Open Source Platform for Subjective Polling of Urban Environments: OpenUrbanOpticon

Bernat Orell Vicens

Dr. Daniele Quercia & Dr. Daniel Villatoro

Enginyeria Informàtica - Universitat de Barcelona

2012-2013

Al meu padrí Bernat

Acknowledgements

Poder dur aquest projecte a bon port no ha estat tasca fàcil, per això he d'agrair primerament l'ajuda i la paciència que ha tingut el meu tutor Daniel Villatoro perquè estic segur que no és tasca fàcil. Per altra banda, vull agrair al treball fet pel meu altre tutor Daniele Quercia, per haver proposat un projecte interessant i per el caire científic que li ha donat al projecte.

També vull agrair l'ànim amb que hem saluden cada vegada que hem truquen des de Mallorca els meus padrins Miquel i Margalida i al meu padrí Bernat, que ens ha deixat recentment, i que estic segur de que estaria orgullós dels seus néts. Al meu pare, per el seu seny, que sempre m'ha donat bons consells i tot un referent i a la meva mare, per que sempre m'ha animat a continuar i a torbar solucions a tots els problemes que han anat sortint. També agrair al suport de la meva germana, que com jo acaba una etapa de la seva vida on junts hem compartit bons i no tant bons moments durant tot aquest temps.

Finalment vull agrair als meus amics Alberto, José Lluis i David que han ajudat a fer aquests cursos d'Enginyeria Informàtica més agradables i entretinguts.

1. Intro

One of the problems that the world has to face in the next years is the fact that more and more people are moving to cities. To accommodate these people in the actual cities will be a challenge and the governors have to work on it. There are a lot of different ways to investigate how people is interacting with the city to make it better but one of them is the recognizability map that everyone have on their mind of the city. This will be very important in the future, how much they know their city and what things can relate with the places. For example, it could be interesting for an administrator of a city to know which areas need to increase their visibility or for a manager that wants to know which are the well-known streets or places of a city to spread his business.

The web page Urban Opticon runs a game which permits to know through a crowdsourced gamification web application, the well-known points of London. Images of Google Street View are shown and users have to point the closest borough or tube station. But this solution has been tailored for the city of London, with all their specific characteristics such as the boroughs or tube stations. Because of these constraints, and despite the interest of the result obtained using this platform, the developed code is not transportable to any city.

Therefore, we decide to extend this previous basic project into a new project named Open Urban Opticon that allows to deploy a server with any city of the world easily for everyone. Our main goals are to make a friendly interface easy to manage that allows to make new studies without writing a single line of code.

Our specific objectives for this project are:

- Develop a web application to help in the task of the subjective polling of the urban environments.
- Learn to use a web framework like Django.
- Interact with external APIs such as Google Street View, Google Places and OpenLayers.
- Provide a friendly administration website for those experiment setters to configure their experiment with a few steps.

The sections that will be found in this document are:

- State of art, where the most interesting crowdsourcing and smartcity projects will be explained and the technologies that they use.

- Problem description, here it is described which are the problems that have to for our general approach to the Urban Opticon.

- Use cases, the interaction with the users.

- System structure, here it is explained the different technologies that have been used and why they engage with our project.

- Technological challenges, this section contains the most challenging problems and their solutions.

- Future work, in this section is an approach to how the Open Urban Opticon could evolve in the future.

- Conclusions, in this section we present the main conclusions and lessons learnt from the project.

2. State of Art

This sections is divided in three different subsections depending on the types of projects that are explained which are related with Open Urban Opticon. The first are the crowdsourcing projects. The crowdsourcing projects can be only understood with the participation of the people. The projects are based in the data that people give when they involve. In the second section the smartcities projects can be found. Smartcities projects have a clear objective that it is make habitability of the urban environments better in different ways like energy saving, public transport, etc. In the last section we have the crowdsourcing/smartcities projects, this projects use the data that users give and this data is used to improve the urban environment.

2.1. Crowdsourcing

2.1.1. Introduction

Crowdsourcing is a process that involves outsourcing tasks to a group of indeterminate people. This process can occur offline but it is more frequent on Internet environments and it makes crowdsourcing much easier. Internet helps in many ways like raising funds to support a cause or organization.

Crowdsourcing can be divided by types depending on the tasks:

- Crowdvoting: it consists in judge certain things that users have uploaded or product owners have published in order to know the crowd opinion. For example: The Hype Machine, Soundcloud, Pinterest, Tumblr, etc.

- Crowdsourcing creative work: it involves different users to design a certain product. For example: 99designs, GeniusRocket, etc.

- Crowdfunding: it exposes a project that need financial help to look for investors, and depending on the investment, the company will give a
reward. For example: Kickstarter, Verkami, etc.

- “Wisdom of the crowd”: it is another type of crowdsourcing that collects large amounts of information and aggregates them to gain a complete and accurate picture of a topic, based on the idea that a group of people is on average more intelligent than an individual. For example: Wikipedia, Yahoo! Answers, etc.

- Microwork: it consists in pay a very low amount of money for do a little task. The tasks can be done for everyone but only the best will be rewarded. For example: Amazon Mechanical Turk, k68.cn, etc.

- Inducement prize contests: it consists in an open fast contest that needs some kind of crowdsourcing in order to win a big prize. For example: DARPA, Netflix or Google experiments, etc.

- Implicit crowdsourcing: Implicit crowdsourcing is less obvious because users do not necessarily know they are contributing. Implicit crowdsourcing involves users doing another task entirely where a third party gains information for another topic based on the users' actions. For example: reCaptcha, Commutio, etc.

2.1.2. The projects

Analyzing this projects helps to have an approximation of how crowdsourcing projects works and which are the bases. This knowledge will help us to improve our project.

2.1.2.1. Kickstarter

Kickstarter is a crowdfunding project that allow creators to find investors that want to make a investment for a reward.

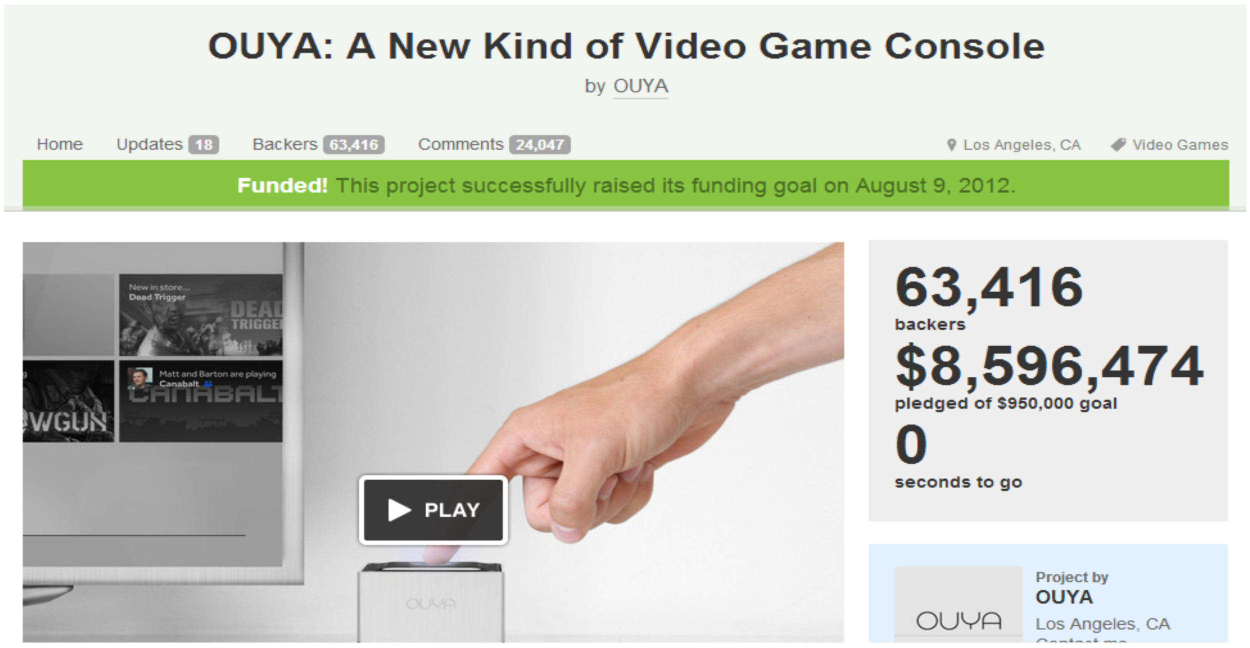
2.1.2.1.1. The goal

The goal of Kickstarter it is to help creators with a good projects and a need of investment to find people who wants to help them with a payback. That helps to open the whole world to the big ideas that maybe without a global research of investors could not reach the amount of money necessary to take off.

2.1.2.1.2. User experience

Kickstarter have two main ways, the investor and the project creator. As an investor the user can navigate the website looking for cool projects or just go through to a project that he has heard.

Ouya is a well known project and a good example. Ouya is a new video game console built on android.



