Títol: **Web i Model de negoci d'una companyia social.**

Socialpackers.com

Autor: Xavier Cases Camats

Data: 20 de juny de 2013

Director: Miquel Barceló Garcia

Departament del director: Departament Enginyeria de Serveis i Sistemes d'Informació

Titulació: Enginyeria Informàtica Tècnica de Gestió

Centre: Facultat d'Informàtica de Barcelona (FIB)

Universitat: Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

i. Table of Contents

0. Preamble

The document contains the Technical Development of **Socialpackers.com**, a website of a social business that will be launched in 2013 by two FIB students, Albert Vellvé i Olivares and Xavier Cases Camats.

The following pages -in conjunction with the project lead by Albert Vellvé i Olivares, which will contain the business plan of the webpage-, hold the final thesis of both students, worth 22.5 credits each, that matches to the Technical Engineering in Computer Management requirements' policy. Therefore, the reader is requested to read in parallel the final project of Albert Vellvé i Olivares in order to have a holistic overview of the business.

As it will be stated in detail on the financial part of the project, we are meant to put our effort so we can make this project sustainable in the long term. Due to this, we have studied all the possible revenue streams that will have an impact on the business. But for now, let us do a quick calculation of the investment that both, Xavier and I have done so far on **Socialpackers.com**:

Hours Xavier		576
Hours Albert		550
Total Hours		**1126**
Cost of hour	€	15
Total cost Hours	**€**	**16,890**
Domains	€	20
Server (1 year)	€	35
Total Investment	**€**	**16,945**

1. Introduction

SocialPackers.com is a free web application that provides lots of social projects around the world. It helps people to find social projects in the destiny of their trips and get a reward for it, helping to along the trip.

1.1. What is SocialPackers.com?

For a user, SocialPackers.com is a social network between travellers and entrepreneurs, a social network were the main goal is share vital experiences with local people wherever they are.

1.2. Objectives

Often the people that travel around want to be in touch with others that experienced something similar, they do not know how to do something in that country or what do they need to cross frontiers, or which are the most meaningful sites in that other place that books and agencies did not mention.

Often entrepreneurs want to do something that they are not capable of, because lack of resources, because lack of knowledge, and they are willing to give food, accommodation or other things in exchange for help.

With this project we intend to connect these two kinds of people to help each other, let they interact, comment and share projects, experiences, tips and so on.

1.3. Main goal

We want to help travellers on their day by day, putting them in touch with other travellers and with people searching for volunteers and mixing individuals with common interests. SocialPackers.com will also be a place to share acquired knowledge, tips and tricks, curiosities, etc.

We offer a daily tracking tool for users and social organizations where they can check not only the tasks accomplished in the past, but also plan the tasks they want to accomplish in the future.

Don't forget the main goal of SocialPackers.com. How do we put in touch people from here and there?

First, 'here' is where you, *the social traveller*, are and 'there' is where they, *the people with a social project*, are. Or not. Maybe 'here' is where you, *a social organization*, are and 'there' is where they, *the social travellers*, are.

The main technical goal of this application is to make easy the publishing of a social project online and to make easy the process of a user to find and subscribe to it.

2. Planning

Here we will show you the initial planning for the project.

2.1. Phases

We separated the planning in four phases. If we look at the Gantt diagram, we can see that some tasks overlaps others, in this planning we take into account different roles as could be Analyst, Developer, Manager and Architect although only one resource is available for all tasks.

2.1.1. Definition Phase

December 3th – February 3th Project Definition

Here we decided the main goals of the web application; we did the planning and decided the end date.

January 1st – February 28th Technology Search /Learning Curve

Find the best matching technologies for our purpose. Find and read some documentation and learn a bit about the technologies chosen.

2.1.2. Design Phase

February 4th – March 1st Web design and Database model

Decide the entities stored in the database model and the look and feel of the web application.

2.1.3. Development Phase

February 12th – April 23th Development

Create the needed entities and make it interact. Create all decided functionalities by iterations.

April 24th – May 23th Testing

Create and execute some test to validate the main functionalities of the web application.

2.1.4. Documentation Phase

May 24th – June 8th Documenting

Create and complete the documentation. Prepare for the presentation of the web application.

2.2. Time estimation

- Definition Phase

Project Definition	45days * 3hours = 135h
Technology Search /Learning Curve	43days * 3 hours = 129h

- Design Phase

Web design and Database model	20days * 3 hours = 60h

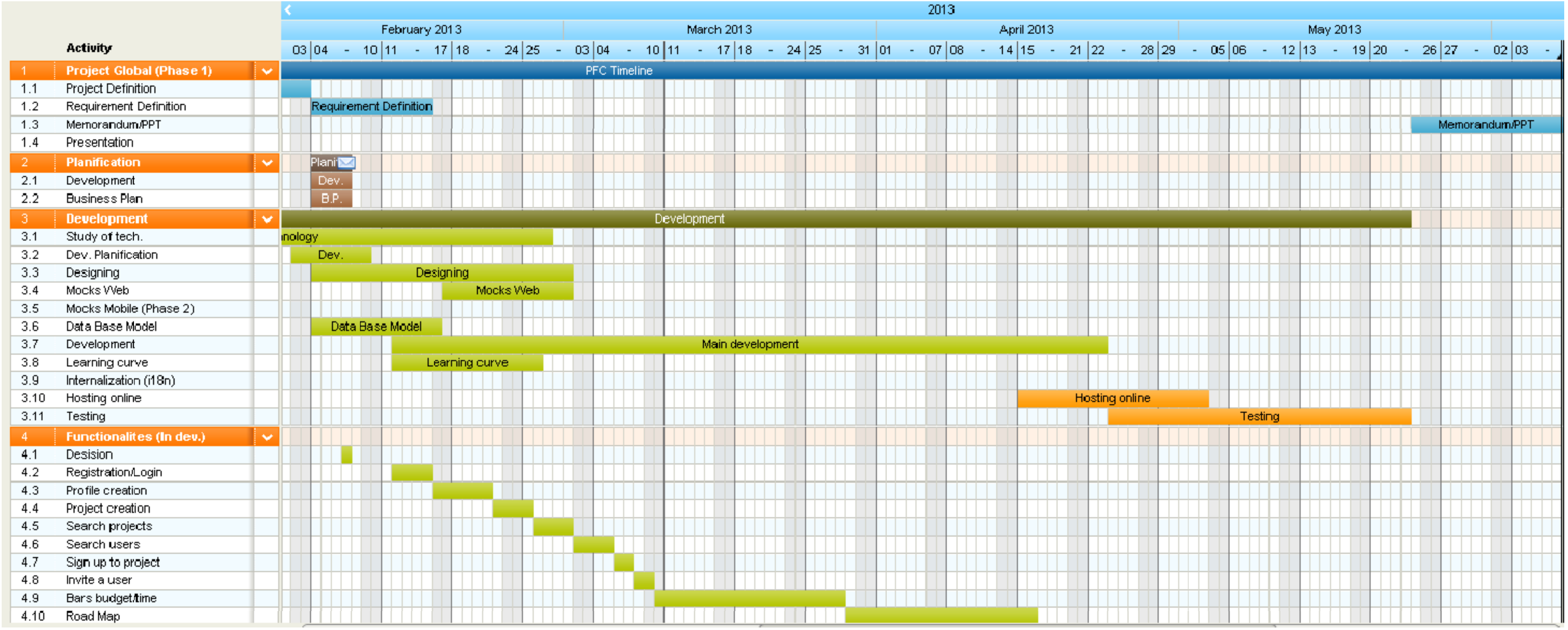
- Development Phase

Development	51days * 3 hours = 153 h
Testing	22days * 3 hours = 66h

- Documenting Phase

Documenting	11 days * 3 hours = 33h

In total we have 576 hours of work. See the following Gantt diagram for a detailed view.



	Activity
1	**Project Global (Phase 1)**
1.1	Project Definition
1.2	Requirement Definition
1.3	Memorandum/PPT
1.4	Presentation
2	**Planification**
2.1	Development
2.2	Business Plan
3	**Development**
3.1	Study of tech.
3.2	Dev. Planification
3.3	Designing
3.4	Mocks Web
3.5	Mocks Mobile (Phase 2)
3.6	Data Base Model
3.7	Development
3.8	Learning curve
3.9	Internalization (i18n)
3.10	Hosting online
3.11	Testing
4	**Functionalites (In dev.)**
4.1	Desision
4.2	Registration/Login
4.3	Profile creation
4.4	Project creation
4.5	Search projects
4.6	Search users
4.7	Sign up to project
4.8	Invite a user
4.9	Bars budget/time
4.10	Road Map

2.3. Risk analysis

2.3.1. Planning times inaccurate

It is possible that in the development phase the tempos will be inaccurate. The developer is learning a new programming language, different from any other used before in the career or work.

Probability: High

Impact: Low if it is detected soon, high otherwise

Solution: Add resources, cut the scope of the project. Do a new planning.

2.3.2. External causes

Because the need of spare more time at work, because an illness

Probability: Low

Impact: Low

Solution: Plan in short term. For long periods, cut the scope of the project and do a new planning.

2.3.3. Low quality

The website does not comply with specifications.

Probability: Very low

Impact: Critical

Solution: Redo the specification and the design. Check for deviations and do a new planning if required.

3. Requirements analysis

3.1. Interested parts

Travellers. Users want to be capable of find projects of their interest, and contact to the owner to get more information about. The users want to share and be able to search on reliable site the information related to the trip they are doing.

Entrepreneurs. Users want to be capable of publish projects to accomplish their goals, to help their community.

3.2. Scope

The application has to allow the traveller to create the roadmap of his/her trip, search and find what he/she needs; projects, relevant information and people with the same interests. The application also has to allow creating a project and publishing it in a way it can be found.

3.3. Functional requirements

3.3.1 Actions in projects.

- Projects management has to allow the owner to:
  - Create projects inserting name, description, tasks and images.
  - Edit pictures and all text inputs.
  - Publish projects to be shown in internet.
  - Access or deny a user signing up to the project.
- Projects use has to allow the user to:
  - See all the information about the project.
  - Sign up to a project he/she is interested.

- In case of sign up approval, write comments on it.

3.3.1 Actions in roadmap.

- Roadmap management has to allow the owner to:
    - Add and delete countries of the roadmap.
    - Change the budget.
    - Add a new status or upload pictures.
    - Manage the friendship requests.
- Roadmap use has to allow the user to:
    - See countries of the roadmap.
    - Request friendship.
    - If they are friends, add comments or pictures to the roadmap.

3.3.1 Actions in tips.

- Tips management has to allow the owner to:
    - Add and delete tips.
- Tips use has to allow the owner to:
    - Like tips.
    - Put tips in his/her backpack for future visualization.
    - Delete tips from his/her backpack.

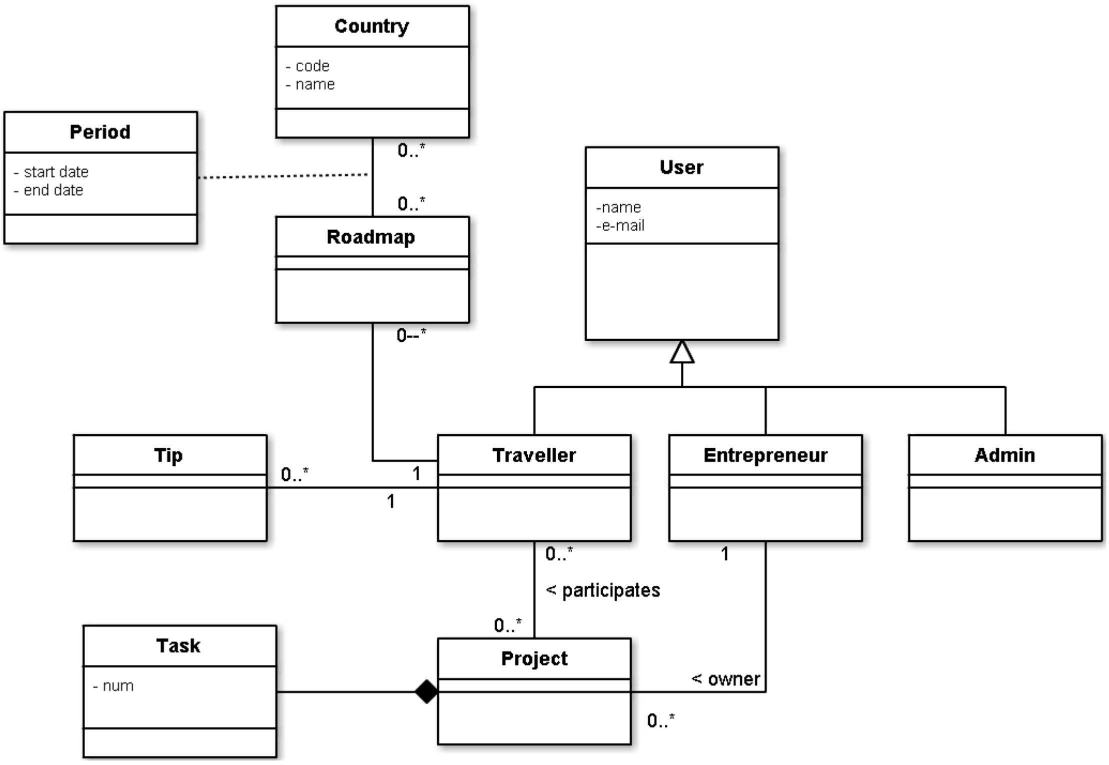
3.4. Non functional requirements

1. The application has to work in the majority of browser to allow people around access to the web.
2. The web has to be extensible to make it grow with new functionalities.
3. The user interface has to be usable and comfortable.
4. Has to be delivered a first phase by June 20th.

4. Specification

4.1. Conceptual Model

Following the conceptual model that illustrates in a simply way the concepts in real life.



Integrity restrictions:

- A user cannot own a project and participates to it.
- Admin has to be unique.

External key

- User: e-mail
- Country: code
- Period: start date, end date

- Task: num

The other objects use the internal identification (id) as a primary key.

- Project

  Represents a project and it contains a name, description and tasks.

- Task

  Represents a specific part of a project, it contains, a num, a name, description, reward, start date, end date

- User

  Represent a user of the application. A part of the name and the e-mail, it contains date of birth, gender, language, etc...

- Traveller

  Represents a user role: Traveller.

- Entrepreneur

  Represents a user role: Entrepreneur.

- Admin

  Represent the role of admin.

- Roadmap

  Represents the list of countries the user wants to visit.

- Country

  Represents a country.

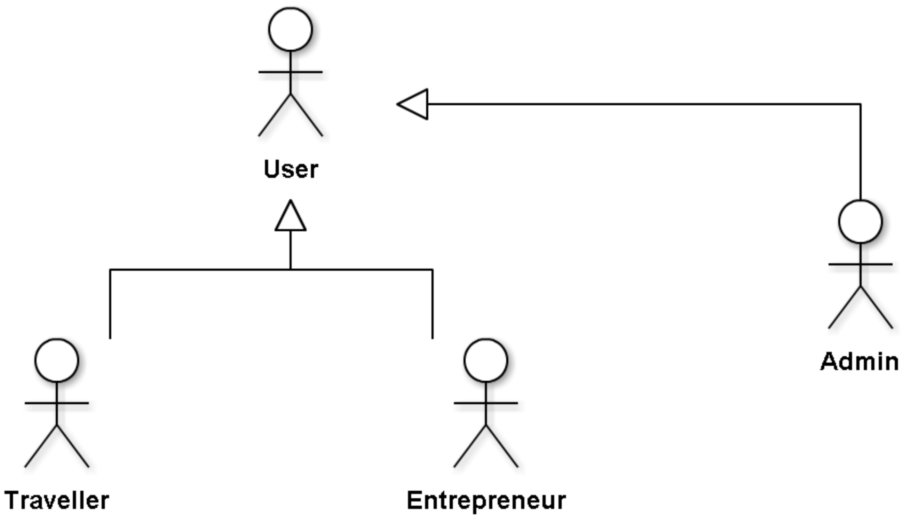
- Period

  Represents a period of time between start date and end date.

4.2. Actors

The actors Traveller and Entrepreneur interact with each other, and both extend from the same concept, it may be confusing due to any user can act as a Traveller and Entrepreneur depending on the actions we will distinguish them.

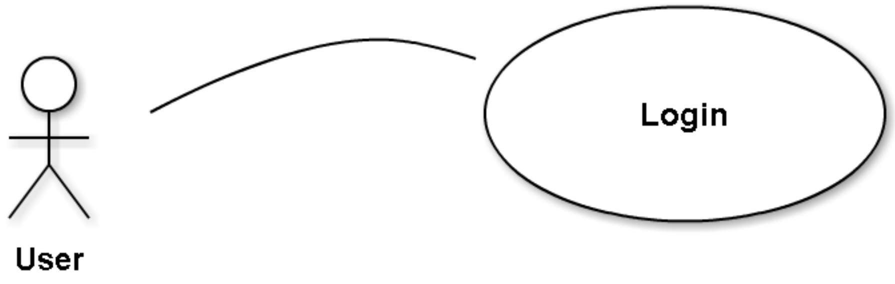
This is also a special user, the Admin. The admin is going to be a superuser, only with some restricted functionalities to do some concrete tasks.



4.3. Use Cases

Now we will define the interaction between actors and the system.

4.3.1. Log In



A user of SocialPackers.com logs in to the web application.

Actors: User

Starting point: This use case starts when a SocialPackers.com user is not logged in to the application and clicks to login.

