1 Counting plankton from microscopic images

1.1 The plan

The plan is to make a computer program to help count plankton from microscopic images. A typical microscopic image may look like this (See figure 1). As much as the goal is to get a functional program, the important part will be the documentation describing the structure and possibilities of the program.

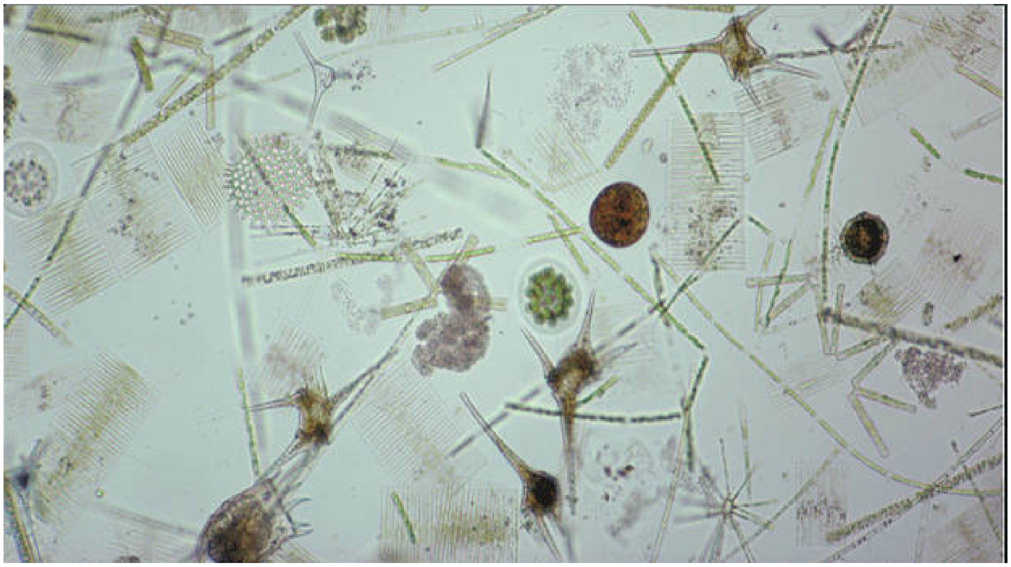
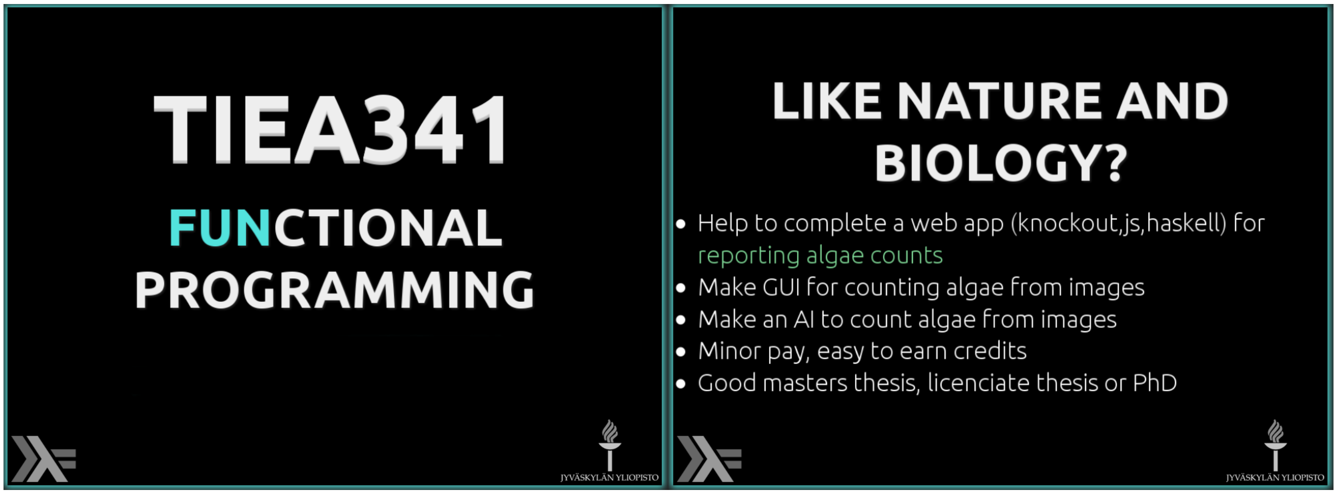


Figure 1: Plankton in microscopic image

1.2 Background

It all started from the course of Functional programming 1 in spring 2017 (See figure 2).



TIEA341

FUNCTIONAL PROGRAMMING

LIKE NATURE AND BIOLOGY?

