda och inmatade markeras arbetsbeskrivningen som slutförd. I *Figur 8* ser man startsidan där slutförda arbetsbeskrivningar listas. För användarna av applikationen är endast öppna arbetsbeskrivningar redigerbara, men vid tillfällen då ett arbetsmoment behöver läggas till en arbetsbeskrivning som är markerad som slutförd kan användaren markera den som öppen igen. Arbetsbeskrivningar som är slutförda och därefter godkända för fakturering kan inte längre redigeras eller visas i applikationen.

Eftersom applikationen inte distribueras via en online butik som Apple's AppStore eller Google Play var det nödvändigt med ett eget system för uppdatering av applikationen. Det löstes genom att då applikationen öppnas, köra en kontroll som från servern hämtar det aktuella versionsnumret och jämför det med nuvarande installationens versionsnummer. Om numrorna inte stämmer överens blir användaren noterad om en uppdatering och den nya installationen laddas ner automatiskt. Det går inte att fortsätta använda applikationen innan nyaste versionen är installerad.

3.4. Webbgränssnitt

Webbgränssnittet är baserat på MVC-modellen (se kapitel 2.7). *Model* kommunicerar med databasen dvs. hämtar data samt lägger in/uppdaterar data. *View* är den visuella delen som skriver ut data hämtad från Model. Controller bearbetar händelser och skickar kommandon till *Model* samt ändrar hur *View* presenteras. Som grund för webbsidan används ramverken CodeIgniter och Bootstrap.

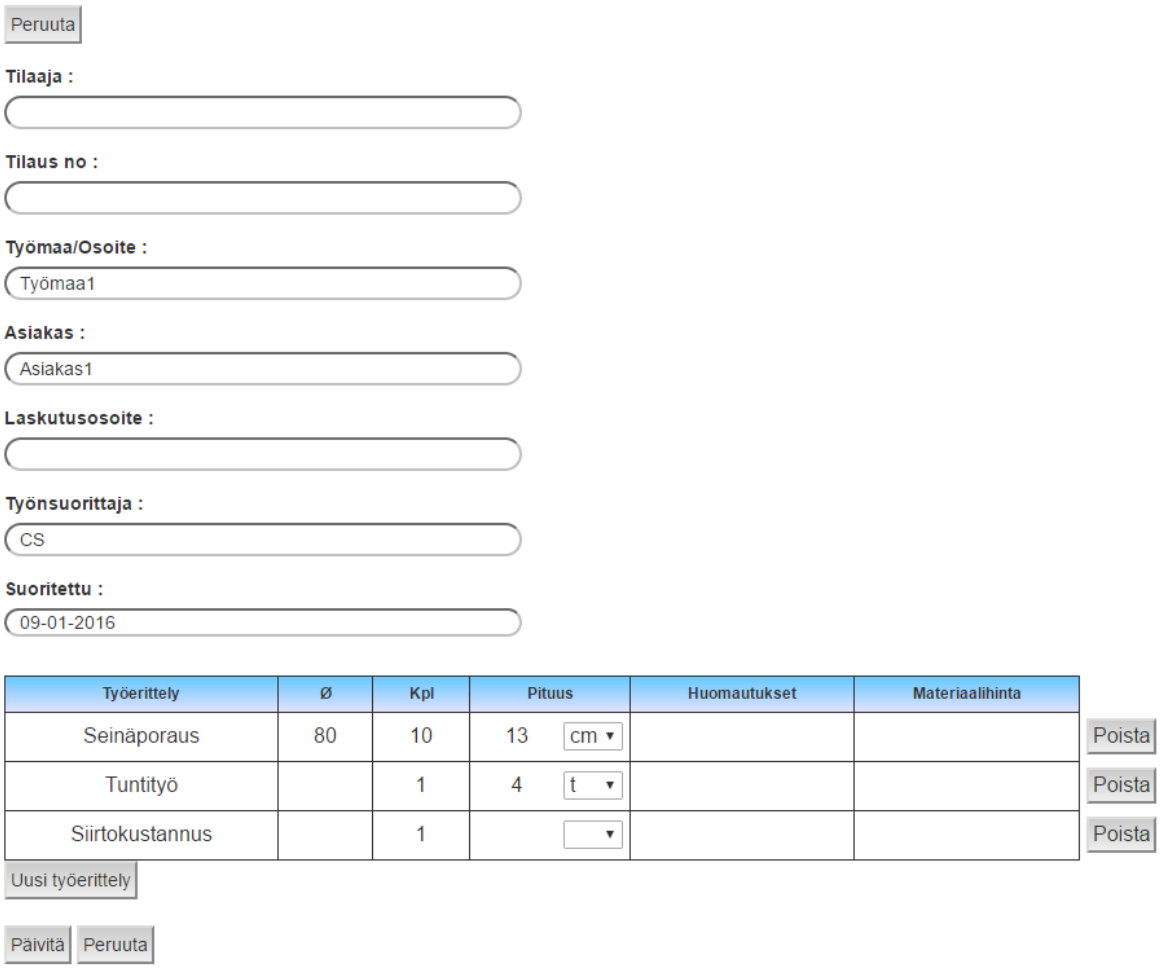
Webbgränssnittet är endast tillgängligt för förmännen och faktureringen. Gränssnittets startsida består av meny med knappar och en lista på arbetsbeskrivningar. Här kan man se informationen för beskrivningarna, skriva ut dem, redigera dem och vid behov ta bort dem ur listan. Om en beskrivning tas bort hamnar de i en särskild lista där de väntar på att raderas helt och hållet. Denna lista finns för att undvika att arbetsbeskrivningar tas bort i misstag.

