Advanced Tools for Application Development
Group Assignment

Final Report

ARA Group
Group components in alphabetical order of surnames

Liesbeth Claessens
lc55@alu.ua.es
Alberto García García
agg180@alu.ua.es
Pablo Martínez González
pmg66@alu.ua.es
Sergiu Ovidiu Oprea
soo6@alu.ua.es
Brayam Stiven Zapata Impatá
bszi1@alu.ua.es

June 3, 2013

Contents

Free software permits students to learn how software works. Some students, on reaching their teens, want to learn everything there is to know about their computer and its software. They are intensely curious to read the source code of the programs that they use every day. To learn to write good code, students need to read lots of code and write lots of code. They need to read and understand real programs that people really use. Only free software permits this. Proprietary software rejects their thirst for knowledge: it says, "The knowledge you want is a secret—learning is forbidden!" Free software encourages everyone to learn. The free software community rejects the "priesthood of technology", which keeps the general public in ignorance of how technology works; we encourage students of any age and situation to read the source code and learn as much as they want to know. Schools that use free software will enable gifted programming students to advance.

Richard M. Stallman

1 Introduction

This is the final report for our group project of the subject. In this document we are going to make a complete overview of our project from its origins to the last changes and opinions.

We have divided the report in nine sections: the introduction, proposal, initial planning, the project itself, user manual, final planning, troubleshooting, enhancements and conclusions. First of all we are going to explain briefly what we are going to do in each one:

1. **Introduction.** Self explanatory.
2. **Proposal.** In this part we will introduce our idea as we did in the previous assignments and the presentation. We will explain the motivation of our project, the underlying idea, how we are going to make it real and what its components are.
3. **Initial Planning.** In this section we will show the initial task distribution that we proposed for the development of the project. The project will be divided among the group components. This was the division for the original idea of the project so it changed a lot during the development; those changes will be reflected in the final planning.
4. **The Project.** In this section we will explain the composition of the project in terms of code and design. We will introduce the basic division: class library, interface and ASP application. We will also talk about the workflow using Git.
5. **User Manual.** This is a small document which will guide non-experienced users through their first contact with our application. We will explain what tasks our application is able to perform and how to carry them out.
6. **Final Planning.** Once the project was finished we were able to know the actual tasks performed by the group components. We will also talk about the divergences with the initial planning and their causes.
7. **Troubleshooting.** We will discuss the encountered difficulties, how we overcame them or why we changed certain specifications to avoid problems. This is the non-trivial problem solving section.
8. **Enhancements.** As the different sections of the project started to work properly, we were thinking in parallel about how to improve them. We will show the different enhancements that we have applied to the project.
9. **Conclusions.** Some final words about the project, the group and personal opinions.

These sections will cover the majority of our project but some things will be missing. We will not explain or paste code directly to this document; the reason is obvious: it is not the purpose of the report. Instead of doing that we have fully commented the whole class library and the more conflictive parts of the ASP application so that any programmer can understand what we are doing and why. Obviously, we do not expect a non-experienced user to try to understand those details and that's why we have not included them here.

2 Proposal

In this section we are going to describe the underlying idea of our project, the way we are going to move from the idea to the real application and the basic components (in abstract terms) of the project/community.

2.1 Description

Our idea is inspired by the work done by Richard Stallman and the GNU Project, the vast majority of our project is going to be focused on the principles of the free software paradigm (which shouldn't be confused with free of charge). As we said, our main source of inspiration is the collection of essays of Richard Stallman: *Free Software, Free Society* (this book can be downloaded for free from the GNU server).

2.1.1 The Key Point

We firmly believe that this equality should always be accomplished:

Software = Freedom + Collaboration + Creativity + Evolution

However, all of those nouns are often confused with free of charge software. Nevertheless, a good software should be funded according to its quality and usefulness but the main goal must remain the same: a contribution to the community which can be shared by everyone.

2.1.2 What are we going to do?

In order to fulfil the conditions of free software and at the same time include the terms creativity, evolution and funding within our idea we are going to develop a free-collaborative-creative-and-evolutive software development community based on big projects and small features that will be funded and rated by the community itself.

The community will be shaped as a website where two different kind of users will interact: developers and contributors. Both of them are registered users, non-registered users will have a more restricted view of the website.

2.2 The Community

As we said previously, we are going to develop a community. All communities require entities that interact to perform actions or produce results. Our project is no exception, it will obviously be composed of many entities that will require others to "move" in order to produce tangible results. In this section we will talk abstractly (not in code or computer like terms) about the entities that inhabit our community: the projects, the users and the credits.

2.2.1 Contributors

The contributors will be users who need a program or an application and for some reason they can't carry it out (maybe because they lack the required tools or knowledge, or simply because they don't want to do it). Their role in our community will be the following: they will propose projects, for instance a task manager (like Wunderlist); along with that proposal, the contributor must provide (invest) credits to the project; those credits will be stored for the project and many other contributors may find the proposal useful for them and then they will give more and more credits to the project creating the piggybank for the project which is the sum of all the invested credits.

2.2.2 Developers

So far, we have projects with credits accumulated in a jar, but what's the purpose? Well, the main objective of the contributions is to draw the attention of the developers. Developers are users with the knowledge needed to design and create programs (if they are experts) but they can also program small pieces of code which will improve the project (a small contribution) if they lack the experience of a senior programmer. In this way, our community is open to all levels of contributions, skills and creativeness.

2.2.3 Unregistered Users

Unregistered users can gain access to projects in any state (recently presented, in development, or declared completed) but can not contribute with his/her development or credits. These project pages will have statistics such as number of developers working on it, credits contributed to its development, popularity of the project on the community.

Additionally, we are considering adding a section accessible without prior registration, where you can view or search algorithms and simple programs, organized by type, language, complexity and difficulty. For example, it would be interesting to look for the Quicksort algorithm, and easily see its implementation in different programming languages. Registered users will be able to post in this section, code (less concrete than in the private section) that want to share easily with anyone on the network. The purpose of creating this section is to create a real platform to unite developers with "normal" users, for example, as students in order to increase the thirst of knowledge in every way as possible.

2.2.4 Projects

A developer may be interested in a project so that he/she decides to work on it. Once the work is done, the developer will publish the code and let the contributors rate his/her work to determine what percentage of the jar will be given to the developer.

In order to do that, the project must meet the specification that was proposed by the creator of the project.

The only one condition to work on the project and to be allowed to get the credits from the jar is that the project must be free. The developer rejects any copyright, and once finished the code will be open for downloading, modification and reusing by the contributors (and for the developers too).

What's more, another developer may come with a new idea, he/she is able to get the code and start improving the project. Once the improvement is finished, the contributors will be able to rate it and distribute the new funds of the jar among all the developers following the principles of meritocracy.

A synergy is established in the community, so that you have to contribute and give something back to the society in order to be able to take advantage of the work done by others. You may improve the project or contribute, it is up to you. By doing this, the developers keep earning credits and reputation thanks to their works and the contributors will get an application that is constantly evolving thanks to the contribution of other developers who also want to work on it and get a part of the jar.

Everybody wins, and knowledge is open if you contribute to the community avoiding the "lurkers" which often populate this kind of networks.

2.2.5 Credits

In our community, credits are money, they are just an intermediate kind of coin which can be used to fund projects or earn benefits. Users can buy credits using a PayPal gateway for instance. It's a safe way to ensure that no money is lost so that the final balance is never negative.

They can of course be changed in reverse for real money. It's moreless the idea behind casino chips. You can buy casino chips using real money and then play with them in all the games; after that you'll hopefully have chips left (maybe more, maybe less) which you can exchange for real world money.

2.2.6 The Point of the Community

Now, we might ask what the purpose of all of this is. For us, this signifies an important opportunity to bring together two worlds: the developer's world looking for work and projects in which they can participate, and the creative people's world with innovative and great ideas but lacking the sufficient knowledge to carry them out. With this, we will provide a new portal and community where developers can choose the projects in which they want to work and meet other developers to work together on ideas that are creative and useful for the society.

2.3 Interacting with the Community

We have introduced the different entities that populate our community and we have established their relationships, actions that make the community grow and move. Now we are going to put ourselves in the perspective of a user, a living entity which can interact with the community. We are going to explain which actions can be performed by a stranger (non-registered/public part) and which ones can be executed by an old citizen (registered/private). An important remark to make is that the public part is a subset of the private part.

2.3.1 Public Part

The public part of the community comprises the home of the community which will show the news, you can also explore the projects and see their descriptions (but you cannot download the code, contribute or develop), you can also fill the contact form to send messages to the administrator of the community with suggestions.

Of course, you can access the sign up form to create a new user and gain access to the private part of the community which is reserved for our most precious users.

2.3.2 Private Part

When you are a registered user you can access the whole community and you unlock the complete power of our application.

