David Brossard

Developing a fotoblog platform
Portugalmail S.A.

End of placement paper - 2004/2005
Developing a fotoblog platform
Portugalmail S.A.

David Brossard

End of placement paper – 2004-2005

FEUP tutor: Andre Restivo
Portugalmail tutor: Sérgio Carvalho

Faculdade de Engenharia da Universidade do Porto
Licenciatura em Engenharia Informática e Computação
To my father
Abstract

As broadband access spreads all over the Western countries and the web becomes accessible and easy to use, myriads of consumers have massively arrived, eager to use the newest technologies, among which blogs.

Blogs have been the latest trend online the past couple of years, rocketing sites such as Blogger and Blog.com to the top sites. Literally hundreds of blog providers have flourished and now battle for their share of the market.

A blog is «a website offering the possibility to create an online diary written by one or several users» (Wikipedia), «a website with text and/or images updated on a regular basis by a unique author who shares his thoughts while readers can write comments. Blogs can deal with journalism, music, and more» (nexenservices), or yet «a personal multimedia diary, easily updatable & reachable with a computer or cellphone» (France Telecom).

A photoblog (or fotoblog) is very similar. Instead of just publishing text, one can also upload images, per diem, or more. Images can be photographs, paintings, media creations. However new problems arise. Firstly, images are bigger and weigh more than simple text. Users need fast Internet access. Secondly, images need to be uploaded, described, and published easily, quickly, and nicely: this aspect will be discussed in this paper.

The placement's objective was to provide Blog.com with a photoblog solution that would be standalone as well as integrated to Blog.com's current blogging platform. Based on a benchmarking of existing solutions, a list of ideas was drawn. A new architecture was designed within its center, the image and important notions revolving around: categories, albums, galleries and tags for instance. Another notion was that of context. Images have different meanings depending on how they are viewed. Furthermore, images call for a precise and yet easy-to-write description.

An entirely new upload/classification/description/publication process was therefore written up. As of today, a beta version of the site is up and running. The work developed during the placement also highlighted new areas to be investigated: images should be entertaining. New image-toys should be developed to prepare for the future generations of fotoblogs.
Acknowledgements

I would like to thank Portugalmail S.A. for offering the opportunity of working there. I would like to thank more specifically Mr. Nuno Lopes and Mr. Sergio Carvalho, my tutor, for their technical help when problems arose.

I would also like to thank Mr. Fernando Costa, and Mr. Manuel Costa for being there, for being welcoming, for making me feel at home, for taking the time to speak more slowly with me.

Many thanks to the Faculty of Engineering of the University of Porto (FEUP): I'd like to thank Mr. Augusto Sousa, Mr. Raúl Vidal, Mr. André Restivo for their help and support before and during the placement.

Many thanks to the National Institute of Applied Sciences (INSA de Lyon): more specifically many thanks to Mr. Stéphane Bres, and Mrs. Sylvie Calabretto.

Specials thanks to Cláudio, «Jorro», Carlos, for bearing with me, and for the endless games of chess.

Lastly, I'd like to thank the «Région Rhône-Alpes» for their funding.
Table of Contents

I. Introduction ........................................................................................................... 1
   About the company: « Portugalmail Inc. » ......................................................... 1
   The project: « Developing a photo platform for Blog.com » ......................... 2
      What is the project about? ............................................................................ 2
      What's a photoblog in the first place? ......................................................... 2
      Why I chose Blog.com ................................................................................. 2
   Designing, specifying and developing a photoblog ......................................... 3
      Identifying tasks – giving priorities ............................................................. 3
      Initial planning ............................................................................................. 3
      Final planning ............................................................................................. 4
      Discrepancies between initial and final plannings .................................... 4
   Blogging the placement ..................................................................................... 5
   In this paper ........................................................................................................ 5

II. Publishing images online – turning one's blog into a media-rich site .......... 6
   Assessment ........................................................................................................ 6
   A new scenario ................................................................................................ 7
   Sorting the images ........................................................................................... 7
   Describing the images ..................................................................................... 8
   Publishing the images ...................................................................................... 8
   Objective .......................................................................................................... 8
   Constraints ....................................................................................................... 8

III. An overview of existing solutions ..................................................................... 9
   Existing solutions online ................................................................................ 9
      A brief study of online « photo sites » ......................................................... 9
      Comparison chart ......................................................................................... 13
      Conclusions ................................................................................................ 14
   Useful Technologies & Standards applied to the photo platform ............... 14
      DB OO2 ........................................................................................................ 14
      Describing images with the Exif format ...................................................... 16
      Laszlo Systems: dynamic content and XML ............................................. 17
      Web services ............................................................................................... 18

IV. Designing / specifying the new solution: Amadeo reborn ......................... 18
   An overview of Blog.com's architecture ......................................................... 18
   Amadeo's current architecture ........................................................................ 19
      Assessment .................................................................................................. 19
      The API ....................................................................................................... 20
      The database .............................................................................................. 20
   New concepts for a better platform ................................................................ 21
      Sorting the image ........................................................................................ 21
      Describing the image ................................................................................. 24
      Interacting with the image ......................................................................... 26
      A step further: playing with the image ..................................................... 27
      The publication level .................................................................................. 28
   New proposed architecture ............................................................................. 29
      Application class diagram ....................................................................... 29
      ImageContainer ......................................................................................... 30
      ImageCategory ......................................................................................... 30
      PersonalImageContainer ......................................................................... 30
      Album ......................................................................................................... 31
      Gallery ........................................................................................................ 32
      ImageComment ......................................................................................... 32
      Tag .............................................................................................................. 32
      TagFamily ................................................................................................. 33
V. Development of the photo platform ........................................... 39
   Development process ............................................................... 39
   Organization ............................................................................. 39
   Technology used ......................................................................... 39
   Developing the database ............................................................. 39
   Creating a new schema ............................................................... 39
   Migrating the current system to the new system .......................... 40
   Developing the website .............................................................. 40
   The site's architecture ............................................................... 41
   Developing the public interface: Amadeo .................................. 42
   Developing the editor interface: Quill ....................................... 45

VI. Conclusion & Outlook ............................................................. 47

VII. Bibliography ........................................................................... 49

VIII. Appendix .............................................................................. 49

   Solutions comparison chart ....................................................... 49
   Os Use Cases do blog de fotos – Parte pública ........................... 53
I. Introduction

About the company: «Portugalmail S.A.»

Portugalmail is a small 10-people company created some six years ago and currently managed by two brothers: Mr. Manuel Costa and Mr. Fernando Costa. Originally, both saw the potential in web development and chose to start their e-business.

The company's core business is e-mail. Owner of the domain names portugalmail.com, portugalmail.pt, and portugalmail.net, it offers a broad range of email products from the usual free account for all with plenty of space, to the hiped accounts for companies and businessmen. It also runs a webmail service on its site.

The company's site, much like any other webmail site, boasts tools to browse the web (search engine) and news tickers.

Portugalmail also provides web access to several thousands of customers throughout Portugal. The type of access is a regular 56k modem-based access and ADSL.

Here's an excerpt from Portugalmail's About page on their website:

"Joining together a fast, reliable, and secure service, the love Portuguese show for their country and who feel pride in owning a "@portugalmail" email, the objective of creating a Portuguese web community quickly became reality.

In two years, Portugalmail became the Portuguese preferred mail provider, both in Portugal and in Portuguese communities abroad. Portugalmail evolved to become a Community portal where users don't just come to check their email but also come to chat about various topics, create new friendships, and much more."

The company, being rather small, has an enormous capacity to adapt and respond to new needs, new trends. Blog.com is one of those highly dynamic projects. Since it owns the domain name Blog.com (and amongst others, Blog.pt), Portugalmail chose to develop a blogging service and go «worldwide». The service started in June 2004 and rapidly grew in the following months.

Blog.com's basic service is free and allows anyone who registers to create as many blogs as he/she wishes and write posts (small dated texts). Those posts are ordered in the reverse chronological order. Posts are a rich media content and allow for images to be posted.

In order for its Blog.com branch to grow, Portugalmail offered three placement positions to students of the Faculty of Engineering of the University of Porto (FEUP). The aim was to increase Blog.com's quality of service, broaden its offers & services and take it a step further in the blogging market.

Sergio Carvalho, the placement tutor, and also Portugalmail's technical director, defined three areas in which to work: the photo platform, the reengineering of Blog.com's platform, and the content aggregation system (RSS, Atom...).
The project: "Developing a photo platform for Blog.com"

I applied for the placement entitled "Desenvolvimento de uma plataforma de fotos para o Blog.com", i.e. Development of a photo platform for Blog.com. After the placement interview, Portugalmail accepted me and on March 2nd, I started working.

What is the project about?

The Blog.com designers and developers had already developed a photo platform in the year prior to my placement. They had noticed the ability to upload pictures to a blog was an extra feature many bloggers-to-be were looking for and few blogging sites had. Therefore, they created a platform, dubbed "Amadeo", allowing for single-file upload, image description, album organization and publication.

However, they noticed the platform could include yet many more features. A tagging system perhaps? An upload tool? More dynamic content? I was to browse through all the ideas, brainstorm some more, set up a new design, and implement it.

What's a photoblog in the first place?

Here's a quick definition given by Wikipedia, the online encyclopedia:

"A photography blog, picture log or simply a photoblog, is a web application which contains periodic posts containing user-taken photographs on a common webpage. These posts are often but not necessarily in reverse chronological order, from the date when the photograph was taken."

A photoblog is slightly different from a photo platform for a blog. The latter is merely storage space for images to be used in one's blog, with a minimal classification system. Blog.com's Amadeo is a photo platform. On the other hand, a photo blog has a more complex system and is more photo-focused. It highlights photos rather than texts.

The idea in the project developed is perhaps to reach a compromise between a photo blog and a simple photo platform. As a matter of fact, the solution developed was first a standalone solution before being integrated into Blog.com.

Why I chose Blog.com

I've always loved developing sites, ever since I first browsed on the web exactly ten years ago. Back then of course, it was HTML, some Javascript snippets, image maps and ugly designs.

In high school, at university, and during one placement, I had the opportunity to develop more websites using different technologies (Java, PHP, Flash...) and I always craved for more. Furthermore, I always loved journalism, writing in general, and joined three newspapers. When I first heard about blogs, I thought "fashion victims" but then I realized it was a
mixture of two of the things I love most: IT and writing. Choosing Blog.com was therefore quite natural.

Lastly, as I had worked with images, and semantics during the semester prior to the placement, I had some ideas I wanted to test. Working at Blog.com was the perfect opportunity to do so.

**Designing, specifying and developing a photoblog**

Being still a student, planning a six-month-long project might actually be the most tedious task in the project. Furthermore, one must learn to work according to a specific schedule and perhaps set secondary tasks aside to proceed with the most important ones.

**Identifying tasks – giving priorities**

The photoblog will have two parts: a public interface visible to all and the Editor visible to the sole user who uploads and describes his/her images. Both parts are equally important. However, it seems wiser to develop the public interface first as it is easier and simpler. Furthermore, the current photo editor already has a set of tools that will remain valid throughout the development of the new platform.

The photo platform will have crucial functions and others not so crucial. For instance, uploading an image is mandatory for the site to work. An image rotation is not. One must keep this in mind while developing.

