Danmarks Tekniske Universitet

NETVÆRKSSYSTEM MED FJERNSTYRET FRONTEND/BACKEND TIL HARDWARE KOMMUNIKATION

Et Bachelorprojekt ved
DTU Compute & Fotonik
af
Frederk Haaning
s134521

I overensstemmelse med
de gældende betingelser
for et Bacheorprojekt

Juni 2016

Opgaven af Frederk Haaning, er godkendt af følgende:

Stig Høgh
Lektor
Projektleder

Lars Staalhagen
Lektor
Studievejleder - Networks Technology and Service Platforms

I samarbejde med Jens Grossmann Larsen
s135254

Abstract

The following report details the considerations for revising the construction of a product to a modern technology, but with the aid of theory and analysis, it is supposed to be understood as a general application for older models or existing technology to consider adapting a newer, emerging one that might be paradigm changing. A scale with an ethernet link is used to show what can be done through the use of clever system structuring, centered by the tools provided by Google Web Toolkit (GWT), this project has created a communication system that allows more complexity to be added to the system, yet simplifying the interaction between operator and equipment. Even though the system gets more complex, the structural & performance enhancing benefits using GWT proves to be superior for the object-oriented programmer. The project details how these conclusions are drawn, as well as how the relatively slower initial speed of project building with JavaScript pales in favor of the maintainability and scalability of a GWT project.

Indholdsfortegnelse

Indledning

Buzzwords som *The Cloud* og *Internet of Things* fylder mere og mere i vores hverdag, efterhånden også hvis ikke man beskæftiger sig med teknologi til dagligt. Derfor er det mere relevant end nogensinde at udvikle sine produkter med henblik på at imødekomme dette.

Dette gælder i allerhøjeste grad også mobile markedet, der har udsigt til større vækst, på trods af at være i høj vækst allerede globalt. Dette betyder at forbrugernes teknologiske enheder vil variere i forhold til hvad vi har været vandt til, og dette har affødt, at vi i denne rapport har valgt at beskrive processen om udviklingen af et produkt der tilpasses netop disse udsigter. Der er taget udgangspunkt i en vægt med ethernet link kapacitet, men gennemgangen kan projiceres til andre typer lignende produkter, og omstillingsprocessen som IoT er tiltænkt.

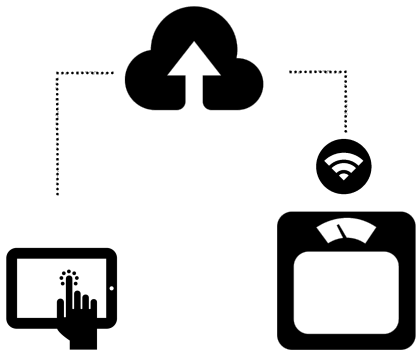
Konkret har dette projekt forsøgt at belyse hvordan man kan tage en eksisterende teknologi, her en vægt, revidere visse elementer, dataoverførselskapacitet, og på den måde få gavn af spirende teknologier i form af trådløs, centraliseret databehandling og formidling der tillader virksomheder at sænke produktionsomkostningerne og øge konkurrenceevnen. Ved at adskille betjeningen, der siden kan udgøres af vilkårlige enheder som eksempelvis en tablet eller smartphone fra datacentre eller fjernt beliggende servere, er det muligt at effektivisere. Til at opnå dette benyttes GWT, der både tilbyder en sikker, robust og strukturelt fordelagtig løsning på dette.

Kapitel 1

Teori & Analyse

1.1 Systemet

I takt med faldende elektronikpriser, hastig protokoludvikling såsom Bluetooth LE og WiFi, og med Internet og Things i horisonten, har motivationen for udvikling af produkter med mulighed for at opsamle og udveksle egen data aldrig være større. Systemet der udformes i løbet af dette projekt tager udgangspunkt i en eksisterende teknologi, en vægt bestående af et System on a Chip (SoC), og giver et kvalificeret bud på hvordan denne kan integreres bedre med henblik på spirende teknologier og deres samspil. Mere specifikt er der tale om en vægt der kan udveksle data med en bestemt hastighed, begrænset af enten det fysiske link eller den software der behandler forespørgsler, afhængig af modellen. Dette projekt foreslår en revidering af systemet, således at vægten, i stedet for at sende data til de forbundne enheder, sender data direkte til en server, eksempelvis ved, men ike begrænset til, at implementere Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS) eller Long-Term Evolution (LTE)/Long-Term Evolution Advanced (LTE-A) chips.



Figur 1.1: GWT Vægtinterface System

1.1.0.0.1 Det nuværende system foregår dataudveksling som følger:

1. Vært forbinder til vægt ved fysisk link.
2. Vært forbinder til den Telnet server der ligger indlejret i vægten på port 8000.

3. Vært sender godkendt kommando, mens vægten godkender denne og transmitterer den pågældende efterspurgte data.

1.1.0.0.2 Det reviderede system vil dataudveksling foregå på følgende facon, *jævnfør også figur 1.1*:

1. Fjernt beliggende server (*i en "sky"*) parres med vægt, nærmere bestemt vægtens IP.
2. Vægten transmitterer løbende data til server der præsenterer disse gennem en webside
3. Enhver anden enhed med de rette privilegier kan nu tilgå web-klient interface og betjene samt aflæse data, eksempelvis fra en tablet.

Måden hvorpå forbindelsen fra vægt til server løses på konkret i dette projekt er ved at forbinde vægten til en router der har statisk IP forbindelse til LTE nettet. Denne løsning effektiviserer således producenten af vægten og sparer arbejdet med at lave et fysisk behandlingspanel på vægten. Der kan være flere årsager til at ønske denne opsætning, der både tillader billigere og hurtigere produktion fra producentens side, mens der fra forbrugerens side kan imødekommes krav fra eksempelvis sundhedsregulerende myndigheder omkring sterilt miljø ved behandling af elektronik. Et konkret eksempel kunne være en fødevareindustri og afmåling af fødevarer, der nu kunne foregå eksternt fra vægten i et mere kontrolleret forløb.

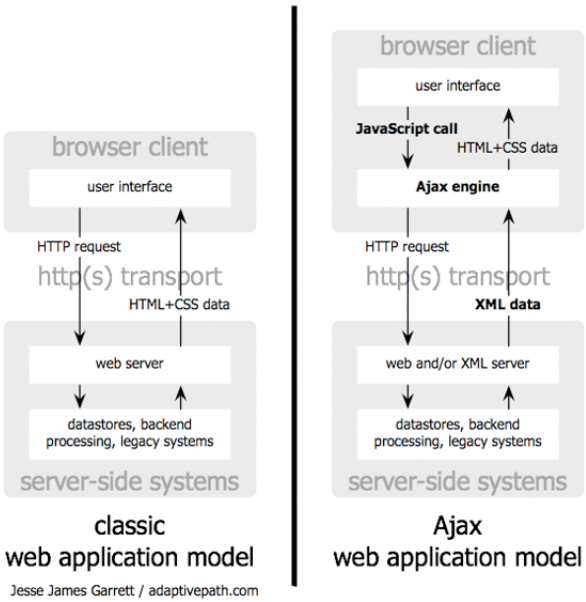
Ud over de økonomiske gevinster, er en anden motivation den effektivisering der medfølger løsningen, nemlig at den i modsætning til den gamle løsning er skalérbar. Ved at outsource de højere software lag væk fra vægtens hardware kan man nemmere udbygge systemer som eksempelvis brugerdatabaser og andre tilbygninger af teknologi der ellers ville kræve helt nye køb af vægt. På grund af web teknologier der beskrives nærmere i de kommende afsnit tillades også flere enheder at monitorere samme proces.

1.2 Google Web Toolkit

For at fokusere på at konstruere et stabilt system og Java kode kompleksitet, samt stadig at tilgodese behovet for at kunne tilgå brugergrænsefladen der benyttes til at håndtere vægten, valgte vi at benytte os af GWT og de værktøjer der tilbydes i dette. Særligt ved GWT er det nemlig, at den tillader cross-compilation fra Java til JavaScript, samt reducerer nogle konstruktioner til kommunikation mellem klient og server således at de trivielle opsætninger håndteres af GWT mens udvikleren er overladt til at opbygge resten.

1.2.1 AJAX

Modsat traditionelle web applikationer, der ved server-forespørgsler var nødsaget til at indlæse hele HTML sider på ny ved endt udførelse, er der med teknologiens udvikling af hovedsageligt JavaScript og dets afledte biblioteker muliggjort en metode til at sende visse forespørgsler fra klient til server uden den overhead der er forbundet med at hente den samme klient data. Denne samling af teknologier går under fælles betegnelsen Asynchronous JAvascript and XML (AJAX) og beskriver hvordan at en forespørgsel på en web applikation bliver sat i en kø der håndteres parallelt, asynkront og i baggrunden, der på den måde tillader brugeren at interagere med web applikationen samtidig. Dette tillades nærmere bestemt ved en seperation af dataudvekslingslaget og præsentationslaget.[1]



Figur 1.2: AJAX model

Ud over CSS & HTML til at præsentere web applikationen for klienten, alle administreret under JavaScript, er AJAX opbygget af følgende teknologier:

XMLHttpRequest (XHR) er et Application Programming Interface (API) der er kernen i Hypertext Transfer Protocol (HTTP) & Hypertext Transfer Protocol Secure (HTTPS) kommunikation. Ud over at tillade de gængse kommunikationsformer tilbyder API'en at implementere sit eget kommunikationsformat på URI standard, hvilket er hvad GWT benytter til at transmittere Java objekter. XHR og dets relevans for systemet er meget aktuelt grundet det sikre lag HTTPS, beskrevet nærmere i senere kapitler. [2]

JavaScript Object Notation (JSON) Et standardiseret format, der kan gruppere & indicere data som nemt kan læses til hurtige lookups.

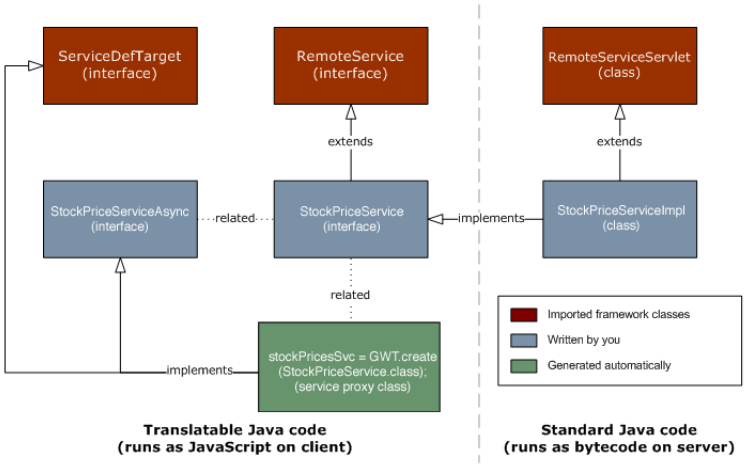
Extensible Markup Language (XML) Ligeledes et standardiseret format til datagruppering. Kategoriseres som et "markup language".

Document Object Model (DOM) Den struktur der tillader JavaScript at interagere med alle elementer i klientsiden af en web applikation som eksempelvis HTML og CSS.

1.2.2 RPC

JavaScript er enkelt-trådet. Dermed står vi som udviklere med et problem hvis vi vil både vil tilbyde at lave forespørgsler til vores server og samtidig vise de succesfuldt transmitterede resultater for klienten, da en enkelt tråd blot eksekveres sekventielt og derfor ville fryse i processen der henter data til og fra serveren. Dette løses netop med AJAX.

En af de måder AJAX bliver implementeret på i GWT er Remote Procedure Call (RPC), i hvert fald hvis man vælger at skrive backend systemet i Java som vi har valgt at gøre i dette system. Beslutningen om at skrive i Java er både betragtet ud fra det førnævnte om fokus på bestemt konstruktioner samt netop RPC. Asynkrone funktioner er en af de essentielle dele af AJAX, og implementeres derfor også i RPC, da det er afledt af AJAX. [3]



Figur 1.3: RPC Hierarki

1.3 Server kommunikation

Vægten der benyttes i dette system har en Telnet server kørende der lytter for indadgående TCP traffik på port 23. Telnet er en protokol beliggende øverst (applikationslaget) i Open Systems Interconnection Reference Model (OSI) modellen. Den er programmeret til at svare på en tabel af genkendte kommandoer, kodet ud fra American Standard Code for Information Interchange (ASCII) tabellen.

Bemærk hvordan en pakke analyseret med Wireshark illustrerer nedbrydningen af TCP protokol strukturen, og hvordan payload delen nemt kan aflæses i overensstemmelse med ASCII tabellen.[4] Sendes en ukendt kommando fås intet svar. Ved analyse af vægtens dokumentation fra producenten konstruerede vi en simulator for at bedre kunne vise hvordan en sådan server tager form. I de følgende uddrag af pakker ses TELNET protokollens rå data, hvilket kan virke modstridende i forhold til den viden om at RPC fungerer ved HTTPS der derfor kunne antyde at payload ikke burde kunne aflæses forklares dog ved at trafikken i dette tilfælde blev udført i et unit test scenarie hvor pakkerne blev transmitteret direkte til den lokale maskine, og at SSL derfor ikke er blevet implementeret. Dette understøttes af de to Kilde og Destination's IP filtre: `127.0.0.1`. Se både figur 3.5 & 3.6 i resultatbehandlings afsnittet.