In the private part of the application you can propose new projects, contribute to existing one or even develop new branches. You can also rate the developments that other users have made or comment the projects descriptions to suggests new features.

In addition to that, a new level of social networking is added and you can use a messaging system to communicate with other users via private messages.

You can also access your profile and buy credits to contribute in projects. In your profile you will be able to see your status: credits, developments, contributions, etc...

In other words, by registering yourself as a new user you are joining the community.

3 Initial Planning and Task Distribution

In this section we are going to present the initial project distribution that we made. It is important to notice that it is not a fixed planning; during the project development we decided that we made some bad decisions in the initial planning and we changed things accordingly. However, the core concept of the project task partitioning will remain almost intact.

This initial distribution takes into account the first and the second deliveries (Class Library and Interface Design) as already done and the rest of the project as remaining tasks. We remark again that some changes may arise from this planning to the final one.

The initials of the table stand for Class Library (**CL**), Design and Interface (**DS**), Sections (**S**), ASP code and Database management (**ASP/DB**), Documentation (**DOC**) and Other (**O**).

C	CL	DS	S	ASP/DB	DOC	O
Liesbeth	TopicBE, NewsBE and TopicDAC, NewsDAC	News feed and panel interface	News	News home page, control panel for writing news and feed for the users	Inline code comments in ASP pages	-
Albert	ProjectBE, ContributionBE and ProjectDAC, ContributionDAC	General CSS Design of the whole application and User Interface	Users	Database Design and Implementation. Exception Management	Full DACs and BEs comments, Second and Final Reports	Design Pattern: Singleton implementation
Pablo	MessageBE and MessageDAC	Message inbox and sending Interface	Messages	Messaging System and Sign Up form	First and Second Delivery Reports	Presentation
Sergiu	UserBE and UserDAC	Default Page Banner Design	Additional Technologies Research	AJAX extensions and LINQ implementation	First and Second Delivery Reports	Presentation
Brayan	CommentBE and CommentDAC	Project search, visualization and description Interface	Projects	Session control and project exploration (search, create and view)	Inline code comments in ASP pages	Nested Master Pages implementation

This planning was originally designed to distribute the tasks fairly so that all the members of the group have the same workload. Although some tasks are more visible than others it is important to remark that the "dirty" work has to be done by someone and thus the task distribution has to be analysed carefully.

We estimated that each component of the group will spend 45 hours approximately to perform all the assigned tasks. We are not counting the practical hours which sum up 30 hours in class.

4 The Project

In this section we are going to describe our project internally. We will talk about the three components division: class library, ASP application and interface. We are not going to comment or show code, we will just talk about the project internal organization in terms of source code files and directories.

Basically, our Visual Studio project which can be viewed by opening the *ProjectShode.sln* file in the root directory of our project *ProjectShode*. This project is composed by two different solutions: *ShodeLibrary* and *ShodeProject*.

The first one is the C# class library of our application and the second one is the ASP Web Application which uses the previous class library in order to implement the BE/DAC model.

In the last subsection we will talk about our interface and how it was designed.

4.1 Git Repository

In order to facilitate the collaborative work we are using the GitHub platform to track the changes in the project code.

You can access our GIT repository here: http://github.com/Blitzman/ProjectShode

We are using the GitHub tool for Windows with the PowerShell to handle Git commands.

4.2 Class Library

Our class library is a standard C# that contains business entities and data access components for each one of the entities that compose our project/community.

In our case we have six different entities in our project:

1. **Comment.** Represents a single comment that can be made by the users in the projects. It is related with a project (the commented one), a user (the author), a content (the comment itself) and a date (the time in which the comment was published).
2. **Contribution.** Represents a single contribution that can be done by one user to a project. A contribution is done to a project by a user in a determined datetime, and the user contributes with a certain amount of credits.
3. **Development.** Represents a development that can be done by one user. The user submits a new development to a project in a certain date and links the git directory where the branch of the project can be found. It also has a vote counter so that the contributors can rate the developments.
4. **Message.** This entity represents a private message that is sent from one user to another. A user is able to send a message to another user with a particular subject and content in a concrete datetime.
5. **News.** This entity represents small pieces of text which will provide information to the users in a similar way as a news feed does.
6. **Project.** This is the main entity of our application, representing a project which has a creator, a description and a total bank of accumulated credits among other properties. Basically, all other entities are related with the projects.
7. **User.** This is the other main entity of the application, when someone uses our application he or she is represented by an user in our community which has a name, surname, a nickname for the application and also a mail address to maintain contact.

All of those entities have a related business entity which represents the entity object in our application and a data access component which provides database functionality to the entity and acts as a layer which separates the database and the code in our layered model.

4.2.1 Business Entities

To represent the previous entities we created another six different business entities which make use of the corresponding data access components to provide CRUD operations to the user of the entity. Further details can be checked in the code documentation.

The six business entities are: **CommentBE**, **ContributionBE**, **DevelopmentBE**, **MessageBE**, **NewsBE**, **ProjectBE** and **UserBE**.

4.2.2 Data Access Components

Alongside the business entities we have the other layer of the model: the data access components of the entities. They serve as an intermediate layer between the entity and the database. They connect with the database and query it to obtain registers, modify, delete or insert new ones. More information and internal functioning can be found in our code documentation.

The six data access components are: **CommentDAC**, **ContributionDAC**, **DevelopmentDAC**, **MessageDAC**, **NewsDAC**, **ProjectDAC** and **UserDAC**.

4.3 ASP Application

The other solution of our project is the ASP Web Application itself which makes uses of the previous class library. In this subsection we are not going to discuss the internal design principles of our application (like the controls used, how to use them; which auxiliary methods are we using and what are we doing in the page load...etc) we are going to provide a general overview of the solution layout so that anyone can explore the source code in a guided manner without getting lost. We will expose the web forms organization of the web application, the different master pages that we have used and how we dealt with session management.

4.3.1 Web Forms

The ASP application is composed of many web forms that compose the whole website. The web forms are grouped in different sections. We have x different sections in our web application that can be accessed through the interface buttons, links and menus:

1. **Home.** This is the main page of the application, it is just a welcome screen where some information is shown. The web form of this section is *Default.aspx*.
2. **Projects.** In this section we show the project system: users can perform searchs, visualize projects and create new ones. The web forms associated to this sections are *ProjectProfile.aspx*, *Projects.aspx* and *CreateProject.aspx*.
3. **Profile.** In this section we show the user information: a resume of the profile, the message inbox and message creation subsections, the developments and contributions and also the user has the possibility to get more credits. The web forms of this section are *ProfileCompose.aspx*, *ProfileContributions.aspx*, *ProfileCredits.aspx*, *ProfileDevelopments.aspx*, *ProfileMessages.aspx*, *ProfileMy.aspx*, *Profiles.aspx* and *Message.aspx*.
4. **Login.** This is a simple form created to allow the registered users to log in by introducing and authenticating their credentials. The web form of this section is *Login.aspx*.
5. **Sign Up.** This is a form we created for registering a new user, it only shows a registration form and validates the input to create the new user. The web form of this section is *SignUp.aspx*.
6. **Contact.** This is a simple form which only shows contact information about the group. The web form of this section is *Contact.aspx*.
7. **About.** This section shows information about the project just to inform new users about the philosophy of the project. The web form used by this section is *About.aspx*.

4.3.2 Master Pages

We are using a master page for the whole project (*Site.master*) and some nested master pages for the different sections that can be navigated using the menu items: the project section uses the *Project.master* nested master page and the profile section uses the *Profile.master* one.

4.3.3 Session Management

In order to handle session management we first considered the *Session* object option, storing in an associative array the data of the authenticated user and checking in each private page for the proper session variable to be set. After some problems, we changed our code to use *Cookies* instead of using directly the session object. Check the *UserDAC* and *Login.aspx.cs* for details.

4.4 Interface and Design

In order to design the whole interface we have used Adobe Photoshop for image editing and CSS3 to apply different styles to the web pages. The CSS stylesheet used by all the sections of the project is located on *Styles/Site.css*. The names are quite self-explanatory and the CSS properties are simple enough to know what each one is doing to a determined section of the page.

The interface is a combination of ASP controls, HTML properties and images whose styles are determined by the CSS rules to make the website look beautiful and stylish.

4.5 Database

The other critical component of our project is the database which will hold all the data related to our application (users, projects, comments, messages and so on). The database has suffered a lot of changes from the initial schema that we released in the first delivery so we are going to show the logical model of the newest version of the database.

The SQL sentences for creating the database can be consulted in the *db.txt* file in the root folder of the project. That file contains all the needed queries to create the whole database (or drop and recreate it) with full keys definitions and referential integrity checking.

It is also important to remark that all the data types are adjusted to follow the SQL Server ones. We have used the SQL Server Management Tools to connect with our database and execute creation sentences (the server explorer that Visual Studio provides is quite limited and it only allows you to perform queries with SELECT).

