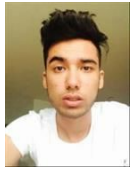


Projektnavn: CDIO Final, udgave 2
Gruppenummer: 18
Afleveringsfrist: Fredag, 19/6 kl 12.00

Denne rapport er afleveret via Campusnet (der skrives ikke under)

Denne rapport indeholder [53] sider inkl. denne side.



Studienummer: s144841
Navn: Nael Rashdeen



Studienummer: s144849
Navn: Delvaux Paulin Richard Niyonzima



Studienummer: s144859
Navn: Kenneth Skovsgaard Frandsen



Studienummer: s133275
Navn: Peter Asmussen



Studienummer: s123503
Navn: Thomas Liljegren



Studienummer: s144839
Navn: Liban Jama Awil Dirir



Studienummer: s123172
Navn: Martin Vieth

Time regnskab

Dette er vores samlede time regnskab, vi har i bilag vedlagt en mere detaljeret udgave hvor vi hver har indsat antallet af timer vi har brugt på.

Design	Impl.	Test	Dok.	Andet	Ialt
65,25	126	11,5	92,5	54,5	350,25

Se bilag 1 for detaljeret regnskab

Brugervejledning til opsætning af software

For at kunne køre applikationen, er der nogle ting som skal være på plads før det virker:

JRE/JDK 1.7 (eller nyere):

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Google Web Toolkit SDK (2.6.1)

<http://www.gwtproject.org/versions.html>

GWT Eclipse Plugin

Åbn Eclipse, og derefter vælg "Help" → "Install new Software" og indtast følgende i adressebaren:

<https://dl.google.com/eclipse/plugin/4.4>

(linket virker til Eclipse Luna 4.4)

Herefter skal følgende elementer vælges:

Google Plugin for Eclipse (required)

GWT Designer for GPE

Når al software er blevet installeret, skal man herefter sikre sig at man er koblet på vægten.

Man kan vælge to forskellige IP-adresser, som der kan ændres på i kildekoden, under

opr → *src* → *server* → *impl* → *ASE.java*

Vægt forbindelse

Global IP (default)

Er brugerens enhed koblet op på et vilkårligt WiFi, er forbindelsen oprettet ved at bruge følgende IP-adresse: **62.79.16.17**

Lokal IP

Er brugerens enhed koblet op på HUAWEI Access Pointet, er forbindelsen oprettet ved at bruge følgende IP-adresse: **169.254.2.2**

```
private static String host = "62.79.16.17";  
private static int port = 8000;
```

(skal IP-adressen ændres, skal `private static String host = "62.79.16.17"` ændres til det ønskede)

Før projektet køres gennem Eclipse, skal brugerens MySQL-server op og køre. Når vedkommende har tændt denne, skal vores database importeres (denne ligger i i Eclipse-projektet). For at bruge den, åbnes kommandoprompten, og følgende kommando skal indtastes: `source <filepath.sql>`.

Nu kan projektet køres fra Eclipse.

Login i programmet:

UserID: 4

password:iloveyou

Indholdsfortegnelse

[Brugervejledning til opsætning af software](#)

[Indledning](#)

[Kravspecifikation](#)

[Systems Delkomponenter](#)

[Web Applikation](#)

[Vægt](#)

[Access-Point](#)

[Aktører](#)

[Bruger](#)

[Afvejning](#)

[Diagrammer](#)

[Use Case](#)

[KlasseDiagram](#)

[System Sekvens diagram](#)

[BCE](#)

[Design system sekvens diagram](#)

[3-Lags modellen](#)

[Google Web Toolkit \(GWT\)](#)

[Interfaces](#)

[Cascading Style Sheets \(CSS\)](#)

[MySQL](#)

[JDBC](#)

[Opbygning af projekt](#)

[Client](#)

[UI](#)

[Service](#)

[Server](#)

[Asynchronous I/O](#)

[Delta Bar](#)

[Placering](#)

[Løbende opdatering](#)

[Intern adgang vs Internettet](#)

[LoginView](#)

[WeightView](#)

[ASE](#)

[Forbindelse til vægten](#)

[Vægt kommunikation](#)

[Database kommunikation](#)

[UnitWeight](#)

[Afvejning](#)

[Test](#)

[JUnit](#)

[Brugertest](#)

[Opsætning af accesspoint og vægt](#)

[Konklusion](#)

[Bilag 1. Time regnskab](#)

[Bilag 2. plakat](#)

Indledning

Der ønskes en Google Web Toolkit-applikation, som kan køres fra en vilkårlig browser på en PC. Applikationen skal fungere som en styre-grænseflade for vægten til simpel afvejning. I denne sammenhæng, vil vægten fungere som en server, der styres af en TCP-klient fra GWT-applikationen. Det vil her være hensigten, at alt styring af vægten foregår i denne applikation.

Der vil således også være lejlighed til at kunne designe en helt ny brugerinterface til apparatet.

Derudover, vil det også være muligt at kunne designe applikationen efter en eller flere afvejnings procedurer.

Vi skal i denne sammenhæng også gøre brug af det nye access-point, som kan tilkobles den fysiske vægt.

Kravspecifikation

Systems Delkomponenter

Web Applikation

- Skal implementeres som en GWT applikation.
- Alle outputs skal valideres iht. gyldige områder
- Forside skal kunne tilgås.
- Output skal kunne modtages fra vægt.
- Der skal vises passende fejlmeddelelser ved fejl.
- Skal være browser-uafhængig.

Vægt

- Gør brug af den fysiske vægt fra 13-ugers perioden.

Access-Point

- Der skal skabes forbindelse mellem access-point, vægt og GWT-applikationen.

Aktører

Bruger

- Bruger skal kunne få en nettovægt fra den fysiske vægt.

Afvejning

- Bruger skal kunne veje sit produkt.
- Vægtens tara-belastning skal gemmes.
- En afvejnings resultat kan kun accepteres hvis det ligger inden for en vis tolerance.
- Der skal udføres netto-kontrol.

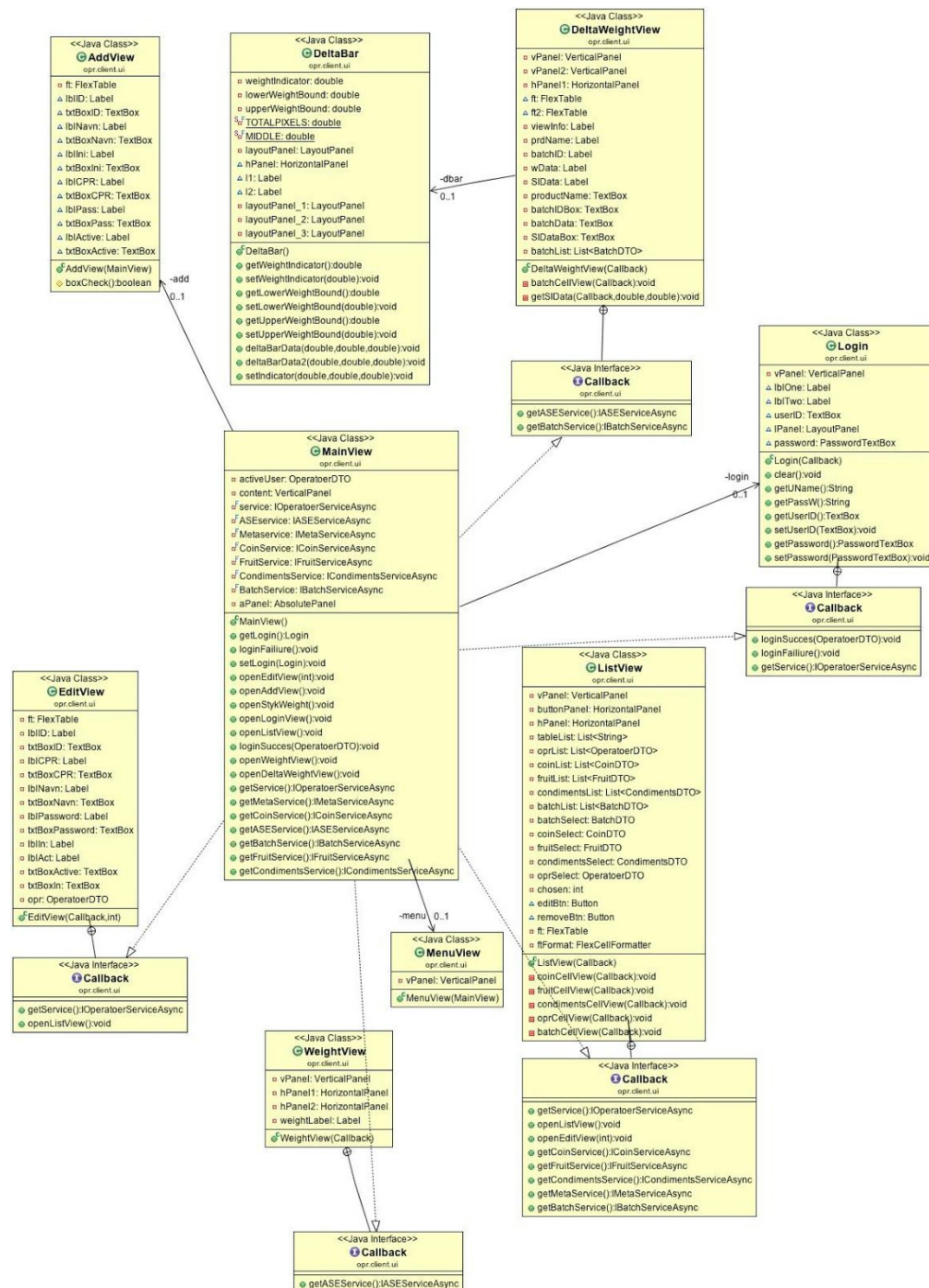
Diagrammer

Use Case

Brugeren logger ind på systemet og får adgang til en menu, hvor man kan gøre følgende ting:

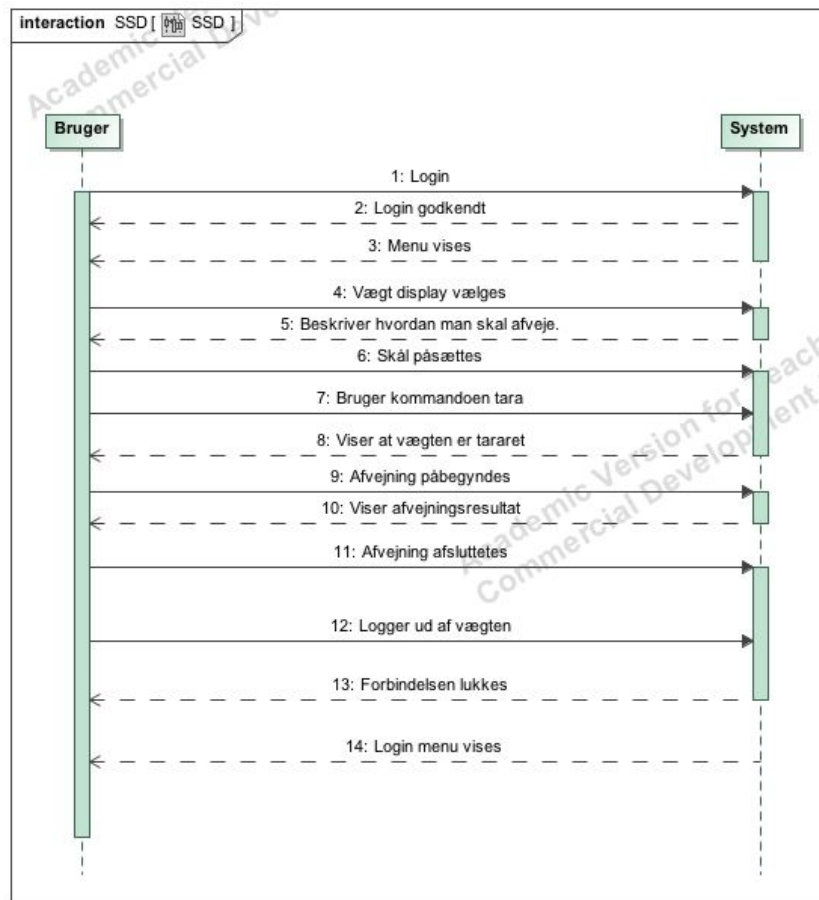
- 1. Weight**
 - a. Get weight
 - b. Tara
- 2. Delta**
- 3. Unit**
 - a. Coins
 - b. Fruit
 - c. Condiments
- 4. List**
- 5. Logout**

