University of Ljubljana
Faculty of Computer and Information Science

Dominik Zulovec Sajovic

Web application for visualization and processing of financial transactions

THESIS

UNIVERSITY STUDY PROGRAMME
UNDERGRADUATE
COMPUTER AND INFORMATION SCIENCE

Mentor: prof. Marko Bajec, PhD

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Faculty of Computer and Information Science is issuing the following work:

Topic:

The aim of this thesis is to develop a web application for the visualization and simple processing of financial transactions. The application should be targeted towards small and medium-sized companies that work in Britain. It should support various functionalities, such as data import (in various formats), display of typical attributes, identification od debtors, risk identification and other functionalities, elicited from potential users (these are to be defined by the thesis author). The development steps should follow a typical software development methodology and need to be documented.

Fakulteta za računalništvo in informatiko izdaja sledeče delo:

Tema:

V okviru diplomske naloge razvijte spletno aplikacijo za vizualizacijo in enostavno obdelavo finančnih podatkov (transkacij). Namenjena naj bo malim in srednje velikim britanskim podjetjem Omogočen naj bo uvoz v različnih formatih. Obdelava naj omogoča poljubno filtriranje vsebine, prikazovanje tipičnih količin, označevanje dolžnikov, ocenjevanje tveganj in druge funkcionalnosti, ki jih boste dorekli s potencialnimi uporabniki (te določite sami). Korake razvoja (vključno z zajemom zahtev, analizo in načrtovanjem) dokumentirajte skladno s prakso razvoja informacijskih sistemov.

I would like to take this opportunity to thank a couple people who contributed to this thesis. Firstly I want to thank Mr Antony Monaghan, CEO of SSGC Ltd. who allowed me to base my thesis on one of his ideas. He gave me the time and the resources to properly research and write this work. Most of all I would like to thank my Mentor prof. Marko Bajec, PhD. His friendly yet very professional approach helped me by making the whole experience less stressful. He was there every time I needed advice on the work. I couldn't ask for anything more than great professional advice and extremely fast response, which is exactly what I got.

0.1 Slovenski Povzetek

0.1.1 Uvod

Razvil bom spletno aplikacijo, ki bo omogočala grafično manipulirati s podatki. Aplikacija bo namenjena finančnim uslugam, kar pomeni da bodo podatki, ki bodo v aplikacijo vnešeni finančne vsebine. Uporabnik bo v vnaprej dogovorjenem formatu naložil podatke v aplikacijo preko več različnih možnosti (CSV, Excel, DB, ...) nato pa bo z njimi lahko manipuliral na vrsto različnih načinov.

Najbol osnovna funkcionalnost bo vizualizacija podatkov in preprosta predstavitev nekaterih ključnih številk kot so: profit, revenue, ... Možna bodo vsa mogoča filtriranja podatkov, od različnih oddelkov nekatere stranke do zaposlenih, ki so sodelovali pri poslu, do dolgov in ostalih finančnih atributov. Namen je da se iz podatkov izvleče vse kar se da, da se da uporabniku božjo moč nad manipulacijo in lahko poizve vse informacije od vsake ure posla do vsakega tedna, meseca, leta.

Aplikacija bo uporabniku pomagala loviti roke dolgov in loviti dolžnike, ki so uporabniku dolžni denar. Uporabniku bo omogočeno simuliranje različnih dogodkov, ki bodo pripomogli k znižanju tveganj in zvišanju profita. Če si bomo želeli ogledati kako izgleda naš posel če določeni strani računamo 10% več in nekaterim drugim nekaj % manj, etc. Različne situacije bo uporabnik lahko predvidel s pomočjo tega orodja in si tako prizanesel z nepotrbnim tveganjem v realnem poslu.

Dinamičen, hitro odziven in enostaven, temveč še vedno zelo napreden grafični vmesnik bo uporabniku omogočal užitek pri upravljanju z ogromno ne tako zanimivih finančnih podatkov.

0.1.2 Razvoj

0.1.3 Okolje Intrexx

Intrexx je nemški produkt za razvijanje programske opreme, ki omogoča lažje kontroliranje velike količine kode in poenostavi nekatere dele razvoja. Razdeljen je na veliko različnih delov poteka razvoja programske opreme, jaz bom opisal 3 od teh, ki jih bom uporabljal pri razvoju mojega produkta. Omenjeni deli so "Design", "Application" in "Process".

0.1.3.1 Design

Predstavlja, kot nam pove ime, izgled naše aplikacije. To ne pomeni, da tu urejamo vsak posamezen element v aplikaciji vendar tu ustvarimo okvir aplikacije, ki bo na vseh straneh enak (statičen). Izgled ustvarjamo s pomočjo "drag and drop" vmesnika, ki ga lahko natančno olepšavamo z CSS. Če želimo ustvariti kakšen element od začetka pa imamo tudi možnost spisati HTML z ata element.

0.1.3.2 Application

Tu opravljamo s klient delom spletne aplikacije. Uporabljamo HTML, CSS in Javascript. Kreiramo del aplikacije, ki je dinamičen. Ponovno imamo "Drag and Drop" vmesnik, ki nam omogoča hitro razvijanje aplikacije. Elemente preprosto povlečemo na kanvas in jih od tam naprej oblikujemo in jim dodajamo funkcionalnosti.

0.1.3.3 Process

V tem delu pišemo kodo, ki bo izvedena na strežniškem delu aplikacije. Koda je vizualno predstavljena z diagramom, ki ga ustvarimo kot želimo da poteka naša koda.

0.1.3.4 Javascript

Programski jezik javscript se uporablja v delu "Application". Imamo en velik JS dokument, kjer pišemo svoje funkcije. Te funkcije nato pripenjamo različnim elementom ob različnih dogodkih. Vse to nam omogoča Intrexx s svojim vmesnikom. Če pa želimo bolj specifično določiti kakšne kriterije, je seveda možno pisati javascript kodo v najbolj osnovni obliki. Intrexx podpira zunanje knjižnice kot so npr. JQuery.

0.1.3.5 HTML

HTML se v Intrexxu v večini kreira automatsko glede na "Drag and Drop". Kot uporabnik postavi elemente je nato generiran HTML, ki bo uporabljen v spletni strani. Uporabnik ima možnost izbire na kakšen način naj se kreira HTML (z elementi "table" ali z elementi "div"). Seveda pa je uporabniku dovoljena možnost kreirati del HTML-ja kot si sam želi.

0.1.3.6 CSS

CSS uporabljamo v "Application" in "design" delu Intrexxa. Kot sem že omenil Intrexx podpira zunanje knjižnice in tako dovoljuje uporabo tehnologij kot so npr. Bootstrap. Intrexx pa ima tudi svoj stil, svoje CSS stile, ki jih lahko uporabimo na različnih elemntih.

0.1.3.7 Groovy

Kot omenjeno v delu "Process" pišemo strežniško kodo in sicer se uporablja programski jezik groovy. Groovy je se uporablja na javini platformi in ima podobnosti z jeziki kot so: Ruby, Python, Pearl in Smalltalk. Intrexx ponuja vrsto knjižnic za efektivno in varno uporabo kode kot npr. varno poizvedovanje po podatkovni bazi in podobno.

Table 1: Vizualizacijska orodja

Priljubljenost	Ime vizualizacijskega orodja
1.	D3.js
2.	FusionCharts
3.	Chart.js
4.	Google Charts
5.	Highcharts
6.	Leaflet
7.	Dygraphs

0.1.3.8 Velocity

Velocity je programski jezik, ki tudi sloni a javini platformi, vendar pa se za razliko od Groovy-ja uporablja na klientu. Velocity uporablja dobro poznani "Modal-View-Controller" programerski vzorec, ki nam omogoče poizvedovanje po podatkovni bazi in tako omogoča uporabo AJAX-a.

0.1.3.9 SQL

V Intrexx-ovem delu "Application" Imamo možnost kreiranja Podatkovnih gruč, ki so v podatkovni bazi predstavljene kot navadne tabele. Intrexx priporoča uporabo SQL Server-ja pred ostalimi podatkovnimi tehnologijami kot MySQL, Oracle, itd.

0.1.3.10 Google Charts

Vizualizacijo podatkov bom predstavil s pomočjo Googlove knjižnice za ustvarjanje grafov, Google Charts. Obstaja veliko različnih knjižnic za grafično predstavitev podatkov kot je visno v tabeli 1.