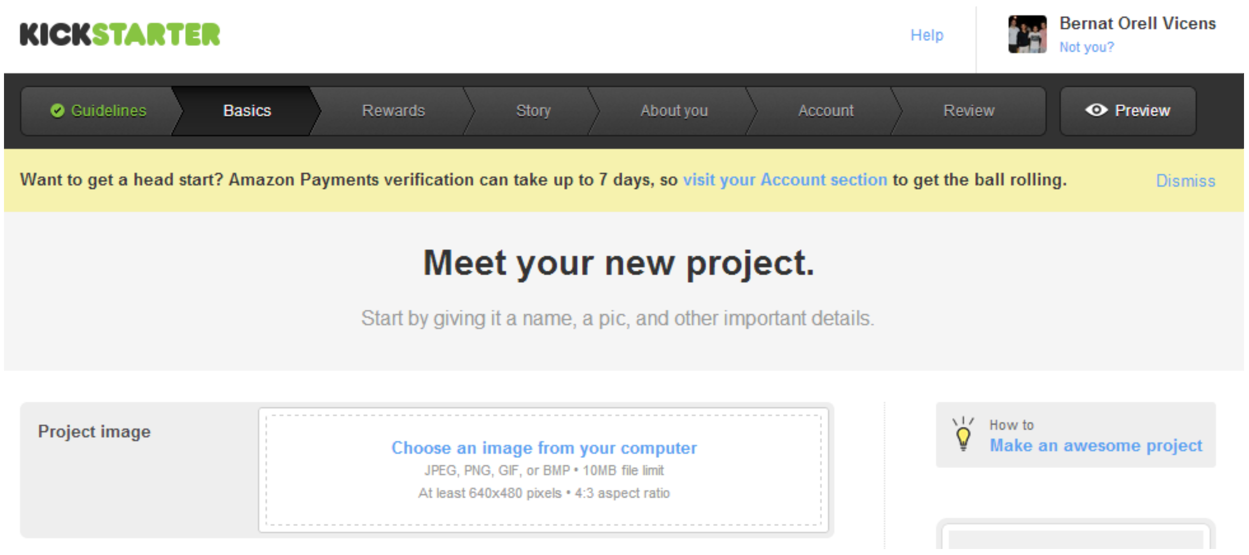
(Figure 1. Kickstarter screenshot)

A project have a goal price where they are pledged to release their product and give the rewards to their investors. In this case the goal price is 950,000$ and these are the rewards depending on the investment.

There are a description of the project and some videos. The user can pay the pledge that he wants and the money will be paid only when they reach 100% of the goal price.

On the other hand the users have the creator way. Firstly the user has to register in Kickstarter and they give some guidelines like the types of project allowed, the rewards, etc.

Then the user can make his project site following the instructions.


KICKSTARTER

Help

Bernat Orell Vicens
Not you?

Guidelines | Basics | Rewards | Story | About you | Account | Review | Preview

Want to get a head start? Amazon Payments verification can take up to 7 days, so visit your Account section to get the ball rolling. Dismiss

Meet your new project.

Start by giving it a name, a pic, and other important details.

Project image

Choose an image from your computer
JPEG, PNG, GIF, or BMP • 10MB file limit
At least 640x480 pixels • 4:3 aspect ratio

How to
Make an awesome project

(Figure 2. Kickstarter screenshot 2)

Finally the user releases his page and he can start looking for contributors.

2.1.2.1.3. Criticism

The most common Kickstarter criticism is that if the creator know that his project will have support before start maybe he does not accomplish all the objectives and he does not release a well polished project. That does not affect our project because it does not have a final state, the more answers it have, the better will be the results.

As many as 75 percent of Kickstarter projects do not deliver on time, according to a recent University of Pennsylvania study, and some never deliver at all. In our project the users do not have anything to deliver, first they will make their polls and they wait for the results.

2.1.2.1.3. Technologies

In general terms Kickstarter uses a web interface and everything is gathered in a central server.

As the web page builtwith and a job offer, Kickstarter uses Ruby on Rails. RoR is a very complete framework for Ruby that have everything that a web-developer needs, management of databases, create templates, gather information, etc. The disadvantage of RoR is Ruby, it have less documentation and community than PHP and is less intuitive than other languages like Python.

They use jQuery too, that it 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.

For the payments they use Amazon Flexible Payment Service that allows the users to pay with their Amazon account.

2.1.2.2. Amazon Mechanical Turk

Amazon Mechanical Turk is a marketplace for work that requires human intelligence and it is the reference platform to distribute retributed tasks in order to obtain crowdsourced results. The Mechanical Turk service gives businesses access to a diverse, on-demand, scalable workforce and gives Workers a selection of thousands of tasks to complete whenever it is convenient.

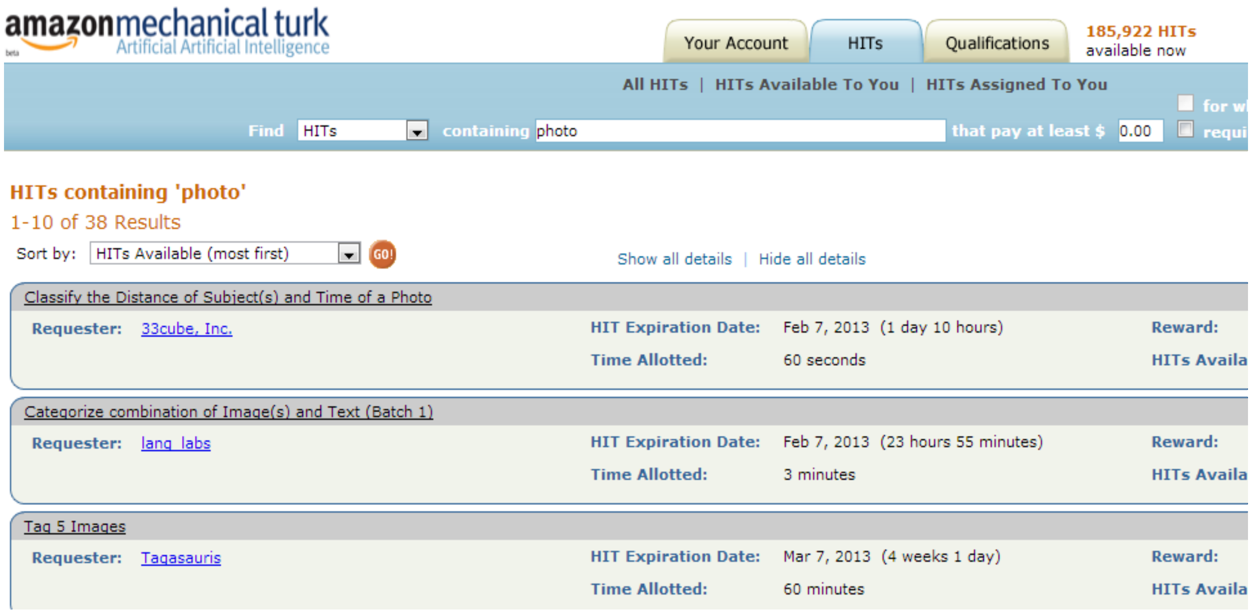
2.1.2.2.1. The goal

Amazon Mechanical Turk is based on the idea that there are still many things that human beings can do much more effectively than computers, such as identifying objects in a photo or video, performing data de-duplication, transcribing audio recordings, or researching data details. Traditionally, tasks like this have been accomplished by hiring a large temporary workforce (which is time consuming, expensive, and difficult to scale) or have gone undone. This tasks are named HIT (Human Intelligence Task).

The goal of Amazon Mechanical Turk is help people who want to work and earn a reward and work requesters that are disposed to pay something for get their work done.

2.1.2.2.2. User experience

If the user want to make a HIT and earn some money he can search something where he is qualified.


amazon mechanical turk
Artificial Artificial Intelligence
Your Account
HITs
Qualifications
185,922 HITs available now
All HITs | HITs Available To You | HITs Assigned To You
Find HITs containing photo that pay at least $ 0.00
for w
requi
Sort by: HITs Available (most first) GO!
Show all details | Hide all details

HITs containing 'photo'

1-10 of 38 Results

Classify the Distance of Subject(s) and Time of a Photo

Requester: 33cube, Inc. HIT Expiration Date: Feb 7, 2013 (1 day 10 hours) Reward:

Time Allotted: 60 seconds HITs Availa

Categorize combination of Image(s) and Text (Batch 1)

Requester: lang labs HIT Expiration Date: Feb 7, 2013 (23 hours 55 minutes) Reward:

Time Allotted: 3 minutes HITs Availa

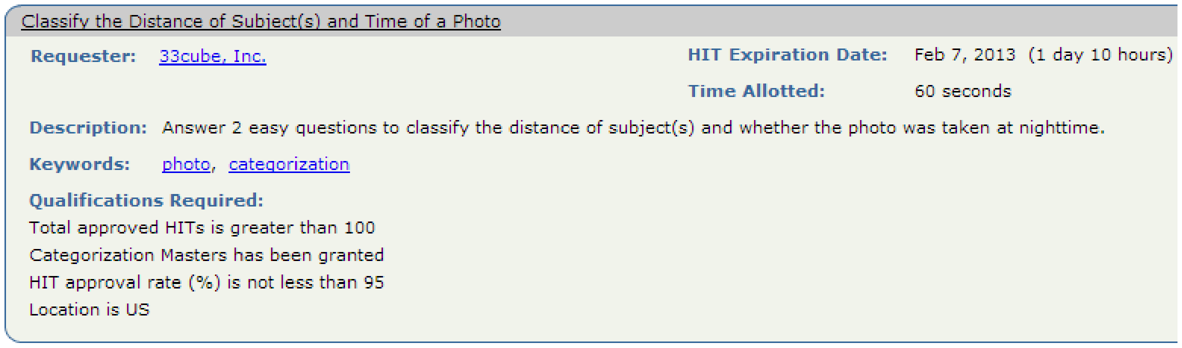
Tag 5 Images

Requester: Tagasauris HIT Expiration Date: Mar 7, 2013 (4 weeks 1 day) Reward:

Time Allotted: 60 minutes HITs Availa

(Figure 3. Amazon Mechanical Turk screenshot)

He can click on a HIT and see the description.