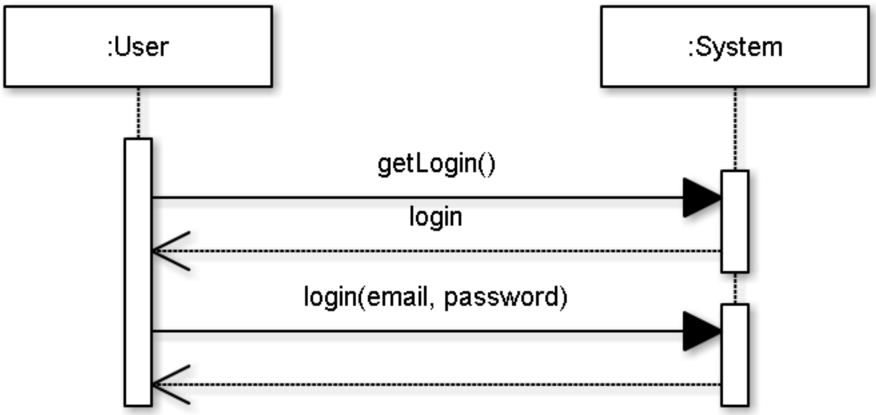
Precondition: The user is not logged in

Main flow:

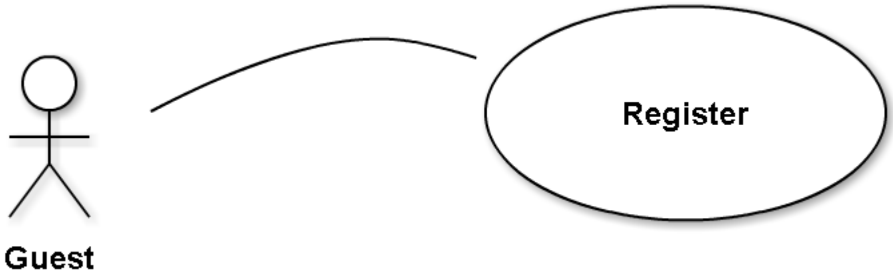
1. The System prompts the user for a username and password to login.
2. The user enters his/her username and password and click Login.
3. SocialPackers.com validates the entered username and password, making sure that the entered username is a valid username, and that the required password is entered for the entered username.

4a. Success: The User is authenticated and SocialPackers.com displays the roadmap page.

4b. Failure: SocialPackers.com redirects the user to the landing page.



4.3.2. Register



A user of SocialPackers.com creates an account.

Actors: Guest

Starting point: This use case starts when a SocialPackers.com user is not registered in to the application and clicks to login/registration.

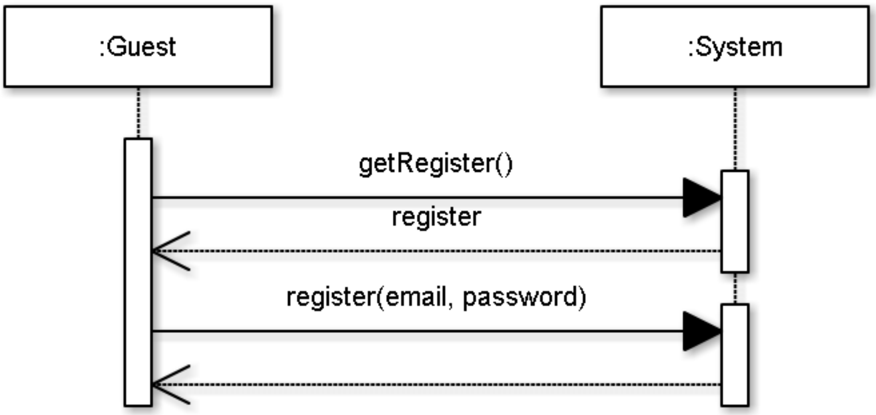
Precondition: The user is not registered

Main flow:

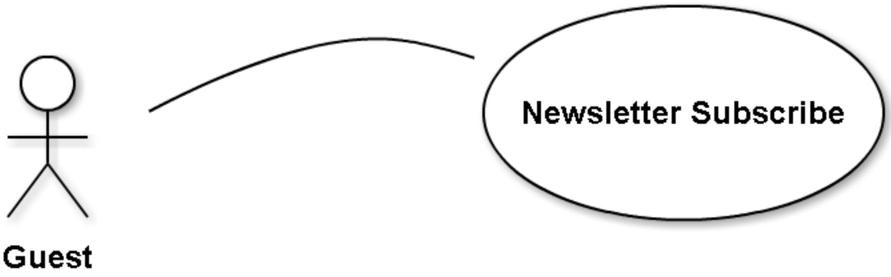
1. SocialPackers.com prompts the guest for a username and password to create a new account.
2. The guest enters in their information and click to Register.
3. SocialPackers.com verifies information and creates account.

4a. Success: SocialPackers.com displays the roadmap page.

4b. Failure: SocialPackers.com redirects the user to the landing page.



4.3.3. Newsletter Subscribe

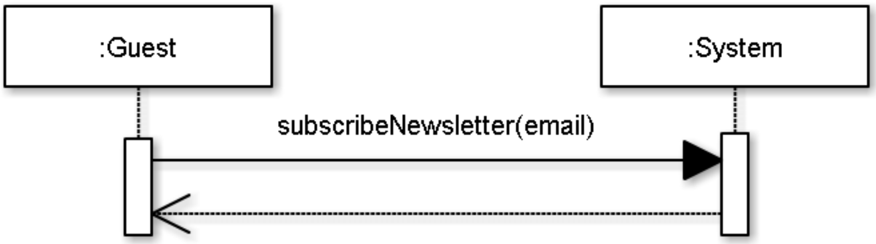


A guest of the SocialPackers.com subscribes to newsletters.

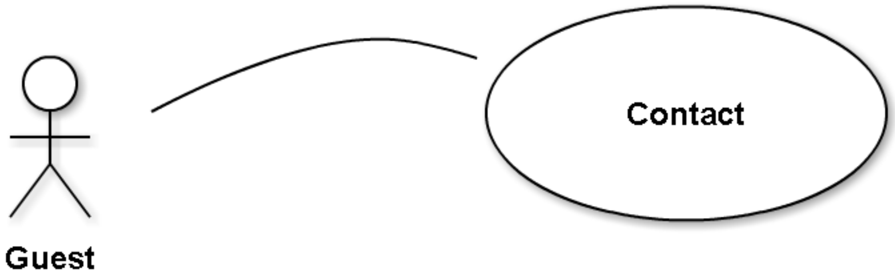
Actors: Guest/User

Main flow:

1. The footer has an input text to enter the e-mail.
2. The guest enters his/her e-mail and clicks Subscribe.
3. SocialPackers.com verifies information and adds the email to the list.



4.3.4. Contact



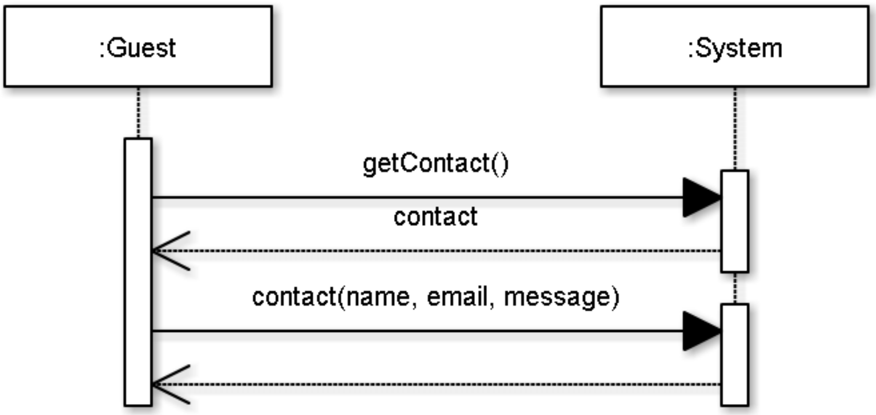
A guest/user of SocialPackers.com contacts with SocialPackers.com.

Actors: Guest/User

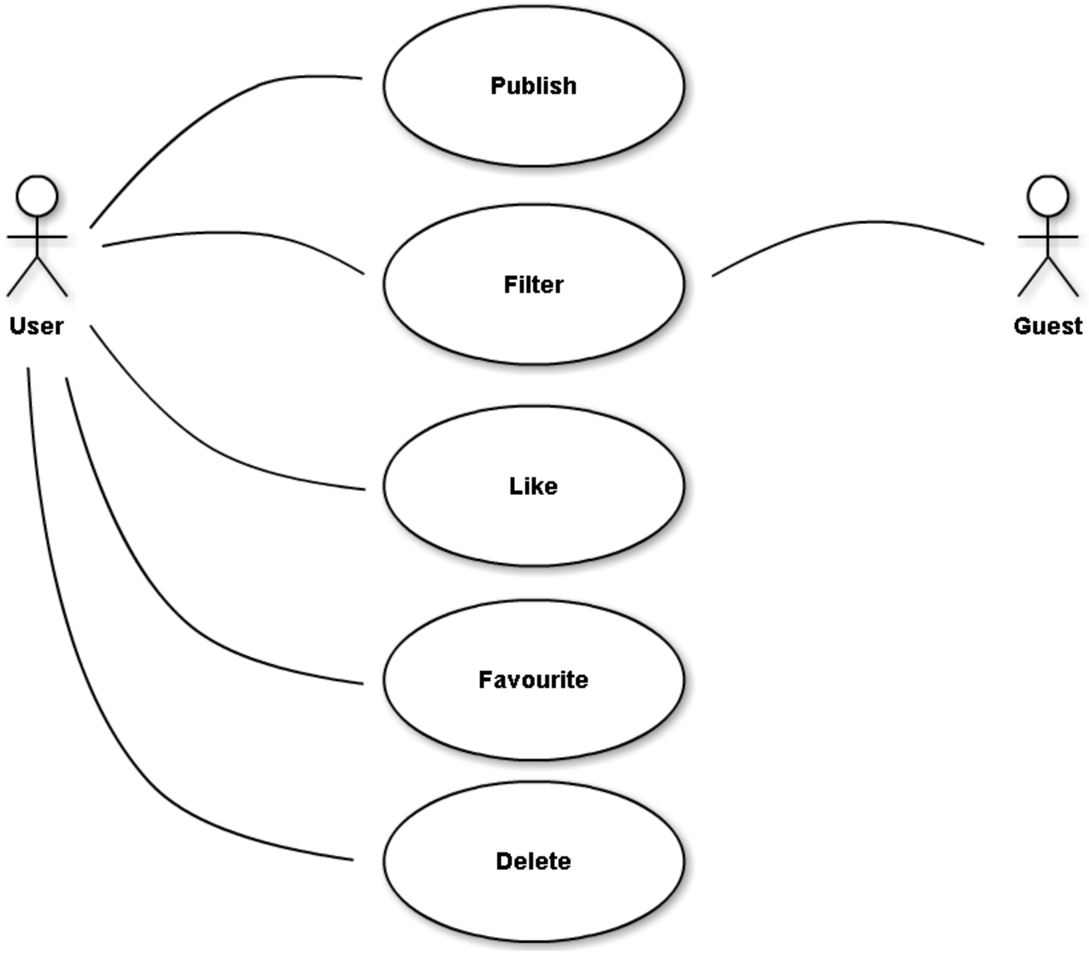
Starting point: This use case starts when a guest/user clicks on contact and goes to contact page.

Main flow:

1. SocialPackers.com shows the user/guest a form to send suggestions, complains or congrats.
2. The guest/user enters the required information and clicks to Send button.
3. SocialPackers.com verifies information and sends an email.



4.3.5. Tips



4.2.5.1. Publish

A logged user of SocialPackers.com publish a Tip

Actors: User

Starting point: This use case starts when a user goes to the tips section.

Precondition: The user is logged in

Main flow:

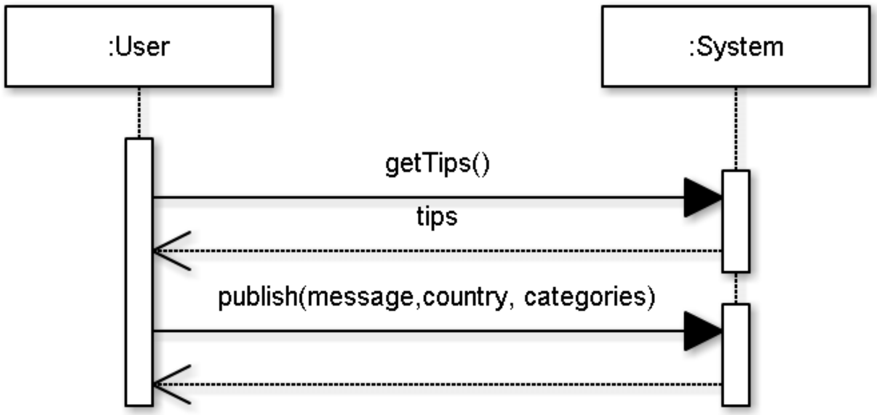
1. SocialPackers.com shows the user a form to publish/search a tip and a list of tips.

2. The user enters the information for a new tip.

3. SocialPackers.com give hints to the user with the Country and the Categories.

4. The user clicks on Publish button.

5. SocialPackers.com creates a tip and refreshes the page.



4.3.5.2. Filter

A guest of SocialPackers.com filters tips from a list.

Actors: Guest

Starting point: This use case starts when a guest goes to the tips section.

Main flow:

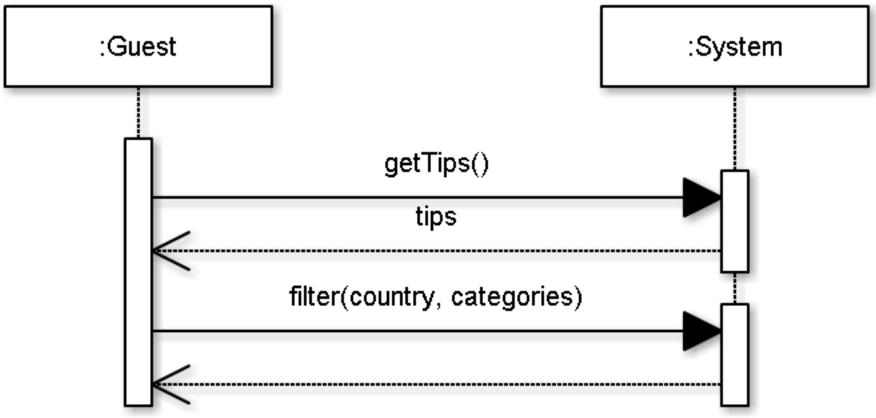
1. SocialPackers.com shows the guest a form to publish/search a tip and a list of tips.

2. The user enters the filters in the boxes to search for his/her preferences.

3. SocialPackers.com gives hints to the user with the Country and the Categories.

4. The user clicks on Filter button.

5. SocialPackers.com filters the list matching the information entered for the user.



4.3.5.3. Like

A logged user of SocialPackers.com likes tips from a list.

Actors: User

Starting point: This use case starts when a user goes to the tips section.

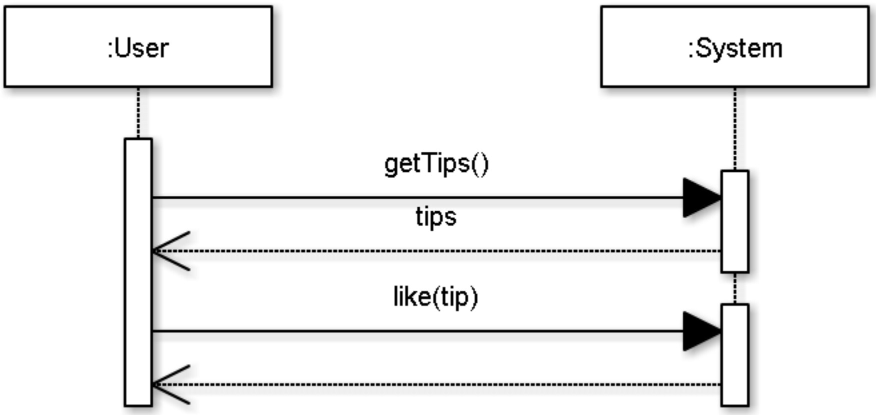
Precondition: The user is logged in

Main flow:

1. SocialPackers.com shows the user a form to publish/search a tip and a list of tips.

2. The user click “Like” button next to a tip he/she likes.

3. SocialPackers.com registers that “Like” and refreshes the information from the tip. (The counts of Likes of the Tip increments by 1.)



4.3.5.4. Favourite

A logged user of SocialPackers.com put a tip on his backpack.

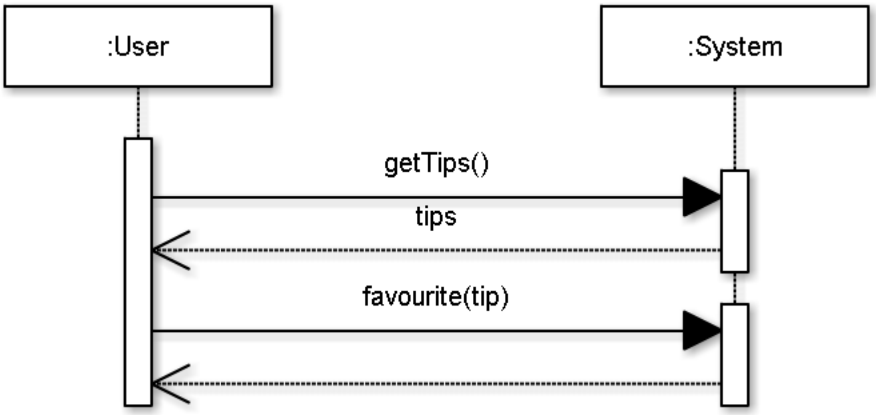
Actors: User

Starting point: This use case starts when a user goes to the tips section.

Precondition: The user is logged in

Main flow:

1. SocialPackers.com shows the user a form to publish/search a tip and a list of tips.
2. The user click Favourite button next to a tip he/she think is useful.
3. SocialPackers.com puts this Tip into the User's backpack.



4.3.5.5. Delete Favourite

A logged user of SocialPackers.com deletes a tip from his/her backpack.

