# Vidareutveckling av en webbportal i .NET miljö för parsning och generering av faktura

Jonas Mäkinen

Examensarbete

Informations- och medieteknik

2016

Jonas Mäkinen

| | |
|---|---|
| EXAMENSARBETE | |
| Arcada | |
| | |
| Utbildningsprogram: | Informations- och medieteknik |
| | |
| Identifikationsnummer: | 5845 |
| Författare: | Jonas Mäkinen |
| Arbetets namn: | Vidareutveckling av en webbportal i .NET, miljö för parsning och generering av faktura. |
| Handledare (Arcada): | Johnny Biström |
| | |
| Uppdragsgivare: | ABB Oy, (Magnus Söderström) |
| | |

Sammandrag:
Detta examensarbete fokuserar på vidareutveckling av en webbportal i .NET miljö. En webbportal hade tidigare utvecklats för parsning och generering av faktura. Den lösningen blev aldrig fullständig och fungerade inte som planerat. Syftet med detta arbete är att utveckla en icke fungerande webbportal till en fungerande version. Arbetet går in på de problem som ursprungliga webbportalen hade och visar hur de blev lösta. I resultaten kan man se de problem den ursprungliga portalen hade och hur skribenten gått till väga för att lösa dessa problem och resultat över utförandet. Med hjälp av en fungerande webbportal kan uppdragsgivaren använda det för rätt syfte, för parsning och generering av faktura.



| | |
|---|---|
| Nyckelord: | Utveckla, webbportal, C#, .NET, parsning, generering, servernivå, back-end, |
| Sidantal: | 38 |
| Språk: | Svenska |
| Datum för godkännande: | |

| DEGREE THESIS | |
|---|---|
| Arcada | |
| | |
| Degree Programme: | Information and Media Technology |
| | |
| Identification number: | 5845 |
| Author: | Jonas Mäkinen |
| Title: | Further development of a web portal with .NET framework for parsing and generating an invoice. |
| Supervisor (Arcada): | Johnny Biström |
| | |
| Commissioned by: | ABB Oy, (Magnus Söderström) |
| | |

Abstract:
This thesis focuses on further development of a web portal with .NET framework for parsing and generating invoices. A web portal had previously been developed, but it was not fully finished and did not work the way it should have. The purpose of this thesis is to develop a non-functioning web portal to a working version. The thesis goes in detail through the problems that the original portal had and finds solutions for these problems. The result of this thesis is a working web portal, which suites the need of the client. With the help of a working portal the customer can parse and generate invoices.



| Keywords: | Develop, web portal, C#, .NET, parser, generating, back-end |
|---|---|
| Number of pages: | 38 |
| Language: | Swedish |
| Date of acceptance: | |

| | |
|---|---|
| OPINNÄYTE | |
| Arcada | |
| | |
| Koulutusohjelma: | Tieto- ja mediatekniikka |
| | |
| Tunnistenumero: | 5845 |
| Tekijä: | Jonas Mäkinen |
| Työn nimi: | Verkkoportaalin jatkokehitys .NET ympäristössä, laskujen jäsentämiseen ja luomiseen. |
| Työn ohjaaja (Arcada): | Johnny Biström |
| | |
| Toimeksiantaja: | ABB Oy, (Magnus Söderström) |
| | |
| Tiivistelmä:<br>Tämä opinnäytetyö keskittyy verkkoportaalin jatkokehitykseen .NET ympäristössä. Portaalin tehtävä on jäsentää tietoa ja luoda laskuja. Aikaisemmin kehitetty verkkoportaali ei ollut toimiva ja tämän työn tehtävänä on alkuperäisen portaalin jatkokehitys. Työ tutkii alkuperäisen portaalin ongelmat ja löytää niihin ratkaisut. Tämän työn avulla voi nähdä miten tekijä on tehnyt toimimattomasta portaalista toimivan. Onnistunut työ tuottaa toimivan portaalin toimeksiantajalle. Toimeksiantaja voi käyttää sitä laskujen jäsentämiseen ja luomiseen. | |
| Avainsanat: | Verkkoportaali, ohjelmointi, .NET, C#, jäsennin, luoda, lasku, |
| Sivumäärä: | 38 |
| Kieli: | Ruotsi |
| Hyväksymispäivämäärä: | |

# INNEHÅLL

# Figurer

## Förkortningar och begrepp

AngularJS = Ramverk för användargränssnittet

API = Application Programming Interface

AWS = Amazon Web Services

Back-end = Servernivå

Bootstrap = Ramverk för användargränssnittet

CDN = Content Delivery Network (ett nätverk för att leverera innehåll)

CloudFront = Amazons CDN

CLS = Common Language Specification

CSV = Comma-separated values

Front-end = Användargränssnitt

GAR_ID = Primär nyckel för att länka ihop data rätt i detta arbete

HTML = HyperText Markup Language

MVC = Model, View, Controller (~ Modell, vyn, kontroller/styrenhet)

.NET = Microsoft .NET ramverk

Parsning = Syntaxanalys

Webbportal, även bara portal = en webbplats på t.ex. allmänna Internet

# 1 INLEDNING

I detta arbete behandlas vidareutveckling av en webbportal i .NET miljö för parsning och generering av faktura. Skribenten presenterar bl.a. arbetets uppdragsgivare, syfte och mål, planering och kravhantering, de olika verktyg som använts, portalens tekniska struktur och utvecklingsarbetets utförande samt de resultat som nåtts.

Uppdragsgivaren för detta arbete är ABB, ett svensk-schweiziskt multinationellt företag inom elkraft- och automationsteknik. Ett team på ABB, Cloud Computing, har en tidskrävande manuell faktureringsprocess. Cloud Computing teamet kan ersätta manuella faktureringsprocessen med en fungerande webbportal som automatiserar operationen.

En tidigare arbetstagare på ABB hade utvecklat en webbportal, men den lösningen blev aldrig fullständig och kunde därför inte användas. I detta arbete presenteras hur skribenten utvecklat den icke fungerande webbportalen till en fungerande version.

## 1.1 Uppdragsgivaren

ABB är ett svensk-schweiziskt multinationellt företag inom elkraft- och automationsteknik. ABB sysselsätter ungefär 135 000 personer och verkar i mer än 100 länder. I Finland fungerar ABB på ungefär 20 orter och sysselsätter ca 5000 personer. De två största orterna är Helsingfors och Vasa. (ABB Oy 2016)

ABB använder sig av Amazon Web Services (AWS) molntjänster för att kunna effektivt använda och utnyttja diverse IT-miljöer. Inom ABB ansvarar ett team, Cloud Computing, för att administrera och hyra ut dessa molntjänster till andra avdelningar och team inom ABB. På basis av använda resurser skickar AWS en räkning åt ABB, AWS genererar också detaljerade tabeller (csv-filer) över kostnader. Dessa tabeller innehåller information över kostnader per team/applikation.

Cloud Computing teamet går sedan igenom tabellen manuellt, för att kunna dela kostnaderna enligt applikation. Denna manuella process är väldigt tidskrävande och därför vill man ersätta det manuella arbetet med en webbportal som automatiserar processen.

## 1.2 Syfte och mål med arbetet

Syftet med detta arbete är att utveckla en icke fungerande webbportal till en fungerande version. Målet med arbetet är att visa hur man, med hjälp av programmering, utvecklar något som ersätter ett manuellt och tidskrävande arbete gällande faktureringen hos uppdragsgivaren.

## 1.3 Arbetssätt och avgränsning

Utvecklingsarbetet kommer att utföras med hjälp av Google-sökningar och med hjälp av dokumentationen som finns för de ramverk och programmeringsspråk som används för detta arbete. Projektet utförs huvudsakligen som ett självständigt arbete, även möten med experthandledaren på ABB ordnas samt möten med kollegor som i första hand kommer att använda webbportalen.

I detta arbete behandlas vidareutveckling av en webbportal för parsning och generering av faktura i .NET. Arbetet avgränsas på att beskriva hur parsningsarbetet sker i backenden. Arbetet går noggrannare in på de problem som den ursprungliga webbportalen inte löste. En fungerande webbportal kräver även ett fungerande användargränssnitt. I detta arbete utnyttjas det tidigare gjorda användargränssnittet och därför valdes det att inte gå noggrant in på hur ett användargränssnitt utvecklas.