**Initial planning**

<table>
<thead>
<tr>
<th>Initial planning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>March:</strong> Benchmarking of the solution – brainstorming – initial design – choice of tools and technologies</td>
</tr>
<tr>
<td><strong>April:</strong> Full design of the solution</td>
</tr>
<tr>
<td><strong>May:</strong> Development of the public interface according to the design</td>
</tr>
<tr>
<td><strong>June:</strong> Minor tests – freezing of the public interface – preparation for the second phase</td>
</tr>
<tr>
<td><strong>July:</strong> Development of the editor’s interface</td>
</tr>
<tr>
<td><strong>August:</strong> Overall tests, report writing, second thoughts on the design developed</td>
</tr>
</tbody>
</table>
Final planning

<table>
<thead>
<tr>
<th>March</th>
<th>Benchmarking of the solution - brainstorming - initial design - choice of tools and technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>April</td>
<td>Full design of the solution</td>
</tr>
<tr>
<td>May-June</td>
<td>Development of the public interface according to the design</td>
</tr>
<tr>
<td>Early June</td>
<td>Freezing of the public interface</td>
</tr>
<tr>
<td>Mid June</td>
<td>Integration of the public interface into Blog.com's platform</td>
</tr>
<tr>
<td>July-August</td>
<td>Development of the editor's main functions. Work on the class documentation. Update of the application's class diagram.</td>
</tr>
</tbody>
</table>

Discrepancies between initial and final plannings

There are some notable differences between both plannings due to the fact I haven't got the habit to plan yet. The most overlooked aspect is the time it takes to develop. I believed when I first started that developing would be quick and easy. I know PHP relatively well and it is not a complicated language.

However, from time to time, one reaches glitches in the design that makes the development irksome, lengthier. It can be small aspects of the site but are required to proceed.

Furthermore, while developing, software and compatibility problems may arise slowing progress.

Moreover, the planning was perhaps too rigid. There were tasks, such as testing, that were done on the fly, as development progressed rather than at the end of the development process. Some tasks were therefore shuffled around.

Lastly, the standalone photo platform developed was integrated into Blog.com's platform. This phase pushed the rest of the planning a couple weeks off.

Blogging the placement

It seemed almost obvious that the perfect tool to create documents, publish them, and write news about the placement, would be a blog. Therefore, I created the following website:

http://fotoblog.blog.com/?password=boavista

The access password is boavista. The website contains all the notes, thoughts, and documents produced during the placement. In addition, I published the diagrams on the
website allowing for the tutors to analyze them. Being a foreign student, the blog helped as
two professors in France were able to monitor my work that way.

**In this paper...**

This paper contains four main parts:

1. **The problem explained**
   
   One’s already mentioned what photoblogs are, what a photo platform is. This small chapter takes the reader a step further and unravels the full problem, its limits and the expected results.

2. **The benchmarking chapter:**
   
   This part deals with the research and benchmarking tasks. It deals with finding out how the others do it. Are there ideas out there we haven’t thought about?

3. **The design and specification chapter:**

   Firstly, this part deals with all the research and designing, some of which may be based on benchmarking previously done. It deals with the ideas brought up by different team members and how these ideas can be pulled together.

   Secondly, this part specifies the entire site to be developed. It is therefore the most important part for the company. It is the part of the document that will help with the site maintenance and future development.

4. **The development chapter**

   This part summarizes the development process and explains each and every problem that may have occurred. It digs into the code and gives any future developer a better insight on the solution developed.
II. Publishing images online – turning one's blog into a media-rich site

Assessment

On top of Blog.com's publishing platform, which allows for news and text publication, Blog.com would like to develop a tool that enables image sharing and publication.

As of today, there already exists such a platform at Blog.com named Amadeo. It allows two ways of adding images to the server:

1. simple image by image upload form (jpg, gif, png)
2. simple zip/tar upload form.

Images are then stored in an album. Both albums and images can be described and labelled although the site shows very few users do so.

Here's a quick overview of the pros and cons of the existing solution:

<table>
<thead>
<tr>
<th>Blog.com's photo platform</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple and straightforward interface. Fast image publication process.</td>
<td>In order to create an album, one must upload an image. It is not very handy.</td>
</tr>
<tr>
<td></td>
<td>The current platform allows for image annotations, i.e. active zones on the image with a small one-line description.</td>
<td>The platform needs a better sorting/description system. Currently, users tend to upload an image per album without renaming the latter. It is therefore labelled «Untitled». This is due to a design flaw in the Upload page.</td>
</tr>
<tr>
<td></td>
<td>The Post to Blog feature makes image publishing simpler by creating a new post and inserting the desired image.</td>
<td>The photo platform doesn't have a multifile upload tool. It might be worthwhile to look into it.</td>
</tr>
</tbody>
</table>

A new scenario

Here's a sample scenario to help the reader understand what is the situation and problem.

Mr. and Mrs. Bucket went on a holiday with their three children. The destination of their choice this year was the Disneyland resort in Orlando, Florida. They took hundreds of digital pictures, not mentioning the pictures the kids took with their disposable Mickey cameras bought at a souvenir stand.

Once home, Mr. Bucket wishes to put his pictures to share with friends and families, and to keep as memories as well. The children will enjoy looking back at the pictures when they grow up. However, Mr. Bucket is only a few pictures
shy of the thousand mark and needs a system easy to use to sort, describe and publish them.

Nowadays, he could choose a photo processing website where to store his images. But it's not very convenient, as the browsing experience is very limited, and the images displayed reduced in quality.

To make things trickier, Mr Bucket is also a blog addict and writes on a regular basis in his blog. He'd appreciate a system that'd combine a minimal photo storage, an advanced description system, and a posting system. Mr. Bucket wants in fact a photoblog.

But Mr. Bucket is also a bit lazy and doesn't feel like writing hundreds of different descriptions for each and every picture. He needs to be more efficient. Using simple words instead of lengthy sentences might sound attractive.

The scenario previously described shows the problem everyone had before the "multimedia era" and that worsened with digital cameras. We have a huge set of image data, and we'd like to sort them and publish them so we can share them with friends and family.

**Sorting the images**

![Simple album sorting diagram]

In real-life, the answer would be running down to the store and buying an album to stick your images in as the above diagram illustrates. You could also write a legend below each image... and end with a sore hand by the time you finished. But back then, users took some 36, 72, or at most 100 pictures per holiday. Some three rolls of film. Nowadays, they shoot away 128Mbytes of data a day, some 100 pictures on a standard camera. That's a seven to ten fold the previous amount.

The album notion, although very simple, now seems very limited and does not provide an answer to the image sorting issue.
Describing the images

Mr. Bucket wants to write small texts for some/all of the images he took. This text cannot be in his blog only. It would dissociate it from the image and the latter would therefore lose its meaning.

Once he's described his images, Mr. Bucket might want to browse through those that have his youngest son in them. This would require him to be able to search in the image description. But full-text search is rather lengthy and inefficient – specially in a web context. So how can the problem be solved?

Publishing the images

Mr. Bucket being a prolific blog writer, would like to use his images, as stated in the scenario, in his blogs. How does one go about publishing images in a blog? Should it be tedious HTML encoding? And what about those who know nothing about HTML tags? We cannot request users to learn the ropes of the web's language.

Mr. Bucket would really appreciate a slick, nice-looking interface, and extra features such as visitor feedback, based on comments or perhaps votes. Features that are of course unfeasible in a « real-world » album case scenario but that replaces the chit-chat one would have with his friends & family while showing the pictures.

Objective

Blog.com wants to develop a more complex photo platform than the one it currently has. It would like to offer a high-quality service to its users where the main keywords are: ease of use, speed of service and organization. In terms of photoblogging, this can be translated into:

1. simple, quick, and easy sorting of images
2. simple, quick, and easy description of images
3. simple, quick, and easy publication of images for enhanced experience / photo browsing
4. and taking it a step further, making the image multimedia, interactive, just like a game!

One has to think of a new sorting mechanism to answer to the increasing number of images. The mechanism has to be efficient, clear, and simple once again. Furthermore, the platform's behavior must be similar to Blog.com's blogging/photo platform in order for the users to feel at home straight off.

A balance must be found between a photo blog application and a simpler photo platform for a blogging application.

Constraints

The most obvious constraints are directly linked to the current Internet infrastructure. As cameras become more advanced, the resulting image files are bigger in terms of Mbytes. The increasing size poses the following problems:
1. Bandwidth problem
   a. The upload time increases for each image,
   b. The server needs a broader bandwidth. As more and more users browse through the pictures, they use up the server's bandwidth, clogging its access up.
   c. The bandwidth problem also rises another issue, that of hotlinking. Users can have their bandwidth stolen if a third party uses a user's image directly from that user's web server.

2. Disk space problem
   This is a minor problem. It's been proven capacity increases quicker than information. Furthermore, storage costs have dropped. It is nonetheless a problem when multiplied by the number of users on a server and poses the question: « how much space should we allow each user? » If the original 10Mbytes allotted to images seemed enough at Blog.com, it might become slightly measly for a full-blown photo platform.

   Most importantly perhaps, another constraint is the user him/herself. A photo blog needs care. In other words, it needs for the user to describe each image thoroughly, or at least with a minimum amount of information. This constraint shows how important it is to create an easy-to-use description system that incites the user to write a couple words about each picture. One can also think of massive description tools.

   If a system were too complex, a user would give up and not describe his/her images, making the whole platform pointless and therefore valueless. A compromise must be found between describing in extenso and no description at all. This may call for a new description mechanism.

   It has been indeed shown that users tend to overlook the description process, specially on photo processing websites. One could argue that the purpose of such sites is not to describe and that therefore the user's behavior is normal. However, when one looks at Blog.com's current photo platform, it is obvious users don't take the time to describe. Why? Is it too tedious, too lengthy a process, not worth it? Or simply because users do not know how to?

   The description system developed in the new platform will have to adapt itself to the users' needs, become more conspicuous, simpler.

III. An overview of existing solutions

Existing solutions online

This chapter is based on the document « Blog.com : Estudo dos fotoblogs existentes e das melhoras possíveis » written at the beginning of the placement.

A brief study of online « photo sites »

We chose to study various photo sites, whether they are purely photo blogs or not. We identified different sorts of photo sites: from the processing photo site (such as Ofoto) to the Amateur site (Olhares).
Wistiti

This French site has become one of the top processing sites in France with outstanding features such as unlimited storage space, an advanced upload tool, many online editing features (red eye, cropping, resizing).

Initially developed to offer the best processing deals to its customers, Wistiti also developed an online community and publication tools.

- http://www.wistiti.fr/
- **Purpose:** photo processing, photo sharing online
- **Technology used:** the website itself is written in ASP. The Edition module is written with javascript and a back-end module. The image upload tools are Java applets. At some point, Wistiti also had an ActiveX component available to upload images. It is no longer on the site. A simple web form can also be used to upload 1 images or less.

- **Pros:** Wistiti has combined photo processing and publishing cunningly. Users can send a URL for friends to view uploaded images. Wistiti also allows users to « sell » their images and earn bonus points. The privacy (who can see what?) system, password-based, is very efficient and simple to use. The upload applet is very efficient and works well. It has been known to have some glitches when used behind a restrictive firewall. Most of all, Wistiti's success comes from the fact it is entirely free and has unlimited storage space for its users.

- **Cons:** the edition tool, javascript-based, does not work well with Firefox. From time to time, an alteration is not effectively done. Furthermore, the images displayed online, in the publication part, are not the actual image file. Both the image quality and the image dimensions have been reduced.