As we said, this is the logical model of our database (there is a table for each entity of our project

```
users (
name VARCHAR(32) NOT NULL,
last_name VARCHAR(32) NOT NULL,
email VARCHAR(64),
nickname VARCHAR(32) NOT NULL,
password VARCHAR(70) NOT NULL,
credit INT NOT NULL)
PK(email)
ALTK(nickname)

projects (
code INT IDENTITY,
title VARCHAR(64) NOT NULL,
description VARCHAR(1000) NOT NULL,
deadline VARCHAR(11) NOT NULL,
creation_date VARCHAR(11) NOT NULL,
state INT NOT NULL,
total_bank INT NOT NULL,
last_partition VARCHAR(11) NOT NULL,
partition_bank INT NOT NULL,
gitdir VARCHAR(128) NOT NULL,
creator VARCHAR(64) NOT NULL)
PK(code)
FK (creator) -> users
```

```
message (
convers_code INT NOT NULL,
code INT,
date VARCHAR(20) NOT NULL,
issue VARCHAR(32) NOT NULL,
body VARCHAR(1000) NOT NULL,
isread INT NOT NULL,
deleted_sender INT NOT NULL,
deleted_reader INT NOT NULL,
sender VARCHAR(64) NOT NULL,
addressee VARCHAR(64) NOT NULL)
PK (code)
FK (sender) -> users
FK (addressee) -> users

news (
code INT,
title VARCHAR(32) NOT NULL,
content VARCHAR(256) NOT NULL,
public_date VARCHAR(20) NOT NULL,
creator VARCHAR(64))
PK (code)
FK (creator) -> users

contributions (
project INT,
usr VARCHAR(64),
date VARCHAR(20),
amount INT NOT NULL)
PK(project, usr, date)
FK(project) -> projects
FK(usr) -> users

developments (
project INT,
usr VARCHAR(64),
date VARCHAR(20),
gitbranch VARCHAR(128) NOT NULL,
ups INT NOT NULL)
PK(project, usr, date)
FK (project) -> projects
FK (usr) -> users

comments (
project INT,
usr VARCHAR(64),
date VARCHAR(20),
comment VARCHAR(140) NOT NULL)
PK (project, usr, date)
FK (project) -> projects
FK (usr) -> users
```

5 User Manual

Throughout this section we are going to teach a new user how to use our application: what he/she can do, how to do it and what are the implications of his/her actions. In order to do that we will advance section by section showing some screenshots of our project and explaining the options that the user is able to perform in that determined page.

5.1 Welcome Screen

The first screen that a user is going to find when he/she enters in our application will be the welcome screen or "Home page". In this page you can see a brief introduction to the ShodeProject and some news about the site. What's more it provides all the functionality that the user needs to navigate through the different pages, log in, log out, sign up and so on.

Next to that you can see the web page statistics. You can check at any moment the amount of Shode Credits that have been used or are being in use on projects. Any time a new project is proposed or someone contributes this counter will increase. In addition, we show you the amount of projects that have been or are being developed in the community, as well as the amount of contributions to them.

5.2 Non-Registered User Functionalities

The first time a user makes use of our application he/she will not be registered in the site. In this sense a limited subset of tasks can be performed using this non-registered profile: you can sign up to join the community by clicking the *Sign Up* link on the upper left part, you can also *Log In* using the proper link if you already have a user or you can just navigate through the different pages using the navigation menu.

You will be able to explore projects, fill the contact form, read more about the project or see the news. However, you won't be able to access your profile if you are not logged in and you can't also contribute to projects or develop. In addition, you are not allowed to comments projects, although you will not see the options.

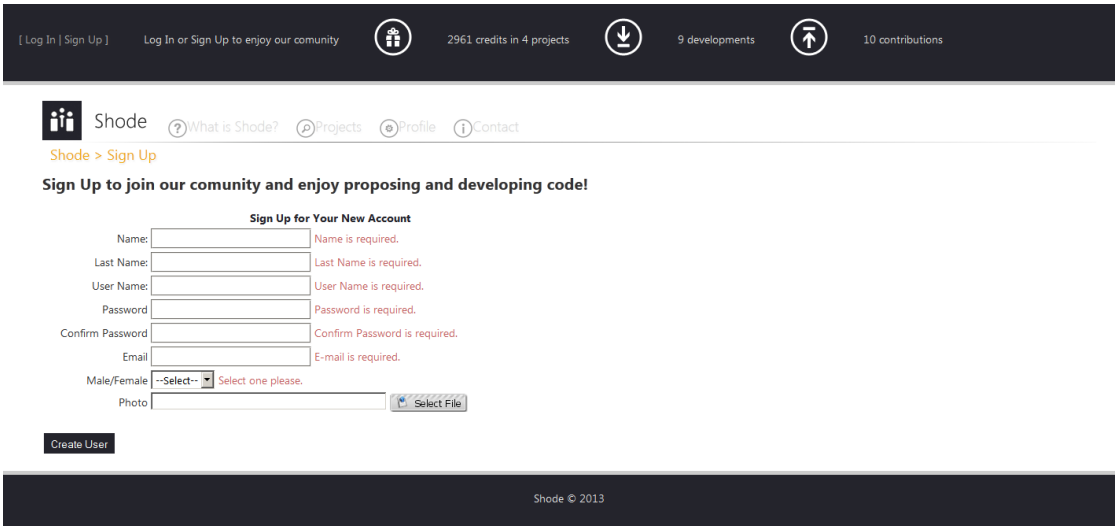
5.3 Creating a new User Account

Getting an account is quite easy. Wherever you are, you will have the *Sign Up* link in the upper left part of the site which will bring you to the sign up form. You have to complete all the fields (except for the profile image or avatar) in order to get a new user. Take into account that the username cannot be repeated (as well as the user email), the website will warn you if that situation occurs.

The system will calculate the strength of your password as you introduce it. We recommend the highest security level possible. However, we never get to know the passwords and we will encrypt them with complex mechanisms so no one can obtain it, even if someone compromises our database.

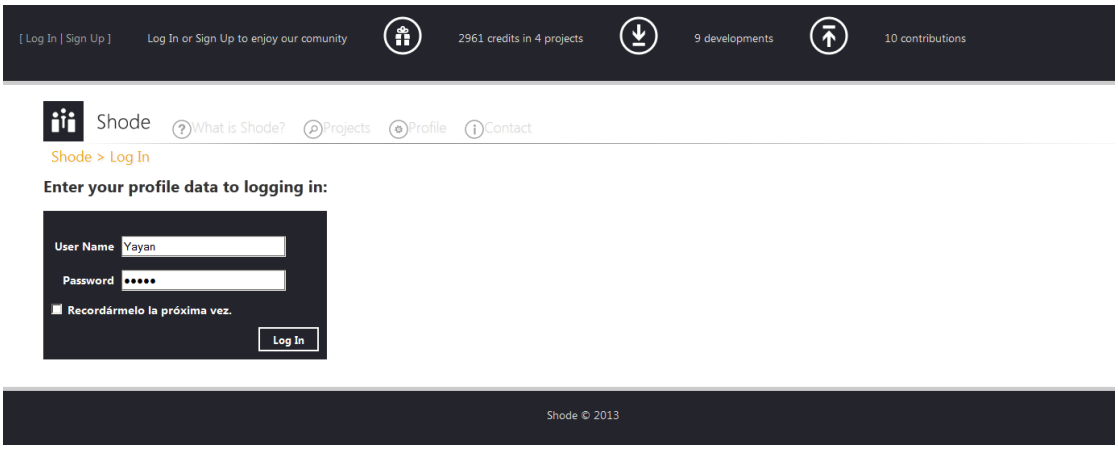
You can also upload an image for your profile (we recommend uploading a JPG file or maybe the image will not be shown properly), but this step is optional.

Once you have completed the form, click the create user button. If everything went right, a message will appear below the form telling you that the user has been successfully created so that you can proceed to *Log In*. If something goes wrong, you may check the error messages and correct the needed things.



5.4 Enter the Community

If you are here, we suppose that you have already registered and you have a user account in our application. Now it is time to unlock the power of Shode by entering the community. It is as simple as clicking the upper left part link *Log In* which will redirect you to the log in form.



You only have to provide your username and your password to proceed. If everything went OK you will be redirected to the main page and the upper left links of the page (which led you to the sign up and log in forms) now are gone and a welcome message appears aside a *Log Out* option to end up your user session.

Please, this site uses cookies to handle the user sessions and log in mechanism so it is highly recommended to activate them in your favourite web browser, otherwise you won't be able to enter the community.

