KATHOLIEKE HOGESCHOOL ST.LIEVEN
Campus Rabot
Gebroeders Desmetstraat 1
9000 Gent, BELGIUM

UNIVERSITAT DE GIRONA
Escola Politècnica Superior
Av.Lluis Santaló s/n (campus Montilivi)
17001 Girona

STUDENT'S WATCHER

Xavi Pujolràs Riera
Enginyeria Tècnica en Informàtica de Gestió
Supervisor: Rogier van der Linde
2006 / 2007

SUMMARY

Evolution of my project

I am an Erasmus student from Universitat de Girona (Spain), and my tutor, Rogier Van Der Linde. When I arrived in Kaho Saint Lieven he proposed to me to do my project about an application to see the evolution of their students.
I liked this idea, and I started to work about it.
At the beginning Student's Watcher was begun in ASP.net. The two main objectives of my project were to do a useful application, and to learn something about ASP.net.
I had been working in two companies with ASP.net before, and in three weeks a small application was working with the essential properties to be used. It was then, when we thought that I had a lot of time to do my project, 4 months exactly, and it could be nice to learn an another way to work totally different than ASP.net.
Finally my project would be developed in PHP, HTML and CSS, a totally unknowns languages by me.
From then, Rogier started to teach me these languages and I started to work in a new way of my project.
Each day I saw more the power of this combination, and finally, I finish my project in these four months. I have learned PHP, HTML, CSS, and something about JavaScript and AJAX at the same time that I have done my project.
This could be a good way to do a comparison between these two languages at the end of the project.

Description of the project

The documentation of the project has been detached in four parts. The first one has been called '*About Student's Watcher',* and there are a small introduction about what is this application, and the User Guide. This last one has two parts: *Starting,* to teach how to install the application and configure it and *Using Student's Watcher* to show a general use when application has been installed.
The second part is the *Pre Study*. Here we find an introduction to all languages used in the project, explaining the main characteristics about these.
The following point is a *Detailed description* of the project. First there is an explication of the *three tier structure* used.
The *classes and directories* subchapter shows the code distribution in directories and classes.
The three tiers are showed separated in the following chapters, showing in the first one the database structure, as tables, attributes…

In the business and presentation tier chapters, it is able to see some fragments of code to show the most important or difficult to understand parts of Student’s Watcher.
Finally the last part is the conclusion, to do the final evaluation of the project, its evolution, and the comparison between what I knew in ASP.net and what I know now about the new languages learned.

I would like to thank to my tutor, Rogier who has taught me everything I know about PHP, CSS, JavaScript and AJAX. He has been helping me a lot.

I also thank to my girlfriend Laura and my family,
Josep, Pepita and Glòria, for their unconditional support.

INDEX

1. ABOUT STUDENT’S WATHCER

1.1 What is Student’s Watcher?

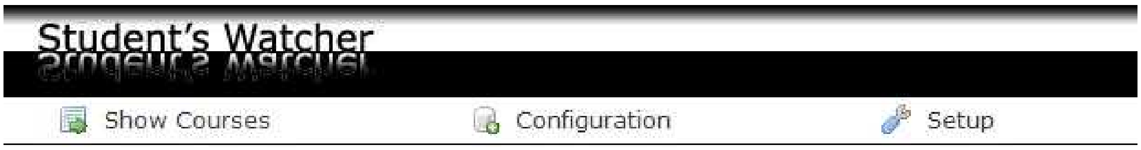
“Student’s Watcher” is a small Web application which wants to show in a visual, simple and fast way, the evolution of the students.

The main project table displays such things as marks and comments about students. We can add a comment for each mark to explain why this mark. The objective is to be able to know if some student has a problem, how is going his year, marks in other courses, or even, to know if he has a bad week in a different subjects.
We can see the evolution of students in past years to do an objective comparison. It could be really interesting especially for students who are repeating a course.
It also allows inserting global comments of student, we have a list of these, and all professors can add new ones, where we can see more general valuations.

1.2 User guide

1.2.1 Installing application

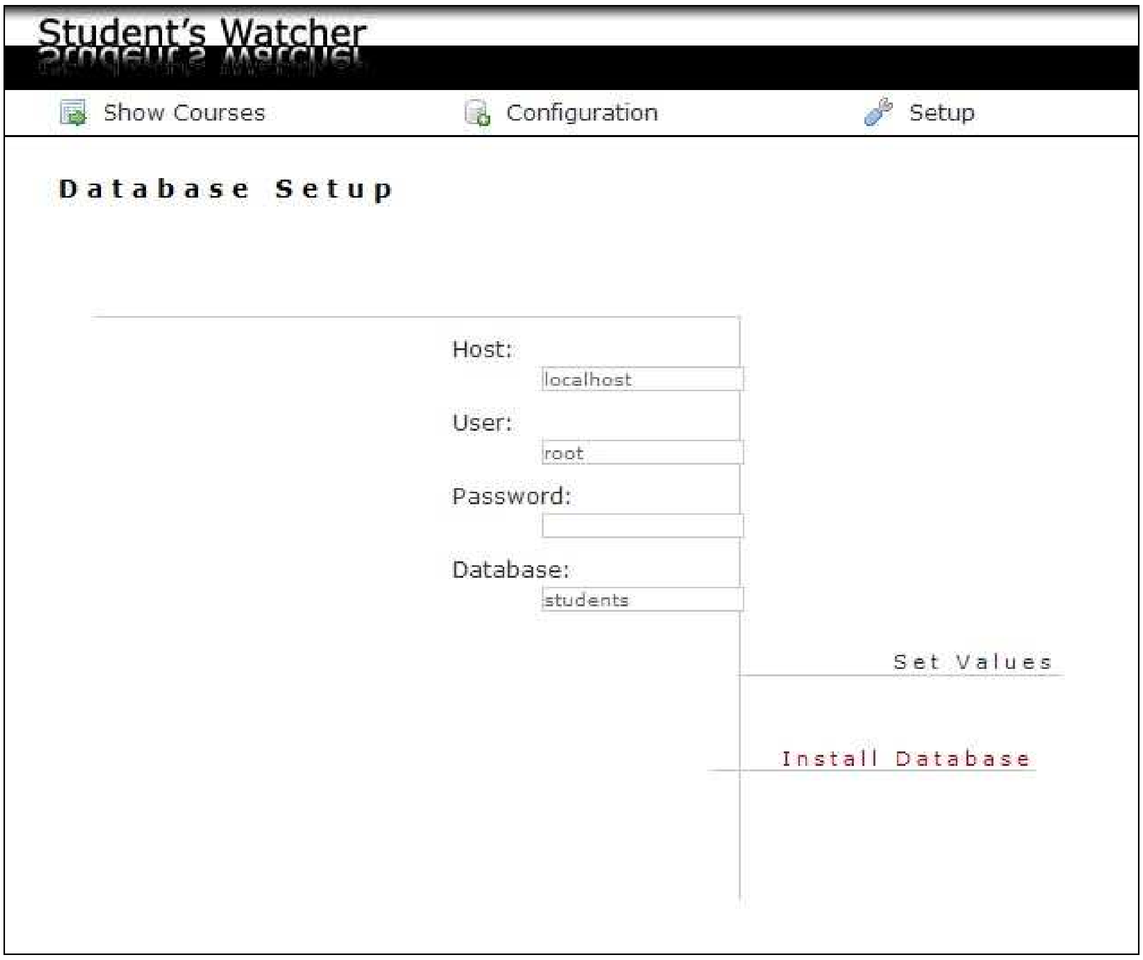
In 'Student's Watcher' main menu, we can find three options. We can see this menu in the following image (img 1).



(img 1)

This menu will be present in all screens of the application, and in all moment it can use it to navigate around it.

Student's Watcher has been designed to be easy to install. It only needs to be copied in our system and open the *index.php* page. If it does not detect an old database it will show automatically the setup page (*setup.php*) (img 2).



Database Setup

Host: localhost

User: root

Password:

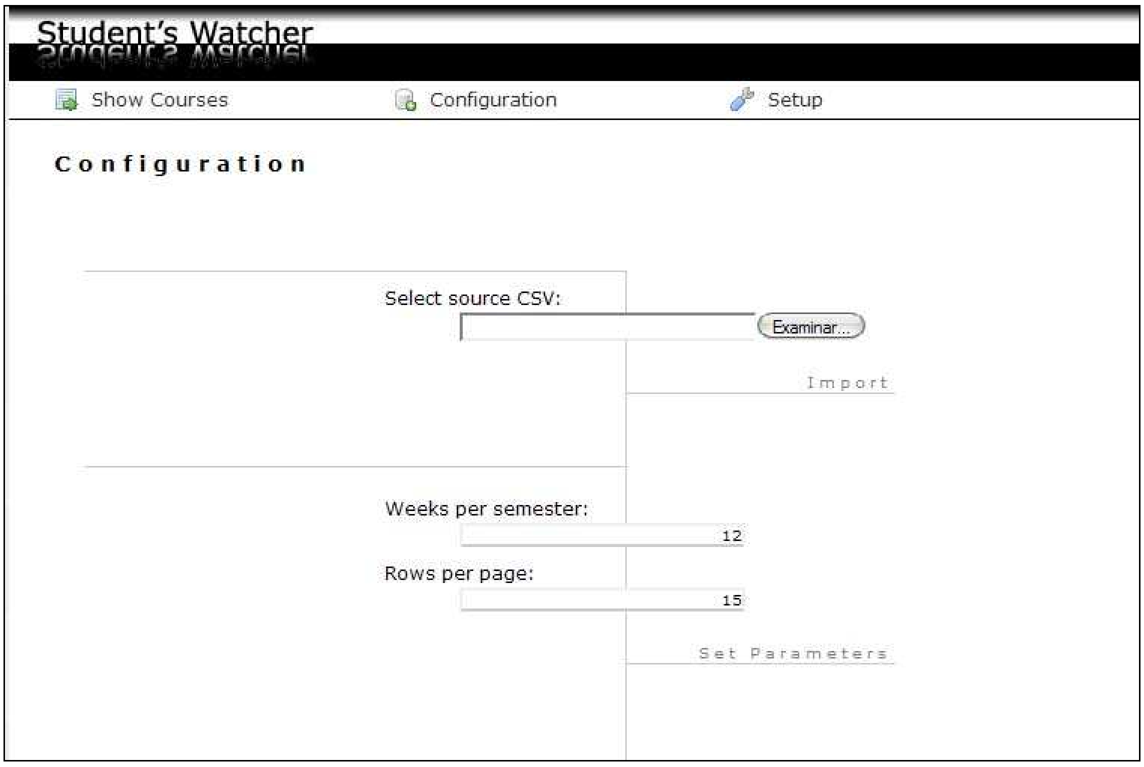
Database: students

(img 2)

In this page we can find a four textboxes to insert some parameters. The first one is called *Host*, here we should write the host to use the application.
The two next textboxes need a valid user and password from our database.
And finally we can choose the database to use, to find the information of the students, courses and all that we will need.
If we do not have all this parameters with correct values, we can't open another page of the application, all links takes to the setup page.
When we have been inserted all required values, we may click *Set Values* option to save all new values of textboxes.
Finally, if we want to Install database, and erase all old tables, we may click *Install Database* button.
Now our application is ready to be used. All options are opened and we can choose what we want to do.
We can always return to the setup page to change these values or reinstall our database.

1.2.2 Configuration

When database has been installed it opens automatically the configuration page (*config.php*) (img 3). Here, we can change some initial parameters and fill our database.