Classify the Distance of Subject(s) and Time of a Photo

Requester: 33cube, Inc. HIT Expiration Date: Feb 7, 2013 (1 day 10 hours)

Time Allotted: 60 seconds

Description: Answer 2 easy questions to classify the distance of subject(s) and whether the photo was taken at nighttime.

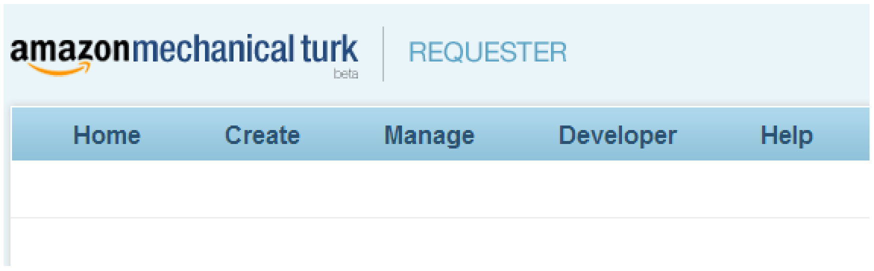
Keywords: photo, categorization

Qualifications Required:
Total approved HITs is greater than 100
Categorization Masters has been granted
HIT approval rate (%) is not less than 95
Location is US

(Figure 4. Amazon Mechanical Turk screenshot 2)

The user can see a preview mode and accept the HIT.

As a HIT requester the user can go to requester website and make his own HIT with a creator.



(Figure 5. Amazon Mechanical Turk screenshot 3)

2.1.2.2.3. Criticism

The main criticism is that in our project we do not want to pay for do a task. Developers should find a way to reward the users with something more useful than a few cents, like information or making them to have a good time.

Because HITs are typically simple, repetitive tasks and users are paid often only a few cents to complete them, and that can be considered some kind of human exploit. In our project, the users will play until they get tired, they do not need to be playing a certain time.

Because workers are paid as contractors rather than employees, requesters do not have to file forms for, nor pay payroll taxes, and they avoid laws regarding minimum wage, overtime, and workers compensation. Workers, though, must report their income as self-employment income. In addition, some requesters have taken advantage of workers by having them do the tasks, then rejecting their submissions in order to avoid paying.

2.1.2.2.4. Technology

AMT has SDK for Java, .NET, Perl and php and Ruby libraries. It can be used with a command line interface provided by Amazon that only uses a text file named .input, a .XML and a .properties.

There are another UI that makes the process easier to do for non-programmers users but it is less personalizable. The process is more complex than that we need for our project but they are very related. It have a few tabs that helps the user how to create a project. That will be mostly what our project have to do for poll creators.

2.1.2.3. 99Designs

99Designs is website where business and individuals make contests in order to find a good design made by designers who want to participate.

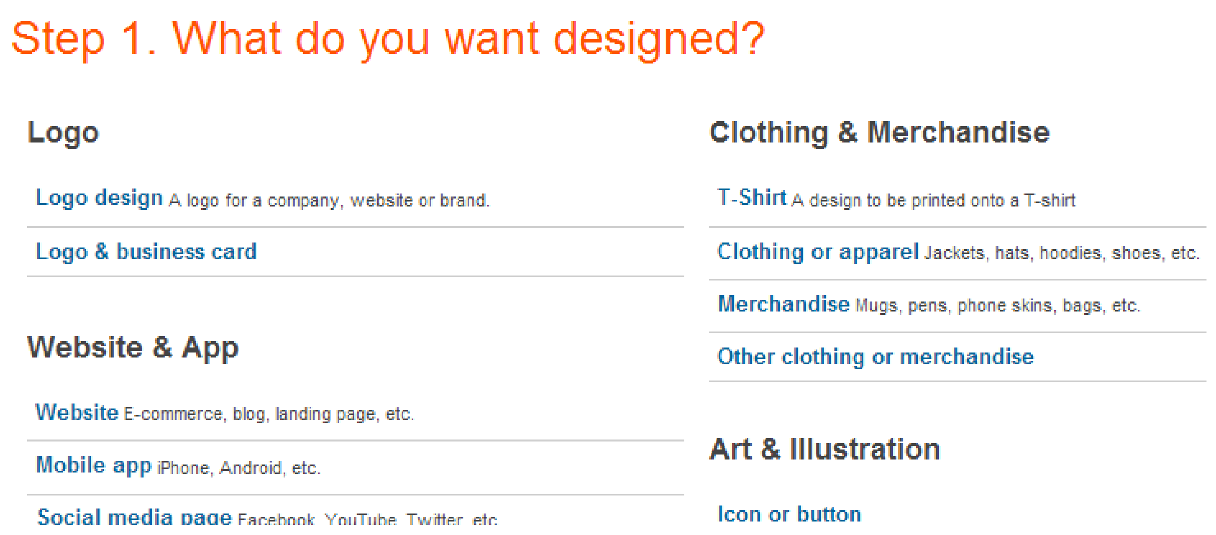
2.1.2.3.1. The goal

The goal of 99Designs is connect the design requesters and the designers. The business make a contest where they put a description of what they need and the designers can submit a design. The manager will pick a design a pay a prize for his work. Therefore, it is easy to see the crowdsourcing characteristics like open spectrum of users (from art designers to web developers) and the contest with prizes.

2.1.2.3.2. User experience

There are two ways to interact with the website. On the one hand there is the hoster of a “design contest”.

Firstly the user has to choose what he wants to be designed. he can choose a website.



Step 1. What do you want designed?

Logo

Logo design A logo for a company, website or brand.

Logo & business card

Website & App

Website E-commerce, blog, landing page, etc.

Mobile app iPhone, Android, etc.

Social media page Facebook, YouTube, Twitter, etc.

Clothing & Merchandise

T-Shirt A design to be printed onto a T-shirt

Clothing or apparel Jackets, hats, hoodies, shoes, etc.

Merchandise Mugs, pens, phone skins, bags, etc.

Other clothing or merchandise

Art & Illustration

Icon or button

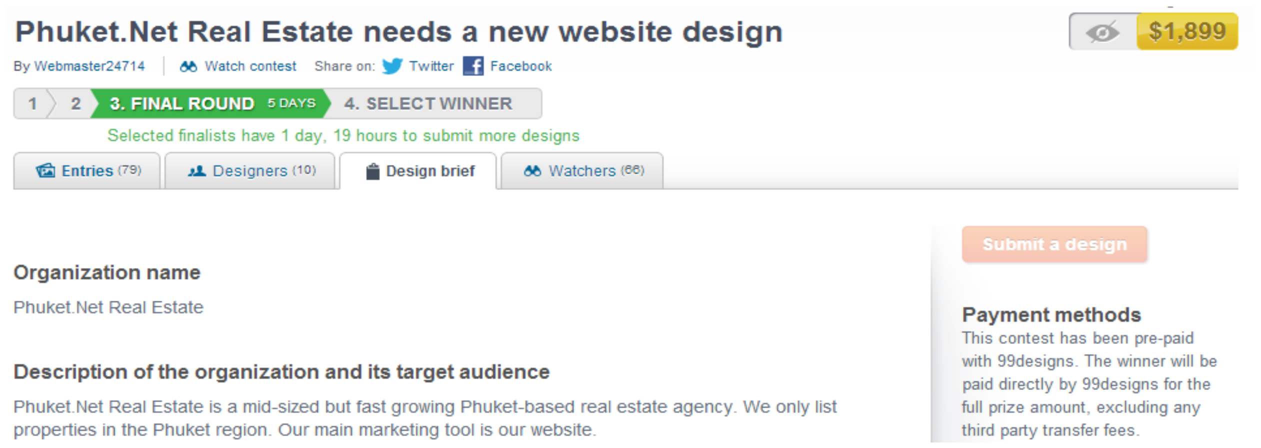
(Figure 6. 99Designs screenshot)

Then the user has to put a description of what he wants, the name, some sketches, etc.

The user selects the prize that wants to pay and finally launch the contest. If no one of the submit designs like the user will have his money back.

On the other hand the website has the designer way where the user can submit his designs. Firstly the user has the open contest and the user can filter for the keywords that want. He can see the reward and a little description it is displayed.

The user can enter in a contest. He will find a more accurate description and if are not hidden, the other projects submitted. The user can submit his own design too.