Figur 9: Webbgränssnitt, startsida.

I *Figur 9* ser man hur startsidan ser ut. Med menyknapparna väljer man vilka arbetsbeskrivningar som visas i listan. Det finns fyra knappar som filtrerar listan enligt vilket läge arbetet är i (öppna, slutförda, godkända samt fakturerade). I Avoimia (Öppna) listas pågående arbeten. Här kan man läsa och redigera arbetsbeskrivningar, skapa nya samt godkänna dem som slutförda. Allt detta kan också göras i den mobila applikationen som arbetarna använder.

Figur 10: Webbgränssnitt, slutförda.

I *Figur 10* ser man att när arbeten är slutförda hamnar de i listan Valmiit (Slutförda) där de väntar på godkännande för fakturering, detta kan endast förmännen göra. Vid tillfällen när nya arbeten beställs till samma projekt kan man skicka beskrivningen tillbaka till listan för öppna projekt.

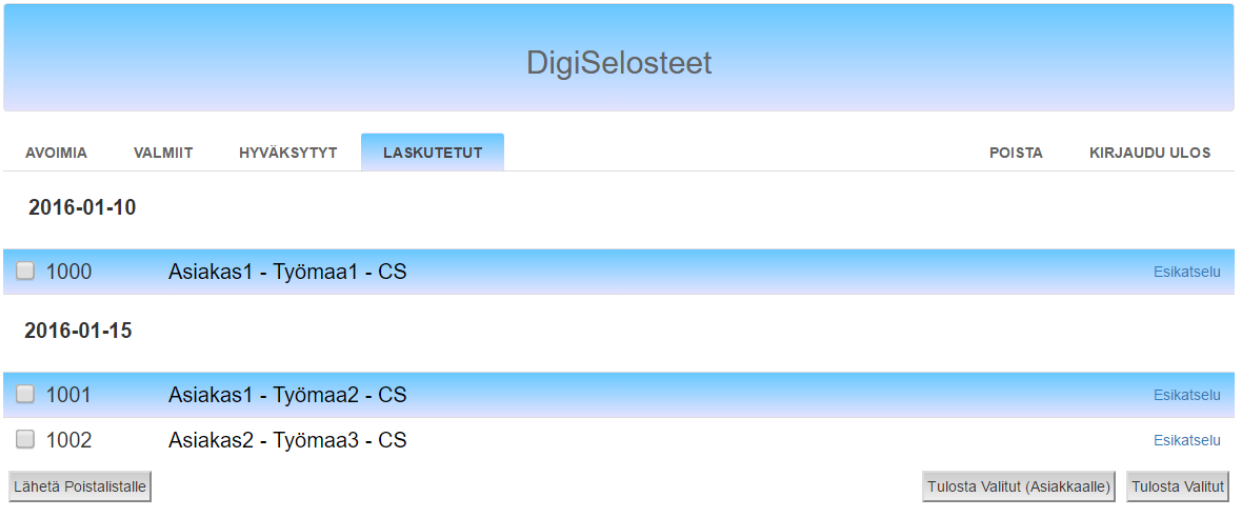


Figur 11: Webbgränssnitt, redigering.