Razlog, da sem sem odločil za googlovo je, ker je kvalitetna (razvidno iz uporabe, na 4. mestu) in ponuja ogromno dokumentacijo. Od naštetih knjižnic je največ dokumentacije na voljo pri googlovi knjižnici kar je za

pričakovati pri takem velikanu kot je google. To da je Google ogromno podjetje je še dodatni razlog, da bom uporabil to knjižnico, saj prihodnost ostalih ni tako sigurna kot je googlova. Po vsej verjetnosti bo Google obstajal še zelo dolgo in tako ohranjal podporo svoje knjižnice.

0.1.4 Ciljna publika in konkurenca

0.1.5 Mala in srednja podjetja

To so podjetja, z največ 250 zaposlenimi uslužbenci in največ 50M evri prometa letno. Razlog, da so ta podjetja naša tarča je, ker smo tako podjetje tudi mi in v podjetju poznajo to področjo, ter zato ker si večja podjetja lahko privoščijo svetovno bolj priznana orodja.

0.1.5.1 Sage

Sage je računovodsko orodje, večine SME podjetij v UK. Omogoča izdajanje računov, opravljanje s strankami in dobavitelji, in na splošno vse mogoče v povezavi s financami v manjših podjetjih. Kar pa sage ne omogoča je vizualizacija podatkov in predvidevanje situacij ob določenih pojavih (zvišanje računov, rast davka, ...). Tak je sage razvijal v preteklosti vendar ga niso nikoli dokončali. Zato razvijam tak produkt, ki se lepo ujame z potrebami, ki jih sage ne pokrije.

0.1.5.2 Excel in konkurenca

Podobno funkcionalnost ponuja zelo znano Microsoftovo orodje Excel. Torej pojavi se vprašanje zakaj ne bi namesto mojega orodja uporabljali excel ? Razlogov je več:

- Excel ne ponuja dobrega in enostavnega grafičnega vmesnika. Uporabnik mora poznati vsaj nekaj osnovnih oprijemov z excelom, da ga lahko učinkovito uporablja. Pri mojem produktu tako znanje ne bo potrebno.

Namen je, da se vmesnik naredi tako enostaven in zanimiv, da bo uporabnik skoraj, da voden s strani aplikacije.

- Če želimo podatke vizualizirati v excelu in z njimi manipulirati, jih moramo vstaviti v excel, ki pa ne sloni na oblaku. Podatki so shranjeni lokalno in tako to ni najboljša možnost za distribucijo, ki jo vsekakor nameravamo izvesti.

- Za razliko od Excela bodo podatki in operacije nad njimi potekale v SQL, kar omogoča sledenje vsem akcijam, ki so bile izvršene.

Contents

List of used abbreviations

kratica	angleško	slovensko
SME	Small to Medium Enterprise	Mala in srednje Velika podjetja
API	Application Program Interface	Programski Vmesnik
CEO	Chief Executive Officer	Glavni Izvršni direktor
CFO	Chief Financial Officer	Glavni Finančni direktor
GP	Gross Profit	Bruto Dobiček
NP	Net Profit	Neto Dobiček
UK	United Kingdom	Velika Britanija
HTML	Hyper Text Markup Language	Označevalni jezik za izdelavo spletnih strani
CSS	Cascading Style Sheets	kaskadne stilske podloge
JVM	Java Virtual Machine	Javin Navidezni Stroj
SQL	Structured Query Language	strukturirani povpraševalni jezik
AJAX	Asynchronous JavaScript and XML	Asinhronična Javascirpt in XML
JSON	JavaScript Object Notation	Javina skriptna objektna notacija
GUI	Graphical User Interface	Grafični Uporabniški Vmesnik
JS	Java script	Javin Skriptni Jezik

Abstract

Title: Web application for visualization and processing of financial transactions

Author: Dominik Zulovec Sajovic

Abstract: In this document, I will be describing the process of developing a financial visualisation tool to manage risks in your business.

I will describe the workflow of developing with Intrexx, which is the software I used to create my product. All other tools, libraries and APIs I used will get a description and reason of usage as well. The document includes a description, of the product field and products that are already on the market.

The product is made for Small to Medium Enterprises in the United Kingdom. The idea of the product is to apply to most of the SMEs throughout the UK. This is the reason a lot of things such as data format are generalised.

Keywords: Finance, Visualization, Risk, Intrexx, GoogleCharts, Simulation, Profit, Filtration.

Povzetek

Naslov: Spletna aplikacija za vizualizacijo in obdelavo finančnih transakcij

Avtor: Dominik Zulovec Sajovic

Povzetek: V tem dokumentu bom opisal proces razvijanja finančnega orodja za vizualiziranje in opravljanje s tveganji v finančnm svetu. Opisal bom postopek razvijanja z nemškim orodjem Intrexx v katerem sem razvil svoj produkt. Tudi za vsa ostala uporabljena orodja, knjižnice in API-jie, bom opisal razlog uporabe in način razvoja. V tem dokumentu bo vsebovano tudi področje mojega produkta in ostali produkti, ki mojemu konkurirajo in so že na marketu. Produkt je narejen za majhna in srednje velika podjetja v Veliki Britaniji. Ideja produkta je, da je narejen tako generično, da ga je mogoče uporabiti skoraj v vseh podjetjih takega opisa. To je tudi razlog, da je v produktu veliko stvari generaliziranih, kot naprimer format podatkov.

Ključne besede: Finance, Vizualizacija, Tveganje, Intrexx, GoogleCharts, Simulacija, Dobiček, Filtriranje .

Chapter 1

Introduction

Finance is a field that deals with the study of investments. It includes the dynamics of assets and liabilities over time under conditions of different degrees of uncertainty and risk. Every company no matter how big or small has a finance department or at least some people dealing with finance operations for the company. They are all dealing with different ways of earning and spending money, however, the format of data that describes every sold and bought service or item is very similar to all the companies. At least all small to medium enterprises (SME) in the UK to which my product will focus on. Imagine a format that would save every, or at least most of (99%) financial operation for SMEs in the UK. With that, we would have a database including all the data in the same format, which means generic operation through all the data.

Now that I shortly explained the financial data and its format, I would like to introduce another benefit of my tool which is now days considered as a requirement or a standard, however not respected in every company and can cause serious security and company insurance issues. All these financial data companies have has to be stored somewhere and the not safe way that is used way too often are simple excel sheets or even worse just on papers stored in huge folders and hit in a dark basement or cellar, etc. This kind of

data storage is not safe due to natural dangers on paper or bad organization in case of excels sheets. For all the reasons above my tool stores data as it is standardized today, in a proper relational database and stores it on a cloud.

Combining the financial data facts, possibilities for expansion and storage options, I have created a demo tool (platform to build on) that takes the described ideas to realization.

1.1 Fields and Working Hypotheses

We can divide finance into many different subcategories or better called fields of finance. The fields are connected by the same data, the difference, however, is the way we see the data and the way we use it. For example, a transaction for one company can be viewed as bill and the same transaction can be a successful sale for the other company. I will describe 2 of the fields which were initially the cause for this tool to be created.

1.1.1 Credit Control

Every real-time action which produces some cash flow for a company can be counted as gained money even though that money doesn't yet exist (the company does not have it). Let's say we make a service to our client, however, we have an agreement that they will pay at the end of the month. So we have made an action that has brought us money, although we don't have the money yet. Given legal and professional agreements between client and supplier, the client is, of course, obligated to pay the money. However, in the real world, it is very rare that the client settles the payment at the agreed time. Usually, clients have their own clients and the payment to us depends on when they get the payments from their clients. These kinds of situations have created the well-known role of credit control. The person executing this role is chasing clients to pay the unpaid bills. He has to make sure the invoices are properly sent and that the client has surely received a reminder and an

official invoice. We can count this ghost money as a part of our earnings or we can discount them until we actually get the money. With my tool, this kind of distinction between the data is possible and gives us a great view of the earning that either has been made to us or will be made in the near future.

What is most important in credit control? It is not software that shows you the money you actually get or money that you will get, in credit control the human factor chips in. As mentioned above, the person who is chasing other companies to pay off their debts is the person executing the role of credit control. Most importantly that person needs to make sure to send the correct invoice to the company and do routine check-up call every week or so (how often do they make check-up calls is agreed in the company). So because there is a too much human factor in credit control I have decided not to count it as a field that gains advantage from my software. It is, however, a possible upgrade. If we imagine the software somehow tells us which companies are paying their debts on time and are most likely to pay on time, which companies are not and so on.