Phuket.Net Real Estate needs a new website design

By Webmaster24714 | Watch contest Share on: Twitter Facebook

1 2 3. FINAL ROUND 5 DAYS 4. SELECT WINNER

Selected finalists have 1 day, 19 hours to submit more designs

Entries (79) Designers (10) Design brief Watchers (66)

$1,899

Submit a design

Organization name

Phuket.Net Real Estate

Description of the organization and its target audience

Phuket.Net Real Estate is a mid-sized but fast growing Phuket-based real estate agency. We only list properties in the Phuket region. Our main marketing tool is our website.

Payment methods

This contest has been pre-paid with 99designs. The winner will be paid directly by 99designs for the full prize amount, excluding any third party transfer fees.

(Figure 7. 99Designs screenshot 2)

2.1.2.3.3. Criticism

Some good designers will work for nothing if their project is rejected and this is very frustrating especially for large design works like a website. This kind of business are putting into jeopardy the business model of classical design studies because in web sites like 99designs the prizes are more competitive and it works like a contract, the design requesters do not know if the designer that is behind a design is exploited or he have a good work environment. Another criticism is the fixed prizes, every design have a different work behind and can be more or less expensive.

In our project will have two users, one who publish the study and the participants. For the publishers will be free to make a study and users will play the game and they will have a good time.

2.1.2.3.4. Technologies

The main server in 99designs runs with PHP. PHP is a scripting language designed for web development to produce dynamic web servers. It is free and there are a lot of documentation and forum with user support. PHP is loosely typed, which makes basic scripts much faster to develop with less attention to design. On the other hand it has some disadvantages like that is slow compared with other compiled languages like C and There are many ways to do one thing, and many cases where a function has ambiguous handling due to legacy support for PHP development history. PHP is a good option for our project because all of these features but it has good competitors as Ruby or Python.

For the storage they use a MySql database. Structured Query Language is a specific programming language designed for managing data in relational database management systems. SQL have a lot of documentation and it is widespread over the world of web-developing. The crashes can be well managed with these kinds of databases. The problems of SQL Databases come with the poor compatibility between them. Our project can have a SQL database too because is easy to manage with the frameworks.

With the transient data they use Memcached, mongoDB and Redis. Memcached is used to manage the RAM memory and helps to reduce the database query activities. MongoDB is an open source document-oriented database system that it is used for logs of errors and statistics. Redis is an open-source, networked, in-memory, key-value data store with optional durability. It captures per-user information about which features are enabled at any given time. Our project can use a mongoDB for logs but the project will not use it for main data because the database will not be very large and the data will not need a lot of updates.

2.2. Smartcities

2.2.1. Introduction

Smart city is a paradigm of the city that look to increase the competitiveness of cities with social and intellectual solutions above the hard infrastructures. Smart cities have some main points that try to improve: economy, citizen participation, mobility, environment, people, living and governance.

Smart cities are a world in expansion. By looking at the statistics more and more people are going to live in urban areas. By 2030, that urban footprint will expand by another 590,000 square miles to accommodate the more than 1.47 billion additional people expected to be living in the world's cities, according to the study, conducted by researchers from four U.S universities — Yale, Arizona State, Texas A&M, and Stanford[1]. City managers need some planification in order to absorb this big impact in wealth creation, construction, transports and buildings.

Data analysis, sensor technologies, and urban experiments will provide new insights into creating a data-driven approach to urban design and planning. To build the cities that the world needs, it is needed a scientific understanding of cities that considers our built environments and the people who inhabit them.

2.2.2. The projects

Our project is closely related with smartcities, for this reason we will analyze some of the most interesting projects that can be useful for our research. Most of the smartcity projects are related with construction and transport but almost all of them use Internet platform and computer

[1] A Meta-Analysis of Global Urban Land Expansion, Karen C. Seto mail, Michail Fragkias, Burak Güneralp, Michael K. Reilly

science to generate their results.

2.2.2.1. HoyRespiro

HoyRespiro is a Basque project that use environmental control networks existing along the city to display the less contaminated zone in the city for people who have lung diseases.

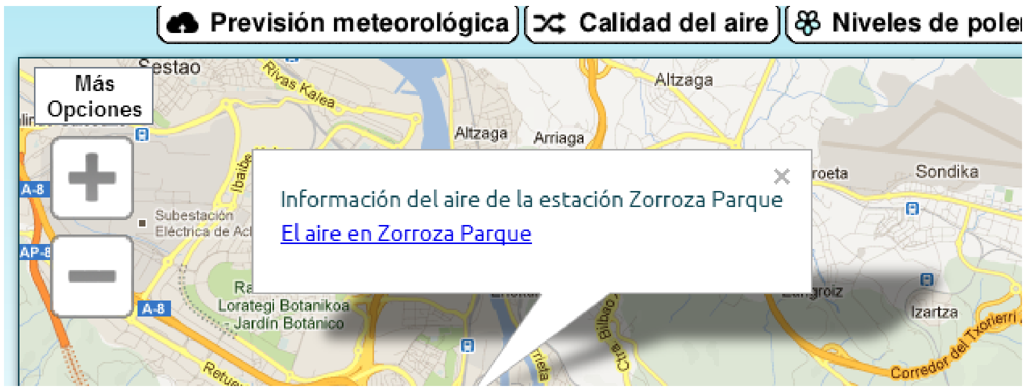
2.2.2.1.1. The goal

There are many people with special sensitivity to environmental allergens that need to know the real-time levels of pollen and predictions in order to take proper medication or avoid certain areas. Also, people with asthma need to know the air quality in their area of residence or work. These levels are different depending the place the user is located. This service would solve this problem by providing accurate information about levels in the city.

2.2.2.1.2. User experience

The website displays a map where the user can see the weather forecast, the air quality, the pollen levels and the weather information.

The user can click on a information flag and go to their measurements.



Información del aire de la estación Zorroza Parque
El aire en Zorroza Parque

(Figure 8. Hoy respiro screenshot)

More information will be displayed in a pop-up. This pop-up have some measurements and a map that displays the how much contaminated is a zone in Euskadi.

2.2.2.1.3. Criticism

There are a lot of static pages like the pollen levels, a better interactivity is need. Map could have zones displayed in clouds with the levels of contamination, that could help users to understand quickly the map. That is very related with our project because we will have some kind of bounding box to separate the boroughs.

More accurate historical of the measurements could be very useful for the users if they want to know if a zone is contaminate for a momentary problem. The implementation of some kind of alerts for being displayed in mobile phones or in a pop-up will be useful for emergencies. In our project is not necessary to make alerts because it does not have any emergency information to be displayed.

2.2.2.1.4. Technologies

HoyRespiro uses Symfony. Symfony is one of the best frameworks for PHP and I will analyze its peculiarities. It is designed to use the pattern Model-View-Controller. MVC is a software architecture pattern that separates the representation of information from the user's interaction with it. Separates the business logic, the server logic and the front-end web. Contains numerous tools and classes aimed at shortening the development time of a complex web application. In addition, it automates common tasks, allowing the developer to involve himself to the specifics of each application. As a database they use MySql. This is a good solution for our project as well as Ruby or Python.

2.2.2.2. SuperHub

The SUPERHUB project (SUstainable and PERsuasive Human Users moBility in future cities) provides a user-centric, integrated approach to multi-modal smart metropolitan mobility systems. It will design and test an open platform able to combine in real time all mobility offers from the relevant stakeholders together with a set of enabling mobility services able to address users mobility needs, to redesign transport route options and to foster behavioural changes.

As front-end technologies they use CSS, a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language and jQuery. They use Google Maps API too.

2.2.2.2.1. The goal

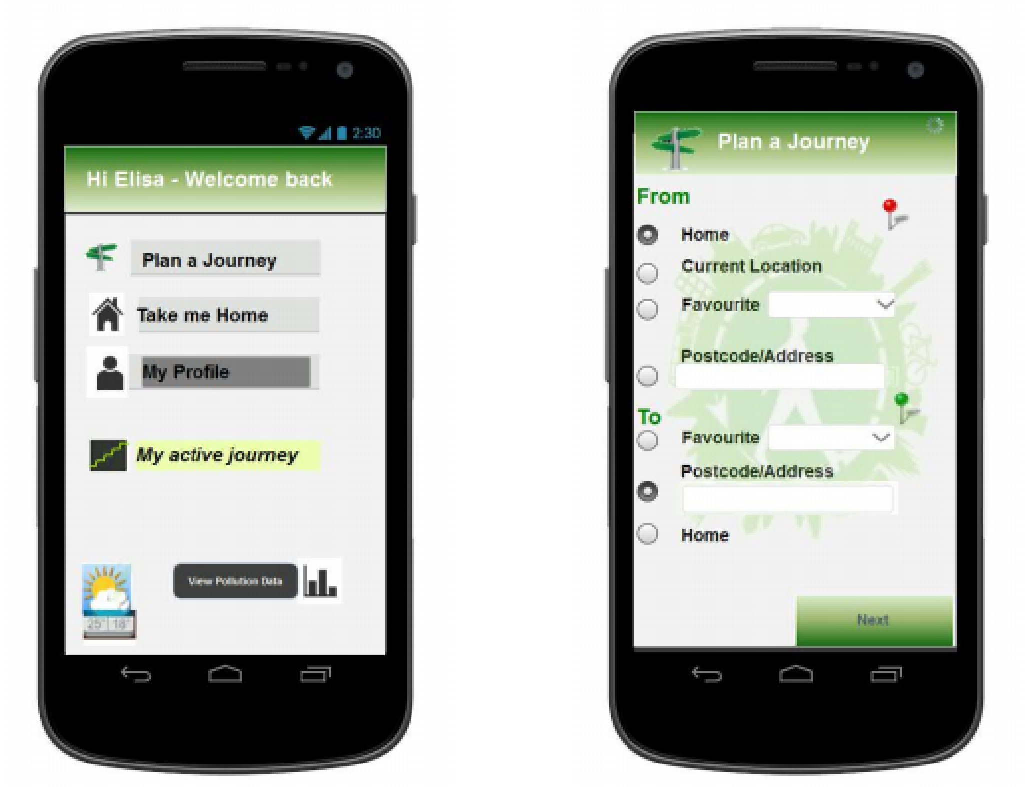
SUPERHUB promotes the creation of a new urban mobility services ecosystem, where all actors are represented and the take-up of virtuous behaviours is facilitated by the development of an open platform able to:

- gather real-time data from all possible mobility sources
- provide matchmaking and negotiation capabilities between providers and consumers of mobility offers for better routing decisions
- enable the development of mobility services able to fulfil users' needs and stimulate behavioural changes.

SUPERHUB foresee the creation of an open platform able to combine in real time all mobility offers together with a set of enabling mobility services able to address users mobility needs, to redesign transport route options and to foster behavioural changes.

2.2.2.2.2. User experience

SUPERHUB is a mobile app that helps users to design the best way to travel in a city, for example, if it is a sunny day and there are a traffic jam on the way to work the app will suggest the user avoid some streets or go walking. This is made dynamically (with GPS coordinates) and the route can be remade when the user is following the instructions.



(Figure 9. SuperHub screenshot)

The user can see the past trips, if he has completed his ecogoals (like low CO2 contamination produced). If he completes his ecogoals he can receive point which can be used to buy rewards like metro cards or bike tours.

The users can report and share incidents that have happened in the city and the other users will be able to see them.

2.2.2.2.3. Criticism

Not all the companies give their information for free, even if they are public. For example, public bike renting in Barcelona (Bicing) do not give too much information to application developers. In our project we do not need information of any institution.

The report of incidents can not be accurate because it is possible that some users make a mistake doing a report or they are just lying. In OUO users can lie too but is a risk that we have to assume and we can try to minimize it making the game funny and interesting.

The mobile applications with a high use of GPS tends to waste a lot of battery running the GPS signal. With the actual mobile technology users can only use these kind of apps one hour a day or less. This is not a problem for OUO because is not oriented to mobile phones and it does not need GPS.

2.2.2.2.4. Technologies

SUPERHUB is a mobile app that works with a server that gather data from the users and other information fonts like public transport services or weather indicators as well as pollution and other contamination sensors. Some of this servers with information about the cities are provided by the city council. In our case we only need communication between users and the server.

2.3. Crowdsourcing/Smartcities

2.3.1. Introduction

Every single city in the world can be reflected in a geographic map but there is also another map that can be even more interesting, the map of the subjective factors constructed by their citizens.

When you move to a new city the first days you can only remember some streets that you are close or on the way to your work but in a few months you know your entire neighborhood or even more. That is your mental map of the city, and you may have feelings related to these streets, places or facilities.

Feelings related to the city are created with scenes that you have seen and how did you feel when you was there. For example, if you were sad and you went to a park you will remember the park with bad feelings but someone who spend a good time there will remember this park like a happy place.

It is impossible to store a subjective map of whole city but combining all these maps it is possible to make something similar to the reality. To know how to get these psychological maps and the projects related with it will explain the best ways to find it.

2.3.2. The studies

The first study about the psychological maps was "The Image of the City" in 1960[2] and it shown that people remember the city of Boston better than others and there were a lot of coincidences on the places that was better known. This study thrown that people thinks in term of points, these points are arranged in some hierarchy and that creates bounded areas, paths and barriers that connect or block these points.

In 1972 a study was published with the objective of discovering the psychological map of the citizens of New York City[3]. Using the information from the previous studies they came to a conclusion: a highly imagible city does not mean that every point is equally identifiable. Rather, there are clearly identifiable focal points throughout the city which are interconnected and thus form a coherent picture. This helps us to know how if an individual is placed at random at a point of a city, how likely is he to know where he is. The scene has to be differentiable, he must match the unique input against some memory of it and he cannot necessarily place it in relation to other parts of the city.

In the their study they divided the map of NYC in the 5 neighborhoods, made a grid with 1000-meter line of longitude and 1000-meter line of latitude. They took 200 subjects differentiated by their place of residence. The scientists displayed images of NYC and they have to answer which neighborhood and street was.

The study shows that Manhattan was the most recognizable of the five boroughs. The recognition of the streets shows exactly the same. Thus it is correct to say that NYC in not merely culturally but also imagistically rooted in Manhattan and the other boroughs are often confused with Queens, which is the most homogenized neighborhood.

The major findings of the study were: an area can only be recognized if people are exposed to it and that happens when it attracts persons from all over the city and the overall architectural or social distinctiveness of the area.

[2] The Image of the City, Kevin Lynch, *MIT Press*, 1960
[3] Milgram S., Greenwald, J., Kessler, S., McKenna, W., & Waters, J. 1972. A phsychological map of New York City. *American Scientist, 60*, 194-200

2.3.3. The projects

The crowdsourcing/smartcities projects that can help us to understand the state of art and what have been done: UrbanGems, UrbanOpticon, PlacePulse and Ushahidi. These are four projects that share more similarities with Open Urban Opticon.

2.3.3.1. UrbanGems

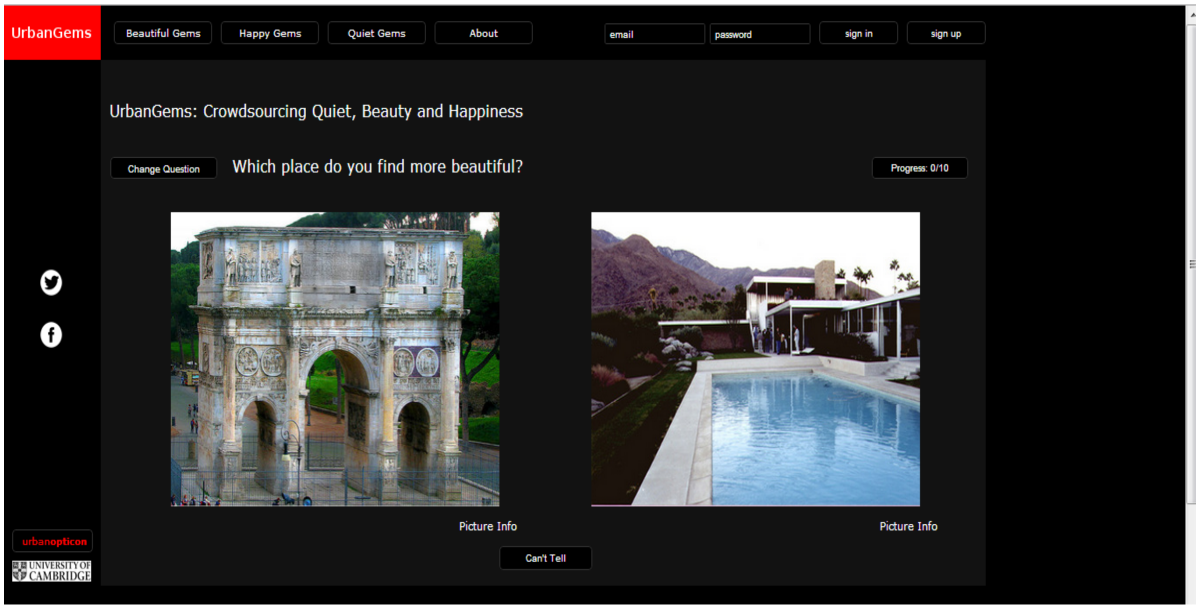
UrbanGems is a website project that uses crowdsourcing to know what people feels about the neighborhoods in London. They are related with the kind of feelings that users undergo, for example, calm, beauty or happiness.

2.3.3.1.1. The goal

This project pretends to know what people think about different streets of London. That is interesting because researchers can relate the beauty, the calm or the happiness with other index of a city statistics like habitability or criminality.

2.3.3.1.2. User experience

The website displays two images from Google Street View and the user have to vote which one looks beautiful, happy or quiet and he has to say the percentage of the people that could think the same.