Configuration

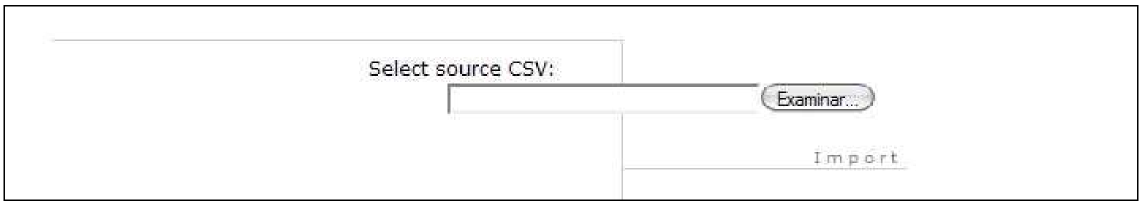
Select source CSV:

Weeks per semester: 12

Rows per page: 15

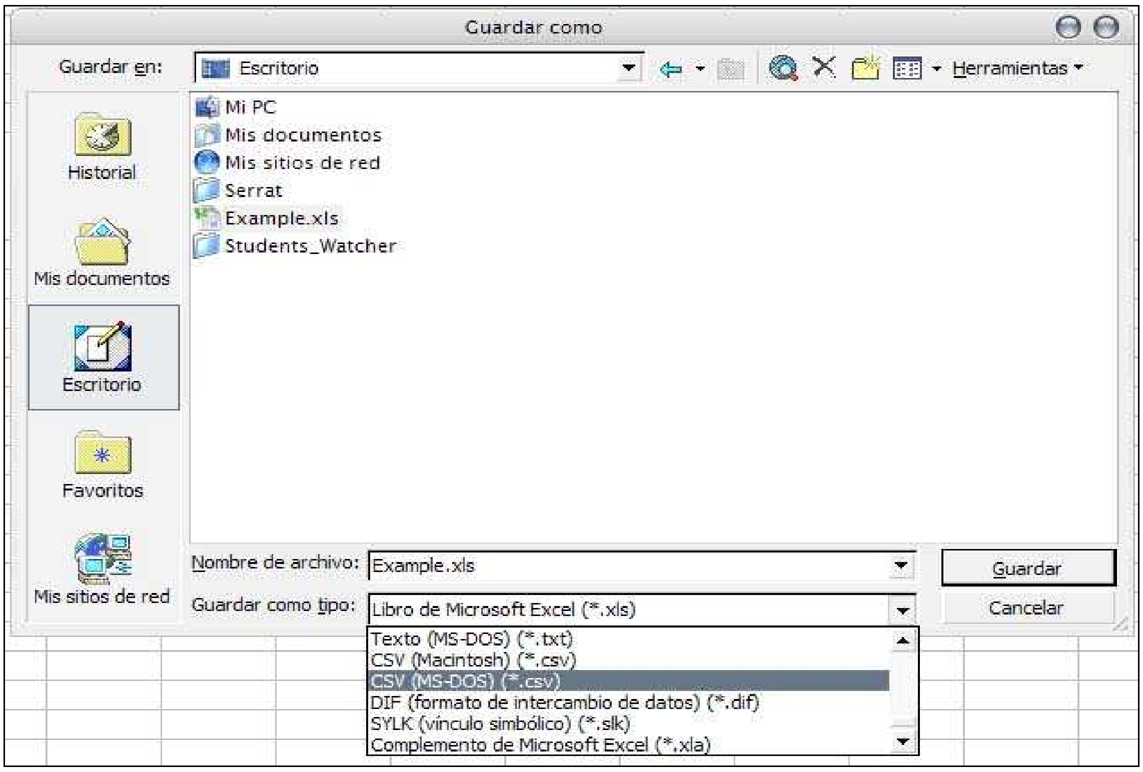
(img 3)

In the lower part of the screen we have two textboxes where we can change different default values of the application. The first one is called *weeks per semester*. We should know how many weeks will have in our semester to say it to Student's Watcher.
The following value is used by the pager. We can select the number of rows that we want to show per page in the tables of the application.
To save the values we should click *Set Parameters* option.
In the higher part of the screen we have one of most important steps to start our application (img 4).



(img 4)

Here we will insert all information in the database. To do it we need a CSV file to take the data. It is easy to create a CSV file from an Excel file. If we have the Excel file with all our data, we open this file, and in the 'File' menu we chose 'Save as'. We will see a new window like the following (img 4).



Nombre de archivo: Example.xls
Guardar como tipo: Libro de Microsoft Excel (*.xls)

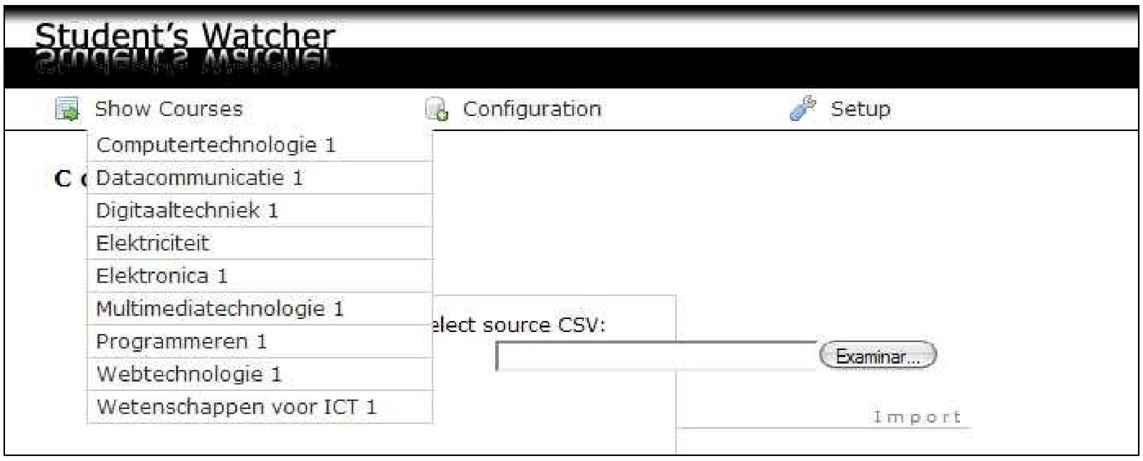
(img 5)

In this window we have to open the combo box called *Save as type* and chose *CSV (MS-DOS)(*.csv)* option and click *Save.*

Our excel file must be with a special structure. We can find an example of a good excel structure, because our program knows how to read it, in a ‘*files’* directory of the documentation CD. This file is called ‘*CSVStructure.xls*’ and also we can find there the ‘*CSVStructure.csv*’ to see the final structure of the file.
When we have the CSV file created we can chose it and click in a *Import* option to load all our data in the database. This process could need few minutes because the program has to create all subscriptions for all students and course.
Now, the application is ready to use, we have the data loaded and all parameters inserted.

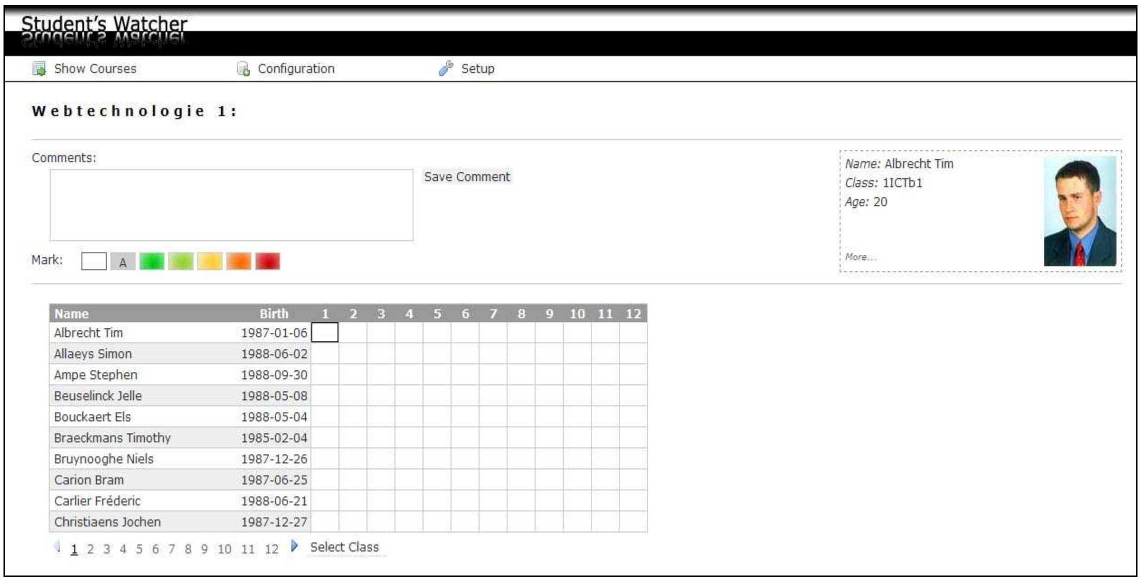
1.2.3 Showing a course

When we have all steps of the last chapter done, we can start to use our application correctly.
The application automatically activates the *Show Courses* menu, in our main menu. Here we can see all courses inserted in the database. We only have to choose which want to show (img 6).



(img 6).

When we click on one course it shows a new page of it called *ShowCourse.php* (img 7).



Webtechnologie 1:

Comments:

Save Comment

Mark: ☐ A 🟩 🟩 🟨 🟧 🟥

Name: Albrecht Tim
Class: 1ICTb1
Age: 20
More...

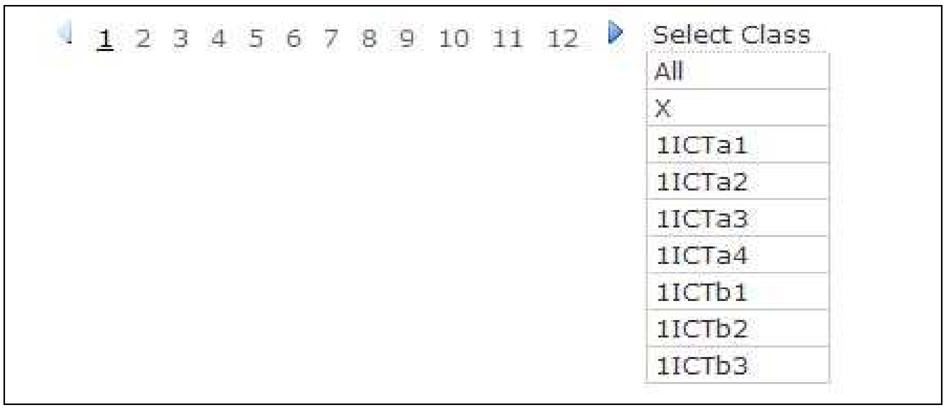
Name	Birth	1	2	3	4	5	6	7	8	9	10	11	12
Albrecht Tim	1987-01-06												
Allaeys Simon	1988-06-02												
Ampe Stephen	1988-09-30												
Beuselinck Jelle	1988-05-08												
Bouckaert Els	1988-05-04												
Braeckmans Timothy	1985-02-04												
Bruynooghe Niels	1987-12-26												
Carion Bram	1987-06-25												
Carlier Fréderic	1988-06-21												
Christiaens Jochen	1987-12-27												

1 2 3 4 5 6 7 8 9 10 11 12 Select Class

(img 7)

In the centre of this page there is a table with the students in the course selected before.
Inside this table we have the name of the student, first column, second is the date of birth, and we have one column for each week in semester. Under the table there is the

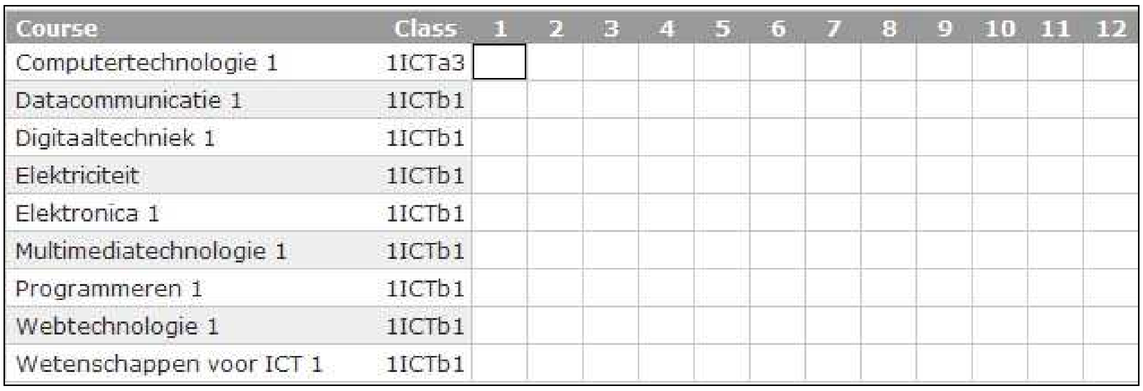
pager, to select the page we want to show in the table, and the class selector menu (img 8).