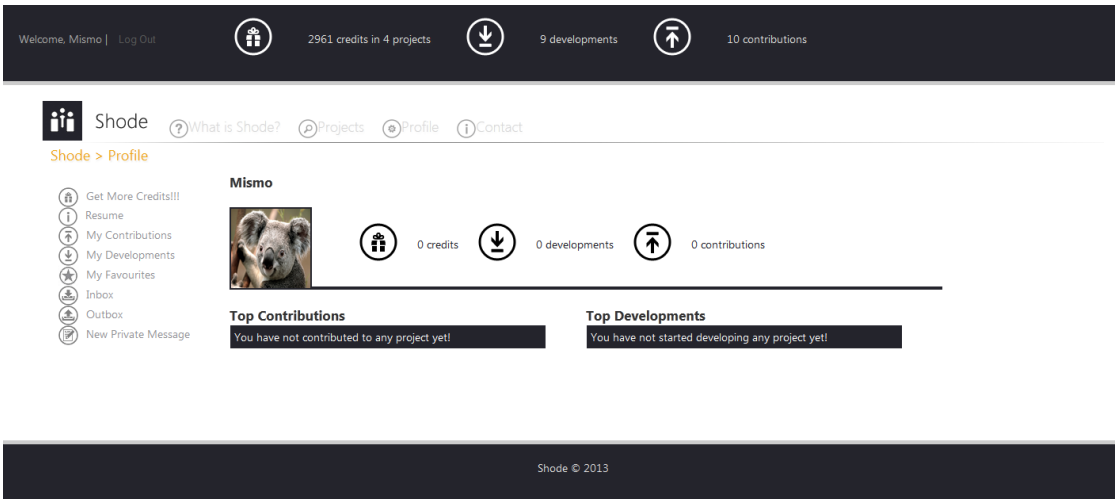
5.5 Your Profile

Use the Profile option at the top menu so you can access to your private place. You will be redirected to the welcoming page at the profile where all the possibilities of this menu option are

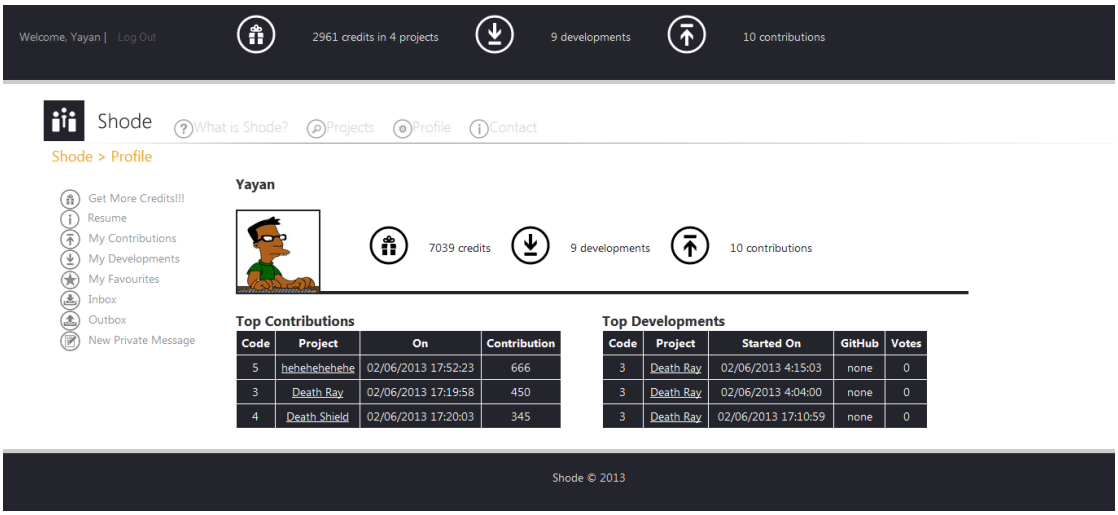
explained: check your status, get more credits, send private messages, and many more options we are going to explain. There will be always a menu at the left so you can navigate through the Profile easily. The top "Shode >" label will be updated so you can check where you are at any moment.

5.5.1 Resume

This is the main page of your profile. At the Resume page you will see your avatar in black borders. If you have no picture profile a black square will be shown. Next to it, your statistics are shown: the amount of Shode Credits you now, how many developments you form part of and also the amount of projects you have contributed to. Those statistics will be changing as you get more involved in the community.

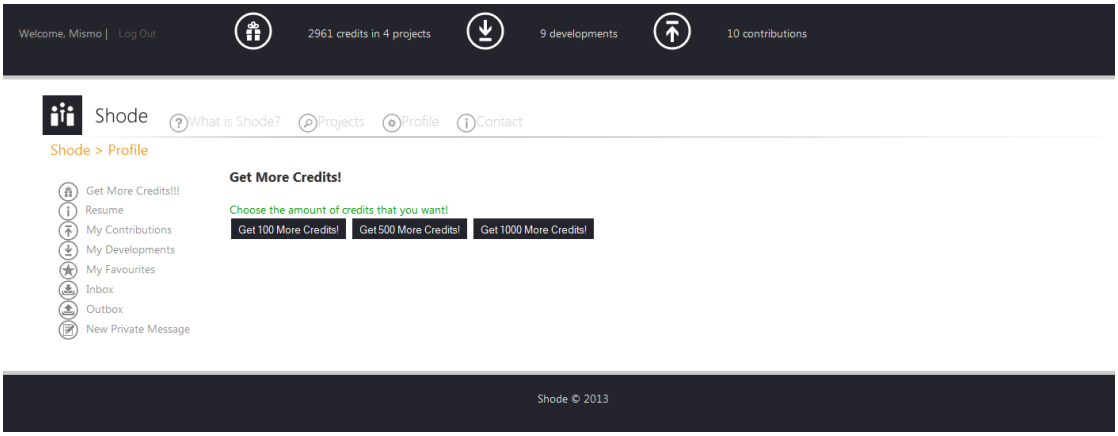


Down below that the top lists are placed. At the left you will se your top three contributions: the three biggest amount of credits you have given to a project. And at the right, the top three developments appear. Those are the developments you have take part in and the users have highly voted.



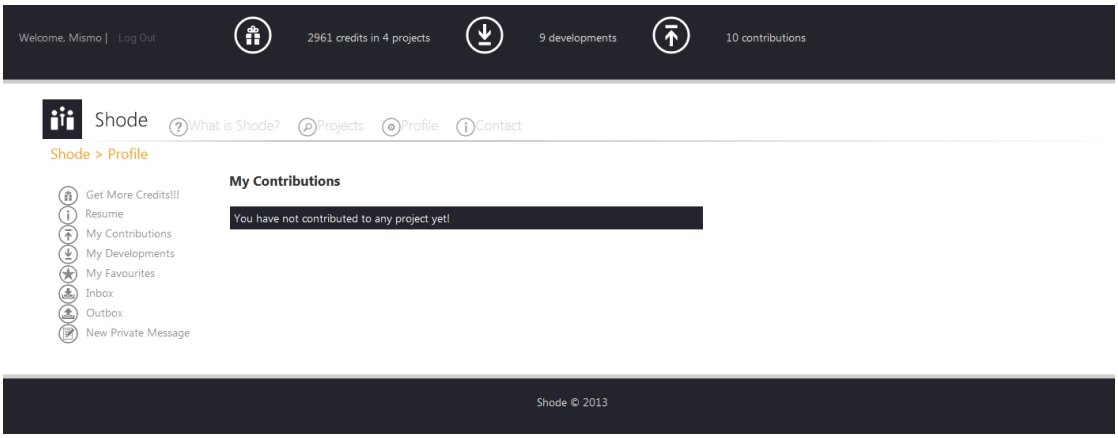
5.5.2 Getting Credits

Anything on this community would work without Shode Credits. Credits are the coin used at Shode so you can begging a new project or collaborate to one. You can get more Credits using this page. We want to integrate this service using PayPal but they are for free right now so take advantage of this situation and get as many as you can!

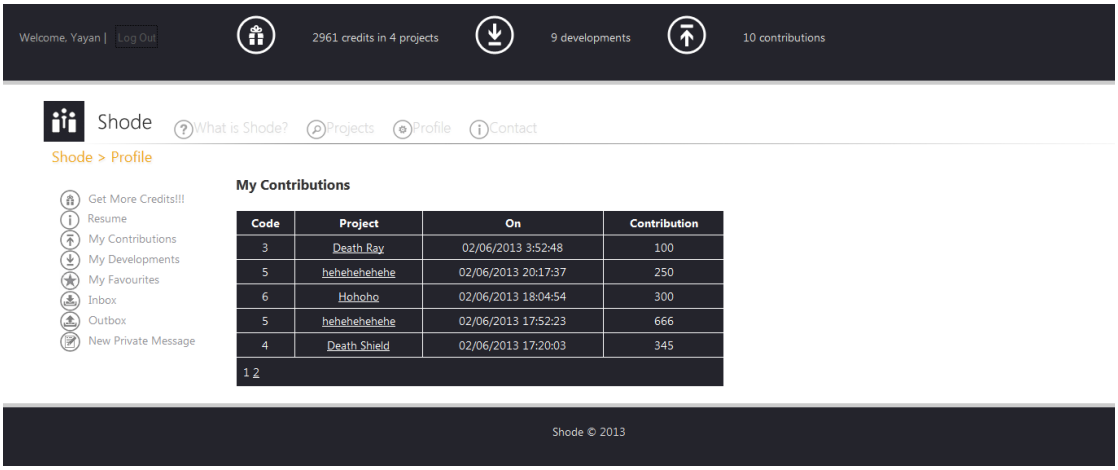


5.5.3 My Developments and Contributions

These two pages are very similar. They both show you a list of projects. However, at the Contributions page you are seeing the projects you have contributed to ordered by date (the most recent goes first). On the other hand, at the Developments page you will see the projects you are developing or you have developed. Those projects will be ordered by date also. However, if you have not contributed or developed anything a message will be shown informing about this situation.