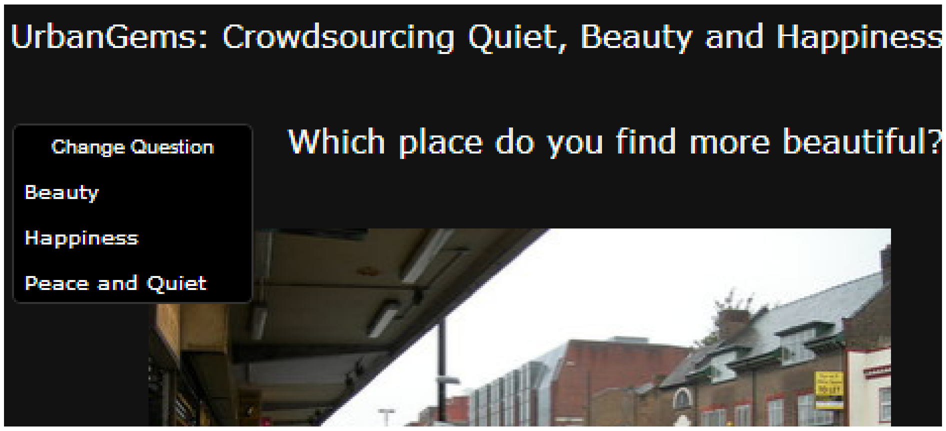


UrbanGems: Crowdsourcing Quiet, Beauty and Happiness

Which place do you find more beautiful?

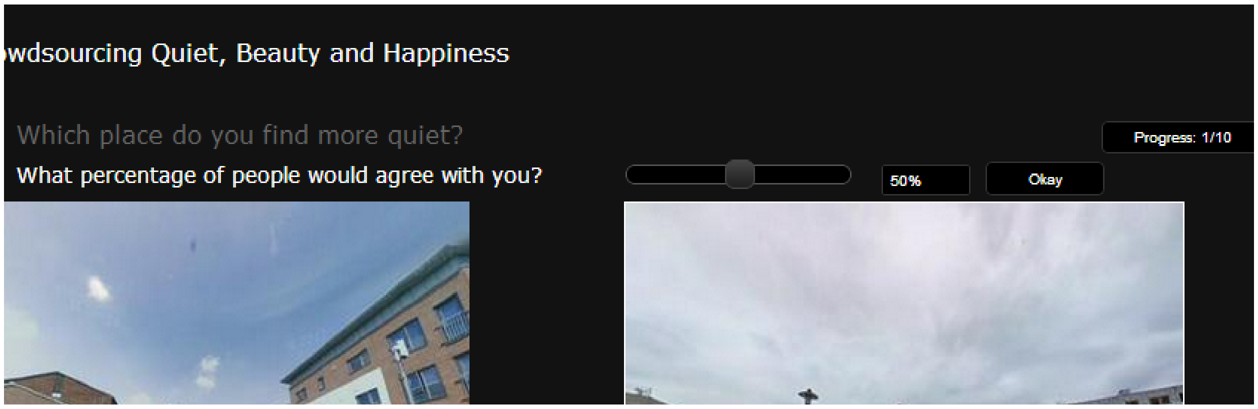
(Figure 10. UrbanGems screenshot)

First of all, the user can sign in and sign up or play like an anonymous player (at the end of the questions will ask to sign up). To be a registered player will give recommendations that suit the user.


UrbanGems: Crowdsourcing Quiet, Beauty and Happiness
Which place do you find more beautiful?
Change Question
Beauty
Happiness
Peace and Quiet

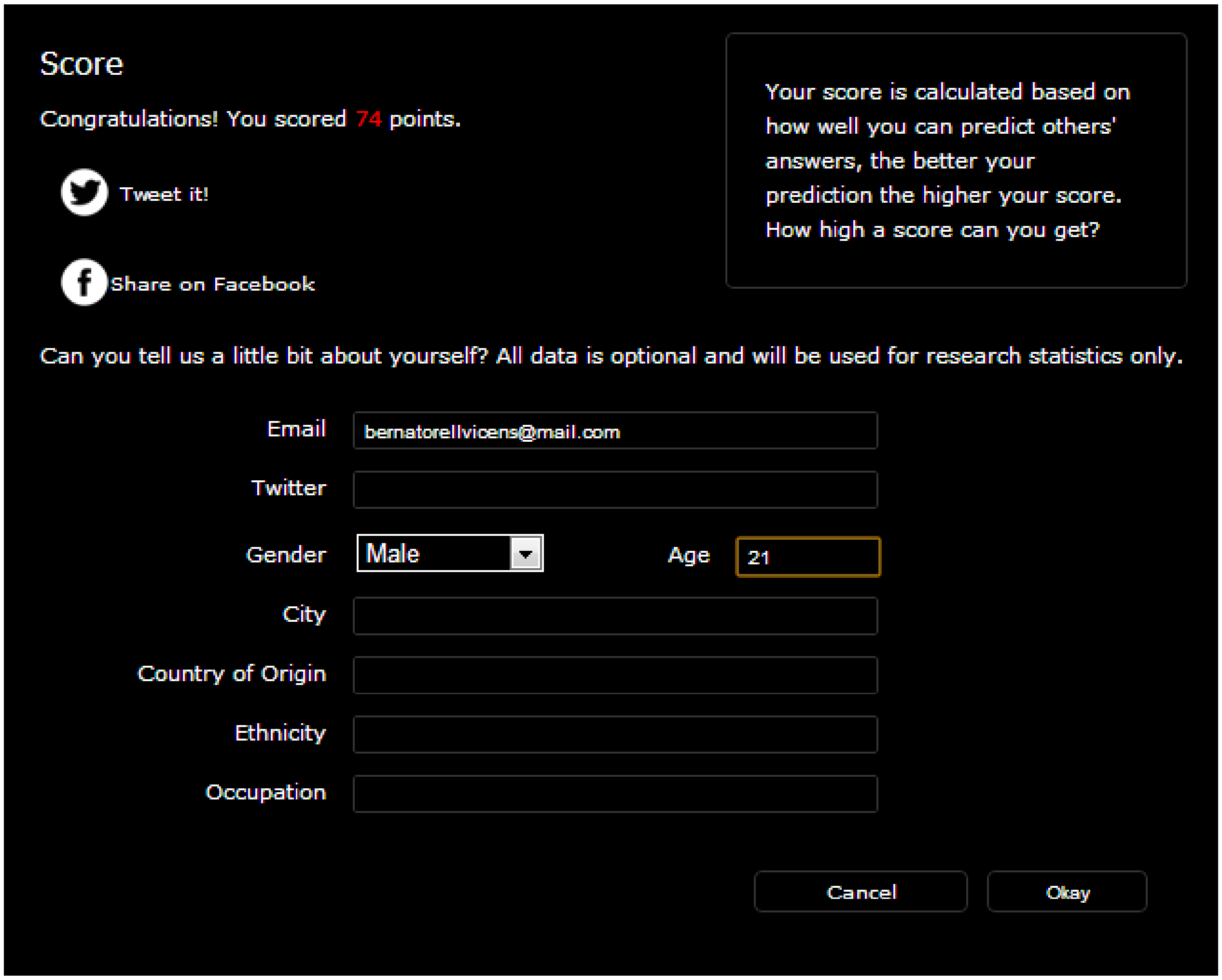
(Figure 11. UrbanGems screenshot 2)

The main page shows two images that can be related with feelings. The images change when the user change the question and the questionnaire it is reset.


wdsourcing Quiet, Beauty and Happiness
Which place do you find more quiet?
What percentage of people would agree with you?
50%
Okay
Progress: 1/10

(Figure 12. UrbanGems screenshot 3)

The user can click over on the image that he thinks it is more beautiful or press “Can’t tell”. If the user click on an image a percentage bar will be displayed and he has to put the scroll on the percentage of people that he thinks that are agree with him.


Score

Congratulations! You scored 74 points.

Tweet it!

Share on Facebook

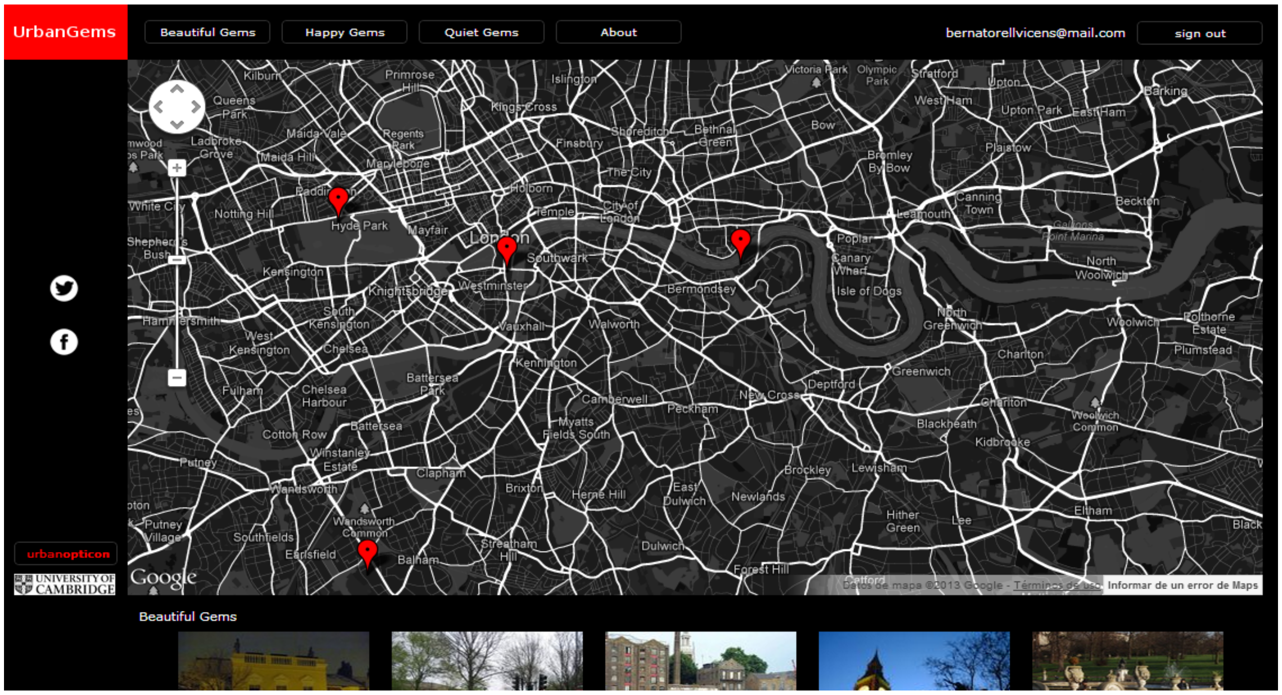
Your score is calculated based on how well you can predict others' answers, the better your prediction the higher your score. How high a score can you get?