(img 8)

1.2.4 Showing a student

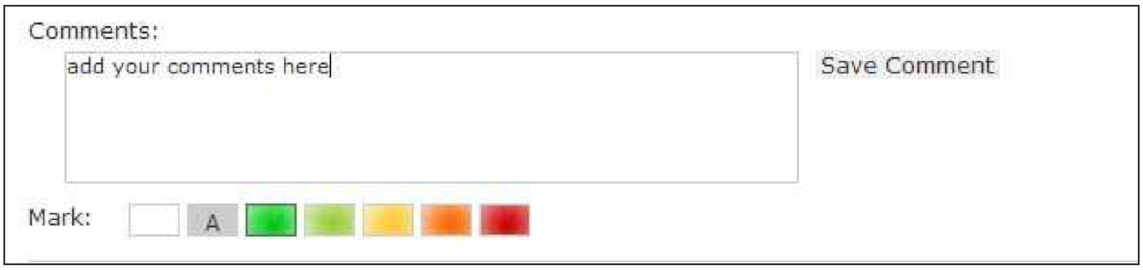
Here we can show only the students in a specific class or all classes. When we open the application, it shows all classes.
We can click on a student name to show a new table with all courses where is subscribed and the respective marks (img 9).



Course	Class	1	2	3	4	5	6	7	8	9	10	11	12
Computertechnologie 1	1ICTa3												
Datacommunicatie 1	1ICTb1												
Digitaaltechniek 1	1ICTb1												
Elektriciteit	1ICTb1												
Elektronica 1	1ICTb1												
Multimediatechnologie 1	1ICTb1												
Programmeren 1	1ICTb1												
Webtechnologie 1	1ICTb1												
Wetenschappen voor ICT 1	1ICTb1												

(img 9)

When we want to add a mark or a comment we only have to click on a cell that we want to change.
It is possible to move around these cells using the direction keys on our keyboard.
To change a mark (img 10) we should click on a colour in the mark panel, on superior part of screen. In the same way, we can add or modify a comment for this week and mark, adding it in the text area called *Comments* and then clicking the *Save Comment* button.

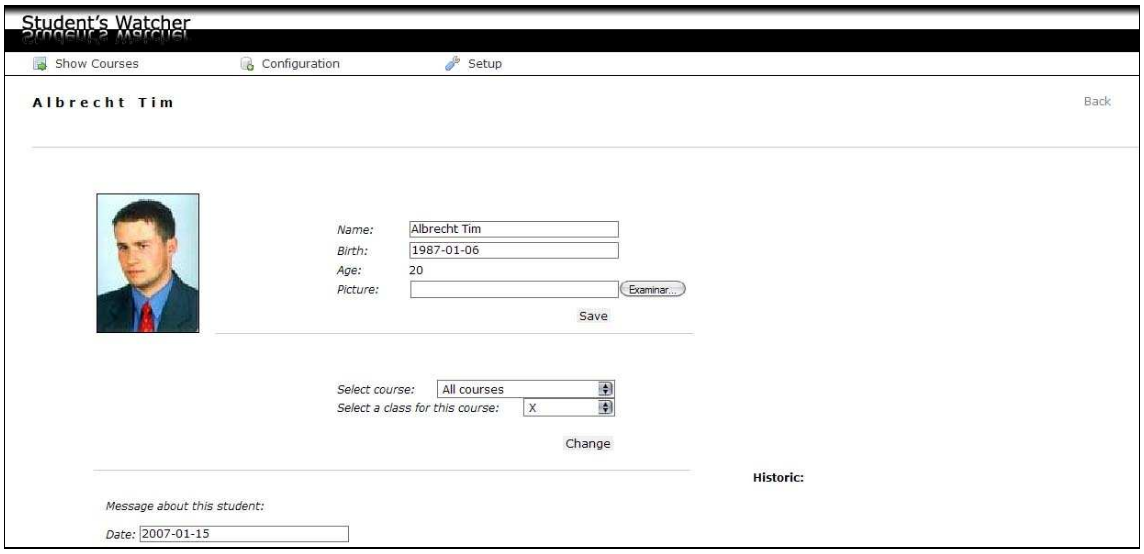


(img 10)

It is possible to insert a mark without comments or vice versa.

1.2.5 Personal fiche

The last part of the main page of Student's Watcher we find it on our right side. There is a panel with a photo and information about the selected student. In this panel we have a small link called *More…* It opens a new page (*fiche.php*) (img 11).



Albrecht Tim

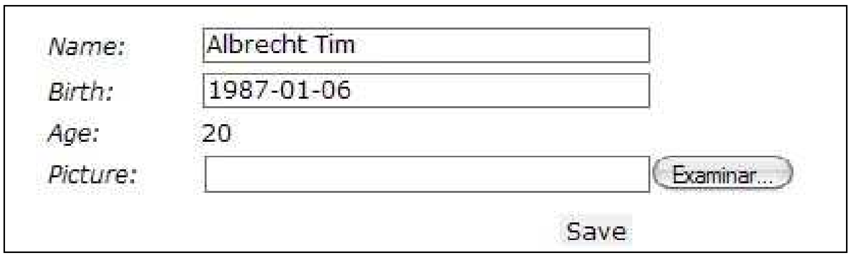
Name:	Albrecht Tim
Birth:	1987-01-06
Age:	20
Picture:	
Select course:	All courses
Select a class for this course:	X

Message about this student:

Date: 2007-01-15

(img 11)

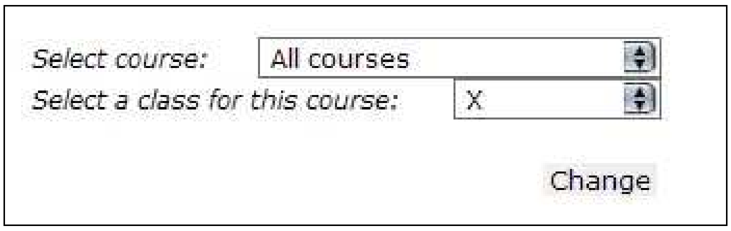
Here we have three different parts to work. First one (img 12) is used to change the personal information about student, like a name, birth or the photograph.



Name:	Albrecht Tim
Birth:	1987-01-06
Age:	20
Picture:	

(img 12)

The next panel (img 13) permit to change the class of the student for each course. We only have to select a course and then a class and click on *change* button. If we would like to change the class of all courses by this student, we just should use the option *All courses* in the first combo box.

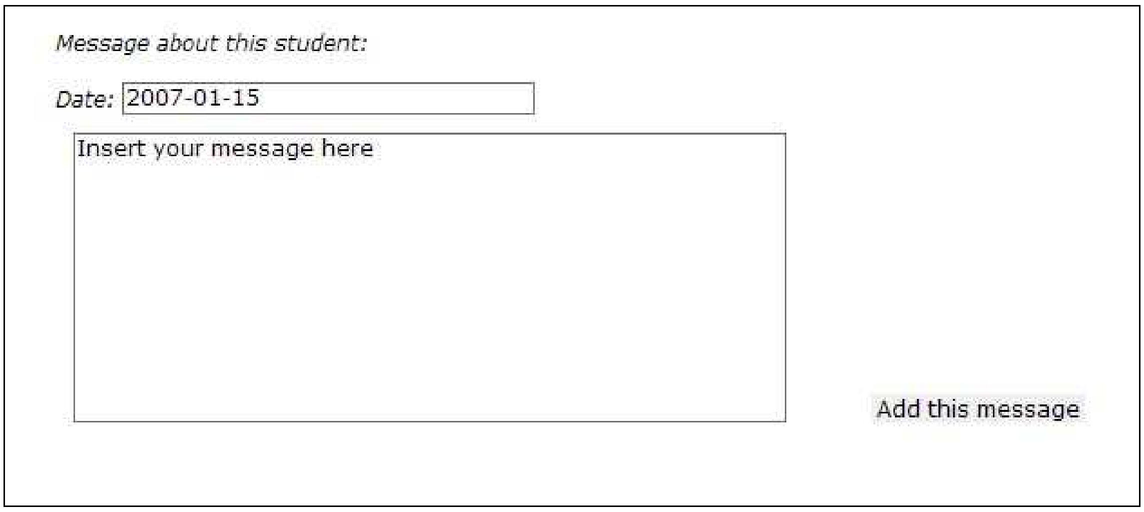


Select course: All courses

Select a class for this course: X

(img 13)

Finally, in the inferior part of this screen (img 14) , we can add messages about the student. These messages will be diferents than the comments inserted before in a mark for each week. Here the professor can add a general messages for this student, and all professors can see it to know a important things about the student.



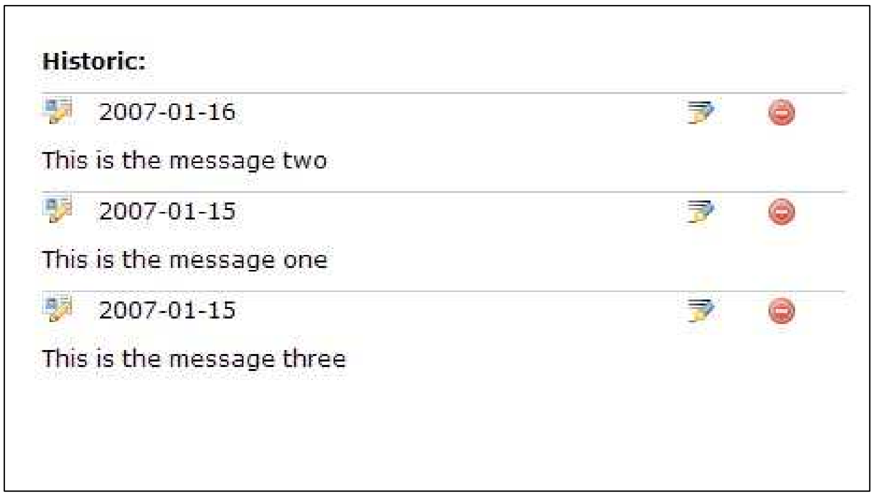
Message about this student:

Date: 2007-01-15

Insert your message here

(img 14)

The messages will be added in a right side of the screen called *Historic* (img 15) there, we see all messages added by all professors for this student.