# 2 PLANERING

I detta kapitel presenteras teori om krav och kravhantering, hur planeringen av arbetet gick till, på vilket sätt en kravspecifikation över nödvändiga egenskaper kom fram och vad som inte fungerade i den ursprungliga webbportalen.

## 2.1 Kravhantering

Kravhantering är en process inom programvaruutveckling för att dokumentera, analysera, spåra, prioritera och komma överens om krav för att kunna styra förändringar och kommunicera effektivt med kunden. Kravhantering är en viktig process som skall genomföras genom hela utvecklingsprojektet. (Danielsson 2009)

Enligt Defense Acquisition University Press (DAUP) är det viktigt att planera väl för att ett projekt skall lyckas. Ett programmeringsprojekt innehåller ofta kundens krav och projektets begränsningar. Med krav menas egenskaper och design som kännetecknar projektet, de är behov och mål som oftast kunden specificerar för att få en fungerande slutprodukt. Med begränsningar menas förhållanden som existerar på grund av avgränsningar från externa gränssnitt, stödmoment, tillgänglig teknologi eller livscykelstöd. (DAUP 2001. s. 35)

Det primära syftet för ett utvecklingsarbete definieras av de krav som en kund ställer. Utvecklingsarbetens huvudsakliga uppgift är att omvandla krav till konstruktioner och design. Under utvecklingsarbetet utvecklas designen inom ramen för begränsningar. Det är viktigt att verifiera att designen uppfylls av både krav och begränsningar. (DAUP 2001. s. 35)

### 2.1.1 Kravtyper

Krav kan vara specificerade på flera olika sätt (DAUP 2001. s. 35-36):

- Kundens krav: Fakta som definieras av kunden, t.ex. mål, miljö, begränsningar,
- Funktionella krav: Nödvändiga uppgifter, aktiviteter eller åtgärder som skall utföras.

- Krav på prestanda: I vilken utsträckning ett arbete, uppdrag eller funktion måste utföras. Mäts ofta i kvantitet, kvalitet och täckning.
- Krav på design: Krav på hur man skall bygga, programmera eller köpa produkter.
- Härledda krav: Krav som finns för att ett annat krav på högre nivå inte skulle kunna uppfyllas utan detta krav.
- Fördelade krav: Krav som finns för att man delat upp ett krav på en högre kravnivå till två eller flera lägre krav.

Goda krav kan definieras enligt följande (DAUP 2001. s. 35-36):

- Krav måste vara genomförbara. Det skall reflektera ett behov eller mål som är tekniskt sett möjligt att nås.
- Krav skall gå att verifiera. Förväntade prestandan och funktionaliteten av ett krav skall uttryckas på ett sätt som möjliggör en objektiv verifiering.
- Krav skall vara entydiga, de skall inte ha mer än ett syfte.
- Krav skall uttryckas i form av behov, inte i form av en lösning. Ställa frågorna varför och vad av behovet, inte hur man utför det.
- Krav skall stämma överens med andra krav, konflikter mellan krav måste lösas.

### 2.1.2 Kravanalys

Kravanalys är en viktig del av kravhantering och innebär att man definierar kundens behov, mål och krav. Dessa kan t.ex. vara: användningsområde, användningssyfte, miljö och karaktäriserande drag för systemet. Tidigare kravanalyser är ofta analyserade, uppdaterade och omformulerade för att fylla kundens nya krav. (DAUP 2001. s. 36)

Kravanalysens syfte är bl.a. att (DAUP 2001. s. 37):

- Förfina kundens mål och krav.
- Att definiera de inledande målen och förfina dem till krav.
- Definiera och identifiera begränsningar som begränsar lösningar.
- Definiera funktionella krav och krav på prestanda på basis av kundens mätningar på effektivitet.

Generellt sett skall en kravanalys resultera i en klar förståelse om (DAUP 2001. s. 37):

- Funktioner: Vad skall systemet göra.
- Prestanda: Hur effektivt funktionerna skall genomföras.
- Gränssnitt: Miljön var systemet skall genomföras i.
- Andra krav och begränsningar som påverkar projektet.

Frågor som är typiska för kravanalys (DAUP 2001. s. 38):

- Vilka orsaker finns bakom utveckling av detta system?
- Vad är kundens förväntningar?
- Vem är användarna och hur kommer de att använda systemet?
- Vad förväntar sig användarna av systemet?
- Vad är användarnas kompetensnivå?
- Vilka krav på miljöegenskaper har systemet?
- Finns det existerande eller planerade användargränssnitt?
- Enligt kundens ord, vilka funktioner skall systemet utföra?
- Vilka begränsningar måste systemet lyda (hårdvara, mjukvara och ekonomiska)?

Kravanalysens grund utgörs av de överenskommelser som görs med kunden under kravanalysskede. Den grund definieras bl.a. av funktionella krav och fysiska mönster som skall följas. En bra kravanalys är en grundläggande faktor för ett lyckat utvecklingsprojekt.

## 2.2 Specificering av krav med uppdragsgivaren

I detta arbete gjordes en kravanalys med uppdragsgivaren i början av projektet. Analysen genomfördes på ett möte, under hösten 2015, där webbportalens olika krav togs upp. Under mötet med uppdragsgivaren presenterades de delar som inte fungerade i ursprungliga portalen. Samtidigt fastslogs bland annat följande krav som behövs för en fungerande webbportal i detta arbete:

1. Varje applikation måste kopplas ihop med rätt GAR_ID.
2. Kostnaderna mellan europeiska och nordamerikanska applikationer måste skiljas klart.

3. Mängden servrar måste räknas rätt.
4. CloudFront kostnaderna måste kopplas till rätt applikation.
5. Möjlighet att föra över applikationsregistret till portalen.
6. Ta bort en del av de hårdkodade värdena.

Gällande prioritering av krav, ville uppdragsgivaren att alla krav skulle lösas för att få en fullt fungerande webbportal. Uppdragsgivaren gav högre prioritet på de fyra första kraven än de två sista. För att portalen skulle kunna generera en giltig faktura, med rätta kostnader, måste de fyra första kraven lösas.

Under mötet behandlades även andra krav, bland annat följande: utvecklingsmiljö, verktyg och programvara, portalens syfte, portalens huvudsakliga användare, portalens testning, kommunikationssätt mellan utvecklaren och portalens huvudsakliga användare/testare.

## 2.3 Skribentens uppfattning om webbportalen

Skribentens egen uppfattning om webbportalen och utvecklingsarbetet bildades genom mötet med uppdragsgivaren och genom att bekanta sig med det som tidigare gjorts. Efter att skribenten fått tillgång till den tidigare gjorda webbportalen och efter att ha bekantat sig med hur den fungerar, blev det klarare vilka delar som måste göras om eller lösas.

Skribenten valde att bemöta ett problem åt gången och försöka lösa det förrän nästa problem åtgärdades. Ifall ett problem uppstod, fick skribenten hjälp och handledning av experthandledaren på ABB och på så vis kunde arbetet fortsätta.

# 3 VERKTYGEN

I detta kapitel beskrivs olika programmeringsspråk, programvara och andra verktyg som är aktuella i detta arbete.

## 3.1 Programmeringsspråk

För detta projekt används C# programmeringsspråket. Den tidigare webbportalen var gjord med hjälp av .NET 4.0 MVC ramverk, som utvecklas med C#, och därför blev det ett naturligt val att fortsätta med samma programmeringsspråk.