KlasseDiagram



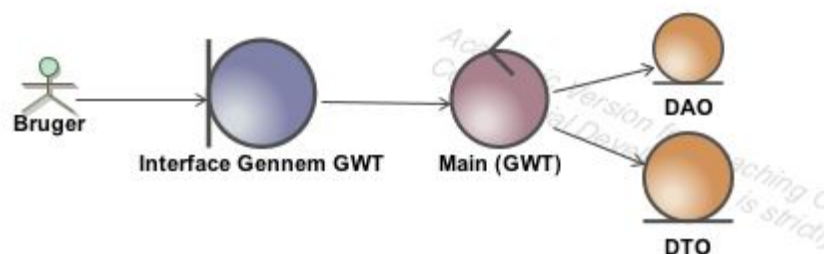
På ovenstående figur ses vores klassediagram, for kernen af programmet. Vi ser at vi har en kerne klasse som er vores mainview. Det er inde under vores mainview klasse at vi har vores bruger overflade for applikationen. Vi ser at alle callback metoderne henviser til mainview. Dette er jo klart det, det er der skal ske. Når et funktion er blevet brugt inde på grænsefladen af programmet, og funktionen er færdig og en ny funktion skal til at ske er det mainview der kommer frem. Derfor har mainview også mange metoder som klasserne arver.

System Sekvens diagram



System Sekvens Diagrammet (SSD) viser et eksempel på en brugers interaktion med systemet af UC1, hvor man ser hvilket ting bruger for besked på at gøre og hvad svar brugeren får, fra de enkelte handlinger brugeren udfører.

BCE



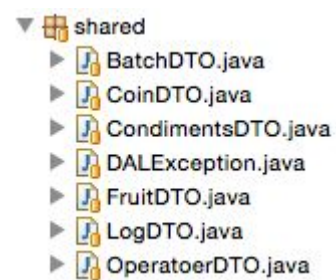
I GWT ligger boundary og controller smeltet sammen som én enhed. Når det skal tegnes ind i en BCE model skiller man dem ad da de laver to forskellige ting. Boundary håndterer/viser vores interface, som er den del brugeren kan se og interagere med. Vores Main(controller)

er koden under UI-pakken i GWT. I UI-pakken ligger koden også til GWT, hvilket derfor gør det svært at, dele det op på nogen måde i koden. Da man f.eks bygger en knap op inde i UI, gemmer man både en ClickHandler som håndterer hvad der skal ske når knappen bliver trykket på, samt udseendet på knappen. Systemet bruger derudover også en masse DAO'er og DTO'er. Vores DAO'er bruges til at hente data i blandt andet vores database, hvorimod DTO'en bruges til at sende dataen i vores system.

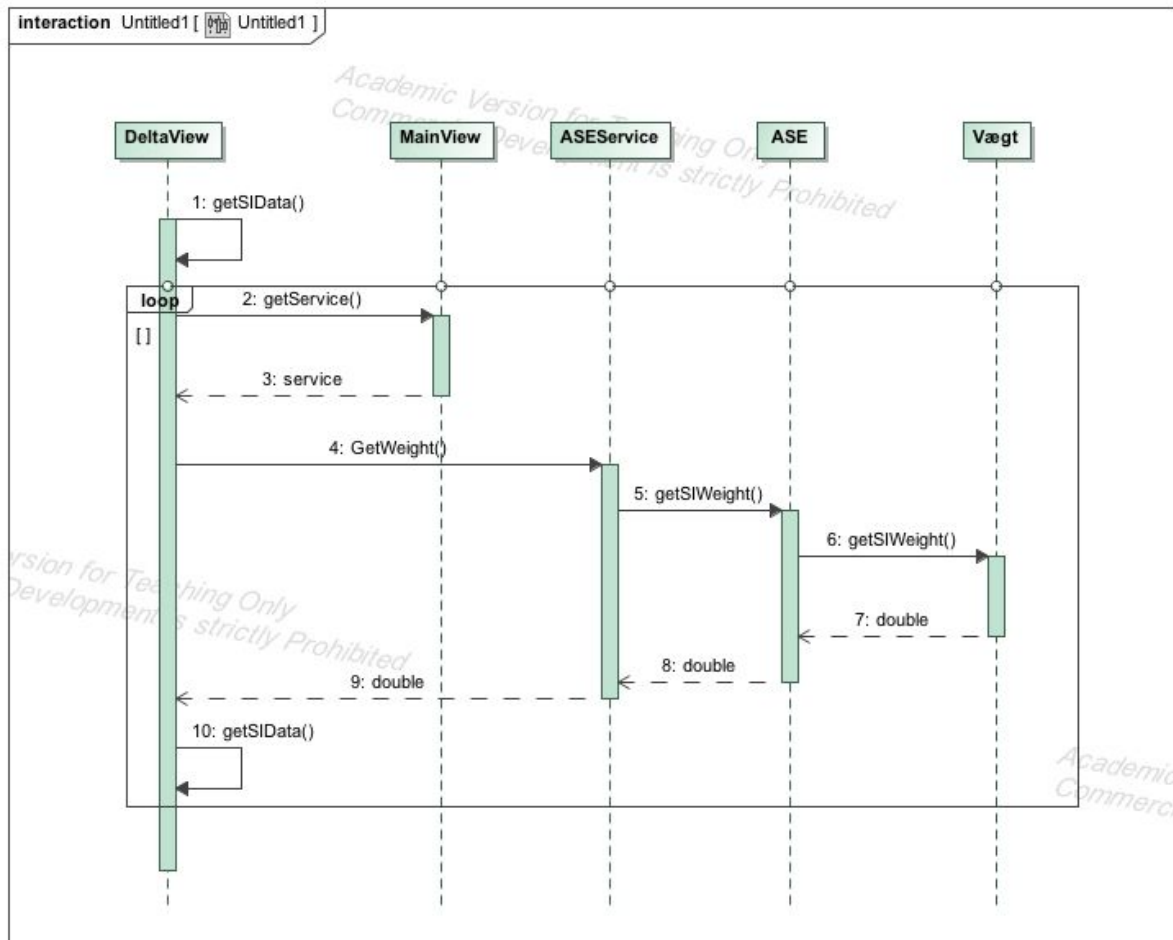
DAO



DTO



Design system sekvens diagram

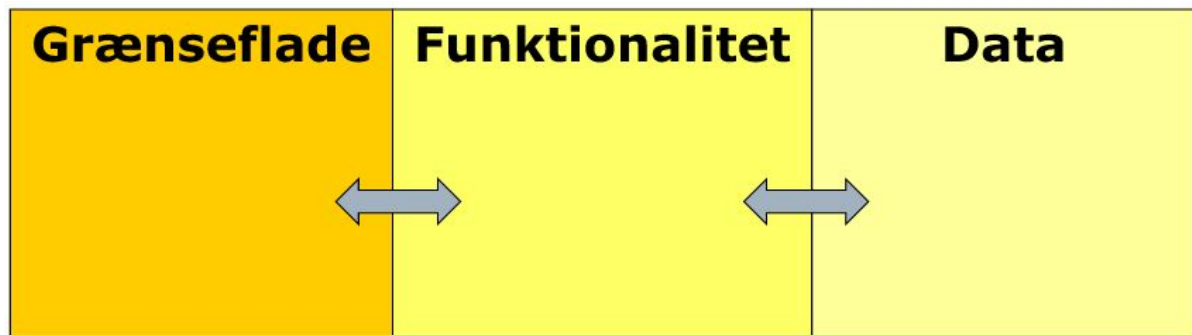


Ovenstående diagram er et udsnit af vores program, det viser kommunikationen der sker ved et brug af metoden getSIData().

3-Lags modellen

Der er gode grunde til at opdele et system i mindre lag, der fungerer sammen med hinanden. På denne måde bliver systemets og dets struktur lettere at overskue, samt at der bliver bedre mulighed for genbrug. Kort sagt kan man ved hjælp af 3-lags modellen opnå større vedligeholdelsesvenlighed. Udover den brede overskuelighed, stræber 3-lags modellen efter, at opnå lav kobling (GRASP), som er let at vedligeholde og let at udskifte og ændre lagene med.

Som der kan ses herunder består modellen af 3 lag, som indeholder en grænseflade, funktionalitetslag og et datalag.

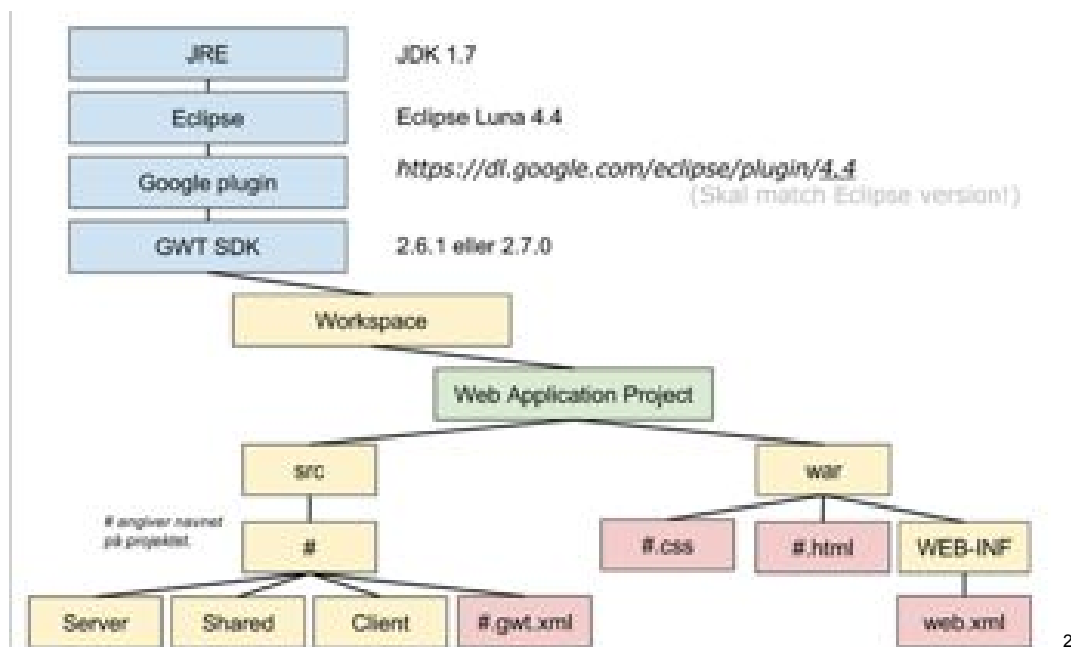


For at få en forståelse af 3-lags modellen kan man se på et simpelt eksempel, hvor hvert lag udgøres af en enhed . Ser man på hvordan lagene hænger sammen, gælder det at sammenhængen skal være fra venstre mod højre. Et grænseflade-objekt skal referere til et funktionalitets objekt, men et funktionalitets objekt kender ikke til grænseflade objektet. På samme måde gælder det også for et funktionalitets objekt, som skal referere til et dataobjekt, men et dataobjekt kender ikke til funktionalitets objektet.¹