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Figur 1.4: ASCII Tabel

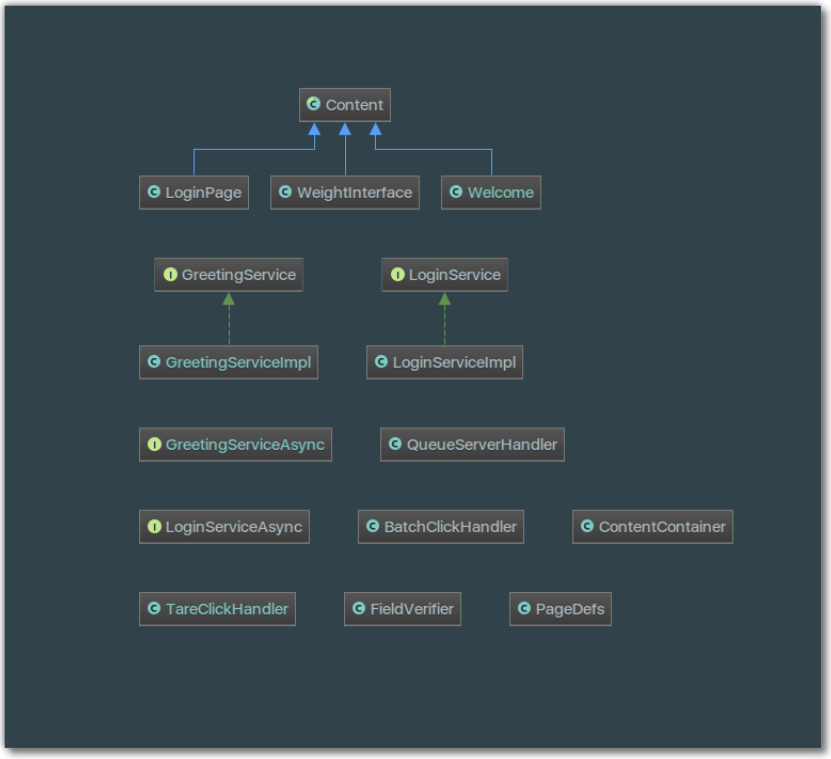
Med værktøjet Wireshark kan direkte aflæses længden af de forskellige pakker angivet i bytes samt den tid der tager at transmittere og dermed også tiden det tager serveren at svare klienten. Denne forskel i tid (*latency*) øges betydeligt ved introduktion af overhead som forsinkelse, der eksempelvis følger med at udvidde fra lokal transmittering til et større netværk.

Kapitel 2

Implementering

2.1 Systemet

Implementeringen introduceres naturligt i forskellige dele, herunder Server & Klient. For at danne et hurtigt overblik observeres et Unified Modeling Language (UML) diagram over hele kodebasen og dets samspil fra et abstrakt standpunkt. Flere diagrammer findes i appendix.



Figur 2.1: UML diagram over hele kodebasen.

2.1.1 Frontend

Frontend opdeles i design og interaktiv JavaScript kode.

2.1.1.1 Layout / CSS/HTML

For at skabe bedre overblik og robust struktur, anvendes det samspil der tilbydes mellem Cascading Style Sheets (CSS) og HyperText Markup Language (HTML). For at give HTML elementer et CSS klassenavn og på den måde give elementet sine design egenskaber gennem GWT's interaktive HTML teknikker, anvendes blot metoden `addStyleName()` der tager en streng for at navngive, og som tilbydes af alle GWT `widgets`. På samme måde kan der tildeles CSS styles direkte til `widgets` med metoder som `setHeight()` og lignende. Da designet søgte at kunne skaleres til en hvilken som helst enhed, var det nødvendigt at bruge procent enheder eller em der er relativt baserede på enhedens egne indstillinger.

Endelig tillader diverse browsere at benytte et debug værktøj navngivet "device mode", der på den måde simulerer forskellige størrelser.

2.1.1.2 Klient / Javascript

Der er med henblik på at udnytte principper fra objektorienteret programmering anvendt en orden der havde indledningsvis stor kompleksitet i implementering, men senere tillod mere 'ren' kode samt bedre mulighed for at udvidde systemet, eksempelvis ved et loginsystem. Det blev i praksis udført ved at udnytte GWT's evne til dynamisk at ændre HTML markup strukturer. Dette er dog kun muligt hvis HTML elementerne bliver skabt gennem den Java kode der siden hen compiles til JavaScript. Det er eksempelvis alle `widget`'s, der kan importeres ud fra `import com.google.gwt.user.client.ui.*;` som eksempelvis Buttons, Panels og lignende.

Derfra er det relevant at vide hvordan RootPanel virker. Hvis man observerer et "hello world" markup i HTML:

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
   <HEAD>
      <TITLE>
         A Small Hello
      </TITLE>
   </HEAD>
<BODY>
   <H1>Hi</H1>
   <P>This is very minimal "hello world" HTML document.</P>
</BODY>
</HTML>
```

Kan det antydes ud fra indenteringen hvordan det i store træk er opdelt i `<HEAD>` og `<BODY>`. `<HEAD>` består mest af alt af metadata, mens `<BODY>` huser indholdet. RootPanel objektet fra GWT kan tilgå `<BODY>` markup ved at kalde dens `RootPanel.get()` metode. Herfra kan ydermere specificeres hvilke opdelinger inde for `<BODY>` der søges, mens et simpelt `RootPanel.get()` kald tilgår roden af `<BODY>`. Således tillader GWT dermed at, interaktivt skifte indholdet af HTML og andet indhold defineret i koden ud fra den logik der oversættes til JavaScript. Dette hjælper med at forstå motivationen for en mere kompleks struktur med baggrund i GWT's dynamisk skiftende JavaScript HTML.

Figur 2.2: UML diagram viser sammenhængen mellem Content & ContentContainer samt de klasser der nedarver dette.

I stedet for at have hele koden i en samlet klasse med entry point, valgte vi i stedet følgende:

Landings siden er den side man mødes med når man tilgår base Uniform Resource Locator (URL)'en. Denne har blot til formål at redirrigere til den næste, og kunne udelades, men tilbød et vigtigt værktøj til at afgøre hvilken side der skulle dirrigeres til, og på den måde kunne springe over login siden og gå direkte til hovedsiden hvis dette var ønsket til test eller lignende.

Login siden har til formål at implementere et robust login system der både anvender den kryptering indbygget i RPC, men også kommunikerer med Server siden gennem `LoginService` objektet og på den måde understøtter alle former for database styring der kan implementeres i Java,

herunder MySQL.

Weight Interface som er hovedsiden der tilbyder adgang til vægten.

2.1.1.3 Content & Abstraction

2.1.1.3.1 Content er en abstrakt klasse og er således blot en idé der skal implementeres af andre klasser. Tankesættet er at hver klasse, kategoriseret som `pages` pakker, specifikt de 3 førnævnte, alle nedarver de to metoder defineret i Content som følger herunder:

Listing 2.1: Content klassen

```
1  package com.weightinterface.myproject.client;
2
3  import com.google.gwt.user.client.ui.Composite;
4
5  /**
6   * Created by haanin on 5/2/16.
7   */
8  public abstract class Content extends Composite {
9      public abstract String getToken();
10     public abstract String getWindowTitle();
11 }
```

Bemærk især de to metoder, der som naturlig følge af at `pages` klasserne alle nedarver `Content`, begge er implementeret i samtlige `pages`.

2.1.1.3.2 Content Container er en klasse der indeholder logikken til at håndtere klasser nedarvet fra `Content`, altså `pages`. Det er denne klasse du kalder hvis du vil skifte fra login side til hovedklasse. Dette sker i `ContentContainers` klasse `setContent()`:

```
1      private void setContent(Content content) {
2          RootPanel contentRoot = RootPanel.get("content");
3          contentRoot.clear();
4          this.content = content;
5          History.newItem(content.getToken(), false);
6          // check for special initializations:
7          if (content.getToken().equals(PageDefs.AppToken)) {
8              ((WeightInterface) content).MyInit();
9          }
10         if (content.getToken().equals(PageDefs.LoginToken)) {
11             ((LoginPage) content).MyInit();
12         }
13         contentRoot.add(content);
14         Window.setTitle(content.getWindowTitle());
15         Window.scrollTo(0, 0);
16     }
```

Metoden tager en af de `pages` der netop er nedarvet af `Content`, og derfor alle har de to funktioner `getToken()` samt `getWindowTitle()` implementeret.

Først ryddes RootPanel, altså HTML markup'et. Dernæst kommer de implementerede metoder i brug til at identificere hvilken `page` der specifikt er tale om. Dette sker i logik tjekket `if (content.getToken().equals(PageDefs.*`, hvor PageDefs er en offentlig klasse der kan tilgås af alle, og blot fungerer som en database for de forskellige unikke tokens der definerer hvilken

side der er tale om. Se hvordan dette benyttes af hovedsiden, `WeightInterface`:

```
    @Override
    public String getToken() {
        return PageDefs.AppToken;
    }

    @Override
    public String getWindowTitle() {
        return PageDefs.AppTitle;
    }
```

Mens der i `PageDefs` klassen kan aflæses de streng-værdier der er repræsenteret i variabelnavnene

```
    public static final String AppToken = "WI";
    public static final String AppTitle = "WeightInterace";
```

2.1.2 Backend

2.1.2.1 GreetingService

Serversiden reagerer på forespørgsler fra vægtloopet. Først oprettes en socket på port 8000 til Vægtens router's IP addresse.

```
    private void serverInit() throws IOException {
        //pingSocket = new Socket("192.168.1.214", 23);
        //pingSocket = new Socket("127.0.0.1",23);
        pingSocket = new Socket("62.79.16.16",8000);
        out = new PrintWriter(pingSocket.getOutputStream(), true);
        in = new BufferedReader(new InputStreamReader(pingSocket.getInputStream()));
    }
```

Ud over forbindelse til Socket'en implementeres logikken for den metode der kaldes gennem RPC for at modtage genkendte vægt-kommandoer:

```
public String sendCommandToWeight(String command) throws IOException {
        String result = null;
        out.println(command);
        result = in.readLine();
        return result;    }
```

2.1.2.2 LoginService

Login skulle implementeres med SQL, men da vi valgte at implementere løsningen på en Raspberry Pi med begrænset lagerplads samt grundet den øgede kompleksitet ved dette valg, nåede vi kun at rulle en midlertidig løsning ud med en lokal tekstfil. Vi udnytter til gengæld en datastruktur, nemlig HashMap, der vil levere markant hurtigere performance ved lookup af information.

```
public class LoginServiceImpl extends RemoteServiceServlet implements LoginService {

    BufferedReader br;
    private HashMap<String,Boolean> usernames = new HashMap<>();

    public LoginServiceImpl() throws FileNotFoundException {

    }
```

Med HashMap implementationen benyttes `.get(key)` mens et Boolean objekt gemmes som `true` hvis brugernavnet er at finde i databasen. Hvis ikke returneres null, der håndteres af metoden og dette returneres som et `false` til JavaScript koden. Dermed er en RPC implementeret i Server-siden.

```
    public Boolean checkLoggedIn(String username) throws IOException {
        URL path = LoginServiceImpl.class.getResource("usernames.txt");
        File f = new File(path.getFile());
        br = new BufferedReader(new FileReader(f));
        usernames.clear();
        String line = null;
        try {
            line = br.readLine();
        } catch (IOException e) {
            e.printStackTrace();
        }

        while(line != null)
        {
            usernames.put(line,Boolean.TRUE);
            try {
                line = br.readLine();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (usernames.get(username).booleanValue()) return Boolean.TRUE;
        else return Boolean.FALSE;
    }
```

2.1.3 Vægtloop

Det loop der sender signalet med forespørgsler til Java serveren ligger som sin egen klasse for at isolere den logik der foregår. Der er dog indbygget en måde for denne klasse at lave metodekald i hovedsiden på uden at der skal være offentlige metoder men samtidig uden at skulle lave et helt nyt instans af klassen. Dette er udført ved at lave et null objekt af `WeightInterface`, `wi`, der videre defineres i constructoren.

```
public class QueueServerHandler implements ClickHandler, KeyUpHandler {


    WeightInterface wi = null;
    boolean isNegative = false;
    private boolean alreadyClicked = false;

    public QueueServerHandler(WeightInterface wi)
    {
    this.wi = wi;
        wi.wb.setVisible(false);
        doInit();
    }

    public void doInit() {
        try {
            weightPrompt();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Dette tillader således at man kan stille `WeightInterface` og de indrettede metoder til rådighed uden at lave en ny kopi af sig selv ved at oprette `QueueServerHandler` klassen som følger:

```
final QueueServerHandler handler = new QueueServerHandler(this);
```

Ydermere fremgår det i det første udsnit at klassen `QueueServerHandler`, ved oprettelse, straks går i gang med at kalde metoden `doInit()`, der siden hen kalder på dens lokale metode `weightPrompt()`. Det er i denne funktion selve loopet foregår. Heri oprettes forbindelse til serveren med et kald, der siden bringer to mulige scenarier, udtryk som `onFailure()` og `onSuccess()`:

```
private void weightPrompt() throws IOException {
            wi.getGreetingService().sendCommandToWeight("SI", new AsyncCallback<String>()
```

`onFailure()` giver blot en overfladisk fejlhåndtering, mens `onSuccess()` regner de rigtige værdier ud (til afbenyttelse af styktælling blandt andet), og siden hen genstarter loopet. På den måde sikres også at vægten ikke får flere forespørgsler end den kan håndtere, mens der dog stadig presses til det yderste.

```
                public void onSuccess(String weight) {
                    //DOM.getElementById("sk-circle").getStyle().setDisplay(Style.Display.
                    isNegative = false;
                    wi.wb.setText("OFF");
                    char c = weight.charAt(7);
                    if (!(c == ' ')) isNegative = true;
                    wi.res = Double.parseDouble(weight.substring(8,14));
                    if(wi.batch.isReady()){
                        double batchdouble = wi.batch.calculateAmount();
                        wi.count.setText(""+(int) batchdouble);};
                    if(isNegative)
                    {
                        wi.currentWeightDisplay.setText("-"+wi.res+" kg");
                        DOM.getElementById("myBar").getStyle().setWidth(0, Style.Unit.PCT)
                        DOM.getElementById("label").setInnerText("0%");
                    } else
                    {
                        wi.currentWeightDisplay.setText(wi.res + " kg");
                        DOM.getElementById("myBar").getStyle().setWidth(wi.res * 16.66, St
                        DOM.getElementById("label").setInnerText(Double.toString(wi.res *
                    }
                    try {
                        weightPrompt();
                        ...
```

2.2 RPC

RPC implementeres i praksis ved at nedarve RemoteService klassen, stillet til rådighed af GWT Software Development Kit (SDK)'et. Både klient og server kode skal nedarve en form af RemoteService, hvor server nedarver RemoteServiceServlet, den del der eksekveres i Java Virtual Machine (JVM)'en og oversættes til bytecode, mens JavaScript delen compiles gennem GWT "RemoteService". Jævnfør figur 1.3.