C# motsvarar även väldigt långt andra språk som Java, C och C++. Därför är det ofta enkelt för utvecklare att hoppa över till C# ifall utvecklaren är bekant med något av de ovan nämnda programmeringsspråk. (Microsoft 2015)

## 3.2 Ramverk

Ett ramverk är en kodsamling som består av klassbibliotek som innehåller återanvändbar och testad kod, utvecklare kan använda ramverk i applikationer som de utvecklar. Med ett ramverk kan utvecklaren styra applikationens flödeskontroll. Då man utvecklar större projekt är det väldigt fördelaktigt att använda sig av ramverk. Med hjälp av ramverk kan utvecklaren ha fokus på att utveckla själva produkten. Utvecklaren behöver inte skriva kod för grundläggande funktioner utan kan använda sig av ett färdigt och tidigare testat ramverk. Samtidigt är det lättare för någon annan att läsa och förstå någon annans kod då man använder sig av något som är tillgängligt för alla. Således är det väldigt normalt att programmeringsteam använder sig av färdiga ramverk för att underlätta utvecklingsprocesser. (Danielsson 2015)

### 3.2.1 Microsoft .NET ramverk

Microsoft .NET är ett ramverk som erbjuder verktyg, tekniker och funktionalitet som behövs för att kunna bygga nätverksapplikationer, distribuerade webbtjänster och webbapplikationer. Detta ger utvecklaren möjlighet att utveckla sina projekt med ett pro-

grammeringsspråk som kommer överens med Common Language Specification (CLS). (.Net-informations.com 2016)

Ursprungliga webbportalen är byggd på .NET 4.0 MVC, på grund av detta har skribenten använt samma ramverk för att vidareutveckla webbportalens funktionalitet.

### 3.2.2 Kendo UI

Kendo UI av Progress är ett HTML5 användargränssnittsramverk för att utveckla interaktiva och högpresterande applikationer och webbsidor. Ramverket har ett stort bibliotek med över 70 olika komponenter för att bygga ett visuellt fint användargränssnitt. Kendo UI har också inbyggt stöd för AngularJS och Bootstrap. (Progress 2016a)

I detta projekt är den ursprungliga webbportalen och dess användargränssnitt byggt med hjälp av Kendo UI. Samma användargränssnitt och ramverk används även vid vidareutveckling av webbportalen.

## 3.3 Programvara

Vid utvecklingsarbeten krävs det att man har bra verktyg. I detta arbete har skribenten i första hand använt programvara och verktyg som ABB gett tillgång till. På grund av att arbetet är att vidareutveckla ett tidigare projekt är skribenten tvungen att använda sig av samma verktyg som den tidigare utvecklaren har valt att använda för att underlätta arbetet.

### 3.3.1 Microsoft Visual Studio

Visual Studio är en programutvecklingsmiljö av Microsoft. Med Visual Studio kan man utveckla applikationer för Microsoft Windows operativsystemet, Windows Phone och applikationer som är anpassade för Internet. (Microsoft 2016)

Visual Studio var även ett naturligt val då .NET ramverket och C# programmeringsspråket är en del av Microsofts ramverk för utveckling av programvara. ABB gav även

tillgång till programmet och de licenser som behövdes för att kunna vidareutveckla webbportalen.

### 3.3.2 Microsoft Excel

Excel är ett kalkylprogram som utvecklats av Microsoft och är en del av deras Microsoft Office programsvit. Med Excel kan man bland annat utföra räkningar, skapa tabeller, använda sig av grafiverktyg och även utnyttja programmeringsspråket Visual Basic for Applications som är en begränsad version av programmeringsspråket Visual Basic. (Microsoft 2016c)

Excel används i detta arbete för att kunna smidigt läsa och identifiera de tabeller som Amazons AWS räkningar innehåller. Excel används också för editering av dokument och dess tabeller som webbportalen använder för att fungera rätt.

### 3.3.3 Team Foundation Server

Team Foundation Server är en Microsoft produkt som har byggts för programmeringsteam. Med hjälp av Team Foundation Server kan man bland annat använda sig av versionshantering, diverse integrationsmöjligheter och verktyg för agila team. (Microsoft 2016b)

I detta projekt sker versionshanteringen med hjälp av Team Foundation Server (TFS). TFS är en versionshanteringslösning som ABB valt att använda och som Microsoft Visual Studio har inbyggt stöd för.

# 4 WEBBPORTALENS TEKNISKA STRUKTUR

I detta kapitel presenteras webbportalens tekniska struktur. Portalens olika delar och moduler presenteras och förklaras i närmare detalj.

Detta projekt är byggt med hjälp av .NET 4.0 MVC för back-enden och Kendo UI, ett HTML5 användargränssnittramverk, för front-enden.

Ett MVC projekt har tre viktiga delar i sig som behövs för att bygga webbaserade applikationer: model, view och controller. Model delen representerar data och är inte beroende av view eller controller delarna. Model innehåller och hanterar bara information. View delen visar data och hanterar handlingar, t.ex. knapptryck, till controllern. View delen kan fungera självständigt och är inte beroende av controller eller model. View kan också vara en controller och behöver då model för att fungera. Controller delen ger model över till view och hanterar handlingar, så som knapptryckningar. Controllern är beroende av view och model och i vissa fall kan controllern och view:n vara samma objekt. (Dalling 2009)

## 4.1 Hänvisningar i projektet

Projektet innehåller flera hänvisningar till andra bibliotek och moduler för att säkra att varje funktion fungerar smidigt och bra.

Hänvisningar och bibliotek:

- ABB.Core.Helper: hjälper till och sköter kommunikationen till Active Directory (AD) för att autentisera Windows användare.
- Ionic.Zip: gratis klassbibliotek och verktyg för editering av zip-filer. Används för att enkelt skapa, packa upp eller uppdatera zip-filer.
- NPOI: gratis klassbibliotek och verktyg för att skapa, läsa och editera Excel-filer.
- Log4net: Apaches log4net bibliotek är ett verktyg som används för att skapa loggfiler vid diverse händelser och handlingar i program.

- AWS SDK för.NET: bibliotek för att bygga applikationer som kan kommunicera med AWS, låter användaren kalla på AWS API.
- AWS SDK för .NET Extensions: en utvidgning av AWS SDK. Används för att hantera sessionstillstånd.
- CSVHelper: gratis bibliotek, som används i detta projekt för att läsa och editera csv-filer.

## 4.2 Allmänna applikationsfiler

Dessa filer innehåller allmänna funktioner som används i hela projektet.

- Excel.cs: innehåller funktioner för att läsa Excel-filer till model-filer. Innehåller också stildekorationer för Excel-filer som skapas av webbportalen.
- LogHelper.cs: funktioner för att skapa loggfiler vid diverse handlingar, t.ex. vid en felkörning.
- UserService.cs: innehåller funktioner för att autentisera och logga in Windows användare automatiskt.
- XAuthorize.cs: autentiseringsfunktioner, t.ex. för att autentisera en administrator.

## 4.3 Controller-filer

Huvudfunktioner för projektet, dessa filer innehåller de funktioner som projektet behöver för att kunna köras.

- AWSController.cs: innehåller funktioner för att se data i realtid från AWS.
- ErrorController.cs: funktioner för att hantera diverse feltillstånd.
- ExcelController.cs: huvudarbetet i projektet, parsern. Funktioner som sköter om parsning av data och generering av fakturor.
- HomeController.cs: funktioner för att definiera huvudsidan.
- UserController.cs: funktioner för en administrator för att hantera användare och deras rättigheter.
- UploadController.cs: innehåller funktioner för att ladda upp filer till portalen.

## 4.4 Model-filer

Model-filerna definierar all data som används i projektet. Separerar data för view:n och controllern.

- CostBalance.cs: definierar nödvändig information för en balansräkningsfil.
- EC2Inst.cs: definierar nödvändig information för AWSController för att kunna rita live data.
- Error.cs: definierar information för feltillstånd.
- Track.cs: definierar nödvändig information för applikationsregistret som är kritisk för webbportalen.
- Usage.cs: definierar information som används för att rita och skapa fakturor.
- UserInfo.cs: definierar information om webbportalens användare.

## 4.5 View-filer

Filer i view definierar alla de sidor som webbportalen har.

- AWS: för att rita och skapa live data från AWS som AWSController hämtar.
- Error: för att rita en sida då ett feltillstånd uppstår.
- Excel: sidan för parsning och generering av faktura, kallar på ExcelController.
- Home: definierar startsidan.
- Shared: definierar den sammanhängande layouten för hela webbportalen.
- Upload: definierar sidan var administratörer kan ladda upp filer till webbportalen
- User: definierar sidan för att hantera användartillstånd.