¹ <http://dtu20.be/3LM.pdf>

Google Web Toolkit (GWT)

Vi implementerer vores webbaserede controller via GWT, som er vores grænseflade, hvis vi skal tage udgangspunkt i 3-lags modellen. GWT er et open-source sæt af værktøjer. Det er netop disse værktøjer vi gør brug af, for at kunne danne vores hjemmeside. Det gode ved, at vi implementerer i GWT er at al koden er Java og derfor også noget vi kan arbejde med. En af de andre fordele ved GWT er, at man også kan bruge metoden til at fremstille applikationer til mobiler og tablets. Dette er dog ikke noget vi har tænkt os at fremstille. En opstilling af GWT, samt dens plug-ins og projekter kan se således ud:



Vi skal også have en form for JDK, Java Development Kit, som arbejder sammen med vores Eclipse. Hertil er Eclipse vores IDE som vi arbejder på. Til denne arbejds metode skal vi have nogle plugins til at kunne arbejde med GWT som det er at vi gerne vil. De to plugins er: *GWT SDK* og *Google plug-in*. Vi har hertil vores Web applikation, som er kernen for vores grænseflade. Men under web applikationen har vi også nogle overflødige klasser, som vi bare har valgt at slette. Disse klasser er markeret med rød på den ovenstående figur. I de to underklasser *src* og *war* har vi vores implementation. Vi har i *src*, vores database og vores implementation for de forskellige klasser i programmet. I *Web-Inf* har vi vores Web interface.

² Billede fra forelæsning -

<https://drive.google.com/folderview?id=0B-pPbZ8YwfkFcnhzZjNsS0V0dG8&usp=sharing&tid=0B1qIt-Xd6kaQalk4REc5Yk5ZWWM>

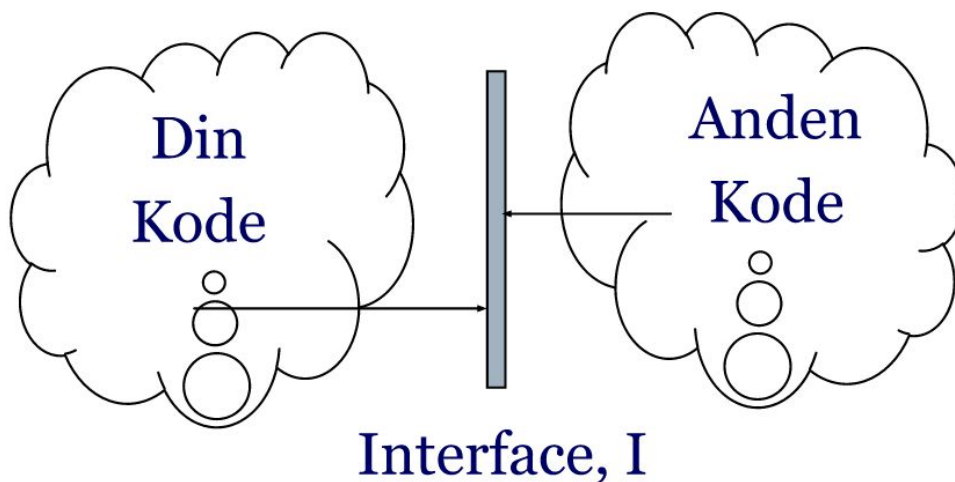
Interfaces

Et interface erklærer en række metoder, som en klasse skal implementere for, at den kan siges at være af typen i. Interfaces er en vigtig mekanisme til at opnå lav kobling, hvor der er lav afhængighed mellem klasserne, samt at der er større potentiale for genbrug. Dette er også et af hovedprincipperne i GRASP. I dens mest normale form, er et interface en gruppe af relaterede metoder med tomme "bodies".

Ved hjælp af interfaces kan man opnå en mere robust kode, idet at ens egen kode refererer kun til Interfacet I, hvilket medfører, at ændringer i implementeringen i "Anden Kode" ikke påvirker ens egen kode.

Alle metoder der erklæres i et interface er pr. definition abstract og public. Samtidigt kan interfaces kun indeholde konstanter og er pr. definition public static og final. En klasse kan implementere flere interfaces, men tager man en klasse der implementerer et interface, må den implementere alle metoder i interfacet, ellers erklæres den abstract.

På billedet herunder kan man se en visuel illustration af hvor interfaces egentlig er placeret i forhold til ens kode.

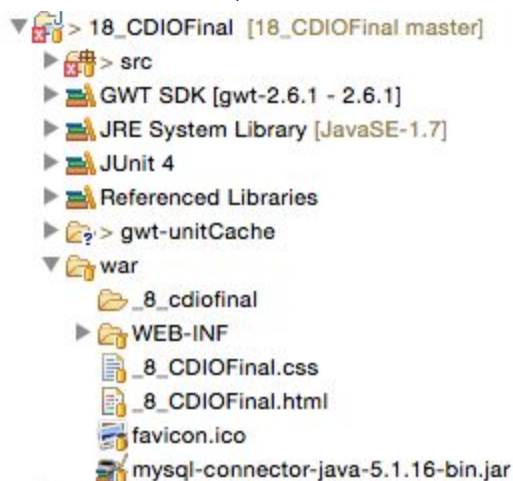


Cascading Style Sheets (CSS)

CSS er et programmeringssprog der bruges i GWT til at kunne farve og formgive interfacet. CSS bruges blandt andet til HTML. I CSS opretter man en klasse, den klasse kan indeholde både farver til kant, baggrundstekst, størrelse på skrift, skrifttyper, margin-justering og en masse andet .

Fordelen ved CSS er at filen kun ligger et sted, derved kan man kalde den samme CSS klasse mange steder. hvilket gør det nemt at rette i da der kun skal rettes et sted.

I GWT er filen altid placeret i war->WEB-INF, som vist herunder.



Ind i vores CSS fil (_8_CDIOFinal.css) har vi sat farver og udseende på blandt andet vores deltabar. Ser man nedenfor kan man se baggrunden er mørk. Den kommer fra det mørke tema vi kalder i vores _8_CDIOFinal.gwt.xml file.



Udover den mørke baggrund er der 3 andre farver i Deltabaren (blå, rødlig og grå)

```
.layoutPanel0{  
    border-style: solid;  
    border-color: #b5b5b5;  
}  
.layoutPanel1{  
    border-style: solid;  
    border-color: #0096fe;  
}  
.layoutPanel2{  
    border-style: solid;  
    border-color: #fb428e;  
}
```

.layoutPanelx er de 3 forskellige klasser oprette til at lave styles inde i.
border-style: solid; definer af panelet skal være tyk og gennemfarvet.
border-color: #5b5b5; er den grå kant i vores layoutPanel.

#5b5b5 er farvekoderne skrevet i hex. Der findes mange farvepaletter på nettet som kan bruges til at finde farven i hex, dog har Eclipse også indbygget en farvepalet som gør det muligt at finde den farve man skal bruge.

Farve temaet giver fx mulighed for man få en default setting fra GWT.

Her findes der nogle metoder der kan bruges til at overskrive de prædefineret settings som GWT sætter.

!important bruges blandt andet til at sætte denne kommando skal overskrive alle de andre eksisterende. Som her i H2 hvor den skal overskrive text-shadow.

```
.H2 td{
    font-size: 2em;
    color: #000000;
    margin: 0px 0px 0px;
    border-width: 1px;
    border-style: solid;
    font-size: 100%;
    font-style: normal;
    text-shadow: #000 0px 0px 0 !important;
}
```

Grunden til der står .H2 td, er fordi den bruger klassen .H2 til at overskrive td med. Dette gør vi for at kunne overskrive den grå lyse farve som var svær at læse i vores tabel inde i DeltaWeightView så det stod med sort.

MySQL

MySQL er et Rational Database Management Program, der bruger programmeringssproget SQL, Structured Query Language. Det bliver brugt til at styre databaser, som er en samling af forskellige matricer kaldet tabeller med tilhørende data. På billedet for neden kan man se en liste over tabellerne i en database og hvordan en matrix kan se ud, når man læser dataen.

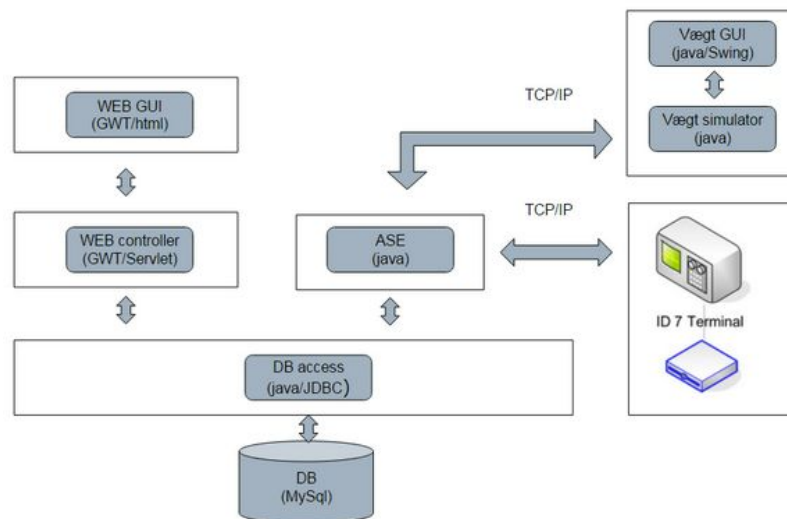
Tables_in_18_cdio_final					
batch					
coins					
condiments					
fruits					
operatoer					

opr_id	opr_navn	ini	cpr	password	aktiv
1	Hej Med Dig	HMD	234897-2342	hello2u	1
2	Thomas Liljegren	TLi	1702921745	1234567	1
3	Luigi Cool	LC	090990-9009	nael101	1
4	Don Juan	DJu	234567-7891	iloveyou	1
5	Kofi Anan	KAn	6969696969	yolo yolo	1
6	Anders Fogh	AFo	4817824293	annemett	1
7	Britney Spears	BSp	1029402949	balis1if	1
8	Barack Obama	BOb	1234539834	facebook	1
9	Lars Larsen	LLa	0203832233	password	1
10	Larsen Lars	LaL	0209231823	password	1
11	Tester Test	TeT	1234560123	1111111	1

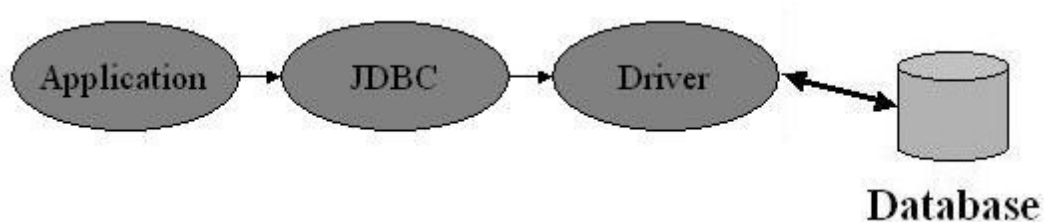
MySQL kører primært uden brug af GUI tools, og for at køre MySQL til at finde eller ændre i data i en database, skal en bruger normalt gennem sin terminal for at komme ind i databasen. Hvis man har en database, hvor flere brugere har forskellige operativsystemer,

JDBC

JDBC, Java Database Connectivity, er et API for programmeringssproget Java og er bindeleddet mellem vores Java applikation og MySQL som indeholder vores database. Dette bruges, da man med JDBC kan etablere forbindelse og sende SQL statements til næsten alle databaser.