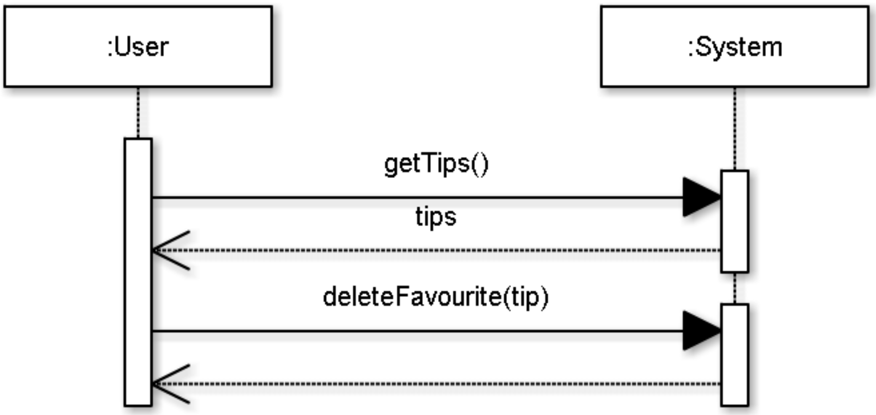
Actors: User

Starting point: This use case starts when a user goes to the tips section.

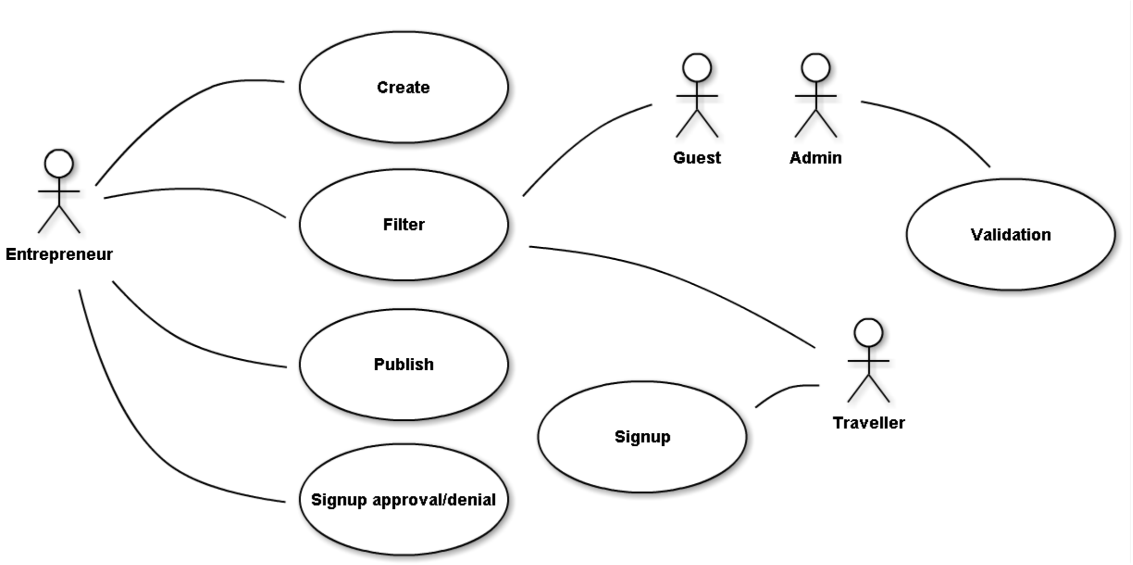
Precondition: The user is logged in

Main flow:

1. The user clicks to Backpack tab.
2. SocialPackers.com shows the user a list of tips.
3. The user click Delete button next to a tip he/she wants to delete.
4. SocialPackers.com deletes the Tip from the User's backpack.



4.3.6. Projects



4.3.6.1. Create

A logged user of SocialPackers.com creates a Project

Actors: User

Starting point: This use case starts when a user goes to the projects section and clicks the tab Create Project.

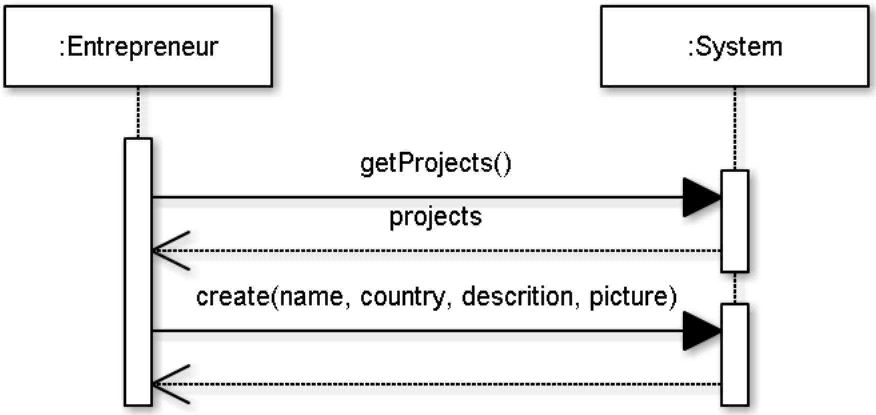
Precondition: The user is logged in

Main flow:

1. SocialPackers.com shows the user a form to create a project and the option of add task dynamically.
2. The user enters the information for a new project.
3. SocialPackers.com gives hints to the user with the Country and other.
4. The user clicks
   a. Create button. Goes to Step 5

b. Add Task. Goes to Step 2.

5. SocialPackers.com creates a project with the validated and published flag to FALSE in a new tab.



4.3.6.1. Edit

A logged user of SocialPackers.com edit a Project

Actors: User

Starting point: This use case starts when a user goes to the projects section and clicks the tab of an owned Project.

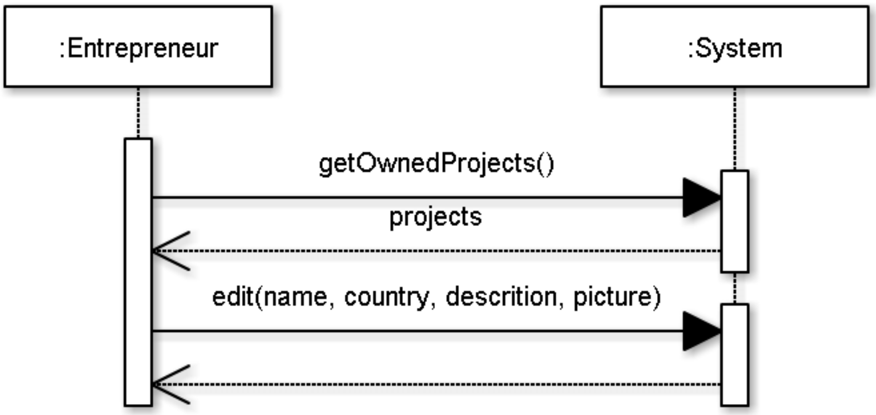
Precondition: The user is logged in

Main flow:

1. SocialPackers.com shows the user a form to edit a project and the option of add task dynamically.
2. The user enters the information for a new project.
3. SocialPackers.com gives hints to the user with the Country and other.
4. The user clicks
   a. Create button. Goes to Step 5

b. Add Task. Goes to Step 2.

5. SocialPackers.com edits the project with the validated and published flag to FALSE.



4.3.6.2. Filter

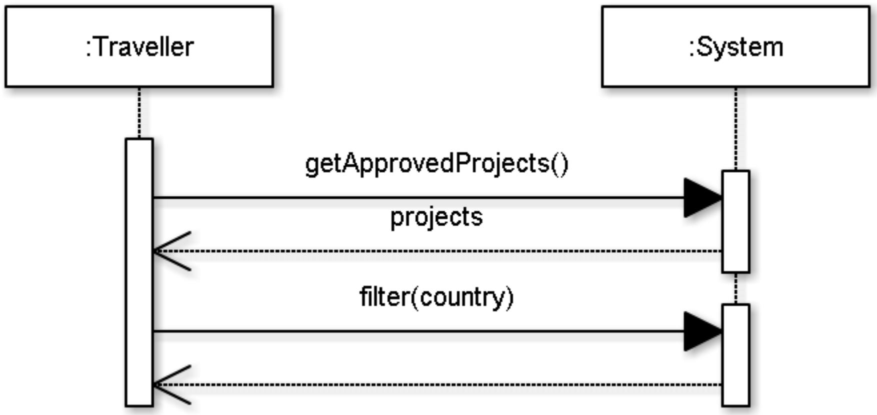
A guest of SocialPackers.com filters projects from a list.

Actors: Guest

Starting point: This use case starts when a guest goes to the projects section.

Main flow:

1. SocialPackers.com shows the guest a form to search a project and a list of projects.
2. The user enters the filters in the boxes to search for his/her preferences.
3. SocialPackers.com gives hints to the user with the Country.
4. The user clicks on Filter button.
5. SocialPackers.com filters the list matching the information entered for the user.



4.3.6.3. Publish

A logged user of SocialPackers.com publish a Project

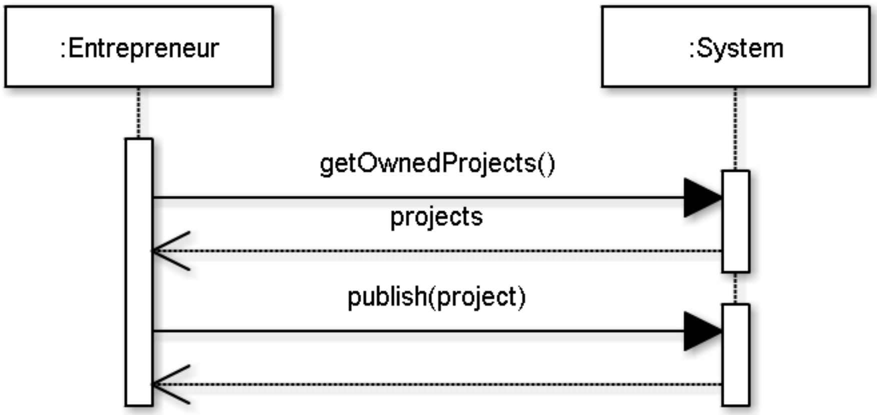
Actors: User

Starting point: This use case starts when a user goes to the projects section and clicks the publish button for a project.

Precondition: The user is logged in. Project must exist with published flag to FALSE.

Main flow:

1. SocialPackers.com shows the user a form to modify a project and the option of add task dynamically.
2. The user clicks the Publish button.
3. SocialPackers.com publishes the project with the validated flag to FALSE.



4.3.6.4. Validation (ADMIN)

An admin user of SocialPackers.com checks a list of projects pending of approval.

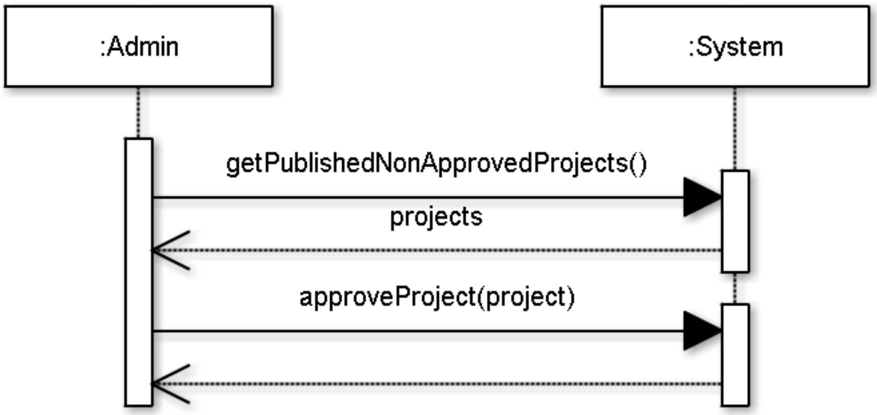
Actors: Admin

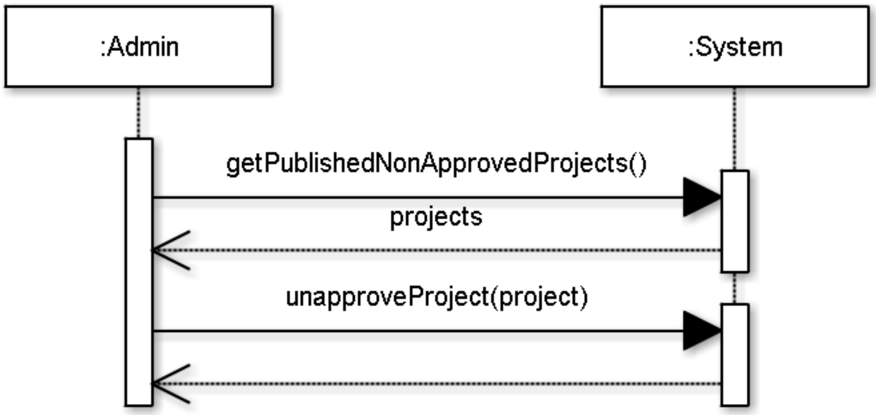
Starting point: The admin user goes to Project Validation section.

Precondition: The user is logged in as admin.

Main flow:

1. SocialPackers.com shows the admin user a list of projects pending of approval.
2. The admin user checks the information and clicks Approve or Deny.
3. If Approve a. If Deny b.
    - *a.* SocialPackers.com changes the status of the project to Approved.
    - *b.* SocialPackers.com changes the status of the project to Denied.
4. SocialPackers.com informs the user with the new status.





4.3.6.5. Signup

A logged user of SocialPackers.com requests a signup to a project.

Actors: Admin

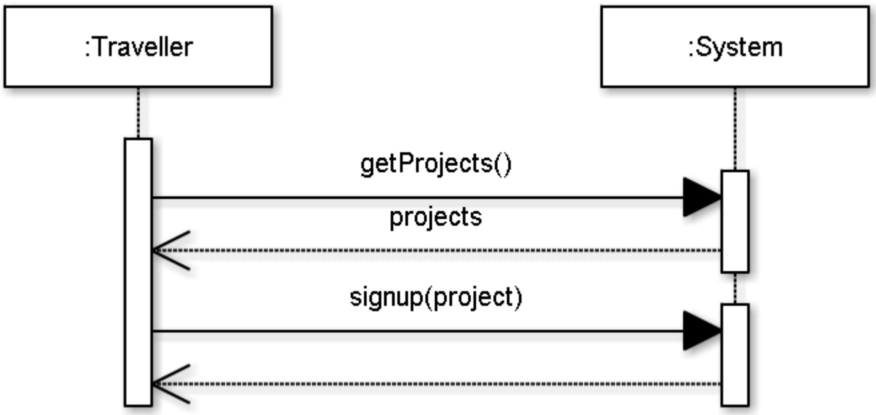
Starting point: This use case starts when a user goes to the search project section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com shows the user a list of projects.
2. The user click Signup button next to a project he/she likes.

3. SocialPackers.com registers that Signup and informs the owner about the request.



4.3.6.6. Signup approval/denial

A logged user of SocialPackers.com approves or denies a request of signup.

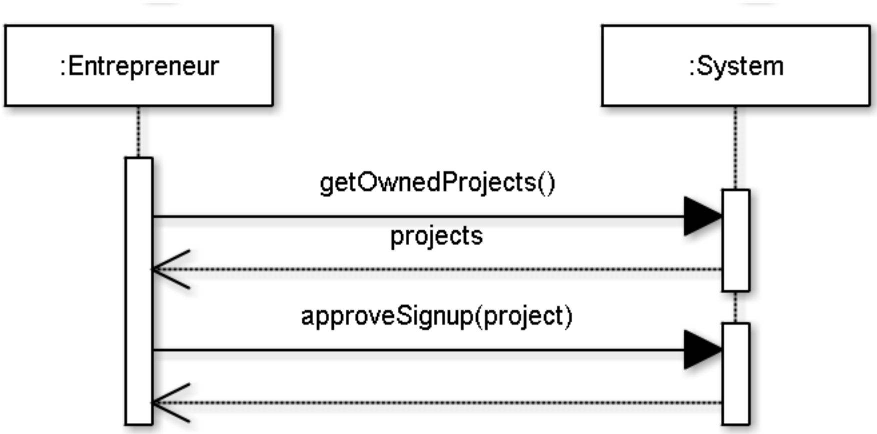
Actors: User

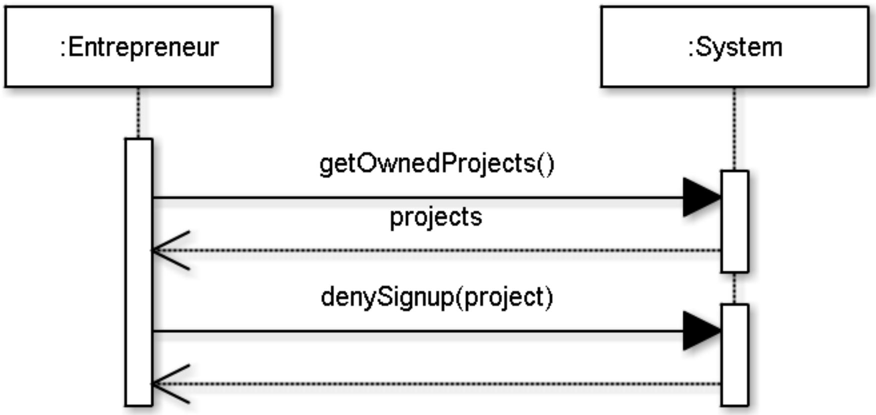
Starting point: This use case starts when a user goes to the owned project section.

Precondition: The user is logged in.