Flickr

This US-based international website is probably the most innovative and simplest as well. With the rise of blogs and the need for image storage, Flickr has grown quickly over the past couple of years.

- http://www.flickr.com
- **Purpose:** image publication
- **Technology used:** the site itself is based on a content management tool ensuring a perfect organization. The image publication and display pages are all written with Adobe Macromedia Flash enabling rich content and an enhanced user experience. As for the upload, Flickr has a myriad of solutions:

  (a)Web form
  (b)Flickr Uploaddr Windows & Mac OS X
  (c)Photo for Mac OS X
  (d) « Send to Flickr » Windows XP Explorer
  (e) « Send to Flickr » bookmarklet
  (f) Cameraphone upload

- **Pros:** first and foremost, the tag notion is perhaps the most innovative « tool ». It allows one-word descriptions that are therefore easy to write up, easy to search through and easy to
correlate to other users’ descriptions. Tags allow to weave connections in between different users’ images. Furthermore, Flickr allows for rich image interaction, flash-based, and has an annotation tool. Lastly, the site allows images to be posted to various blogs, as it has an extensive API blog sites can use to communicate with Flickr.

- **Cons:** Although Flash brings a high-quality output, it also requires users to have the Flash player and have a fast access. Dial-up access-based users would experience problems on the site. Furthermore, the only « sorting » and « description » tool is the tag. Although it is very useful, it lacks a stricter organization, album-based for example.

**Ofoto**

Ofoto is Wistiti’s North American counterpart. With a slicker interface, and Kodak support behind (it is a Kodak owned company), Ofoto provides users with many options, products, and of course a high quality processing.

- **http://www.ofoto.com**
- **Purpose:** photo processing
- **Technology used:** the site is entirely written in JSP. As for image upload, much like with Flickr, the user has several choices:
  - **(a) OfotoNow**
    
    This software must be installed on the user’s computer and allows the user to choose a set of images in his computer, alter them (rotation, cropping, red eye removal), create a slideshow or even a screensaver, and eventually upload them. Unfortunately, this software exists only for Windows.
  - **(b) EasyUpload**
    
    Online tool that only works with IE and Windows. It is based on an ActiveX component
  - **(c) Web form**
    
    The usual upload form.
  - **(d) EasyShare**
    
    A Kodak-developed software for image editing and upload. Not available on the site.

- **Pros:** Being part of the Kodak group gives this site a headstart in terms of photo processing. Users will trust the brand and think of it in terms of quality of service. The site is also the only one to have regional versions (US, UK, Germany, France, Europe). The online editing tools are powerful and work fine, much better than those found at Wistiti. Lastly, Ofoto has a guestbook for each user. Each time a visitor browses through a user’s photos, he/she can leave notes, comments. The user can also have the complete list of visitors.

- **Cons:** Ofoto only allows JPEG files. Furthermore, the album management is somewhat weak. Furthermore, the privacy (who can see what?) system isn’t very good. One must invite friends by email only and images cannot be shared with other Ofoto users nor can the images be published on the site as it is at Wistiti.fr. The browsing interface, whether the visitor is a simple visitor or the user him/herself is the same. It results in confusion for the visitor and an unintuitive browsing system.
Photoways

Photoways is yet another French website offering photo processing and much more. The website, created half a decade ago, bet on its e-shop and derived products to thrive. It sells photo books of various sizes, mugs, T-shirts... Virtually anything a customer can think of.

- **http://www.photoways.com**
- **Purpose:** Photo processing, photo-based products
- **Technology used:** The website is developed in PHP. As for image upload, there are three options:
  (a) « Transfert simple » (simple transfer): for IE and Windows only, this ActiveX-based upload tool was developed by an American firm and allows uploads of up to 250 photos in one go. Each photo can weigh up to 5Mbytes. Upload time is limited to 2 hours.
  (b) « Téléchargement classique » (Standard upload): the usual web form
  (c) « Transfert avec logiciel » (software-aided upload): Photoways has a software users can download to upload pictures to the server. However the software is for Windows and Mac only and can only upload Jpegs.
- **Pros:** Photoways's main features are that of the software one can dowload. It can do basic and advanced image editing very easily:
  1. Image zoom,
  2. red eye correction,
  3. image rotation & crop,
  4. image tint
  5. create image negative,
  6. alter sharpness, brightness and contrast,
  7. add messages to the image,
  8. insert messages in the image.

Furthermore, editing on the user's computer speeds up the whole process. Fewer accesses to the server are required.

- **Cons:** Photoways seems to ignore all those users using Linux. Furthermore, the online albums do not allow user-visitor interaction (no comments space for instance).

**ImageShack**

Imageshack is an experimental free website that aims at creating a large online photo gallery. Imageshack offers a series of innovative & unique tools unseen elsewhere.

- **http://www.imageshack.us**
- **Purpose:** online image storing and publishing with external publishing possibilities (i.e. on sites other than the hosting site)
- **Technology used:** dedicated Linux server, PHP scripting language.
- **Pros:** Imageshack offers users the possibility to upload images to a server, organize them in galleries and share them with other users. But the main feature Imageshack has is the external publication feature and the possibility to hotlink to images. Imageshack generates code snippets for users to add to their own websites, blogs, or even forums.
- **Cons:** This site isn't very easy to use and is mainly for advanced users, people already familiar with the web environment. Furthermore, there is no easy-to-use interface. A lot of functions work based on email activation and request codes.
Developing a fotoblog platform

Olhares

Olhares (means Glances in Portuguese) is a Portugal-based site, in Portuguese, dedicated to photography. Quality rather than quantity is privileged in this site. Olhares has very interesting sorting features as well as user interaction.

- **http://www.olhares.com**
- **Purpose:** publication of amateur and professional photography. The aim is to share photos a user deems outstanding.
- **Technology used:** PHP. No online editing tool.
- **Pros:** There are two distinct parts to the site, with nonetheless a very similar interface: the public part where visitors and users can browse through a series of photos, and the Editor where users can add their photos. The site is very dynamic, and a community is thriving. Leaving comments is easy provided one has an Olhares account. Furthermore, Olhares has self-managed categories that organizes the site's contents and provides visitors with means to browse the photos:
  - Most recent
  - Most viewed
  - Top rated
  - Most commented
  - Favorites
  - Our (Olhares) Choice
  - My Favorites (for a given user)
  - This Month's feature photo
- **Cons:** Olhares has no interface for blog use, nor the slightest API. Furthermore all the images must be Jpeg, and have a maximum dimension of 500x500px. Lastly, it isn't possible to edit images online.

**Comparison chart**

See appendix for full comparison chart.

The comparison chart was built based on the study of the different sites mentioned previously. The chart tries to highlight several important criteria that are essential in a photo blog application. The criteria are the following:

(a) **Upload features:** what are the different formats accepted, what are the different ways to upload, the speed, the limitations, the ease.

(b) **Service registration:** does one need to create an account? Is the service free? Are there different levels of service? What are the disk space and bandwidth quotas?

(c) **Publication levels:** can we publish items and make them visible to a certain group of people? Can content be private/public? How is privacy managed?

(d) **Organization, description, and browsing:** Does the site have albums? Galleries? Categories? Auto-generated categories? Favorites? Tags?

(e) **Image display:** Does the site generate thumbnails? Does it display the original image? Can we view a slideshow? A random image?

(f) **User-visitor interaction:** can we leave comments, vote for our favorite image? Is there any form of syndication (RSS, Atom)? Are there private messaging abilities? Forums?
(g) *Site help:* is there any FAQ, any support page?

(h) *Shopping:* does the site sell any derived products?

**Conclusions**

By studying competitors' online platforms, important areas have been highlighted. It is important the new photo platform be as successful as can be. This success goes through making sure it has an upload tool users feel at ease with. Furthermore, the description and organization process must be simple yet complete in order to have advanced publication and an enjoyable browsing experience.

One mustn't also forget the photo platform's main purpose: photo support for Blog.com. Some publishing tools linked to the blogs might be useful.

Overall, Flickr might well be the sort of site the new platform should look like, but with more aspects derived from Olhares, for instance.

Before getting on to the Design Chapter, let's have a look at technologies yet undiscovered and that may come in handy.

**Useful Technologies & Standards applied to the photo platform**

**DB ОО2**

*What is it?*

DB ОО2 (Database, object oriented version 2) is a PHP API written by Sergio Carvalho that simplifies access to databases.

The idea is to use the structure of a table in a given database to construct an object with setter and getter functions and attributes bearing the names of the given table's attributes. For instance, if a database has a table Author with the following attributes:

- name
- dob
- town
- pseudonym

Then, an Author object will be created with the functions getName(), setName(), getDob(), setDob(), and so on...

This object creation is very useful. First and foremost it saves developers time as they will not need to write the code. Secondly it simplifies down database access, works with Postgre and MySQL. Lastly, it makes sure classes and the database structure remain coherent as any alteration brought to the database is automatically reflected in the code.

However, data access might be slightly slower, and some specific uses involving views are not straightforward.
Blog.com's website is already based on DB_OO, but in the photo platform case, DB_OO2 will be used. The main difference is that DB_OO2 supports views.

How to use it?
There are two ways to call DB_OO2 to create an object based on a table.

1. Inline call: anywhere, on the fly, one can use DB_OO2 to create an object based on the desired table. For instance:

   ```php
   $imagesObject = DB_OO2_Table::factory($dbOo->db, $tagInImageTableName);
   $imagesObject->fetchAllByQuery($query, $queryParams);
   ```

   The call needs the open database connection ($dbOo->db) and the table on which to operate ($tagInImageTableName). One then only needs to send a query, with the right parameters to the object.

   Once the query has been successful, and provided there are no database or SQL errors, the object now points to the result of the query, a series of rows read from the targeted table.

   To access those rows, one needs to run the following sample code

   ```php
   while($imagesObject->next())
   {
       echo $imagesObject->getLabel();
   }
   ```

   It can be assumed label is an attribute of the targeted table.

2. Call by extension: the object model is used to build a class, for instance class Image. This class contains the usual operations one would expect of such a class: resize(), getList() but does not have any basic functions such as getLabel(), getAuthor(). And yet we do need to know who took a specific image. All these functions that merely return data read in the database will be left to DB_OO2 to write. All the developer needs to do is extend his own class the following way:

   ```php
   require_once("api/DB_OO_2_config.php");
   class Image extends DB_OO2_pgsql_Table
   {
       ...
   }
   ```

   The class constructor should be as follows:

   ```php
   function Image( $id = null ){
       global $dbOo;
       global $extendedImageViewName;
       parent::__construct($dbOo->db, $extendedImageViewName);
       if ($id !== null)
       {
           $this->fetchAllByWhere("id = ", array($id ) );
           $this->next();
       }
   } // end of member function Image (constructor)
   ```

   The configuration file needed in both code excerpts defines $dbOo as follows:

   ```php
   define('CONFIG_DBURL', 'pgsql://postgres@localhost:5432/com_blog');
   $dbOo = DB_OO::getSingleton(CONFIG_DBURL);
   ```
$dbOo has all the information one needs to connect and maintain a connection to the given database (e.g. com_blog) on the given server (e.g. localhost).