Can you tell us a little bit about yourself? All data is optional and will be used for research statistics only.

Email	bernatorellvicens@mail.com
Twitter	
Gender	Male
Age	21
City	
Country of Origin	
Ethnicity	
Occupation	

(Figure 13. UrbanGems screenshot 3)

Once the user has completed the test, composed of 10 comparisons he will have a score based on the similarity with the results of all the other people. He can give more information about himself that will be used for research statistics.



(Figure 14. UrbanGems screenshot 4)

Moreover, UrbanGems have other uses than the game, the user can be recommended with beautiful, quiet or happy gems (places) and if he is logged he has a personalized recommendations depending on the results of his previous games.

2.3.3.1.3. Criticism

There are some characteristics on the photo that can influence the opinion of the user like the quality of the photo, the weather (rainy or sunny), what was happening when the photo was taken (traffic jam or a quiet Sunday) or every single detail that can make the user thinks that an image is better than other without being completely objective for the characteristics beyond the photo. That is very related with our project because a bad photo can confuse the user and maybe he fails a photo of a place that he knows.

The results trends to homogenize the lifestyle and the cities, everyone loves parks in the spring and hate a highway with a lot of cars. That is very related with our project to but it is usefull for us because is the goal of our research.

2.3.3.1.4. Technologies

The main language of the UrbanGems server is Ruby and it uses Ruby on Rails that is a web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Control pattern. Our project can use Ruby too.

As a relational database they use sqlite. As front-end technologies they use javascript and html. OUO will use javascript and html too because are the most used and can be interpreted by all the browsers.

2.3.3.2. UrbanOpticon

The main objective of the UrbanOpticon is known if the user can recognize what parts of London with a displayed image of Google Street View.

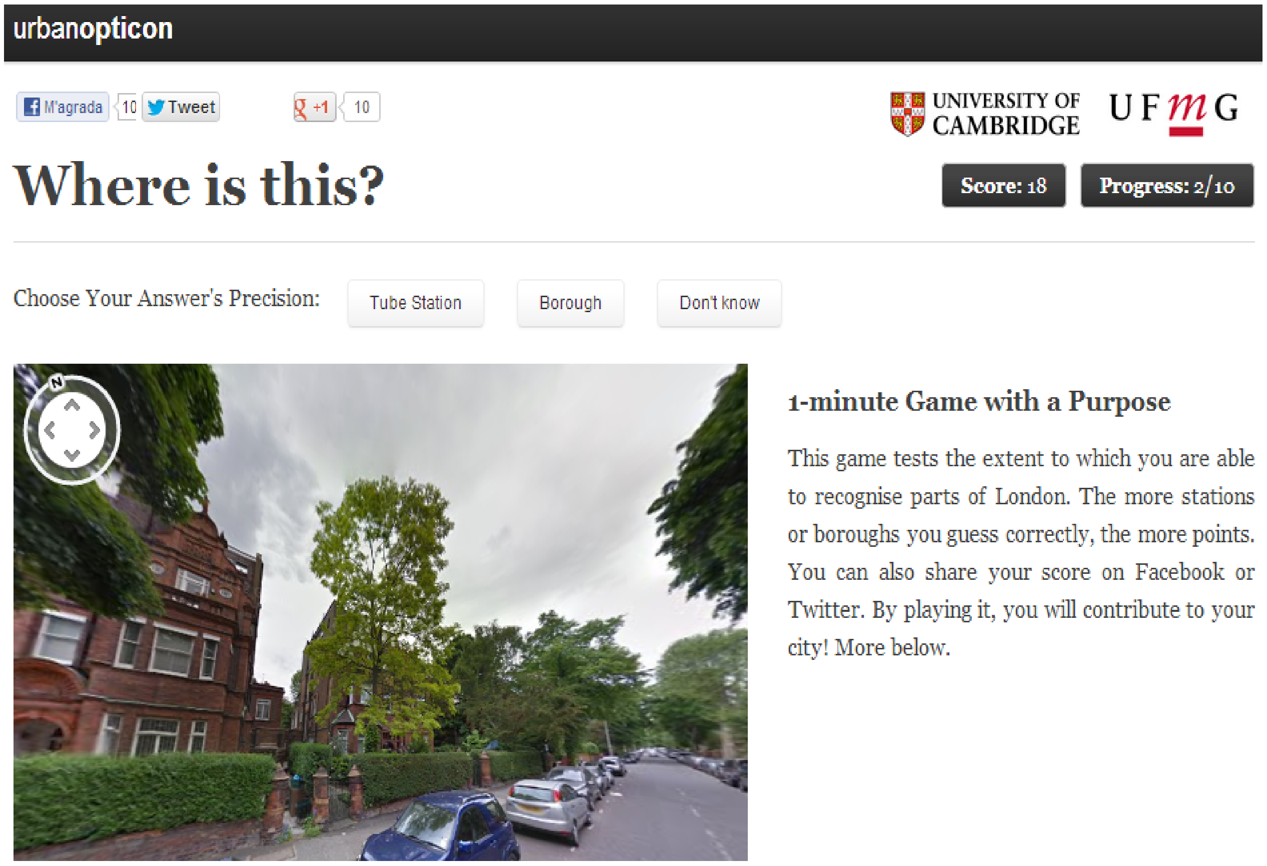
2.3.3.2.1. The goal

The goal of UrbanOpticon, apart of having a good time playing a game about London, is to obtain information about the remarkable places in London and to draw the map of Londoners' mental images of the city.

The mental maps of the citizens are related with public facilities or urban interventions that makes react the viewer, for example, colorful streets or horrible buildings. This map is interesting by itself and it will be useful for a lot of studies as those that are trying to discover what public facilities improve the image of the boroughs.

2.3.3.2.2. User experience

In the UrbanOpticon website the users have only one picture and the user has to answer with the closest tube station or borough or in case the user does not know the place he just has to click "Don't know".



Where is this?

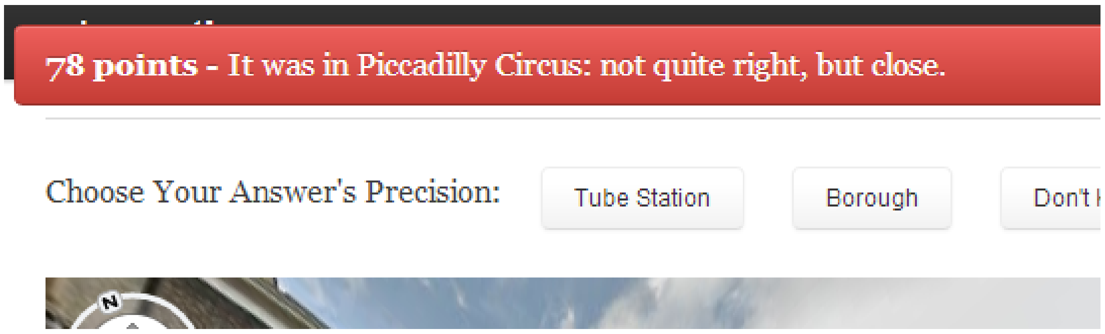
Choose Your Answer's Precision:

1-minute Game with a Purpose

This game tests the extent to which you are able to recognise parts of London. The more stations or boroughs you guess correctly, the more points. You can also share your score on Facebook or Twitter. By playing it, you will contribute to your city! More below.

(Figure 15. UrbanOpticon screenshot)

Depending on the proximity the user will score points, he do not need to know the right point. If he clicks "Don't know" he will score 15 points and the next image will be shown.