1.1.2 Cash Flow

A cash flow in its narrow sense is a payment (in a currency), especially from one central bank account to another. The term 'cash flow' is mostly used to describe payments that are expected to happen in the future, are thus uncertain and therefore need to be forecaster with cash flows. For example, we can imagine a debt that is supposed to be paid to us by our client in the near future (As described in the Credit Control section). By inserting forecasted data to my application we are able to see the predicted money on a visual chart and how it blends with other money inflow and outflow. This feature is amazing for managers, CEOs, CFOs and any sort of analyst who wants to view and manipulate the data. The hypothesis behind the tool is the ability to present new investments for the company, potential clients or any sort of income or outcome.

Imagine having a great idea that brings a fair amount of income to your company; however, it needs some investment to be realized. With this tool, you can fill up the data and present it to your bosses with a nice looking visualization backing up your idea, and it might earn you a great project or a promotion, etc.

Breaking the data down to hours allows you to see how well your business does on an hourly basis. This allows you to improve the management of your business by spreading the services and financial outcomes better. Of course, the hourly view is not applicable for every business but is very important for some. An airline company, for example, sells their tickets every minute and the data should for that reason be viewed even on a smaller scale than hours in some cases. Flight tickets after all change prices every hour or even less. For companies that don't need to see their data in an hourly format, can switch to daily, weekly or monthly format. This option, of course, is available for every user, it does not matter which format they primarily use.

The format of the data that we upload allows us to tell the software the difference from our payments our earnings and our overheads for example. Overheads are payments that we make in order to run our business, for example, office rent, electricity bill, internet usage, repairs, computers, etc. All the earnings we make from our clients are called revenue. For example, if we sell a chocolate bar for 5 pounds, our revenue is 5 pounds. Of course, we had to buy the chocolate bar ourselves first. So let's say we bought a chocolate bar for 2 pounds and sold it for 5. Then our revenue is 5 pounds, but our GP (Gross Profit) is 3 pounds as that is the actual money we made (Not regarding the tax). Now let's see how overheads come into all of this. Let's say that in the month of March we bought 100 chocolate bars for 2 pounds. We also sold 100 chocolate bars for 5 pounds. When we subtract the amount of sold from the amount of bought we get 300 pounds of GP. To make these sales we had an on-line store set up and the hosting for the website cost us 55 pounds in the month of March. When we subtract the 55

pounds of overheads from our GP, we get 245 pounds, which is our NP (Net Profit) in the month of March. This is exactly the amount of money we got in March.

As said my tool allows us to differentiate from payments, billing, overheads, etc. So we are able to see for every hour, day, week or month how much is our Revenue, GP, and NP. This also provides a great view into our finances from the standpoint of how we arrange our overhead payments. Since overhead payments like office rent usually come once a month or once a year even, we are not able to see very precisely how it mashes together with our incomes. With this view, however, we are able to see how we should arrange our overheads in order to prevent our business to be in a negative state at any point (hour, day, week ...)

1.1.3 Analysis and filtering

The final field of mention belongs to simple views of your financial data. Every line of a transaction is it a payment, a bill, an overhead or whatever, can be grouped by a similar or different attribute. In our situation

I have allowed filtering by 3 different categories. Every line of a transaction or however you want to call a single finance action must be defined by 3 categories. I will explain more about these categories in the further sections (Data upload). For now, it's just important to now that every action belongs to one of each 3 different levels and sub levels. One of these levels is the client level. So every action belongs to one client. For example, if we got paid then the action belongs to the client that paid us. If we paid someone than the client is ourselves. This way we can filter all of our data by every client we have had deals with. Moreover, instead of 1 level of filtering we have 3 levels. For example, each client has its own portfolios and each portfolio has its own locations. Let's say our client is McDonald's. It has 3 portfolios: Basic sales, Drive through and McCaffe. Each of these portfolios has its own location like London, Oxford Street 80 and London Camden Princess Street 76. Now we are able to determine each action by its exact attributes. We can now

say a transaction happened with McDonalds' on a drive through and it was in London Oxford Street 80.

Following these guidelines, we can filter the data down to its very location or in other businesses down to every product, etc. This brings another extraordinary look at our finances especially when we combine it with our timeline filtering.

1.2 Thesis Objectives

Objectives of this thesis are to describe software that can be applied to any or at least most of UK SMEs. Throughout the document I will describe some of the advantages this tool has before other tools like excel and I will also explain how exactly it can be used in the real world. Very important objective does not really need a description but was already accounted upon creating the tool that is the simplicity of the software. While creating my product the objective was to make it as simple as possible to use, so it doesn't need any additional training to be used.

Lastly, the objective of this assignment is to create, demonstrate and describe a tool that could be sold to companies (SMEs) in the UK in order to gain a financial profit.

1.3 Methodological Approach

In this section, I will describe the methods I used to create each part of my tool. I will start by explaining the development process in Intrexx which is the software I used to create my tool. Following I will describe each part of the application and how I developed it, which language I used and what problems I encountered. Of course, I will also provide reasons to why I chose certain ways to develop the parts of the application.

1.3.1 Development Technology

As mentioned in the introduction part of this section, I have used a software developer system called Intrexx, to develop my application. The company I am working at is using Intrexx 6.0, which is currently not the newest edition since version 8.0 has already been released. According to Wikipedia this is the short summary of Intrexx:

"Intrexx is an integrated cross-platform development environment for the creation and operation of web-based applications, enterprise portals and intranet portals. A portal is created according to the drag and drop principle, without the need for programming."

Let's touch the last part of the definition, "Without the need for programming". This is true, for some very simple applications, like forms or view tables and so on. However, for any larger more complex application, programming is of course needed. Intrexx provides many other benefits that usually take some time to programme but are done much easier and faster in Intrexx. For example user groups or portal installation, etc. Like it is with building any kind of web application Intrexx also uses server-side programming language, client-side, HTML and CSS and a Database.

On the server side, Intrexx uses a programming language called Groovy. It is a programming language based on the java platform. Like Java it is Object-oriented. It gets dynamically compiled to java Virtual Machine (JVM) and can be used with plain Java along with its libraries. Groovy uses Java-like curly-bracket syntax. Most Java code is also syntactically valid Groovy, although semantics may be different. In Intrexx we use groovy in terms of processes. What I mean by that is, we define a process and add some groovy code to it. Now we have a process which we can use multiple times and not repeat the code. Processes can be joined together to chains, meaning one gets called after another; this is done visually by drag and drop. Processes are usually called for harder calculations to be done on the server or for Database operations, so nothing unusual here.

Intrexx can be connected to a database and then automatically creates new tables by a simple click in the Intrexx Graphical Interface. Tables are called Data groups and are mostly used for simple application with forms, view tables, drop downs and so on. There is of course also a possibility to manage the database from whichever other Database managing tool you prefer. In my company, we are using SQL Server and we have Intrexx connected to it. For developing this tool, I used Intrexx Data groups to create my tables, however, later I managed them by Microsoft SQL Server Management Studio.

And for the client side in Intrexx we have a nice design platform for creating GUIs via drag and drop. While this is a very easy and nice way of doing so I rather design my interfaces a bit different. To some degree, I used the design platform to simply place my code visually to the correct side. For example, the left side toolbar code was placed to the left of the screen; the code for the slider was placed where the slider initially stands. I also used some of the inputs for simple drag and drop. A normal text field that I used for searching and a used a big div in the middle of the screen which I later used for the visualization chart. So because a lot of my application is dynamic and designed from my own head I used only some basic designing platform elements and others I created by coding.

On the client side, Intrexx has HTML and CSS as any other web application. It also supports javascript and initially by itself includes JQuery. I manually included Twitter bootstrap for design and responsiveness.

Besides these usual technologies that are used on the client side Intrexx also uses Apache velocity. Apache Velocity is a template engine based on Java. It provides a template language for referring to objects which are defined in Java code. Its main objective is to declare a clear difference between the business and the presentation part in a Web application (MVC). I mostly used velocity for calls to the database or for the use of AJAX in connection of javascript and Database calls.

1.3.1.1 HTML

HyperText Markup Language (HTML) is a markup language. It is standardly used to create web pages and web applications. Alone it presents only a structure of a page. But together with Cascading Style Sheets (CSS), and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Browsers like, Google Chrome, Microsoft Edge, Internet Explorer, Safari, etc. receive HTML documents from servers. Then the browsers render the documents into multimedia web pages.