- Help to complete a web app (knockout,js,haskell) for reporting algae counts
- Make GUI for counting algae from images
- Make an AI to count algae from images
- Minor pay, easy to earn credits
- Good masters thesis, licenciate thesis or PhD

Figure 2: TIEA341 Functional programming 1

2 The program

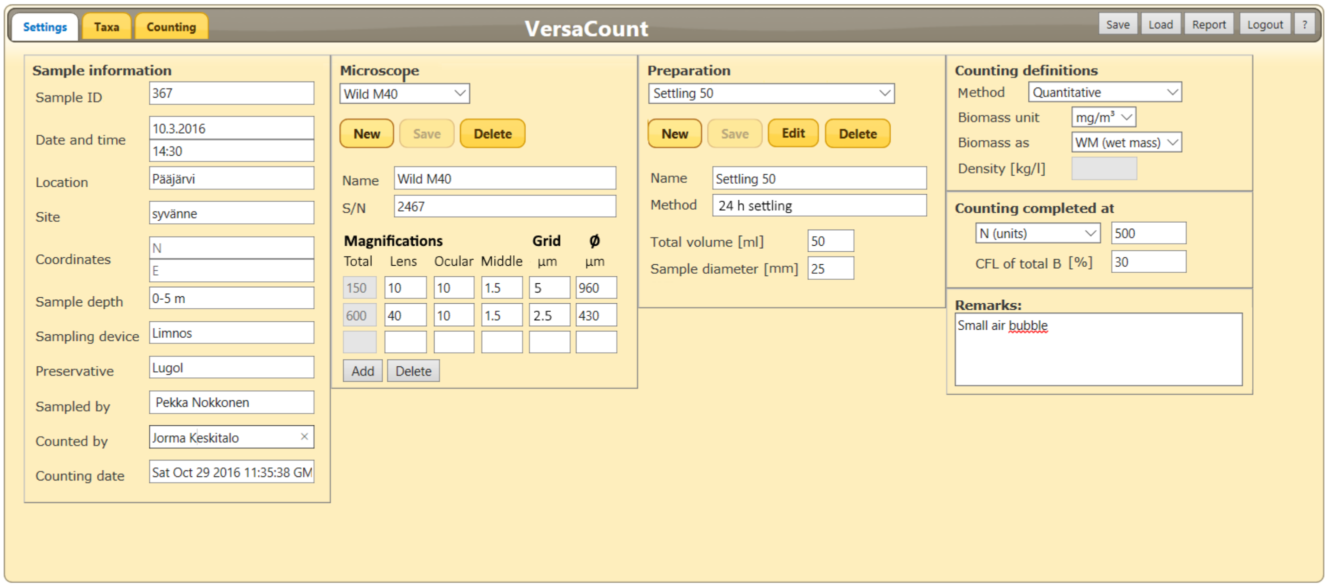
The program should follow the common client and server relation, the client being a web form using JavaScript and the server being programmed in the modern and efficient functional language Haskell.

The web client has already a planned user interface (has it?), with a custom css-file to provide the style of inputs.

2.1 Web client in a browser window

2.1.1 Basic settings

The basic settings are given in this tab (See figure 3).



Sample information

Sample ID	367
Date and time	10.3.2016 14:30
Location	Pääjärvi
Site	syvänne
Coordinates	N E
Sample depth	0-5 m
Sampling device	Limnos
Preservative	Lugol
Sampled by	Pekka Nokkonen
Counted by	Jorma Keskitalo
Counting date	Sat Oct 29 2016 11:35:38 GM

Microscope

Wild M40

Name	Wild M40
S/N	2467

Magnifications				Grid	ø
Total	Lens	Ocular	Middle	µm	µm
150	10	10	1.5	5	960
600	40	10	1.5	2.5	430

Preparation

Settling 50

Name	Settling 50
Method	24 h settling
Total volume [ml]	50
Sample diameter [mm]	25

Counting definitions

Method	Quantitative
Biomass unit	mg/m³
Biomass as	WM (wet mass)
Density [kg/l]	

Counting completed at

N (units)	500
CFL of total B [%]	30

Remarks:

Small air bubble

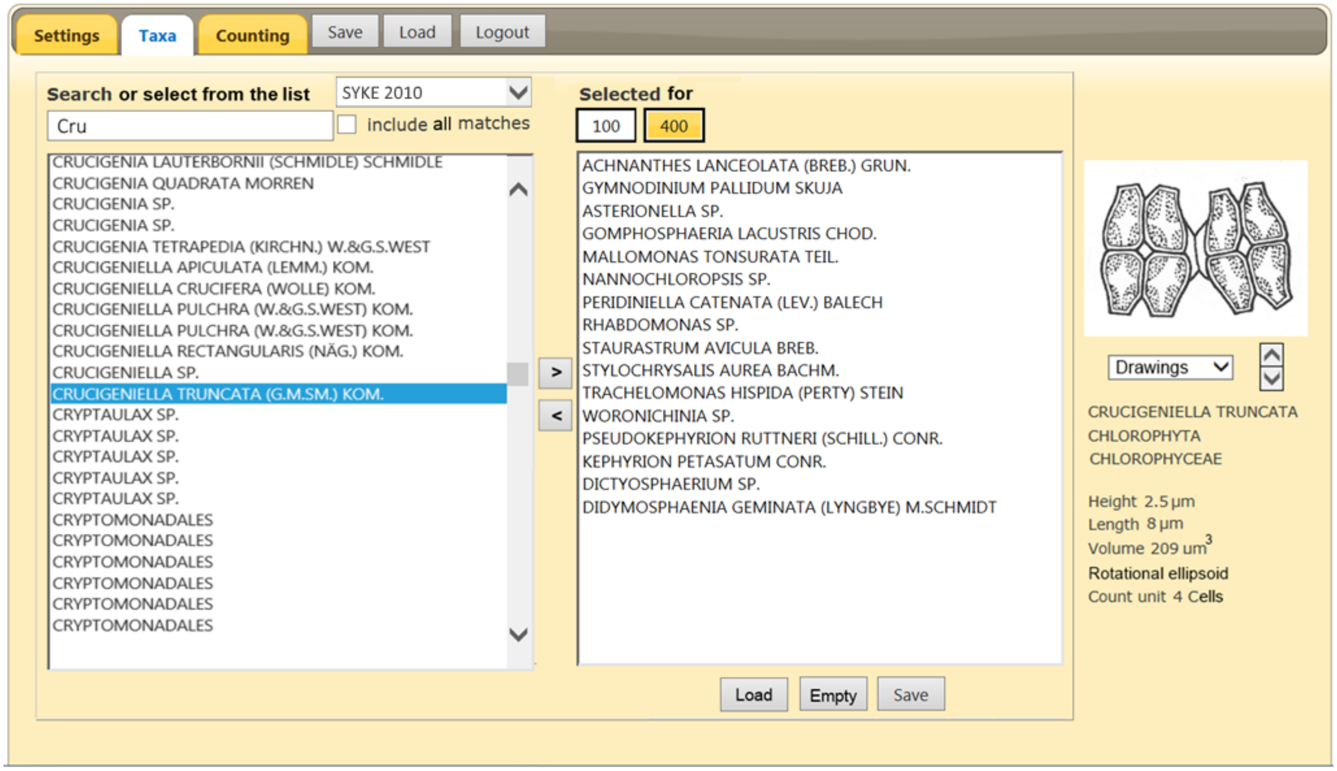
Figure 3: Tab for basic settings

2.1.2 The taxa

The taxa is chosen in this tab (Figure 4).

2.1.3 Counting

The actual counting happens in this tab (Figure 5). Hotkeys and color coding strongly used. Pictures can be shown and statistical analyses can be made in interaction with the server program. JavaScript programs do not have access to the file system.



Search or select from the list: SYKE 2010

Cru

☐ include all matches

CRUCIGENIA LAUTERBORNII (SCHMIDLE) SCHMIDLE
CRUCIGENIA QUADRATA MORREN
CRUCIGENIA SP.
CRUCIGENIA SP.
CRUCIGENIA TETRAPEDIA (KIRCHN.) W.&G.S.WEST
CRUCIGENIELLA APICULATA (LEMM.) KOM.
CRUCIGENIELLA CRUCIFERA (WOLLE) KOM.
CRUCIGENIELLA PULCHRA (W.&G.S.WEST) KOM.
CRUCIGENIELLA PULCHRA (W.&G.S.WEST) KOM.
CRUCIGENIELLA RECTANGULARIS (NÄG.) KOM.
CRUCIGENIELLA SP.
CRUCIGENIELLA TRUNCATA (G.M.SM.) KOM.
CRYPTAULAX SP.
CRYPTAULAX SP.
CRYPTAULAX SP.
CRYPTAULAX SP.
CRYPTAULAX SP.
CRYPTOMONADALES
CRYPTOMONADALES
CRYPTOMONADALES
CRYPTOMONADALES
CRYPTOMONADALES
CRYPTOMONADALES

