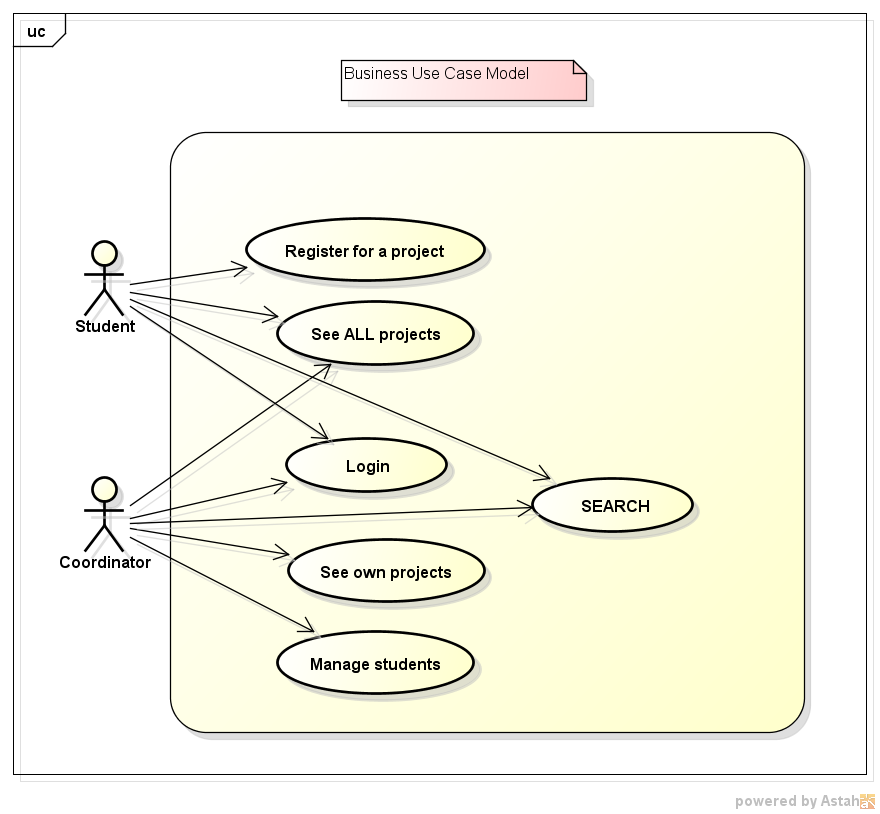
Scope

The scope of the application is to store and manage bachelor projects. Each project has a title, a short description, keywords, a coordinator (with name etc), students working at that project. The students can browse through the list of all projects, and can view the information about the projects. The students and the coordinators can use the search functionality to easily find projects of interest. If a project is available (there are still available spots for that project), the student can apply to it. The project coordinator can view the list of all applications and can either accept or reject an application. If a student is accepted into a project, he is notified and he can no longer apply to other projects. Also, all other applications become void.

The application uses a notification system. Every time an event that concerns a student or a coordinator happens (e.g. a student got rejected for a project), all the parties involved receive a notification. All notifications are displayed in the home page. The notifications remain there until the user dismisses them.



Architecture

For the ‘Bachelor’ application we use the three tier architecture, meaning that we organized the structure in three major parts, each component designed for a specific service:

* **The presentation tier**

**This layer contains the service that provides the graphical user interface (GUI) and interactive windows or application-specific forms. This is accomplished through the use of dynamic front-end technologies, such as:**

* **Bootstrap (HTML5, CSS3) - using this framework proved a decisive factor in time management. We included some ready-made pieces of code into our application, so instead of laboriously working out and writing code, we simply found the right pieces to incorporate. Also, Bootstrap is very consistent, the results are uniform across platforms, meaning that you’ll see the same structure on multiple browsers.**

**Another important feature is responsiveness, if you shift between different screen size and various platforms (from laptop to a tablet or a phone), the content will be adapted effectively.**

* **AJAX – allows web pages to be updated asynchronously by exchanging data with the server. This helped us in keeping the natural flow of events consistent, without reloading the entire page**
* **SVG – defines graphics in XML format and w**ords are more influential on the Internet than pictures. Search engines look at words, specifically keywords. If you upload a JPEG file, you are limiting yourself to the title of your graphic. With SVG coding, you expand on the possibilities. Being vector based, it is scalable and will not distort quality because the system is mathematic.
* **The middle tier**

**This layer consists in acting as the server for client request and workstations. It determines and enforces business rules, such as business algorithms, regulations, data rules which are designed to keep the data structures consistent within a specific database.**

* **PHP -** Server-side scripting language, used for development of dynamic and interactive web pages. The code can simply be mixed with html code or it can be used in combination with various engines and web frameworks. After the PHP code is interpreted and executed, the server sends the results to its clients.
* **The data tier**

**The data service layer interacts with persistent data usually stored in a data base. It can be accessed through the business service layer (middle tier) and helps in resource sharing and allows information to be configured without installing the DBMS libraries on each client.**

* **XML - the markup language that defines a set of rules for encoding documents and the emphasis is put on simplicity. In our case, xml was used as storage for all the relevant data that is presented and manipulated.**

With three tiers or parts, each part can be developed concurrently by different members of the team in different languages from the other tier developers. Because the programming for a tier can be changed or relocated without affecting the other tiers, it was easier for us to use **GITHUB** to continually evolve the application as new needs and opportunities were presented.