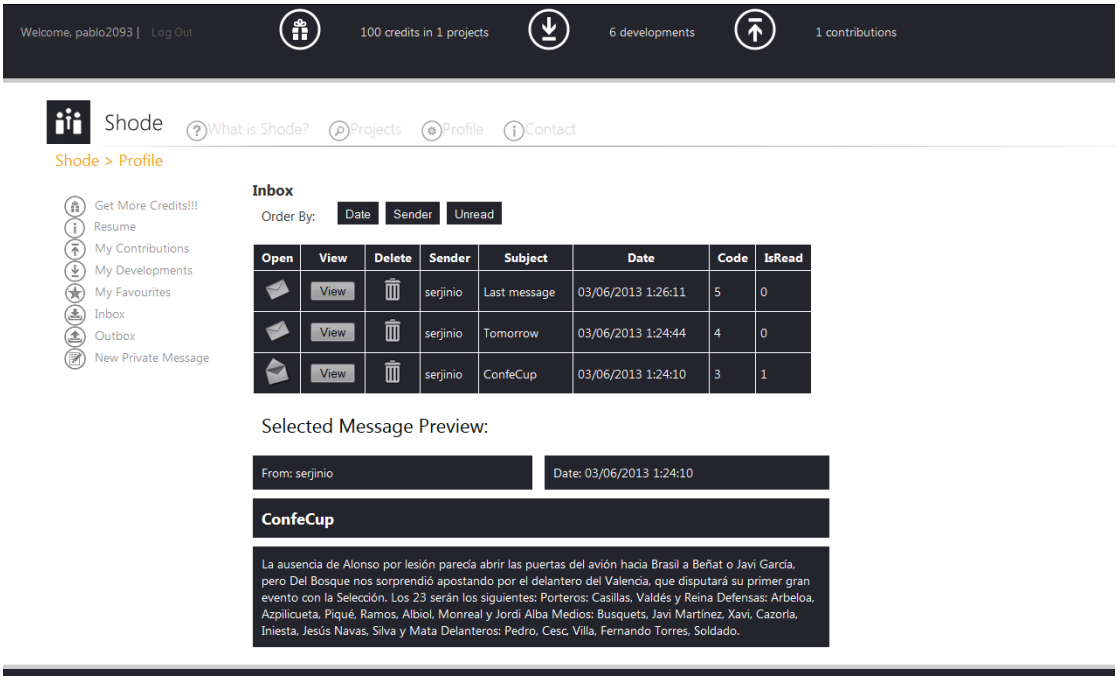
This page is quite useful and we are sure you will check it frequently as you want to track the state of those projects you have show some interest. This page completes that necessity because those list will have a link to the projects profile.



5.5.4 Communicating with other Users

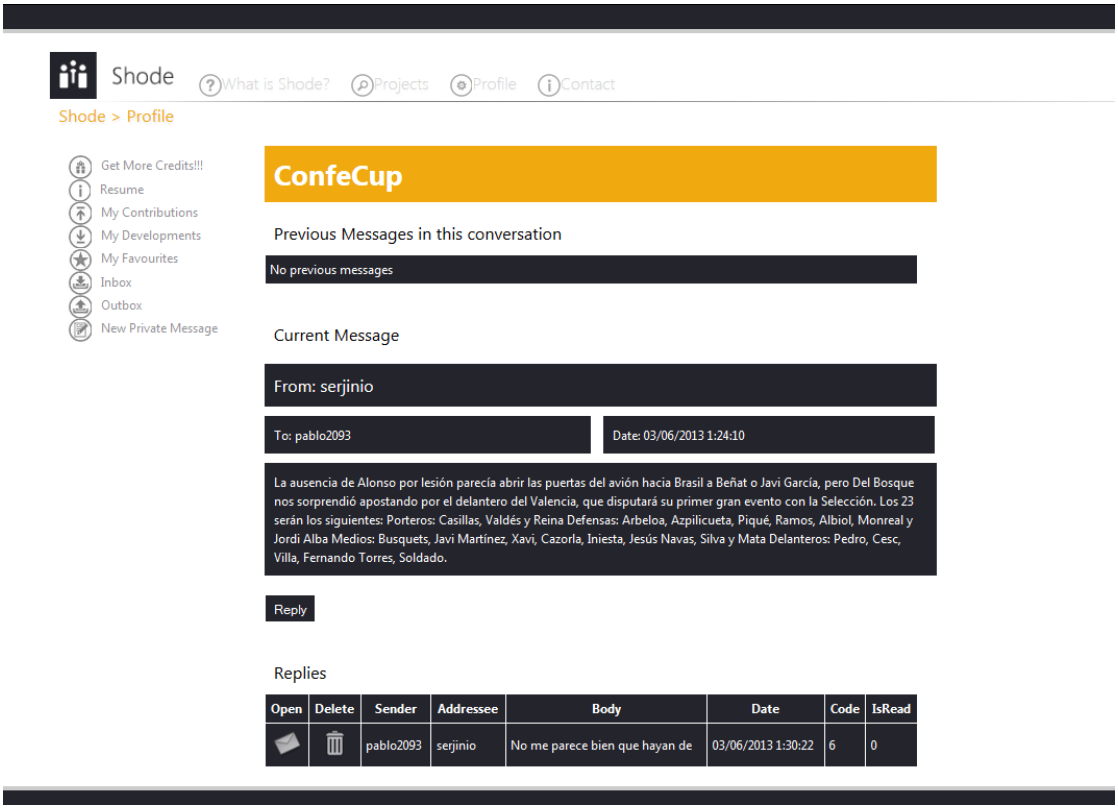
What is a community where no one can contact the others? That is not a community. That is the main reason we decided to include a message system. At this profile page you will have three different pages created with this porpoise.

The first one is the Inbox page. Here you can check your messages inbox. Every time a user writes something to you, it will be shown here as Not Read. There will be a closed envelope image indicating this state. You have two options: previewing the message or opening it completely. The preview allows you to read the message but not previous messages from the same conversation.



However, opening the message will display the previous and following messages. In addition, you will be able to write a reply to the message that will be added to the same conversation.

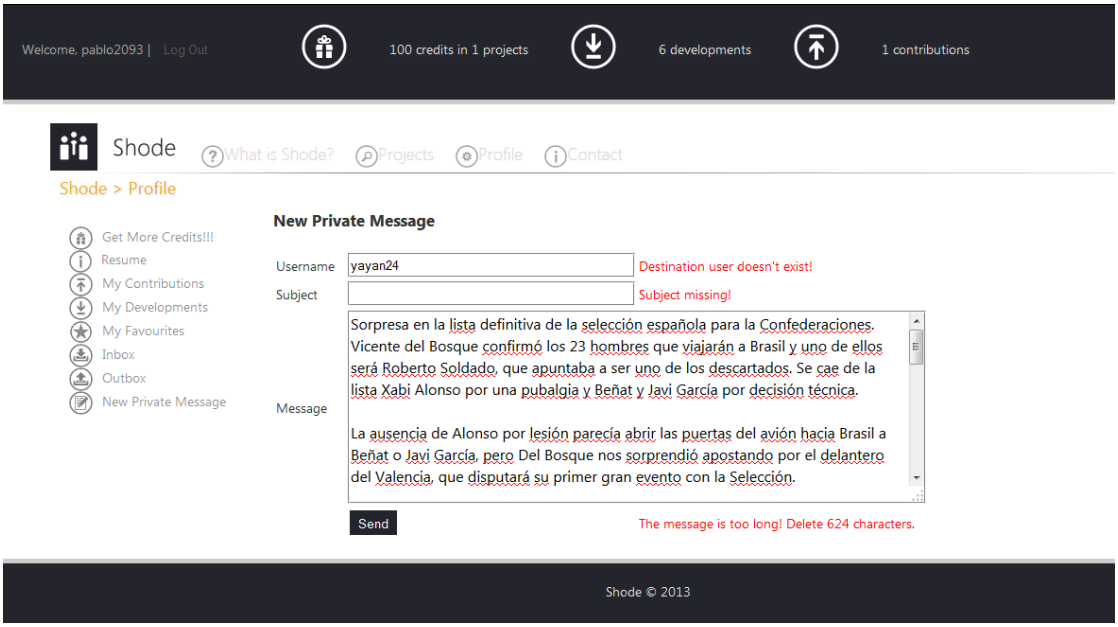
Finally, you will be able to delete messages. Once we press the trash button, that message will not be shown again in our inbox list. Nevertheless, the message will not be removed from database until both sender and addressee had deleted it from their outbox/inbox lists respectively.