For at kunne lave dette samarbejde mellem Java og MySQL, skal der bruges en JDBC driver, som er en software komponent, der gør det muligt for en Java applikation at kommunikere og samarbejde med en database.



Billedet taget fra: <http://www.wideskills.com/jdbc-tutorial/introduction-to-jdbc>

Java-applikationen sender et SQL statement til JDBC, hvorefter JDBC loader driveren, der laver en for en forbindelse til databasen og implementerer protokollen, som overfører en forespørgsel og et resultat mellem klienten og databasen.

Opbygning af projekt

Et GWT projekt er bygget op i en client, en server og en shared del. Klienten skal have en UI-del og en service-del, der snakker med serveren. På serversiden ligger det meste af den eventuelle logik. I dette tilfælde ligger alle vores DAOs her.

Shared pakken indeholder kode der skal kendes af både server og client. Det er blandt andet her brugerdefinerede Exceptions og DTOs lægges.

Client

I pakken client.ui, kan følgende klasser findes

- **AddView** - bruges til at tilføje elementer til vores database, eksempelvis en ny operatør.
- **DeltaBar** - bruges i DeltaWeightView til at vise/visualisere belastningen på vægten, og derudover viser den vejningens tolerance.
- **DeltaWeightView** - bruges sammen med DeltaBar, til at bruge en specifik afvejning på en given genstand.
- **EditView** - bruges til at ændre eksempelvis operatører (der kan her ændres navn, ID, cpr-nummer osv.)
- **ListView** - bruges til at vise listen af operatører, mønter og batch. Det er også her EditView og AddView bruges.
- **Login** - bruges til login-siden.
- **MainView** - fungerer som broen mellem kode og visualisering - det er gennem MainView, at alle andre views bliver åbnet når en given knap bliver aktiveret.
- **MenuView** - tilføjer knapper, således at views kan blive åbnet. Dette medfører en let og overskuelig navigation på siden.
- **StykWeight** - tillader brugeren at veje en mængde af eksempelvis mønter, hvorefter der vises hvor mange af en vis genstand, der ligger på vægten.
- **WeightView** - bruges som et simpelt afvejnings-display, hvor brugeren af mulighed for at se belastningen på vægten, samt at tarére denne.

UI

Vi har i dette projekt oprettet views til alt grænseflade. Det er bygget op således, vi har et MainView, som er UI-controller for de resterende views. MainView indeholder metoder til at åbne alle vores views, fx

```
public void openDeltaWeightView() throws Exception {
    content.clear();
    DeltaWeightView dwView = new DeltaWeightView(this);
    content.add(dwView);
    aPanel.add(content);
    aPanel.setWidgetPosition(content, Window.getClientWidth()
        /8, Window.getClientHeight()/8);
}
```

Alle de andre views, end MenuView, kan kalde udvalgte metoder fra MainView ved at gøre brug af et callback. Eksempelvis kan vores DeltaWeightView gøre brug af MainViews IASEServiceAsync, og IBatchServiceAsync, ved deklarerer følgende interface i DeltaWeightView og implementere det i MainView.

```
public interface Callback {  
    IASEServiceAsync getASEService();  
    IBatchServiceAsync getBatchService();  
}
```

Service

I pakken client.service, finder man interfaces til serverens metoder og deres asynkrone modparter. Hvert service interface skal implementeres af den korresponderende service på server siden, så metoderne er tvunget til at blive implementeret.

Server

Server delen af vores GWT projekt indeholder implementeringen af alle services på klient siden. For god ordens skyld indeholder server siden også egne interfaces til hver af implementeringerne. Derudover er der også en Connector klasse til at skabe forbindelse til databasen gennem JDBC driveren.

Asynchronous I/O

For asynkron I/O til serversiden gøres der i GWT brug af klassen AsyncCallback<T>. Asynkron, eller non-blocking I/O, gør en applikation i stand til at udføre andre processer før der er kommet svar tilbage fra forespørgslen. Når der oprettes et nyt AsyncCallback<T>, skal der implementeres metoder til succesfulde og fejlede forespørgsler. Det ser således ud:

```
service.someMethod(new AsyncCallback<T>() {  
    @Override  
    public void onFailure(Throwable caught) {  
        // TODO  
    }  
  
    public void onSuccess(T result) {  
        // TODO  
    }  
});
```

Hvis der kastes en exception af serveren returneres den over forbindelsen og behandles af onFailure metoden. Hvis kaldet går som det skal, kan resultatet behandles i onSuccess.

For hver service skal der være en synkront interface IService, og et asynkront interface IServiceAsync. Begge disse skal indeholde samme metoder, hvor returtypen i det asynkrone interface, skal defineres som T i et new AsyncCallback<T>.

Nedenfor ses interfaces på klientsiden til vores Afvejnings Styringsenhed Enhed, som ligger på serversiden. Alle Async metoder har eksplicit returtype void.

```
public interface IASEService extends RemoteService{
    void setBrutto(double brutto);
    void connect() throws IOException;
    double getSWeight() throws IOException;
    double getSIWeight() throws IOException, DAException;
    void tara()throws Exception;
    void run() throws IOException, NumberFormatException;
    void changeSocket(String Ip, int port) throws IOException;
}

public interface IASEServiceAsync {
    void connect(AsyncCallback<Void> callback);
    void getSWeight(AsyncCallback<Double> callback);
    void getSIWeight(AsyncCallback<Double> callback);
    void tara(AsyncCallback<Void> callback);
    void run(AsyncCallback<Void> callback);
    void changeSocket(String Ip, int port, AsyncCallback<Void> callback);
    void setBrutto(double brutto, AsyncCallback<Void> callback);
}
```

Delta Bar

Ideen med delta baren, er hurtigt at kunne se en grafisk repræsentation for, om man overholder tolerancen for en given afvejning af et batch. En afvejningsbatch og dens tolerance findes i databasen i vores batch table.

```
mysql> select * from batch;
+-----+-----+-----+-----+-----+
| batch_id | raavare_id | raavare | batchweight | tolerance |
+-----+-----+-----+-----+-----+
| 1 | 1 | tomat | 1 | 0.05 |
| 2 | 1 | tomat | 5 | 0.05 |
| 3 | 2 | løg | 2.5 | 0.05 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Hvis batch med batch_id = 1 vælges, vil batchweight(vægt der sigtes efter) være 1,0 kg. Tolerancevægten udregnes som $tolerance \cdot batchweight$, og grænserne på baren sættes til 3 gange tolerancen. Altså hvis tolerancen er 5% af 1,0 kg, sættes barens grænser, a og b, til

$$\pm 3 \cdot 5\% \cdot 1,0 \text{ kg} = 0,85 \text{ kg} \vee 1,15 \text{ kg}.$$

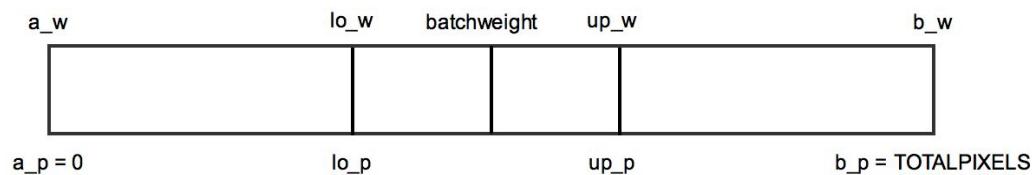


udsnit af delta bar fra vores web applikation

vægt indikatoren, upper og lower bound for tolerancen indikeres at to blå og en rød streg, repræsenteret af LayoutPanels, ligesom selve delta baren.

Placering

For at placere vægt indikatoren, skal vægt inputtet i kg, omregnes til en position i pixels. Hertil skal der bruges et forhold mellem pixels og kg. Herunder ses en beskrivelse af delta baren, hvor $_w$ betyder målt i kg og $_p$ betyder målt i pixels.



$$pxPrUnit = \frac{TOTALPIXELS}{(b_w - a_w)}$$

Derefter skal vi først definere den øvre og nedre tolerance målt i kg.

$$lo_w = batchweight - (batchweight \cdot tolerance)$$

$$up_w = batchweight + (batchweight \cdot tolerance)$$

Da indikatoren kun skal rykkes for hver vægtenhed den kommer over a , kan tolerance grænserne målt i pixels defineres således

$$lo_p = (lo_w - a_w) \cdot pxPrUnit$$

$$up_p = (up_w - a_w) \cdot pxPrUnit$$

Hvis den aktuelle belastning på vægten er under den nedre grænse på delta baren, skal vægt indikatoren blot blive på grænsen. Nedenfor ses et udsnit af koden.

```
//hvis vægten er under barens nedre grænse, står viseren blot på grænsen.
if(w_input < a_w) weightIndicator = 0.0;

//det samme hvis den er over den være grænse
else if(w_input > b_w) weightIndicator = TOTALPIXELS;

//ellers viser den nuværende vægt
else weightIndicator = (w_input - a_w) * pxPrUnit;
```

Løbende opdatering

For at gøre baren responsiv, skal den opdateres løbende. Det gøres i vores tilfælde ved at lave et rekursivt kald af `getSIData` hver gang vi får et svar fra vægten. Både ved fejl og succes.

```
private void getSIData(final Callback c, final double bw, final double tol) {
    c.getASEService().getSIWeight(new AsyncCallback<Double>(){
        @Override
        public void onFailure(Throwable caught) {
            if(caught.getMessage().equals("Weight Overload")) {
                SIDataBox.setText("N/A");
                getSIData(c,bw,tol);
            }else{
                Window.alert("Error accesing weight" + caught.getMessage());
            }
        }
        @Override
        public void onSuccess(Double result) {
            SIDataBox.setText(Double.toString(result));
            dbar.deltaBarData2(result, bw, tol);
            getSIData(c,bw,tol);
        }
    });
}
```

Ved fejl ser vi om fejlen er at vægten har fået for meget vægt. Denne fanger vi ved en if sætning og derefter opdaterer vi display i GWT med et N/A eller "not available" da der ikke er et gyldig værdi at vise.

Derefter checker den igen og igen indtil at der ligger en vægt under maks som derefter vises.

Denne opsætning virker kun fordi vi kan bruge AsyncCallback, dette sørger for at vores system ikke sætter sig ned og venter på et svar. Dette gør den ved at give besked til browseren om at vente på en besked fra serveren, mens browseren venter kan javascript køre videre indtil den får et svar fra serveren som derefter bliver eksekveret.

Grunden til at vi er nødt til at bruge asynchronous metoder er fordi javascript er single threaded og dermed ikke kan udføre 2 eller flere opgaver på samme tid.

Intern adgang vs Internettet

En af problemstillingerne i dette projekt, er at lave et webbaseret, platform uafhængigt interface til vores netværkstilsluttede vægt fra Mettler Toledo.

En anden ér, at vores applikation skal kunne tilgå vægten fra internettet - altså ikke på et lokalt netværk. Dette kan være med til at give betydeligt længere svartider, især da vores HUAWEl access point kører på mobilnettet.

LoginView

LoginView er en vigtig egenskab som er en del af vores program. Ved hjælp af implementationen af LoginView skal brugeren logge ind, med sit eget brugernavn og password. Som det ses i koden herunder, er LoginView bygget op ved hjælp af vertikale paneler, hvor User ID og password er tekstbokse. Derudover er der et label hver for henholdsvis User ID og password, som indikerer hver deres felt.