Selected for: 100 | 400

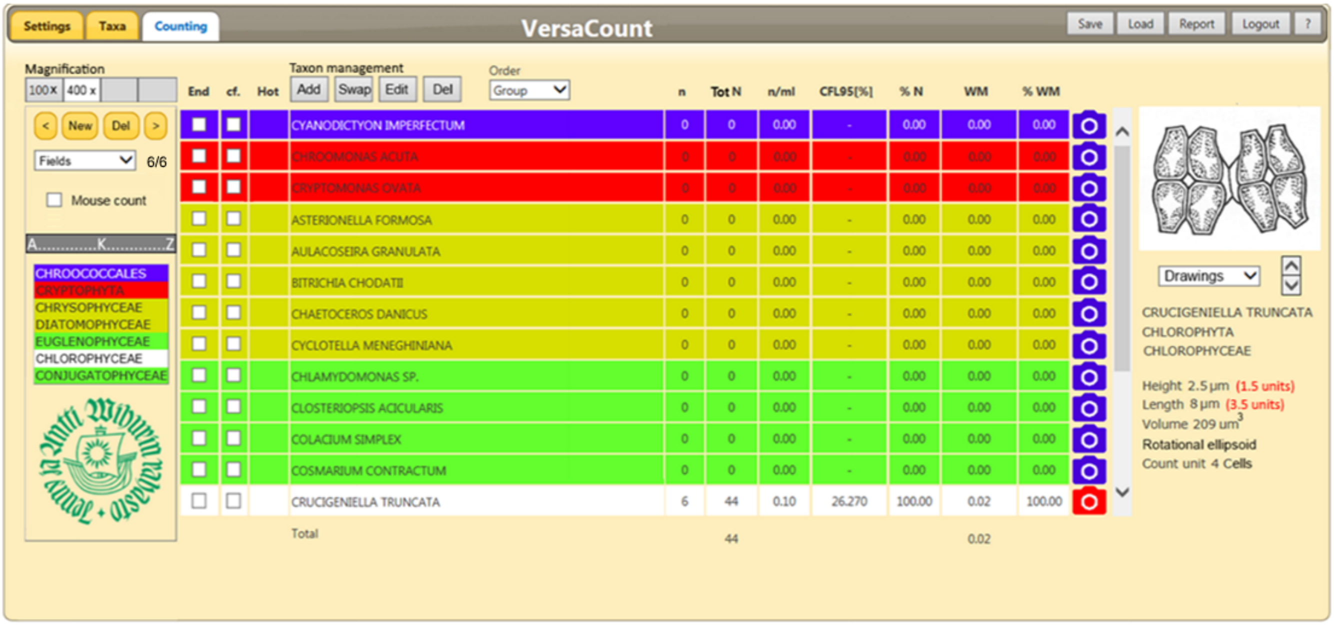
ACHNANTHES LANCEOLATA (BREB.) GRUN.
GYMNODINIUM PALLIDUM SKUJA
ASTERIONELLA SP.
GOMPHOSPHAERIA LACUSTRIS CHOD.
MALLOMONAS TONSURATA TEIL.
NANNOCHLOROPSIS SP.
PERIDINIELLA CATENATA (LEV.) BALECH
RHABDOMONAS SP.
STAURASTRUM AVICULA BREB.
STYLOCHRYSALIS AUREA BACHM.
TRACHELOMONAS HISPIDA (PERTY) STEIN
WORONICHINIA SP.
PSEUDOKEPHYRION RUTTNERI (SCHILL.) CONR.
KEPHYRION PETASATUM CONR.
DICTYOSPHAERIUM SP.
DIDYMOSPHAENIA GEMINATA (LYNGBYE) M.SCHMIDT

Drawings

CRUCIGENIELLA TRUNCATA
CHLOROPHYTA
CHLOROPHYCEAE

Height 2.5 µm
Length 8 µm
Volume 209 um^3
Rotational ellipsoid
Count unit 4 Cells

Figure 4: Tab for choosing the taxa



Magnification: 100x | 400 x

Taxon management

Order: Group

Fields 6/6

☐ Mouse count

CHROOCOCCALES
CRYPTOPHYTA
CHRYSOPHYCEAE
DIATOMOPHYCEAE
EUGLENOPHYCEAE
CHLOROPHYCEAE
CONJUGATOPHYCEAE

End	cf.	Hot		n	Tot N	n/ml	CFL95[%]	% N	WM	% WM
☐	☐		CYANODICTYON IMPERFECTUM	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CHROOMONAS ACUTA	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CRYPTOMONAS OVATA	0	0	0.00	-	0.00	0.00	0.00
☐	☐		ASTERIONELLA FORMOSA	0	0	0.00	-	0.00	0.00	0.00
☐	☐		AULACOSEIRA GRANULATA	0	0	0.00	-	0.00	0.00	0.00
☐	☐		BITRICHIA CHODATII	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CHAETOCEROS DANICUS	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CYCLOTELLA MENEGHINIANA	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CHLAMYDOMONAS SP.	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CLOSTERIOPSIS ACICULARIS	0	0	0.00	-	0.00	0.00	0.00
☐	☐		COLACIUM SIMPLEX	0	0	0.00	-	0.00	0.00	0.00
☐	☐		COSMARIUM CONTRACTUM	0	0	0.00	-	0.00	0.00	0.00
☐	☐		CRUCIGENIELLA TRUNCATA	6	44	0.10	26.270	100.00	0.02	100.00
			Total		44				0.02	