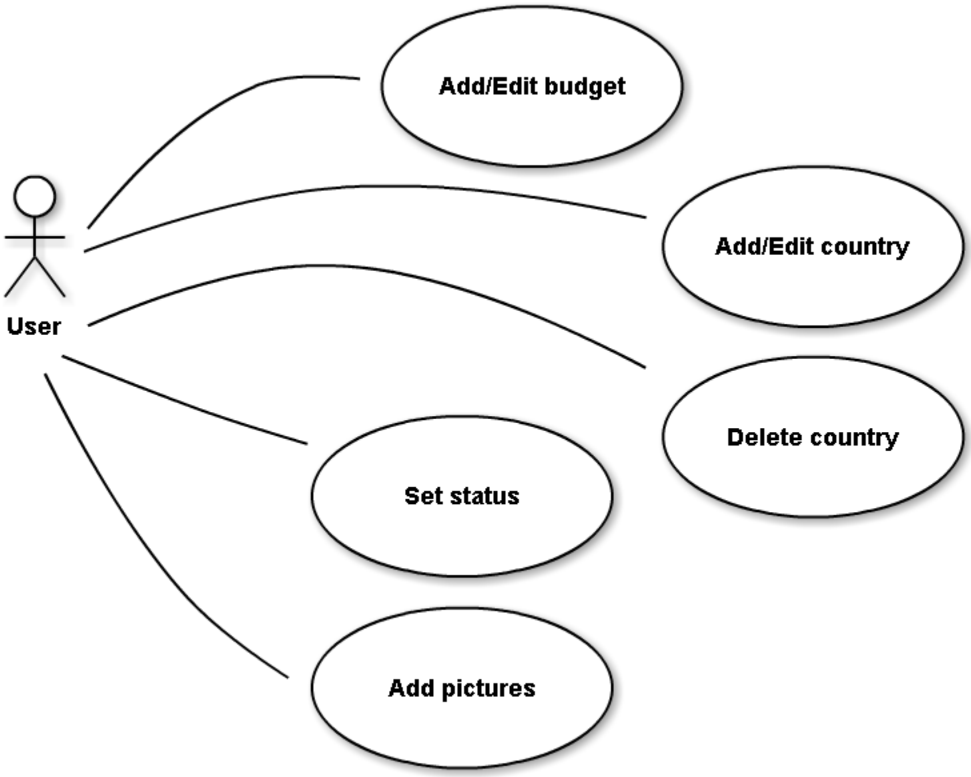
Main flow:

1. SocialPackers.com shows the user a list of projects owned by him/her with a list of signups request from users.
2. The user click Approve/Deny button next to a user request.
3. SocialPackers.com links the User and Project and informs the user about the approval or it deletes the request in case of denial.


:Entrepreneur
:System
getOwnedProjects()
projects
approveSignup(project)


:Entrepreneur
:System
getOwnedProjects()
projects
denySignup(project)

4.3.7. Roadmap



4.3.7.1. Add budget

A logged user of SocialPackers.com adds the budget to the roadmap.

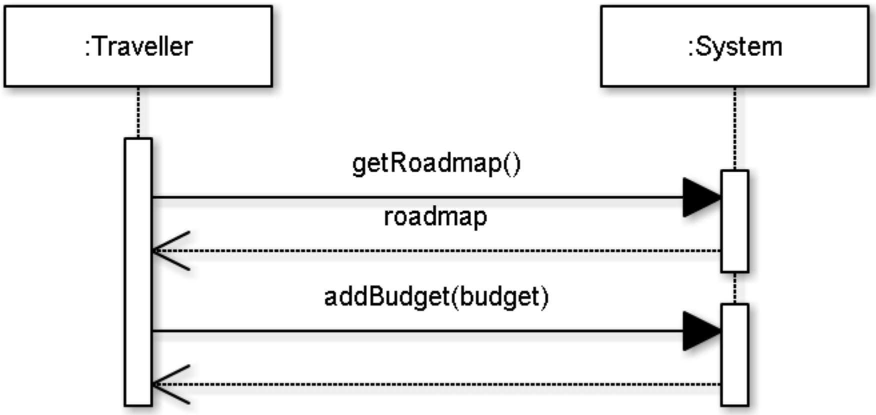
Actors: User

Starting point: This use case starts when a user goes to the roadmap section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com shows the user an input box to add the budget for the roadmap bars.
2. SocialPackers.com registers the budget dynamically or when the user clicks to add, and push the info to the corresponding bar.



4.3.7.2. Add country

A logged user of SocialPackers.com adds a country to his/her roadmap.

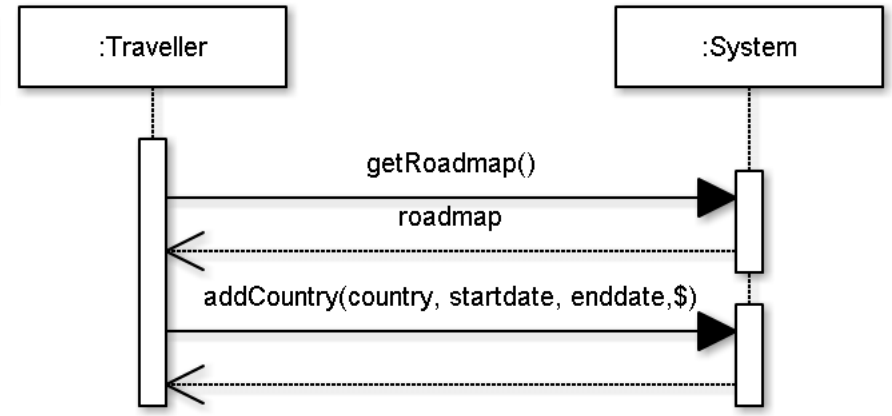
Actors: User

Starting point: This use case starts when a user goes to the roadmap section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com clicks to the AddCountry button.
2. SocialPackers.com shows a row with inputs to enter information, the country, date and budget.
3. SocialPackers.com registers the country dynamically or when de user clicks to save.



4.3.7.3. Delete country

A logged user of SocialPackers.com deletes a country to his/her roadmap.

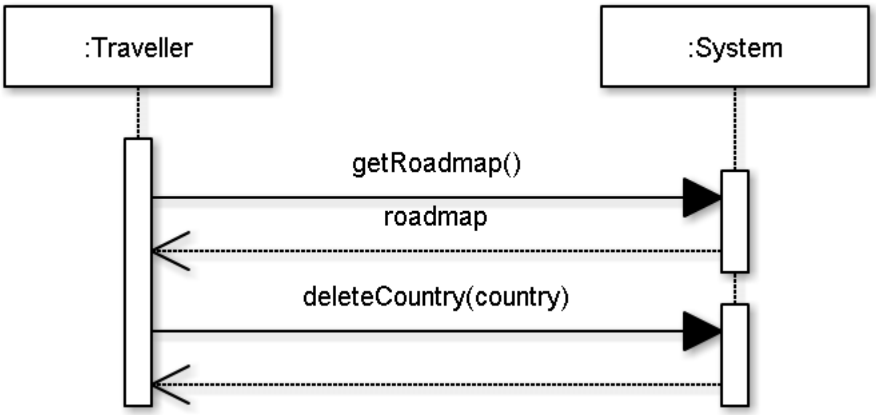
Actors: User

Starting point: This use case starts when a user goes to the roadmap section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com clicks to the “X” button next to the country
2. SocialPackers.com deletes the row.
3. SocialPackers.com deletes the country dynamically or when de user clicks to save.



4.3.7.4. Set status

A logged user of SocialPackers.com sets a status on his roadmap.

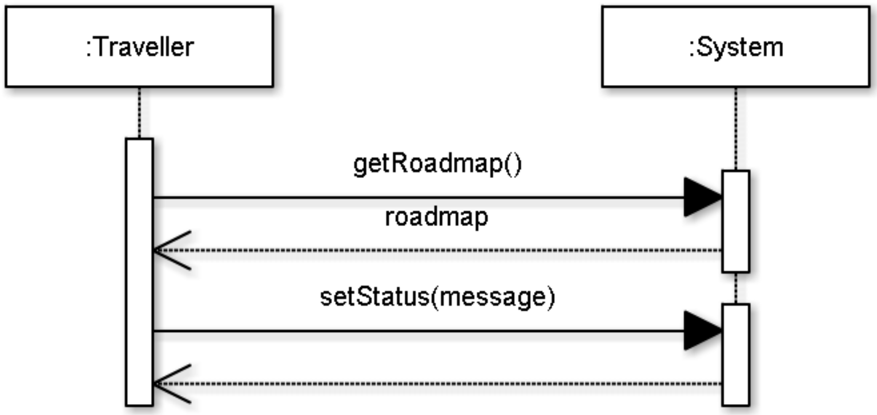
Actors: User

Starting point: This use case starts when a user goes to the roadmap section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com writes text to the status input on the beginning of the roadmap section.
2. The user clicks the button to submit the sentence.
3. SocialPackers.com publishes the sentence to the roadmap.



4.3.7.5. Add pictures

A logged user of SocialPackers.com upload pictures on his roadmap.

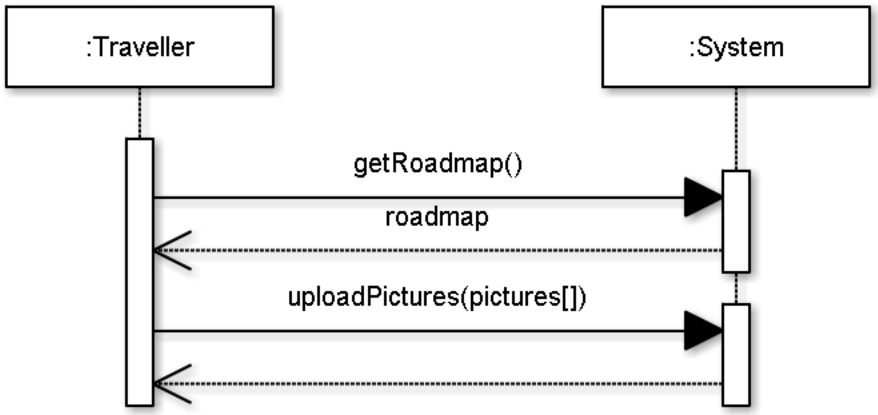
Actors: User

Starting point: This use case starts when a user goes to the roadmap section.

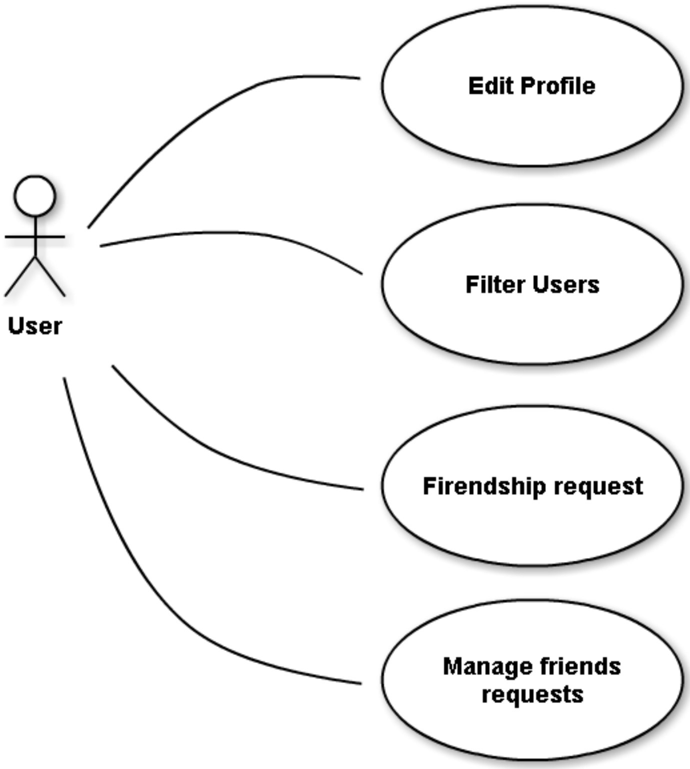
Precondition: The user is logged in.

Main flow:

1. SocialPackers.com drag pictures on the beginning of the roadmap section.
2. The user clicks the button to upload pictures.
3. SocialPackers.com publishes the pictures to the roadmap.



4.3.8. Users



4.3.8.1. Edit Profile

A logged user of SocialPackers.com edits his/her profile of SocialPackers.com

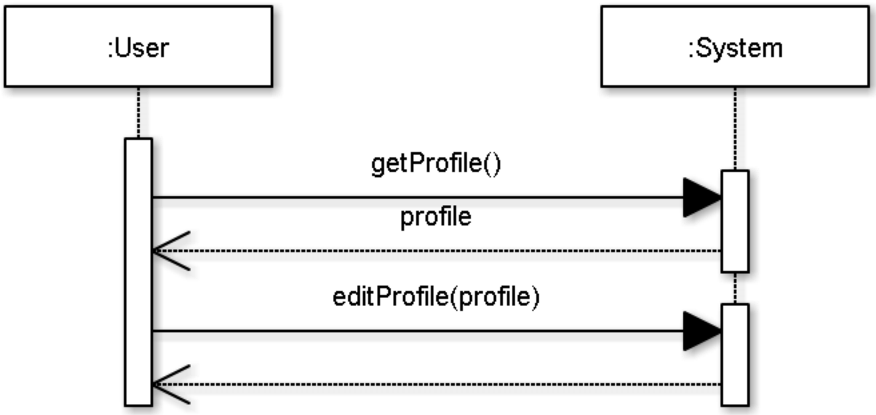
Actors: User

Starting point: This use case starts when a user clicks on Profile contact and goes to profile page.

Precondition: The user is logged in

Main flow:

1. SocialPackers.com prompts the user a form to edit his/her personal data and picture.
2. The user enters the information he/she wants to change and clicks to Modify button.
3. SocialPackers.com verifies information and save the new information for the logged user and reload the profile page.



4.3.8.2. Filter

A logged user of SocialPackers.com filters users from a list.

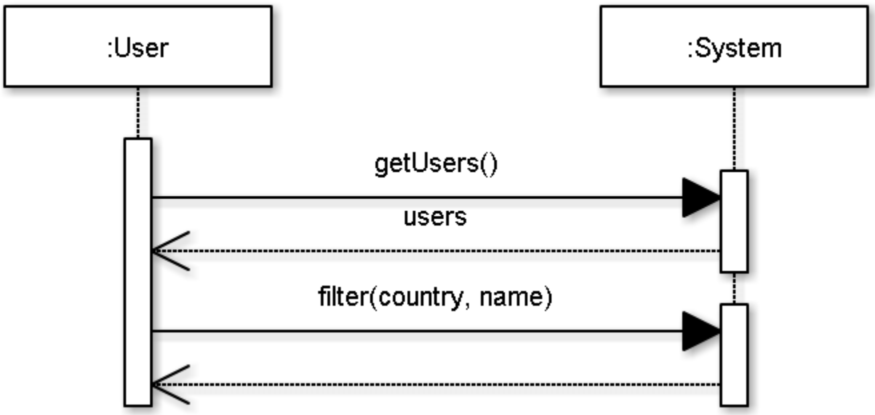
Actors: User

Starting point: This use case starts when the user goes to the user section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com shows the user a form to search users and a list of them.
2. The user enters the filters in the boxes to search for his/her preferences.
3. The user clicks on Filter button.
4. SocialPackers.com filters the list matching the information entered for the user.



4.3.8.3. Friendship request

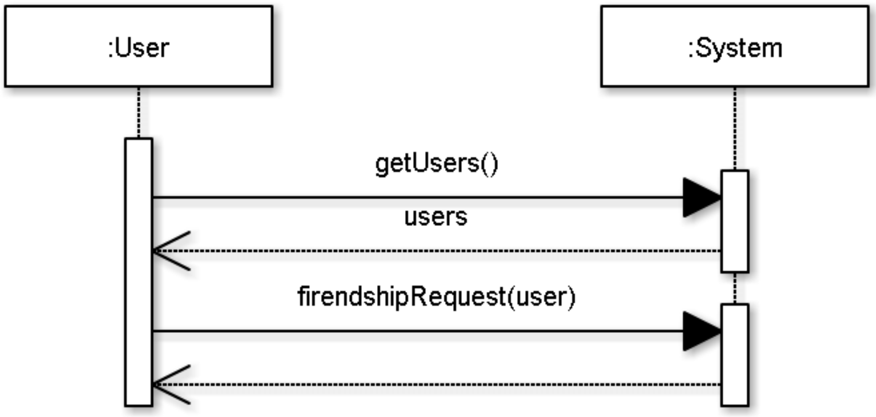
A logged user of SocialPackers.com requests friendship to another user.

Actors: User

Starting point: This use case starts when a user goes to the user section.

Main flow:

1. SocialPackers.com shows the user a form to search users and a list of them.
2. The user clicks the “Friendship request” next to the user.
3. SocialPackers.com sends the request to the destiny user.



4.3.8.4. Manage friends requests

A logged user of SocialPackers.com accept/deny friendship to another user

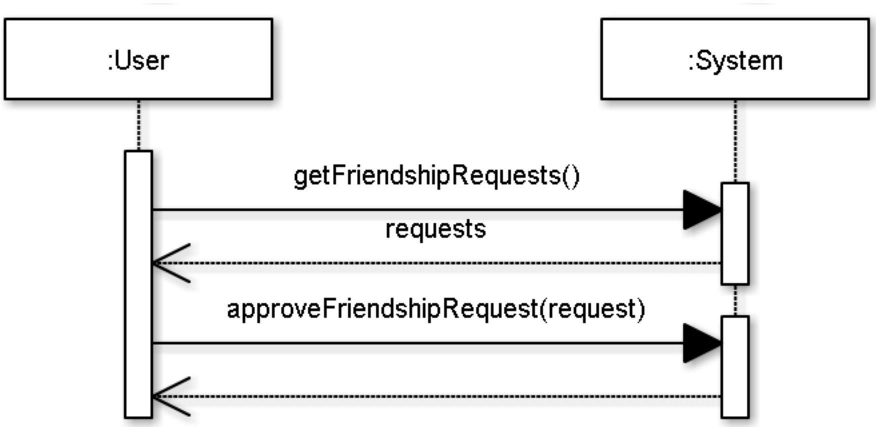
Actors: User

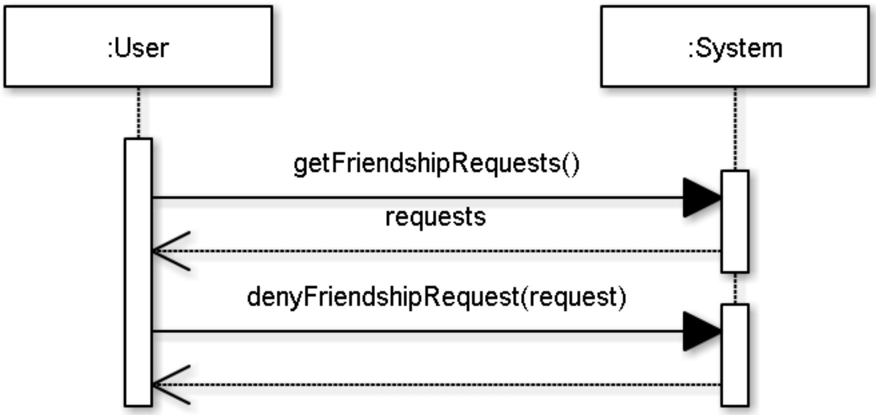
Starting point: This use case starts when a user goes to the roadmap section.