1.3.1.2 CSS

Cascading Style Sheets (CSS) is stylesheet language. We use it to describe the presentation of a document written in a markup language, like HTML. Usually it is used to set the visual style of web pages written in HTML, XHTML or HTML5, however, can be used for any XML document. As said in the HTML section, CSS is a part of a trio that forms a cornerstone technology used in most websites, along with HTML and Javascript.

1.3.1.3 Javascript

JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It has been standardised by the ECMAScript language specification. By itself, it is not an object-oriented language yet it supports Object Oriented coding. It is prototype-based with first-class functions making it a multi-paradigm language. It doesn't include any I/O, to support networking, storage or graphics. It entirely relies on the host environment. As said before together with HTML and CSS it forms the core technologies for the World Wide Web.

1.3.1.4 Overview

In order to help with the understanding of the application, its architecture and functionality, I have made a diagram (Figure 1.1) which shows us where

each part is located and how do they communicate with each other.

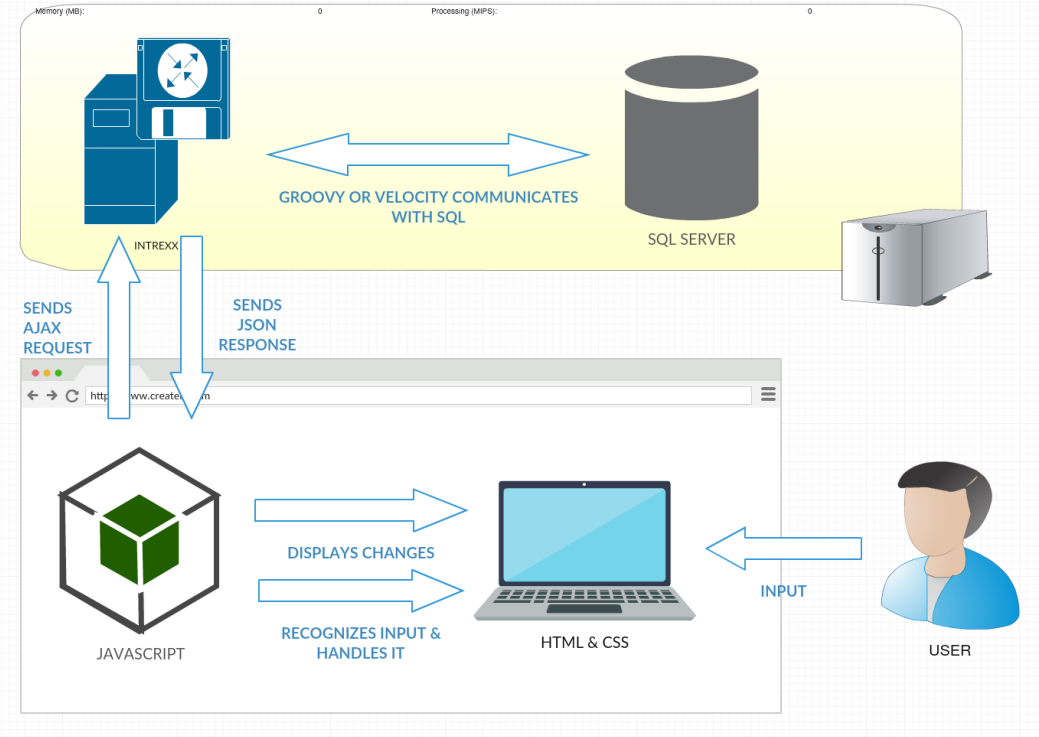

Figure 1.1: High-Level Architecture Diagram.

In the High-Level Architecture Diagram, we can see all the components of the application and the communication between them. Firstly let us discuss the client-side, so the bottom browser window. When a user makes a certain input or any kind of request on the application, which is displayed to him by HTML and CSS, it gets recognised by javascript. It recognises the request that the user made and then makes a decision. Either it deals with it on its own or send a request to the server and lets velocity deal with it. In case it lets velocity deal with it, they communicate via AJAX and exchange information via JSON and POST method. When results get back to the client-side. Javascript takes care of displaying the information to the user.

1.3.2 Analysis & Design

When I was first informed about this project and the whole idea of what it is supposed to do I thought it would be fun and a good learning opportunity. I tried to understand the big picture of the plan, however, in the beginning, I had a completely different presentation about it in my mind.

I was firstly instructed to spend 1 month simply learning how the finance runs in this company. I was given a large data set from sage, which is an accounting software used around the UK and whole Europe. With the help of the company accountant, I got to understand each field on the data set. Sage was a presentation of the official money the company got or paid, the dataset were basically invoices of money transactions. Later I was provided with another data set, this one was made by the company itself and stored in excel files. this data set represented all the services that were provided and the amounts of money which the company should have made. I took some time to play around with the data. I parsed all of it and saved it into my local database. After a while, I found some mistakes that were made in the past and actually cost the company around 500.00 GBP. This was all a great learning experience to prepare me for the product I was supposed to make. After my training month, I set down with the CEO again to discuss the plan and usage of the application. At the end, we came to a conclusion that it should be used as it is shown in the use case diagram Figure 1.2.

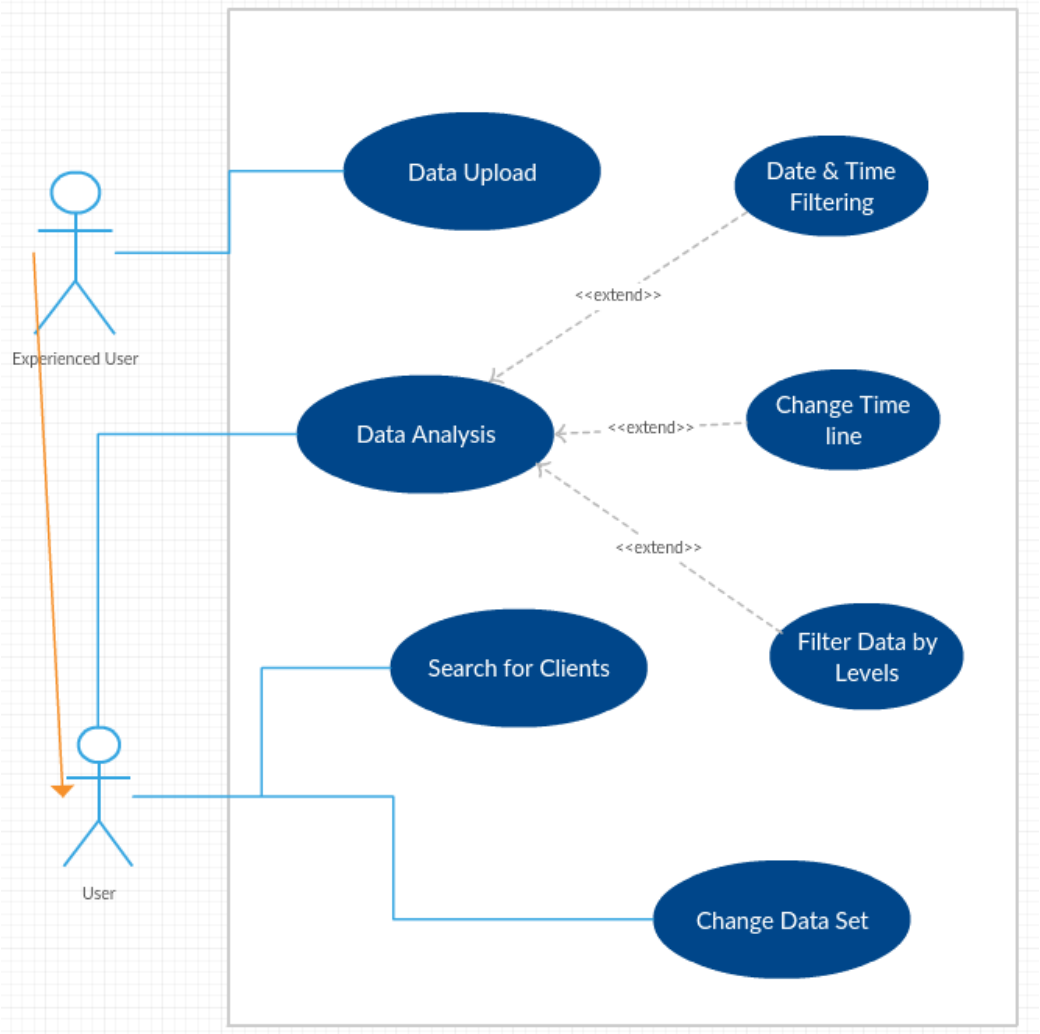

Figure 1.2: Use Case Diagram.