Drawings

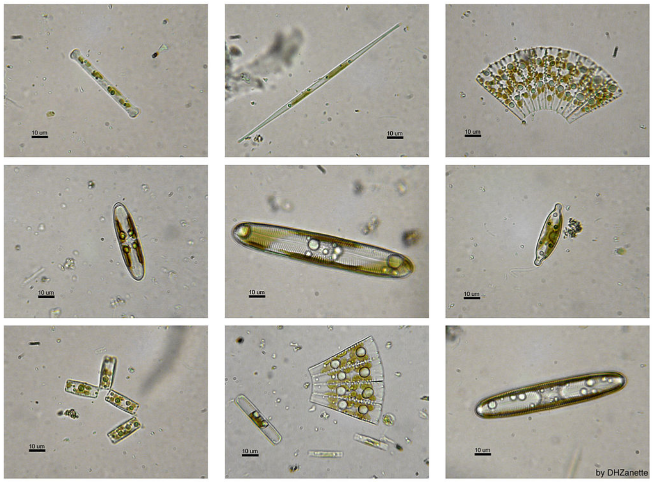
CRUCIGENIELLA TRUNCATA
CHLOROPHYTA
CHLOROPHYCEAE

Height 2.5 µm (1.5 units)
Length 8 µm (3.5 units)
Volume 209 um^3
Rotational ellipsoid
Count unit 4 Cells

Figure 5: Tab for counting

2.2 Haskell web server

Web server can be installed in different location or it can be run locally in the same machine as the client. Databases are kept here. Databases can include pictures, counting results, statistics, base information, etc.



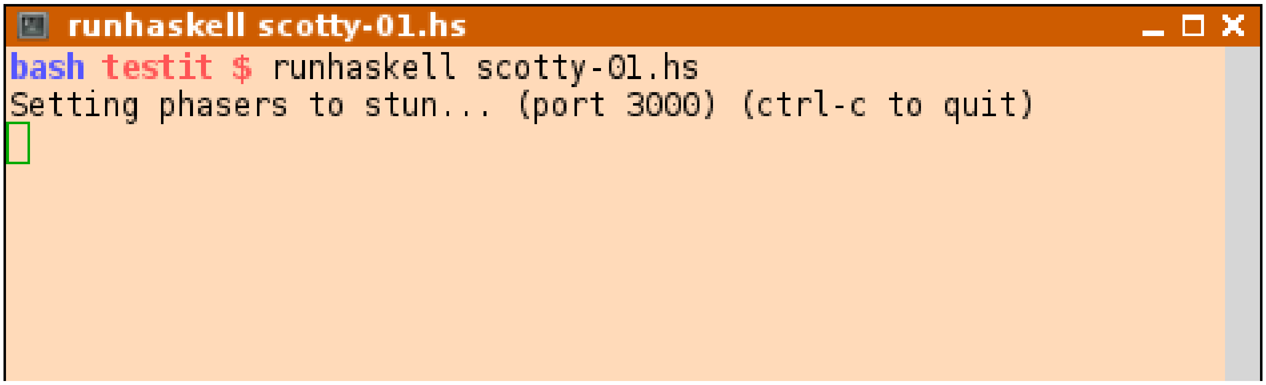
by DHZanette

Figure 6: Database of pictures. Diatomes.

3 Haskell Scotty web framework

To implement a Haskell web server we have several possibilities for a choise of web framework. In this example we have chosen the Scotty web framework, which is one of the easiest to install and run.

Having the following example code written in source file `scotty-01.hs` we can run it in interpreted mode by the command `runhaskell scotty-01.hs`.