Precondition: The user is logged in.

Main flow:

1. SocialPackers.com clicks on the envelope on the top of the roadmap section.
2. SocialPackers.com shows a list of requests.
3. The user can Deny/Accept friendship.





4.4 Contracts

Function getLogin(): login

Get the login page

Preconditions: -

Postconditions: Returns the login page

Function login(user: String, password :String)

Authenticates the user as a logged user.

Preconditions: User and password are not empty.

Postconditions: Returns true if the users exists in database, otherwise returns false.

Function getRegister(): register

Get the register page

Preconditions: -

Postconditions: Returns the register page

Function register(email: String, password :String)

Creates a user with the email and password

Preconditions: The email and the password are not empty.

Postconditions:

1. Creates a User with email and password.
2. Sends an email to the given email

Function subscribeNewsletter(email :String)

Subscribes the email to the newsletter.

Preconditions: The email is not empty

Postconditions: Creates a new entry in newsletter

Function getContact(): contact

Get the contact page

Preconditions: -

Postconditions: Returns the contact page

Function contact (name :String, email :String, message :String)

Sends an email to the web administrator

Preconditions: Name, email and message are not empty

Postconditions: An email is send to the web administration with the email, name and message

Function getTips(): tips

Get the tips page

Preconditions: -

Postconditions: Returns the tips page

Function publishTip(message :String, country :String, categories :String)

Publish a new tip.

Preconditions: The message are not empty

Postconditions: Creates a tip with the message and add country and categories if they are not empty

Function filterTips(country :String, categories :String): tips

Filter the list of tips of the page

Preconditions: This is at least one parameter informed

Postconditions: Return a list of tips that match with the informed parameters.

Function likeTip(tip : Tip)

Likes a tip

Preconditions: tip is not empty and exists.

Postconditions: The number of likes of the tip increases by one.

Function favouriteTip(tip : Tip)

Adds the tip to the backpack of the user

Preconditions: tip is not empty and exists

Postconditions: The tip is added to the backpack of the logged use.

Function deleteFavouriteTip(tip : Tip)

Deletes a tip from the backpack of the user

Preconditions:

1. tip is not empty, exists
2. tip exists in the backpack of the logged user.

Postconditions: The tip is deleted from the backpack of the logged user logged.

Function getProfile(): profile

Get the profile page

Preconditions: The user is logged in.

Postconditions: Returns the profile page for the user logged.

Function editPorfile(name :String, middle_name :String, last_name :String, email :String, birth_date :Date, about :String, what :String, gender :String)

Edits the profile of the logged user.

Preconditions: email is not empty

Postconditions: Modifies the profile of the logged user.

Function getUsers() : users

Get all users

Preconditions: -

Postconditions: Returns a list of users

Function filterUsers(country: String, name: String) : users

Get the list of users matching the values of the parameters.

Preconditions: -

Postconditions: Returns a list of users matching the values of the parameters.

Function friendshipRequest(user: User)

Sets a request to friendship

Preconditions: user is not empty and exists.

Postconditions: Creates a request of friendship from user logged to user as a parameter

Function approveFriendshipRequest(user: User)

Set a friendship request to TRUE.

Preconditions: A request from the user exists.

Postconditions: Set the friendship request to TRUE.

Function denyFriendshipRequest(user: User)

Delete a friendship request.

Preconditions: A request from the user exists.

Postconditions: Deletes the friendship request from the system.

Function getProjects(): projects

Get all projects

Preconditions: -

Postconditions: Returns a list of all projects.

Function createProject(name, country, description, picture)

Creates a project.

Preconditions: name is not empty.

Postconditions: Creates a project with name, country, description and picture with the flags approved and published to FALSE

Function editProject(name, country, description, picture): project

Edits a project

Preconditions: name is not empty

Postconditions: Edits a project with name, country, description and picture and set the flags approved and published to FALSE

Function getApprovedProjects() : projects

Get the projects that has been approved.

Preconditions: -

Postconditions: Returns a list of projects that has been approved (has the approved flag FALSE)

Function filterProjects(country: String) : projects

Get the projects matching the country

Preconditions:

Postconditions: Returns a list of projects matching the country, if it is informed.

Function getOwnedProjects(): projects

Get the projects owned by the logged user.

Preconditions:

Postconditions: Returns a list of projects owned by the logged user.

Function publishProject(project: Project)

Publish a project.

Preconditions: project is not empty and exists.

Postconditions: the flag publish of the project is set to TRUE.

Function signupToProject(project:Project)

Makes a request to sign up to a project.

Preconditions: project is not empty and exists.

Postconditions:A request to sign up to project is created.

Function approveSignupToProject(project: Project, user:User)

Approve the request to sign up from the user to the project.

Preconditions:

1. project is not empty and exists
2. user is not empty and exists

Postconditions: The request to sign up to project is approved.

Function denySignupToProject(project: Project)

Denies the request to sign up from the user to the project.

Preconditions: -

1. project is not empty and exists
2. user is not empty and exists

Postconditions: The request to sign up to project is denied.

Function getPublishedNonApprovedProjects() :projects

Get the projects that has been published but not approved.

Preconditions: -

Postconditions: Returns a list of projects that has been published but not approved.

Function approveProject(project: Project)

Approves the project

Preconditions: project is not empty and exist

Postconditions:The flag approved of project is setted to TRUE

Function disapproveProject(project: Project)

Approves the project

Preconditions: project is not empty and exist

Postconditions:The flag approved of project is set to TRUE

Function getRoadmap() : roadmap

Get the roadmap for the logged user.

Preconditions: -

Postconditions: Returns the roadmap associated to the logged user.

Function updateBudget(budget: Integer)

Updates the budget of the roadmap

Preconditions: budget is not empty

Postconditions: Updates the buget of the roadmap of the logged user.

Function addCountry(country: String, startdate: Date, enddate: Date, budget: num) :country

Add a country to the roadmap

Preconditions: country is not empty

Postconditions: A Country with country, startdate, enddate and budget is created and associated to roadmap.

Function deleteCountry(country: Country)

Deletes a country of the roadmap

Preconditions country is not empty and exists

Postconditions: Deletes the country of the roadmap of the logged user.

Function setStatus(message: String)

Get the login page

Preconditions: message is not empty

Postconditions: the message is saved and linked to the roadmap of the logged user.

Function uploadPictures(pictures: Array[Pictures])

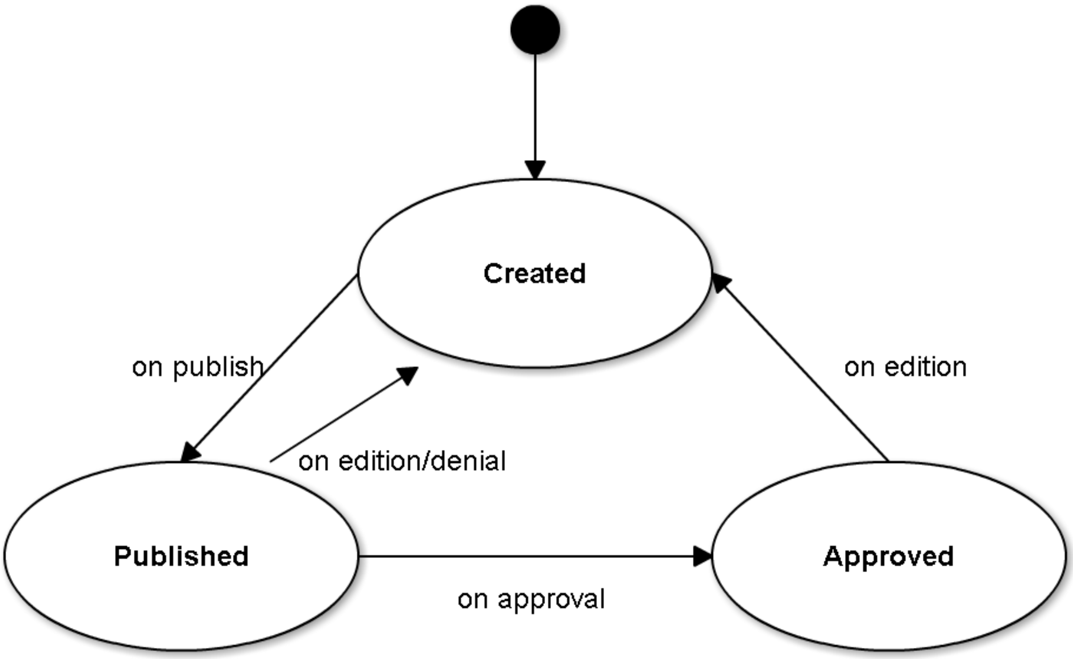
Upladod pictures to the roadmap

Preconditions: pictures is not empty

Postconditions: the pictures are uploaded to the filesystem and linked to the roadmap of the logged user.

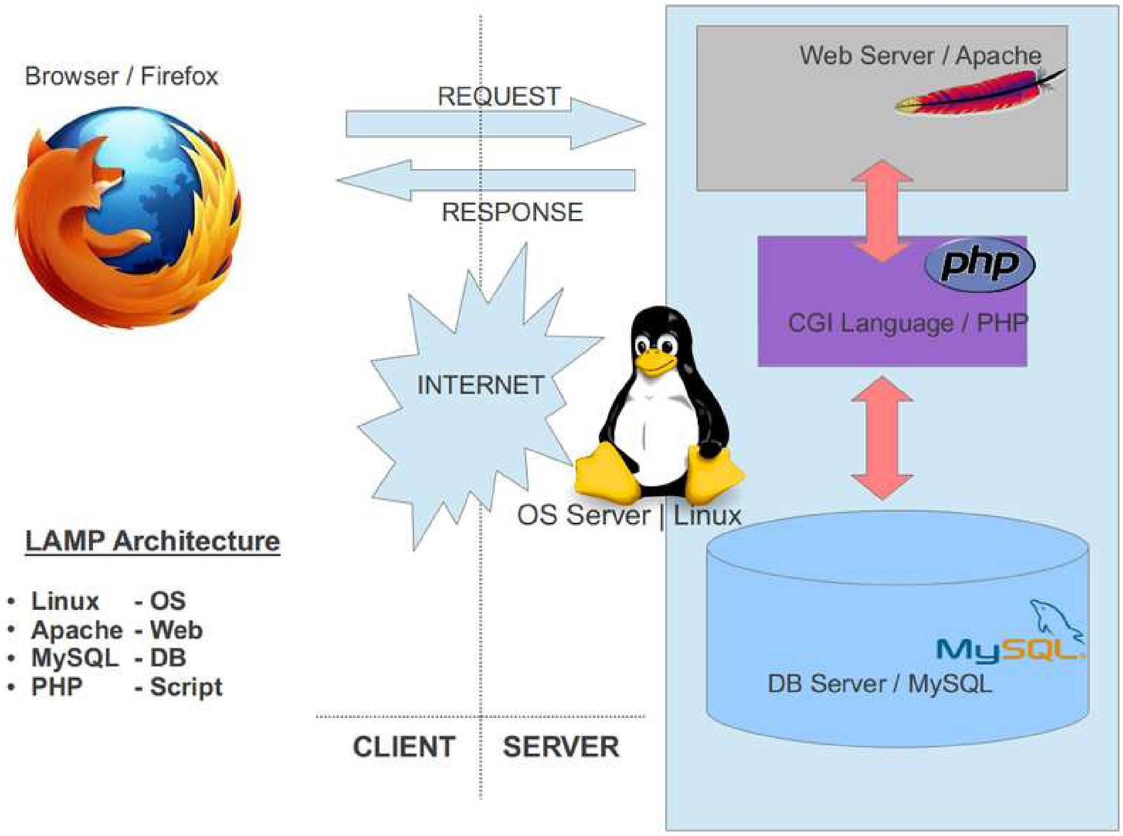
4.5 State diagrams

The next illustration shows the state diagram of a project.



5. Design

5.1. Physical view



Here explained what we have to run the web application.

It is called LAMP because of Linux + Apache + MySQL + PHP.

Take into account:

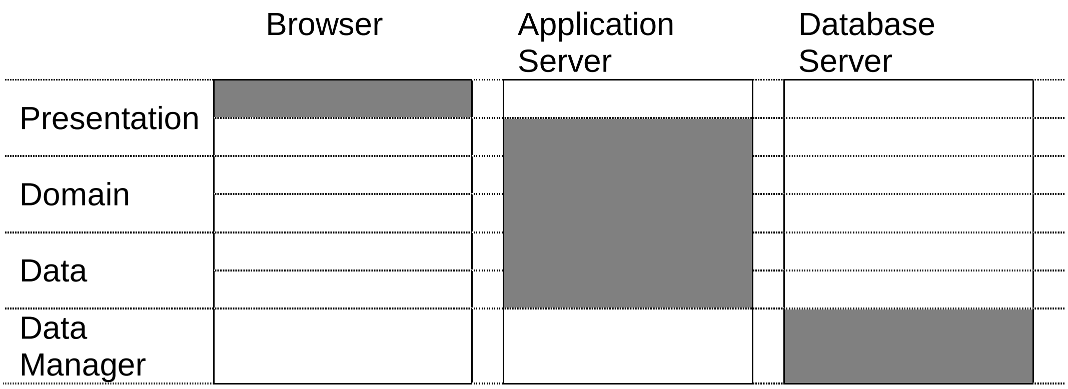
We will have a MySQL 5.x database installed in a different Server (provided by the hosting). This way we assure that no data will be lost.

At the moment we will use the filesystem of the app server to store multimedia files. We are searching more options.

5.2. Logic architecture: Layer design

5.2.1. 3-layer design

We decided to use a 3-layer design.



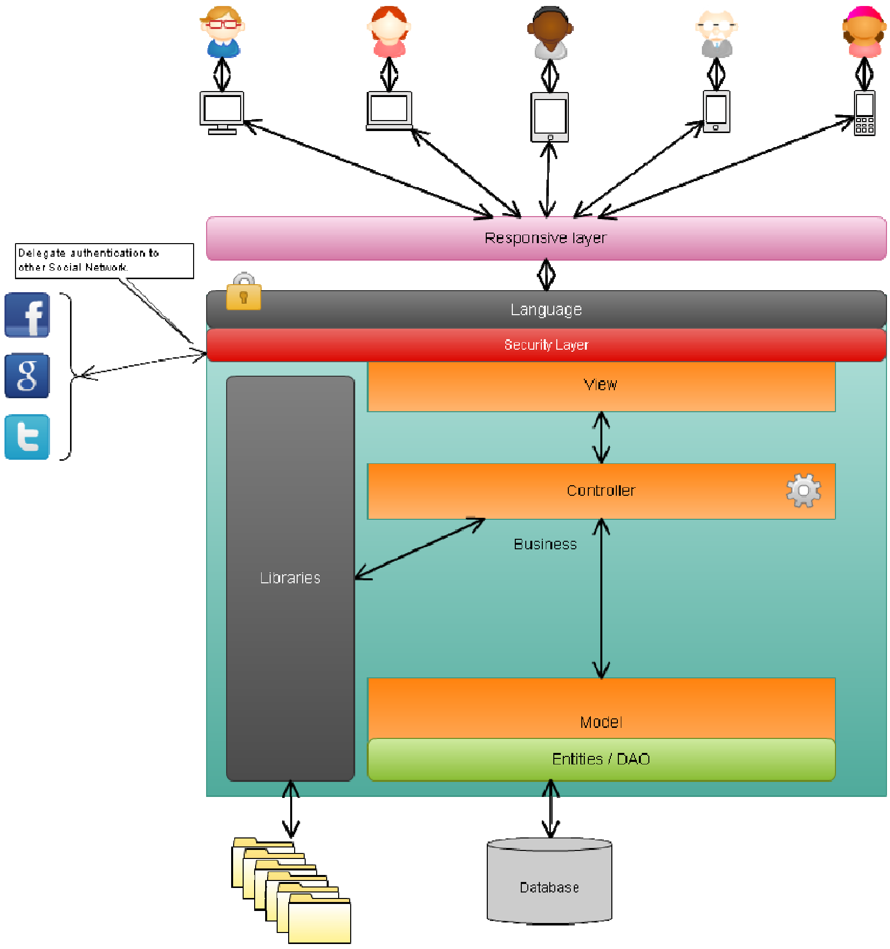
The browser only has a thin part of the presentation layer. The application server has part of the presentation layer and the domain and data layers. The database server has the database manager installed.

5.2.2. X-layer design

The application is following the 3-layer design, but we separated it in a conceptual way to keep all easy to understand and to have an easy-maintenance -way.

Following we can see an illustration of what we have.

Added the Responsive layer, the Language layer and the Security layer.