Where is it used in the photo platform's code?
Nearly all the classes in the photo platform use DB ОО2. As one will read in the Specifications chapter, almost all the classes, save those extending another class in the photo platform API, extend DB ОО2. Those classes not based on a table in the site's database do not use DB ОО2 either.

DB ОО2 proved to be a very useful tool.

Describing images with the Exif format

One of the first answers to the image description problem came from the Kodak labs. Researchers at Kodak labs wanted to design a description system for the photos, consumers took with their Kodak cameras. A customer might well want to store such information as shooting time, camera time and date. Exif was also designed to improve printing performances.

Exif stands for Exchangeable Image file Format and is a series of fields included in the imag file that describes a set of image parameters. Exif has shown to be very useful as digital cameras use them to store taking conditions and parameters such as lighting.

Initially, the Exif format was developed to improve printing quality. Several important information are not included in the image's « pixels ». Brightness, contrast, sharpness are information printers could use to print images to a higher quality standard.

However, the Exif information has found several other fields of application, specially in image editing software or in image album software. In Blog.com's future photo platform, the use of Exif will be useful as each field an Exif file contains can be used to « tag » images or to search through an image set.

Here are some of the fields one can find in an Exif file:
Developing a fotoblog platform

SubjectArea – CustomRendered – ExposureMode – WhiteBalance – DigitalZoomRatio –
FocalLengthIn35mmFilm – SceneCaptureType – GainControl – Contrast – Saturation – Sharpness
– DeviceSettingDescription – SubjectDistanceRange – ImageUniqueId – GPSProcessingMethod
– GPSAreaInformation – GPSDateStamp – GPSDifferential – LightSource – Flash

Needless to say the use of all the information fields would be useless. One will have to sort between those interesting fields (such as CaptureDate) and those pointless to an photo platform application (ImageUniqueId for instance).

**Laszlo Systems: dynamic content and XML**

Laszlo is all about making the web dynamic, while using the Adobe Macromedia Flash format, without the slightest knowledge or use of the Macromedia Flash Editor. Moreover, the Laszlo developers claim their system is the right mixture of HTML-based technology and rich presentation techniques while offering object-oriented design capabilities:

**HTML is a great thing; DHTML is a great thing. Macromedia's Flash™ is a great thing. But none of them allow developers to create astounding user experiences, because all of them are based on fundamentally limited models. The variants of HTML are based on the metaphor of a "page," which forces users to process information in arbitrary discrete chunks separated by jarring transitions and "page refreshes." Flash offers the possibility of a much more continuous user experience, but its "movie" metaphor forces developers to leave behind many essential tools and techniques of modern software development.**

In contrast, Laszlo's solution was built from the ground up for application development — not "page" development, not "movie" development — and is centered around standard development approaches. LZX applications are written in XML files with embedded JavaScript, which provides an ideal foundation for teams of serious developers. Laszlo supports standards-based object-oriented development and data binding, and enables rich interactivity without requiring complex, timeline-based visual authoring.

Excerpt from the preface to the Laszlo Applications guide

For Blog.com, the main features Laszlo has to offer is that it allows Flash development with free tools. No Flash licences are necessary. Furthermore, all the development can be dynamic. With Laszlo, since the applications are written with XML, all a developer needs to do is hack into the XML source and create applications nearly on the fly.

At Blog.com, prior to the development of the photo platform and with a preliminary design in mind, the possibilities of using Laszlo were looked in, mainly to deal with tags. An interesting application could be a tag tree – or rather tag web showing siblings interwoven together. The user interaction would be tremendous.

However there are some drawbacks to Laszlo, mainly speed performances. Furthermore, the development process isn't very straightforward.
Web services

Web services can extend the possibilities of not only photoblogging, but also plain blogging. In this study, we will only look at what web services can offer us, or already offer.

Web services are a generic term for any sort of API or standardized means of communication between distant servers, or between a series of clients and a server. For instance, one of the Lazslo demos, the weather application, uses a web service to retrieve weather information online.

In order to use a web service, the developer/user needs to know its endpoint, i.e. its URL. He also needs to know how to call the web service, what parameters are needed. In the case of a weather service, it might be the zip code.

The web service is no more than a public entry point to a system stored on a separate server. It allows queries to come from outside the server. When a web service is requested on a given server, the latter will reply in a standardized way. SOAP is one XML-based web service standard among others.

So how can web services be useful to a blogging site?

Flickr, for instance, allows users to publish photos stored on the Flickr server to various blogs. They use their special publication API. This API deals with the communication between the blog hosting server and Flickr. The API is no more no less than a web service that lets a « foreign » server query data stored on the Flickr server via specific methods defined in the API. Photos stored on the Flickr server become readily available for publication on virtually any blog.

Web services could also be used between a client and the Photo platform server. One can imagine for example a user's installed a software developed by Blog.com and that would allow more editing functionalities, more description abilities, more ease of use (client-based GUIs tend to be more user-friendly than web-based GUIs). All the communication with the server and the user's data would be done via a web service.

Lastly, since web services can communicate on top of the HTTP protocol, via the most standard ports, they nearly always work, whether a firewall is active or not, provided basic web traffic is allowed.

IV. Designing / specifying the new solution: Amadeo reborn

An overview of Blog.com's architecture

Blog.com is broken down into three main parts:
- the blog publishing / browsing platform: Oraculo
- the photo platform: Amadeo
- the editor, both for blogs and photos: Quill.
The following diagram illustrates this architecture. This paper will mainly focus on Amadeo, the photo platform, but will also focus on Quill as the work discussed affects both these platforms.

Illustration 2Blog.com's three-part architecture

**Amadeo's current architecture**

**Assessment**

Users at Blog.com and visitors can currently view users' images by browsing to Blog.com's homepage (http://www.blog.com) or by going to a given user's blog.

If a user/visitor selects an image he will be taken directly to the image's page where he/she can view the image's title and description as well as the medium-sized image. If the visitor clicks on the displayed image, he/she will be taken to the original image file.

If a user/visitor selects an album in a given user's blog's sidebar (where a list a of albums shows up), he/she will be taken to the album's homepage. The latter displays a series of
thumbsnails, each of which points to the Image page described in the previous paragraph. The default number of thumbnails is 9.

The current platform defines user-blog-photo relations as follows:

A photo belongs to an album, and only one album. A photo also has an author. The album belongs to only one blog. The album also has an author. The blog belongs to one and only one owner and can have several authors, which is why both the album and the photo have author information.

Blog.com allows multi-authoring, which means a given user, currently editing a blog he does not own, may want to create albums and upload photos.

The API

The current API that makes up Amadeo is very limited. It contains a class Album and a class Photo. The relation between both is that an album can contain several photos.

The classes are broken down into two files: Photo.php and Album.php The Photo class allows basic description functions as well as the upload/deletion functions. It also has support for annotations. An annotation is an active zone in a photo users can associate comments to. When a user/visitor hovers over the photo, the active zones «pop up» and on entering each zone, the matching comment shows up.

The Album class allows basic description functions.

The database

The following is based on «Migrating from com_blog v.1 to com_blog v.2 :: Adding image data support to the existing database:: », a document written during the integration phase of the public interface.

The database has several tables needed for Amadeo:

- Album
- Annotation
- Author_album
- Blog_photo
Developing a fotoblog platform

• Photo
• Photo_Album

However, several tables are unused: Author_album, Blog_photo, and Photo_Album are unused. They will have to go with the new architecture.

New concepts for a better platform

In short, the current platform allows users to have photos stored in albums. Users can choose whether to display a link to his/her albums in his blog's sidebar.

The navigation process is simple. It is based on the blog-album-photo relation.

However the platform lacks more browsing options, more publication options, more visibility. The notion of photo needs to be rethought altogether.

From now on, the word image will be used in lieu of the word photo. Overall, in the new design and specifications, the image will be the center item. The decision to change arose from the fact photos have a “camera” connotation and sound restrictive. Images on the other hand designate any sort of graphics.

Amadeo will stand for “photo platform”.

Sorting the image

In order to make the image more easily reachable, one must add criteria, filters, categories. In other words, containers. One can imagine a folder with thousands of image files. A user would then create subfolders to store the images in and create smaller groups.

The category

This is the simplest mechanism to break down a vast set of images into smaller groups. Based on Olhares’s use of categories, Amadeo, the new platform, will have a set of categories so that users can “categorize” their images making it easier for visitors to find related images.

A category is much like a theme. The whole set of image categories will represent a thesaurus, a thematical index where visitors can browse images by generic, top-level, image-based keywords.

The category is designed, created, and managed by the webmaster. It is therefore a non-personal container. It cannot be customized by the users to match their specific needs. This may seem like a constraint, but it simplifies down sorting. Furthermore, visitors will tend to become familiar with the category list and will jump to their favorite categories to look for any new image.

The category list is created when Amadeo first goes public but can evolve to match the latest trends.

A category should have a title, or label, and a brief description.
To keep things simple, images should belong to only one category at a time. This is a controversial choice as it limits the possibilities categories offer. Imagine the site has a category « Urban skyline » and a category « Nudes » (sample categories that exist at Olhares.com), where would you put the image of a nude woman posing in front of the famous NYC skyline? On the other hand, if users could add as many categories as they wanted, the system would lose some of its usefulness. Images wouldn't be as sorted, and it is better to keep the image-category relation to one category only per category.

In some ways, this makes categories very similar to the Dewey System used in public libraries.

**The album**

The most common sort of container for an image is an album. Everyone thinks of albums when it comes to images. All the sites analyzed in the preliminary study had albums, save Flickr which doesn't have any container structure.

The new album design extends the one currently existing in the Amadeo platform. The album will belong to an author. It is indeed a specific person who owns the camera used. If the album belongs to a given user, then the container becomes personal (one would remember the category is non-personal) which implies users can act on an album, create it, alter it, rename it...

Most importantly, the user expects to upload his/her images to an album. Therefore the new Amadeo architecture will respect that fact. The relationship between uploading and albums underlines the fact albums are a specific sort of container: they are physical containers. One should think of an album as a shoe box, or a folder where the images would physically be stored.

Of course, if albums are considered as a physical storage space, then images can only be in one given album at a time. In « real-life », one cannot have an image in a given album, and that same image in another album, unless copies are made. In the Amadeo world, it's identical. If a user wanted the same image in two different albums, he/she would have to make a copy which would imply there would be two files.

Furthermore, albums are often associated to a timeframe. If a person were to look back at his/her pictures from – let's say – the last ten years, he would dig them up based on a time scale. There would be the album labelled « Christmas 91 – Kansas City », and then maybe « Easter 92 - Arizona »). This shows albums need dates. They need a creation date (when was the album first online?) and perhaps a last updated date.

Albums complete the sorting process categories initiated. Now one can look for images belonging to the « Nude » category that belong to a specific album.

Albums should also be an easy way to manipulate, edit, and/or alter images. This aspect will be studied later on: it is known as mass editing.

Lastly, like the category, the album should have a title and a description.

The current result is that of a two-layer sorting system, still however rather rigid. The gallery is going to bring in some flexibility and « cross-sorting ».
The gallery

One has seen the album is a personal container, which means it can be customized by the users who create and manage them. It is exactly the same for galleries. So what is a gallery's purpose? Here's a scenario to get a better understanding of the problem:

One could think of Mr. B. who recently attended his friend's wedding. The wedding took place in the countryside and many animal farms came out on the pictures Mr. B. took, and specifically, the fiancé's cat showed up in several pictures even posing with the happy couple.