Listing 2.2: Remote Service udsnit

```
import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
import java.io.IOException;
/**
 * The client-side stub for the RPC service.
 */
@RemoteServiceRelativePath("greet")
public interface GreetingService extends RemoteService {
  //String greetServer(String name) throws IllegalArgumentException;
  String sendCommandToWeight(String command) throws IOException;
  String closeConnection() throws IOException;
  String openConnection() throws IOException;
}
```

I udsnittet foroven ses RPC implementeret for vægt kommunikationen. Hver gang der skal implementeres et callback skal RemoteService implementeres således, og i objektorienteret programmerings ånd har vi opdelt den såkaldte `GreetingService` fra `LoginService` der henholdsvis håndterer klient-server-kommunikation omhandlende Vægtkommunikation og Loginsystem/Database

kommunikation. Følges denne struktur skal hver callback tjeneste implementeres for sig, hvilket også kan aflæses i koden.

Listing 2.3: Remote Service Servlet udsnit

```
1  import com.weightinterface.myproject.client.services.GreetingService;
2  import com.google.gwt.user.server.rpc.RemoteServiceServlet;
3
4  import javax.servlet.ServletException;
5  import java.io.BufferedReader;
6  import java.io.IOException;
7  import java.io.InputStreamReader;
8  import java.io.PrintWriter;
9  import java.net.Socket;
10
11 /**
12  * The server-side implementation of the RPC service.
13  */
14 @SuppressWarnings("serial")
15 public class GreetingServiceImpl extends RemoteServiceServlet implements
16     GreetingService {
17
18     private Socket pingSocket;
19     private PrintWriter out;
20     private BufferedReader in;
21
22     public GreetingServiceImpl() throws ServletException {
23
24     try {
25         serverInit();
26
27     } catch (IOException e) {
28         // TODO: Set text to error message of failed construction.
29         //return;
30     }
31     }
```

Slutteligt er der også en Asynkron modparts klasse der skal opbygges for fuldt at understøtte callback fra GWT's RPC. Denne implementeres således med de metoder der skal benyttes i callbacket på samme facon som de andre callbacks.

Listing 2.4: Asynkron interface modport af GreetingService

```
package com.weightinterface.myproject.client.services;

import com.google.gwt.user.client.rpc.AsyncCallback;

import java.io.IOException;

/**
 * The async counterpart of <code>GreetingService</code>.
 */
public interface GreetingServiceAsync {

  //String sendCommandToWeight(String s) throws IOException;
  void sendCommandToWeight(String command, AsyncCallback<String> callback)
      throws IOException;

  void closeConnection(AsyncCallback<String> asyncCallback) throws IOException;
  void openConnection(AsyncCallback<String> asyncCallback) throws IOException;

}
```

2.3 Vægtsimulator

Vægtsimulatoren kodes også i Java for at holde en konsistent stil, men kunne sagtens programmeres i andre sprog. Det vigtigste er at bibliotek standarder som Sockets og Threads er tilgængelige eller bliver implementeret rigtigt. Her benyttes Javas indbyggede biblioteker til at tilbyde tjenesterne. Tråde er vigtige for der kan kommunikeres asynkront. Nærmere bestemt anvendes den ene tråd i programmet til at lytte aktivt efter input fra en klient socket med det formål at tømme og fylde bufferen.

```
public class WeightSimulator implements Runnable
{
    ServerSocket serverSocket = null;
    Socket clientSocket = null;
    Thread listener = null;
    }
```

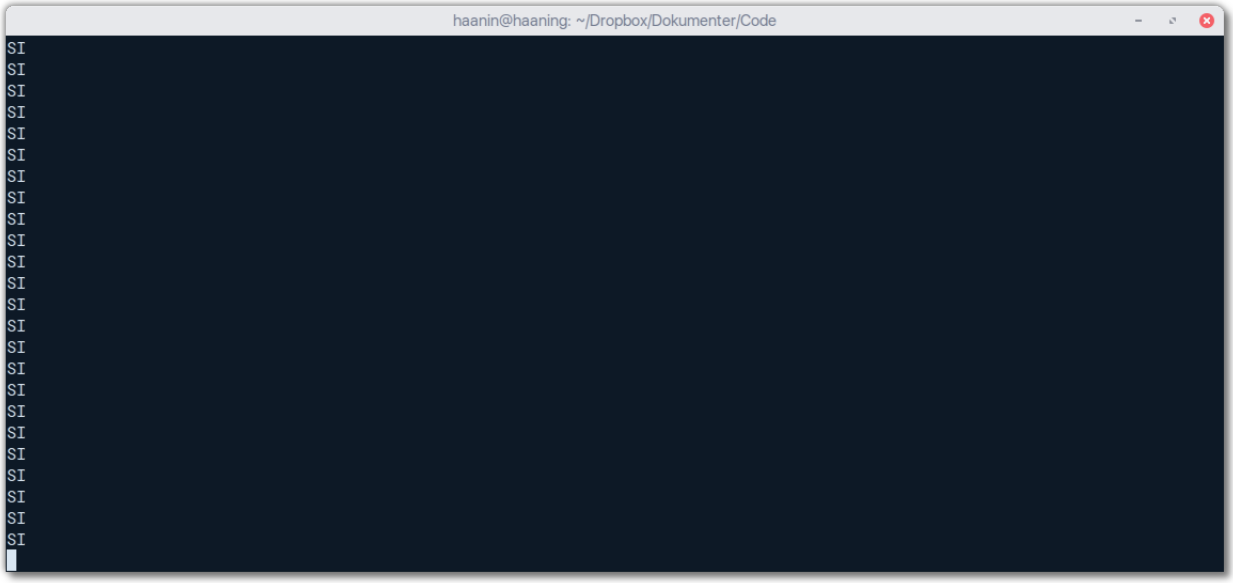
I det følgende kodeudsnit er forsøgt udtrukket det mest essentielle logik i simulatoren. run() metoden implementeres i et loop der er baseret på et flag der kun bryder while condition tjekket hvis der opstår fejl. Dette håndteres senere i koden. BufferedReader objektet indsamler data fra klientens Socket, og så længe der er data i Bufferen tilgås den logik der skal vurdere hvilken kommando er der sendt og på den måde afgøre hvilket svar der skal sendes.

```
public void run()

    ...

    PrintWriter out2 = new PrintWriter(getOutputStream(),true);


                        Random random = new Random();
                        BufferedReader in =
                                new BufferedReader(
                                        new InputStreamReader(clientSocket.getInputStream(
                        String line;
                        while((line = in.readLine()) != null)
                        {
                            System.out.println(line);
                            if(line.contains("SI"))
                            {
                                out2.println("Weight: "+getRandomValue(random,0,6,4)+" kg"
                            }
                            else if(line.contains("TI"))
                            {
                                out2.println("Tare");
                            }
                        }
```

Her benyttes Random objektet fra Javas bibliotek til at generere et tilfældigt tal mellem 0 og 6 med 4 decimaler, der på den måde leverer et resultat der er i overensstemmelse med den længde kodede streng der sendes fra den rigtige vægt, og på den måde vil være kompatibel med vores system.

```
public static String getRandomValue(final Random random,
    final int lowerBound,
    final int upperBound,
    final int decimalPlaces){

    if(lowerBound < 0 || upperBound <= lowerBound || decimalPlaces < 0){
        throw new IllegalArgumentException("Put error message here");
    }

    final double dbl =
        ((random == null ? new Random() : random).nextDouble() //
            * (upperBound - lowerBound))
            + lowerBound;
    return String.format("%." + decimalPlaces + "f", dbl);

}
```

Åbnes en terminal fra serverens side, kunne en typisk session derfor bekræftes visuelt således som figur 2.3:



Figur 2.3: Terminal fra serversiden af vægt simulatoren.

Kapitel 3

Test & Resultater

3.1 Systemet

3.1.1 Frontend

3.1.1.1 Layout

I de følgende billeder bekræftes at designet, der var ment til at skalere med forskellige opløsninger og enheder, var effektivt og korrekt implementeret. Se figurene 3.1, 3.2 & 3.3

3.1.1.2 Klient

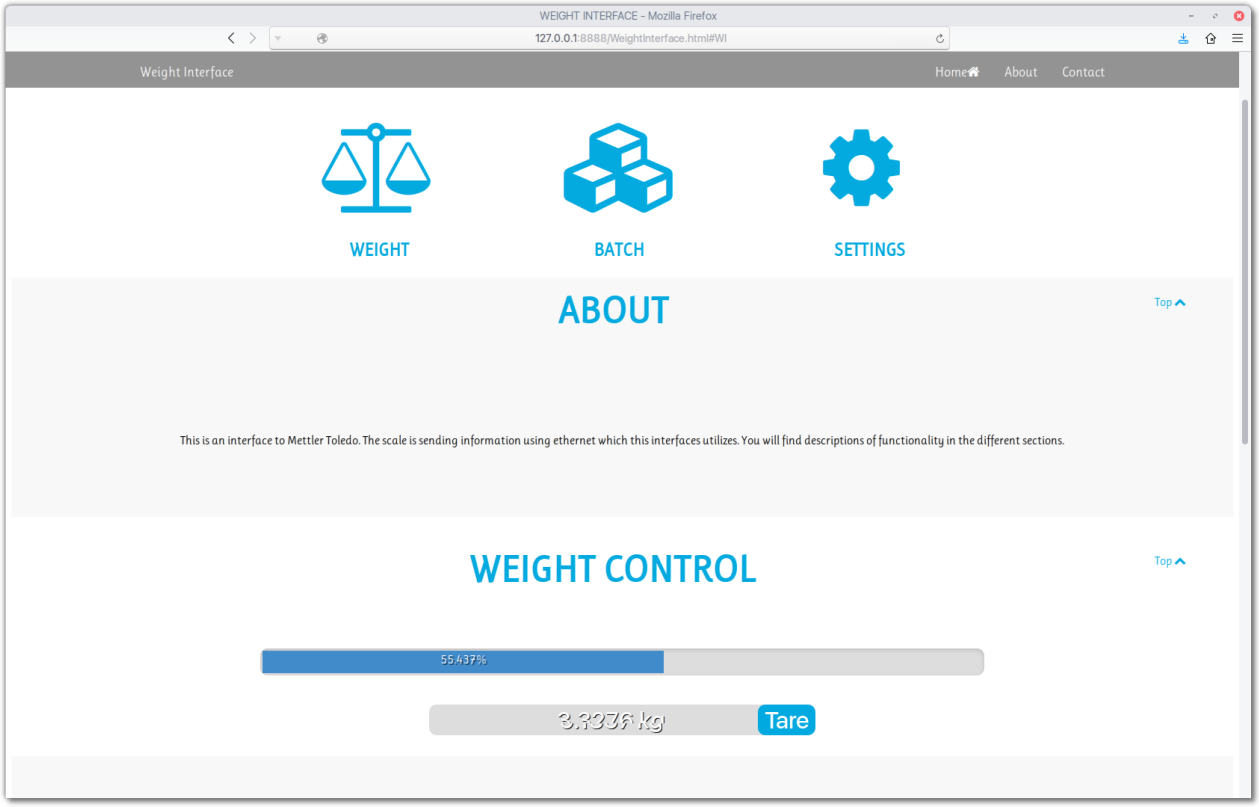
Både simulator og vægt resultater håndteres ens af det interaktive design. På grund af at projektet er struktureret således at det ikke er brugeren der sætter gang i kommunikationen, men i stedet starten af serverens køretid, muliggør dette at forskellige brugere kan observere vægtresultater og andet samtidig. Alle forbundne brugere kan samtidig manøvrere hele systemet samtidig, uafhængigt af hinanden.

Indbygget i loopet der forespørger vægten om data er en mekanisme der tjekker for tarerings forespørgsler, således at det ikke afbryder flowet men samtidig iværksættes øjeblikkeligt. Styktælling virker ved at `BatchClickHandler` henter data fra hovedklassen `WeightInterface` og kan håndtere både negativ vægt og antal op til vægtens maksimum. Se figur 3.4. Det er ikke lykkedes os at finde nogle fejl i disse.