The second one is related to this first page: it is the Outbox page. Here you can check your sent messages and it works completely equal to the previous page. You can preview messages or open them, delete them and see the full conversation.

In both inbox and outbox pages, we will be able to sort messages by different attributes. By default, they will be sorted in descendant order by date (first the latest ones), but using the buttons above the list we will be able to order them by date in ascendant order, group them by sender/addressee or show first unread messages.

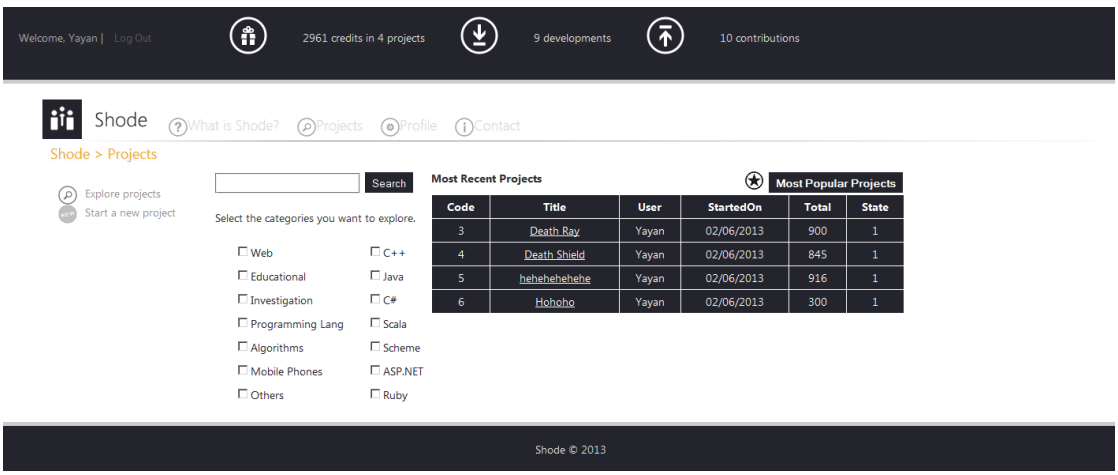
The last message related page is, of course, the one that allows us write messages and send them. For this purpose, you must give the addressee user's nickname. If it does not exists, the system will warn you. In addition you must give it a subject and obviously write the message. Database has a limit of 1000 characters, so, if you write a message longer than that and try to send it, system will warn you and will indicate how many characters should be deleted. If everything is OK, when you press the Send button, the system will redirect you to Outbox message list.



5.5.5 Exploring the Projects

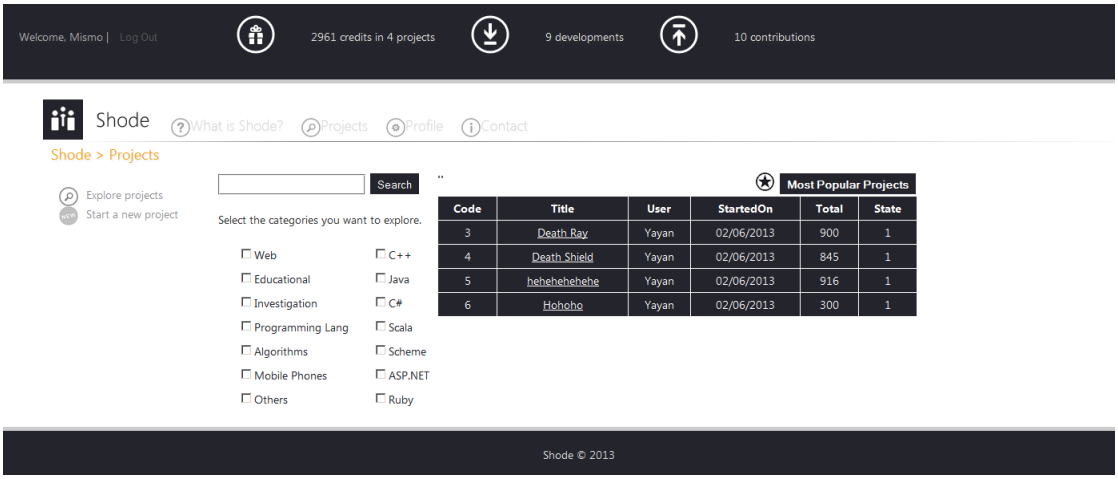
Exploring projects is that main and default project webpage. You can access to this page by clicking to the Projects tab at the main menu and at Explore Projects at the projects menu placed at the left of the screen. This menu will be there at any project related page.

At exploring projects the first thing you can notice is a list of projects at the right in a black-cell table. Those projects are the most recent created projects. These are projects created at the very same day you entered this exploring projects page for the first time. Why do we use the first time you entered day? Because you could be browsing those projects someday at 23:59 and we do not want you to lose your exploring history when the clock hits midnight and the day changes.

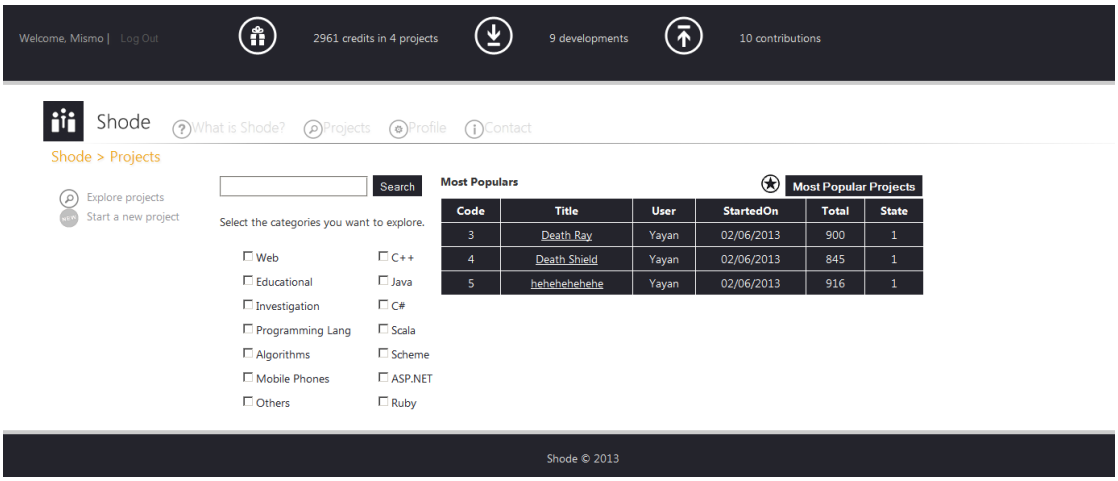


In addition, there is a searching box so you can look for anything. You just have to entered a keyword or a sentence and the projects whose title has the very same searching text will be shown.

That is why we recommend searching using a keyword. For example, if you are interested in some musical applications do not try searching *"Musical applications"*. Instead, use the keyword *"Music"*. It will produce a more effective search. There is also a hidden functionality: if you begging a search with an empty text you will get the whole projects data base so you can check every project that have been created at Shode.



At the top left corner you can see a star and the most popular button. You can see the most popular projects clicking on it. The result grid will change and will show those projects whose amount of credits are the greatest.