```
Button btnOne = new Button("Submit", new ClickHandler() {
@Override
public void onClick(ClickEvent event) {
    try {

        c.getService().loginVerify(Integer.parseInt(userID.getText()), password.getText(),
        new AsyncCallback<OperatoerDTO>() {

            @Override
            public void onFailure(Throwable caught) {
                Window.alert("Server fejl! " + caught.getMessage());
                userID.setText("");
                password.setText("");
            }
            @Override
            public void onSuccess(OperatoerDTO data) {

                if(data.getOprId() >= 4){
                    c.loginSucces(data);
                }

            }
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
});

vPanel.add(btnOne);
btnOne.addStyleName("btnOne");
vPanel.setBorderWidth(2);
btnOne.setPixelSize(160, 30);

}
```

Ovenfor ses koden til "Submit"-knappen. I første linje oprettes knappen, og der skal derefter tilføjes en ny ClickHandler. For at knappen skal virke, skal der tilføjes et onClick-event, som verificerer brugernavnet og kodeordet. I dette tilfælde henter den to metoder som er `userID.getText()` og `password.getText()` ved hjælp af et callback. Der er efterfølgende en `onFailure()` og en `Success()`, som determinerer hvad der sker efter onClick-eventet. Opstår der en fejl, bliver fejlen catchet, og denne bliver vist både i en pop-up og i konsollen.

Sker der derimod ikke en fejl, følger man on success scenariet, hvor der er et tjek for om, bruger id er lig med eller større end 4. Er dette tilfældet, får brugeren lov til at logge ind. Til sidst men ikke mindst er det vigtigt, at tilføje mest muligt funktionalitet til programmet. Som der ses i slutningen er koden, er det tilføjet en knap, som man herefter kan positionere og fin placere. Først tilføjes knappen til det vertikale panel, hvorefter størrelsen bliver sat, og til sidst sættes en omkreds omkring panelet, for kosmetikkens skyld.

WeightView

I WeightView har man muligheden for, at se den nuværende belastning på vægts fysiske display. Dette sker ved at sende "S"-kommandoen, som først returnerer vægten af genstanden, når selve vægten er blevet stabiliseret.

Det er bygget bygget op ved brug af to paneler, et vertikalt og et horisontalt. Derudover bruges der to knapper, hhv. "Get weight" og "Tara", og et label der viser belastningen på vægten.

```
Button getWeightButton = new Button("Get weight", new ClickHandler(){
    @Override
    public void onClick(ClickEvent event){
        try {
            c.getASEService().getSWeight(new AsyncCallback<Double>(){
                @Override
                public void onFailure(Throwable caught) {
                    Window.alert("An error occurred: " + caught.getMessage());
                }
                @Override
                public void onSuccess(Double result) {
                    weightLabel.setText("Netto: " + result + " kg");
                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

hPanel1.add(getWeightButton);
getWeightButton.setPixelSize( 105, 30);
hPanel1.setBorderWidth(3);
```

Ovenfor ses koden til "Get weight"-knapen. I første linje oprettes knappen, og der skal derefter tilføjes en ny ClickHandler. For at knappen skal virke, skal der tilføjes et onClick-event, som i dette tilfælde henter metoden getSWeight() vha et callback. Der er efterfølgende en onFailure() og en onSuccess(), som determinerer hvad der sker efter onClick-eventet. Opstår der en fejl, bliver fejlen catchet, og denne bliver vist både i en pop-up og i konsollen.

Sker der derimod ikke en fejl, bliver svaret sat på det label, som er nævnt tidligere.

Når knappens funktionalitet er blevet tilføjet, kan man herefter positionere den, som vist i de tre sidste linjer af koden. Først tilføjes knappen til det horisontale panel, størrelsen bliver sat, og der sættes en omkreds omkring panelet, for kosmetikkens skyld.

ASE

Forbindelse til vægten

For at forbinde til vægten bruger vi følgende metode, så vi kan sende og modtage data fra vægten.

Vi sætter en socket med en host og port op og derefter sætter vi en inputStream og en outputStream op, så vi kan modtage og sende data ved brug af disse streams

```
@Override
public void connect() throws UnknownHostException, IOException {

    if(sock!=null) return;

    sock = new Socket(host, port);
    in = new BufferedReader(new InputStreamReader(sock.getInputStream()));
    out = new DataOutputStream(sock.getOutputStream());

}
```

Vægt kommunikation

I dette projekt bruger vi to af de kommandoer som vægten kender - vi bruger "S", som giver os en vægt når den er stabil, og SI som giver os det der måles lige nu.

S kommando

```
@Override
public double getSWeight() throws IOException {
    out.writeBytes("S\r\n");
    String response = in.readLine();
    brutto = Double.parseDouble(response.substring(3,response.length()-2).trim());
    return returnNetto();
}
```

Ovenstående kode sender beskeden "S" til vægten samt en afslutning af kommandoen som vægten forventer.

Når kommandoen bliver sendt, læses svaret med en `BufferedReader`, og svaret bliver parset til en `double`. I sidste linje bliver `returnNetto()` returneret, som er en metode defineret ved

```
public double returnNetto() {
    netto = (brutto - tara);
    return netto;
}
```

`returnNetto` bruger vi til at sikre at sikre at tara fratrækkes fra brutto før denne bliver sendt videre til displayet.

Tara

```
private double brutto;  
private double tara;  
private double netto;
```

Metoden `tara()` fungerer simpelt nok ved, at `tara` bliver gemt som hvad end den nuværende brutto er.

```
@Override  
public void tara() throws Exception {  
    tara = brutto;  
}
```

SI kommando

```
public double getSIWeight() throws IOException, DAException {  
    out.writeBytes("SI\r\n");  
    String response = in.readLine();  
    if(response.startsWith("ES")) {  
        return -1;  
    }else if(response.startsWith("S +")){  
        throw new DAException("Weight Overload");  
    }  
    double weight = Double.parseDouble(response.substring(3,response.length()-2).trim());  
    return weight;  
}
```

Den ovenstående metode fungerer meget ligesom `getWeight()` metoden, dog med et par forskelle, vi checker efter fejl beskeder, først ser vi efter standard fejlbeskeden "ES", dette har vi gjort for at sikre at vi kan modtage fejlbeskeder og behandle disse, derefter checker vi om fejlbeskeden er "S +", denne fejlbesked bruger vi til at kaste en exception, ved at kaste en exception sådan, kan vi sikre at når vægten bliver overbelastet så kan vi fange denne og behandle den som vi ønsker det, dette bliver brugt i `deltaweight`.

Database kommunikation

I dette afsnit vil vi gennemgå hvordan vi kontakter mysql serveren, vi vil ikke gennemgå hver af DAO'erne vi bruger i projektet da de fungerer meget ens, vi vil dog gennemgå `MySQLOperatoerDAO` da denne er den mest omfattende og indeholder metoder som ligner dem som de andre DAO'er bruger.

Ved opstart af serveren vil `MySQLOperatoerDAO` lave en opstart af connector klassen. Connector står for at forbinde vores server kode med MySQL serveren, connector bruger data fra constant klassen og derefter sættes den en forbindelse op ved brug af JDBC.

De mest interessante metoder i `MySQLOperatoerDAO` som bruger andet end MySQL forespørgsler er metoderne `getOperatoerList` og `loginVerify`.

getOperatoerList

Som det ses herunder virker getOperatoerList ved at den connecter til databasen, hvor den så laver en SQL efterspørgsel. Hvis forbindelsen skabes og kommandoen udføres korrekt, tilføjer den operatoer id, navn, initialer, cpr, kodeord, samt om brugeren er aktiv eller ej til en liste. Når alle disse informationer så er listede, returnerer koden herefter listen.

```
public List<OperatoerDTO> getOperatoerList() throws DALException {
    List<OperatoerDTO> list = new ArrayList<OperatoerDTO>();
    ResultSet rs = Connector.doQuery("SELECT * FROM operatoer");
    try{
        while (rs.next())
        {
            list.add(new OperatoerDTO(rs.getInt("opr_id"), rs.getString("opr_navn"),
            rs.getString("ini"),rs.getString("cpr"), rs.getString("password"),
            rs.getInt("aktiv")));
        }
    }
    catch (SQLException e) { throw new DALException(e.getMessage()); }
    return list;
}
```

loginVerify

Login verify virker ved at modtage operatoer ID, med denne ID finder vi en given operatoer, derefter checker vi om metodens password også passer med denne operatoers password. Derefter er der 3 cases der kan ske, hvis oplysningerne passer så checker vi om den fundne operatoer har en aktiv værdi over 0, hvis denne er god så returnerer vi operatoeren, hvis ikke, laver vi et popup som siger til brugeren at den givne bruger ikke er aktiv. Det sidste case er hvis koden og ID ikke passer sammen, hvis dette sker laves et popup og fortæller brugeren at koden og password passer ikke.

UnitWeight

UnitWeight er vores vægt, der skal bruges til at udregne, hvor mange enheder vi har af et givent produkt. Dette gøres ved at vi i en database har en stykvægt for vores produkter.

value	weight	tolerance
0.5	0.0043	0.01
1	0.0036	0.01
2	0.0059	0.01
5	0.0092	0.01
10	0.0071	0.01
20	0.0093	0.01

Ved en afvejning skal man i vores interface vælge hvilket type af produkt man vil afveje. Der er muligheden for at veje mønter (coins), frugter (fruits) og krydderier (condiments).

Når man trykker på et af disse knapper, kommer alle produkterne fra for typen.

Ved det tryk, vil man kunne se, hvor mange enheder man har af sit produkt. Ved fx. at trykke på “20”, kan man finde ud af hvor mange 20’ere man har. Det virker kun, hvis det man afvejer, er det eneste man afvejer. Vægten vil ikke kunne give et korrekt resultat, hvis man vælger at afveje 20’ere, men der på vægten er andet end 20’ere.

Vægten på vores produkter er ikke nogen vi selv har afvejet, men fundet. Ved mønter har vi brugt Nationalbankens data³. Ved frugter⁴ har vi valgt at inddele “stykkerne” i portioner, da forskellen mellem to ens frugter kan variere for meget. Det samme problem gælder for krydderierne⁵. Så derfor har vi valgt at inddele det i teskeer.

³ http://www.nationalbanken.dk/da/sedlerogmoenter/danske_moenter/Sider/default.aspx

⁴ <http://www.goldendrop.com.au/mangoes.html/>(Mango vægt)
http://www.kalorie-let.dk/maal_og_vaegt.php(banan, pære, æble og kiwi)

⁵ <http://www.arla.dk/opskrifter/>

Afvejning

Afvejningen er bygget op ved brug af et FlexTable med to tables. I det ene kan der findes knapper og i det andet er der et vertikalt panel. Der er tre knapper, der åbner hver deres cellView, ved brug af en ClickHandler.