3.1.2 Backend

Som tidligere beskrevet undgås 'oversvømning' af serveren ved hjælp af et kontrollerert loop der først sender en ny forespørgsel når den gamle er besvaret. Dette betyder såvel også, at vægten kun svarer på de forespørgsler den kan håndtere. Dette hjælper programmet med at skalere videre til forskellige vægte eller enheder med forskellig kapacitet, og kom tydeligt til udtryk da interfacet blev brugt på henholdsvis vægten over netværk og simulator (på en større CPU) på et lokalt netværk. Antallet af forespørgsler kan måles eksempelvis på de tider der observeres i Wireshark, hvor den forskel der er et måle er i størrelsesordenen 0.0000x sekunder, altså xx mikrosekunder, eksempelvis udledt af at trække tiderne fra hinanden og danne et gennemsnit som følger

$$3.236876407 - 3.236811257 = 65.15\mu s$$

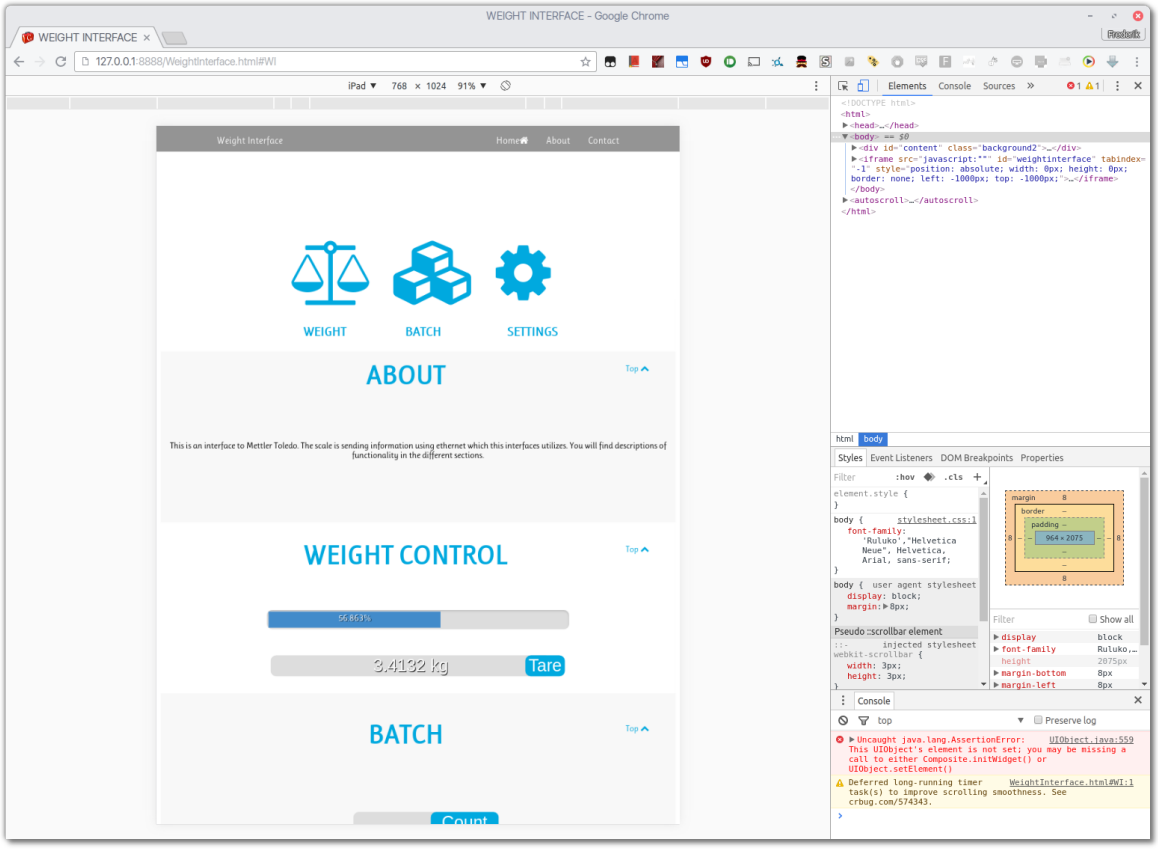


Figur 3.1: Layout på en computer

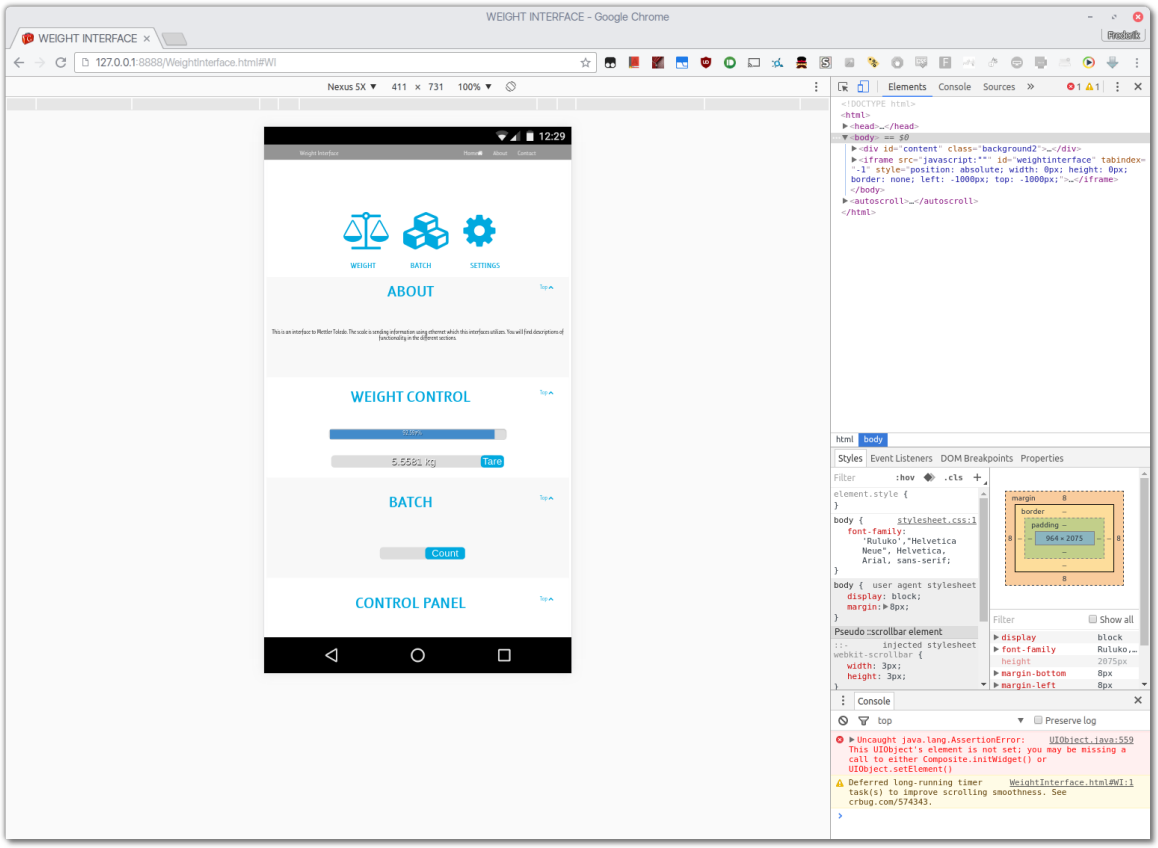
Altså milliontedele af et sekund. Til sammenligning kunne vi måle omtrent $\frac{1}{7}$ sekunder forskel i transmission over optimale forhold ved vægten alene.

3.2 Kommunikation

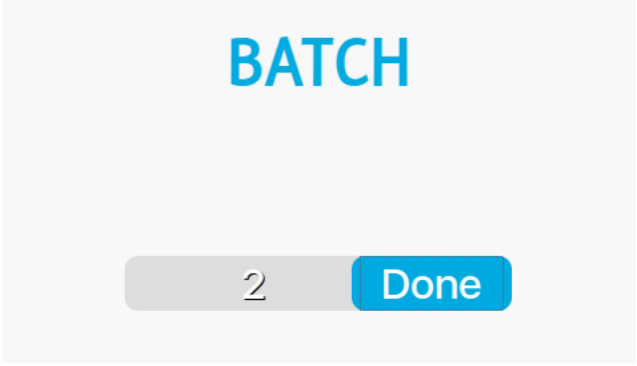
Her bekræftes mekanismen bag den kommunikation der foregår mellem Server og Vægt. Da der ikke i Telnet er indbygget noget sikkerhedslag er pakkerne kun udgjort af TCP headers samt plaintext kodet i ASCII format. På trods af at kommunikationen mellem frontend og backup i server opstillingen udnytter Secure Sockets Layer (SSL) over HTTP er der altså ingen sikkerhed mellem vægt og Java server. Man kunne have forsøgt at implementere Secure Shell (SSH) eller lignende teknologier med dette indbygget, eller udbygget laget mellem vægt og router således at routeren havde en krypteret forbindelse til Java serveren.



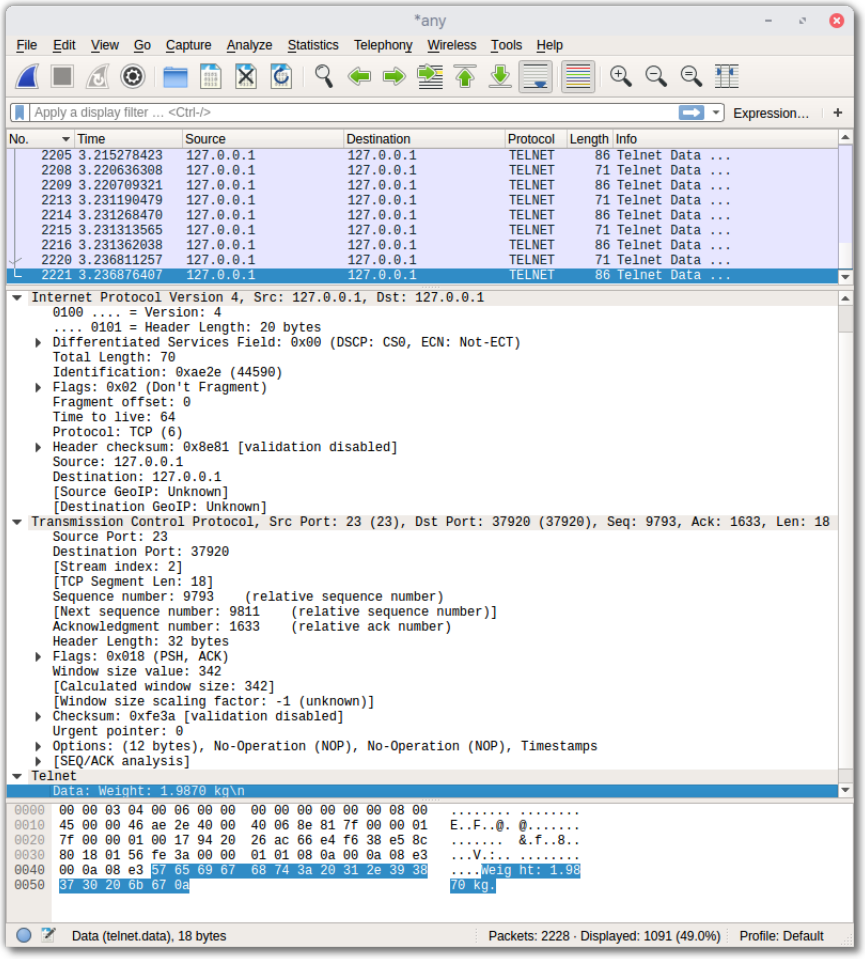
Figur 3.2: Layout på en iPad tablet



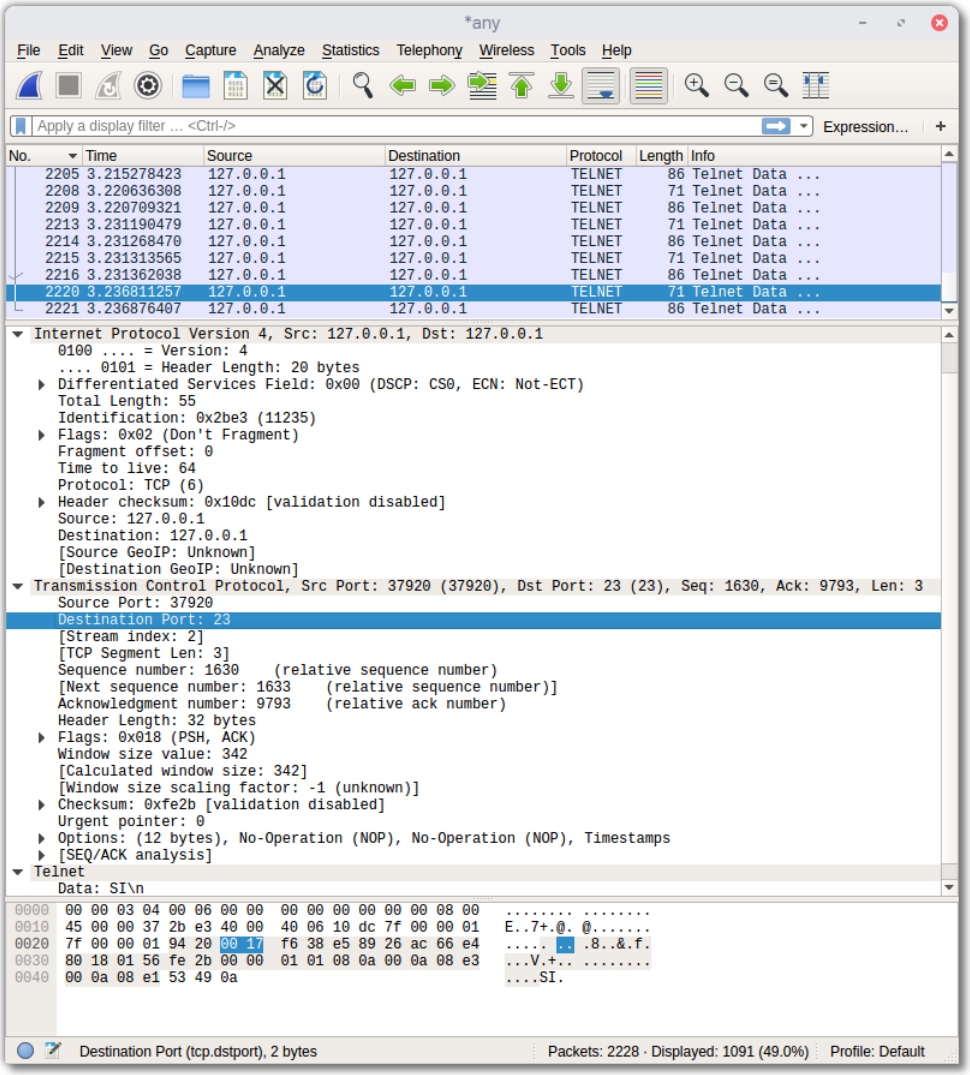
Figur 3.3: Layout på en Nexus 5X smartphone



Figur 3.4: Styktæller i arbejde



Figur 3.5: Wireshark analyse af pakker mellem server og vægt (Simulator til Server)



Figur 3.6: Wireshark analyse af pakker mellem server og vægt (Server til Simulator)

Kapitel 4

Diskussion

4.1 GWT

GWT har vist sig at være et komplekst system til at skrive og integrere web applikationer på en sådan måde at der bringes objekt-orienteret kodebase struktur til JavaScript, der ellers er et mere dynamisk skrevet sprog. Dette gør det nemmere at holde styr på større projekter, hvilket en web applikation af danne dimension vil karakteriseres. GWT tilbyder også at denne cross-compilation bliver effektiv og performance-minded, på en måde der ofte påkræver stor tilknytning til sproget. Koden bliver desuden håndtérbar for alle typer browsere uden at skulle tage særlige hensyn. Til gengæld tager koden længe om at compile.