Base current use

1. An Experienced User Uploads the data to the system.
2. The system starts the calculations, and meanwhile presents the user with a animated loader symbol.
3. When the system is done with the calculations and all the data is stored, it informs the user.

4. The user selects the data set he or she wants to work with.
5. The system loads the data and lets the user know when its done.
6. The user searches for the client or supplier they want to analyse.
7. The user then select the client or supplier.
8. The System Shows all the data for that client.
9. The user adjust the timeline where he wants to inspect the data.
10. The user sees that there is to much data and therefore changes the timeline view to hourly format.
11. The system displays the data for exact filters the user applied.

Alternative current use

1. An Experienced User Uploads the data to the system.
2. The system starts the calculations, and meanwhile presents the user with a animated loader symbol.
3. The system finds an error in the uploaded file and stops all the calculations.
4. The System lets the user know there was an error and that the data was not saved.

Alternative current use

1. The user adjust the timeline where he wants to inspect the data.
2. The System finds out there is no data in the time the user specified.
3. The system Tells the user there is no data in that tie frame.

Alternative current use

1. The User searches for a client
2. The system finds out there is no result to the string that the user provided.
3. The system tells the user there is no client by that string.

Regarding Analysis and design, I only focused on specification requirements. Anything more than that, I didn't find necessary, since the architecture is more or less defined by the used technology mostly because of Intrexx and its forced MVC framework.

Chapter 2

Application Development

2.1 Data Upload

The first part of my development process was to upload the data to the cloud (in the database) with which I could later work with. So I had to come up with a format to ensure it would cover most of the businesses in the UK and would be able to be configured accordingly. For this, I was working with the finance team and the CEO of the company in order to come up with the most useful format. The format we came up with looks like this.

Client	Portfolio	Location	Employee Number	Start Time
McDonalds	McCaffe	Oxford Street	23	03/09/2016 09:00

End Time	Bill	Pay	Unit	Calculator
03/09/2016 15:00	10.0	7.0	6	MULTI

So let me explain the format. The first three columns are connected to each other. They describe the Who did we have our dealings with and what was It about. Each client has portfolios and each portfolio has its own locations/products/services, etc. In our example I am using location as that is what my company is dealing with. Next is the employee number which is optional. It tells us who was making the service. Start and finish time

are used to calculate how long did the service take and to record it on our timeline. The bill column means how much we charged our client and the pay column means how much we spent to provide the service. The units is another field that is optional and is there to tell us the amount of something doesn't matter what. The most important field is the calculator. We decide on certain code names like "MULTI" to tell the application how to calculate the Bill, Pay, and Units. For example, the MULTI calculator is the most basic one. It takes the amount in the Bill and Pay columns as the amount we charge and pay for in 1 hour. In the example above it simply multiplies 6 (The hours worked) with 10.00 and the result is how much we bill the client for. We can imagine some other calculator like "Double multi" which does the same only it multiplies everything by 2. This is in the UK used on Bank Holidays.

So this is the format we came up with and we believe can be applied to most of SMEs in the UK.

For Data Upload my application currently supports upload in a tab delimited text file format. Later, of course, the idea is to be able to insert it into excel file or most common delimited files like comma delimited, or to insert data directly from a database source (with the same format).

2.2 Algorithm for Data Calculation

After the file is uploaded and the data is read and recorded in the database, the application automatically starts all the necessary calculations. The time that the calculations take, depends on how big the data file is and how long are the services (Difference between start and finish time).

The calculations are done hourly. By that, I mean that for every line where a service was recorded, the algorithm calculates how much money was earned or lost. Calculations are done according to the "Calculator" field. For example, if there was an 8-hour shift, the algorithm makes 8 new lines, each one representing one hour and how much was earned and lost in it.

One problem can appear if we upload a service that lasted for 1 month for example. Then the algorithm would create as many new lines as there are hours in that month. That would make the calculations slow and ineffective. For this reason, the service difference has been limited to 48 hours. The limit is applied only to some calculators like "multi" or "double multi". If a user desires to upload a line that shows data for an entire year for example, then a different calculator has to be applied.

By now you should understand how data is saved and stored.

To summarize:

- If it is labelled with the most basic calculator, which calculates by the hour, then each line represents one hour.
- Example: We upload an 8-hour shift; it transforms it into 8 lines in our table.
- If there is a different filter each line can represent whatever we agree the calculator will mean.

Previously I also mentioned that the data can be viewed 4 different ways: Monthly, Weekly, Daily and hourly. Yet now I only talked about storing it hourly. That is because I store the data only hourly and when I want to present it in a different format I simply select it from the database grouped by the required timetable. The client side of the program sends a request to the server that it wants data grouped by day. The server side code executes a query that selects the data and groups it by day. The data gets returned to the client side via JSON format, where javascript takes care of the proper visualization via the google visualization API.

2.3 Methods of Data Manipulation

The last part of the previous section indicates to how do we get the data to the client side, but not so much to how do we use it. Here I am going to

explain how the data is stored on the client side after it gets back from the server and how it is manipulated in order to get the desired view.

Once the data is on the client side javascript takes over. The JSON format gets read by javascript and stores the data into a single array. That array needs to have a required format in order to be visualized by the google visualization API. With javascript, I transform JSON into the required array that has to 2 rows, one for headers and one for values. Headers are for example the days (if we are dealing with a daily view) and the values have the GP on that day. The array is ordered by days so when a user select a certain time period that he or she wants to look at, the program simply cuts the array from both sides accordingly. So the client side takes care of timeline filtering. The client-level filtering, however, happens while selecting the data. That means that when the user chooses some a client they want to view, the data gets selected from the database again, the only difference, this time, is that it is filtered as the user selected.

2.4 Graphical User Interface

Here I will describe the user interface and how it is used. The idea of the design was to make it look very high tech and professional while also being easy to use. Before I started the design I decided I would base the design on windows. By that I mean the design of the application would have similarly designed components as windows. Everything would be more of a square or rectangle shape and wouldn't have borders. Only buttons are used as circles. I applied this kind of attributes where ever it was possible. For some components, however, it just looked better leaving them as they are.

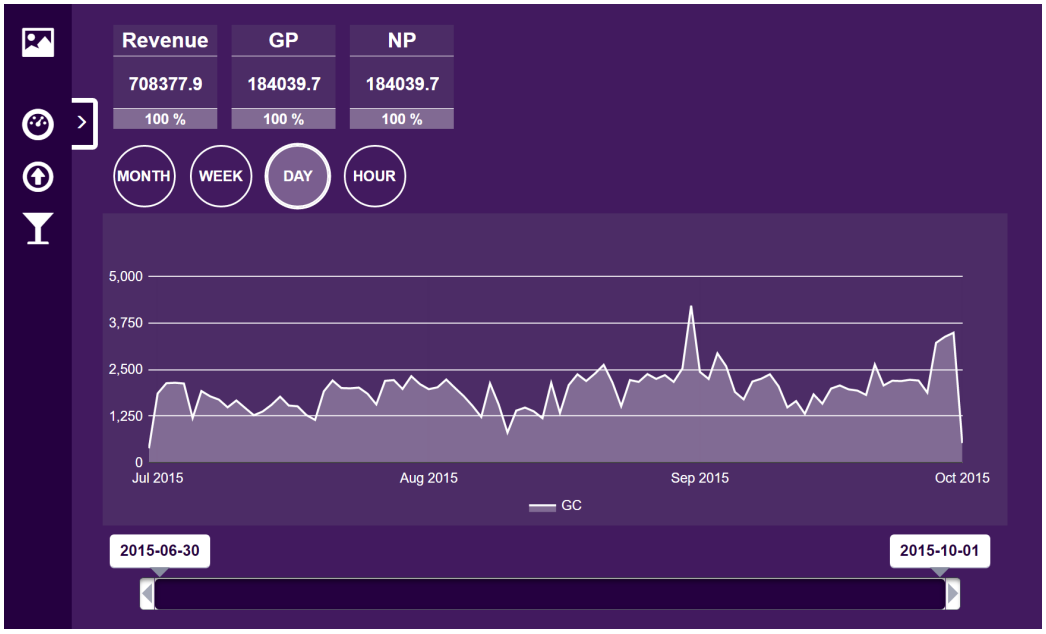

Figure 2.1: GUI normal.