A couple months later, Mr B. went to the Bermudas for a well-deserved one-week holiday. During his stay, he snapped many pictures of the sea, the coconuts and the local fauna, including many cats.

Back home, Mr. B. would like to create an album dedicated to cats. He's uploaded the wedding album and the Bermudas album to Blog.com – Amadeo and would now like to show off with the beautiful pictures with cats on them. He couldn't possibly create an album. What would be its place in a time frame? Furthermore, he would have to make copies of those pictures. It would be a simple waste of webspace. Lastly, he doesn't want to dissociate the images containing cats from their original album. It would be incoherent. Creating a new album is simply out of the question.

This is where the gallery notion comes in. Mr. B. is going to create a gallery. Galleries are simply the web equivalent of real-world « exhibition halls ». Mr. B is going to reference his cat images in a gallery labelled « Cats ». Galleries do not therefore contain images per se but references to those images stored in their respective albums. Galleries could then reference to any image in any album, whether the gallery's user were the owner of the referenced image (provided the image is public and the given image's author authorized referencing). A vivid metaphor would be that of an art gallery. One could think of a painter's repertoire as a set of albums. The art gallery's curator would choose several works from several painters' repertoires and add them all together on display in his gallery. The exhibit would be based on a specific theme, which corresponds to the gallery's title in Amadeo.

Galleries are virtual personal containers; virtual because they contain no image and personal because they are user-created and managed just like an album is. The structure is exactly the same as the album's structure. The only difference is its function. One could argue Amadeo could have had two types of albums. However leaving the same name for two different concepts would have been confusing.

Of course, images can be referenced in several galleries and not just one as in albums.

Galleries do have one drawback. It might be confusing at first for Amadeo users, and developers will have to make sure the public interface (Amadeo) and the editor's interface (Quill) are simple and easy to understand & use.

Galleries introduce a new concept, that of contexts as one will see in a following section.

With categories, albums, and galleries, Amadeo now has a comprehensive sorting system, with three different layers.
The diagram above summarizes the new Amadeo sorting system. The sample image (a photography of «Palacio do Freixo», a manor in Porto, Portugal designed by Nicola Nasoni, belongs to only one album, but is referenced in three different galleries, and is also categorized in category C).

**Describing the image**

Sorting is not enough. What are images without words? What would Mr. B.'s wedding album without image titles and descriptions, bearing for example the happy couple's names?

**A simple image description**

This is the usual description field often found in image album software or in the online image albums. The image comes with a label and a description users can fill in.

However, it has been shown that users tend to overlook the description process. While specifying and developing, one will have to think of ways to incite users to describe.

Image descriptions might be useful and improve the interface and the publication aspect of albums. However, because they are small texts, they are not easily searchable. Doing a full-text search on descriptions would prove slow, and correlating images based on their description is nearly unthinkable. Flickr's system, the tag, is going to resolve this.
A context for an image

With the development of galleries, one has to think of a new concept: the context. When images are viewed, they are viewed either in a category, surrounded by other image members of this category, or in an album, or in a gallery. In each case, the "environment" is different.

More specifically, in the case of albums versus galleries, the image description might be suitable for the album, but becomes totally unsuitable for the gallery context. Therefore there needs to be a gallery-specific description for each image.

For instance, if the description for the image of the happy married couple is "What a beautiful couple" in the wedding album, it might be "this is a Siamese cat" in the "Cats" album.

The context notion is very important and while browsing, the specific context will have to be sent from one page to another.

An alternative to lengthy descriptions: the "tag"

Descriptions are very useful and suitable for user-end publication but are a nightmare when it comes to searching for specific images.

Flickr, the online Picture gallery, found a solution, now widely used on other sites such as del.icio.us, a bookmarking site. This solution is the tag. A small word, or expression that will be reduced to a word used to "tag" an image.

- Origin
A long time ago, when the web started up and HTML was designed, designers had already thought about tags. The latter were even included in the HTML language, in the head part, more specifically in the meta section. Any webmaster could add a small description and keywords as well. Those keywords were meant to be used by search engines back then.

But then, over time and because many webmasters didn't know about the meta section or just didn't bother to use it, those keywords faded in the midst of the web, that is until recently when websites reinvented the principle making keywords visible this time. Moreover users could now choose any given keyword and "play" with it.

- Principle
When users upload their images, they can, in a following step, associate their images to tags. Those tags can be any word, whether it exists or not. It is, technically speaking, a string stripped of white space and any forbidden "nonsense" characters (such as $ or %).

By adding tags to their images, users are actually contributing to a tag pool. And since all the users are humans, tend to take images of similar things, and usually think the same way, many users use the same tag.

This is exactly where tags come in handy. A given user can tag all his/her images "rose", and dozens of users can do so. When a visitor clicks on the tag "Rose", he/she will be able to see all the images containing the given tag.

Furthermore, when a user clicks on a tag and then chooses an image, he can see that image's other tags, and can start browsing along a new tag stream.
In comparison to Flickr, Amadeo will have an extra important feature: the tag family. Currently, Flickr only allows single-tag browsing which reduces the tag filtering capacities. For instance, what if a user wants an image of Paris, Texas, a movie by Wim Wenders, does the user have to browse all the images tagged « Paris »? Or all the images « Texas »? Either tag is very popular.

To solve this, Amadeo lets a visitor add tags together to create a family. This way, when a visitor clicks on « Paris » and sees all the images of the French capital, he can get rid of them by adding the tag « Texas » provided the tag is in the sibling list, i.e. provided there is at least one image having both tags.

In some ways, tags are customized, user-managed categories.

**Why is successful?**

Tags are a success because they are quick and easy to use. All a user has to do is type a word up and click on « tag it » to associate the typed word to the image being currently edited. Furthermore, one could think of a suggestion system similar to that used at [http://del.icio.us](http://del.icio.us): when a user tags an image with a given tag, then other tags related to the first are suggested for additional tagging.

Moreover, tags are fun and simple from a user's point of view. Users tend to tag a lot, creating a vast database of words. Tags can be read in so many ways. User behavior and interests can be deducted from tags. Blog.com can create a « Most popular » list, a « Most recent » list, or better yet, a popular list on a time frame: popular tags per week. One would probably notice the tag « Pope » was more frequent in April-May for instance.

**Limitations**

Since tags are words, virtually all the language limitations are also found in tags: synonyms, different meanings for a same given word. The limitations are many. How can we restrict a tag search « rose » to the flower rather than the flower, the software, and the female name? Amadeo, with its tag family notion limits this side effect somewhat.

**Interacting with the image**

**Image comments**

When users browse images online, they may want to take part in the site's life. In order to make the browsing experience, one can rely on image comments. Olhares, for instance, offers its registered users the possibility to leave comments behind.

Comments let users know what visitors think of their images. They allow discussions to take place. Visitors and users can exchange their points of view.

A maximum number of visitors should be able to leave comments. Therefore the right to leave a comment shouldn't be reserved to registered users. On the other hand, to prevent massive spamming, there should be anti-spam mechanism set up (see Blog.com's post comments)

Furthermore, as comments are a visitor's appreciation of a given user's « work », albums and galleries, in addition to images, should be commentable.
Lastly, the user should be able to choose whether he/she wants comments to be automatically published or whether he/she wants to validate them one by one. This choice should be part of the publication level definition.

A comment should have a title, a body, and user information. The latter could be a simple name, email, and website/blog URL, all of them being optional. However to simplify things down for registered users, the latter could use their current profile and not have to fill in the user information fields. Doing so incites users to own an account and to create a profile if they haven't done so already. Lastly, a comment should have a date field.

**Image votes**

We all love to give our opinion, we all love to vote, and the web has many sites that offer voting options: movie sites to elect the year best movie, sports site to elect a soccer player, and so on.

In a photo platform, it would be interesting to have users vote for their favorite images creating a sort of hall of fame. Olhares has already implemented the vote feature on its platform.

In addition to giving users the impression visitors actually like and rate their images, the voting feature makes the site more dynamic. Users are going to expect more and more votes and they will themselves vote for more pictures. Furthermore, it helps visitors know which images are more popular.

Unfortunately, there is a drawback: for a given user, voting several times for the same given image should be forbidden. Otherwise there would be «false» votes, and the score would have no value whatsoever. In order to prevent this, on the Amadeo platform, it will be required for visitors to be registered and logged in in order to vote. With their author id and image id, there is no way they could vote for another image. The drawback is that voting becomes restrictive. On the other hand, it might well incite visitors to register, thus increasing the number of users at Blog.com.

A vote will have an user id, an image id, and a score between 0 and 5.0 is useful to bring a score down in the case an user doesn't like a given image.

**A step further: playing with the image**

Describing, browsing might be useful. But it's not half the things visitors and users could do with an image. This section deals with ideas that could extend Amadeo in a near future.

Here is a list of «bright» ideas:

- Edit buffer: as client-server communication may be slow, it's best to do a maximum number of operations locally and then send them in one go. The edit buffer would remember what sort of image editing operations a user would want and when the latter decided to sent the whole «editing» batch, there would a single client-server communication to send the series of orders. It could a series such as rotate left – resize 50% - crop left – reduce quality...
• Edit history: this could be a database-based solution. Any alteration made to an image file could be stored in the database. The user could look up for a given image if there had been any alteration of any kind in order to cancel, for example, that alteration. For this feature to be truly useful, it'd be best to store a backup copy of the image file before editing it. This being disk space consuming, this could be an advanced feature for advanced users.
• Image cube: this could be an artistic way to display and interact with images.
• Image calendar: users could create various calendars by choosing an image and choosing a calendar type & format.
• Image wallpaper: quick and easy feature that allows users to choose an image and apply it as a background image. Most OS already have a feature offering this option.
• Image basket: one could image a visitor would want to download a specific set of images that belong to different albums and users. A simple feature would be to have a basket where visitors – whether registered or not – could add images to download later on.

The publication level

One could think of a privacy/publication system that would define typical behaviors. For instance a privacy level of 0 could mean that albums and images at that level require a password to be viewed. In addition comments wouldn't be automatically published. They would need user revision.

A privacy level of 1 could mean that albums and images still require a password, but that comments are directly published once posted.

A privacy level of 2 could mean that albums and images are public and that comments need user revision.

Finally, a privacy level of 3 could mean that albums and images are public, as well as comments.
New proposed architecture

Application class diagram

The following diagram lays out the new architecture planned for the Amadeo platform. Please note in the diagram the following important points:

- PersonalImageContainer and ImageCategory both extend ImageContainer. The latter extends DB_OO2, the data access API.
- The relationship between Image and Gallery is a qualified association, *id est*, for each image in a gallery there is a description.
- Images compose Albums and Galleries, but are aggregated in Category. This is due to the fact an empty category can indeed exist, whereas empty albums and galleries do not make sense.

![Image Class Diagram]

*Illustration 5 - The application class diagram*

**ImageContainer**

ImageContainer sets the basis of the sorting part of the Amadeo platform.
Attributes

- id: integer
- label: string
- description: text
- imageCount: integer

Methods

- ImageContainer(tableName: string, id: integer): class constructor. It takes in the name of the table to use in the database to construct the object with id « id ».
- exists(tableName: string, id: integer): function that checks whether the object with the given id exists in the database's given table.
- getImageCount(containerIdName: string, containerId: integer): retrieves the number of images the containerIdName table with id containerId contains.
- getImages(containerIdName: string, count: integer, offset: integer, order: integer): retrieves count images starting at offset in the required order belonging to the current container indicated by containerIdName.
- getList(tableName: string, count: integer, conditions: integer, order: integer): retrieves the list of count containers of type tableName in the order given matching the conditions given.
- getRandomContainer(tableName: string, minImageCount: integer): looks for a container of type tableName having a minimum of minImageCount images.
- count(tableName: string, conditions: array): returns the number of containers of type tableName matching the given conditions.