Historic:

2007-01-16

This is the message two

2007-01-15

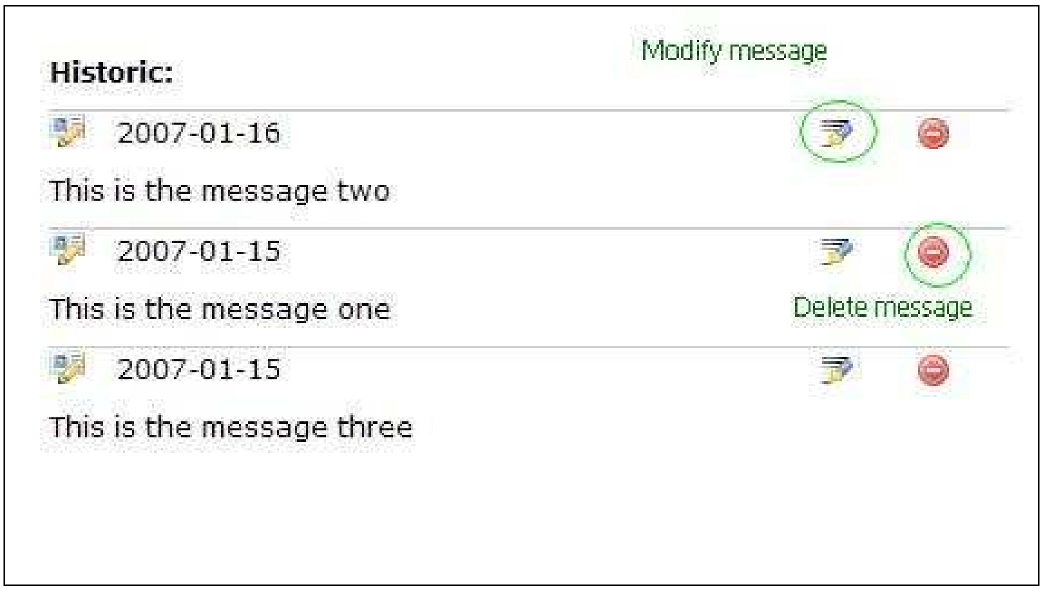
This is the message one

2007-01-15

This is the message three

(img 15)

It is possible to modify the messages clicking on the icon with a pen and some lines. If we click there, this message will be inserted in the panel messages (img 14) and we have to click on *add this message* to save the modification.
In the same way it is possible to delete messages using the icon on the right with a red circle.
It can see it in the next image: (img 16)



Historic:

2007-01-16
This is the message two

2007-01-15
This is the message one

2007-01-15
This is the message three

(img 16)

2. PRE STUDY

Due to the rapid pace of change in the high tech sector, we often need to evaluate new technologies in order to decide whether to allocate time to learning and using new systems. Jump on the bandwagon too early, and you risk becoming involved with something that just heads downhill or does not go anywhere. Wait too long, and you may find yourself behind the times with regards to “the latest thing”.
One step in all programming projects is to choose the development language to work. Sometimes this can get to be really complicated due to the great variety of good languages that at the moment we have. This decision will influence in the rest of our project.
Programming languages are a particular area of interest. Selecting a language involves many factors, and certainly is not something that can be considered in a vacuum. Of course, it is important to pick something that can do the job correctly and efficiently, but depending on what you need to accomplish, and who you have to work with, the availability of external libraries, people to help you out, or even to hire you or be hired by you can all be important factors to weigh.

Programming languages, like any product, have certain properties. Obviously, like any other sort of information good, production costs in the sense of making copies are essentially zero. Research and development (sunk costs) are needed to create the software itself, which means that an initial investment is required, and if the language is not successful, chances are the investment can not be recouped.
This applies to many information goods, but programming languages also have some qualities that make them special within this grouping. Namely, that they are both a means of directing computers and their peripherals to do useful work, but they are also a means of exchanging ideas and algorithms for doing that work between people. In other words, languages go beyond simply being something that's useful; they are also a means of communication. Furthermore, in the form of collections of code such as packages, modules or libraries, programming languages are also a way to exchange useful routines that can be recombined in novel ways by other programmers, instead of simply exchanging finished applications.

2.1 ASP.NET

Principles of ASP.NET

Even though ASP.NET takes its name from Microsoft's old web development technology, ASP, the two differ significantly. Microsoft has completely rebuilt ASP.NET, based on the Common Language Runtime (CLR) shared by all Microsoft .NET applications. Programmers can write ASP.NET code using any of the different programming languages supported by the .NET Framework, usually C#, Visual Basic.NET, or JScript .NET, but also including open-source languages such as Perl and Python. ASP.NET has performance benefits over other script-based technologies because the server-side code is compiled to one or a few DLL files on a web server.

ASP.NET attempts to simplify developers' transition from Windows application development to web development by offering the ability to build pages composed of controls similar to a Windows user interface. A web control, such as a button or label, functions in very much the same way as its Windows counterpart: code can assign it properties and respond to its events. Controls know how to render themselves: whereas Windows controls draw themselves to the screen, web controls produce segments of HTML and JavaScript which form part of the resulting page sent to the end-user's browser.

ASP.NET encourages the programmer to develop applications using an event-driven GUI paradigm, rather than in conventional web-scripting environments like ASP and PHP. The framework attempts to combine existing technologies such as JavaScript with internal components like "Viewstate" to bring persistent (inter-request) state to the inherently stateless web environment.

ASP.NET uses the .NET Framework as an infrastructure. The .NET Framework offers a managed runtime environment (like Java), providing a virtual machine with JIT and a class library.

The numerous .NET controls, classes and tools can cut down on development time by providing a rich set of features for common programming tasks. Data access provides one example, and comes tightly coupled with ASP.NET. A developer can make a page to display a list of records in a database, for example, significantly more readily using ASP.NET than with traditional web technologies like ASP or PHP.

2.2 PHP

History

PHP was written as a set of CGI binaries in the C programming language by the Danish-Canadian programmer Rasmus Lerdorf in 1994, to replace a small set of Perl scripts he had been using to maintain his personal homepage. Lerdorf initially created PHP to display his résumé and to collect certain data, such as how much traffic his page was receiving. "Personal Home Page Tools" was publicly released on June 8, 1995 after Lerdorf combined it with his own Form Interpreter to create PHP/FI.

PHP 5

On July 13, 2004, PHP 5 was released, powered by Zend Engine II. PHP 5 included new features such as:

- Robust support for Object-Oriented Programming (or OOP) through PHP Data Objects
- Performance enhancements taking advantage of the new engine
- Better support for MySQL through a completely rewritten extension
- Better XML support through a suite of interoperable tools
- Embedded support for SQLite
- Integrated SOAP support
- Data iterators
- Error handling through exceptions

The latest version as of November 2006 is PHP 5.2.0.

Usage

PHP generally runs on a web server, taking PHP code as its input and creating Web pages as output, but command-line scripting and client-side GUI applications are part of the three primary uses of PHP as well. PHP can be deployed on any web server and on almost every OS platform free of charge. The PHP Group also provides the complete source code for users to build, customize and extend for their own use.

Server-side scripting

Originally designed to create dynamic web pages, PHP's principal focus is server-side scripting. While running the PHP parser with a web server and web browser, the PHP model can be compared to other server-side scripting languages such as Microsoft's ASP.NET system, Adobe ColdFusion, Sun Microsystems' JavaServer Pages, Zope, mod_perl and the Ruby on Rails framework, as they all provide dynamic content to the client from a web server. To more directly compete with the "framework" approach taken by these systems, Zend is working on the Zend Framework - an emerging (as of June 2006) set of PHP building blocks and best practices; other PHP frameworks along the same lines include CakePHP and Symfony.

2.2.1 PHP Templates

Templates are a good way to separate clearly PHP and HTML code. It needs a file called *xxx.php* for each structure we will use. There, it will be all PHP code that this page will need, like variables, functions, loops, imports…
On the other side it needs a new file called *xxx.tpl* when we can find all HTML code of the page. Here it has tags in *{xxxxxx}* format to say to PHP where the variables are.
In the following code we can see the difference between a page developed with templates and without it.

dataLoader.php

```
<body>
 <div id="header">
    <h1><a href="index.php"><span>Student's Watcher</span></a></h1>
  </div>
     <div id="menu">
      <ul>
        <li><a href="#">Show Courses</a>
      <ul>
<?php
$db=new DataBase();
  $rs=$db->FindAllCourses();
  $i=0;
  while($row = mysql_fetch_object($rs))
  {
?>
<li><a href="./ShowCourse.php?courseId=<?=$row->courseid?>"><?=$row->coursename?>
</a></li>
<?
  }
?>
     </ul>
</li>
    <li><a href="./DataLoader.php">Data Loader</a></li>
     </ul>
    </div>
 <div id="content">
<h2>Data Loader</h2>
<form action="<?=$_SERVER['PHP_SELF']?>" method="post" enctype="multipart/form-data">
   <div class="formUpload">
          <label for="search">Select Document:</label>
          <input class="text" type="file" name="search" size="35" id="search" />
          <input class="submit" type="submit" class="goBtn" name="btnSearchDocument"
Value="Ok" />
        </div>
    </form>
 </div></body>
```

Market in black words we can see the PHP code inside the HTML code. In the next code the black words will be changed for Templates tags. And the old code will be in another page.

Dataloader.tpl

```
<body>
 <div id="header">
    <h1><a href="index.php"><span>Student's Watcher</span></a></h1>
  </div>
     <div id="menu">
      <ul>
        <li><a href="#">Show Courses</a>
      <ul>
            {MENUSHOWCOURSES}
     </ul>
       </li>
        <li><a href="./DataLoader.php">Data Loader</a></li>
        <li><a href="./installer.php?index=1">Setup</a></li>
     </ul>
    </div>
 <div id="content">
<h2>Data Loader</h2>
<form action="{PHPSELF}" method="post" enctype="multipart/form-data">
 <div class="formUpload">
          <label for="search">Select Document:</label>
          <input class="text" type="file" name="search" size="35" id="search" />
          <input class="submit" type="submit" class="goBtn" name="btnSearchDocument"
Value="Ok" />
        </div>
    </form>
 </div>
</body>
```

menuShowCourse.tpl

```
<!-- BEGIN SHOW -->
            <li><a href="./ShowCourse.php?courseId={COURSEID}">{COURSENAME}</a></li>
<!-- END SHOW -->
```

dataLoader.php

```
include_once("template.class.php");
$tpl=new Template();
$tpl->set_file("html_tp", "Templates/Dataloader/dataloader.tpl");
$tpl->set_file("menu_tp", "Templates/Menu/menuShowCourses.tpl");
$tpl->set_block("menu_tp", "SHOW", "show");
$db=new DataBase();
  $rs=$db->FindAllCourses();
  while($row = mysql_fetch_object($rs))
  {
     $tpl->set_var("COURSEID", $row->courseid);
     $tpl->set_var("COURSENAME", $row->coursename);
     $tpl->parse("show", "SHOW", true);
  }
$tpl->set_var("PHPSELF", $_SERVER['PHP_SELF']);
$tpl->parse("MENUSHOWCOURSES", "menu_tp");
$tpl->parse("HTML", "html_tp");
```

```
$tpl->p("HTML");
?>
```

2.3 HTML and XHTML

HTML

In computing, HyperText Markup Language (HTML) is the predominant markup language for the creation of web pages. It provides a means to describe the structure of text-based information in a document — by denoting certain text as headings, paragraphs, lists, and so on — and to supplement that text with interactive forms, embedded images, and other objects. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

HTML is also often used to refer to content of the MIME type text/html or even more broadly as a generic term for HTML whether in its XML-descended form (such as XHTML 1.0 and later) or its form descended directly from SGML (such as HTML 4.01 and earlier).