Client view layer

To manage and implement this layer we used JavaScript with jQuery framework.

Responsive layer

With the help of Twitter Bootstrap framework we achieved the hard task to adapt the webpage to different screen sizes, from

Language layer

We detect the language of the user's browser (if the user has not one set) and if we support it we load the files to show the webpage with the language desired.

Security layer

Custom implemented layer that validates the authenticity of the credentials of the session of the user.

View layer

Set of classes that dynamically creates the HTML showed to the user. Codeigniter framework helps us to manage it.

Controller layer

This layer is one that interacts between the view and the model making all the logic needed for the task to be completed. Codeigniter framework helps us to manage it.

Model layer

The model layer has all the classes that interact directly with the database. This is automatically created by Doctrine 2.

5.2.2.1. Used technologies in each layer

- View Layer
    - JavaScript + jQuery framework + Twitter Bootstrap framework.
- Controller Layer
    - PHP + Codeigniter framework.
- Model layer
    - PHP + Doctrine 2 framework.
- Data layer

- MySQL 5.x

JavaScript

Is an interpreted programming language originally created to interact with the DOM in client side. Therefore, nowadays there are applications that have all the logic in the JavaScript.

jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.[1]

Twitter Bootstrap

Twitter Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions.

It helps to look and behave great in the latest desktop browsers (as well as IE7!), but in tablet and smartphone browsers via responsive CSS as well.

PHP

[1] definition from http://www.jquery.com

Is an interpreted programming language by a web server with a PHP processor module which generates dynamic web pages: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data.

CodeIgniter

CodeIgniter is a powerful PHP framework with a very small footprint, built for PHP coders who need a simple and elegant toolkit to create full-featured web applications. If you're a developer who lives in the real world of shared hosting accounts and clients with deadlines, and if you're tired of ponderously large and thoroughly undocumented frameworks, then CodeIgniter might be a good fit.[2]

Doctrine 2

It is a persistence framework. For one side it has a Object Relational Mapper (ORM) and in the other side the Database Abstraction Layer (DBAL).

DBAL is a powerful database abstraction layer with many features for database schema introspection, schema management and PDO abstraction.

ORM if sits on top of a DBAL. One of its key features is the option to write database queries in a proprietary object oriented SQL dialect called Doctrine Query Language (DQL), inspired by Hibernates HQL. This provides developers with a powerful alternative to SQL that maintains flexibility without requiring unnecessary code duplication.

[2] Full or part of the text was extracted from http://ellislab.com/codeigniter

MySQL 5.x

SocialPackers.com uses MySQL 5.x as a data base management system. MySQL can use several engines for database implementation. In the case of SocialPacker.com we use InnoDB; it allows us to deal with referential integrity better than the MyISAM engine.

5.2.3. View layer design

We decided to make tips and projects visible to outsiders to attract people, because what we want is to have people interested in travel, interested in social projects.

We decided to make projects and tips public to let people see and try the possibilities, a part of let search robots index our projects info to make us more visible on the web.

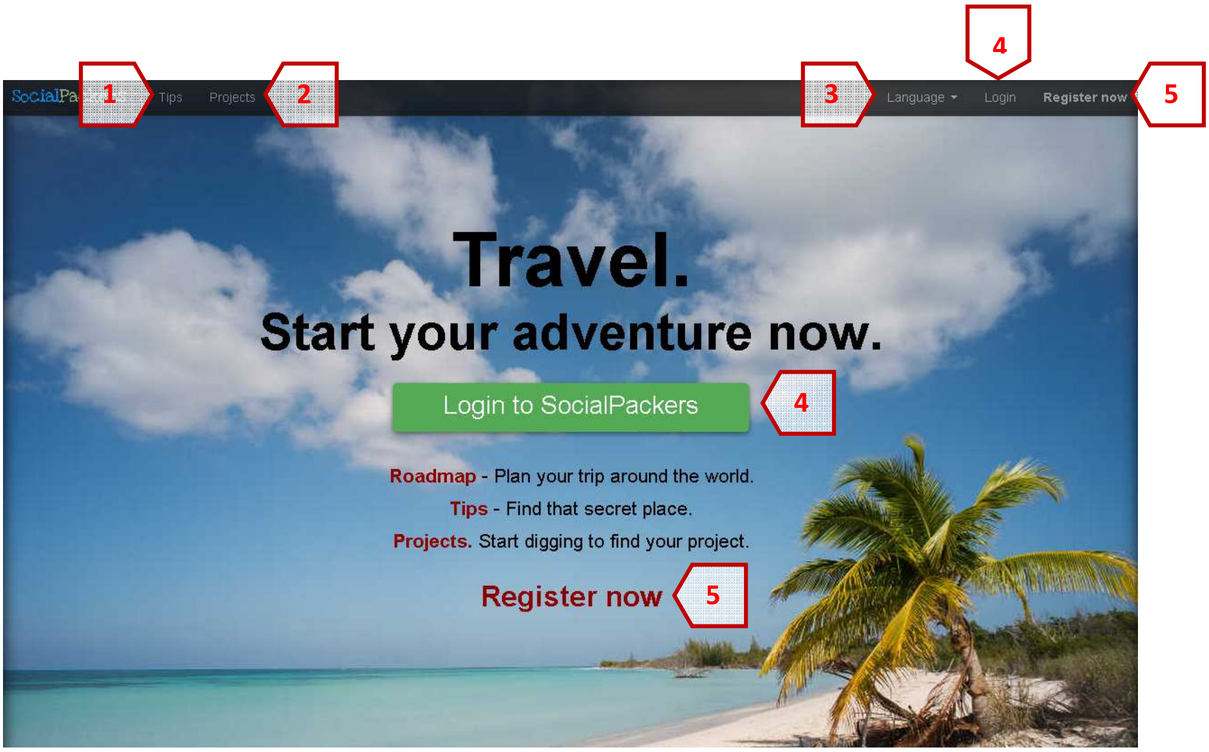
We also would like to have several public pages in the footer that allows people to see who we are, see the new features and news related to Socialpackers.com in the blog, contact us to suggest or comment anything, subscribe to our newsletter to keep up to date information about Socialpackers.com events and novelties.

They could also read our privacy and terms and condition notes.

5.2.3.1. Landing page

In the landing page we can see that we already have access to the Tips (1) and Projects(2) section. They are partially public. The people can search and see Projects and Tips, but if they want to interact somehow with them, it redirects to the landing page, that propose to Login or register.

We also can see that we can switch between languages (3), we can login(4) and we can register (5).



Travel.

Start your adventure now.

Login to SocialPackers

Roadmap - Plan your trip around the world.

Tips - Find that secret place.

Projects. Start digging to find your project.

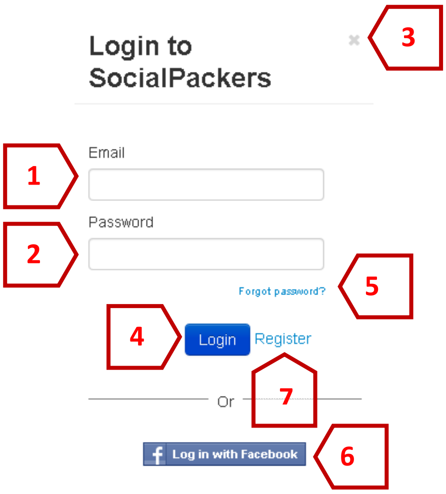
Register now

(1) If we click to Tips we go to the tip section. Point 5.1.2 of this document.

(2) If we click to Projects we go to the projects section. Point 5.1.3 of this document.

(4) (5) If we click to Login or Register we can see the following popups.

Login popup



We have to inputs to introduce the email(1) and the password(2).

We are going to close the popup if we click to the X(3) or if we click everywhere else outside the window.

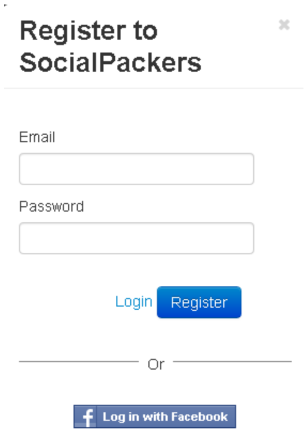
If we click to Login(4), we submit the form to check if the credentials entered are correct.

If we click to Forgot password?(5) we will be redirected to a page that will help us to create a new password.

If we click to Log in with Facebook(6), a popup of Facebook is raised and the credentials are going to be validated by Facebook.

If we click to Register(7), we change the popup for the following.

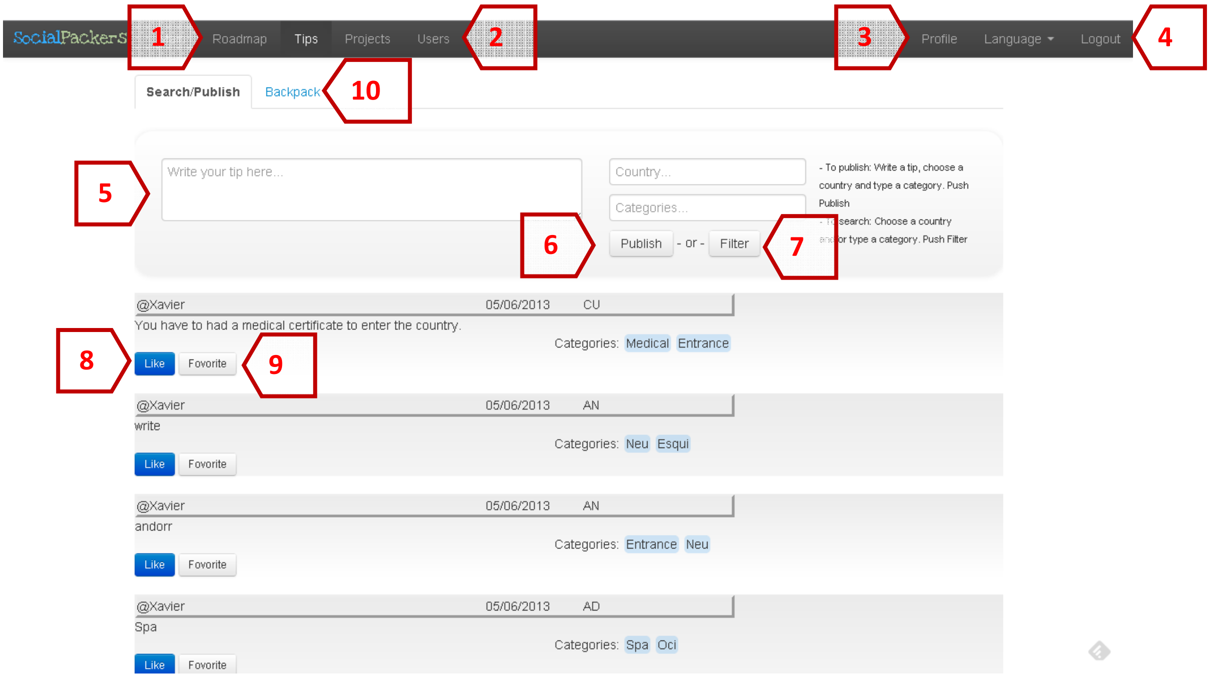
Register popup



Here like in the Login popup the behaviour is the same.

5.2.3.2. Tips page

Only for the record, here we can see the Tips section from a logged user because if we look at the header menu, it appears Roadmap(1), Users(2), Profile(3), Logout(4)



- To publish: Write a tip, choose a country and type a category. Push Publish
- To search: Choose a country and/or type a category. Push Filter

@Xavier 05/06/2013 CU
You have to had a medical certificate to enter the country.
Categories: Medical Entrance

@Xavier 05/06/2013 AN
write
Categories: Neu Esqui

@Xavier 05/06/2013 AN
andorr
Categories: Entrance Neu

@Xavier 05/06/2013 AD
Spa
Categories: Spa Oci

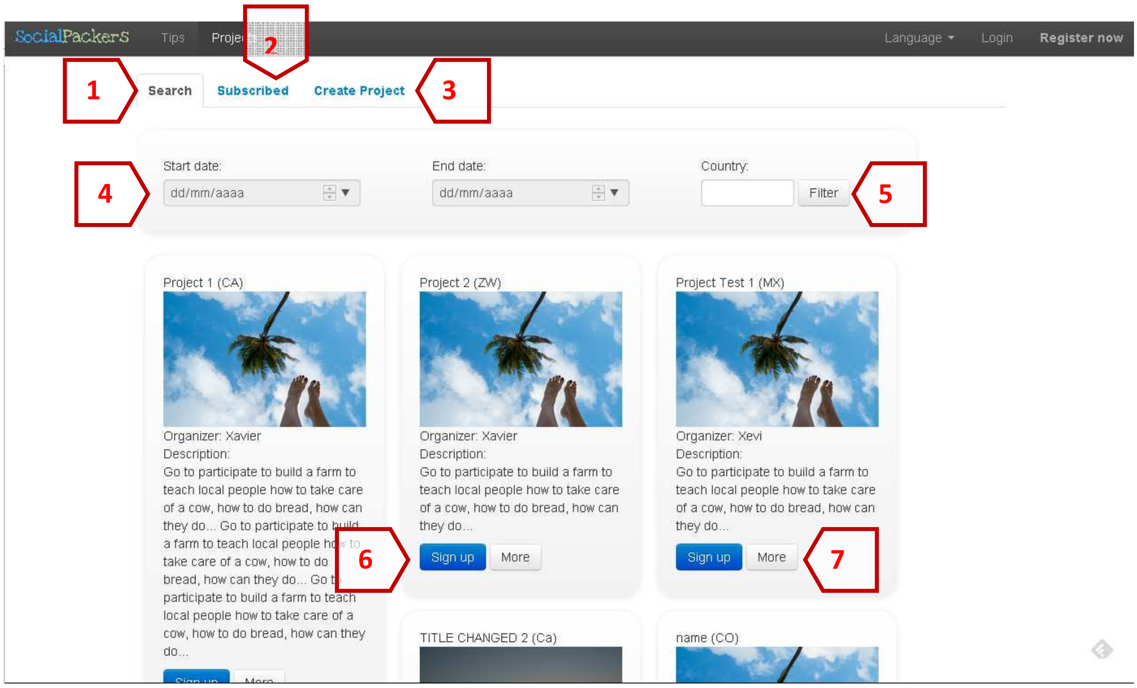
We can publish a tip writing on the big box(5) and tagging it with the country and categories we like. Then press the Publish button(6) to submit it.

If we want to search what we have to do is type a country or categories separated by semicolon and push the Filter button (7).

We also could click to the Like button (8) to make this tip more visible for users or/and click to Favourite button(9) to add the item in the backpack(10).

5.2.3.3. Project page

We can find three tabs, the Search(1) tab to search and dig, the Subscribed(2) tab to see the projects we have signed up and approved and the Create Project(3) tab to create projects.



Project 1 (CA)

Organizer: Xavier
Description:
Go to participate to build a farm to teach local people how to take care of a cow, how to do bread, how can they do... Go to participate to build a farm to teach local people how to take care of a cow, how to do bread, how can they do... Go to participate to build a farm to teach local people how to take care of a cow, how to do bread, how can they do...

Project 2 (ZW)

Organizer: Xavier
Description:
Go to participate to build a farm to teach local people how to take care of a cow, how to do bread, how can they do...

Project Test 1 (MX)

Organizer: Xevi
Description:
Go to participate to build a farm to teach local people how to take care of a cow, how to do bread, how can they do...

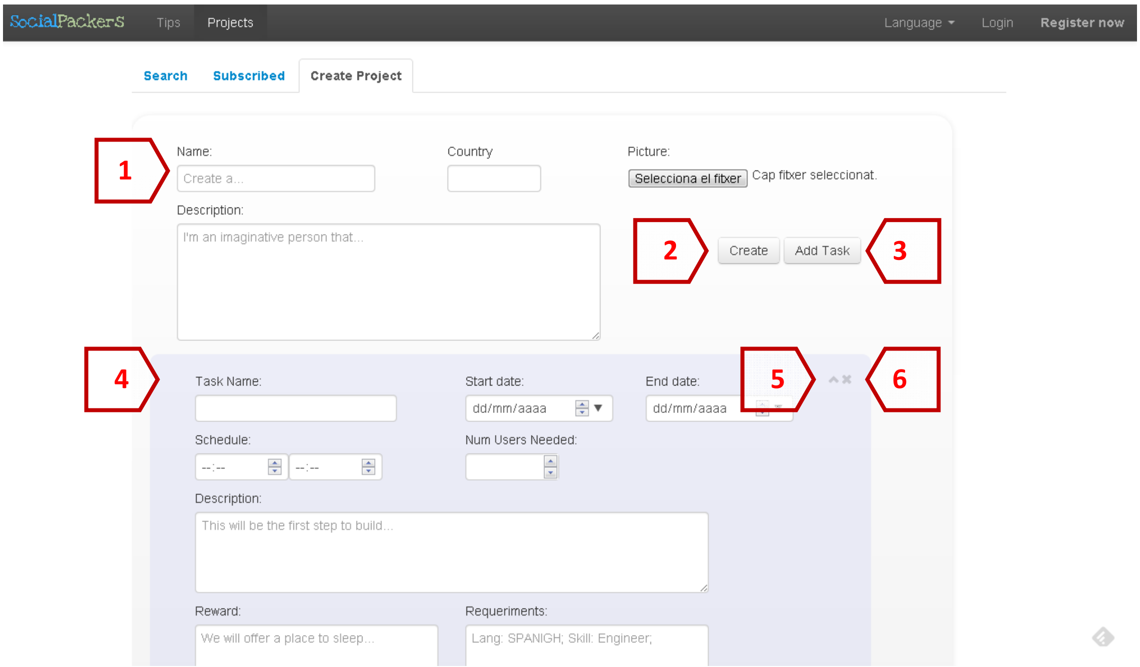
TITLE CHANGED 2 (Ca)

name (CO)

If we want to search we have to introduce the parameters(4) we desire and click to Filter button(5)

We can then, from the list of projects, select one to see More(7) or Sign up (6) directly.