ImageCategory

ImageCategory extends ImageContainer and has no other attributes than that of ImageContainer.

In addition, ImageCategory has a series of methods bearing the same name as its parent methods and that call the parent methods. These methods are not described here.

Methods

- lookupRelatedTags(pattern: string, count: integer, popularity: integer, order: integer): looks for tags used in the current image category. Tags used in a category are tags used by images included in the given category.

PersonallImageContainer

Attributes

- authorId: integer, the id of the author of the given personallImageContainer.
- publicationLevel: integer representing the privacy level of the personallImageContainer and the behavior expected when comments are dropped in.
Developing a fotoblog platform

- lastupdated: date when the personalImageContainer was last edited
- creationdate: date when the personalImageContainer was created
- numthumbs: integer, number of images to display on each Amadeo page when the personalImageContainer is viewed.

**Methods**

- getAuthor(): returns the author for the current personal image container.
- getComments(personalImageContainerId: integer, personalImageContainerType: string, count: integer, order: integer): returns count comments the personalImageContainer has.
- getValidComments(personalImageContainerId: integer, personalImageContainerType: string, count: integer, order: integer): returns count comments the personalImageContainer has and that are valid.

**Album**

**Attributes**

- blogId: integer, the id of the blog the current album belongs to.
- showonblog: boolean, indicates whether the album should show up in its parent blog.

**Methods**

- getImageSize(unit: string, precision: integer): This function retrieves all the images for the given album from the database and calculates the total space they occupy on the disk. This function is only a member of the class Album as it is the only container actually containing images. It returns the total disk space in bytes, kbytes, or Mbytes (depends on unit) used up in the given album. Precision indicates the number of floating digits.
- lookupRelatedTags(pattern: string, count: integer, popularity: integer, order: integer): This function creates a query to look for the given album's tags matching pattern. Popularity is the number of images each tag describes.
- getTags(): this function calls the previously defined one with an empty pattern in order to retrieve all the tags an album has.
- getBlog(): this function returns the blog object to which the given album belongs.
- remove(): deals with the album removal, i.e. with the images removal, the images' comments, the images' annotations, the images' tags, the images' votes, and finally the album's comments.
- addImage(imageId: integer): adds an image to a given album.
Gallery

Methods

- `lookupRelatedTags(pattern: string, count: integer, popularity: integer, order: integer)`: This function creates a query to look for the given album's tags matching pattern. Popularity is the number of images each tag describes.
- `addImage(imageId: integer)`: adds the specified image to the given gallery.
- `removeImage(imageId: integer)`: removes the specified image from the gallery. It does not remove anything else. It does not delete any image or any object related to the given image.

ImageComment

Attributes

- `id: integer`
- `profileId: integer`
- `label: string`
- `body: text`
- `albumId: integer`
- `galleryId: integer`
- `imageId: integer`
- `date: timestamp with time zone`
- `valid: boolean`
- `authorName: string`
- `authorEmail: string`
- `authorUrl: string`

Methods

ImageComment extends DB ОО2. Therefore all the « setters » and « getters » are created on the fly.

- `ImageComment(id): class constructor.`
- `exists(): checks whether the given comment exists. Returns true or false.`
- `hasImage(): checks whether the profile associated to the comment has a profile image. Returns true or false.`
- `getPathToProfileImage(): reads the site's configuration files and returns the correct path to the profile image associated to the comment.`
- `getValidList(itemId: integer, itemType: string, count: integer, order: integer)`: reads the list of count valid comments in the order order that belong to item `itemId` of type `itemType(album, gallery, or image)`.
- `getList(itemId: integer, itemType: string, count: integer, order: integer, valid: boolean)`: reads the list of count comments in the order order that belong to item `itemId` of type `itemType(album, gallery, or image)`.
- `getAuthor()`: returns the author of the image comment.

Tag

Class Tag extends DB ОО2.
Attributes

- id: integer
- label: string
- popularity: integer
- numberOfSiblings: integer

Methods

- Tag(id: integer): class constructor. If an id is given, retrieves the matching data in the database.
- existsById(id: integer): checks whether tag bearing the id given in parameter exists. Returns true or false.
- exists(label: string): checks whether tag bearing the label given in parameter exists. Returns true or false.
- fetchByLabel(label: string): looks for tags in the database that have this exact label.
- count(): returns the number of tags used in the site.
- getList(count: integer, popularity: integer, order: integer): returns a list of count tags having a minimum popularity and in the order given.
- getSiblings(count: integer, popularity: integer, order: integer): looks for and returns a given tag's siblings (brothers). Tags are siblings when they describe the same image.
- isMySibling(tagId: integer): checks whether the current tag is sibling with the tag which id is given.
- cleanLabel(label: string): takes in a proposed label and strips it of any forbidden characters. Returns the cleaned label.

TagFamily

Attributes

- family: array of tags

Methods

- TagFamily(tags: array): class constructor. Takes in an array of tags to build the family.
- getLabel(): returns a string made of the labels of all the tags in the tag family.
- getFamilyMembers(order: integer): returns the members of the family in the given order.
- getNumberOfMembers(): returns the number of tags in the family.
- getSiblings(count: integer, popularity: integer, order: integer, labelFilter: string): returns at most count siblings of the given family. The siblings must have a minimum popularity and are sorted according to the given order.
- getImageCount(): returns the number of images the tag family applies to.
• getMatchingImages(count: integer, offset: integer, minVoteScore: integer, order: integer): returns the images described by the tag family and that match the minimum requirements sent in parameter.

**Vote**

Vote extends DB ОО2.

**Attributes**

• image: integer
• author: integer
• vote: integer

**Methods**
There are no other methods than those provided by DB ОО2.

**Image**

Image extends DB ОО2.

**Attributes**

• id: integer
• authorid: integer
• size: integer
• label: string
• description: text
• publicationlevel: integer
• uploaddate: timestamp with time zone
• albumid: integer
• type: string
• categoryid: integer
• creationdate: timestamp with time zone
• totalscore: integer
• numberofvoters: integer
• popularity: integer

**Methods**

• Image(id: integer): class constructor. If an id is given, the constructor retrieves the data in the database.
• getRandomImage(popularity: integer): retrieves an image at random from the database.
exists(id: integer): checks whether the image with given image id exists. Returns true or false.

getList(count: integer, conditions: array, order: integer): returns count images matching the criteria given in conditions.

getRelativePathToImage(size: string): reads the site configuration file and returns the appropriate path to the wanted image (mini nail, thumbnail, medium file, or original file)

getAuthor(): returns the author of the given image.

getAlbum(): returns the album of the given image.

getImageCategory(): returns the image's image category.

getAvgVote(): returns the average vote for a given image. It takes the total number of points that image has and divides it by the number of voters.

getEndPath(): computes the path to the image based on the author id, album id and image id.

getFilePath(size: string): call the getEndPath function and computes the path to the image asked for.

addTag(tagId: integer): adds the tag given in parameter to the current image.

removeTag(tagId: integer): removes the tag given in parameter from the current image.

hasTag(tagId: integer): checks whether the image has the given tag in parameter.

getTags(count: integer, popularity: integer, order: integer): returns a list of count tags, that describe the current image, with a given popularity in the given order.

getAdjoiningImage(container: string, containerId: integer, imageSide: string): in the given context (container and containerId), returns the next or previous image (depends on imageSide).

getPreviousImage(container: string, containerId: integer): calls function getAdjoiningImage.

g getNextImage(container: string, containerId: integer): calls function getAdjoiningImage.

getImageByUploadDate(date: timestamp): retrieves an image at random from the database that was uploaded at a given date.

getComments(): calls ImageComment::getList() and retrieves the list of comments the image has.

getValidComments(): calls ImageComment::getValidList() and retrieves the list of comments the image has.

getGalleries(): retrieves the list of galleries the given image is referenced in.

getDescriptionInGallery(galleryId: integer): returns the description the given image has in the gallery given in parameter.
• setDescriptionInGallery(galleryId: integer, description: text): sets the description the given image has in the gallery given in parameter.

• getMiniWidth(): returns the width of the mini nail used by the site.

• getMediumWidth(): returns the width of the medium image used by the site.

• getThumbnailWidth(): returns the width of the thumbnail used by the site.

The following functions deal with image manipulation.

• getImageFile(size: string): looks for one of the reduced images (mininail, thumbnail, medium file) and if it does not exist, then it creates it.

• getMiniNail(): calls the getImageFile() function.

• getThumbnail(): calls the getImageFile() function

• getMediumImage(): calls the getImageFile() function

• getOriginalImage(): calls the getImageFile() function

• getGDImage(): uses GD image reading functions (imagecreatefromjpeg for instance) and returns the « image source ».

• getDimensions(size: string): returns the current image's width and height for the given size (mini, thumb, medium, original).

• getWidth(size: string): calls the getDimensions function and returns the width.

• getHeight(size: string): calls the getDimensions function and returns the height.

• generateSquareFile(size: string, wantedWidth: integer): generate a square thumbnail for display in Amadeo (it's a smart crop 'n resize).

• generateMediumFile(): creates the image file with medium dimensions (it is a simple resize).

• generateFile(size: string): calls any of the two previous functions depending on size.

• getCameraData(): reads camera-related data in the exif part of the image.

• getImageData(): reads image-related data in the exif part of the image.

• rotate(angle: integer): simple rotation function. It deletes any nail that had been created previously.

• flip(mode: string): simple flipping function. It deletes any nail that had been created previously.

• compressBy(compressionRate): for jpeg images only, reduces the image quality. This function is specially useful if users realize they've used up their space quota.

• deleteAllNails(): removes mini, thumb, and medium nails.

• deleteFile(): removes the image file from the hard drive.

• moveToAlbum(albumId: integer): moves the image file to the targeted album's folder.
developing a fotoblog platform

- remove(): removes the image object. It removes the row from the table in the database and any object depending on it: the tags, the image votes, the image comments, the image nails, and lastly the image file itself.