Om förmännen har behov av att redigera arbetsbeskrivningen kan de lätt göra det genom att klicka på länken ”Muokkaa” vid beskrivningen i listan. I *Figur 11* ser man hur redigeringssidan ser ut. Denna sida har samma funktioner som den mobila applikationens sida för arbetsbeskrivningar, där man kan redigera information och lägga till/ta bort arbetsmoment.

Figur 12: Webbgränssnitt, godkända.

När en arbetsbeskrivning blivit godkänt för fakturering är förmännens del av processen klar. I *Figur 12* ser man listan Hyväksytyt (godkända) där det listas alla arbetsbeskrivningar som blivit godkända och väntar på fakturering. Faktureringen matar sedan in informationen i ett faktureringsprogram när ett projekt blivit fakturerat tilldelas arbetsbeskrivningen ett fakturanummer av den som fakturerat. Slutligen godkänns de fakturerade arbetsbeskrivningarna och flyttas till nästa lista.



Figur 13: Webbgränssnitt, fakturerade.

Efter att en arbetsbeskrivning blivit fakturerad hamnar de i listan Laskutetut (fakturerade), som ses i *Figur 13*. Här är de inte längre redigerbara utan kan bara läsas och skrivas ut.
Här sorteras och listas arbetsbeskrivningarna enligt vilket datum de blivit fakturerade.

3.5. Testning

När systemet var så långt utvecklat att alla preliminära krav uppfyllts och en tillfällig server blivit ibruktagen var det dags för testning.

För att testa systemet behövdes minst tre personer, en arbetare för att skapa arbetsbeskrivningar och mata in arbetsmoment med den mobila applikationen, en förman för att kontrollera och godkänna beskrivningar och en fakturerare som fakturerar de godkända arbetsbeskrivningar.

Under testningsfasen dök det upp diverse buggar i programmen som löstes vartefter de dök upp. De flesta buggar hade att göra med jQuery's händelser och datans visuella representation.

Under testningen visade det sig att många mänskliga fel kunde förekomma vid användning av den mobila applikationen, därför behövde flera funktioner åtgärdas och göras felsäkra samt begränsa hur många saker som kan hända när användaren rör vid pekskärmen, t.ex. att när man trycker på en knapp som byter sida registreras trycket på den sidan också.

4. Resultat och diskussion

I det här kapitlet beskrivs resultaten, problem som uppstått under examensarbetet och diskussion.

4.1. Resultat

Resultatet av examensarbetet blev en mobil applikation samt ett webbgränssnitt som uppfyllde alla preliminära krav och även de krav som dök upp under testningen av systemet. Den slutliga produkten togs i bruk på företagets server.

Genom att ersätta det gamla systemet, som endast använde sig av pappersblanketter, med ett digitalt system, förändras en del förhållanden för personalen. Pappersblanketter är inte längre nödvändiga. Med mobilen kan arbetarna snabbt mata in arbetsmoment i beskrivningar vartefter de utförts. Detta gör det enklare för förmännen att följa med i realtid hur arbetet framskrider, när projekten är slutförda och arbetsbeskrivningarna är färdiga. Därefter kan de fortare än tidigare godkänna arbetsbeskrivningarna för fakturering, då de inte behöver vänta på att blanketten förs till kontoret. Faktureringen är inte belägen på samma adress, vilket kan göra att det ibland tagit lång tid innan arbetsbeskrivningar blivit fakturerade. Nu kan beskrivningarna faktureras så fort de är godkända och det går inte onödig tid till att vänta på att arbetsbeskrivningarna skickas mellan olika platser.

4.2. Diskussion

Eftersom jag har jobbat på faktureringen i företaget hade jag från början en god förståelse över hur processen för arbetsbeskrivningarna fungerar. Arbetet var p.g.a. detta helt självständigt, vilket gjorde planeringen enklare och det var lättare att komma igång med projektet.