If you look at Figure 2.1, you can see the main graphical user interface of the application. On the GUI we see the chart, the time line, the time line selectors (day, month, hour, week), and the numbers which show how much do we actually see on the chart. On the left side, we see the fixed sidebar which allows us to navigate through the application. The application currently has only two screens, the one you see in the picture and a file upload screen. The sidebar also has a button which allows us to expand the sidebar. On the event of clicking the button, it expands the sidebar and shows the filters as we see them in the bottom picture.

In the sidebar, we see filters which are the 3 levels of portfolios. Currently, we see the clients (Some are selected and some are not). We can also expand the clients to see their portfolios and select only portfolios. Further, on we can expand the portfolios into locations as seen in Figure 2.2.

Figure 2.2: GUI lower levels.

Here the sidebar is expanded as are the client filters. We can see the Client, its portfolio and all of its locations. This Data suits the company I am working for, which had clients, each client has portfolios and each portfolio has sites. These sites are the locations of manned guarding since the company is dealing with sending officers to guard locations. Besides the filters, we also see a drop down at the top which allows you to change between groups of data which you might have uploaded. For example two different data sets for two different companies. Or one data set for your past business and one for the future (predicted) one. Under the drop down menu, there is a text box which allows us to search for our clients. When the focus is on the textbox and we press any key the filtering process begins. It looks for substrings in client names, portfolios, and locations and then displays the ones that suit the string. And lastly, we have a select all button, which simply selects all filters.

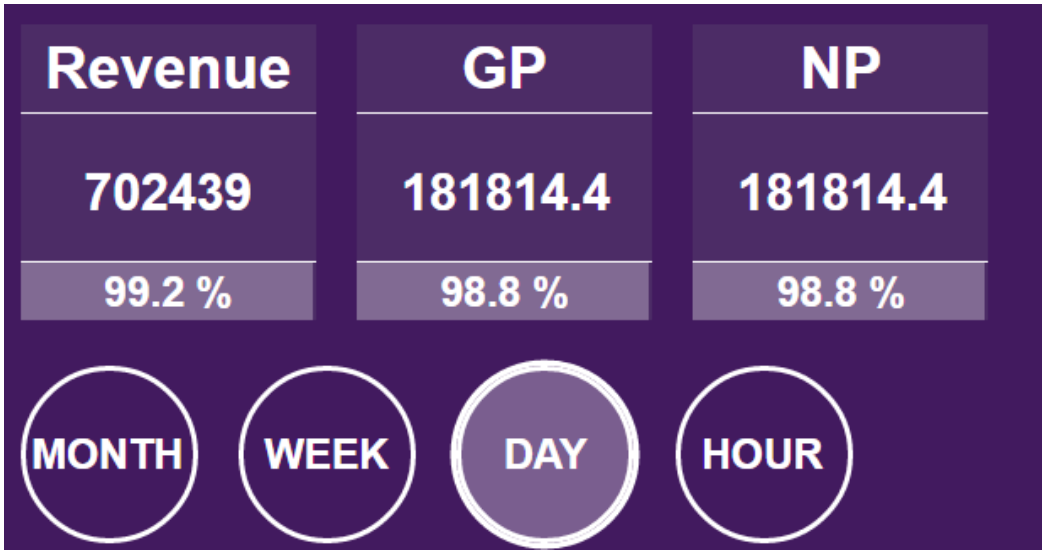

Figure 2.3: GUI top.

If we look at the three square labels at the top of the GUI (Figure 2.3) we see it's showing Revenue, GP, and NP. Revenue represents all the money that we got from our services or our sales. The GP or Gross profit represents the difference between the revenue and the money we had to pay to execute the services. And the NP or the Net Profit shows us what money we actually got after we subtract all our overheads from the GP. On the bottom part of the GUI, we can see the timeline. The timeline is there to filter the data by dates. The filtering happens by dragging the time labels at each end of the timeline. On any change of the timeline, the filtering process occurs right away. The chart redraws again and the money numbers in the square labels change.

The main part of the GUI, of course, is the chart. It always shows the GP of the applied filters. In the future, selecting what do you want to chart to show (Revenue, GP, NP, etc.) is a good upgrade. The chart is a part of the Google Visualization API which I used in the development of this application. The explanation of adding the library to the Intrexx design process will follow in the technical analysis of the application. Even if the

chart doesn't give us the most information of the entire GUI it is the easiest to read and is the first thing you see when you first look at the screen. I used an area chart because it is even easier to notice than a line chart and is great to present us the flow of time. For future upgrades of the application, it is planned to have more possibilities of the chart to show the data.

Lastly, I will show and explain the data upload screen seen in Figure 2.4.

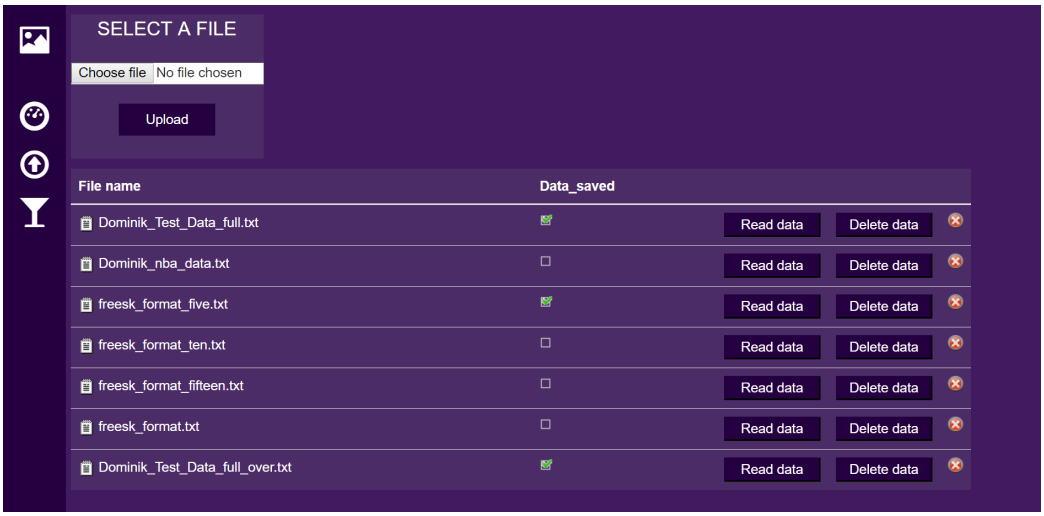

Figure 2.4: GUI upload.

In the top left corner, we see a squared label platform which serves to select a file and upload it. When the file is uploaded it appears in the table below. When it appears the little check box is unchecked, which mean the data is not read. After we click the read data button the program starts reading the data and if it finished without any error the check box becomes checked. Delete data simply deletes the data of the file from the database but keeps the file on the server. If we want to remove the file from the server we need to click the little X button at the right side of every row.

Chapter 3

Case Study

3.1 Technical Analysis of the Applications

The process of uploading data, parsing and saving it has already been explained. Here I will explain the process of creating the front-end part of the application. The front end is built with HTML, CSS, JavaScript, JQuery, Google Visualization API and the help of Intrexx framework. I tried to avoid using Intrexx here as much as possible since it doesn't provide good responsiveness. With Intrexx I created the HTML for the side menu because it allows using it as a sort of a layout to have it on every page. Other components I created with CSS and JS.

Let's start with expanding of the sidebar. When the button for expanding is pressed the sidebar slides in from the left side with help of CSSs transition animation. The part of the menu that slides in is before the event hidden 300px out of the screen. On the sidebar we have a drop down, a text box, a button and a list of elements containing clients designed by me. We can see it closely in Figure 3.1