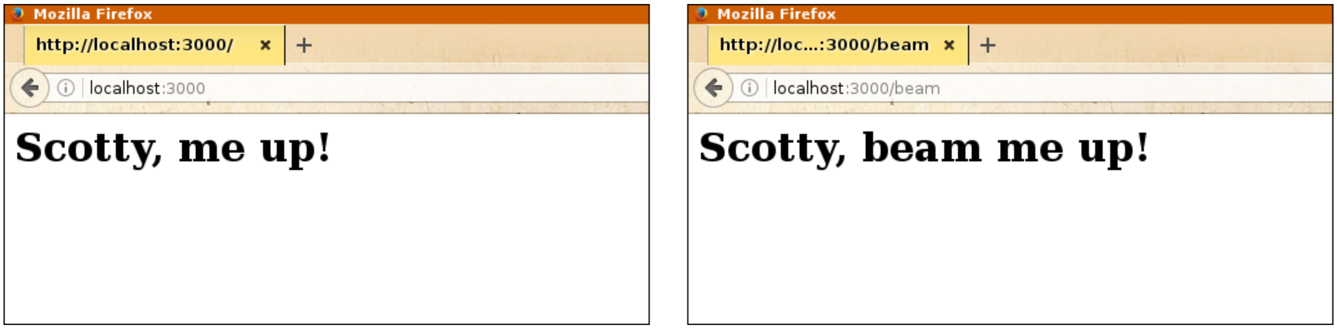
```
bash testit $ runhaskell scotty-01.hs
Setting phasers to stun... (port 3000) (ctrl-c to quit)
```

Figure 7: Running Scotty in terminal window

```
{-# LANGUAGE OverloadedStrings #-}
import Web.Scotty
import Data.Monoid (mconcat)

main = scotty 3000 $ do
    get ”/:word” $ do
        beam <- param ”word”
        html $ mconcat [”<h1>Scotty, ”, beam, ” me up!</h1>”]
```

After starting the server, we can direct our browser to `localhost:3000`. We can change the route and get a different answer (Figure 8). This is where we can ask Scotty for complicated calculations or pictures.



Scotty, me up!

Scotty, beam me up!

Figure 8: Scotty responding to different routes at localhost

4 Knockout.js

The program for counting plankton runs in a web browser. The functionality of the elements is implemented in JavaScript. We use the open source library Knockout.js. The following are some examples of the functionality from the Knockout.js web page.

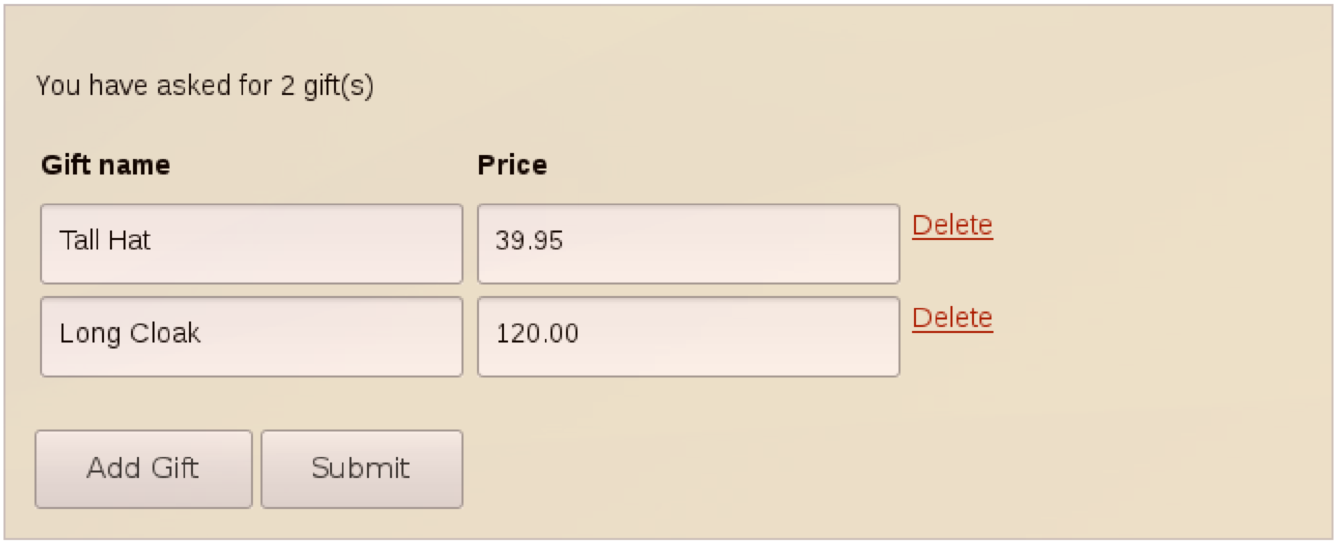
4.1 Grid editor example

The grid editor example demonstrates the interaction of page elements (See figure 9).