XHTML

The Extensible HyperText Markup Language, or XHTML, is a markup language that has the same depth of expression as HTML, but a stricter syntax. Whereas HTML is an application of SGML, a very flexible markup language, XHTML is an application of XML, a more restrictive subset of SGML. Because they need to be well-formed (syntactically correct), XHTML documents allow for automated processing to be performed using a standard XML library—unlike HTML, which requires a relatively complex, lenient, and generally custom parser (though an SGML parser library could possibly be used). XHTML can be thought of as the intersection of HTML and XML in many respects, since it is a reformulation of HTML in XML. XHTML 1.0 became a World Wide Web Consortium (W3C) Recommendation on January 26, 2000. XHTML 1.1 became a W3C recommendation May 31, 2001

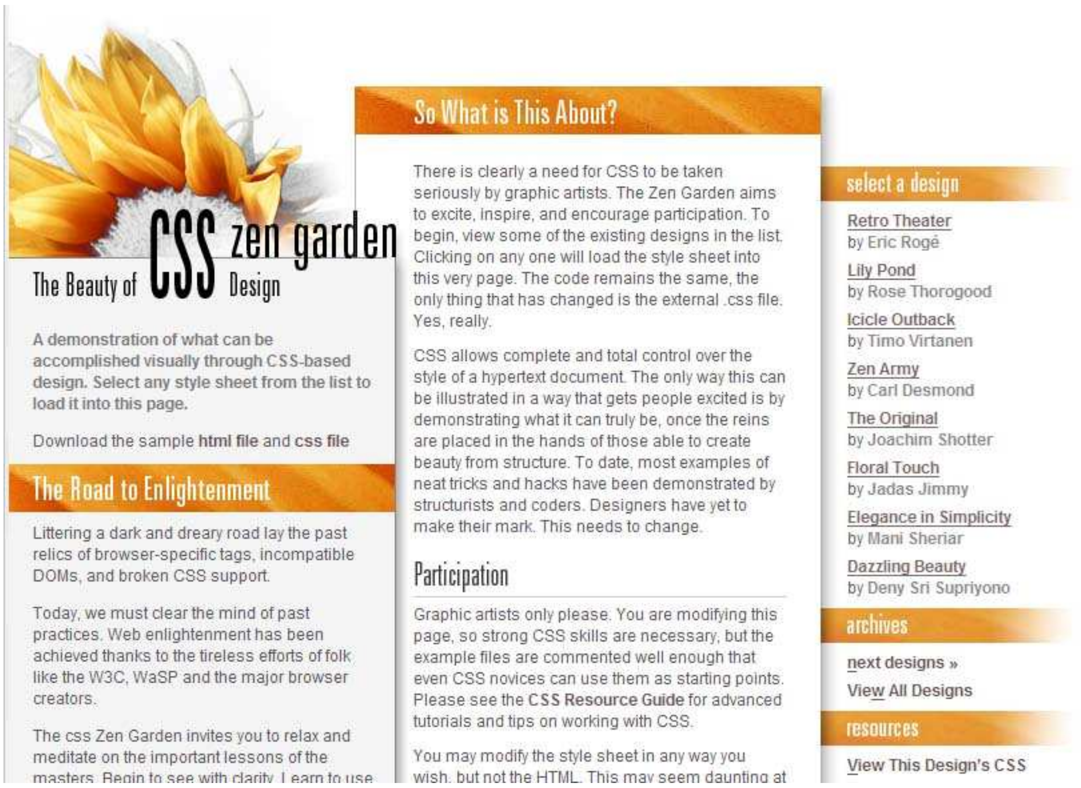
Differences from HTML

The changes from HTML to first-generation XHTML 1.0 are minor and are mainly to achieve conformance with XML. The most important change is the requirement that the document must be well formed and that all elements must be explicitly closed as required in XML. In XML, all element and attribute names are case-sensitive, so the XHTML approach has been to define all tag names to be lowercase. This contrasts with some earlier established traditions which began around the time of HTML 2.0, when many used uppercase tags. In XHTML, all attribute values must be enclosed by quotes (either 'single' or "double" quotes may be used). In contrast, this was sometimes optional in SGML, and hence in HTML, where quotes may be omitted in some circumstances. XML dispensed with the intricate rules for determining when quotes were required or when they could be omitted by simply requiring them in all cases . All elements must also be explicitly closed, including empty (aka singleton) elements such as img and br. This can be done by adding a closing slash to the start tag: <img /> and <br />. Attribute minimization (e.g., <option selected>) is also prohibited as the attribute “selected” contains no explicit value; instead, use <option selected="selected">. More differences are detailed in the W3C XHTML 1.0 recommendation

2.4 CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in a markup language. It Is most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document, including SVG and XUL. The CSS specifications are maintained by the World Wide Web Consortium (W3C).

CSS has various levels and profiles. Each level of CSS builds upon the last, typically adding new features and are typically denoted as CSS1, CSS2, and CSS3. Profiles are typically a subset of one or more levels of CSS built for a particular device or user interface. Currently there are profiles for mobile devices, printers, and television sets. Profiles should not be confused with media types which were added in CSS2.
A useful example of the power of CSS could be the next page (www.zengarden.com), showed first with a nice style (img 17) and then with the styles deactivated (img 18).



A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample **html file** and **css file**

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the **CSS Resource Guide** for advanced tutorials and tips on working with CSS.

You may modify the style sheet in any way you wish, but not the HTML. This may seem daunting at

select a design

Retro Theater
by Eric Rogé
Lily Pond
by Rose Thorogood
Icicle Outback
by Timo Virtanen
Zen Army
by Carl Desmond
The Original
by Joachim Shotter
Floral Touch
by Jadas Jimmy
Elegance in Simplicity
by Mani Sheriar
Dazzling Beauty
by Deny Sri Supriyono

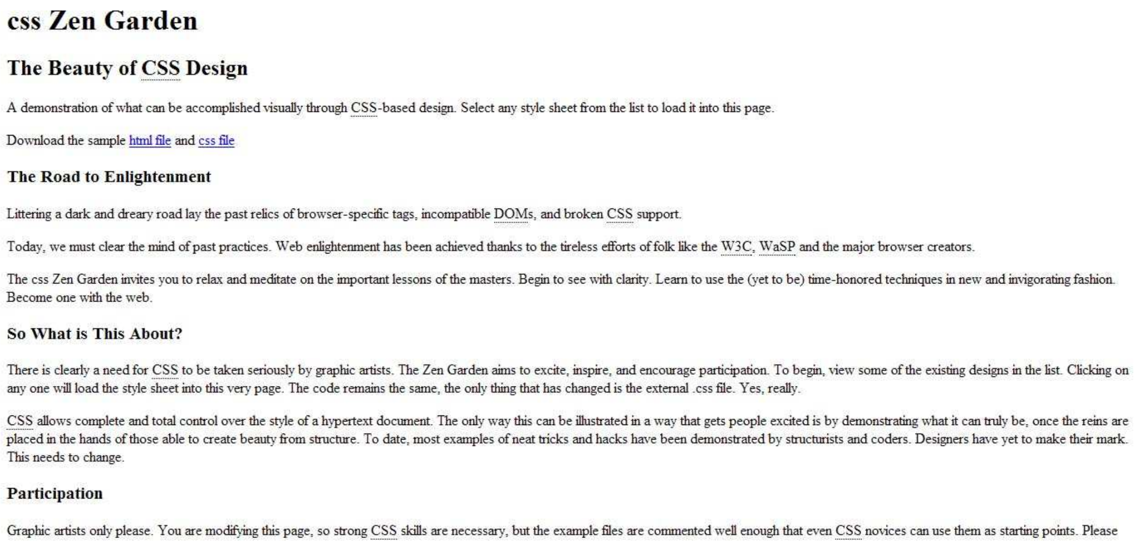
archives

next designs »
View All Designs

resources

View This Design's CSS

(img 17)



css Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please

(img 18)

2.5 JavaScript

JavaScript is a prototype-based scripting language with a syntax loosely based on C. Like C, the language has no input or output constructs of its own. Where C relies on standard I/O libraries, a JavaScript engine relies on a host environment into which it is embedded. There are many such host environment applications, of which web technologies are the best-known examples. These are examined first.

One major use of web-based JavaScript is to write functions that are embedded in or included from HTML pages and interact with the Document Object Model (DOM) of the page to perform tasks not possible in HTML alone. Some common examples of this usage follow.

- Opening or popping up a new window with programmatic control over the size, position and 'look' of the new window (i.e. whether or not the menus, toolbars etc are visible).
- validation of web form input values to make sure that they will be accepted before they are submitted to the server.
- Changing images as the mouse cursor moves over them: This effect is often used to draw the user's attention to important links displayed as graphical elements.

The DOM interfaces in various browsers differ and don't always match the W3C DOM standards. Rather than write different variants of a JavaScript function for each of the many browsers in common use today, it is usually possible, by carefully following the W3C DOM Level 1 or 2 standards, to provide the required functionality in a standards-compliant way that most browsers will execute correctly. Care must always be taken to ensure that the web page degrades gracefully and so is still usable by any user who:

- has JavaScript execution disabled - for example as a security precaution
- has a browser that does not understand the JavaScript - for example on a PDA or mobile phone
- is visually or otherwise disabled and may be using an unusual browser, a speech browser or may have selected extreme text magnification. For more information on this, see the Web Accessibility Initiative

Other examples of JavaScript interacting with a web page's DOM have been called DHTML and SPA.
A different example of the use of JavaScript in web pages is to make calls to web and web-service servers after the page has loaded, depending upon user actions. These calls can obtain new information, which further JavaScript can merge with the existing page's DOM so that it is displayed. This is the basis of **Ajax** programming. PnP JavaScript design pattern was adopted gradually after commonly use of Ajax to reduce JavaScript maintenance cost.

2.6 AJAX

Ajax, shorthand for Asynchronous JavaScript and XML, is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is meant to increase the web page's interactivity, speed, and usability.