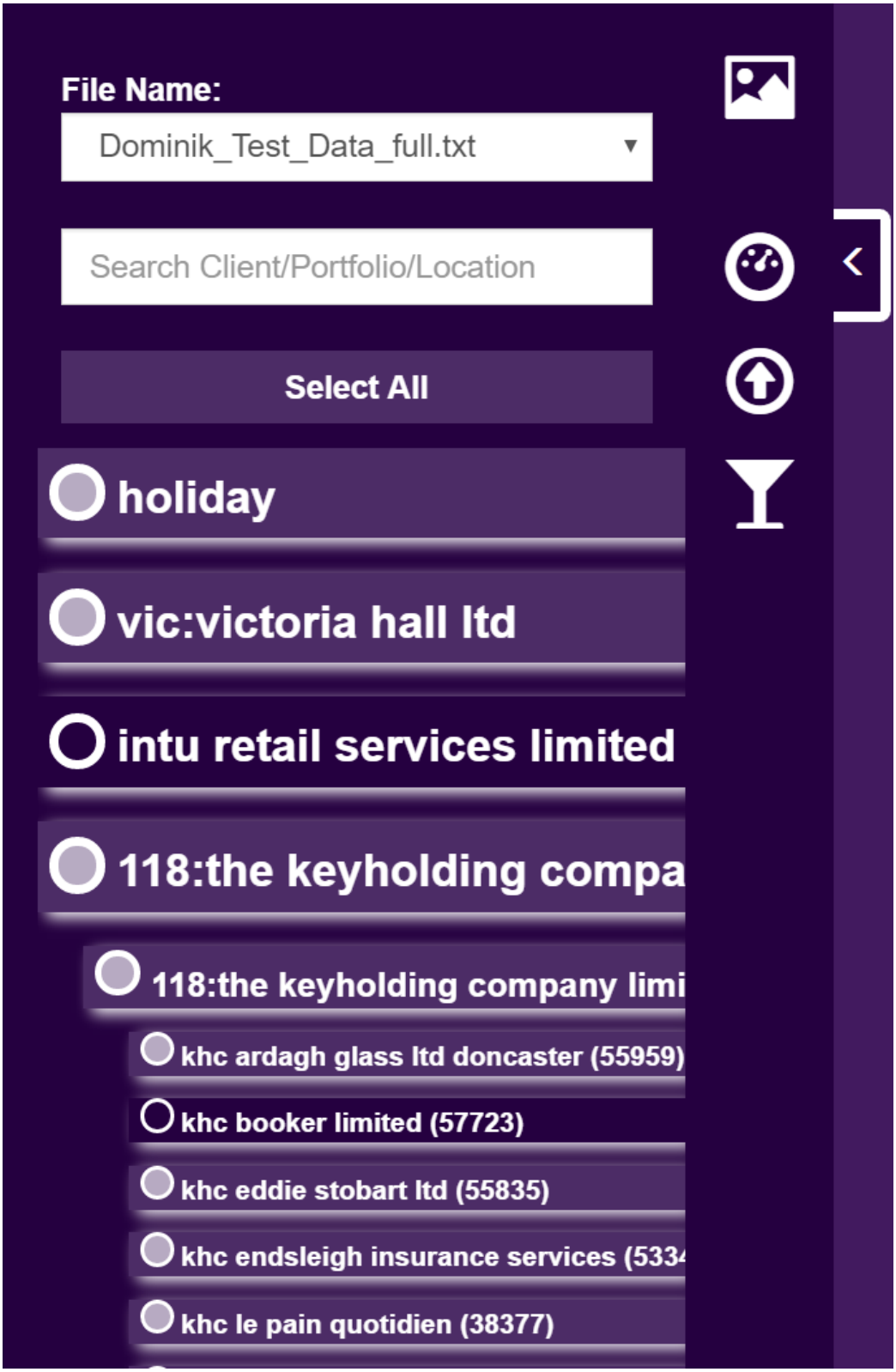

Figure 3.1: levels.

Each element is expandable when you click on it, a sort of a drop down opens beneath it which shows the portfolios. The same functionality is applied to the portfolios. Each element also has a circle button which presents the option of selecting it. On every select of a new element the application sends an asynchronous request to the server (AJAX) and gets the correctly filtered data back.

The next part I am going to explain is the square labels and circled button at the top of the application. Besides the reason of their design which I explained in the GUI section, I have to touch the functionality. The square labels simply change their number by AJAX requests. The percentage line at the bottom of the square labels gets calculated at each call of AJAX as well and the percentage is taken out of the complete data set. It is showing percentage between data currently shown and all data.

The circled buttons have radio button characteristic. By that I mean only one at a time can be selected. On selection, an AJAX request gets send to the server to retrieve the data grouped as we want (hourly, daily, weekly, monthly). The selected button also gets coloured when it is selected. In Figure 3.2 we can see an example of data presented by hours.

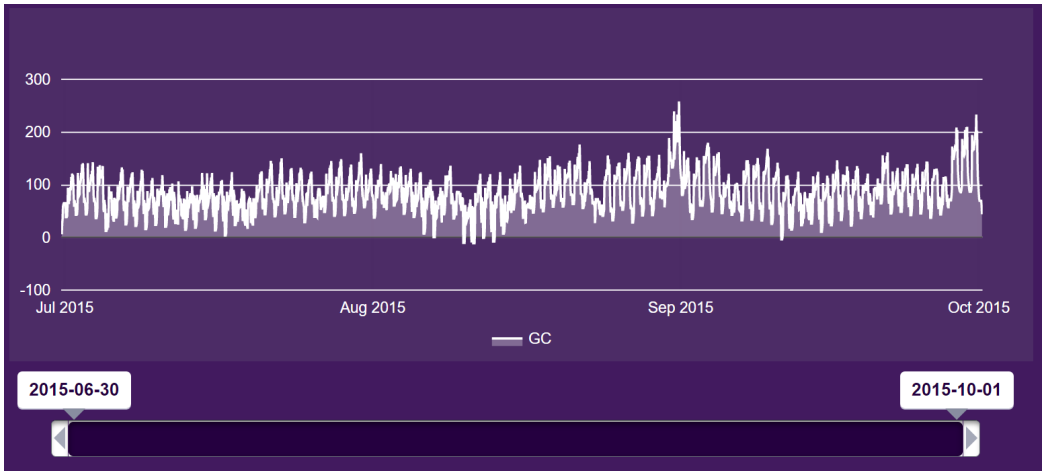

Figure 3.2: Hourly look.

We can clearly see there is a lot more data than in a daily view. So if we adjust the timeline (Figure 3.3) we can see at what time in a day were we making money and at what time we were not.

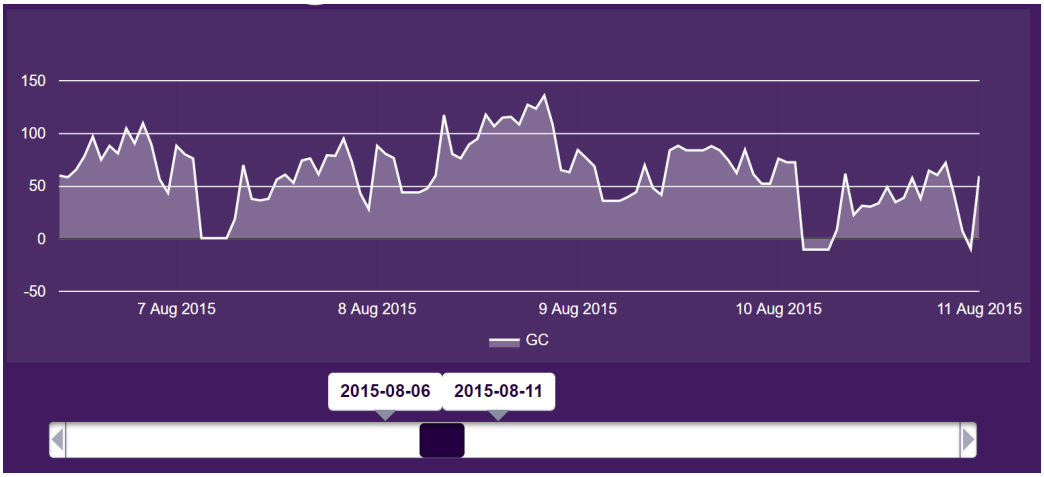

Figure 3.3: Hourly closer look.

The usage of the timeline has already been described in the GUI section, so here I will only explain the developing part. The component is a part of a JQRangeSLider. Inserting the library was a sort of a problem. Intrexx didn't allow me to add this library to the server and simply call it from there. This is why I simply copy and pasted the entire library between two ¡script¿ tags in the middle of the HTML code. After all, I had left to do besides implementing the functionality, was changing the colour. Intrexx has one huge CSS file where all the styling is stored. There I copied the initial slider style settings and later simply changed the colour into purple.

The last component to comment on is the chart. In the GUI section, I mentioned that the chart is a part of the Google visualization API. This means I had to insert that library into Intrexx. Unfortunately, this time neither copying the whole library to the code didn't help. The reason was that the library has to be inserted only when the HTML document is ready. So I had to use a way around typical library insertion and call it a synchronically.

I used JQueries' Document Ready function to call the library in via AJAX.

```
/*
 * Used in all the pages (on document ready)
 * Asynchorincally loads "Google Visualization API" library to browser
*/
function async_load(){
    $.ajax({
        url: '//www.google.com/jsapi',
        dataType: 'script',
        cache: true,
        success: function() {
            google.load('visualization', '1', {
                'packages': ['corechart'],
                'callback': fake
            });
            google.load('visualization', '1', {
                'packages': ['controls'],
                'callback': fake
            });
        }
    });
}
```

Figure 3.4: Async Load.