Alla programmeringsspråk och verktyg jag använt hade jag åtminstone en grundläggande förståelse för innan jag började. Jag anser att jag genom detta arbete har fått en hel del erfarenhet främst inom webbsideutveckling.

Valet av programmeringsspråk för den mobila applikationen gjordes med tanke på innehållet och framtida ändringar och uppdateringar. Objective-C för iOS och Java för Android hade varit alternativ, men med hjälp av Apache Cordova kunde jag utveckla applikationen i ett gemensamt språk för både iOS och Android. Om det i framtiden finns behov av att göra ändringar i programmet räcker det därför att koden ändras en gång för att uppdatera applikationen både för iOS och Android.

Det skulle ha varit möjligt att istället för ett webbgränssnitt för godkännande och fakturering skapa t.ex. en skrivbordsapplikation. Denna lösning kunde på många sätt ha fungerat lika bra som en webbsida. En skrivbordsapplikation hade dock begränsat användandet till datorer och hade krävt en installation eller körbar fil. För att möta kundens behov valde jag därför att skapa en webbsida, vilket gav mera möjligheter.

Kunden har varit nöjd med produkten och jag är själv nöjd med resultatet. Arbetet och utkomsten har varit vad jag förväntade mig då jag började med projektet.

Om det i framtiden skulle finnas behov för liknande system för andra blanketter eller dylikt, är det lätt att utöka funktionaliteten i programmen eller utveckla nya med detta examensarbete som grund.

Källförteckning

[1] HTML & CSS [Online]
https://www.w3.org/standards/webdesign/htmlcss [Hämtat: 14.02.2016].

[2] Bootstrap CSS [Online]
http://getbootstrap.com/css/ [Hämtat: 08.04.2016].

[3] *What is Responsive Web Design* [Online]
http://smallbiztrends.com/2013/05/what-is-responsive-web-design.html
[Hämtat: 08.04.2016].

[4] Welling, Luke & Thomson, Laura. (2009). *PHP and MySQL Web Development.* Pearson Education, Inc.

[5] *JavaScript – General Introduction* [Online]
http://www.quirksmode.org/js/intro.html [Hämtat: 16.02.2016].

[6] jQuery [Online]
https://jquery.com/ [Hämtat: 16.02.2016].

[7] Ajax [Online]
https://developer.mozilla.org/en-US/docs/AJAX [Hämtat: 16.02.2016].

[8] DB-Engines Ranking [Online]
http://db-engines.com/en/ranking [Hämtat: 17.02.2016].

[9] Apache Cordova [Online]
https://cordova.apache.org/ [Hämtat: 20.02.2016].

[10] *What Are APIs, And How Are Open APIs Changing The Internet* [Online]
http://www.makeuseof.com/tag/api-good-technology-explained/
[Hämtat: 20.02.2016].

[11] CodeIgniter Web Framework [Online]
http://www.codeigniter.com/ [Hämtat: 20.02.2016].

[12] *MVC Architecture* [Online]
https://developer.chrome.com/apps/app_frameworks [Hämtat: 20.02.2016].

[13] Notepad++ [Online]
https://notepad-plus-plus.org/ [Hämtat: 26.02.2016].

[14] XAMPP [Online]
https://www.apachefriends.org/index.html [Hämtat: 26.02.2016].

[15] EC2 – Amazon Web Services [Online]
https://aws.amazon.com/ec2/ [Hämtat: 02.03.2016].

[16] HeidiSQL [Online]
http://www.heidisql.com/ [Hämtat: 02.03.2016].

[17] WinSCP [Online]
https://winscp.net/eng/index.php [Hämtat: 02.03.2016].

TYÖSELOSTE
ARBETSBESKRIVNING

Työn Tilaaja - Beställare	
Tilaus no - Beställn. nr	
Työmaa - Arbetsplats	
Laskutusosoite - Faktureringsadress	

Työerittely Arbetsspecifikation	Ø	Pituus Längd	Kpl St	Huomautukset Anmärkningar

Kalustonsiirto - Förflyttning av redskap	Ajo km - Kör km
Odotusaika - Väntetid	Matka-aika - Resetid
Työnsuorittaja - Utfört.	Hyväksytään laskutettavaksi - Godkännes för fakturering