```
btnCoins.addClickHandler(new ClickHandler(){
    public void onClick(ClickEvent event) {
        try {c.getASEService().getSWeight(new AsyncCallback<Double>(){
            @Override
            public void onFailure(Throwable caught) {
                Window.alert("An error occurred: " + caught.getMessage());
            }
            @Override
            public void onSuccess(Double result) {
                wText.setText("Netto: " + result + " kg");
            }
        });
        coinCellView(c);
        ft.setHTML(0, 0, "Choose one of the coins");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

Når man har klikket på hvilken kategori man vil afveje, får man af vide, hvad der står på vægten. Efterfølgende vil der blive vist tabellen for den kategori man har valgt. Dette bliver vist som et CellTable under tabellen med knapperne og vertikal panelet. For at gøre det mere overskueligt, bliver CellTablet kombineret med tablet til venstre for den. På den måde, er der ikke nogen tomme tabeller i bunden af FlexTablet. Dette bliver gjort med HasHorizontalAlignment

```
cellFormatter.setHorizontalAlignment(0,0, HasHorizontalAlignment.ALIGN_LEFT);
ft.setWidget(5, 0, coinTable);
cellFormatter.setColSpan(5, 0, 2);
```

Ved at klikke på en enhed på tabellen, kan man måle antallet af enheder der er for den afmålte vægt.

```
if (selected != null) {
    try {c.getASEService().getSWeight(new AsyncCallback<Double>(){
        @Override
        public void onFailure(Throwable caught) {
            Window.alert("An error occurred: " + caught.getMessage());
        }
        @Override
        public void onSuccess(Double result) {
            wText.setText("Netto: " + result + " kg");
            double tWeight = (result)/(selected.getWeightPerUnit());
        }
    });
}
```

```

        stkText.setText(""+tWeight);
        stkLabel.setText("# of coins");
    }
});
}

```

Test

I følgende afsnit vil vi gennemgå vores tests. Testene vi udfører er JUnit udført på server-siden af vores projekt. Udover JUnit har vi også udført brugertests af GWT og det UI, som vises her.

JUnit

BatchDAOTest:

Test	Succes	Fejl
getBatch	X	
getBatchList	X	

CoinDAOTest

Test	Succes	Fejl
getCoinInfo	X	
GetCoinList	X	

CondimentsDAOTest

Test	Succes	Fejl
getCondimentsInfo	X	
getCondimentsList	X	

FruitDAOTest

Test	Succes	Fejl
getFruitInfo	X	
getFruitList	X	

ASETTest

Test	Success	Fejl
Connect()	X	
getSWeight()	X	
getSIWeight()	X	
Tara()	X	
returnNetto()	X	

For at den ovenstående test kører skal vægten være sat op rigtigt, hvis ikke vil alle test resultere i fejl da der ikke kunne oprettes forbindelse til vægten. Evt kunne man også bruge en vægt simulator.

MetaDAOTest

Test	Success	Fejl
getTables()	X	

MySQLOperatoerDAOTest

Test	Succes	Fejl
getOperatoer	X	
createOperatoer	X	
updateOperatoer	X	
getOperatoerList	X	
loginVerify	X	
deleteOperatoer	X	

Brugertest

For at sikre at den brugerflade vi har sat op er til at bruge og de features som vises virker har vi sat følgende brugertest op. Disse tests vil bliver udført i den skrevne rækkefølge samt dokumenteret med billeder når nødvendigt.


Login

Vi har valgt at bruge brugeren Don Juan med følgende login oplysning.

UserID: 4

Password: iloveyou

Efterfølgende vises det den næste side hvor menuen hvor er synlig, der er ikke valgt nogen af de muligheder som er i menuen og dermed er vinduet tomt.

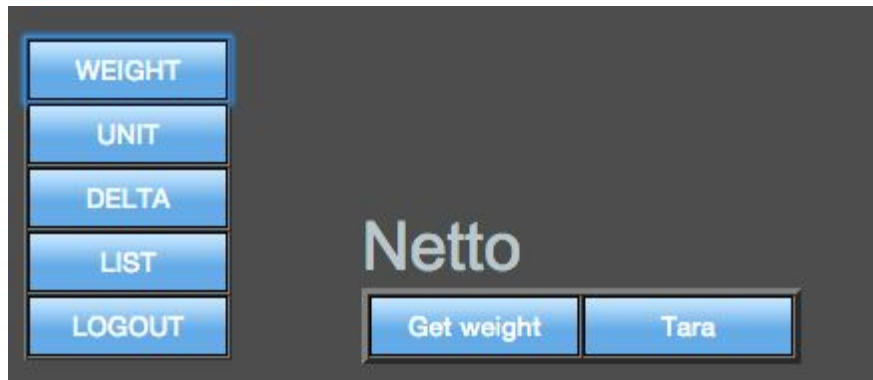


A login form with two input fields. The first field is labeled 'User ID' and contains the number '4'. The second field is labeled 'Password' and contains seven dots. Below the fields is a blue button labeled 'Submit'.



Weight

ved klik på weight vil billede skifte og det simple vægt system bliver synligt.



Her ses at der er et display hvor netto vægten vises samt 2 knapper med hver deres funktion.

Get weight henter den nuværende vægt som er stabil på vægten og sætter den ind ved siden af Netto.

Tara gør at serveren tager den nuværende netto vægt og logger den ned på serveren, alle fremtidige "Get weight" tryk vil resultere i den nuværende vægt af objekt på vægten fra trukket den tara som der blev sat ved trykket af Tara.



Unit

Unit giver mulighed for at vælge et produkt vi gerne vil arbejde med, og ud fra denne vil vi kunne beregne antallet ud af dette produkt ud fra netto vægten.

Fra venstre ses hvordan UI ser ud uden valg af produkt, efterfølgende ses ved valg af coins og vejning at der er fem 1kr mønter.

Unit-weight

Choose one of the buttons

Coins	Weight
Fruit	
Condiments	# of items

Unit-weight

Choose one of the coins

Coins	Weight
Fruit	Netto: 0.018 kg
Condiments	# of coins
	5

Coin Value

0.5
1
2
5
10
20

Delta

Delta menuen giver følgende ved valg.



I dette vindue kan man vælge et produkt, dette produkt vil blive brugt til at sætte grafisk opsætning i delta baren, hvorefter en afvejning indenfor den givne tolerance.

Ved valg af BatchID 2, og ved en at lægge noget på vægten.



Her kan det dog ikke ses at under "SI - WeightData" bliver vægten opdateret løbende mens der lægges objekter på vægten, samt at den røde indikator i deltabaren bevæger sig i forhold til rammen den er givet.

Hvis en bruger sætter end 6 kg på vægten vil displayed ændrer sig til N/A da dette overstiger vægtens maksimale belastning.

0.9

1.1

Product Name	BatchID	Batch weight	SI - WeightData
tomat	1	1	N/A

Batch ID	Raavare	Raavare ID	Batch Weight	Tolerance
1	tomat	1	1	0.05
2	tomat	1	5	0.05
3	løg	2	2.5	0.05

Ved valg af nyt produkt i delta opstår der en bug ved tallene omkring deltabaren, disse tal bliver opdataeret og dermed skubber de til placeringen af baren. Tror vi.

List

List giver mulighed for at se dataene i databasen. Her ses fx ved valg af fruits.

Table name	Fruit Name	Tolerance	Weight Pr Unit
batch	Apple	0.03	0.12
coins	Pear	0.03	0.132
condiments	Banana	0.03	0.14
fruits	Mango	0.03	0.2
operator	Avocado	0.05	0.3
	Kiwi	0.05	0.076

En sekundær feature er at ved valg af operatorer bliver der lavet 2 nye knapper som giver mulighed for at ændre data i den valgte operatorer, eller man kan slette en operatorer.

Table name	Opr ID	name	Initials	opr	Password	active rights ID
batch	1	Hej Med Dig	HMD	234897-2342	hello2u	1
coins	2	Thomas Liljegren	TLi	1702921745	1234567	1
condiments	3	Luigi Cool	LC	090990-9009	naellol	1
fruits	4	Don Juan	DJu	234567-7891	iloveyou	1
operator	5	Kofi Anan	KAn	6969696969	yoloyolo	1
	6	Anders Fogh	AFo	4817824293	annemett	1
	7	Britney Spears	BSp	1029402949	balislif	1
	8	Barack Obama	BOb	1234539834	facebook	1
	9	Lars Larsen	LLa	0203832233	password	1
	10	Larsen Lars	LaL	0209231823	password	1
	11	Tester Test	TeT	1234560123	1111111	1
	12	Peter Petersen	PP	0611921234	PeterErSej	1

Edit Remove

hvis man ikke vælger en operatorer før man trykker på edit eller remove vil der komme en popup med en besked.

Ved valg af en en operatorer og edit vil vinduet skifte til editView som ikke kan vælges fra menuen, denne gør at man kan ændre data i en operatorer.

Vi ændrer Luigi Cool til at hedde Luigi NotCool.

3	Luigi Cool	LC	090990-9009	naellol	1
3	Luigi NotCool	LC	090990-9009	naellol	1

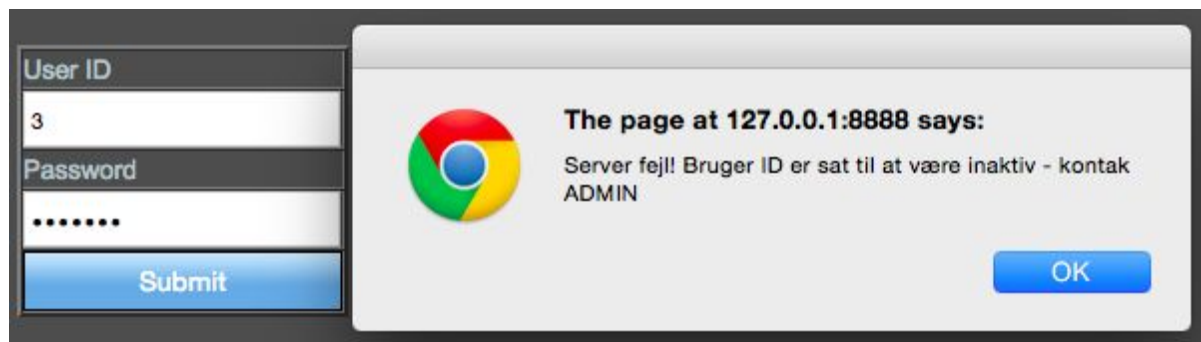
Hvis man vælger at fjerne en bruger ved remove vil dens aktive værdi sættes til 0.

3	Luigi Cool	LC	090990-9009	naellol	0
---	------------	----	-------------	---------	---

Logout

Tilslidst har vi logout knappen, denne logger brugeren ud af systemet så en anden kan logge ind.

Hvis en bruger som er sat til ikke aktiv prøver at logge ind vil der komme en popup som informerer om dette.



Opsætning af accesspoint og vægt

Der tages udgangspunkt i, at routeren er gendannet til fabriksindstillinger!

[Opsætning af accesspoint og vægt](#)

[1. Login på router](#)

[2. Opsætning af indstillinger](#)

[2.1 Opkalds-opsætning](#)

[2.2 DHCP opsætning](#)

[2.3 Opsætning af Virtuel Server \(port-forward\)](#)

[3. Opsætning af vægt](#)

1. Login på router

Når 4G-routeren er blevet tændt, skal du tilkoble din PC/enhed på internettet.