In Figure 3.4 is the code I call on document ready to load the Google JS Visualization API into the browser. After inserting the library successfully, adjusting the chart visually wasn't t hard. Google API provides great documentation about changing little parts on the charts like colour, the name of the axis, size, legend, and so on.

Altogether most of the front end was designed by me besides the Google Visualization API and the JQuery Date Slider. The largest part of it is JQuery, which enables all the user interactivity and functionality. A smaller part of it is provided by CSS which allows us to see the design and all the animated transitions. And HTML, as it is supposed to, serves as the structure.

3.2 Questionnaire

After the product was completed and labelled as version 1.0 beta, I gave it to our companies CEO and the Finance Manager to test it. Later I have prepared a list of questions for them in order to see the how the product serves them and where to they see a room for improvement. I gave them a week to use the application and later sat down with them to discuss the answers. Here you can see the questions I asked and the answers described in my words.

Did you find the application understandable to use?

This being one of the main objectives of my work I tried my best to make the application as easy to use as possible. Applications can always be improved in any perspective and the same goes for this one; however, the response to this question was very positive as they found the application very intuitive and easy to use. Their argument was that once you see the format and you upload the dataset, there is no question of what each component of the application does.

What were some of the problems you found while using the application?

The application has not been tested by anyone else except me before giving it to them, so, of course, there were so bugs to be expected. The finance Manager mostly stressed that once the filter sidebar is expanded the date range slider starts behaving differently. Functionality still works but visually it is not in the correct place. While the CEO was not so much focused on any particular bugs as bugs a part of every software, he was more focused on error handling and user experience. One example he mentioned was when the filters selected are not supposed to show any result, the Google Visualization API error note appears, instead of a self-designed message.

What would you add to the application?

Since the idea for the application came from the CEO he knows exactly what is missing for him or not so much missing as to what is still to be implemented in the future. Some of those future implementations are to configure the Calculator field as you want. Which means the user defines a formula how the system will interpret the number in each row. In the future, he also plans to have filters for overheads and the ability to choose what you want to see on the chart, not only GP. While using it he came up with another idea he said would be nice. That was to present all the sales on a map to see where exactly a certain transaction took place, in order, there is a specified location, which there is in the case of his business.

The Finance Manager who isn't aware of all the plans there are still in the future for this application, simply added he would like to have the option to add a row to a debt. And that the user could specify how often to remained or chase the client for this debt. The system would then automatically send an email to the client with all the attached documents needed to provide proof of the chased debt. The other functionality that he mentioned was to export the data in a format of your choosing when you are done filtering. So the ideas to filter the data as you wish and at the end click an export button to get all those rows back. That could be used for invoicing or many other benefits.

Would you use this application to help you get a better perspective on your business finances?

Again, a question mostly aimed towards the Finance Manager, as the CEO had the idea for the application, of course, he would use it and found it useful. The Finance manager divided his answer into two parts. Firstly he said if the application stays as it is now, I already find it useful for the bigger perspective and the bigger picture of our situation. Also while talking

on the phone with clients or suppliers there are many occasions where I am to specify them how much many they owe us or how much revenue have we made for them or any sort of data that they demand. This would help me greatly to get those answers way faster. The second part of his answer was that if I added the functionalities he described in the previous question, this would probably become the application he would use the most at work and would almost half his workload.

What did you like the least while using the application?

Regarding this question, they both agreed on the same thing. The application relies heavily on the front end, by that I mean while the user is interacting with the application there are a lot of changes happening on the client side (javascript). That is the reason the application sometimes works slowly depending on the computer it is running on and the resources available. They didn't like that sometimes it runs very slowly and needs to be refreshed in order to continue working. If there is no other way of fixing this they suggested automated refresh with saving the state of the filters.

What did you like most while using the application?

The CEO was pleased about the User experience the smooth transactions and visual effects. He really appreciated the fact that with this tool even without upgrades he will have an easier time to show present data to his clients and use it better in meetings. The finance manager said the best thing is that it can at least to some degree, for some small functionality right now, replace excel. He says Excel is neither that user-friendly nor that fast to get what you want. Also by using excel you can make much bigger mistakes or errors simply by clicking something wrong.

Chapter 4

Usage in Reality

Throughout the Thesis, I have been explaining the possible usages of the application. Here I will focus on the company and how they will use the application. It will be used in two different fields; the first one will be in the finance department and the second one for meetings with clients and presentations. Also, it will serve as a good official record for the CEO in order to see how each week is progressing. As explained before this is a security company. They have their own clients like stores, factories, supermarkets, that require security guards. This company provides security guards to their clients. For this service, the clients, of course, need to pay. Our company has regular meetings with clients in order to keep good relations and discuss the past and future business. In these meetings they also discuss finance and showing each client how they stack up in that field is very important. This tool will be used in those meetings as its primary functionality is exactly what is needed.

Besides using it for clients, future clients and suppliers the tool will also be used for weekly check-ups of the business. The money in this company is weekly, that means the income is being checked each week and the invoices are being processed, at least to most clients each week. At the end of every week, it is possible to see how successful or unsuccessful they were. With this tool it will be able to see if the week was good or not and if it wasn't it

will show why exactly was it not. The possibility of seeing which clients or which guards produced the most money and which ones produced less than they should have is going to play great importance to decision making for the future.

Also, I mentioned before that the financial manager spends a big amount of his day on the phone talking with clients and suppliers. These talks are very friendly even if it goes for an important matter, which is known for the UK. The talks mostly depend on facts; whoever provides more and better facts usually gets the better part of the business. Facts of course relay around money made or lost between the client and supplier. This application will help the finance manager get those facts together very fast, even while talking on the phone with the client or supplier. This is how it will represent a very good asset to the company.

Altogether the current version without any upgrades works as sort of a quick query visualizer to use for non-IT personnel. With that I mean by now it simply collects and displays the data in a very fast manner. It also compares it to the bigger picture and the entire business.

Chapter 5

Conclusion

Getting the data you want and present it as you want is a part of every business. Every day in every corner of the world people gather data, either from paper, computer, some other device, cloud or where ever. The most time-consuming part is not getting the data it's getting the exact data you need. We have seen that this application definitely covers that part it makes the process of getting the correct data easier and faster. We can definitely use this application to get the data quicker and faster. However, the real question is can we use it to manage risks? Can this application reduce the risks in this company?

Yes. The answer is yes and here is why. When nearing a contract deal with a client that has fixed shifts, which means we know when we will be providing service, so we will know when we will make money. If we know that, we can simply add that data into the application and we will see what is the smallest difference between charging clients and paying the guard, which will still bring us a certain amount of profit. When we add our overheads into the system we get a complete picture into our dealings with the client and if it is worth to take the risk.

The current system provides exactly that. It provides the visual into the data we want. Will it provide risk management? It will if we use it correctly, smart. So the risk management of the business depends on the user. Let us

conclude that the system does provide risk management under valuable and experienced usage.

If we take into account all the upgrades we were discussing throughout the document we get a completely different system. The features for credit control, service mapping, overhead filtering, calculator building, tax changing, bill forecasting and so on. That system would provide with real risk management. It would lessen the human factor which allows mistakes and replace it with software that is much more reliable.

This being a system to be used for a general business, not just this company, we would have to make much more research and talk with another kind of businesses. The application that provides risk management to any SME in the UK only depends on the format of the data and the format is definitely adaptable to most businesses. The only difference is how to calculate every row of data, which is defined by the calculator field. A conclusion is that this application can currently serve as a risk management system for this company and needs the calculator builder feature, in order to be applicable for other companies.

Bibliography

[1] Css description. https://en.wikipedia.org/wiki/Cascading_Style_Sheets, 10.9.2016.

[2] Html description. https://en.wikipedia.org/wiki/HTML, 10.9.2016.

[3] Javascript description. https://en.wikipedia.org/wiki/JavaScript, 10.9.2016.

[4] Oficial intrexx website. https://www.intrexx.com/en/intrexx, 10.9.2016.

[5] Oficial website for groovy. http://www.groovy-lang.org/, 10.9.2016.

[6] Velocity description. http://en.wikipedia.org/wiki/Apache_Velocity, 10.9.2016.

[7] Google charts library. http://developers.google.com/chart/, 30.3.2016.