## 4.6 Databasfiler

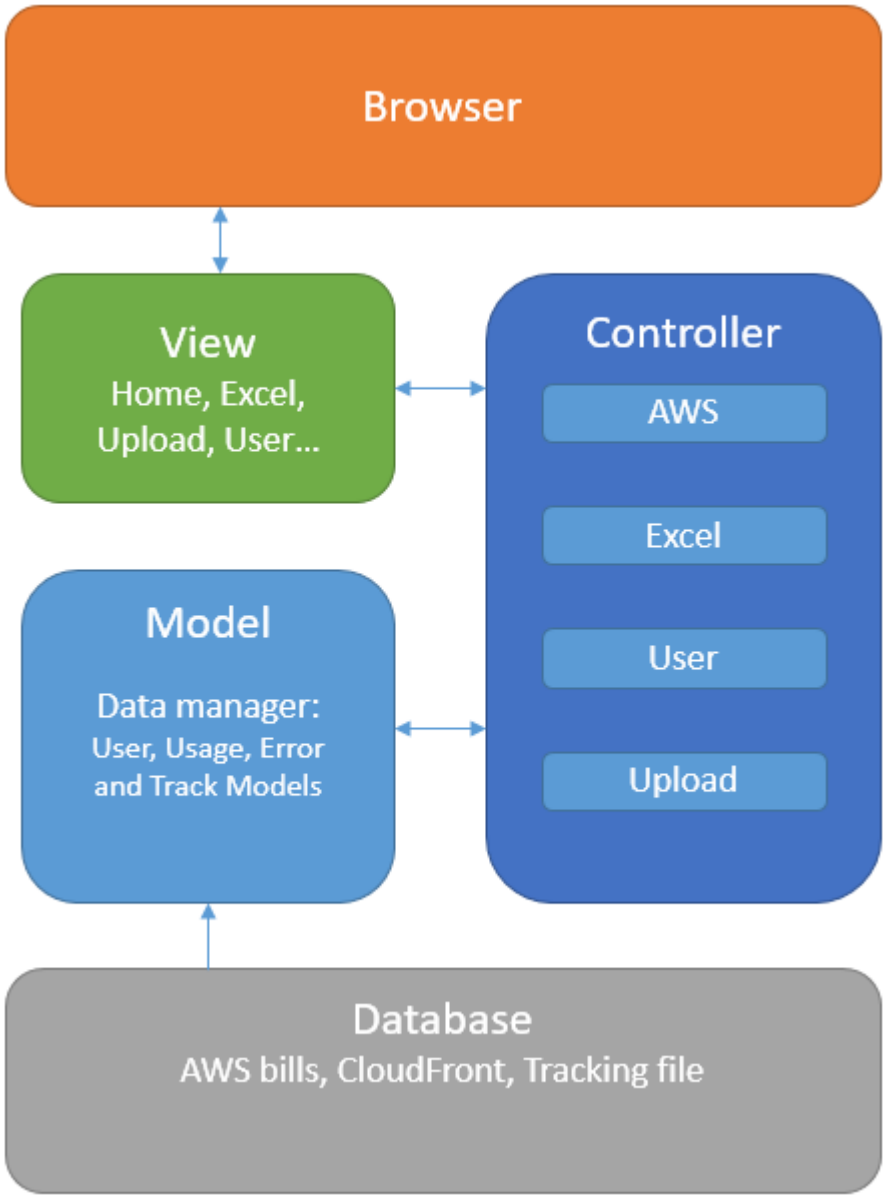
Detta projekt använder sig inte av en egentlig databas, istället använder webbportalen sig av några olika Excel-filer för att kunna hämta all nödvändig information.

Det finns två olika mappar som innehåller Excel-filer. Den första mappen innehåller månatliga räkningsfiler som uppdateras automatiskt från AWS. Den andra mappen in-

nehåller ett applikationsregister som är huvuddatabasen för webbportalen. Applikations-registret uppdateras manuellt av webbportalens användare.

## 4.7 Arkitektursbild över webbportalen

Följande figur, Figur 1 (Singapore Management University 2012), är en arkitektursbild över webbportalen. I figuren kan man se hur olika delar av ett .NET MVC projekt är beroende av varandra.

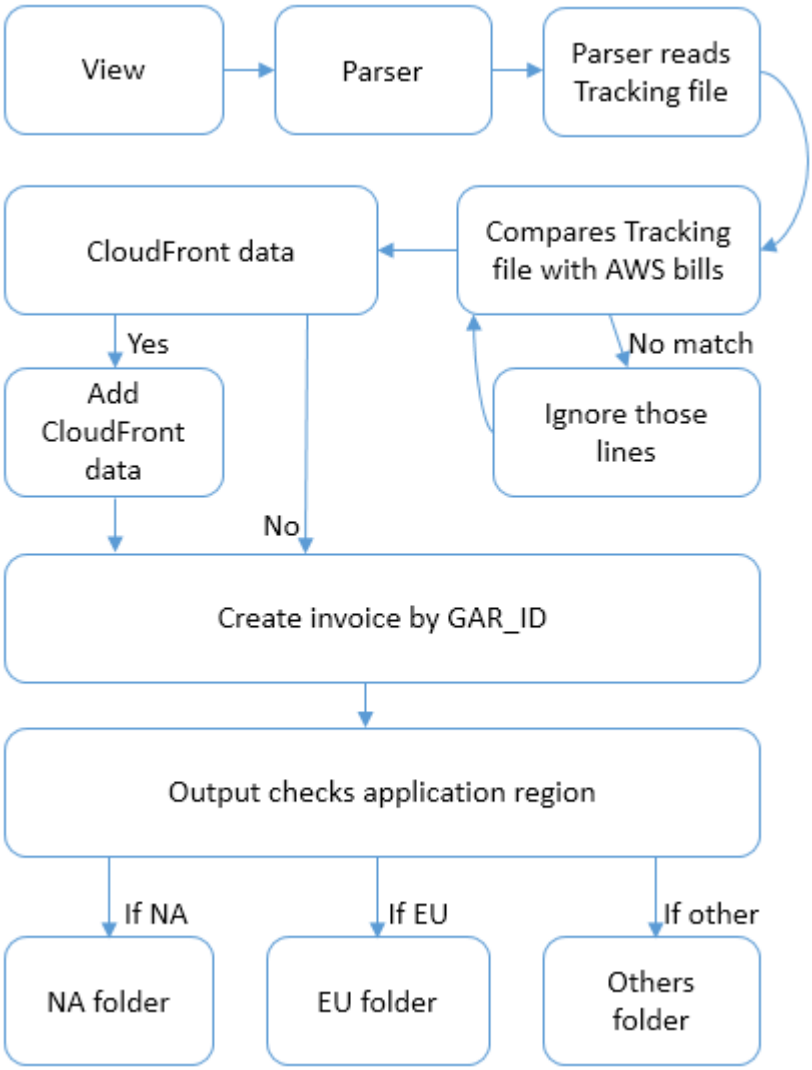



*Figur 1. Arkitektursbild*

I detta webbprojekt sker kommunikation till webbportalen med hjälp av en webbläsare (browser). Användaren öppnar portalen via en browser, med hjälp av ett knapptryck kan användaren kommunicera med view delen. View kallar sedan på controllern, som vid behov kallar på model. Model delen innehåller bara data, data som används av controllern. Data som används kommer i detta fall från Excel-filer, i form av ett applikationsregister och räkningsfiler från AWS.

## 4.8 Flödesschema: parsning och generering av faktura

I följande flödesschema, Figur 2, presenteras parsnings- och genereringsfunktionen.




*Figur 2. Flödesschema*

Användaren startar funktionen via view, som sedan kallar på parsern i controllern. Parsern läser sedan igenom applikationsregistret (Excel-filen). Efter en lyckad läsning jämför webbportalen varje enskild applikation från applikationsregistret mot räkningsfilen från AWS. Webbportalen ignorerar de rader som inte stämmer överens med applikationsregistret. Efter det jämför portalen alla applikationer som är definierade som CloudFront applikationer i applikationsregistret. Vid en träff läggs CloudFront kostnaderna till specifika applikationens faktura.