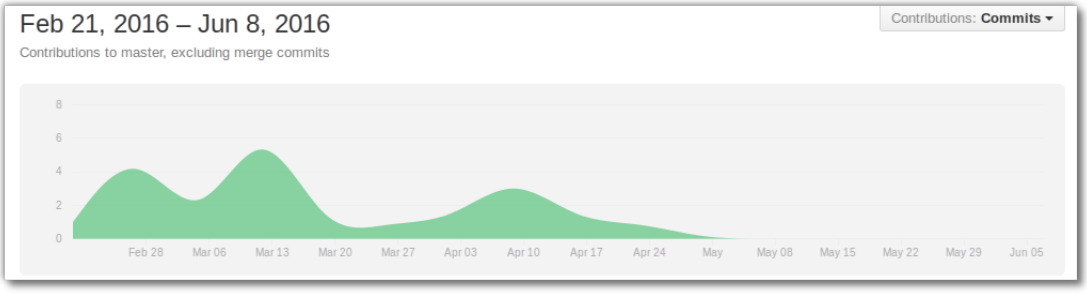
4.2 Systemet

4.2.0.0.1 VCS Til versionskontrol benyttede vi os flittigt af git, og hostede koden på GitHub eksternt i et privat repo. Vi har begge benyttet os af bash git kommandolinien for at udføre opgaverne. Vores .gitignore har sørget for hurtigere overførselstider samt mindre overhead ved blot at overføre kildekodefiler (ikke .class), spare de genererede .js filer samt generelle dupliketter:

```
*.class

# Package Files #
*.jar
*.war

# gwt caches and compiled units #
war/gwt_bree/
gwt-unitCache/

# boilerplate generated classes #
.apt_generated/

# more caches and things from deploy #
war/WEB-INF/deploy/
war/WEB-INF/classes/

#compilation logs
.gwt/

#caching for already compiled files
gwt-unitCache/

#gwt junit compilation files
www-test/

#old GWT (1.5) created this dir
.gwt-tmp/
```

Vi benyttede os af branching til at holde styr på forskellige opgavestillinger, og mergede sidenhen ind i en midlertidig branch der havde til formål at udvikle, for siden hen at merge ned på master branchen til det færdige produkt `"1.0"`. Se i øvrigt commit graf ved figur 4.1

4.2.0.0.2 Agil udvikling For at udnytte teknikkerne i software engineering valgte vi modellen agil udvikling. Denne har vist sig effektiv til større projekter hvori mindre implementeringer stykkes sammen, en efter en, på en måde hvorved der er testet og fejlbehandlet, i stor kontrast til vandfaldsmodellen der har til formål at bygge alle faser en af gangen for hele projektet. Dette kom os især til gode når vi fik idéer til anvendelige funktioner der kunne bygges ovenpå det allerede eksisterende projekt. For at holde styr på dette anvendte vi noget der tog afsæt i Scrum.



Figur 4.1: GitHub commit frekvens graf

4.2.0.0.3 Scrum På trods af at der ikke blev oprettet en Jira server, anvendte vi den model der minder allermest om Scrum, afledt af agil softwareudvikling. Vi havde bug trackers i et dokument på git der løbende blev opdateret med både backlog og sprints.

Kapitel 5

Konklusion

GWT er som teknologi under udvikling et stabilt værktøj til udvikling af robuste web applikationer der er browser-hensyns-uafhængige, sikre og ydeevne kraftige. Med GWT som værktøj effektiviseres processen ved at tilbyde de strukturer der normalt er reserveret objektorienterede programmeringssprog, og bringer Java programmører udenom JavaScript, jQuery, php og andre sprog der gør sig gældende når samme betingelser gør sig gældende omkring målet for en web applikation.

GWT tilbyder også muligheden for at nøjes med at implementere forskellige dele af dets elementer ind i et nyt eller i forvejen eksisterende projekt, eksempelvis ved en web applikation uden AJAX, og dermed uden kontakt til server der kan lave kraftigere beregninger over sikker kommunikation.

Opsætning af et komplet system med udgangspunkt i GWT har givet stor indsigt i den proces der følger med, særligt med henblik på de forskellige elementer såsom eksternt beliggende server, opsætning af virtuelle servere til routere samt diagnostik med tidsforsinkelse og håndteringen af dette.

Kapitel A

Opsætning & Hello World GWT

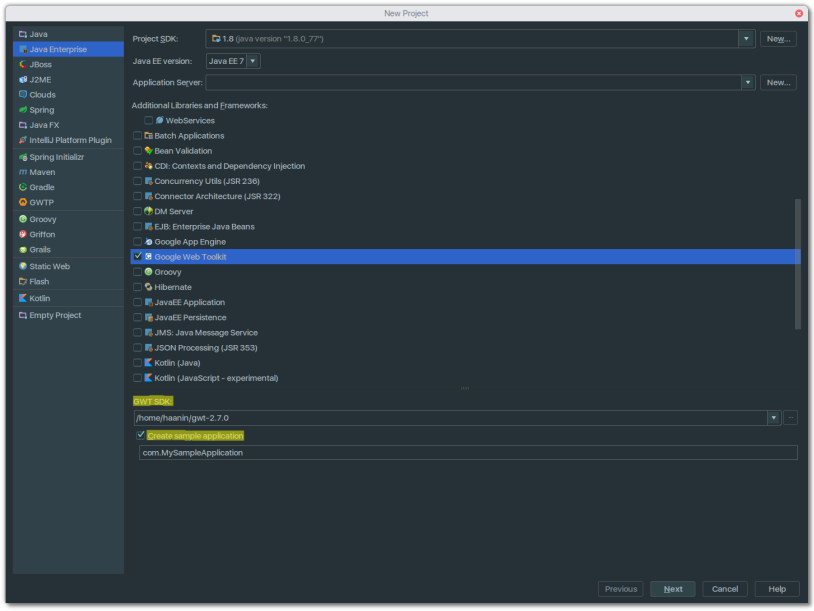
A.1 GWT SDK

I pakken GWT SDK kombineres kernebibliotekerne med cross-compileren. Denne fås i forskellige versioner, i denne opsætning anvendes 2.7.0. Disse værktøjer kan findes på deres download sektion. Bemærk at denne opsætning beskrives udfra et GNU-Linux/UNIX perspektiv, dog er de essentielle dele af opsætningen ikke er tynget af et bestemt operativ system.

A.2 Udviklingsmiljø

Ud over de standalone værktøjer, fås der også plug-ins som en direkte integration i en række udviklingsmiljøer.

A.2.0.0.1 IntelliJ / Eclipse
Da Jens G. Larsen's præferencer var forskellige fra mine, vil denne opsætningsbeskrivelse foregå ved JetBrain's *IntelliJ*. Ønskes en gennemgang af Eclipse opsætning henvises derfor til hans tilsvarende. Denne vil møntet på Windows brugere.

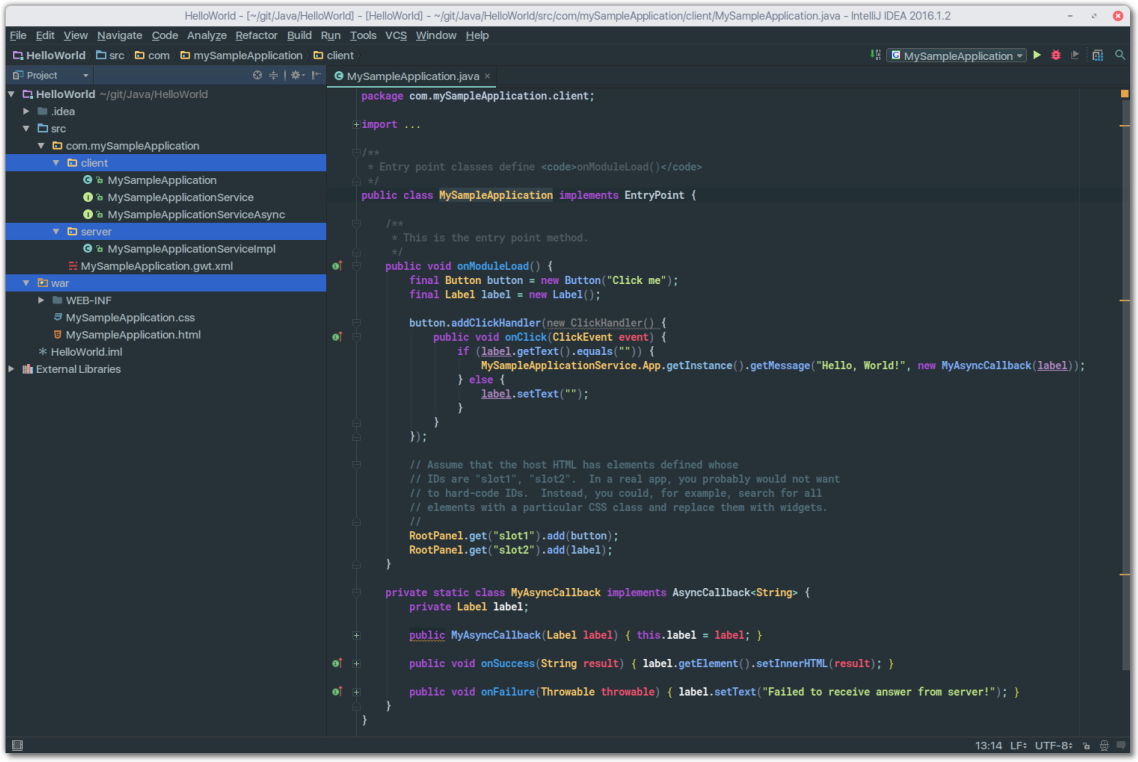


Figur A.1: IntelliJ - GWT Modul, *"Hello World"* applikation.

I alle nyere versioner af IntelliJ IDEA medfølger GWT Support Plugin.[5]
Der er også skabeloner til at komme igang, praktisk til opsætning af RPC, både synkroniseret og asynkront. Der kan både oprettes projekt fra IntelliJ som helt tomt, eller udviddelse af allerede-skrevet kodebase, eksempelvis ved det script der stilles til rådighed i GWT pakken til at skabe projekter. Her beskrives hvordan der

oprettes et helt nyt projekt:

IntelliJ åbnes, og der vælges `New Project`. Dialogboksen, afbilledet her, vises, og de rigtige moduler



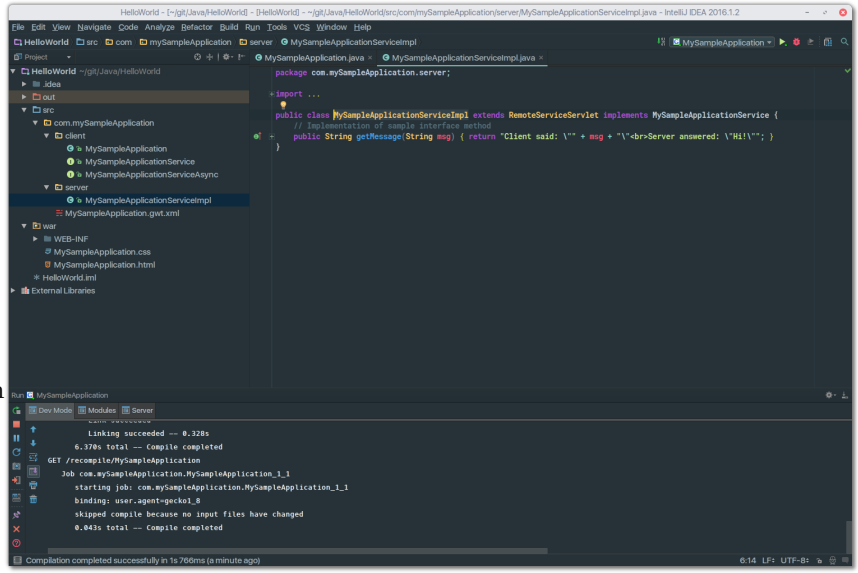
Figur A.2: IntelliJ - GWT Modul, *"Hello World"* applikation.

skal afkrydses, jf. figur A.1 s.28.