Create Project tab



Name: Create a...

Country

Picture: Cap fitxer seleccionat.

Description: I'm an imaginative person that...

Task Name:

Start date: dd/mm/aaaa

End date: dd/mm/aaaa

Schedule: --:-- --:--

Num Users Needed:

Description: This will be the first step to build...

Reward: We will offer a place to sleep...

Requeriments: Lang: SPANIGH; Skill: Engineer;

Here we can create a project with its subtasks. To do so, we have to fill the inputs(1) of the project. Then we can press the Create button(2) or add a Task (3) and fill the task info(4), visible for the blue background, and then click to Create button(2).

If we have lots of tasks we can collapse each task clicking the arrow(5).

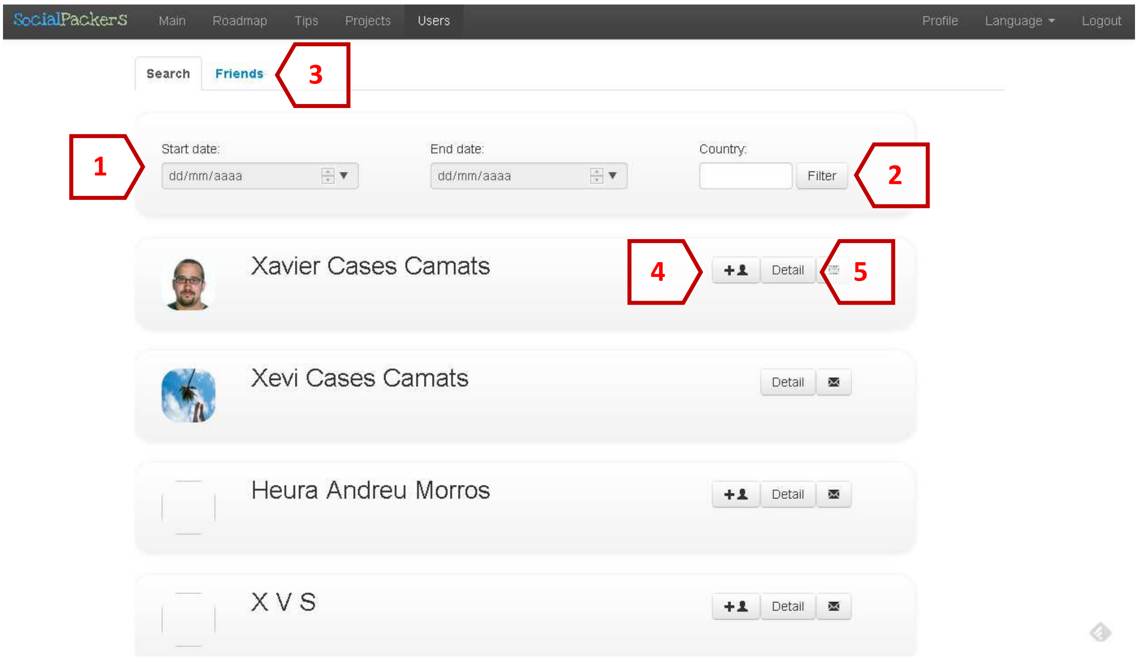
We also can delete a task clicking the x(6).

5.2.3.4. Users page

In the user’s page, we have to opportunity to connect with other people. We can search people filtering for some criteria(1) by writing in the inputs and clicking the Filter button(2).

We also have a Friends tab(3) with a list of the people we are connected already.

To connect with new people we can press de “plus person” button(4) or go into detail in Detail button(5) and click “plus person” button inside.

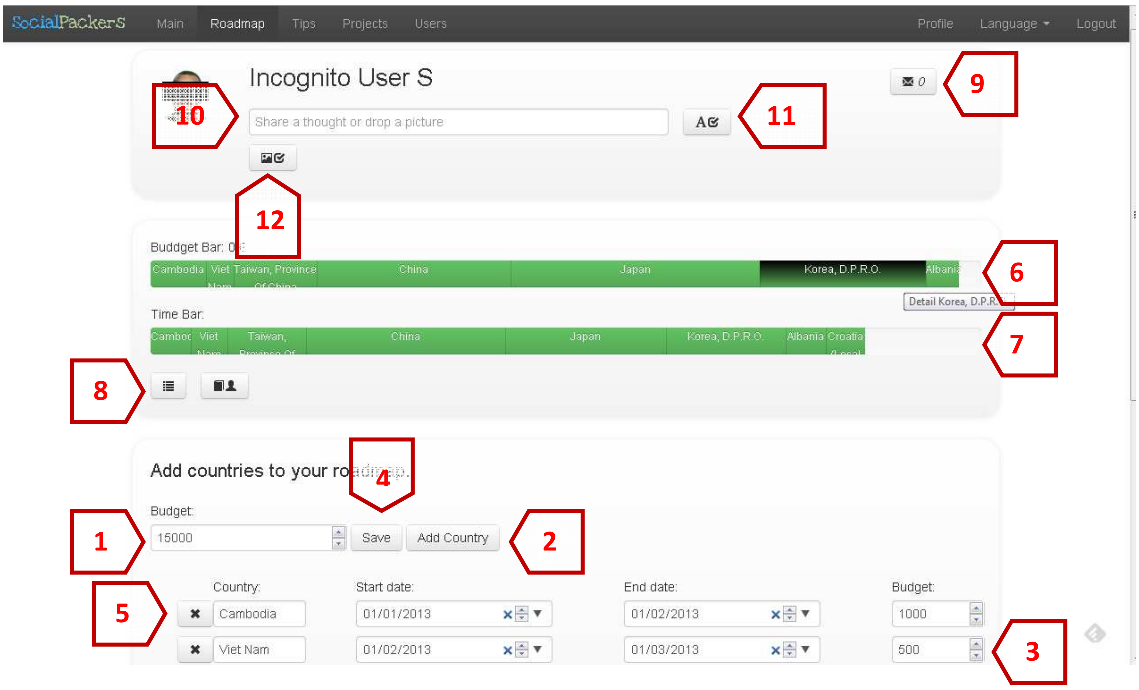


5.2.3.5. Roadmap page

Meant for create the roadmap of a live-time trip it allows us to add countries as a destiny.

We can add a budget(1). We can Add Country(2) that it will aggregate a new row(3) and in there we will define the destiny country and the dates/budget we want. Then we can Click to the Save button(4). If we want to delete a country we only have to click to the X button(5) of the row, and click the Save button later(4).

All that information is going to be reflected directly on the budget bar(6), where the width of a country represent a % spent on the country of the total budget and in the time bar(7) represents the % of time spent against the total.



Incognito User S

Share a thought or drop a picture

Buddget Bar: 0

Cambodia | Viet Nam | Taiwan, Province Of China | China | Japan | Korea, D.P.R.O. | Albania

Time Bar:

Cambodia | Viet Nam | Taiwan, Province Of | China | Japan | Korea, D.P.R.O. | Albania | Croatia

Add countries to your roadmap.

Budget: 15000

Country:	Start date:	End date:	Budget:
Cambodia	01/01/2013	01/02/2013	1000
Viet Nam	01/02/2013	01/03/2013	500

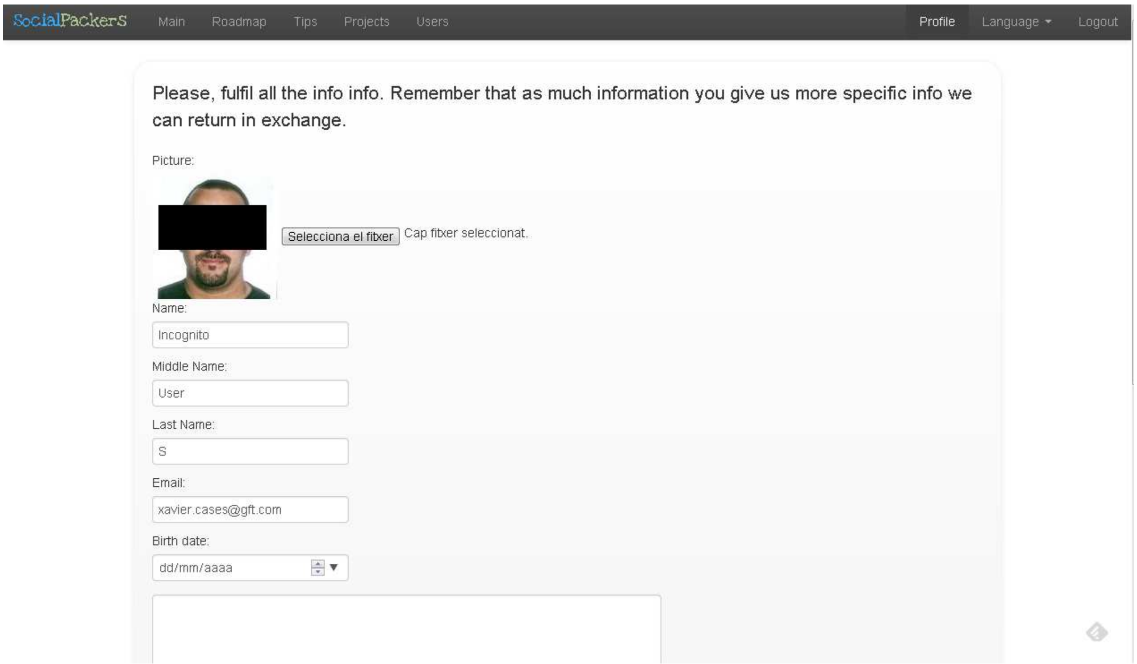
We also can hide and show the detail of countries by clicking in the “List” button(8).

In the top of the page we have three functionalities more, manage friendship requests(9), as we intend to centralize the user activity on the roadmap.

We also can add some info in the input (10) and click the submit button (11) to set a status that will be shown at bottom with all other activities.

If we want to aggregate images we will have to drag & drop it to the submit button (11) and click to the submit button (12).

5.2.3.6. Profile page



Please, fulfil all the info info. Remember that as much information you give us more specific info we can return in exchange.

Picture:

[redacted]

Cap fitxer seleccionat.

Name: Incognito

Middle Name: User

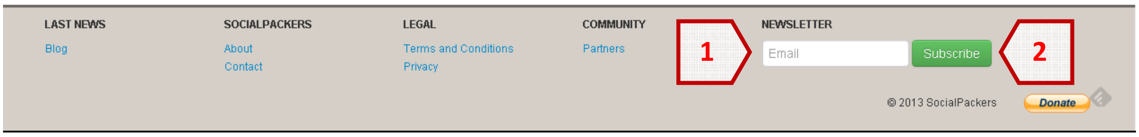
Last Name: S

Email: xavier.cases@gft.com

Birth date: dd/mm/aaaa

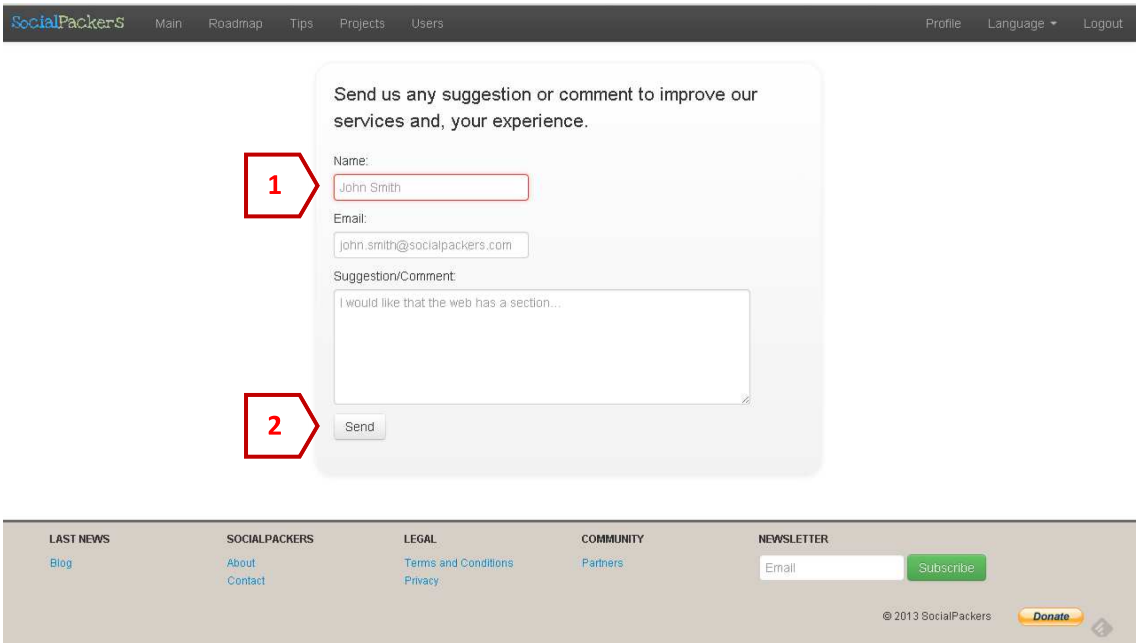
Here in the profile section we can modify the basic data, and then click to the Modify button on top of the page.

5.2.3.7. Footer



Here we can see the footer where we can see the static pages like Terms and Conditions, Privacy, Partners, but also Contact, Blog and even we can subscribe to the newsletter typing our email(1) and clicking the Subscribe button(2) or if we feel good we can make a donation.

5.2.3.8. Contact



Send us any suggestion or comment to improve our services and, your experience.

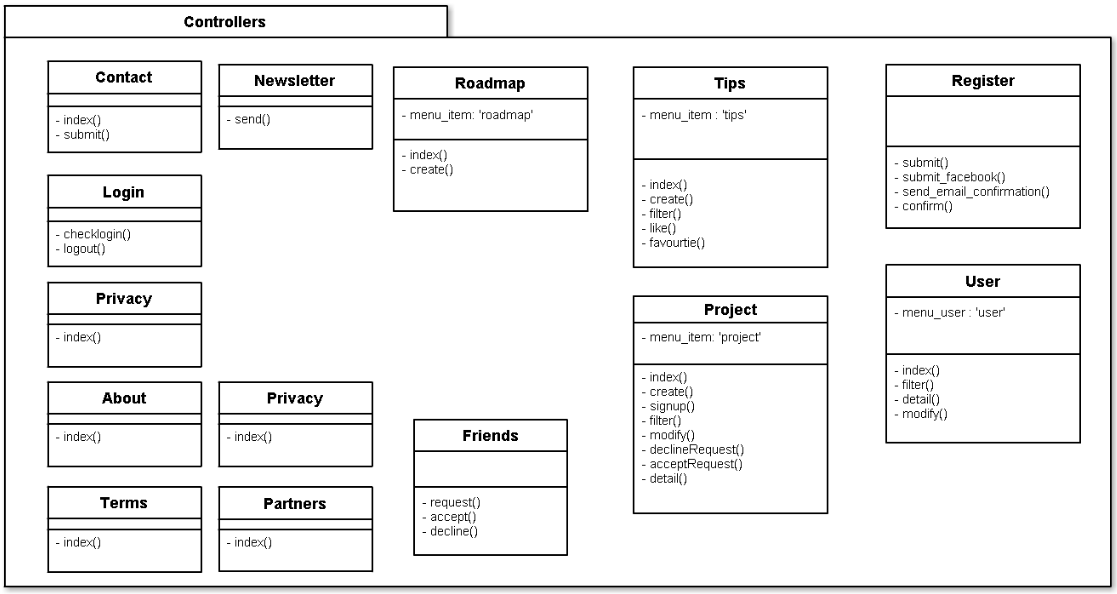
Name: John Smith

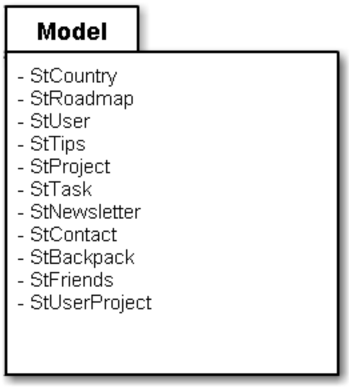
Email: john.smith@socialpackers.com

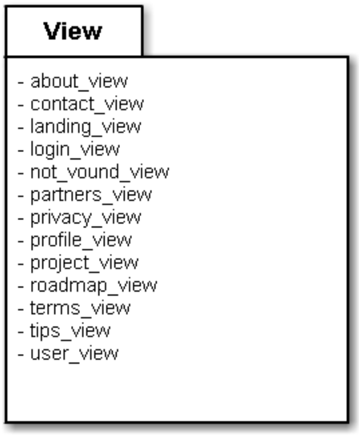
Suggestion/Comment: I would like that the web has a section...

In the contact section, we can Send a suggestion or a comment to the web application administrator adding the corresponding information(1) and clicking in Send button(2).