Efter det genererar webbportalen en faktura på basis av all information som den fått på den specifika applikationen från diverse räkningsfiler. Till sist genererar portalen en faktura (Excel-fil) och kollar ifall applikationen har en region definierad i applikationsregistret. Ifall en region finns läggs den specifika fakturan i en mapp för den regionen.

# 5 UTVECKLINGSARBETET

I detta kapitel beskrivs uppdragsgivarens krav, hur skribenten gått till väga för att lösa dessa krav och resultat över utförandet.

## 5.1 Koppla ihop applikation med rätta kostnader

Ett av uppdragsgivarens krav var att varje enskild applikation skall kopplas ihop med rätta kostnader från faktureringsfiler. Nedan presenteras en noggrannare beskrivning över utförandet och hur skribenten gått till väga för att lösa kravet.

Varje enskild kund identifieras som en applikation och varje applikation får en identifikationssträng som kallas GAR_ID. Denna GAR_ID är den primära nyckeln för den applikationen, med hjälp av en GAR_ID kan en applikation kopplas ihop med dess specifika kostnader.

På webbportalen kan användaren mata in en GAR_ID i taget och göra sökningar per GAR_ID. Användaren kan även välja att göra en sökning på basis av ett applikationsregister som skall innehålla alla applikationer som finns i ABB:s AWS molntjänst. Detta register måste uppdateras manuellt för att få med alla de applikationer som man vill generera en faktura för. Applikationsregistret presenteras i korthet i Figur 3.

| | A | B | C |
|---|---|---|---|
| 1 | | | |
| 2 | **Account** | **GAR_ID** | **Application Name** |
| 6 | ABB | GARID101 | Application01 |
| 7 | ABB | GARID102 | Application02 |
| 8 | ABB | GARID103 | Application03 |
| 9 | ABB | GARID104 | Application04 |
| 10 | ABB | GARID105 | Application05 |
| 11 | ABB NAM | GARID106 | Application06 |
| 12 | ABB NAM | GARID107 | Application07 |
| 13 | ABB NAM | GARID108 | Application08 |
| 14 | ABB NAM | GARID109 | Application09 |
| 15 | ABB NAM | GARID110 | Application10 |
| 16 | ABB NAM | GARID111 | Application11 |

*Figur 3. Exempel på applikationsregistret.*

Från detta register använder sig webbportalen av väsentlig information för att kunna generera en faktura. Den viktigaste kolumnen är GAR_ID, vilken länkar ihop en applikation med information som finns i csv-filen om kostnader, vilken AWS generar månadsvis. Från de övriga kolumnerna kopieras information över, som t.ex. applikationens namn. Ett exempel på en AWS genererad csv-fil kan ses i Figur 4.

| | J | M | P | W | X | AG |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | LinkedAccountName | ProductCode | UsageType | UsageQuantity | BlendedRate | user:GAR_ID |
| 3 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID101 |
| 4 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID102 |
| 5 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID103 |
| 6 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID104 |
| 7 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID103 |
| 8 | ABB | AmazonEC2 | EU-BoxUsage | 123.0 | 0.112 | GAR_ID105 |
| 9 | ABB NAM | AmazonEC2 | BoxUsage | 321.0 | 0.312 | GAR_ID107 |
| 10 | ABB NAM | AmazonEC2 | BoxUsage | 321.0 | 0.312 | GAR_ID105 |
| 11 | ABB NAM | AmazonEC2 | BoxUsage | 321.0 | 0.312 | GAR_ID108 |
| 12 | ABB NAM | AmazonEC2 | BoxUsage | 321.0 | 0.312 | GAR_ID109 |
| 13 | ABB NAM | AmazonEC2 | BoxUsage | 321.0 | 0.312 | GAR_ID108 |

*Figur 4. Exempel på AWS faktura*

Portalen jämför kolumn user:GAR_ID i Figur 4 med kolumn GAR_ID från Figur 3 och då kan den koppla ihop rätta kostnader med rätt applikation. Portalen gör sedan en sökning på basis av GAR_ID, som presenteras kort i Figur 5.

```
public JsonResult GetResult(string gar_id, string filename, double vat, string region)
{
    try
    {
        if (gar_id == "")
            return Json("", JsonRequestBehavior.AllowGet);

        if (region == null || region == "")
        {
            data = AppCode.Excel.openBillingCSV(AppDomain.CurrentDomain.BaseDirectory + "billfolder/"
            + filename + ".csv").FindAll(i => (i.Gar_ID == gar_id));
        }

        //region is for separating applications cost per region
        region = region.Replace("%20", " ");
        if (region != "")
        {
            data = AppCode.Excel.openBillingCSV(AppDomain.CurrentDomain.BaseDirectory + "billfolder/"
            + filename + ".csv").FindAll(i => (i.Gar_ID == gar_id) && (i.LinkedAccountName == region));
        }
        if (data.Count == 0)
        {
            return Json("", JsonRequestBehavior.AllowGet);
        }
```

*Figur 5. Sökning enligt GAR_ID*

## 5.2 Skilja åt applikation per region

Ett av uppdragsgivarens krav var att skilja åt applikationer per region. Nedan presenteras en noggrannare beskrivning över hur skribenten gått till väga för att lösa detta krav.

För tillfället finns applikationer i både Europa och Nordamerika. En applikation kan ha samma GAR_ID både i Europa och Nordamerika och därför måste kostnaderna också identifieras enligt region. För att webbportalen skall kunna skilja åt kostnader enligt region måste kolumn Account i Figur 3 jämföras med kolumn LinkedAccountName i Figur 4. Applikationer i Europa är märkta som ABB medan applikationer i Nordamerika är märkta som ABB NAM.

Ifall kolumn Account i Figur 1 har ett värde, jämför webbportalen detta värde med kolumn LinkedAccountName från Figur 4, enligt exemplet i Figur 5. Om en applikation saknar värde i kolumn Account söker webbportalen upp alla kolumner enligt GAR_ID och ignorerar att jämföra region, se Figur 5.

Vid exportering av flera fakturor skiljer webbportalen åt applikationer med att lägga dem i olika mappar på basis av värdet region, se Figur 6.

```
if (item.region.Contains("NAM")) //for NAM
{
    System.IO.File.WriteAllBytes(dirNAM + filename, file);
}
else //if (item.region == "ABB") //for EU and others
{
    System.IO.File.WriteAllBytes(dirEU + filename, file);
}
```

*Figur 6. Mapp per region*

## 5.3 Räkna antalet servrar

Det tredje kravet uppdragsgivaren hade var att räkna antalet servrar per applikation. I detta delkapitel presenteras skribentens utförande och lösning till detta krav.

En applikation kan använda sig av flera servrar och därför behöver webbportalen räkna rätt antal servrar som används per applikation.

För att räkna antalet servrar per applikation måste webbportalen använda sig av två kolumner från AWS fakturan (se Figur 4). Första kolumnen är kolumn UsageType som innehåller diverse strängar. I detta fall är portalen intresserade av celler som innehåller värden BoxUsage, EU-BoxUsage, RDS eller EU-RDS. Dessa värden beskriver att det är frågan om en server. Då en sökning görs kollar portalen igenom ifall den hittar dessa värden och antalet träffar indikerar mängden servrar per en applikation, se Figur 7.