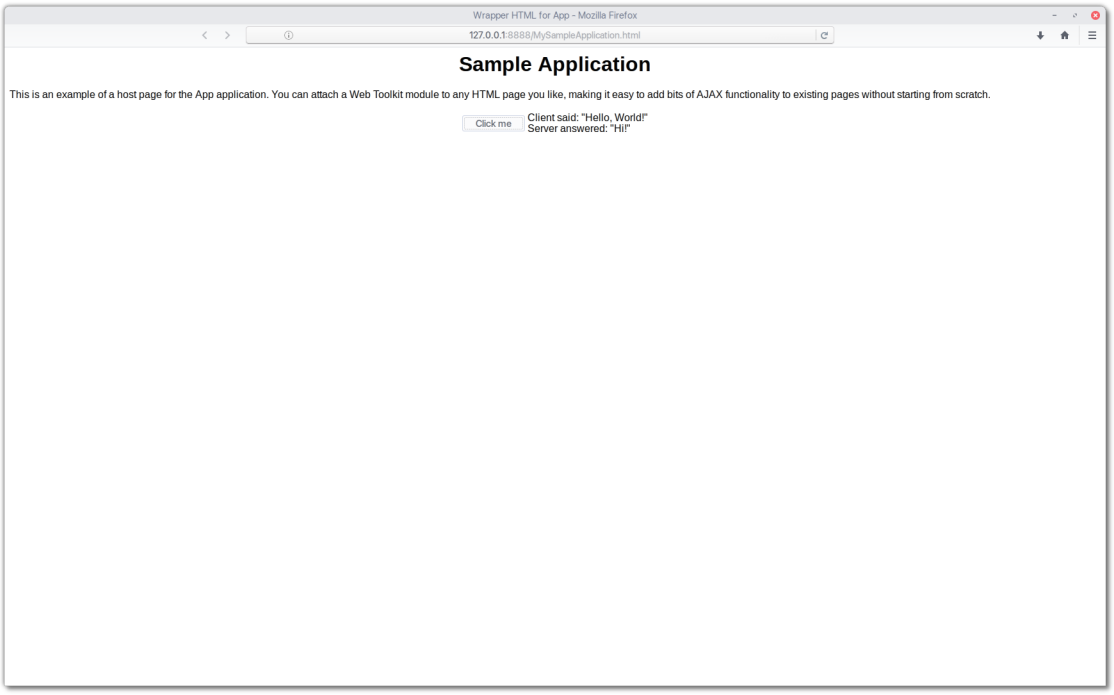
Ved `Next` klik åbnes derved en skabelon der har GWT's Java klasser implementeret i deres mest basale form der stadig tillader at compile og kørsel.

Bemærk hvordan GWT inddeler pakkerne i `client` og `server`. `war` mappen dannes også automatisk med henblik på nem kompression og senere udrulning til Tomcat. Her findes .HTML filer, .css styling og de cachede .js filer der generes i cross-compile fasen fra Java til Javascript (*jf. GWT kapitel*.

Ved compile startes `Super Dev Mode` og starter automatisk browseren. Her vises kommunikationen mellem server og klient udtrykt ved et klik.



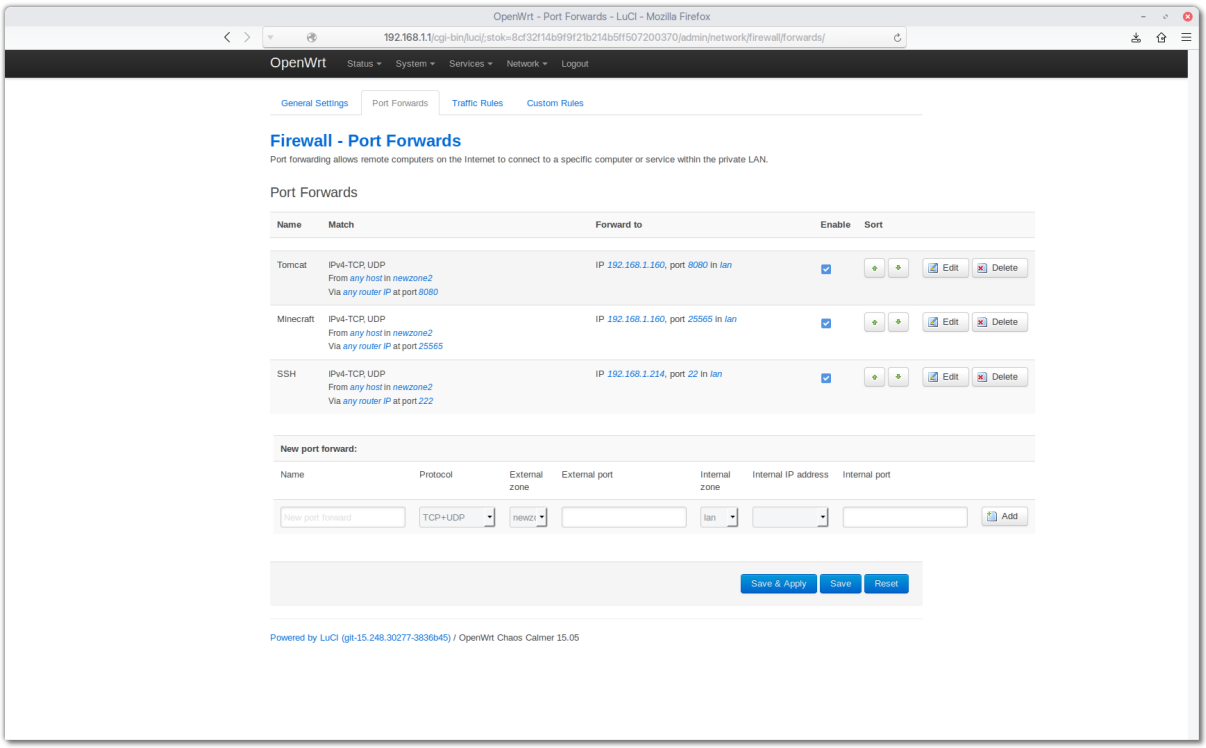
Figur A.3: IntelliJ - GWT Modul, *"Hello World"* applikation.



Figur A.4: IntelliJ - GWT Modul, *"Hello World"* applikation.

A.3 Port forward

A.3.0.0.1 Server Dette afhænger af routeren og dens firmware. Vi benyttede os af en `TP-LINK`, flashet med open source OpenWrt firmware. Denne tillader, på trods af at være linux baseret, en brugergrænseflade. Port forward skulle blot henvise til den IP addresse på det lokale netværk tildelt Raspberry Pi enheden, i dette tilfælde `192.168.1.160`. Vi havde givet enheden et *static lease*, en Dynamic Host Configuration Protocol (DHCP) egenskab der tillader at beholde ip addresser på det lokale netværk statiske.



Figur A.5: OpenWrt Port Forward Interface

A.3.0.0.2 Huawei / Vægt Huawei routeren der i vores setup agerer det led der kommunikerer til og fra vægten skulle også forwarde traffik. Dette er nødvendigt således at server koden, med en socket der kalder på routerens statiske IP `69.72.16.16`, videredirrigerer det til den rigtige port, vægtens telnet protokol port.

A.4 Server

A.4.0.0.1 Raspberry Pi Til dette projekt benyttes Ubuntu Server udgaven (uden grafisk skrivebordsmiljø) i versionen 16.04. Dette kan downloades her.[6] MicroSD kortet skal formateres til filsystem typen `ext4` og derefter skal følgende kommandoer køres fra en anden linux maskine *(instruktioner på download siden findes hvis .img filen ønskes klonet over fra et Windows miljø)*:

```
sudo apt-get install gddrescue
```

```
unxz ubuntu-standard-16.04-server-armhf-raspberry-pi.img.xz
```

```
sudo ddrescue -d -D -force
ubuntu-standard-16.04-server-armhf-raspberry-pi.img /dev/sdx
```

hvor `/dev/sdx` skal tilpasses den enhed, navngivet af SD kortet, typisk `/dev/mmblkp0`. SD Indsættes i Raspberry Pi'en, der benytter den som både systemdisk og datadisk. For at kunne køre Java skal der naturligvis installeres Java Development Kit (JDK), der i denne udgave af GWT anbefales version 8. Dette installeres nemt med følgende kommandoer:[7]

```
sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java8-installer
```

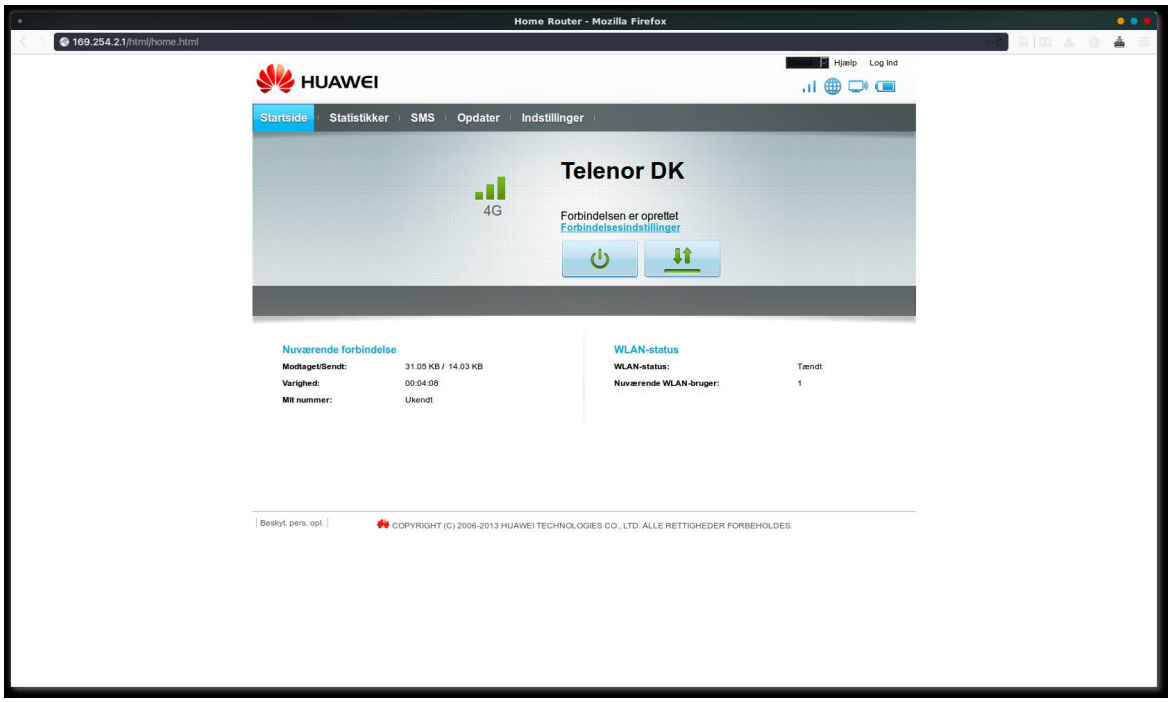
Dette installerer både JDK og Java Runtime Environment (JRE) af 8. version. Hvis man yderligere ikke ønsker at operere serveren med skærm kan SSH også installeres med et enkelt

```
sudo apt-get install openssh-server
```

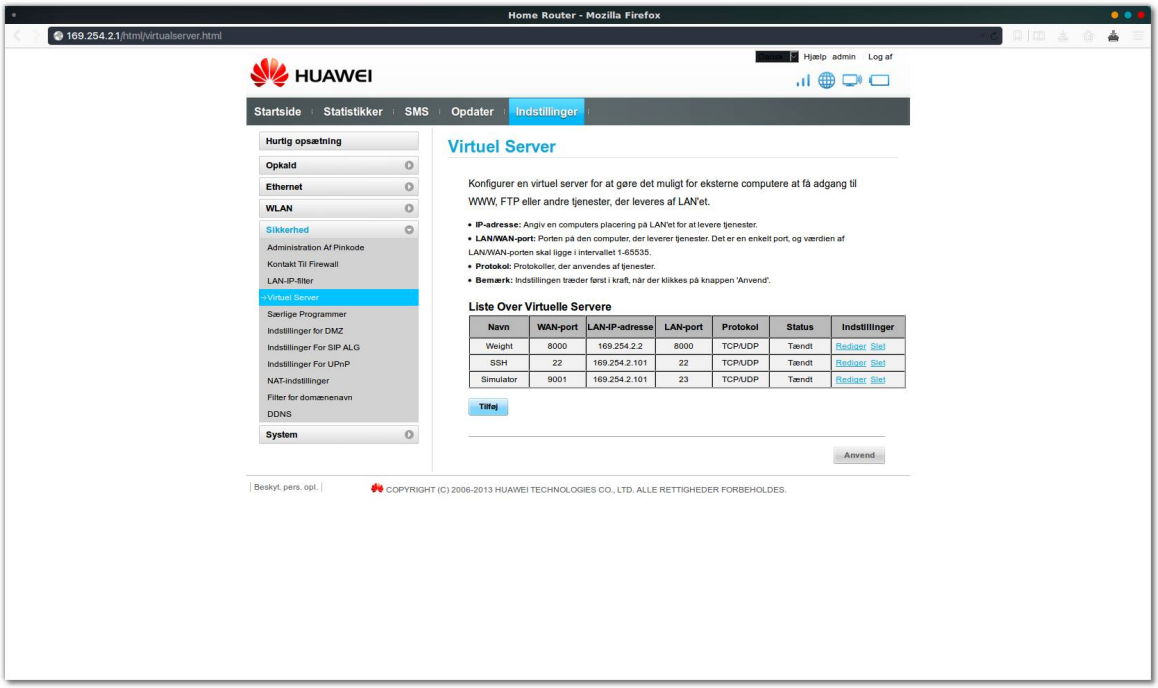
A.4.0.0.2 Tomcat Tomcat skal installeres. På Ubuntu eller Debian, der også tilbydes til Raspberry Pi, gøres dette med kommandoen

```
sudo apt-get install tomcat8.
```