**Database specification**

The final Amadeo schema, fotoblog, will contain all the data needed to run Amadeo, apart from tables Blog, and Author. Here is the table list and description:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Album</td>
<td>table</td>
<td>Stores album data</td>
</tr>
<tr>
<td>Category</td>
<td>table</td>
<td>Stores category data</td>
</tr>
<tr>
<td>Comments</td>
<td>table</td>
<td>Stores image comments data. Not to be confused with public comment which is the table containing post comments.</td>
</tr>
<tr>
<td>Gallery</td>
<td>table</td>
<td>Stores gallery data</td>
</tr>
<tr>
<td>Image</td>
<td>table</td>
<td>Stores image data</td>
</tr>
<tr>
<td>ImageInGallery</td>
<td>table</td>
<td>Stores the relation between images and galleries. The relation being a multiple-to-multiple, this table is needed.</td>
</tr>
<tr>
<td>Publication</td>
<td>table</td>
<td>Should store the different levels of publications that should exist.</td>
</tr>
<tr>
<td>Tag</td>
<td>table</td>
<td>Stores each tag used in the website</td>
</tr>
<tr>
<td>TagInImage</td>
<td>table</td>
<td>Stores the relation between images and tags. The relation between one-to-many, the table is needed. In addition, one needs to add a constraint to make sure a tag is not applied twice to the same image.</td>
</tr>
<tr>
<td>Vote</td>
<td>table</td>
<td>Stores vote data</td>
</tr>
<tr>
<td>Extended_album</td>
<td>view</td>
<td>This view takes table Album, table Image and precalculates the number of images in each album.</td>
</tr>
<tr>
<td>Extended_category</td>
<td>view</td>
<td>Same as above</td>
</tr>
<tr>
<td>Extended_gallery</td>
<td>view</td>
<td>Same as above</td>
</tr>
<tr>
<td>Extended_image</td>
<td>view</td>
<td>This view takes table Image, table Vote, table TagInImage and calculates the image's popularity (number of tags it has), the total number of points it received and how many voters gave their opinion</td>
</tr>
<tr>
<td>Extended_tag</td>
<td>view</td>
<td>This view takes table Tag and table TagInImage to calculate the tag popularity (number of images it describes) and the number of siblings it has.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Popularity_tag</td>
<td>view</td>
<td>These three views are intermediate views used to simplify the queries used. They are based on Tag and are used by others views.</td>
</tr>
<tr>
<td>Related_tag</td>
<td>view</td>
<td></td>
</tr>
<tr>
<td>Sibling_list</td>
<td>view</td>
<td></td>
</tr>
</tbody>
</table>
V. Development of the photo platform

Development process

Organization

It is important to install, write and develop the database first as the API-to-be cannot be developed with its data layer.

To make sure the new development will not disturb the current Blog.com platform, the new API and the database will be separated at first and a simple public interface will be developed in order to test the API functionalities.

The Amadeo database schema (fotoblog) will be a standalone database at first called fotoblog. Then when the new API is up and running and tests have been run, the new database will be integrated, as previously mentioned, into the current database.

As for the API itself, the development should be done based on the dependencies that exist in between classes. Cul-de-sac classes such as Tag should be developed first. As for classes that extend other classes, it is obvious the parent class should be developed first. In this case, ImageContainer, then PersonalImageContainer, then Category, Album, and Gallery.

Technology used

The platform will be written in PHP 5.0, based on an object model, as seen in the Design chapter, will use DB_OO2 to access the data layer, and Smarty to generate the graphic user interface (GUI).

The server is an Apache 2.0 server compiled with the following specific Gentoo flags:

```
  exif gd -gpm php -tidy -xmlrpc
```

Tags indicate which options are to be built into the Apache binary and which are not. For instance, as the Amadeo platform uses Exif, it is mandatory to compile Apache with that option activated.

The database will be a PostgreSQL 8.0 relational database.

Developing the database

Creating a new schema

PostgreSQL, version 8.0 allows views, rules, and triggers which are used in the development of the Amadeo platform. In addition, PostgreSQL allows schemas to be created. Schemas are a set of tables, views, rules specific to a given application, or application part. They permit a better organization of the database. Schemas can communicate in between themselves.
Migrating the current system to the new system

We currently have two databases, com_blog and fotoblog. We want to export the second database's architecture to a separate schema, com_blog.fotoblog in the first database and then feed data from the first database's main schema com_blog.public to the new schema com_blog.fotoblog.

The final database will have two schemas:

- public
  It deals with all of Blog.com except for the image-related data.
- fotoblog
  It deals with all the tables related to images and containing image data.

The following tables should be removed from the database altogether: author_album, blog_photo, photo_album.

The following tables should be transferred to the new fotoblog schema: album, annotation, and photo (which should change its name to image).

Auto-incrementing numbers (SERIAL) often used in PostGREs are based on a memorized sequence number in the database. Each time a new row is inserted into a table with a SERIAL, the right value is stored in the row and the memorized SERIAL is incremented.

However, upon creation of the new schema, the sequence numbers all are initially set to 0. After the schema's table have been automatically filled in, the sequence numbers should be reset to the highest current value plus one.

Furthermore, some foreign keys should be altered. In the new schema fotoblog, all the foreign keys pointing to fotoblog.author.id should now point to public.author.id. The same goes for fotoblog.blog.id which should change to public.blog.id.

Developing the website

As mentioned earlier, the website architecture has three layers:

- the data layer
- the API level
- the graphic user interface level

DB ОО2 takes care of the communication in between Amadeo’s API, and the latter uses Smarty to format its results and output it to the visitor's browser.
The site's architecture

The Amadeo platform will be integrated to Quill from now on. Indeed, Quill uses all of Amadeo's API and it makes more sense to have a single platform.

Architecture overview

The architecture (where is stored what) follows the main Blog.com guidelines. The new platform combining Amadeo and Quill is stored in a top folder labelled for instance « F ». F contains the website's configuration files (that deals for example with breadcrumbs), and the following three folders (among others irrelevant in this study and therefore not mentioned):

- docs
  
  docs includes all the files developed and that make the site up, that use the Amadeo API and call Smarty to output webpages.

- includes
  
  This folder contains a subfolder called api which itself contains all the class sources developed during this project.

- smarty
  
  This folder contains the templates to be used by the docs/ PHP files to output data to the browser.

In the diagram below, one can see the source organization. Each Amadeo folder contains an architecture directly based on the API: there are several folders among which album, gallery, image.

The folder actions contains all the scripts that actually manipulate data and/or images. For instance, a web page that alters an image description will call a script stored in actions/photos/edit/.
Note that in the architecture, the name « photos » remained to prevent naming confusions with the Images folder that contains image files used by the website (logos...) as well as the users' images.

*Illustration 7 - Basic site organization & architecture*

**Storing the images**

The path to an image will be as follows:

`/images/authorId/albumId/imageId.(jpg|png|gif)`

The images are broken down into folders that bear the images' album id. Each album folder is stored in a parent folder representing the user. The folder's label will be the author's id.

Lastly, the image name is the image's id.

This solution will facilitate album downloading and gives the webmaster a clear overview of the images stored. On the other hand, the « simple » album edit operation for a specific image is slower. When a user wishes to move an image to an album, the server will have to move the image to another folder.

**Developing the public interface: Amadeo**

The public interface was initially thoroughly studied in a document labelled « Os Use Cases do blog de fotos – Parte pública » (Photo blog – Public interface use cases). This document described the main pages in the public part.

These pages are as follows:

- Amadeo home (for instance [http://amadeo.blog.com](http://amadeo.blog.com))
- Albums home, two modes
  - no album id given: the list of the latest albums is displayed
  - an album id is given: the album information and a series of images are displayed
- Galleries home (same behavior as above)
- Categories home (same behavior as above)
- Tags home: a series of tags are shown: a random tag family, a set of popular tags, latest tags.
- Tag home: provided a tag id has been given and is valid, then the tag information is displayed. This very same page is used to display tag families, in which case more than one tag id are sent to the page.

In addition to these pages, there are elements in common. Those elements are:

- Amadeo’s header
  It contains Blog.com’s and Amadeo’s logos. In a near future, when a better interface is designed and CSS models are written up, it will also contain a quick menu to browse through Amadeo.

- Amadeo’s left menu
  It contains Amadeo’s main menu with links to the Albums page, Categories page, Galleries page...
  In addition, it contains the login section where a user can authenticate. The left menu also contains a tag soup: a selection of tags at random. Moreover it contains the user’s information and a link to the editor (quill) if the user is logged in.
  Lastly, if the user logged in visits his/her own albums/galleries/images, he/she has a list of smart shortcuts he/she can use to go directly to Quill to edit the album/gallery/image he/she is viewing.

- The copyright footer

- Amadeo’s right menu
  The right menu contains Amadeo’s category list (similar to Olhares).

- the page footer.

Each separate part is a single template file called by the parent template (for instance the album page script) and assembled together.

Because the former Amadeo platform didn’t have the same style as the rest of Blog.com, and wasn’t based on templates at all, the templates needed to be created from scratch.

Below, the initial planned Album layout and a screenshot of the final Album page when no album is given.

Note there is a search engine. It is used to look tags up and returns to the page from where it was called. There are two ways of searching tags: either in the current context, or in the whole site.
Developing a fotoblog platform

Illustration 8 - Planned album page layout

Illustration 9 - Amadeo's Album page for album 11063

Developing the editor interface: Quill
Now that Amadeo's public interface is complete, one needs to build the Quill part that will take care of the Amadeo Edition side, or user private area. This is perhaps the most important part as it needs to be clear and simple (one reason for under-describing is the complexity of the editors).

Quill exists for blogs, posts, and images. It is a given user's backoffice where he/she can edit his/her items.

Therefore there are already template files and scripts in common to be used in each new file. Those files are the following:

- header.php
  This file contains Quill's quick access menu and calls the corresponding template file.

- tip.php
  This file contains the left part of the page that displays a useful tip.

- AssignBreadcrumbs.php
  This file calls the breadcrumb generator and displays the results below Quill's menu.

- footer.php
  This file contains the privacy statement link.

Quill requires many more action scripts than Amadeo. Amadeo being the publication platform is relatively passive (save the comments and votes). On the other hand Quill is highly dynamic. Therefore action scripts are plentiful in this part.

Since users manipulate data, the system needs to communicate back to the user. Notifications are used. There are three levels: success, warning, error. Here's an example screenshot below:

![Warning message]

Illustration 10 - A sample notification in Quill

Due to a lack of time, the Quill editor interface for Amadeo was only partially developed. For instance, even though the Gallery class is fully developed, the interface hasn't been set up yet.
Here are some other sample screenshots of the Quill - Photos application.

Illustration 11 - Quill Photos Summary Info

Illustration 12 - Quill Albums control panel

Illustration 13 - Quill Tags control panel
VI. Conclusions & Future Work

Blogs are the trend nowadays, there is no doubt they will keep on growing in the next couple of years. They have become an entirely new part of the web and have shaken off the image of fashion some stuck to them.

The Economist, a financial newspaper, once published an article about blogging. For the author of the essay, blogging or rather chit-chatting is nothing new. Coffee houses in XIXth century London had numerous gossipers, or everyday people exchanging their thoughts to fellow men. Much like blogs today, coffee house discussions covered a broad range of topics, from law to taxes, or entertainment: theater back then, movies nowadays.

Indeed, blogging isn’t just a phenomenon that will disappear when web surfers find a new fashionable widget. After talking it over with Portugalmail’s CEO and after reading several papers, I now feel convinced blogs are here to stay. The right question is in what shape? Who will be there?

The latter question is a matter of battling for the share of the market, offering users and customers the right product at the right price. This marketing aspect might influence R&D and therefore the aspect blogs will have.

The former question is the most interesting. This paper has dealt with the design of a photo platform. Images are probably the most readily available type of media online. Even though sound and videos are popping up everywhere online, they are still a media reserved to those lucky ADSL or cable users. Photos will therefore be an important element in the blog evolution. Needless to say that users can expect of their blog provider to have minimal image support. What would be a blog without a banner, a profile image, or the latest vacation pics for friends to view and perhaps download?