1. Tag den øverste del af routeren, og frakobl den fra den nedre del
2. På undersiden, skal den hvide plade skubbes opad
3. På batteriet sidder en lille sticker med passwordet til WiFi-forbindelsen

Når din enhed er koblet på nettet, skal du tilgå routeren. Åbn din browser, og indtast følgende i adresselinjen

192.164.1.1

og tryk enter. Skærmen burde gerne se således ud



Tryk derefter på "Login", og indtast følgende

Brugernavn: admin

Password: admin

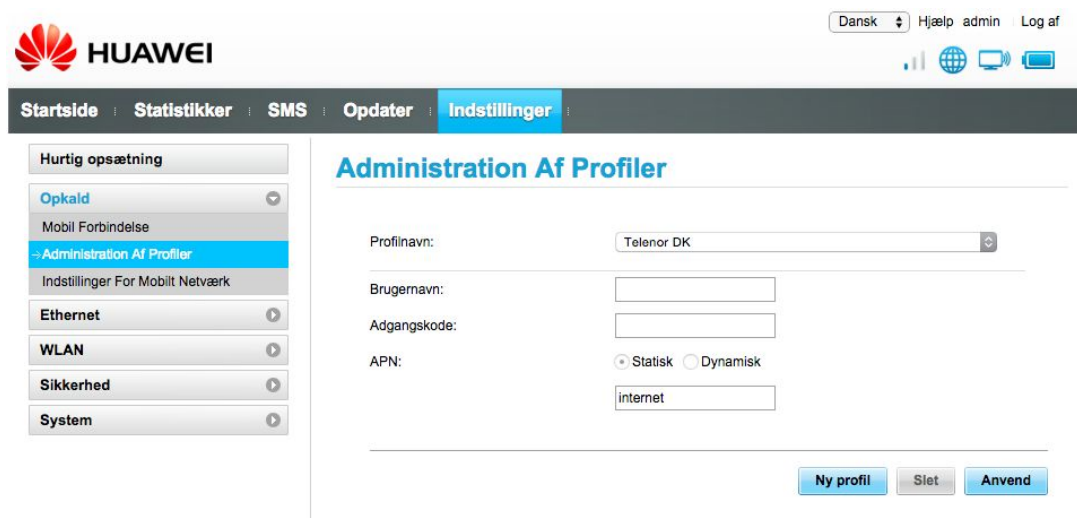
2. Opsætning af indstillinger

Tryk på “Indstillinger”. Skærmen ser således ud



2.1 Opkalds-opsætning

Tryk på “Opkald” i venstre menu, og vælg “Administration af Profiler” i følgende drop-down menu. Skærmen ser således ud



Tryk på “Ny Profil” nede i højre hjørne, og udfyld pop-up på følgende måde:

The screenshot shows a 'Ny profil' (New profile) pop-up form. It has a title bar with a close button (X). The form contains the same fields as the main screen: 'Profilnavn:' with the value 'static(standard)', 'Brugernavn:' with the value 'admin', 'Adgangskode:' with masked characters '*****', and 'APN:' with the value 'static.telenor.dk' and the 'Statisk' radio button selected. At the bottom right, there are two buttons: 'Gem' (Save) and 'Annuller' (Cancel).

Profilnavn: dette felt er ligegyldigt, men vi foreslår ovenstående.

Brugernavn: *admin (behøves ikke at udfyldes)*

Password: *admin (behøves ikke at udfyldes)*

APN: *denne skal sættes til Statisk, og feltet nedenfor fortæller til udbyderen, at der ønskes en statisk IP-adresse, hvilket gør at routeren får en konstant IP-adresse til forbindelse udefra.*

Afslut ved at trykke på “Gem”, efterfulgt af “Anvend”.

2.2 DHCP opsætning

Tryk på “WLAN”, og vælg “DHCP” fra dropdown menuen. Udfyld felterne på følgende måde

Hurtig opsætning

Opkald

Ethernet

WLAN

- Grundlæggende Indstillinger For WLAN
- Avancerede Indstillinger For WLAN
- MAC-filter for WLAN
- DHCP**

Sikkerhed

System

DHCP

IP-adresse: 169.254.2.1

Subnetmaske: 255.255.0.0

DHCP-server: ☒ Aktivér ☐ Deaktiver

Første IP-adresse: 169.254.2.100

Sidste IP-adresse: 169.254.2.200

Leasetid (er) for DHCP: 86400

Anvend

Afslut ved at trykke på “Anvend”.

2.3 Opsætning af Virtuel Server (port-forward)

Tryk på “Sikkerhed”, og vælg “Virtuel Server” fra dropdown menuen. Tryk herefter på “Tilføj” under listen i midten af skærmen, og indtast følgende (markeret med rødt)

Liste Over Virtuelle Servere

Navn	WAN-port	LAN-IP-adresse	LAN-port	Protokol	Status	Indstillinger
RDP	3389	192.168.1.100	3389	TCP/UDP	Tændt	Rediger Slet
weight2	8000	169.254.2.2	8000	TCP/UDP	Tændt	Rediger Slet

Tilføj

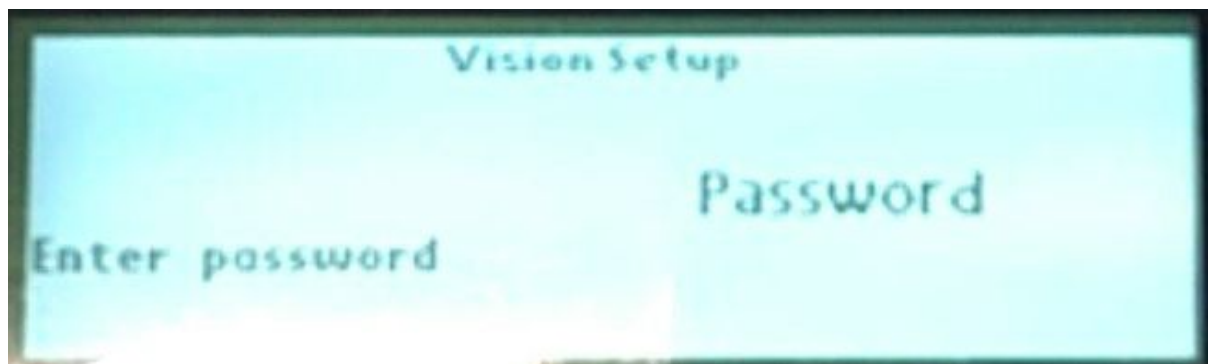
Afslut ved at trykke på “Ok”, efterfulgt af “Anvend”.

3. Opsætning af vægt

Til at starte med, skal du logge ind i opsætningsmenuen. Dette gøres ved, at holde følgende knap nede, indtil displayet ændres



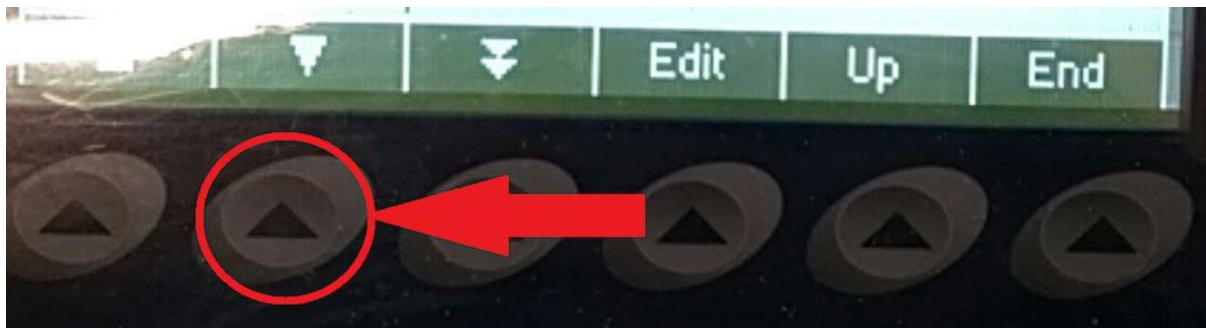
Displayet vil se således ud



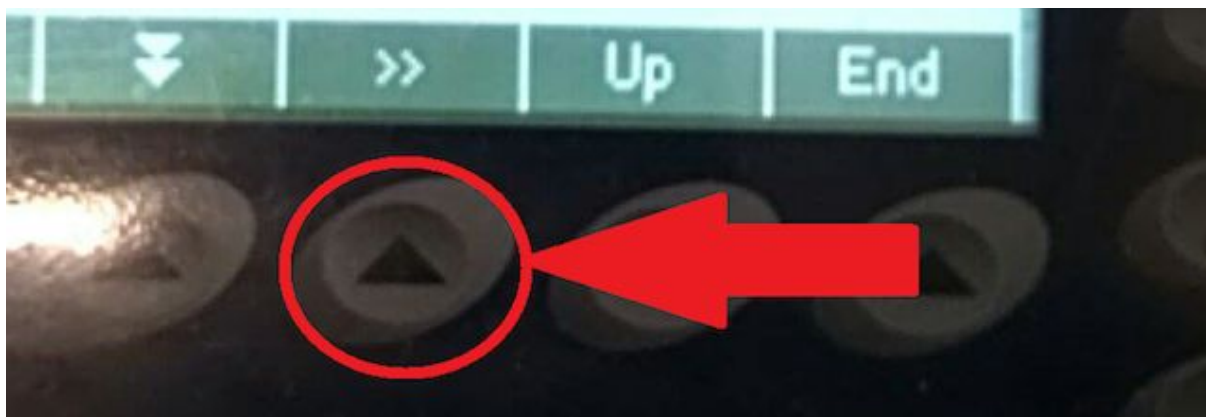
Tryk en enkelt gang på samme knap, og displayet vil se således ud



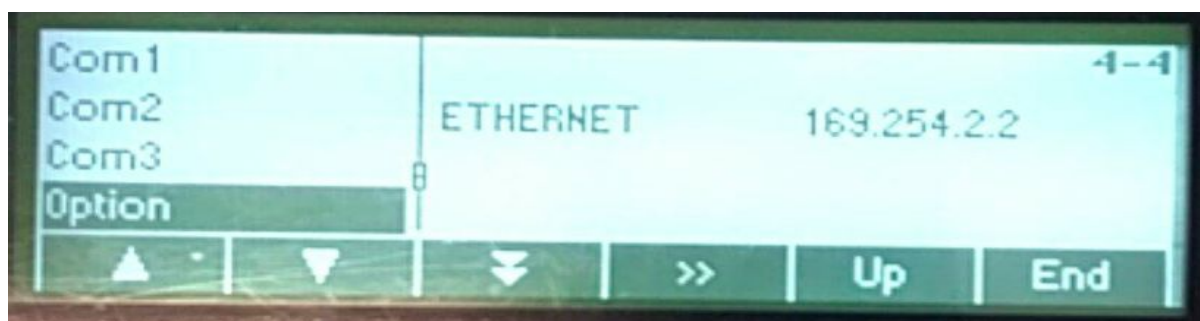
Tryk på følgende knap, indtil "Communication" er markeret (læg mærke til, at knappen som er markeret på billedet, er under "Pil ned" på displayet)



og tryk herefter på (læg mærke til, at knappen som er markeret på billedet er under ">>")



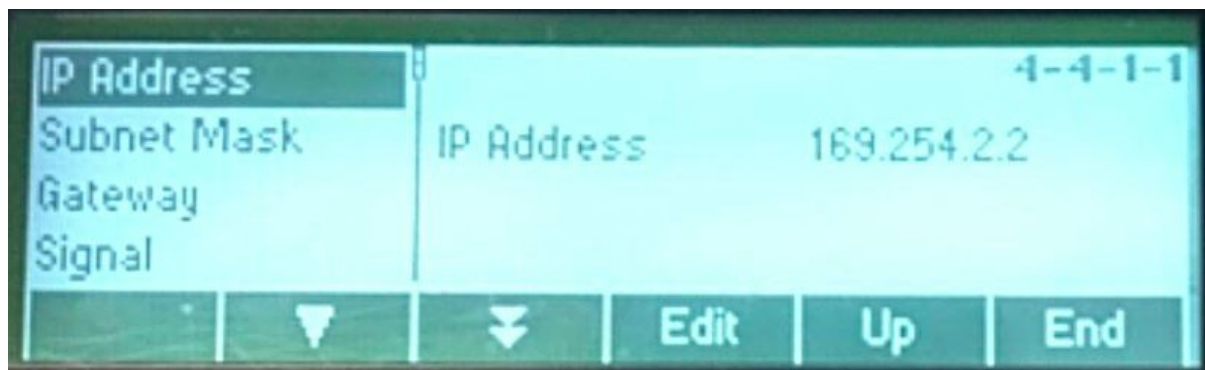
Gentag nu brugen af disse knapper, og vælg følgende menu på displayet