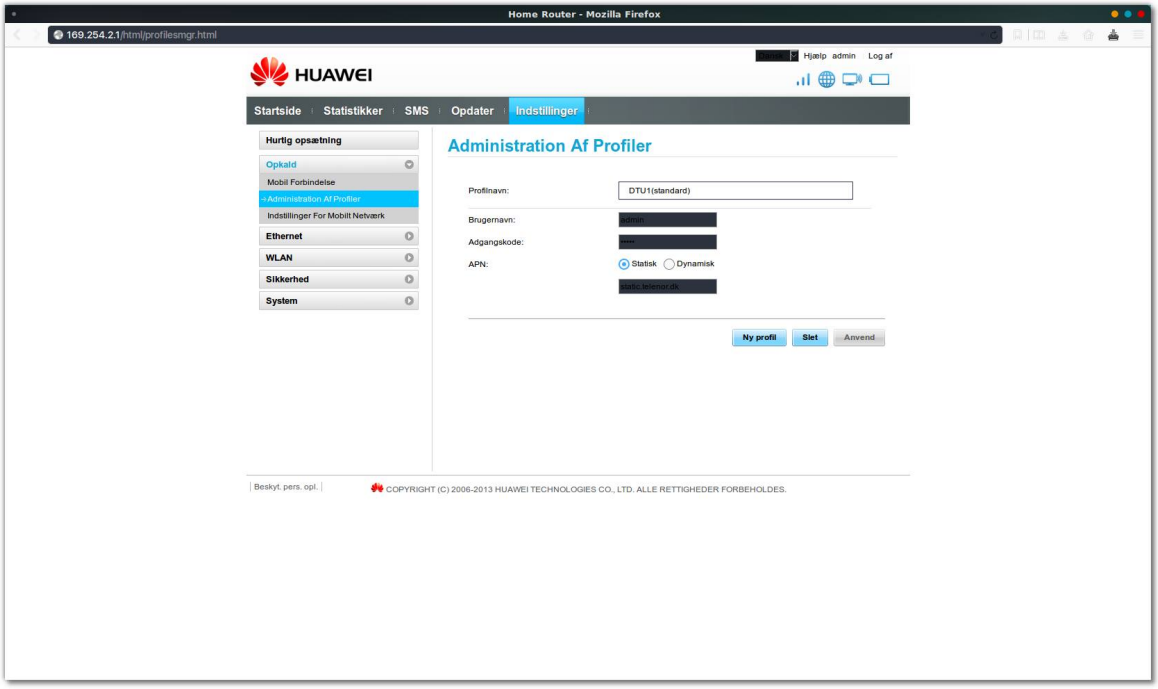
Dette installerer servere, og programmet befinder sig typisk i `/var/lib/tomcat8`. Herfra skal kildekoden compiles og pakkes til en .war fil. Hvis man har compilet programmet gøres dette blot ved at .zip pakke projektet, ændre forlængelsen fra .zip til .war og så flytte det til `/var/lib/tomcat8/webapps/`. Hvis man har konstrueret sit program ud fra den skabelon der tilbydes med henholdsvis IntelliJ eller GWT gøres dette med kommandoen `ant clean && ant war` i terminalen.



Figur A.6: Huawei Router - Front page



Figur A.7: Huawei Router - Port forwarding



Figur A.8: Huawei Router - Static IP profile

Kapitel B

Kode

B.1 Client

```
1  package com.weightinterface.myproject.client;
2
3  import com.google.gwt.user.client.ui.Composite;
4
5  /**
6   * Created by haanin on 5/2/16.
7   */
8  public abstract class Content extends Composite {
9      public abstract String getToken();
10     public abstract String getWindowTitle();
11 }
```

```
1  package com.weightinterface.myproject.client;
2
3  import com.google.gwt.event.logical.shared.ValueChangeEvent;
4  import com.google.gwt.event.logical.shared.ValueChangeHandler;
5  import com.google.gwt.http.client.Header;
6  import com.google.gwt.user.client.History;
7  import com.google.gwt.user.client.Window;
8  import com.google.gwt.user.client.ui.RootPanel;
9  import com.weightinterface.myproject.client.pages.LoginPage;
10 import com.weightinterface.myproject.client.pages.Welcome;
11 import com.weightinterface.myproject.client.pages.WeightInterface;
12
13 /**
14  * Created by haanin on 5/2/16.
15  */
16 public class ContentContainer implements ValueChangeHandler<String> {
17
18     @Override
19     public void onValueChange(ValueChangeEvent<String> event) {
20         String token = (String) event.getValue();
21         this.setContentByToken(token);
22     }
23
24     public void setContentByToken(String token) {
25         if (token.equals(PageDefs.WelcomeToken)) {
26             this.setContent(new Welcome());
27         } else if (token.equals(PageDefs.AppToken)) {
28             this.setContent(new WeightInterface());
```

```
29          } else if(token.equals(PageDefs.LoginToken)) {
30                  this.setContent(new LoginPage());
31          }
32      }
33
34      private void setContent(Content content) {
35          RootPanel contentRoot = RootPanel.get("content");
36          contentRoot.clear();
37          this.content = content;
38          History.newItem(content.getToken(), false);
39          // check for special initializations:
40          if (content.getToken().equals(PageDefs.AppToken)) {
41              ((WeightInterface) content).MyInit();
42          }
43          if (content.getToken().equals(PageDefs.LoginToken)) {
44              ((LoginPage) content).MyInit();
45          }
46          contentRoot.add(content);
47          Window.setTitle(content.getWindowTitle());
48          Window.scrollTo(0, 0);
49      }
50
51      private static ContentContainer myInstance = new ContentContainer();
52      public static synchronized ContentContainer getInstance() {
53          return myInstance;
54      }
55      private ContentContainer() {
56          History.addValueChangeHandler(this);
57      }
58
59      private Content content;
60
61  }
```

```
1  package com.weightinterface.myproject.client;
2
3  /**
4   * Created by haanin on 5/2/16.
5   */
6  public class PageDefs{
7      public static final String WelcomeToken = "WELCOME";
8      public static final String WelcomeTitle = "Welcome";
9      public static final String LoginToken = "LOGIN";
10     public static final String LoginTitle = "Login";
11     public static final String AppToken = "WI";
12     public static final String AppTitle = "WeightInterace";
13 }
```

B.1.1 Pages

B.1.1.1 Welcome

```
1  package com.weightinterface.myproject.client.pages;
2
3  import com.google.gwt.core.client.EntryPoint;
4  import com.weightinterface.myproject.client.Content;
5  import com.weightinterface.myproject.client.PageDefs;
6
7  /**
8   * Created by haanin on 5/2/16.
9   */
10 public class Welcome extends Content implements EntryPoint {
```

```
11      @Override
12      public String getToken() {
13          return PageDefs.WelcomeToken;
14      }
15
16      @Override
17      public String getWindowTitle() {
18          return PageDefs.WelcomeTitle;
19      }
20
21      @Override
22      public void onModuleLoad() {
23          new LoginPage();
24          //new WeightInterface();
25      }
26  }
```

B.1.1.2 WeightSimulator

```
1
2   import java.net.ServerSocket;
3   import java.net.Socket;
4   import java.io.InputStream;
5   import java.io.OutputStream;
6   import java.io.IOException;
7   import java.io.PrintWriter;
8   import java.io.*;
9   import java.util.Random;
10
11
12  public class WeightSimulator implements Runnable
13  {
14      ServerSocket serverSocket = null;
15      Socket clientSocket = null;
16      Thread listener = null;
17
18
19      public WeightSimulator(int port) throws IOException
20      {
21          serverSocket = new ServerSocket(port);
22
23          listener = new Thread (this);
24
25          listener.start();
26      }
27
28
29      public void run()
30      {
31          boolean bError = false;
32          while(!bError)
33          {
34              try
35              {
36                  clientSocket = serverSocket.accept();
37                  synchronized (clientSocket)
38                  {
39                      try
40                      {
41                          PrintWriter out2 = new PrintWriter(getOutputStream(),true);
42
43
44                          Random random = new Random();
45                          BufferedReader in =
```

```
                                    new BufferedReader(
                                            new InputStreamReader(clientSocket.getInputStream(
                            String line;
                            while((line = in.readLine()) != null)
                            {
                                System.out.println(line);
                                if(line.contains("SI"))
                                {
                                    out2.println("Weight: "+getRandomValue(random,0,6,4)+" kg"
                                }
                                else if(line.contains("TI"))
                                {
                                    out2.println("Tare");
                                }
                            }


                        }
                        catch (Exception e)
                        {
                            System.err.println("Exception in wait, "+ e.getMessage());
                        }
                        try
                        {
                            clientSocket.close();
                        }
                        catch (Exception e)
                        {
                            System.err.println("Exception in close, "+ e.getMessage());
                        }
                    }
                }
                catch (IOException e)
                {
                    bError = true;
                }
            }

            try
            {
                serverSocket.close();
            }
            catch (Exception e)
            {
                System.err.println("Exception in close, "+ e.getMessage());
            }
        }

        public static String getRandomValue(final Random random,
        final int lowerBound,
        final int upperBound,
        final int decimalPlaces){

        if(lowerBound < 0 || upperBound <= lowerBound || decimalPlaces < 0){
            throw new IllegalArgumentException("Put error message here");
        }

        final double dbl =
            ((random == null ? new Random() : random).nextDouble() //
                * (upperBound - lowerBound))
                + lowerBound;
        return String.format("%." + decimalPlaces + "f", dbl);

    }
```

```
110
111
112     public void disconnect()
113     {
114         synchronized (clientSocket)
115         {
116             try
117             {
118                 clientSocket.notify();
119             }
120             catch (Exception e)
121             {
122                 System.err.println("Exception in notify, "+ e.getMessage());
123             }
124         }
125     }
126
127
128     public void stop()
129     {
130         listener.interrupt();
131         try
132         {
133             serverSocket.close();
134         }
135         catch (Exception e)
136         {
137             System.err.println("Exception in close, "+ e.getMessage());
138         }
139     }
140
141
142     public InputStream getInputStream() throws IOException
143     {
144         if(clientSocket != null)
145         {
146             return(clientSocket.getInputStream());
147         }
148         else
149         {
150             return(null);
151         }
152     }
153
154
155
156     public OutputStream getOutputStream() throws IOException
157     {
158         if(clientSocket != null)
159         {
160             return(clientSocket.getOutputStream());
161         }
162         else
163         {
164             return(null);
165         }
166     }
167
168     public static void main(String[] args) throws IOException
169     {
170         WeightSimulator server = new WeightSimulator(23);
171     }
172 }
```

B.1.1.3 LoginPage

```
1  package com.weightinterface.myproject.client.pages;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.event.dom.client.KeyDownEvent;
7  import com.google.gwt.event.dom.client.KeyDownHandler;
8  import com.google.gwt.user.client.Window;
9  import com.google.gwt.user.client.rpc.AsyncCallback;
10 import com.google.gwt.user.client.ui.*;
11 import com.weightinterface.myproject.client.Content;
12 import com.weightinterface.myproject.client.ContentContainer;
13 import com.weightinterface.myproject.client.PageDefs;
14 import com.weightinterface.myproject.client.services.LoginService;
15 import com.weightinterface.myproject.client.services.LoginServiceAsync;
16
17 import java.io.IOException;
18
19 /**
20  * Created by haanin on 5/2/16.
21  */
22 public class LoginPage extends Content {
23
24     private final LoginServiceAsync loginService = GWT.create(LoginService.class);
25     final Label loginLabel = new Label("WRITE YOUR ASSIGNED USERNAME");
26     final Button loginbutton = new Button("Login");
27     final TextBox username = new TextBox();
28     final VerticalPanel loginPanel = new VerticalPanel();
29     final VerticalPanel login2 = new VerticalPanel();
30     final HTMLPanel contentPanel = new HTMLPanel("");
31
32     public LoginPage() {
33     MyInit();
34     }
35
36     public void MyInit() {
37
38         loginbutton.setStyleName("login__submit");
39         RootPanel.get("content").setStyleName("background");
40         contentPanel.add(login2);
41         contentPanel.setStyleName("container");
42         loginPanel.setWidth("100%");
43         loginPanel.setHeight(Window.getClientHeight() + "px");
44         loginPanel.setHorizontalAlignment(HasAlignment.ALIGN_CENTER);
45         loginPanel.setVerticalAlignment(HasAlignment.ALIGN_MIDDLE);
46         loginPanel.add(contentPanel);
47         login2.setSpacing(70);
48         login2.setVerticalAlignment(HasAlignment.ALIGN_MIDDLE);
49         login2.setHorizontalAlignment(HasAlignment.ALIGN_CENTER);
50         login2.add(loginLabel);
51         login2.add(username);
52         login2.add(loginbutton);
53         RootPanel.get("content").add(loginPanel);
54         username.setStyleName("input");
55
56
57         loginbutton.addClickHandler(new ClickHandler() {
58             @Override
59             public void onClick(ClickEvent event) {
60                 login();
61             }
62         });
```

```
63          username.addKeyDownHandler(new KeyDownHandler() {
64              @Override
65              public void onKeyDown(KeyDownEvent event) {
66                  if(event.getNativeKeyCode()==13)
67                  {
68                      login();
69                  }
70              }
71          });
72      }
73
74      private void login() {
75          if(!username.getText().isEmpty())
76          {
77              try {
78                  loginService.checkLoggedIn(username.getText().trim(), new AsyncCallback<Bo
79                      @Override
80                      public void onFailure(Throwable caught) {
81                          Window.alert(caught.getLocalizedMessage());
82                      }
83
84                      @Override
85                      public void onSuccess(Boolean usernameCorrect) {
86                          if(usernameCorrect)
87                          {
88                              ContentContainer.getInstance().setContentByToken(PageDefs.AppT
89                          }
90                          else
91                          {
92                              Window.alert("Wrong username.");
93                              username.setText("");
94                              username.setFocus(true);
95                          }
96                      }
97                  });
98              } catch (IOException e) {
99                  e.printStackTrace();
100             }
101         }
102         else Window.alert("Write something please.");
103     }
104
105
106     @Override
107     public String getToken() {
108         return PageDefs.LoginToken;
109     }
110
111     @Override
112     public String getWindowTitle() {
113         return PageDefs.LoginTitle;
114     }
115
116
117 }
```

B.1.2 Services

B.1.2.1 GreetingService

```
1 package com.weightinterface.myproject.client.services;
2
3 import com.google.gwt.user.client.rpc.RemoteService;
```

```
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

import java.io.IOException;

/**
 * The client-side stub for the RPC service.
 */
@RemoteServiceRelativePath("greet")
public interface GreetingService extends RemoteService {
  //String greetServer(String name) throws IllegalArgumentException;
  String sendCommandToWeight(String command) throws IOException;
  String closeConnection() throws IOException;
  String openConnection() throws IOException;
}
```