På grund av att en server kan t.ex. startas om, kan ett värde förekomma mer än en gång i samma tabell, därför jämförs också kolumn user:NAME för att få reda på serverns namn. Ifall ett namn finns flera gånger så räknar webbportalen bort dem och varje server räknas med bara en gång, se Figur 7.

```
List<Usage> serverList = new List<Usage>();     //list for listing the server names(column user:NAME)
for (int i = 0; i < data.Count; i++)    //traverse all lines for the resource usage sheet
{
    Usage server = new Usage();
    if (data[i].UsageType.StartsWith("BoxUsage"))  //if NAM EC2 instance cost
    {
        data[i].CostBeforeTax = data[i].BlendedRate * data[i].UsageQuantity;
        server.userName = data[i].userName;
        serverList.Add(server);
    }
    else if (data[i].UsageType.StartsWith("RDS"))  //if NAM RDS instance cost
    {
        data[i].CostBeforeTax = data[i].BlendedRate * data[i].UsageQuantity;
        server.userName = data[i].userName;
        serverList.Add(server);
    }
    else if (data[i].UsageType.StartsWith("EU-BoxUsage"))  //if EUR EC2 instance cost
    {
        data[i].CostBeforeTax = data[i].BlendedRate * data[i].UsageQuantity;
        server.userName = data[i].userName;
        serverList.Add(server);
    }
    else if (data[i].UsageType.StartsWith("EU-RDS"))  //if EUR RDS instance cost
    {
        data[i].CostBeforeTax = data[i].BlendedRate * data[i].UsageQuantity;
        server.userName = data[i].userName;
        serverList.Add(server);
    }
    else
    {
        if (i + 1 < data.Count)
        {
            if (data[i].UsageType == data[i + 1].UsageType)
            {
                data[i + 1].UsageQuantity += data[i].UsageQuantity;
                data[i + 1].CostBeforeTax = data[i + 1].BlendedRate * data[i + 1].UsageQuantity;
                data.Remove(data[i]);
                i--;
                continue;
            }
        }
        if (data[i].CostBeforeTax < 0.000001)
        {
            data.Remove(data[i]);
            i--;
            continue;
        }
    }
    data[i].TotalCost = data[i].CostBeforeTax * (1 + vat / 100);
}
//removing duplicates, and getting correct server number
int serverCount = serverList.GroupBy(x => new { x.UsageType, x.userName }).Select(g => g.First()).ToList().Count;
```

*Figur 7. Räkna antal servrar*

## 5.4 Kostnader från CloudFront

Uppdragsgivaren hade som ett krav att kostnaderna från CloudFront skulle läggas till per applikation. Skribentens tillvägagångssätt för att lösa detta krav presenteras nedan samt resultat för hur kravet blev löst.

CloudFront kostnaderna identifieras från en skild fil än de övriga kostnaderna, dessa kostnader skall också räknas med per applikation och GAR_ID. CloudFront kostnaderna är inte identifierade enligt GAR_ID och därför används en skild tabell för att länka ihop dessa kostnader med rätt applikation.

CloudFront kostnaderna länkas ihop med hjälp av tabellen i Figur 8. Med hjälp av GAR_ID vet webbportalen vilken applikation som kostnaderna skall läggas till. Med hjälp av ResourceId kan webbportalen hitta rätt kostnader från AWS-filen över CloudFront kostnader.

För att hitta rätta kostnader söker webbportalen igenom filen som innehåller CloudFront kostnader och jämför kolumn ResourceId i Figur 8 med ResourceId i Figur 9. Efter det förs kostnaderna över till rätt applikation på basis av GAR_ID, se Figur 10.

| | A | B | C |
|---|---|---|---|
| 1 | | | |
| 2 | **Application Name** | **GAR_ID** | **ResourceId** |
| 3 | Application02 | GAR_ID101 | ResourceID123 |
| 4 | Application01 | GAR_ID101 | ResourceID111 |
| 5 | Application03 | GAR_ID111 | ResourceID001 |
| 6 | Application04 | GAR_ID103 | ResourceID772 |
| 7 | Application06 | GAR_ID103 | ResourceID313 |
| 8 | Application07 | GAR_ID107 | ResourceID201 |
| 9 | Application08 | GAR_ID107 | ResourceID007 |
| 10 | Application10 | GAR_ID105 | ResourceID333 |
| 11 | Application09 | GAR_ID108 | ResourceID222 |

*Figur 8. CloudFront länkning*

| | Q | R | U | V |
|---|---|---|---|---|
| 1 | UsageQuantity | BlendedRate | UnBlendedCost | ResourceId |
| 2 | 123.0 | 0.1 | 0.312 | ResourceID123 |
| 3 | 123.0 | 0.02 | 0.312 | ResourceID111 |
| 4 | 123.0 | 0.003 | 0.312 | ResourceID001 |
| 5 | 123.0 | 0.0004 | 0.312 | ResourceID772 |
| 6 | 123.0 | 0.00005 | 0.312 | ResourceID313 |
| 7 | 123.0 | 0.02 | 0.312 | ResourceID201 |
| 8 | 123.0 | 0.0004 | 0.312 | ResourceID007 |
| 9 | 123.0 | 0.003 | 0.312 | ResourceID333 |
| 10 | 321.0 | 0.0004 | 0.112 | ResourceID222 |

*Figur 9. Exempel på CloudFront kostnader*

```
//This is for cloudfront fee
List<Usage> clouddata = cloudfrontTracklist.FindAll(i => i.Gar_ID == gar_id);
foreach (var obj in clouddata)
{
    Usage use = new Usage
    {
        Gar_ID = gar_id,
        ProductCode = "AmazonCloudFront",
        ItemDescription = "ResourceID: " + obj.resourceId,
        ProductName = "Amazon CloudFront"
    };

    List<Usage> cloudfrontList = AppCode.Excel.CloudfrontCSV(AppDomain.CurrentDomain.BaseDirectory
    + "xlsbase/cloudfront/" + filename + "-cloudfront.csv").FindAll(i => i.resourceId.Contains(obj.resourceId));
    foreach (var obj2 in cloudfrontList)
    {
        use = new Usage
        {
            Gar_ID = gar_id,
            ProductCode = "AmazonCloudFront",
            ItemDescription = "ResourceID: " + obj.resourceId,
            ProductName = "Amazon CloudFront",
            //UsageQuantity = use.UsageQuantity += obj2.UsageQuantity,
            //BlendedRate = use.BlendedRate += obj2.BlendedRate,
            CostBeforeTax = use.CostBeforeTax += obj2.UnBlendedCost,
            TotalCost = (use.CostBeforeTax += obj2.UnBlendedCost) * (1 + vat / 100)
        };
    }
    data.Add(use);
}
```

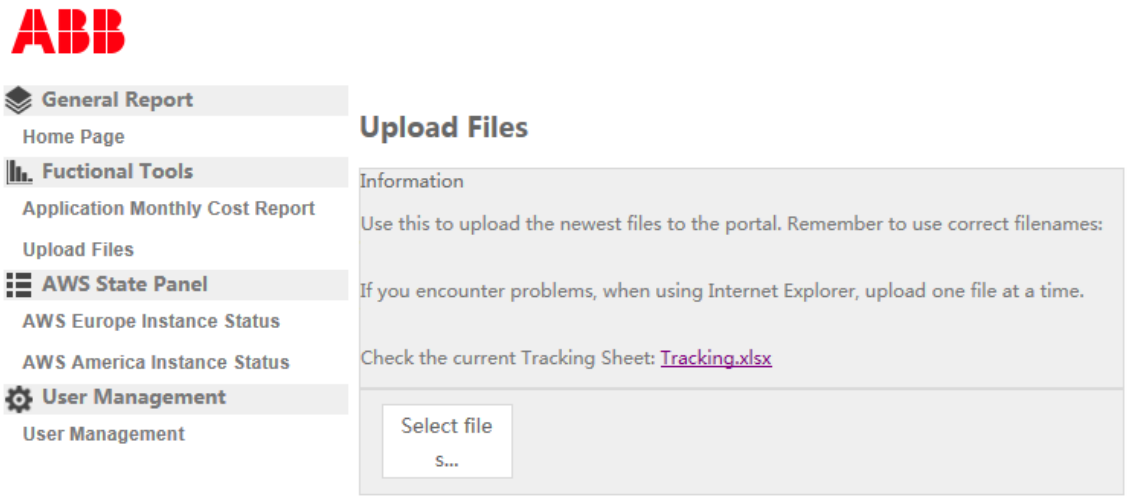
*Figur 10. CloudFront back-end*

## 5.5 Överföring av applikationsregistret till webbportalen

Det femte kravet uppdragsgivaren hade handlade om att överföra applikationsregistret till webbportalen. Nedan presenteras skribentens arbetssätt och resultat för att lösa detta krav.