The Ajax technique uses a combination of:

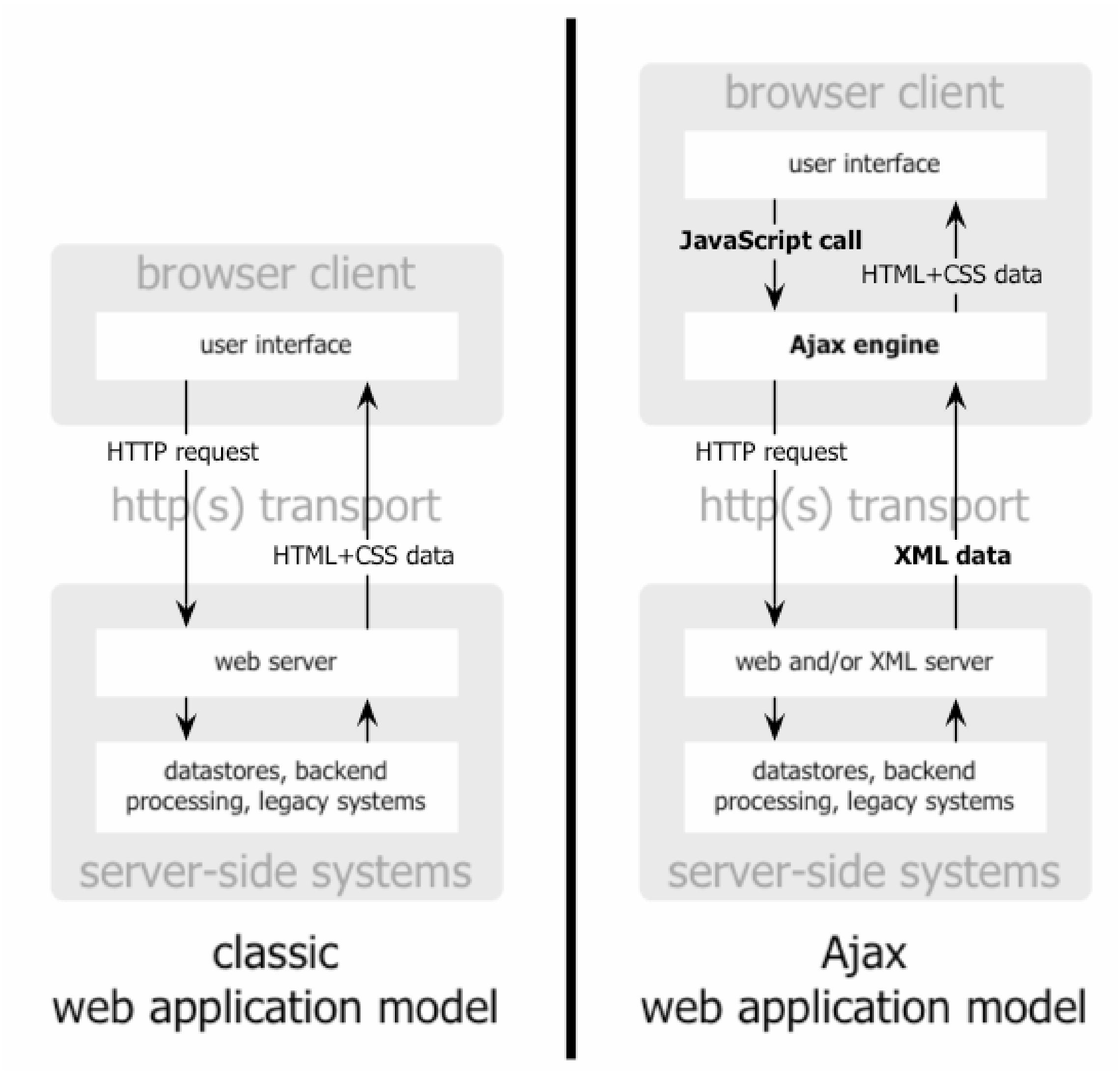
- XHTML (or HTML) and CSS, for marking up and styling information.
- The DOM accessed with a client-side scripting language, especially ECMAScript implementations such as JavaScript and JScript, to dynamically display and interact with the information presented.
- The XMLHttpRequest object is used to exchange data asynchronously with the web server. In some Ajax frameworks and in certain situations, an IFrame object is used instead of the XMLHttpRequest object to exchange data with the web server, and in other implementations, dynamically added <script> tags may be used.
- XML is sometimes used as the format for transferring data between the server and client, although any format will work, including preformatted HTML, plain text, JSON and even EBML. These files may be created dynamically by some form of server-side scripting.

The core justification for AJAX style programming is to overcome the page loading requirements of HTML/HTTP-mediated web pages. AJAX creates the necessary initial conditions for the evolution of complex, intuitive, dynamic, data-centric user interfaces in web pages - the realization of that goal is still a work in progress.

Web pages, unlike native applications, are loosely coupled, meaning that the data they display are not tightly bound to data sources and must be first marshalled into an HTML page format before they can be presented to a user agent on the client machine. For this reason, web pages have to be re-loaded each time a user needs to view different datasets. By using the XmlHttpRequest object to request and return data without a re-load, a programmer by-passes this requirement and makes the loosely coupled web page behave much like a tightly coupled application, but with a more variable lag time for the data to pass through a longer "wire" to the remote web browser.

For example, in a classic desktop application, a programmer has the choice of populating a tree view control with all the data needed when the form initially loads, or with just the top-most level of data - which would load quicker, especially when the dataset is very large. In the second case, the application would fetch additional data into the tree control depending on which item the user selects. This functionality is difficult to achieve in a web page without AJAX. To update the tree based on a user's selection would require the entire page to re-load, leading to a very jerky, non-intuitive feel for the web user who is browsing the data in the tree.

Comparison of classic and Ajax web application model.



(img 20)

2.7 MySQL

In order to use the project in the Web has been decided to use *MySQL*. It could be another one, like *SQLServer*, but it was chosen this one because it is free.
It is used *MySQL 4.1,* it could be possible to use *MySQL 5.0* but this project does not need the new advantages of this one. With older version it is enough.

MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, which holds the copyright to most of the code base. This is similar to the JBoss model and how the Free Software Foundation handles copyright in its projects, and dissimilar to how the Apache project does it, where the software is developed by a public community, and the copyright to the codebase is owned by its individual authors. The company develops and maintains the system, selling support and service contracts, as well as proprietary-licensed copies of MySQL, and employing people all over the world who collaborate via the Internet. MySQL AB was founded by David Axmark, Allan Larsson, and Michael "Monty" Widenius.
The MySQL company also sells another DBMS, MaxDB, which is from an unrelated codebase.

History

- MySQL was first released internally on May 23, 1995
- Windows version released on January 8, 1998 for Windows 95 and NT
- Version 3.23: beta from June 2000, production release January 2001
- Version 4.0: beta from August 2002, production release March 2003 (unions)
- Version 4.1: beta from June 2004, production release October 2004 (r-trees, subqueries)
- Version 5.0: beta from March 2005, production release October 2005 (cursors, stored procedures, triggers, views, XA transactions)
- Version 5.1: currently pre-production (since November 2005) (event scheduler, partitioning, plugin API, row-based replication, server log tables)
- Version 5.2 will include foreign key support for all storage engines (at the moment only InnoDB supports this)

3. DETAILED DESCRIPTION

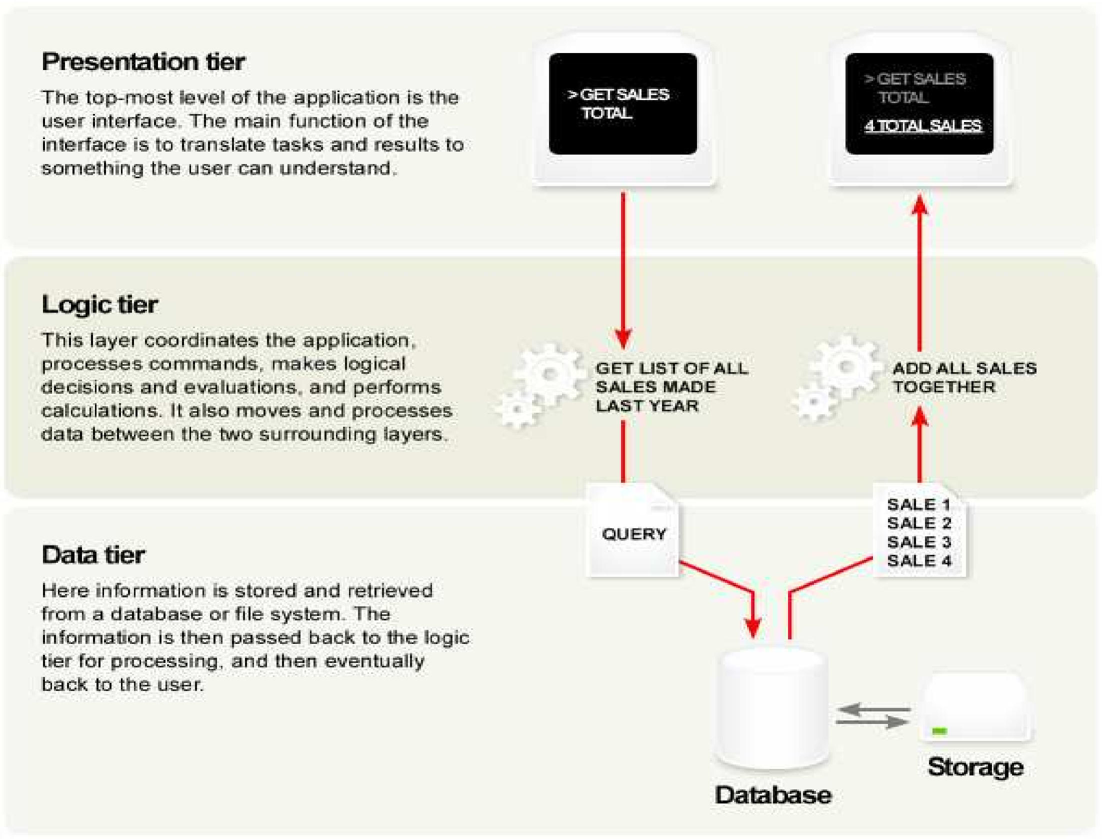
3.1 Three tier structure

Three-tier is a client-server architecture in which the user interface, functional process logic ("business rules"), data storage and data access are developed and maintained as independent modules, most often on separate platforms. The term "three-tier" or "three-layer", as well as the concept of multitier architectures, seems to have originated within Rational Software. (Citation Needed)

The three-tier model is considered to be a software architecture and a software design pattern.

Apart from the usual advantages of modular software with well defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or technology change. For example, a change of operating system from Microsoft Windows to Unix would only affect the user interface code.

Overview of a three-tier application **(img 19)**



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

(img 21)

Comparison with the MVC architecture

At first glance three-tiers may seem similar to the Model-view-controller (MVC) concept, however topologically they are a different. A fundamental rule in a three tier architecture is the Client tier never communicates directly with the Data tier; in a three-tier model all communication must pass through the Middleware tier. Conceptually the three-tier architecture is linear. However, the MVC architecture is triangular: the Controller updates the Model, and the View's updates come directly from the Model. Historically the three-tier architecture concept comes from observations of distributed systems (for example, web applications) where the Client, Middleware and Data tiers run on physically separate platforms. Whereas MVC comes from an era of observations of applications that ran on a single graphical workstation; MVC was applied to distributed applications much later in its history (see Model 2).

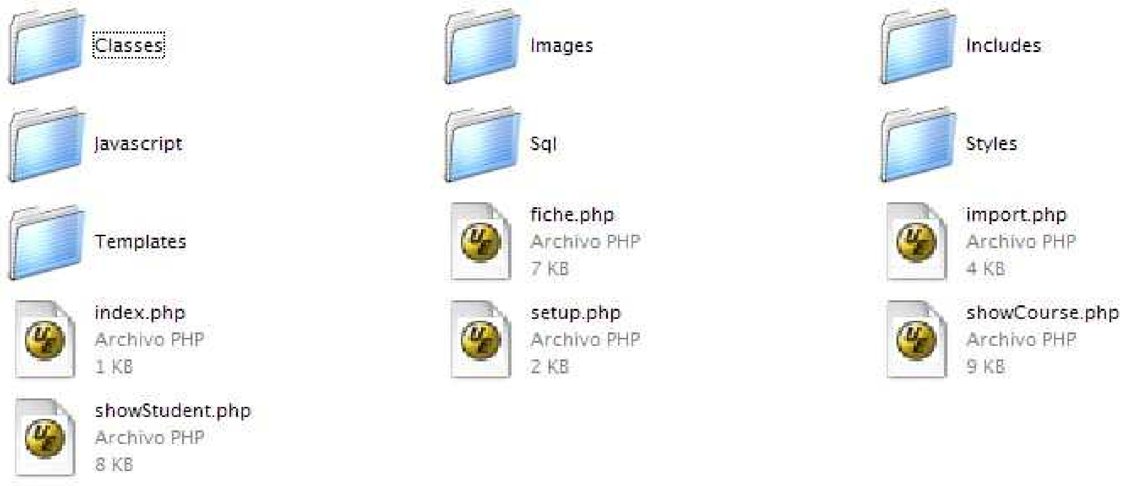
Web Development usage

In the Web development field, three-tier is often used to refer to Websites, commonly Electronic commerce websites, which are built using three tiers:

1. A front end Web server serving static content
2. A middle dynamic content processing and generation level Application server, for example Java EE platform.
3. A back end Database, comprising both data sets and the Database management system or RDBMS software that manages and provides access to the data.