XML tools

Php language has a lot of libraries for XML manipulation, the most common are:

* DOM — Document Object Model
* libxml
* SDO — Service Data Objects
* SDO-DAS-Relational — SDO Relational Data Access Service
* SDO DAS XML — SDO XML Data Access Service
* SimpleXML
* WDDX
* XMLDiff — XML diff and merge
* XML Parser
* XMLReader
* XSL

All of this libraries are built-in PHP 5.4+, and are really well documented. Some offer read only function, some offer basic api to access XML, some offer other functionalities.

For our project we choose to use SimpleXML to read, query and modify XML document, and XSL, to perform transformation on XSL Template file.

# SimpleXML

The SimpleXML extension provides a very simple and easily usable toolset to convert XML to an object. This extension requires the libxml PHP extension.

CODE: --enable-libxml is required (from php 5.2 libxml is enabled by default).

The SimpleXML extension requires PHP 5.

Simple XML limitation is the possibility to use only Xpath, and not XQuery, but the object structure compensate this limitation really well.

## Functions

Those are the most important and most used functions

### simplexml\_load\_file

simplexml\_load\_file ( string $filename [, string $class\_name = "SimpleXMLElement" [, int $options = 0 [, string $ns = "" [, bool $is\_prefix = false ]]]] )

This function take a load a XML file into a SimpleXML object. The new created object has methods to access, modify, query, store the XML.

### simplexml\_load\_string

simplexml\_load\_string ( string $data [, string $class\_name = "SimpleXMLElement" [, int $options = 0 [, string $ns = "" [, bool $is\_prefix = false ]]]] )

This function perform the exactly same action of simplexml\_load\_file, but he take a string as input. It’ important to know that BOTH load\_file and load\_string requires well-formed XML, or will be an error.

From now we assume exist an object $xml created with simplexml\_load\_\*, for example

$xml = simplexml\_load\_file('file.xml');

### xpath

xpath ( string $path )

xpath is a method of SimpleXML object, to execute an xpath query. The result is stored in an array of SimpleXML object. In this example we get all user that has email address mail@mail.com

$result = $xml->xpath('/users/user[email="mail@mail.com"]’);

The result array five us the possibility to easely have some information, for example

sizeof($result)

gives us the length of array (0 = empty, 1 = 1 element, …, n = n element)

If we want to explore all the xpath element, we can use foreach statement, for example

foreach($result as $user)

{

echo $user->email;

}

Each $user in the array still an SimpleXML object, each variable is the XML content of the object.

### addChild

addChild ( string $name [, string $value [, string $namespace ]] )

addChild is a method of SimpleXML object, to add a child to the selected object. The result is a new reference to SimpleXML object where the new child is a branch of the XML tree.

If we want to insert a new user, with an email, for example

$newUser = $xml->addChild('user');

$newUser ->addChild('email', 'user@mail.com');

Modify a child is very simpe, we don’t need to use a specific function for that. Using the example before, to modify the new inserted element mail, we can simple

$newUser->email = 'user\_new\_email@mail.com'

Also remove child is very simple. Using the example before, to erase the new inserted user, we perform

unset($newUser)

### saveXML

Save XML is an alias function for asXML(), it is used because the name is more user friendly.

### asXML

asXML ([ string $filename ] )

The asXML function is used in two ways. If we specify a filename, the function store the XML content in well-formed XML file.

$xml->asXML('file.xml');

$xml-> saveXML('file.xml');

# XSL

The XSL extension implements the XSL standard, performing XSLT transformations using the libxslt library. This extension requires the libxml PHP extension. This extension uses libxslt library. PHP 5 includes the XSL extension by default.

## Function

To perform an XLST transformation we need several component. We need a well-formed XSL file. We need a well-formed XML file. We need a XSLTProcessor.

**importStyleSheet**

XSLTProcessor::importStylesheet ( object $stylesheet )

This method imports the XSL stylesheet into the XSLT Processor for transformations.

### transformToXML

string XSLTProcessor::transformToXML ( DOMDocument $doc )

This method transforms the source node to a string applying the XSL stylesheet given.

### Usage Example

Perform the XSLT transformation is really easy, this is an example of use.

$data = simplexml\_load\_file ('file.xml');

$xsl = simplexml\_load\_file('file.xsl');

$xslt = new XSLTProcessor;

$xslt->importStyleSheet($xsl);

echo $xslt->transformToXML($data);