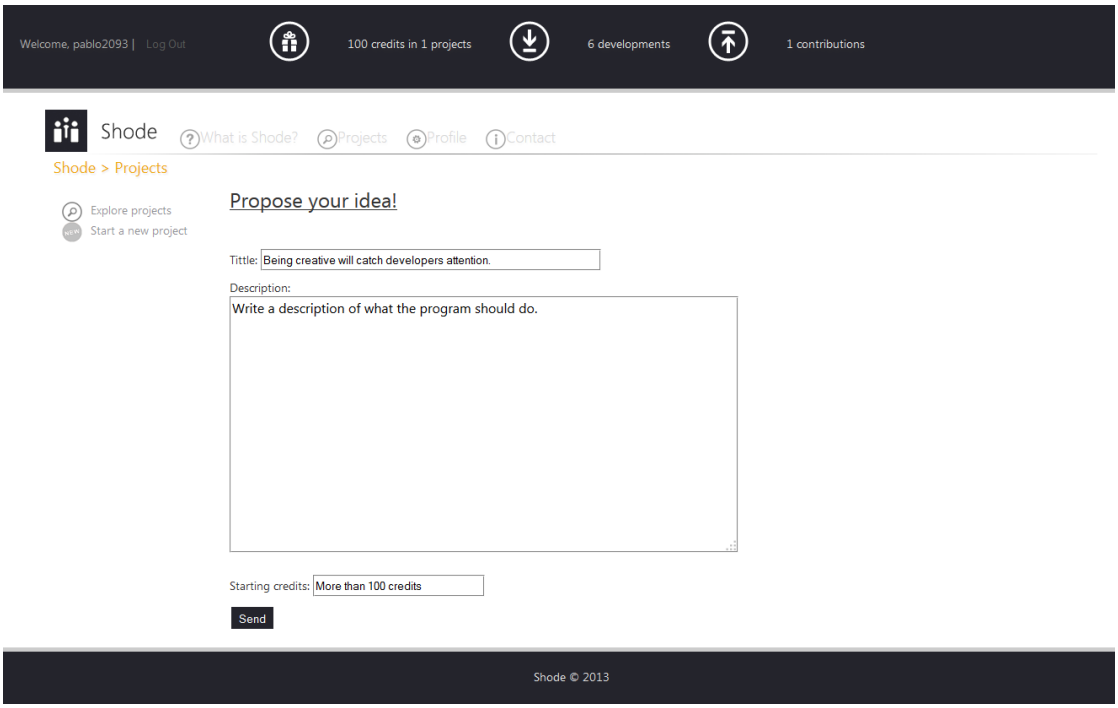
5.5.6 Starting a new Project

At the left menu you can see the create a new project option. Clicking on it will redirect you to this page. At this page you are able to propose new projects. But, if you are not logged in you will be redirected to the logging page.

Basically, you must give it a title and as the default text indicates: try to be creative. Choosing a good name is very important for the future of the project. It should be clear, principally. In addition you must give a description. Do not try to re-write the holy bible: you are allowed to write 1000 characters. Those are a enough!

At the bottom you have the starting credits field. You cannot propose a project without giving it a first contribution. The minimum is set to 100 Shode Credits and the maximum depends on your credit, of course. If you try to give a number out of the range [100 - your credit], you will be warned by the system.

Below this box you have the creating button. We will check that everything has been filled correctly and the system will give you a message. If everything went OK, you will have proposed your idea.



5.5.7 Project Profile

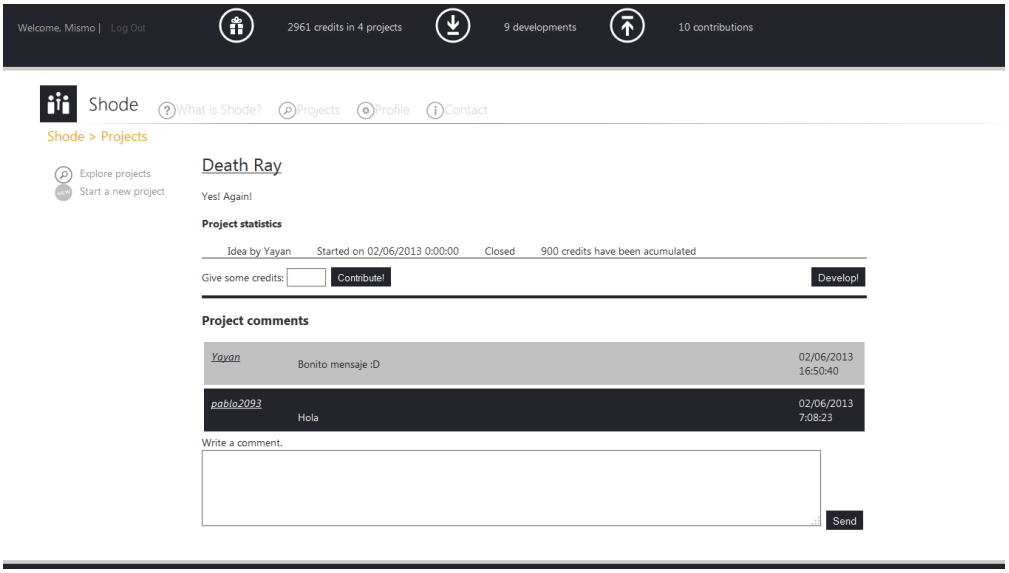
As we have said, you can explore projects but how do you access to them? At the explore projects section, the results have a link in the project title which will drive you to their project profile. At this project profile, you can consult several information related to them.

At the top you will see the project's title and description given by the creator user. Below that, we present you the project statistics: the creator user, when did the project began, its state (presented, developing, closed...) and also the credits accumulated.

Now, if you are a registered user who has logged into the system you are able to see the Contribution and Develop options. Those options are hidden to the general public and are only shown to you: Shode registered users. We will come back to these options later.

In the middle of this page there is a black line. It divides the project information from the commentaries. At every project profile you are able to read the commentaries other users have done about the project. In addition, you can write your own comment but, again, only if you are logged.

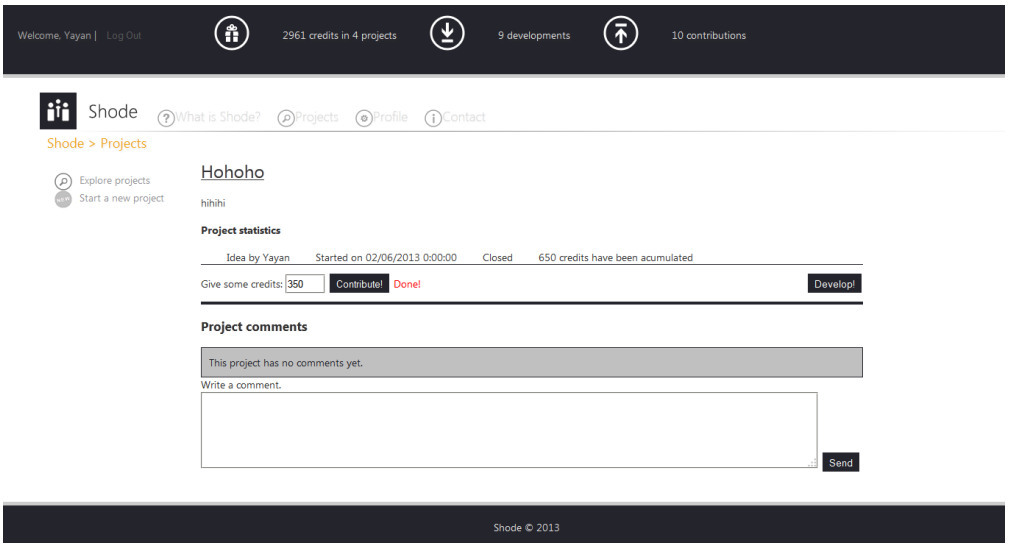
Writing a comment is very easy: just type in the text box what you want to say and click on the button at the left. If the message format is correct you will see you uploaded comment automatically!



5.5.8 Contributing to a Project

So, what if I want to contribute to a project. Being logged to the system, you will see the text box and the button *Contribute!*. Just write in the text box the amount of credits you want to give to the project and that is all. We will check, as previously done, that you are giving a correct amount in the range of 100 and your credit limit.

It is very important contributing to projects because developers are willing to get credits by developing projects. If you are interested in the project, express your interest contributing. In addition, every contributing increases the general counter and your personal counter is also increased. The contribution will be shown in the contributions profile option we have talked about before.



5.5.9 Developing a Project

Finally, we get to the develop area. As a developer you are interested in this part.

Only when logged, you will see the develop button. Right now this button allows everybody to develop a project. It will give you a feedback text telling you that you have been accepted to develop the project and you can start whenever you want. Pressing this button will store in our data base your intention of developing the project and this will be shown in the statistics (global and personal).

However, this is not the principal idea we have. In the future we are planning to integrate this functionality with GitHub so every time a developer wants to develop a project it will be given a GitHub repository in order to get the last version of the project if it is still being developed. We will create the repo otherwise.

6 Final Planning and Performed Tasks

In this section we are going to present the final project distribution. It includes all the changes in the task partitioning that were made during the development of the project. In fact, it reflects the actual work done by each group component when the project was finished.

This distribution takes into account the whole project phases, from the very beginning when we created the class library to the final version that will be delivered on Monday 1st of June.

The initials of the table stand for Class Library (**CL**), Design and Interface (**DS**), Sections (**S**), ASP code and Database management (**ASP/DB**), Documentation (**DOC**) and Other (**O**).