3.2 Classes and directories

On the following image (img 20) we can see the Student's Watcher directories structure.



Classes
Images
Includes
Javascript
Sql
Styles
Templates
fiche.php
Archivo PHP
7 KB
import.php
Archivo PHP
4 KB
index.php
Archivo PHP
1 KB
setup.php
Archivo PHP
2 KB
showCourse.php
Archivo PHP
9 KB
showStudent.php
Archivo PHP
8 KB

(img 22)

In the first folder there are the classes of all objects we need to simulate the database in an object structure:

- *Comment.class.php*
- *Course.class.php*
- *Student.class.php*
- *Subscription.class.php*
- *Weekeval.class.php*

There are, also, *Csvhandler.class.php*, *Database.class.php* and *Template.class.php.*
In the *images* folder we can find all images from the students to show on the application.
Includes folder contains two files called *Config.inc.php* and *Functions.inc.php.* The first one includes information to configure the database:

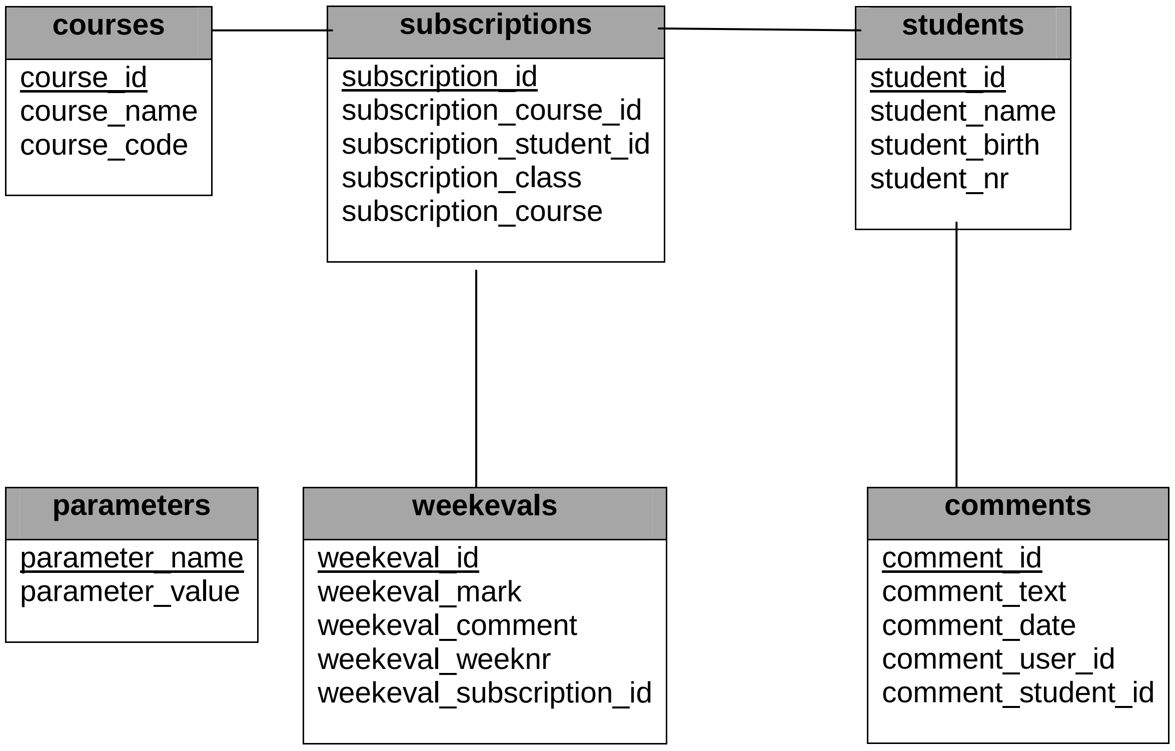
```
<?php
  // database
  $MYSQLCONFIG = array(
    'host'=>'localhost',
    'user'=>'root',
    'password'=>'root',
    'database'=>'sw'
  );
?>
```

It is able to change these fields using the setup page in Student’s Watcher or modifying directly from this file.
The second one, *Functions.inc.php* includes functions like data validation, url builder or to show errors.
The next folder, *Javascript*, has all classes needed to use the JavaScript properties added. It has: *functions.js, import.js, page.js, prototype.js* and *showcourse.js.*
Sql folder only contains *dbsetup.sql* and there are the sql instructions to create all tables in the database.
Styles directory is used to group all CSS files. It has inside a folder with all icons and images used in these files. There is a common style for all application called *common.css.* To use a specific style we have *fiche.css, loader.css,* and *setup.css.*
The last folder is used to group all templates files. Inside it are *error.tpl, fiche.tpl, import.tpl, index.tpl, page.tpl, setup.tpl, showcourse.tpl* and *showstudent.tpl.*

3.3 Data Tier: Database

3.3.1 Class Diagram

The following diagram is the Entity-Relation Model of Student Watcher. Here it can see the relation between all application classes.



3.3.2 Tables structure

This is the structure of the tables in the database. On the left there is the name of the table, on the right, attributes of this one.

weekevals:	weekeval_id, weekeval_mark**,** weekeval_comment**,** weekeval_weeknr**,** weekeval_subscription_id
contains:	*marks and comments about one subscription*
subscriptions:	subscription_id, subscription_class, subscription_course, subscription_course_id, subscription_student_id
contains:	*the class, for each course where student is registered*
students:	student_id, student_name, student_birth, student_nr
contains:	*information about student*
courses:	course_id, course_name, course_code
contains:	*information about courses*
parameters:	parameter_id, parameter_name, parameter_value
contains:	*parameters to use internally*
comments:	comment_id, comment_text, comment_date, comment_user_id, comment_student_id.
contains:	*text with date about students*

---- → shows foreign key

___→ shows primary key

3.3.3 MySQL Tables

```
DROP TABLE IF EXISTS `courses`;
DROP TABLE IF EXISTS `students`;
DROP TABLE IF EXISTS `subscriptions`;
DROP TABLE IF EXISTS `weekevals`;
DROP TABLE IF EXISTS `comments`;
DROP TABLE IF EXISTS `parameters`;

CREATE TABLE IF NOT EXISTS `courses` (
  `course_id` int(10) NOT NULL auto_increment,
  `course_name` varchar(255) NOT NULL default '',
  `course_code` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `students` (
  `student_id` int(10) NOT NULL auto_increment,
  `student_name` varchar(255) NOT NULL default '',
  `student_birth` date NOT NULL default '0000-00-00',
  `student_nr` varchar(255) NOT NULL default '',
  PRIMARY KEY  (`student_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `subscriptions` (
  `subscription_id` int(10) NOT NULL auto_increment,
  `subscription_course_id` int(10) NOT NULL default '0',
  `subscription_student_id` int(10) NOT NULL default '0',
  `subscription_class` varchar(20) NOT NULL default '',
  `subscription_course` varchar(9) NOT NULL default '0',
  PRIMARY KEY  (`subscription_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `weekevals` (
  `weekeval_id` int(10) NOT NULL auto_increment,
  `weekeval_mark` int(2) NOT NULL default '0',
  `weekeval_comment` text NOT NULL,
  `weekeval_weeknr` int(2) NOT NULL default '0',
  `weekeval_subscription_id` int(10) NOT NULL default '0',
  PRIMARY KEY  (`weekeval_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `comments` (
  `comment_id` int(10) NOT NULL auto_increment,
  `comment_text` varchar(200) NOT NULL default '',
  `comment_date` date NOT NULL default '0000-00-00',
  `comment_user_id` int(10) NOT NULL default '0',
  `comment_student_id` int(10) NOT NULL default '0',
  PRIMARY KEY  (`comment_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `parameters` (
  `parameter_name` varchar(50) NOT NULL default '',
  `parameter_value` text NOT NULL,
  PRIMARY KEY  (`parameter_name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `parameters` VALUES ('complete', '1');
INSERT INTO `parameters` VALUES ('loaded', '0');
```

Parameter *complete* is used to says that all tables are created succefully.
Loaded is used to knows if these tables are empty or not.

3.3.4 Database.class.php

This is the biggest class of the project, and here we find de independence between business class and database. Here are all functions we need to communicate our application with the database. Inserts, deletes, updates, open connection …
If we never change de database provider, we only should modify this class. All other classes would be intact.
The constructor of this class is who watch if the database is ready to use and it opens the other pages if it is all right.

3.4 Business Tier: PHP Classes

The business tier contains all PHP and JavaScript classes used to build HTML, but not HTML specific classes.
Student’s Watcher can works with JavaScript disabled. JavaScript was added later to be more complete. But it is not necessary to use the application.
In the same way, AJAX is an extra option. It has been used to be able to use the direction keys to move around the table and to refresh the student information in a faster way because the entire web page does not have to be reloaded.

3.4.1 Important code

The main classes of the business tier are *showStudent.php, showCourse.php, import.php, setup.php, index.php* and *fiche.php*. The two first are really similar. Its have on the beginning the AJAX actions. AJAX actions are only in these pages.
This is the code used in these classes:

```
// AJAX actions
if (isset($_REQUEST['ajaxWeekevalId'])) {
  // incoming
  $ajaxWeekevalId   = $_REQUEST['ajaxWeekevalId'];
  $ajaxWeekeval     = $db->getWeekevalById($ajaxWeekevalId);
  $ajaxSubscription = $db->getSubscriptionById($ajaxWeekeval->getSubscriptionId());
  // get student
  $ajaxStudent      = $db->getStudentById($ajaxSubscription->getStudentId());
  //outgoing
  echo "comment=".$ajaxWeekeval->getComment()."&mark=".$ajaxWeekeval->getMark()."
       &studentName=".$ajaxStudent->getName()."&studentClass=".$ajaxSubscription-
>getClass()."
       &studentAge=".$ajaxStudent->getAge()."&studentPhoto=".$ajaxStudent->getPhoto()."
       &weekevalId=".$ajaxWeekevalId;
  die();
 }
```

Here it sees if is set the request with AJAX parameter. If it is true it takes all parameters it needs. Then return all new parameters to JavaScript, and it will use these. Changing name, age, photograph… of the student.
Finally it dies, because if we do not do it the application continues executing the rest of the page, and we do not want it.