5.2.4. Model layer design



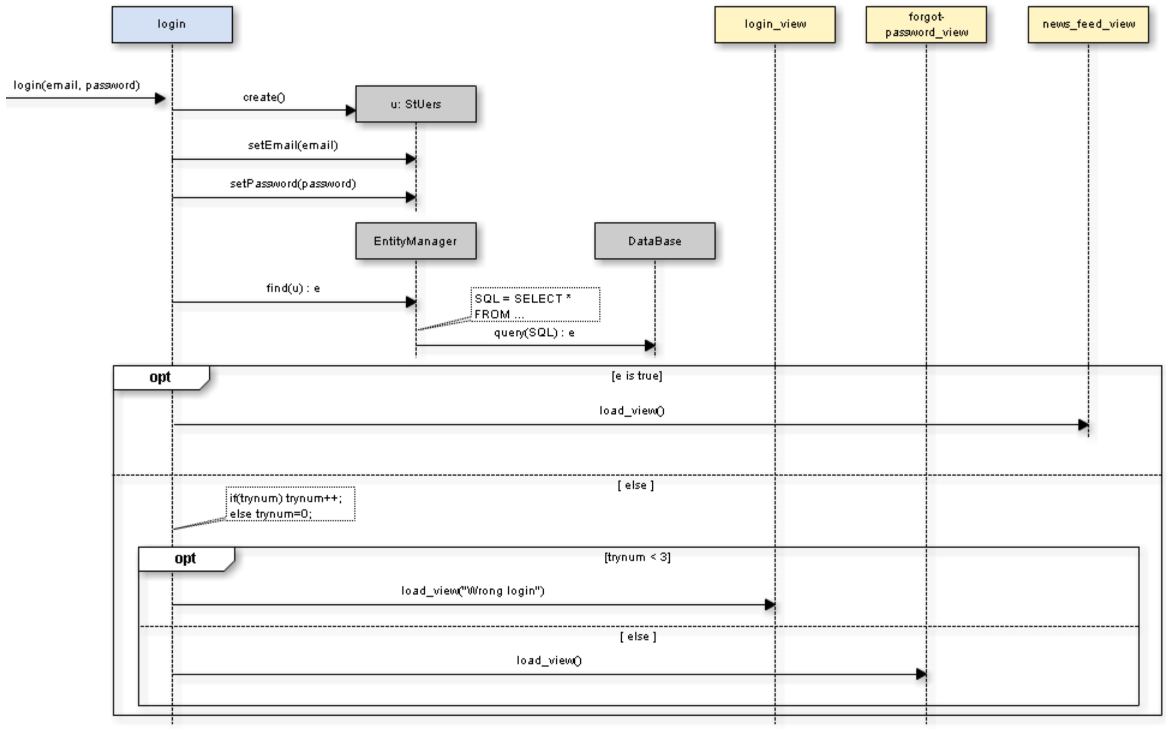




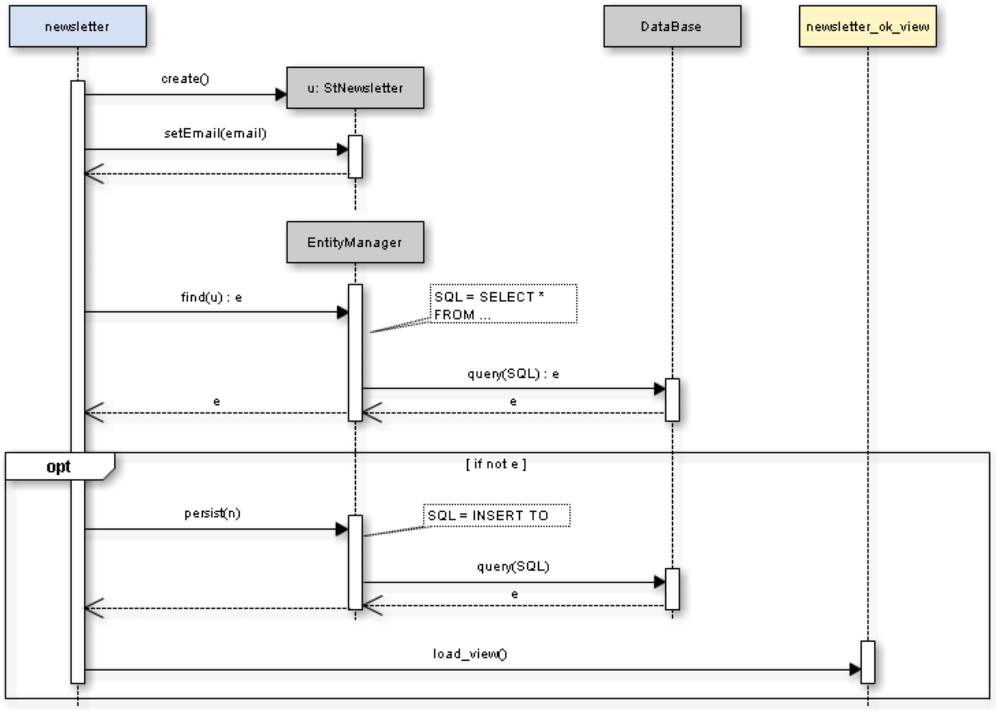
5.2.3.8. Sequence diagrams

Here we are going to put the most relevant sequence diagrams.

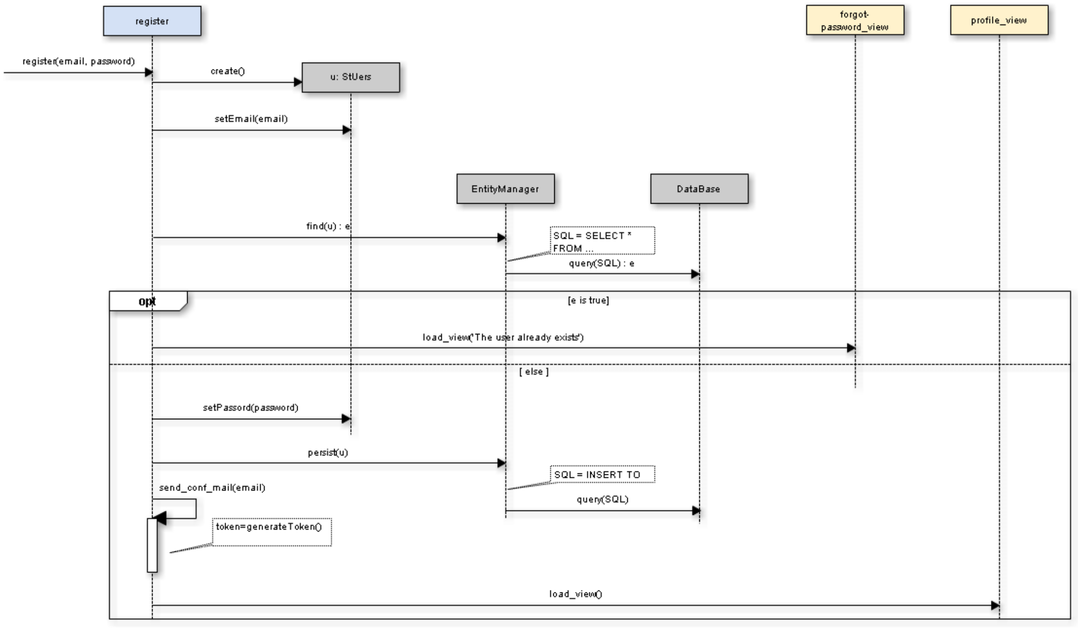
Login



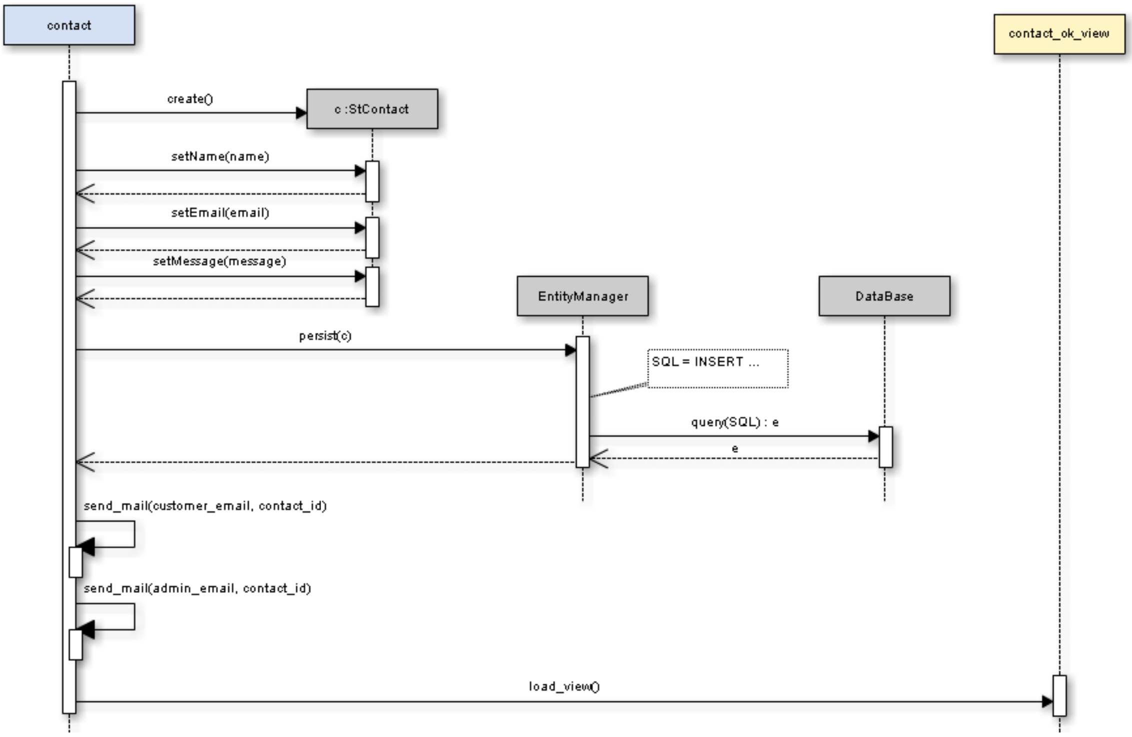
Subscribe to Newsletter



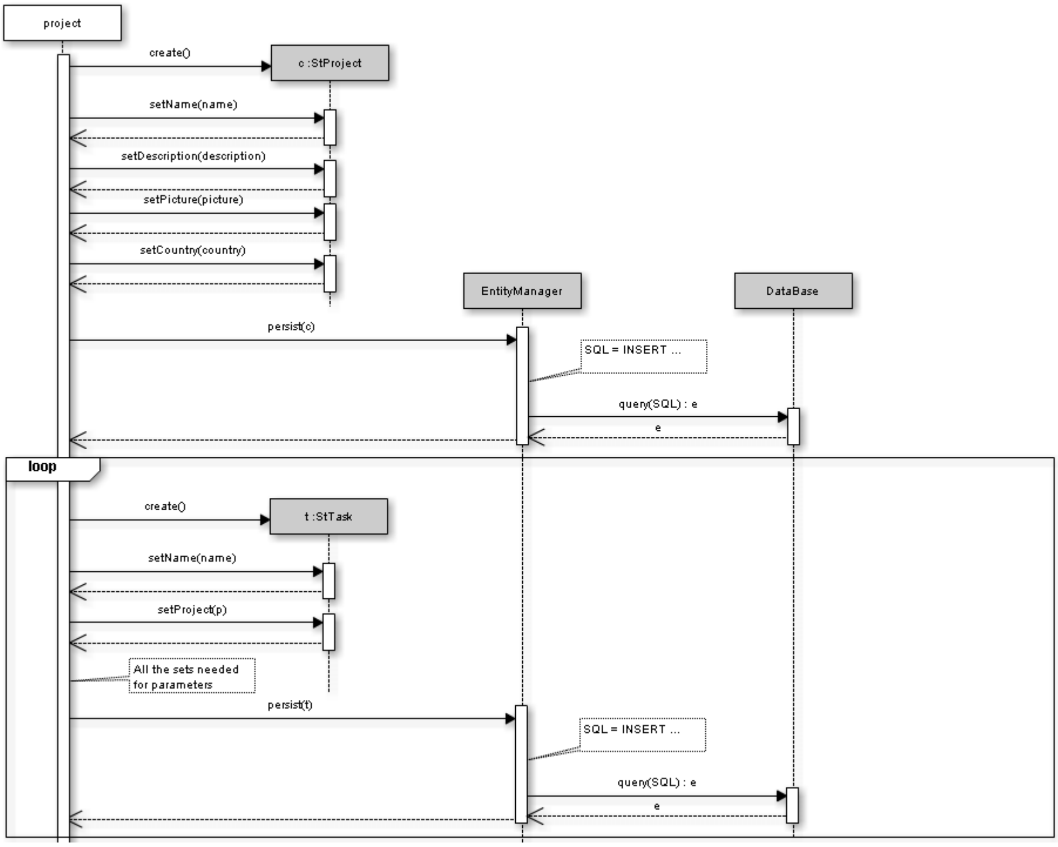
Register



Contact

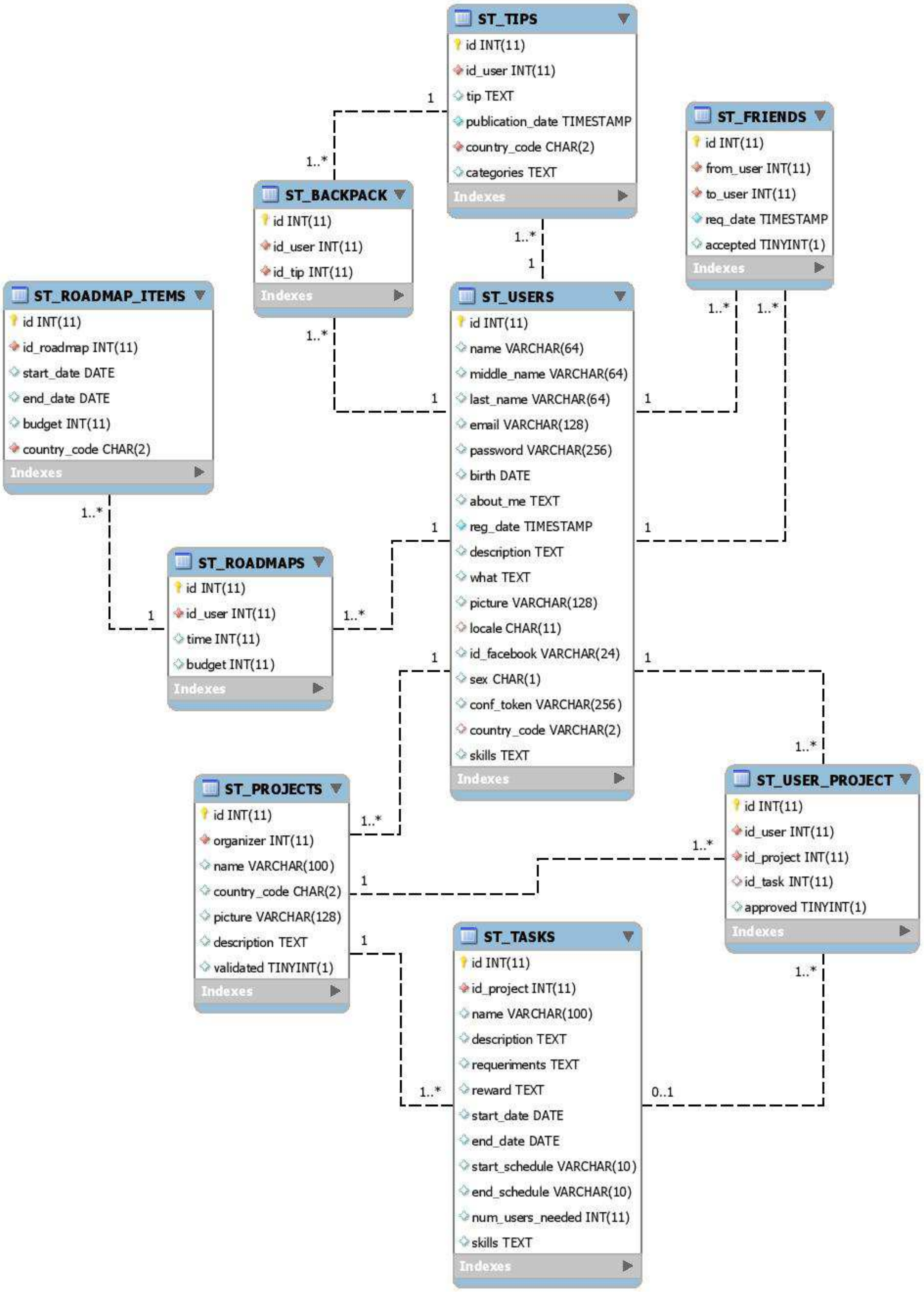


Create Project



5.2.5. Data layer design

To do the database design we normalized some relationships make the model as follows:



6. Implementation

6.1. Development environment

6.1.1. Installation

We used the WAMP Server 2.2 composed by an Apache 2.2.22, a PHP 5.3 and a MySQL 5.5.24 over a Windows 7 Ultimate.

We added the *rewrite_module* to the apache to allow us to modify the URL of the web for SEO purposes.

6.1.2. Development

What we will have on our computer to develop?

- A WAMP Server installed and working.
- A Google Chrome browser as it has become the most used browser over the world.
    - We can use the "Incognito mode" to see always a fresh website. This mode does not have cache and does not save cookies at the end of each session.
- Development Tools / Firebug
    - Essential tool to debug JavaScript and to see all the calls to the server.
- PSPad
    - Text editor for all the files.
    - It can be used a most powerful IDE, that is a developer decision.

7. Conclusions

Working in a team, discussing how it could be, how it **will** be is a very strong and satisfactory emotion; and more if after some time (work included) it is done and functional.

Working for a social cause motivates me a lot. It is a powerful way to make give the maximum of a person. Think that you can help with a little, little, little thing in society is gratefully.

I have to say I worked a lot and I am not done. I will continue with more background, experience and with new knowledge acquired.

The general feedback is positive, but I realised how hard can be learn and do something or even worst plan and estimate it. The part I am referring is to the view part. We now that nowadays is almost more important how it looks than how it works, but not only that, if you put a component in a place is not meant to be, it can be understood in a wrong way. So yes, I learnt to give more value to the view part, it is very important.

I have had to dedicate more than estimated time to the front end part due to mentioned before, I know that there are parts quite pretty, but others looks awful. So, one of the next steps will be take care of that.

And that is not all; we are already planning some new functionalities, such as:

- Let people create a test roadmap to show the powerful the application can be.
- Improve security to make feel user save.
- Make the presentation even more dynamic.

- Add functionalities and more power to the roadmap.
- Make interaction between users more comfortable and easy.
- And lots more...