C	CL	DS	S	ASP/DB	DOC	O
Liesbeth	NewsBE and NewsDAC	News feed and control interface.	News	News home page, control panel for writing news and feed for the users	Inline code comments in ASP pages	-
Albert	ProjectBE, ContributionBE, DevelopmentBE, UserBE and ProjectDAC, ContributionDAC, DevelopmentDAC, UserDAC	General CSS Design of the whole application and User Interface	Database Management, SQL. Code Cleanup and Normalization.	Database Design and Implementation. Business Entities and Data Access Components. Exception Management.	Full DACs and BEs comments, Second and Final Reports. User Manual redaction	Design Pattern (Singleton implementation) and Query Parameterization.
Pablo	MessageBE and MessageDAC	Message inbox and sending Interface. Log In form Interface	Messages	Messaging System and Sign Up form. Normalization of DB access from DAC's (GridViews)	First and Second Delivery Reports. User Manual redaction.	Presentation. Error Page.
Sergiu	-	Default Page Banner Design	Additional Technologies Research	AJAX Extensions and Password Security	First and Second Delivery Reports	Presentation. File Upload. Contact Mail.
Brayan	CommentBE and CommentDAC	Project search, visualization and creation Interface	Projects and User	Session control, user profile and project exploration interface (search, create, view, contribute and develop)	Inline code comments in ASP pages. User Manual redaction.	Nested Master Pages implementation. File Upload.

As we can observe, the equal distribution was impossible for personal causes. What's more, some components showed a great potential in one field so that they were reassigned to achieve better results.

In the end, due to the task distribution rearrangement the workload wasn't completely equal so that the estimated hours to complete all the tasks that we calculated in the initial planning ended up being false.

Using the GitHub commit history we calculated that the rest of the group components needed approximately 60 hours to complete all the assigned tasks.

7 Remarkable Troubleshooting

In this section we are going to discuss the most remarkable problems that we have faced during the project development. We are not going to consider common coding problems but hard situations where we were stuck for a long period of time until we brought a solution up. The majority of this problems were not "hard" at all but due to the lack of some subtle details they stalled our progress significantly.

7.1 SQL Manager Permissions

Since we could not perform SQL database creation queries using the server explorer that Visual Studio Express provides, we had to install and use the SQL Server Management Utilities to

perform complex queries like CREATE TABLE, DROP TABLE and also the creation of complex CONSTRAINTS like FOREIGN and multicolumn PRIMARY KEYS.

The SQL statements were quite simple but due to the Windows privilege manager we weren't able to load the database in the SQL Manager application. After trying and trying we discovered that we needed to install the SQL Management utilities directly in the HDD root directory (C:/), copy the Visual Studio project (or the database MDF and LDS files) in the same HDD root directory and also run the SQL Manager as administrator in Windows 8/7.

Although this problem may seem simple, we spent hours figuring out how to make it work and without the proper documentation of use in Windows 8/7 it was nearby impossible to discover the combination of facts that led us to the solution.

7.2 Using the ConfigurationManager

In order to store the connection string in a fixed location we used the *Web.config* file creating a new key *connectionString* to hold the string needed to connect with our database (located in the *APP_DATA* folder). We introduced in the code the needed *using* statements to get the *System.Data* and other namespaces in our data access components.

However, we weren't able to access to the *ConfigurationManager* class in our data access component classes; the compiler wasn't recognizing any reference to that class. We checked the slides and the online documentation of the *ConfigurationManager* class and we weren't able to solve this problem for a long period of time so we rolled back to defining the connection string inside each data access component class.

Fortunately, we discovered an MSDN article which mentioned that in order to make that class work you have to include in the *References* of the solution, a new reference to the *System.Configuration* namespace. As we said, a quite simple problem that had a huge impact on our progress due to a lack of documentation.

7.3 Dealing with DateTime

In our C# code we are dealing with dates using the DateTime data type (class), however in order to ease the connection with the database those dates are converted to strings with a determined format: dd/MM/yyyy for simple dates and dd/MM/yyyy hh:mm:ss for complex dates.

Both of them are stored in the database using VARCHAR type, the simple date will be stored in a VARCHAR(11) and the complex one in a VARCHAR(20). We will use the *DateTime.ParseExact* function to convert from string to DateTime and *DateTime.ToString(format)* to reverse the conversion from DateTime to string. The first one will be used to retrieve a date from the database and the second one to send a DateTime to the database.

We decided to use this implementation because we were having trouble when trying to insert a DateTime in C# to a datetime in the database using parameterized query due to the regional configuration differences.

8 Enhancements

As the different sections and components of the project were working properly we started to think about some improvements that were outside the original box of the development. In this section we are going to list them, both implemented and not implemented. What's more we will explain briefly what the implemented enhancements are.

8.1 Implemented

In this subsection we will explain the implemented enhancements and the reason for including them aside the task they are performing in our application.

8.1.1 Image Uploading

We have added image uploading functionality to the sign up form. Now the user is able to upload a single JPG image which will be his/her avatar in the profile. This image will only be uploaded if the user was successfully created, and it will be uploaded to a predetermined folder *Uploads* which will be used to read the profile images.

8.1.2 AJAX

We have configured our solution to use AJAX extensions. Since we started with a non-enabled AJAX template we had to configure the project manually to work with the ASP AJAX 2.0 Extensions. We have used AJAX extensions for two purposes: in the sign up form we check the password strength informing the user whether she/he should consider another password or not; what's more we have added a small Twitter integration in the home page to read the latests tweets of our project Twitter account. We are also using AJAX extensions to create placeholders for some text boxes (you can see it in the project creation interface).

8.1.3 Design Pattern: Singleton

We have used a simple design pattern, the singleton, to create a class that allows us to read statistical information from the database using all the available data access components. This class will be used to compute general statistics which appear in the upper part of the master page. The class name is *StatisticalSingleton*.

8.1.4 Password Security

We have added a new layer of password security by storing the hashed password with a determined salt in the database. The hashing methods are located in the user business entity which makes use of them. You can see the class which is fully documented to know more about internal hashing principles that we have used.

8.1.5 Parameterized Queries

To improve the security when dealing with database querying we have changed all the SQL commands to use the parameterized queries that ASP offers. You can see the data access components to know how we implemented the parameterized queries.

8.2 Future Enhancements

Because of the exaggerated difficulty of some operations we have left behind some enhancements that we would have wanted to include in the project. The main one was a full integration with GitHub using the Git API that the service provides.

Another interesting enhancement we want to include is a PayPal gateway so that the users can buy credits. In the current version of the project, in order not to deal with money, all the users can get credits for free by clicking the right option.

9 Conclusions

After describing our whole project, all its components, the situations we had to overcome and the numerous enhancements that we implemented to improve our learning and the project itself, it is time to look back and see compare our final result with the established objectives to see what we have achieved.

Regarding to the interface usability criteria, we have used validators in all the fields that require input from the user providing error messages when the conditions are not fulfilled. We think that the interface is quite user-friendly and intuitive (it was designed with that purpose on mind). We decided to take a minimalistic approach forcing the user to navigate in a depth (few options, many decisions) way instead of a breadth (lot of options, few decisions) which can be very confusing.

Moving to the interface design criteria we want to remark that we have used fully CSS styles, with consistent navigation with nested master pages (with a site master one and individual sections sub-master pages), complex controls (grid views, validators, buttons and much more) and extra features using AJAX extensions (password strength check, twitter integration and control placeholders).

About the data validation criteria, as we said we are using validators for all user input and after that input the data is sanitized using parameterized queries before producing an effective query to be inserted in the database. After the execution, all kind of exception is handled: the known errors return a string which yields information about the exception and the unknown (or not possibly expected ones) redirect to an error page which is reused.

Now let's look at the business entity layer: we have created a total of seven business entities (more than one per component) all of them with private fields and properties (and also extra methods apart from the CRUD ones to provide more functionality).

On the other hand we have the data access component layer: we have provided CRUD methods for all business entities and all external communication with the database is done through this layers. What's more all of them make a good work handling the possible exceptions as we remarked previously. In addition we have used both connected and disconnected access depending on the situation (connected for normal queries and disconnected to provide source data to grid views). We also included a singleton design pattern compliant class (see the enhancements).

All the business entities and data access components are fully commented following the ASP/C# rules providing complete summaries and useful information. What's more we have included inline comments in some ASP pages which clarify complex functions and situations.

We have also implemented a good number of enhancements as we previously discussed in the corresponding section. Those enhancements implied a deep learning of some unknown concepts which ended up improving the project (password security, AJAX integration and file uploading).

It is important to remark that in order to synchronize our work we have been using a version control system (GIT) with a remote repository in the GITHub platform (check it here http://github.com/Blitzman/ProjectShode).

In conclusion, we think that we have met all the proposed objectives going further in certain aspects and staying inside the box in some others. In our opinion, we have developed a solid and technologically interesting project which makes use of all the contents that we have learned through the theoretical sessions.

To sum up, if we could choose a sentence to resume what we have done this would be it: ***"If anything deserves a reward, it is social contribution. Creativity can be a social contribution, but only in so far as society is free to use the results. If programmers deserve to be rewarded for creating innovative programs, by the same token they deserve to be punished if they restrict the use of these programs."*** *Richard M. Stallman.*