4.1.1 Source code: View model

```
var GiftModel = function(gifts) {
    var self = this;
    self.gifts = ko.observableArray(gifts);

    self.addGift = function() {
        self.gifts.push({
            name: ””,
            price: ””
```



You have asked for 2 gift(s)

Gift name	Price	
Tall Hat	39.95	Delete
Long Cloak	120.00	Delete

Figure 9: Grid Editor

```
        });
    };

    self.removeGift = function(gift) {
        self.gifts.remove(gift);
    };

    self.save = function(form) {
        alert(”Could now transmit to server: ” + ko.utils.stringifyJson(self.gifts));
        // To actually transmit to server as a regular form post, write this:
        // ko.utils.postJson($(”form”)[0], self.gifts);
    };
};

var viewModel = new GiftModel([
    { name: ”Tall Hat”, price: ”39.95”},
    { name: ”Long Cloak”, price: ”120.00”}
]);
ko.applyBindings(viewModel);

// Activate jQuery Validation
$(”form”).validate({ submitHandler: viewModel.save });
```

4.1.2 Source code: View

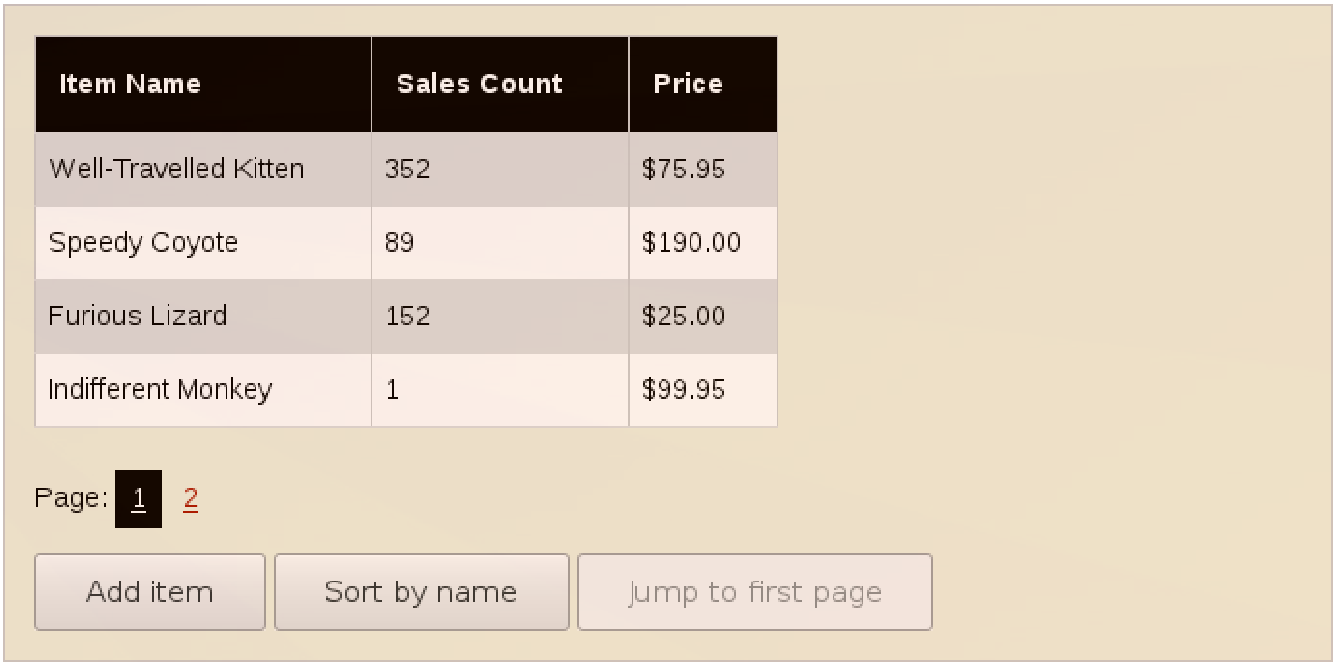
```
<form action='/someServerSideHandler'>
 <p>You have asked for <span data-bind='text: gifts().length'>&nbsp;</span>
    gift(s)</p>
 <table data-bind='visible: gifts().length > 0'>
```

```
     <thead>
      <tr>
        <th>Gift name</th>
        <th>Price</th>
        <th />
      </tr>
  </thead>
  <tbody data-bind=’foreach: gifts’>
      <tr>
        <td><input class=’required’ data-bind=’value: name,
                                          uniqueName: true’ /></td>
        <td><input class=’required number’ data-bind=’value: price,
                                                 uniqueName: true’ /></td>
        <td><a href=’#’ data-bind=’click: $root.removeGift’>Delete</a></td>
      </tr>
     </tbody>
 </table>

 <button data-bind=’click: addGift’>Add Gift</button>
 <button data-bind=’enable: gifts().length > 0’ type=’submit’>Submit</button>
</form>
```

4.2 Paged grid

This is a similar example including a paged grid (See figure 10).