Når "Option" er markeret, ligesom på billedet ovenfor, tryk igen på knappen under ">>" og displayet vil efterfølgende se således ud

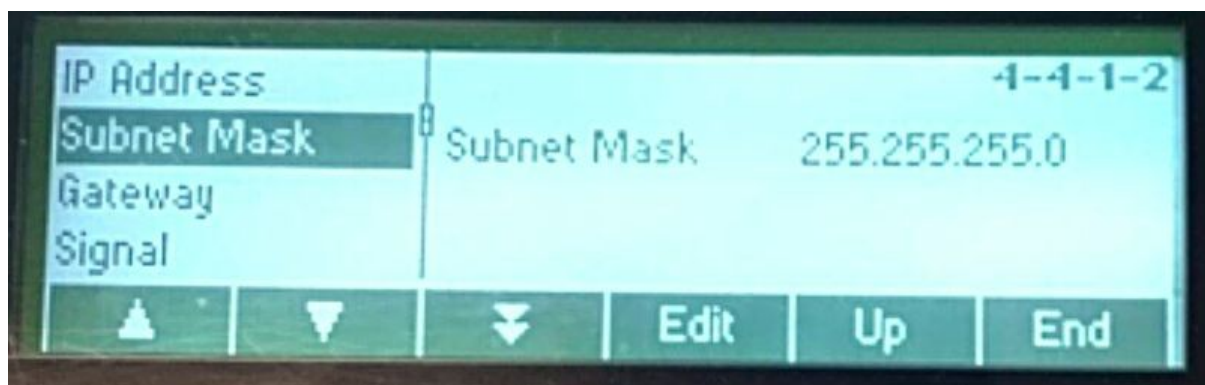


Tryk igen på knappen under ">>", og displayet ser efterfølgende således ud



IP-adressen skal være den samme, som på billedet ovenover. Hvis den ikke er det, tryk på knappen under "Edit", ændr IP-adressen til det ønskede, og save (dette skal under alle omstændigheder være en 169.254.x.x-adresse)

Samme procedure gælder for menuen "Subnet Mask", og denne skal have følgende værdi



Til sidst ændres "Gateway" til 169.254.2.1, med samme procedure som tidligere. Afslut ved at trykke "End" og gem ændringerne.

Konklusion

Vi har i dette projekt sammensat et solidt produkt som opfylder de fleste af de krav som vi har stillet os selv samt dem som opgave oplægget har givet.

Vores produkt er cross platform og burde virke i de browsere som GWT er kompatibel med.

Vi har dog ikke opfyldt at kunne gemme de vægt data vi modtager, dette kunne dog sagtens implementeres enten i databasen eller ved en logning til en fil.

Bilag 1. Time regnskab

Time-regnskab	Gruppe 18							
Dato	Deltager	Design	Impl.	Test	Dok.	Andet	Ialt	
04/06/2015	Nael					1	1	
04/06/2015	Thomas					1	1	
04/06/2015	Peter					1	1	
04/06/2015	Martin					1	1	
04/06/2015	Liban					1	1	
04/06/2015	Paulin					1	1	
04/06/2015	Kenneth					1	1	
		Design	Impl.	Test	Dok.	Andet	Ialt	
05/06/2015	Nael						0	
05/06/2015	Thomas						0	
05/06/2015	Peter						0	
05/06/2015	Martin						0	
05/06/2015	Liban						0	
05/06/2015	Paulin						0	
05/06/2015	Kenneth	1					1	
		Design	Impl.	Test	Dok.	Andet	Ialt	
06/06/2015	Nael						0	
06/06/2015	Thomas						0	
06/06/2015	Peter				0,5		0,5	
06/06/2015	Martin						0	
06/06/2015	Liban				1,5		1,5	
06/06/2015	Paulin						0	

06/06/2015	Kenneth						0	
		Design	Impl.	Test	Dok.	Andet	Ialt	
07/06/2015	Nael				5		5	
07/06/2015	Thomas	0,5			4,5		5	
07/06/2015	Peter	0,5			4,5		5	
07/06/2015	Martin	1			5		6	
07/06/2015	Liban				5		5	
07/06/2015	Paulin				5		5	
07/06/2015	Kenneth						0	
		Design	Impl.	Test	Dok.	Andet	Ialt	
08/06/2015	Nael				4	1	5	
08/06/2015	Thomas				3	2	5	
08/06/2015	Peter	1			5	0,5	6,5	
08/06/2015	Martin	0,5			3	2	5,5	
08/06/2015	Liban				4	1	5	
08/06/2015	Paulin				4	1	5	
08/06/2015	Kenneth	3				1,5	4,5	
		Design	Impl.	Test	Dok.	Andet	Ialt	
09/06/2015	Nael				0,5	4	4,5	
09/06/2015	Thomas						0	
09/06/2015	Peter				2,5	3,5	6	
09/06/2015	Martin				1	2	3	
09/06/2015	Liban	1,5					1,5	
09/06/2015	Paulin				1	1	2	
09/06/2015	Kenneth	1			1	4	6	
		Design	Impl.	Test	Dok.	Andet	Ialt	
10/06/2015	Nael		2			2	4	
10/06/2015	Thomas		2		2	1	5	

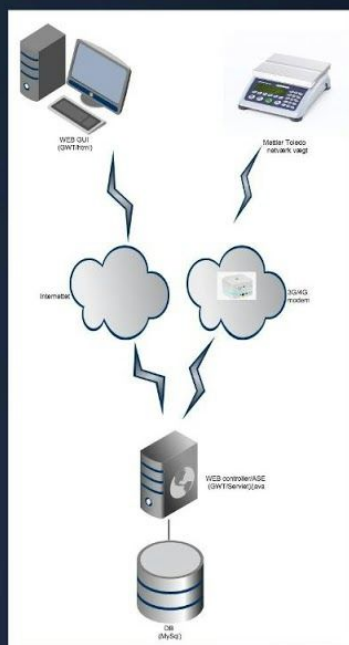
10/06/2015	Peter	1,5			1,5	1	4	
10/06/2015	Martin						0	
10/06/2015	Liban	2			0,5	2	4,5	
10/06/2015	Paulin		2,5		0,5		3	
10/06/2015	Kenneth	5				1	6	
		Design	Impl.	Test	Dok.	Andet	Ialt	
11/06/2015	Nael	3	1			1	5	
11/06/2015	Thomas						0	
11/06/2015	Peter	1	4				5	
11/06/2015	Martin		5				5	
11/06/2015	Liban	1,5	2	1		0,5	5	
11/06/2015	Paulin	3	1				4	
11/06/2015	Kenneth	5					5	
		Design	Impl.	Test	Dok.	Andet	Ialt	
12/06/2015	Nael		1			3	4	
12/06/2015	Thomas		2		1	2	5	
12/06/2015	Peter	2	2,5	0	0	0	4,5	
12/06/2015	Martin		5				5	
12/06/2015	Liban	2	1,5			1	4,5	
12/06/2015	Paulin	4,5	1				5,5	
12/06/2015	Kenneth	0,5	4				4,5	
		Design	Impl.	Test	Dok.	Andet	Ialt	
13/06/2015	Nael						0	
13/06/2015	Thomas		3		2		5	
13/06/2015	Peter	0	0	0	0	0	0	
13/06/2015	Martin		3		2		5	
13/06/2015	Liban						0	
13/06/2015	Paulin						0	

13/06/2015	Kenneth						0	
		Design	Impl.	Test	Dok.	Andet	Ialt	
14/06/2015	Nael						0	
14/06/2015	Thomas						0	
14/06/2015	Peter	0	0	0	0	0	0	
14/06/2015	Martin						0	
14/06/2015	Liban				1		1	
14/06/2015	Paulin						0	
14/06/2015	Kenneth		4				4	
		Design	Impl.	Test	Dok.	Andet	Ialt	
15/06/2015	Nael	2,5	3,5				6,5	
15/06/2015	Thomas		4			2	6	
15/06/2015	Peter	3,5	3,5				7	
15/06/2015	Martin		4	1			5	
15/06/2015	Liban	2	2		2		6	
15/06/2015	Paulin		6				6	
15/06/2015	Kenneth		6				6	
		Design	Impl.	Test	Dok.	Andet	Ialt	
16/06/2015	Nael	1	4				5	
16/06/2015	Thomas		4		1		5	
16/06/2015	Peter	2	4				6	
16/06/2015	Martin	0	0	0	0	0	0	
16/06/2015	Liban	2	2		1		5	
16/06/2015	Paulin	2	3				5	
16/06/2015	Kenneth		5				5	
		Design	Impl.	Test	Dok.	Andet	Ialt	
17/06/2015	Nael	1,5	5				6,5	
17/06/2015	Thomas		3				3	

17/06/2015	Peter	2,5	2,5	1	0	0	6	
17/06/2015	Martin		2		3		5	
17/06/2015	Liban						0	
17/06/2015	Paulin						0	
17/06/2015	Kenneth	0,75	5				5,75	
		Design	Impl.	Test	Dok.	Andet	Ialt	
18/06/2015	Nael						0	
18/06/2015	Thomas		3		3	3	9	
18/06/2015	Peter	0	2,5	2,5	3	0	8	
18/06/2015	Martin	0	2	4	3	0	9	
18/06/2015	Liban	1	1		3	1	6	
18/06/2015	Paulin	0,5	2	2	1		5,5	
18/06/2015	Kenneth	3	0,5		2	3,5	9	
		Design	Impl.	Test	Dok.	Andet	Ialt	
19/06/2015	Nael						0	
19/06/2015	Thomas						0	
19/06/2015	Peter						0	
19/06/2015	Martin						0	
19/06/2015	Liban						0	
19/06/2015	Paulin						0	
19/06/2015	Kenneth						0	
		65,25	126	11,5	92,5	54,5	350,25	

WEB grænseflade til vægt

WEB baseret applikation som kan connecte med en vægt over nettet. Vægten kører på en 3G/4G router som er på mobilnetværket. Dataen gemmes på en MySql server. Dette setup giver mulighed for at kunne være platformsuafhængig da vi bruger Google Web Toolkit(GWT).



```
//weight-button, opens WeightView when pressed
//
Button weightBtn = new Button("WEIGHT", new ClickHandler() {
    @Override
    public void onClick(ClickEvent event) {
        try {
            main.openWeightView();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

//unit/weight-button, opens SelectWeight when pressed
//
Button unitWtBtn = new Button("UNIT", new ClickHandler() {
    @Override
    public void onClick(ClickEvent event) {
        try {
            main.openSelectWeight();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

```
@Handler
public void onWeightClick(ClickEvent event) {
    try {
        main.openWeightView();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Handler
public void onUnitClick(ClickEvent event) {
    try {
        main.openSelectWeight();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Delta-weight

0.0 0.0

Product Name	BatchID	Batch weight	SI - WeightData
Batch ID	Quantity	Quantity ID	Batch Weight
1	tomat	1	0.06
2	tomat	1	0.06
3	ag	2	0.06

Klientens pc kobler sig op til WEB serveren. Serveren connecter så både til vægten og databasen. I databasen tjekkes bruger login for at kunne få adgang til vægten. Når valideringen er udført og adgangen er givet kan klienten følge afvejningen i en browser.

Teknologier der er brugte

- Sockets
- Google Web Toolkit
- MySQL
- Java
- CSS
- JDBC
- TCP/IP