In the classes folder we have a retort of our database. One class for each important table.
A good example of this kind of class could be the following:

```
<?php
  class Course {
    var $name;
    var $id;
// constructor
    public function Course($name="", $id=-1) {
      $this->name = $name;
      $this->id   = $id;
    }
// getters
    public function getName() {
      return $this->name;
    }
    public function getId() {
      return $this->id;
    }
 // setters
    public function setName($name) {
      $this->name=$name;
    }
    public function setId($id) {
      $this->id=$id;
    }
  }
?>
```

Then, when we want to use information about a course, for example, we ask to *Database.class.php* and normally it should return a Course Object. On this Object we will be able to use the getters and setters to get the information or to modify it.
A simple example to show how it uses Templates in PHP could be the following code form *index.php:*

```
<?php
  require_once "Classes/DataBase.class.php";
  require_once "Classes/Template.class.php";

  $db = new DataBase();
  $courses          = $db->getCourses();

  // BUILD PAGE
  $tpl = new Template("templates/");
  // set files and blocks
  $tpl->set_file("page_tp", "page.tpl");
  $tpl->set_block("page_tp", "LICOURSE", "licourses");
  // generate page
  foreach ($courses as $course) {
    $tpl->set_var("NAME_LICOURSE", $course->getName());
    $tpl->set_var("URL_LICOURSE",  "showCourse.php?courseId=".$course->getId());
    $tpl->parse("licourses",  "LICOURSE", true);
  }
  $tpl->set_var("TITLE", "StudentsWatch - view course");
  $tpl->parse("HEADEXTRA", "HEADBLOCK");
  $tpl->parse("CONTENT", "CONTENTBLOCK");
  $tpl->pparse("htmlcode", "page_tp");
?>
```

Here we can see how first we create a Template, and using *set_file* and *set_block* we starts to reference the html code in the *page.tpl* . Then we generate the page setting the variables. We can do loops like the *foreach* in the example, and at the end of it we have to parse it with the function *parse.*
In the *page.tpl* we find tags called, {NAME_LICOURSE}, {URL_LICOURSE} …. Which will takes the values of the *$courrse->getName()* and *showCourse.php?courseId=".$course->getId().*
Finally we parse the HEAD and the CONTENT blocks to write the HTML code in the page.

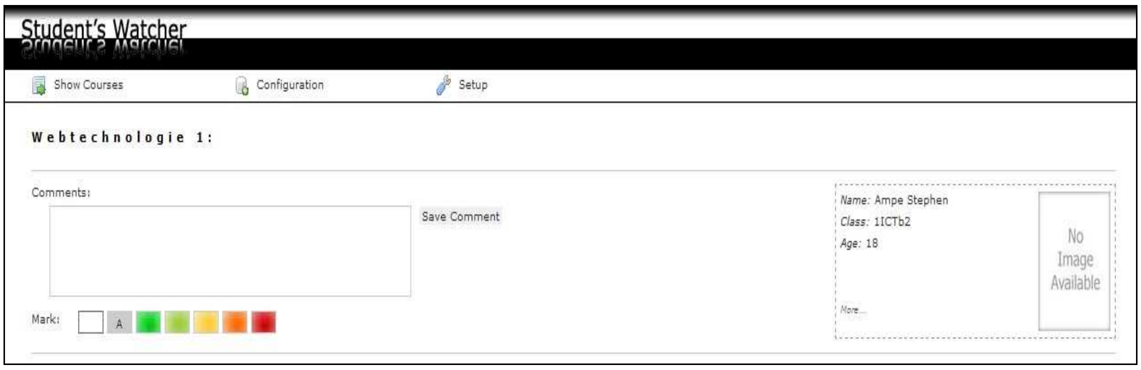
3.5 Presentation Tier: HTML & CSS

Using templates, the HTML pages are generated combining the PHP classes and .tpl files. PHP sets into HTML the values of the variables, and generate dynamically some HTML parts.
In the presentation tier we find all classes called *xxx.tpl.* One example of these kind of files is *page.tpl:*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl">
<head>
  <title>{TITLE}</title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <link rel="stylesheet" type="text/css" href="Styles/common.css" />
  <script type="text/javascript" src="Javascript/prototype.js"></script>
  <script type="text/javascript" src="Javascript/functions.js"></script>
  <script type="text/javascript" src="Javascript/page.js"></script>
{HEADEXTRA}
</head>
<body>
  <div id="header">
    <h1><a href="#">Student's Watcher</a></h1>
  </div>
  <div id="menubar">
    <ul id="mainmenu" class="dropdown">
      <li>
        <a class="lnkshowcourse" href="showCourse.php">Show Courses</a>
        <ul id="courselist">
<!-- BEGIN LICOURSE -->
          <li><a href="{URL_LICOURSE}">{NAME_LICOURSE}</a></li>
<!-- END LICOURSE -->
        </ul>
      </li>
      <li><a class="lnkimport" href="import.php">Configuration</a></li>
      <li><a class="lnksetup" href="setup.php">Setup</a></li>
    </ul>
  </div>
  <div id="content">
{CONTENT}
  </div>
</body>
</html>
```

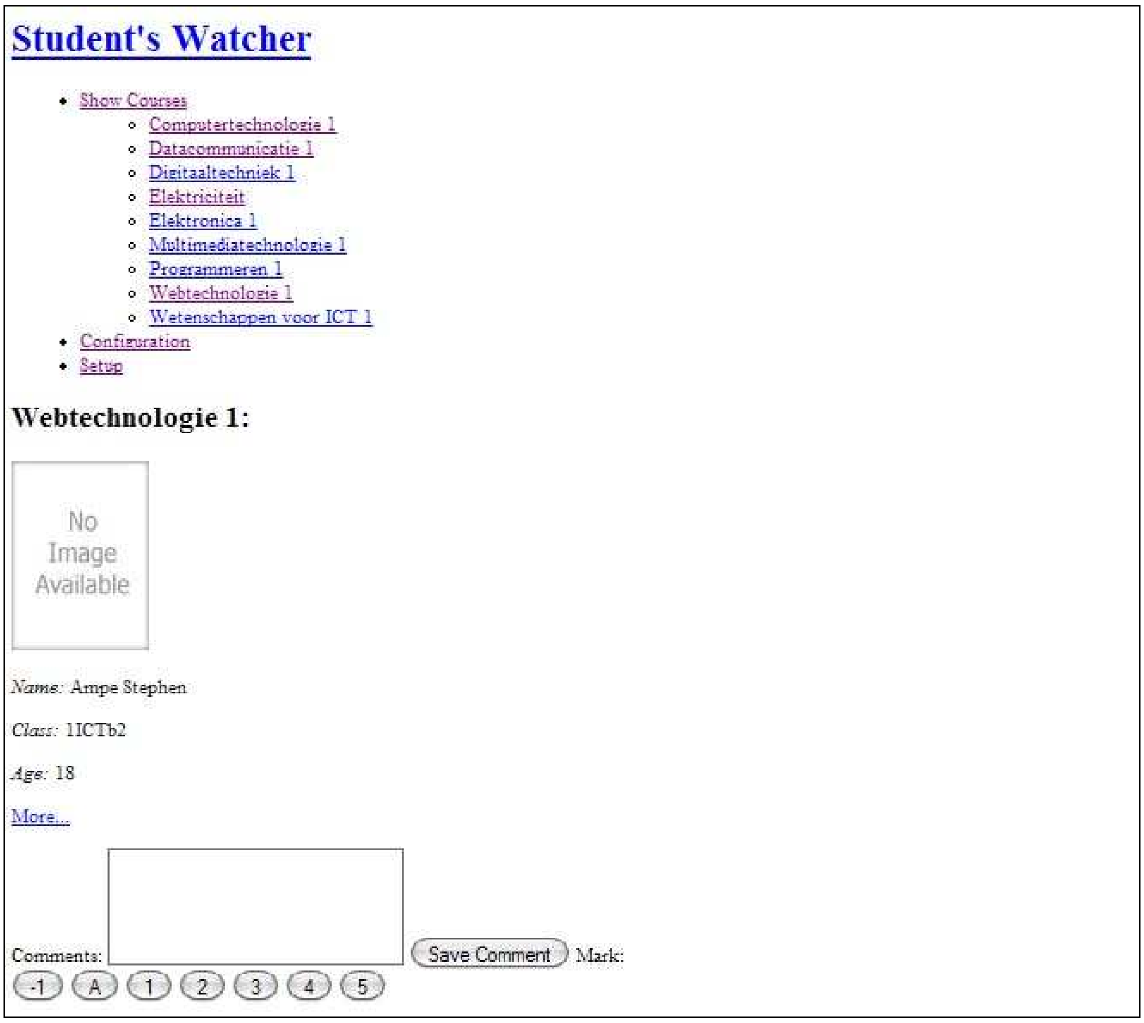
All tags in {XXX} format will be substituted by a value when the page will be generated by PHP.

It is important to observe that the HTML code in Student’s Watcher never has a style inside. The style is totally independent of HTML code. In this way, the HTML code is really simple and then, using CSS we give format to the page.
The following images show the differences using styles or not in Student’s Watcher.
The first one shows a part of HTML code with CSS activated. (img 21)



(img 23)

The second one is exactly the same code but with the CSS deactivated (img24)



Student's Watcher

- Show Courses
  - Computertechnologie 1
  - Datacommunicatie 1
  - Digitaaltechniek 1
  - Elektriciteit
  - Elektronica 1
  - Multimediatechnologie 1
  - Programmeren 1
  - Webtechnologie 1
  - Wetenschappen voor ICT 1
- Configuration
- Setup

Webtechnologie 1:

Name: Ampe Stephen

Class: 1ICTb2

Age: 18

More...

Comments: Mark:

(img 22)

With these two images we can appreciate the power of CSS. For example if sometimes we want to change our web style, like colours, structure of panels …, we only should change our Style Sheet or create a new one with the new style.

4. CONCLUSION

This project wants to be a comparison between two of most important languages used nowadays, *ASPX* and *PHP*. It is not looking for which is the best or the worst language, surely one is better in some ways and the other in other things. It only wants to find these ways and be able to know when we should use one or other.

During this project I have learned a lot about PHP and now I can do a personal comparison between PHP and ASP.net. But I have read some other experiences on internet to do my vision a little bit more objective.
In the internet community I have found more people who prefer PHP than ASP.net. But sometimes it could be because the most of web developers who write in forums are Open Source users and, sometimes, is difficult to know if they are Anti Microsoft or they are objectives. Here I do not want to enter in this kind of discussion, and I have chosen some opinions that I like or I think that are objective. When is better PHP?

- It does not need Windows to work. It can works under Linux, for example which one is more secure than Windows.
- A real developer community has grown up around PHP. This means that bugs are found and fixed quickly.
- PHP is totally free.
- The final code is really clear and simple to read.
- It makes easy to follow the W3C standards.
- It’s permit to design all style in CSS.
- We have a HTML version without styles if we want, to use in a PDA, or other devices which do not read CSS.

On the other hand ASP.net has some advantages to PHP. When is better ASP.net?

- Visual Studio has a really nice debugger.
- We can now manipulate Oracle Database objects directly from within the IDE with the Oracle Developer Tools for VS.NET add-in.
- .NET provides classes for mark up abstraction, meaning that, behind the scenes, it takes care of the various browsers with which you might be connecting to the site

Finally, in my opinion, I guess that maybe ASP.net is more useful to use in a really big projects. Of course we can use too PHP, but for me, this one could be better to use in a small or medium applications.

New version of Student's Watcher

I have started a new version of Student's Watcher which allows saving all years in the database. It is included in the CD Documentation of the project. This version just needs to fix some small problems to works correctly.
It could be really better than the first version because it permits to do the comparison between different years of the students in the same grid and we can see their evolution in a quick way.

Personal opinion

It has been a pleasure for me, to have had this opportunity to learn about these new technologies. Now I have more experience and I have been able to put in practice all the knowledge learned in my studies.
I had read something about CSS before, but I did not know it was so powerful. I would like to work in more projects with these technologies in my future job. I have been working with ASP.net, but for me, is more creative and I like more the other way. On the other hand, I hope that more people work with Open Source, if I can, I will do it.