Keeping that perspective in mind, I tried to develop a platform that can be extended, reworked easily and always growing. Working on images is particularly interesting as they need a broad description system. I was able to develop important notions such as albums, categories and galleries. The latter is, I believe, a relatively new idea unseen elsewhere. Inspired directly from art galleries, who choose « items » (e.g. paintings, sculptures...) from different sources (museums, private collections, artist him/herself) to then display them, explain to the public. Galleries combined with albums allow for cross-referencing. In some ways, galleries extend albums. Whereas albums have this notion of physical container, galleries seem more presentation-oriented. And the result, if galleries are indeed well used, is amazing. Users can reorganize images the way they want them. They can take images from their own albums and « save » them to a new gallery, becoming in some ways e-artists.

Tags also bring new perspectives, not only to the image world, but also the web. Del.icio.us, a bookmark aggregation website, uses tags to sort users’ links. Tags have allowed more connections to be interwoven in between contents and users. In the specific case of images, where images were previously disconnected, they now take part in a web of meaning. Not only it is simpler to correlate images by the simple use of tags, but it eases browsing. Viewing images becomes more of a game, an interactive experience. The popular Flash game « Guess-the-Google » is an illustration of what can be done with tags.
But of course an image platform is not just about sorting or publishing online. Users expect more from an image, more specifically a photography, than ever before. And companies also aim at creating needs. Photoways and Ofoto – sites mentioned in the preliminary study – both strive at making profits by selling derived products. Whether it be a mug with a picture, or the latest miniature photobook for «Mummy to slip in her purse», ideas surge. Games can also be online. In the month of July, for instance, Blog.com developed «Faces of Blogs» creating a mosaic of images using its users' profile images. The result is a stunning new way of browsing blogs. Other more photo-oriented games could be «Guess-the-Blog», a game inspired by «Guess-the-Google», or a photocube, or a scrambled image. There are truly few limits to the photo platform. The only boundaries might well be time and imagination.

Blogging is also undergoing another revolution with the rise of cellphones, specially camera-equipped cellphones. A new word has even been coined: moblogging, id est blogging with a cellphone / mobile phone. A user's on the move? He/she wants to jot down a couple notes in his blog? It's easy. Either by using WAP access, or SMS/MMS, he/she can do so with some blog providers. And of course, he could even post photos. With mobile phone commercials inciting users to take pictures and with the constantly increasing quality of phone cameras, who isn't taking a picture of every moment we live? In the subway, out in the street...

With such a great influx of information and data of all sorts (text, sound, video), the web is growing faster than ever. The web needs a sorting system. Search engines that were created a while back might not just be enough, and researchers are looking into what is known as «semantic web». As one knows, blogs have syndication tools, such as RSS, or Atom, a slightly different version of RSS. Syndication tools help users track updates made to their favorite sites or blogs. One can use syndication much as he checks his emails. As a matter of fact, many mail clients offer the possibility to track RSS content. The semantic web is about so much more. It is about giving a meaning to everything and making everything easily searchable, sortable, reachable. Semantic comes from the Greek Semantikos which means «significant, meaning». The future of the web lies in the intensive development of the semantic web. Steve Cayser gives us a vision of what semantic blogging can be in his paper on semantic blogging:

The semantic web promises to make the web more useful by endowing metadata with machine processable semantics. Blogging is a lightweight web publishing paradigm which provides a very low barrier to entry, useful syndication and aggregation behaviour, a simple to understand structure and decentralized construction of a rich information network. Semantic blogging builds upon the success and clear network value of blogging by adding additional semantic structure to items shared over the blog channels. In this way we add significant value allowing view, navigation and query along semantic rather than simply chronological or serendipitous connections. Our vision is to use semantic web tools and ideas to help move blogging beyond communal diary browsing to rich information sharing scenarios.

This placement allowed me to learn many more things about web technologies I was already familiar with. Moreover I learned about organizing work, about organizing code.
development. For instance, I became familiar with Smarty, the template « engine » that allows users to create websites and ensure data is separated from formatting structures. I also became familiar with PostgreSQL databases, whereas I'd only used MySQL databases for web development. I became familiar with the Pear website and the many APIs it offers. But most of all, I finally learned the basic ropes of Linux with a Gentoo installation, and along with it minimal server management (Apache 2). I learned to install a server with the right options, and not just rely on the ready-made Windows binary such as the popular EasyPHP.

The placement wasn't just about designing, specifying and developing a web solution. I also had the opportunity to create the company's communication blog (http://our.blog.com) dubbed « The itch to write ». This blog allowed to do what I like best: writing. Posts dealt with books I'd read, films, or even etymology. The company's blog has a capital importance in communication. It helps to bring Blog.com users together and it gives them a clearer view of who is Blog.com. It humanizes in some ways what may seem like just another website.

However, in the near future, I chose to work for a different company, in a different type of position. I'm off to England to work in web security research. It is needless to say that many things I learned during the six months at Portugalmail will prove useful.

Nonetheless, once again, I'd like to thank all the Portugalmail staff for their warm welcome, their help, their support. I'd like to thank them for the great opportunity I had while working on Blog.com. I'd very much like to stay in touch with the Blog.com team and continue developing.

VII. Bibliography


VIII. Appendix

Solutions comparison chart

(Extracted from « Estudo dos fotoblogs existentes e das melhoras possíveis)
### The site

<table>
<thead>
<tr>
<th></th>
<th>Blog.com</th>
<th>Wistiti</th>
<th>Flickr</th>
<th>Ophoto</th>
<th>PhotoWays</th>
<th>Digibao</th>
<th>ImageShack</th>
<th>Olharex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>PHP</td>
<td>ASP</td>
<td>hosting</td>
<td>JSP</td>
<td>PHP</td>
<td>?</td>
<td>PHP</td>
<td>PHP</td>
</tr>
<tr>
<td>Purpose</td>
<td>publication</td>
<td>photo processing</td>
<td>publication</td>
<td>photo processing</td>
<td>photo processing</td>
<td>publication</td>
<td>publication</td>
<td></td>
</tr>
</tbody>
</table>

### Image upload

| Image format     | JPEG, GIF, PNG | JPEG, BMP, PSD, PNG | JPEG, GIF, PNG | JPEG | JPEG | jpeg, jpg, gif, bmp, tif, swf, & png, bmp, tif, & tiff converted to png |
| Max image size   | 10 MBytes      | 10 MBytes           | unlimited      | 5 MBytes  | 80 MBytes / upload | 1024 kBytes | 2 MBytes   |
| Max upload       | 10 MBytes/month, unlimited with a paid account |

| Image web form   | Yes, 1 image | yes, up to 10 images | yes | yes | yes | yes | yes | yes |
| Zip/Tar web form | yes          | no                  | no  | no  | no  | yes (registered users) | no |
| FTP              | no           | no                  | no  | no  | no  | yes | yes | no |
| Multiple file    | yes via zip  | yes (JAVA applet) up to 100. | yes | yes | yes | yes | yes | no |
| Local Software   | no           | no                  | yes | yes | yes | yes | yes | no |
| Mail-a-CD        | no           | yes                 | no  | no  | no  | yes | no  | no |

### Signing up

<p>| Free account     | yes | Yes – no restrictions whatsoever | yes | Yes – limited to 1 Gbyte storage space | yes | No registration process | 25 photos max. | 1 photo/day |
| 10 MBytes/month upload limit | Display of up to 100 |</p>
<table>
<thead>
<tr>
<th>Blog.com</th>
<th>Wistiti</th>
<th>Flickr</th>
<th>Ofoto</th>
<th>PhotoWays</th>
<th>Digibao</th>
<th>ImageShack</th>
<th>Oihares</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no</td>
<td>1GB/ month</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>Unlimited number of photos 3 photos/day max upload</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>-</td>
<td>US$50 / year</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Image Shack is based on a point system. Users can earn or buy points to make extra uploads.</td>
<td>1€/month</td>
</tr>
<tr>
<td><strong>Payment methods</strong></td>
<td>-</td>
<td>Credit card</td>
<td>Credit card (Paybox) or check</td>
<td>Credit card</td>
<td>-</td>
<td>MB, PayPal</td>
<td></td>
</tr>
<tr>
<td><strong>Publication level</strong></td>
<td></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Faculty and friends</strong></td>
<td>yes with password (eg <a href="http://www.wistiti.fr/my">http://www.wistiti.fr/my</a> gallery.pwd)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>Public</strong></td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Description, organization, browsing**

<p>| Albums | yes | yes | yes | yes | yes | yes | no | yes |
| Galleries | no | no | no | no | no | no | yes | no |</p>
<table>
<thead>
<tr>
<th>Blog.com</th>
<th>Wistiti</th>
<th>Flickr</th>
<th>Ofoto</th>
<th>PhotoWays</th>
<th>Digibao</th>
<th>ImageShack</th>
<th>Olhares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Favorites</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Categories</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Tags</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

### Auto-managed categories

<table>
<thead>
<tr>
<th>The best</th>
<th>no</th>
<th>yes</th>
<th>no</th>
<th>no</th>
<th>no</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>The most popular</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>The latest</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Image display

<table>
<thead>
<tr>
<th>Thumbnail</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original file</td>
<td>yes</td>
<td>no</td>
<td>no (except in paid account)</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Slideshow</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Random picture</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Code generation</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### User interaction

<table>
<thead>
<tr>
<th>Comments</th>
<th>no</th>
<th>yes</th>
<th>no</th>
<th>no</th>
<th>no</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private messaging</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Votes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Forum</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>RSS Feed</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

### Help

<table>
<thead>
<tr>
<th>FAQ</th>
<th>no</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer support</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Shop

| photo processing | no | yes | no | yes | yes | yes | yes | - | no |
| Derived products | no | Mugs, T-shirts, | no | Greeting cards, | Photos, books, cameras, | no | - | Books, cameras, |
## Developing a fotoblog platform

<table>
<thead>
<tr>
<th>Blog.com</th>
<th>Wistiti</th>
<th>Flickr</th>
<th>Ofoto</th>
<th>PhotoWays</th>
<th>Digibao</th>
<th>ImageShack</th>
<th>Olhares</th>
</tr>
</thead>
<tbody>
<tr>
<td>posters...</td>
<td>books, calendars, mouse pads...</td>
<td>calendars, greeting cards...</td>
<td>microsites...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Overall appreciation

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy-to-use interface.</td>
<td>Lack of advanced tools and a better navigation.</td>
</tr>
<tr>
<td>Easy-to-manage albums, 3-level publication.</td>
<td>The online editor does not always work fine.</td>
</tr>
<tr>
<td>Tag-based browsing, annotation possibilities</td>
<td>With a free account, it is impossible to retrieve the original file. Furthermore, navigation is solely tag-based.</td>
</tr>
<tr>
<td>Comprehensive online image editing tool.</td>
<td>Album publication tools</td>
</tr>
<tr>
<td>Image editing software.</td>
<td>The Help section is not up-to-date. No online editing tools.</td>
</tr>
<tr>
<td>Several upload methods</td>
<td>Photo sharing is not easy.</td>
</tr>
<tr>
<td>Code generation to cut 'n paste in emails, sites, forums... Simple image storing</td>
<td>Site for advanced users. Minimal interface.</td>
</tr>
<tr>
<td>Image quality, user interaction, community life.</td>
<td>Image dimensions cannot exceed 500*500px</td>
</tr>
</tbody>
</table>
The photo platform's public use cases

Album

- Check user status & rights
- Create Album
- Edit Album
- Publish album
- Unpublish album
- Delete Album
Browsing images

Developing a fotoblog platform