Cloud Computing teamet ansvarar över att uppehålla ett applikationsregister över applikationer som finns i ABB:s AWS molntjänst. Webbportalen använder sig av detta register för att kunna generera fakturor. Det är därför viktigt att kunna enkelt överföra en uppdaterad version av applikationsregistret till webbportalen då man vill generera nya fakturor.

För att kunna överföra det nyaste applikationsregistret utvecklades en enkel överföring med hjälp av funktionaliteten som Kendo UI ramverket erbjöd (Progress 2016b). En skild sida utvecklades till portalen där användaren kan ladda ner det nuvarande applikationsregister och föra över en uppdaterad version. Sidans layout kan ses i Figur 11 och back-enden i Figur 12.




*Figur 11. Överföringsvy*

```
public ActionResult Submit(IEnumerable<HttpPostedFileBase> files)
{
    try {
        if (files != null)
        {
            //TempData["UploadedFiles"] = GetFileInfo(files);
            foreach (var file in files)
            {
                string date = DateTime.Now.ToString("yyyy.MM.dd-HH.mm.ss");
                var fileName = Path.GetFileName(file.FileName);
                var pathFile = Path.Combine(Server.MapPath("~/xlsbase/"), fileName);
                var path = Server.MapPath("~/xlsbase/");

                if (System.IO.File.Exists(pathFile))
                {
                System.IO.File.Move(pathFile, path + date + "-" + fileName); //renames the earlier file
                }
                file.SaveAs(pathFile); //uploads the new file

                //trys to remove the oldest file, saves the 5 newest
                var files2 = new DirectoryInfo(path).GetFiles("*" + fileName);
                foreach (var file2 in files2.OrderByDescending(x => x.LastWriteTime).Skip(5))
                {
                    file2.Delete();
                }
            }
        }
        //return RedirectToAction("Index");
        return Content("");
    }
    catch (Exception ex)
    {
        Error e = new Error();
        e.err = ex.ToString();
        log.Error("Error Occurs when accessing files on server", ex);
        return Json(ex.ToString(), JsonRequestBehavior.AllowGet);
    }
}
```

*Figur 12. Överföring back-end*

## 5.6 Minska på de hårdkodade värdena

Ett av kraven som uppdragsgivaren hade var att bli av med hårdkodade värden. I detta delkapitel presenteras vad skribenten utfört för att lösa kravet och redovisning för resultatet.

En del av den tidigare funktionaliteten använder sig av hårdkodade värden. På grund av att värdena är hårdkodade blir man tvungen att programmera om dem ifall en förändring sker i en databastabell (Palermo 2009). I detta fall är det inte en effektiv lösning och därför önskade uppdragsgivaren att detta skulle programmeras om.

I funktionen som överför information från AWS faktureringstabellen var detta tidigare löst med att ha absoluta värden, enligt tabellens positioner, dvs. att kolumn A var identifierad som 0, kolumn B som 1, kolumn C som 2 osv. För att bli av med absoluta positioner ändrades funktionen om till att identifiera kolumner enligt deras namn och inte enligt deras position, se Figur 13.

```
while (csv.Read())
{
    if (csv.GetField<string>("user:GAR_ID") != "")
    {
        Usage line = new Usage();
        line.Gar_ID = csv.GetField<string>("user:GAR_ID").ToUpper();
        line.ProductCode = csv.GetField<string>("ProductCode");
        line.ProductName = csv.GetField<string>("ProductName");
        line.UsageType = csv.GetField<string>("UsageType");
        line.ItemDescription = csv.GetField<string>("ItemDescription");
        line.UsageStartDate = csv.GetField<string>("UsageStartDate");
        line.UsageEndDate = csv.GetField<string>("UsageEndDate");
        line.UsageQuantity = double.Parse(csv.GetField<string>("UsageQuantity"), System.Globalization.CultureInfo.InvariantCulture);
        line.BlendedRate = double.Parse(csv.GetField<string>("BlendedRate"), System.Globalization.CultureInfo.InvariantCulture);
        line.CostBeforeTax = double.Parse(csv.GetField<string>("CostBeforeTax"), System.Globalization.CultureInfo.InvariantCulture);
        line.Server = csv.GetField<string>("user:SERVER");
        line.LinkedAccountName = csv.GetField<string>("LinkedAccountName");
        line.userName = csv.GetField<string>("user:Name");
        data.Add(line);
    }
}
return data;
```

*Figur 13. Kolumn enligt namn*

## 5.7 Testfaser

Kravspecifikationen presenterades tidigare i arbetet, dessa krav var kritiska för att få en fullt fungerande webbportal. Under och efter utvecklingsprocessen har webbportalen blivit testad av portalens huvudsakliga användare. Testarna dokumenterade de buggar som de hittade och informationen fördes sedan vidare till utvecklaren (skribenten). Skribenten löste sedan dessa buggar och testarna fortsatte att testa portalen tills inga buggar hittades och testningen nådde lyckade resultat. Skribenten har även testat webbportalen under utvecklingsprocessens olika skeden och nått positiva resultat. På basis av de kriterier som uppdragsgivaren ställde vid det första utvecklingsmötet har utvecklingsarbetet lyckats väl och kravspecifikationen har blivit nådd.

## 5.8 Ersättning av manuellt arbete

Webbportalens syfte var att ersätta ett manuellt och tidskrävande arbete. På basis av det lyckade resultat utvecklingsarbetet gett har detta mål nåtts. Tidigare krävde fakture-

ringsarbetet en persons fulla fokus då parsningsarbetet gjordes för hand. Med hjälp av den fungerande webbportalen kan man nu slopa det tidskrävande manuella arbetet.

### 5.8.1 Exempel på en faktura

Då webbportalens olika utvecklingskrav var lösta blev portalen användbar enligt detta examensarbetets syfte. En lyckad parsning i portalen resulterar i en faktura som presenteras i Figur 14. Exemplet på fakturan redogör hur utvecklingsarbetets resultat ser ut i praktiken.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | GAR_ID: GAR_ID101 | | Application: Application101 | | Date: 2016-08 | | | | | All Values in USD |
| 2 | ProductCode | ProductName | UsageType | ItemDescription | UsageStart | UsageEndD | UsageQ | BlendedR | CostBef | TotalCost |
| 3 | AmazonElastiCach | Amazon ElastiCache | EU-NodeUsage:cache. | $0.2 per Enhanced Large Cache | 2016/08/01 | 2016/08/31 | 30,00 | 0,20 | 6,00 | 7,20 |
| 4 | AWSDataTransfer | AWS Data Transfer | EU-DataTransfer-Regi | $0.010 per GB - regional data tr | 2016/08/01 | 2016/08/31 | 0,43 | 0,01 | 0,00 | 0,01 |
| 5 | AWSDataTransfer | AWS Data Transfer | EU-USW2-AWS-Out-By | $0.02 per GB - EU (Ireland) data | 2016/08/01 | 2016/08/31 | 0,00 | 0,02 | 0,00 | 0,00 |
| 6 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-EBS:VolumeUsage. | $0.11 per GB-month of General | 2016/08/01 | 2016/08/31 | 14,15 | 0,11 | 1,56 | 1,87 |
| 7 | AWSDataTransfer | AWS Data Transfer | EU-USE1-AWS-Out-Byt | $0.02 per GB - EU (Ireland) data | 2016/08/01 | 2016/08/31 | 0,02 | 0,02 | 0,00 | 0,00 |
| 8 | AWSDataTransfer | AWS Data Transfer | EU-DataTransfer-Out- | $0.090 per GB - up to 10 TB / m | 2016/08/01 | 2016/08/31 | 0,04 | 0,09 | 0,00 | 0,00 |
| 9 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-DataProcessing-By | $0.008 per GB Data Processed | 2016/08/01 | 2016/08/31 | 0,18 | 0,01 | 0,00 | 0,00 |
| 10 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-LoadBalancerUsag | $0.028 per LoadBalancer-hour | 2016/08/01 | 2016/08/31 | 30,00 | 0,03 | 0,84 | 1,01 |
| 11 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-BoxUsage:c3.large | $0.188 per On Demand Windov | 2016/08/01 | 2016/08/31 | 15,00 | 0,19 | 2,82 | 3,38 |
| 12 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-BoxUsage:c3.large | $0.188 per On Demand Windov | 2016/08/01 | 2016/08/31 | 15,00 | 0,19 | 2,82 | 3,38 |
| 13 | AmazonEC2 | Amazon Elastic Compute Cloud | EU-BoxUsage:m3.med | $0.129 per On Demand Windov | 2016/08/01 | 2016/08/31 | 15,00 | 0,13 | 1,94 | 2,32 |
| 14 | AmazonCloudFront | Amazon CloudFront | | ResourceID: ResourceID111 | | | 0,00 | 0,00 | 71,78 | 86,14 |
| 15 | Hosting Service | Cloud Protection Manager Subs | | | | | 3,00 | 5,00 | 0,00 | 15,00 |
| 16 | Hosting Service | | | | | | 3,00 | 150,00 | 0,00 | 450,00 |
| 17 | TOTAL | | | | | | 0,00 | 0,00 | 87,76 | 570,32 |