B.1.2.2 GreetingServiceAsync

```
package com.weightinterface.myproject.client.services;

import com.google.gwt.user.client.rpc.AsyncCallback;

import java.io.IOException;

/**
 * The async counterpart of <code>GreetingService</code>.
 */
public interface GreetingServiceAsync {

  //String sendCommandToWeight(String s) throws IOException;
  void sendCommandToWeight(String command, AsyncCallback<String> callback)
      throws IOException;

  void closeConnection(AsyncCallback<String> asyncCallback) throws IOException;
  void openConnection(AsyncCallback<String> asyncCallback) throws IOException;

}
```

B.1.2.3 LoginService

```
package com.weightinterface.myproject.client.services;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

import java.io.IOException;

/**
 * Created by haanin on 4/28/16.
 */

@RemoteServiceRelativePath("login")
public interface LoginService extends RemoteService {
    Boolean checkLoggedIn(String username) throws IOException;
}
```

B.1.2.4 LoginServiceAsync

```
package com.weightinterface.myproject.client.services;

import com.google.gwt.user.client.rpc.AsyncCallback;

import java.io.IOException;
```

```
6
7  /**
8   * Created by haanin on 4/28/16.
9   */
10 public interface LoginServiceAsync {
11     void checkLoggedIn(String username, AsyncCallback<Boolean> asyncCallback) throws IOExc
12 }
```

B.1.3 Handlers

B.1.3.1 BatchClickHandler

```
1  package com.weightinterface.myproject.client.handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.ui.Button;
7  import com.google.gwt.user.client.ui.TextBox;
8  import com.weightinterface.myproject.client.pages.WeightInterface;
9
10 /**
11  * Created by haaning on 3/31/16.
12  */
13 public class BatchClickHandler implements ClickHandler {
14
15     private Button countButton;
16     private TextBox count;
17     private TextBox currentWeightDisplay;
18     private boolean batchIsReady = false;
19     private double average;
20     private double weight;
21     private boolean isConnected = true;
22     private WeightInterface wi = null;
23
24     public BatchClickHandler(Button countButton, TextBox count, TextBox currentWeightDispl
25     {
26         this.wi = wi;
27         this.countButton = countButton;
28         this.count = count;
29         this.currentWeightDisplay = currentWeightDisplay;
30     }
31
32
33     @Override
34     public void onClick(ClickEvent event) {
35         {
36
37             if(count.getText().matches("[0-9]+\\.?\\,?[0-9]*") && isConnected())
38             {
39                 if(!batchIsReady)
40                 {
41                     average = wi.res / Double.parseDouble(count.getText());
42                     countButton.setText("Done");
43                     count.setEnabled(false);
44                     Window.alert("1 weighs: "+average+"\n"+"Place full amount now and press the 'Done' butt
45
46                     batchIsReady = true;
47                 }
48                 else
49                 {
50                     count.setText("");
51                     countButton.setText("Count");
```

```
52                      count.setEnabled(true);
53                      batchIsReady = false;
54                  }
55
56              } else Window.alert("Cannot parse input or is not connected.");
57          }
58      }
59
60      public double calculateAmount()
61      {
62          return wi.res / average;
63      }
64
65      public double getAverage()
66      {
67          return average;
68      }
69
70      public boolean isReady() {
71          return batchIsReady;
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(currentWeightDisplay.getText().substring(0,(currentWeigh
77      }
78
79      public void setConnected(boolean status) { isConnected = status; }
80
81      public boolean isConnected() { return isConnected; }
82  }
```

B.1.3.2 QueueServerHandler

```
1  package com.weightinterface.myproject.client.handlers;
2
3  import com.google.gwt.dom.client.Style;
4  import com.google.gwt.event.dom.client.*;
5  import com.google.gwt.user.client.DOM;
6  import com.google.gwt.user.client.Timer;
7  import com.google.gwt.user.client.Window;
8  import com.google.gwt.user.client.rpc.AsyncCallback;
9  import com.weightinterface.myproject.client.pages.WeightInterface;
10
11 import java.io.IOException;
12
13 /**
14  * Created by haaning on 4/21/16.
15  */
16 public class QueueServerHandler implements ClickHandler, KeyUpHandler {
17
18
19      WeightInterface wi = null;
20      boolean isNegative = false;
21      private boolean alreadyClicked = false;
22
23      public QueueServerHandler(WeightInterface wi)
24      {
25      this.wi = wi;
26          wi.wb.setVisible(false);
27          doInit();
28      }
29
30      public void doInit() {
```

```
31          try {
32              weightPrompt();
33          } catch (IOException e) {
34              e.printStackTrace();
35          }
36      }
37
38      /**
39       * Fired when the user clicks on the sendButton.
40       */
41          public void onClick(ClickEvent event) {
42              /*if(alreadyClicked)
43              {
44                  return;
45              }
46              else
47              {
48              try {
49                  alreadyClicked = true;
50                  weightPrompt();
51              } catch (IOException e) {
52                  e.printStackTrace();
53              }
54              }*/
55          }
56
57
58          public void timerOff()
59          {
60              wi.timerOn = false;
61              wi.wb.setText("ON");
62          }
63
64          public void onKeyUp(KeyUpEvent event) {
65              /*if (event.getNativeKeyCode() == KeyCodes.KEY_ENTER) {
66                  try {
67                      weightPrompt();
68                  } catch (IOException e) {
69                      e.printStackTrace();
70                  }
71              }*/
72          }
73
74          private void weightPrompt() throws IOException {
75              wi.getGreetingService().sendCommandToWeight("SI", new AsyncCallback<String>()
76                  public void onFailure(Throwable caught) {
77                      //DOM.getElementById("sk-circle").getStyle().setDisplay(Style.Display.
78                      wi.currentWeightDisplay.setVisible(true);
79                      wi.currentWeightDisplay.setText("Weight communication failed.");
80                      wi.wb.setText("ON");
81                      alreadyClicked = false;
82                      wi.currentWeightDisplay.setVisibleLength(wi.currentWeightDisplay.getTe
83                      try {
84                          weightPrompt();
85                      } catch (IOException e) {
86                          e.printStackTrace();
87                      }
88                  }
89
90                  public void onSuccess(String weight) {
91                      //DOM.getElementById("sk-circle").getStyle().setDisplay(Style.Display.
92                      isNegative = false;
93                      wi.wb.setText("OFF");
94                      char c = weight.charAt(7);
```

```
95                      if (!(c == ' ')) isNegative = true;
96                      wi.res = Double.parseDouble(weight.substring(8,14));
97                      if(wi.batch.isReady()){
98                          double batchdouble = wi.batch.calculateAmount();
99                          wi.count.setText(""+(int) batchdouble);};
100                     if(isNegative)
101                     {
102                         wi.currentWeightDisplay.setText("-"+wi.res+" kg");
103                         DOM.getElementById("myBar").getStyle().setWidth(0, Style.Unit.PCT)
104                         DOM.getElementById("label").setInnerText("0%");
105                     } else
106                     {
107                         wi.currentWeightDisplay.setText(wi.res + " kg");
108                         DOM.getElementById("myBar").getStyle().setWidth(wi.res * 16.66, St
109                         DOM.getElementById("label").setInnerText(Double.toString(wi.res *
110                     }
111                     try {
112                         weightPrompt();
113                     } catch (IOException e) {
114                         e.printStackTrace();
115                     }
116                 }
117             });
118         }
119 }
```

B.1.3.3 TareClickHandler

```
1  package com.weightinterface.myproject.client.handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.rpc.AsyncCallback;
7  import com.weightinterface.myproject.client.pages.WeightInterface;
8
9  import java.io.IOException;
10
11 /**
12  * Created by haaning on 4/14/16.
13  */
14 public class TareClickHandler implements ClickHandler{
15
16     WeightInterface wi = null;
17     QueueServerHandler qh = null;
18
19     public TareClickHandler(WeightInterface wi, QueueServerHandler qh)
20     {
21         this.wi = wi;
22         this.qh = qh;
23
24
25     }
26     @Override
27     public void onClick(ClickEvent event) {
28         try {
29             //qh.tareTimerOff();
30             wi.getGreetingService().sendCommandToWeight("TI", new AsyncCallback<String>()
31                 @Override
32                 public void onFailure(Throwable caught) {
33                     wi.windowAlert(caught.getLocalizedMessage());  } //qh.startTimer(); }
34
35                 @Override
36                 public void onSuccess(String result) {
```

```
37                      if(result.equals("TI -"))
38                      {
39                          try {
40                              wi.getGreetingService().sendCommandToWeight("T", new AsyncCall
41                                  @Override
42                                  public void onFailure(Throwable caught) {
43
44                                  }
45
46                                  @Override
47                                  public void onSuccess(String result) {
48
49                                  }
50                              });
51                          } catch (IOException e) {
52                              e.printStackTrace();
53                          }
54                      }
55                      //qh.startTimer();
56                      /*wi.setWeightTextBar("0");
57                      wi.setAnalogBar(0);
58                      wi.setTimer(false);*/
59
60              }
61              });
62          }   catch (IOException e) {
63              e.printStackTrace();
64          }
65      }
66  }
```

B.2 Server

B.2.0.1 GreetingServiceImpl

```
1  package com.weightinterface.myproject.client.handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.ui.Button;
7  import com.google.gwt.user.client.ui.TextBox;
8  import com.weightinterface.myproject.client.pages.WeightInterface;
9
10 /**
11  * Created by haaning on 3/31/16.
12  */
13 public class BatchClickHandler implements ClickHandler {
14
15     private Button countButton;
16     private TextBox count;
17     private TextBox currentWeightDisplay;
18     private boolean batchIsReady = false;
19     private double average;
20     private double weight;
21     private boolean isConnected = true;
22     private WeightInterface wi = null;
23
24     public BatchClickHandler(Button countButton, TextBox count, TextBox currentWeightDispl
25     {
26         this.wi = wi;
27         this.countButton = countButton;
28         this.count = count;
```

```
29          this.currentWeightDisplay = currentWeightDisplay;
30      }
31
32
33      @Override
34      public void onClick(ClickEvent event) {
35          {
36
37              if(count.getText().matches("[0-9]+\\.?\\,?[0-9]*") && isConnected())
38              {
39                  if(!batchIsReady)
40                  {
41                      average = wi.res / Double.parseDouble(count.getText());
42                      countButton.setText("Done");
43                      count.setEnabled(false);
44                      Window.alert("1 weighs: "+average+"\n"+"Place full amount now and press the 'Done' butt
45
46                      batchIsReady = true;
47                  }
48                  else
49                  {
50                      count.setText("");
51                      countButton.setText("Count");
52                      count.setEnabled(true);
53                      batchIsReady = false;
54                  }
55
56              } else Window.alert("Cannot parse input or is not connected.");
57          }
58      }
59
60      public double calculateAmount()
61      {
62          return wi.res / average;
63      }
64
65      public double getAverage()
66      {
67          return average;
68      }
69
70      public boolean isReady() {
71          return batchIsReady;
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(currentWeightDisplay.getText().substring(0,(currentWeigh
77      }
78
79      public void setConnected(boolean status) { isConnected = status; }
80
81      public boolean isConnected() { return isConnected; }
82  }
```

B.2.0.2 LoginServiceImpl

```
1  package com.weightinterface.myproject.client.handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.ui.Button;
7  import com.google.gwt.user.client.ui.TextBox;
```

```
import com.weightinterface.myproject.client.pages.WeightInterface;

/**
 * Created by haaning on 3/31/16.
 */
public class BatchClickHandler implements ClickHandler {

    private Button countButton;
    private TextBox count;
    private TextBox currentWeightDisplay;
    private boolean batchIsReady = false;
    private double average;
    private double weight;
    private boolean isConnected = true;
    private WeightInterface wi = null;

    public BatchClickHandler(Button countButton, TextBox count, TextBox currentWeightDispl
    {
        this.wi = wi;
        this.countButton = countButton;
        this.count = count;
        this.currentWeightDisplay = currentWeightDisplay;
    }


    @Override
    public void onClick(ClickEvent event) {
        {

            if(count.getText().matches("[0-9]+\\.?\\,?[0-9]*") && isConnected())
            {
                if(!batchIsReady)
                {
                    average = wi.res / Double.parseDouble(count.getText());
                    countButton.setText("Done");
                    count.setEnabled(false);
                    Window.alert("1 weighs: "+average+"\n"+"Place full amount now and press the 'Done' butt

                    batchIsReady = true;
                }
                else
                {
                    count.setText("");
                    countButton.setText("Count");
                    count.setEnabled(true);
                    batchIsReady = false;
                }

            } else Window.alert("Cannot parse input or is not connected.");
        }
    }

    public double calculateAmount()
    {
        return wi.res / average;
    }

    public double getAverage()
    {
        return average;
    }

    public boolean isReady() {
        return batchIsReady;
```

```
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(currentWeightDisplay.getText().substring(0,(currentWeigh
77      }
78
79      public void setConnected(boolean status) { isConnected = status; }
80
81      public boolean isConnected() { return isConnected; }
82  }
```

Bibliography

[1] Google (2016) "Ajax Communication: Introduction," *http://www.gwtproject.org/doc/latest/tutorial/*

[2] WHATWG (2016) "XMLHTTPRequest Standard," *https://xhr.spec.whatwg.org/*.

[3] Google (2016) "GWT RPC," *http://www.gwtproject.org/doc/latest/tutorial/RPC.html*.

[4] Lookup-Tables (2016) "ASCII tables encoding," *http://www.asciitable.com/*.

[5] JetBrains (2016) "Enabling GWT Support," *https://www.jetbrains.com/help/idea/2016.1/enabling-gwt-support.html*.

[6] Foundation, C. . R. P. (2016) "Ubuntu Pi Flavour Maker," *https://ubuntu-pi-flavour-maker.org/download/*.

[7] Upd8, W. (2016) "Oracle Java PPA Updated with ARM Support," *http://www.webupd8.org/2013/12/oracle-java-ppa-updated-with-arm-support.html*.