Item Name	Sales Count	Price
Well-Travelled Kitten	352	$75.95
Speedy Coyote	89	$190.00
Furious Lizard	152	$25.00
Indifferent Monkey	1	$99.95

Page: 1 2

Figure 10: Paged Grid

4.2.1 Source code: View

```
<div data-bind=’simpleGrid: gridViewModel’> </div>

<button data-bind=’click: addItem’>
    Add item
</button>

<button data-bind=’click: sortByName’>
    Sort by name
</button>

<button data-bind=’click: jumpToFirstPage, enable: gridViewModel.currentPageIndex’>
    Jump to first page
</button>
```

4.2.2 Source code: View model

```
var initialData = [
    { name: ”Well-Travelled Kitten”, sales: 352, price: 75.95 },
    { name: ”Speedy Coyote”, sales: 89, price: 190.00 },
    { name: ”Furious Lizard”, sales: 152, price: 25.00 },
    { name: ”Indifferent Monkey”, sales: 1, price: 99.95 },
    { name: ”Brooding Dragon”, sales: 0, price: 6350 },
    { name: ”Ingenious Tadpole”, sales: 39450, price: 0.35 },
    { name: ”Optimistic Snail”, sales: 420, price: 1.50 }
];

var PagedGridModel = function(items) {
    this.items = ko.observableArray(items);

    this.addItem = function() {
        this.items.push({ name: ”New item”, sales: 0, price: 100 });
    };

    this.sortByName = function() {
        this.items.sort(function(a, b) {
            return a.name < b.name ? -1 : 1;
        });
    };

    this.jumpToFirstPage = function() {
        this.gridViewModel.currentPageIndex(0);
    };

    this.gridViewModel = new ko.simpleGrid.viewModel({
        data: this.items,
        columns: [
            { headerText: ”Item Name”, rowText: ”name” },
            { headerText: ”Sales Count”, rowText: ”sales” },
            { headerText: ”Price”, rowText: function (item) { return ”$”
              + item.price.toFixed(2) } }
        ],
        pageSize: 4
    });
};

ko.applyBindings(new PagedGridModel(initialData));
```

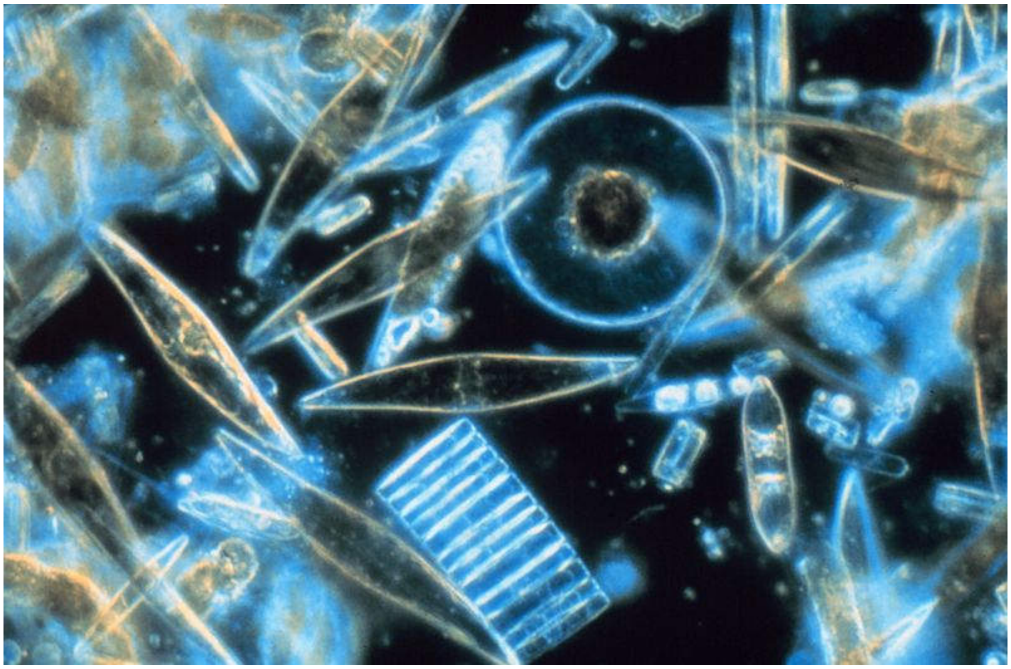


Figure 11: Diatoms in microscopic image