*Figur 14. Exempel på en faktura*

# 6 SLUTSATSER OCH DISKUSSION

I detta kapitel presenterar skribenten sina slutsatser och tankar kring arbetets genomförande och diverse utmaningar.

Syftet med detta arbete var att utveckla en icke fungerande webbportal till en fungerande version. Arbetets mål var att visa hur man, med hjälp av programmering, kan utveckla något som ersätter ett manuellt och tidskrävande arbete. Uppdragsgivaren och skribenten hade på ett möte lagt ihop krav som skulle nås för att webbportalen skulle bli funktionsenlig. Enligt detta arbetets syfte och mål och på basis av resultatredovisningen, som presenterades i kapitel fem, svarar arbetets resultat mot dess syfte.

I kapitel två presenterades bra praxis gällande kravhantering. Uppdragsgivaren och skribenten påbörjade kravhanteringsprocessen för detta arbete på mötet där projektets krav fastställdes. I stora drag utfördes detta projekt enligt god kravhantering och projektet nådde sitt syfte och mål. I detta projekt fastställde vi klart och tydligt kundens krav och funktionella krav: hur vill kunden att webbportalen skall fungera och vilket syfte har webbportalen. Vi har även snuddat på andra delar gällande god kravhantering, men efteråt tänkt kunde vi även ha fokuserat mera på t.ex. krav gällande webbportalens utseende. Då jag ser tillbaka på denna utvecklingsprocess kan jag klart se hurdan vikt god kravhantering har. Vilket jag, som utvecklare, kan ta med mig till framtida projekt och kanske hjälpa en framtida kund eller arbetsgivare att arbeta kring en god kravhanteringsprocess.

En stor utmaning vid vidareutveckling är att förstå den ursprungliga utvecklarens kod. Tyngden på att skriva god kod är väldigt viktig, ju klarare kod man skriver och ju tydligare man kommenterar, desto enklare blir det för en annan utvecklare att förstå och komma in i en annan persons projekt. Vid stora utvecklingsprojekt är det sällan en person som skriver all kod självständigt och då är det viktigt att andra utvecklare kan läsa och förstå vad en annan utvecklare skrivit. För en utvecklare är det också mycket givande att kunna bidra till ett projekt och se lyckade resultat framför sig. God kod kan specificeras på bland annat följande sätt: renlighet, förenlighet, sträckbarhet och felfrihet. (Gnome Developer 2014)

För mig var det i början en stor utmaning att komma in detta projekt, att förstå vad varje funktion och metod gör. Tack vare god kod och gott om tid blev koden mer och mer läsbar under arbetets gång. Sist och slutligen är det en givande upplevelse att kunna förstå någon annans kod och även kunna förbättra och vidareutveckla den. På detta område har jag lärt mig mycket och sett detta utvecklingsarbete som en lyckad och givande process.

Programmeringen av webbportalen har utförts som ett självständigt arbete och all kommunikation till portalens huvudsakliga användare har skett elektroniskt. Detta på grund av att Cloud Computing teamet i stort sett befinner sig i Polen. Detta har varit en utmaning, fastän man kan hålla videomöten över Internet, är det en annorlunda situation att diskutera med någon över Internet än i person. Tack vare att min experthandledare, på ABB i Finland, befann sig i samma arbetsutrymmen som jag, blev det enklare att komma vidare ifall jag stötte på ett problem som kändes för svårt.

I och med att detta arbete var ett beställningsarbete är en del av informationen sekretessbelagt och därför hade jag inte möjlighet att lägga portalens källkod som bilaga. Ifall läsaren skulle få tillgång till källkoden skulle de kunna se portalens olika funktioner i närmare detalj och få en ännu noggrannare bild över hur portalen är programmerad.

Från min personliga synvinkel har detta projekt varit ett givande och lärorikt arbete som har utvecklat mina programmeringskunskaper. Arbetet var lyckat och de krav som ställdes blev nådda. Uppdragsgivaren har fått en fungerande webbportal som ersätter manuella arbetet. I framtiden, och vid behov, kan webbportalen vidareutvecklas ifall nya krav och behov uppstår.

# KÄLLOR


ABB Oy. 2016, *ABB Suomessa.* Tillgänglig: http://new.abb.com/fi/abb-lyhyesti/suomessa. Hämtad: 5.11.2016.

Dalling, Tom. 2009, *Model View Controller Explained,* Tillgänglig: http://www.tomdalling.com/blog/software-design/model-view-controller-explained/. Hämtad 15.11.2016

Danielsson, Lars. 2009, *Kravhantering borde vara högsta prioritet. Tillgänglig:* http://computersweden.idg.se/2.2683/1.230687/kravhantering-borde-vara-hogsta-prioritet. Hämtad: 16.11.2016

Danielsson, Lars. 2015, *Ramverk eller inte ramverk – det är frågan.* Tillgänlig: http://computersweden.idg.se/2.2683/1.606077/ramverk-eller-inte-ramverk--det-ar-fragan. Hämtad: 12.11.2016

Defense Acquisition University Press. 2001, *System Engineering Fundamentals*, 223 s. Tillgänglig: http://www.dtic.mil/docs/citations/ADA606327. Hämtad: 16.11.2016

Gnome Developer. 2014, *The Importance of Writing Good Code.* Tillgänglig: https://developer.gnome.org/programming-guidelines/stable/writing-good-code.html.en. Hämtad: 13.11.2016

Microsoft. 2015, *Introduction to the C# Language and the .NET Framework.* Tillgänglig: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx. Hämtad: 11.11.2016

Microsoft. 2016a, *Welcome to Visual Studio 2015.* Tillgänglig: https://msdn.microsoft.com/en-us/library/dd831853.aspx. Hämtad: 12.11.2016

Microsoft. 2016b, *Team Foundation Server.* Tillgänglig: https://www.visualstudio.com/tfs/. Hämtad: 12.11.2016

Microsoft. 2016c, *Microsoft Excel.* Tillgänglig: https://products.office.com/en/excel Hämtad: 12.11.2016

.Net-informations.com. 2016, *What is Microsoft .Net Framework.* Tillgänglig: http://vb.net-informations.com/framework/what_is_net_framework.htm. Hämtad: 12.11.2016

Palermo, Jeff. 2009, *Hardcoding Considered Harmful - or is it?* Tillgänglig: http://jeffreypalermo.com/blog/hardcoding-considered-harmful-or-is-it/. Hämtad: 12.11.2016

Progress. 2016a, *Welcome to Kendo UI!* Tillgänglig: http://docs.telerik.com/kendo-ui/introduction. Hämtad: 12.11.2016

Progress. 2016b, *Upload HtmlHelper Overview*. Tillgänglig: http://docs.telerik.com/aspnet-mvc/helpers/upload/overview. Hämtad: 12.11.2016

Singapore Management University. 2012, *MVC Model*. Tillgänglig: https://wiki.smu.edu.sg/is480/IS480_Team_wiki:_2012T1_Team_Glocal#MVC_Model. Hämtad: 18.11.2016