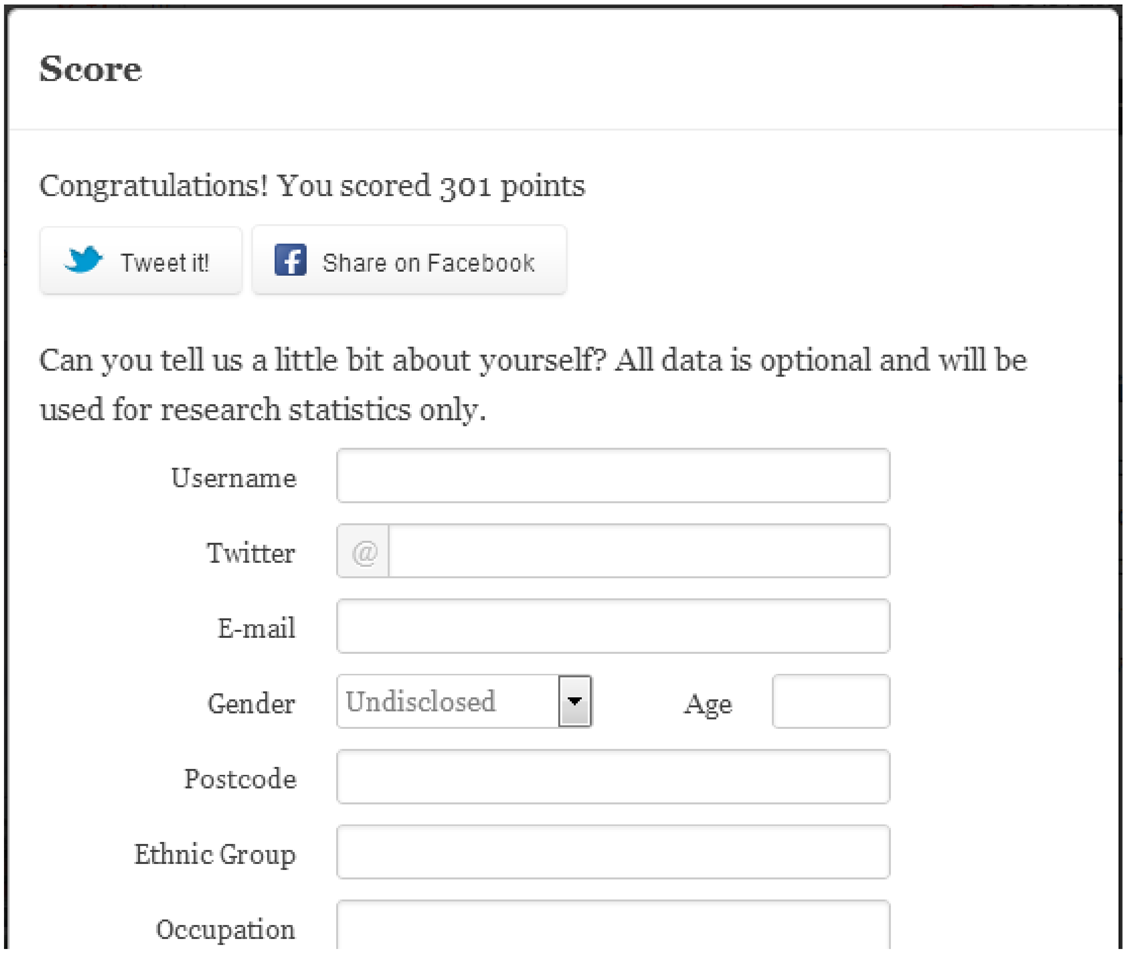
78 points - It was in Piccadilly Circus: not quite right, but close.

Choose Your Answer's Precision:

(Figure 16. UrbanOpticon screenshot 2)

The image of Google Street View is a static image where the user can not move but the user can move the camera 360º.

When the user has finished the test, he will be asked for some personal information for research statistics.



Score

Congratulations! You scored 301 points

Can you tell us a little bit about yourself? All data is optional and will be used for research statistics only.

Username	
Twitter	@
E-mail	
Gender	Undisclosed
Age	
Postcode	
Ethnic Group	
Occupation	

(Figure 17. UrbanOpticon screenshot)

2.3.3.2.3. Criticism

The game does not have enough reward for the users, it need a way to compete against other players. For example a ranking in the website with the player who scored more points or the total games that he has played. This can be implemented in our project, depending on what way of gamification we want to follow.

The dynamic result map of the most recognizable places of London could be interesting. The user would feel grateful if he sees the result of his interaction with the website. That will be implemented in our project.

Another criticism as an interested user is the poor interactivity when the user have to mark a tube station or a borough. If users could click on the station in the tube map or a neighborhood would increase the interactivity with the user and therefore provide a better user experience. It is planned to have this feature in our project.

2.3.3.2.4. The problems

The documentation of the project shows the problems that they had. Firstly, users were very frustrated when they played because even if they had been living in the city for a lot of years they could not recognize enough places because of the random image selection. To solve this problem the developers put some pictures easy to recognize although the evaluations obtained for such pictures were discarded in order to ensure the validity of the results.

Secondly, the beta version did not show any feedback about the correct answer so the user who was playing to learn more about London was very disappointed. The final version of the game shows the correct answer.

The website did not have anything about the purpose of the game and some testers wanted to know. To solve this they put a little description of the project in the website.

The testers did not know exactly what tube station was but they know the borough. To solve this the developers add the possibility of answer with a borough or region but with less score than if the tester knows the tube station.

2.3.3.2.5. Technologies

In UrbanOpticom the main language of the server PHP without any framework. PHP is a scripting language designed for web development to produce dynamic web servers. It is free and there are a lot of documentation and forum with user support. PHP is loosely typed, which makes basic scripts much faster to develop with less attention to design. On the other hand it has some disadvantages like that is slow compared with other compiled languages like C and There are many ways to do one thing, and many cases where a function has ambiguous handling due to legacy support for PHP development history. As database they use MySql. PHP with Symfony is one of the options that OUO can use to make the server.

In front-end utilities they use Bootstrap a software designed to help web developers. It includes HTML, CSS and Javascript. It has list of components that can be used easily like progress bars, buttons, labels, etc. and it has a few example layout. It can be used in a lot of web browsers, even in mobile web browsers. Our project will use Bootstrap because make the process of develop a front end website easier.

They use LESS too. It extends CSS with dynamic behavior such as variables, mixins, operations and functions. With LESS the user do not need to write all of these painful lines of CSS code. We can use LESS in our project as well.

2.3.3.3. Place Pulse

Place Pulse is another project that helps us to discover the hidden knowledge of the people about their perceptions and display it on data sets.

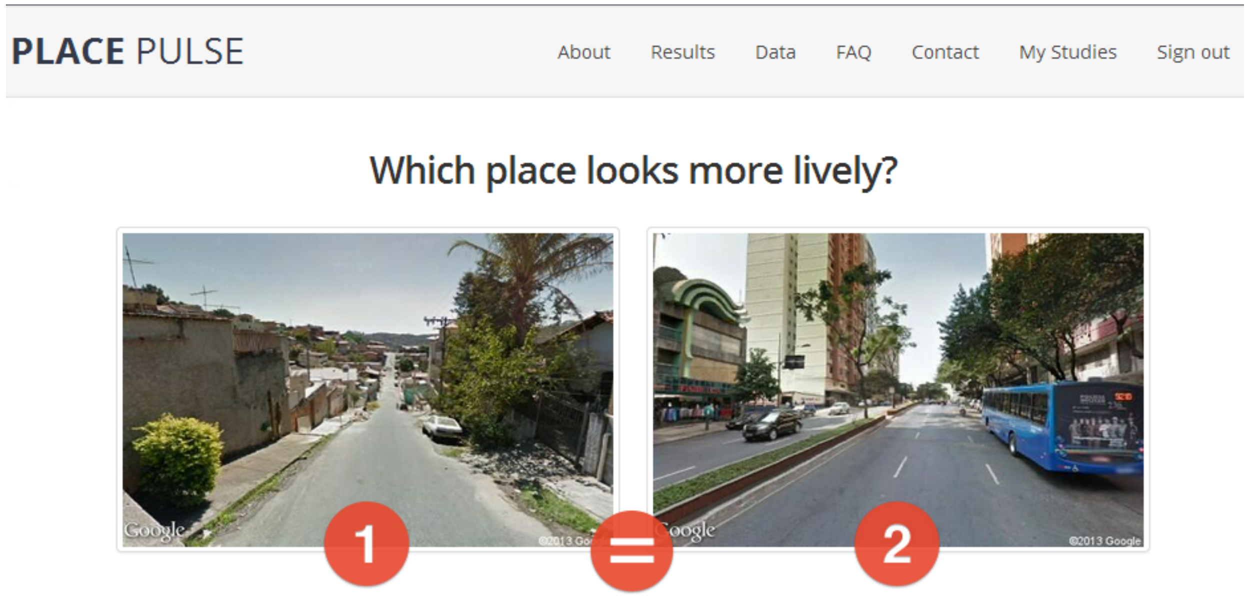
2.3.3.3.1. The goal

The goal of Place Pulse is do an open source platform that allows developers make an easy website with a question with the pattern "Which place looks (safer, happier... or any other comparison that users can imagine)" and two images. After that, users can answer this question with the poll link.

2.3.3.3.2. User experience

The user experience in Place Pulse is very similar to the other two projects, the main differences are highlighted following.

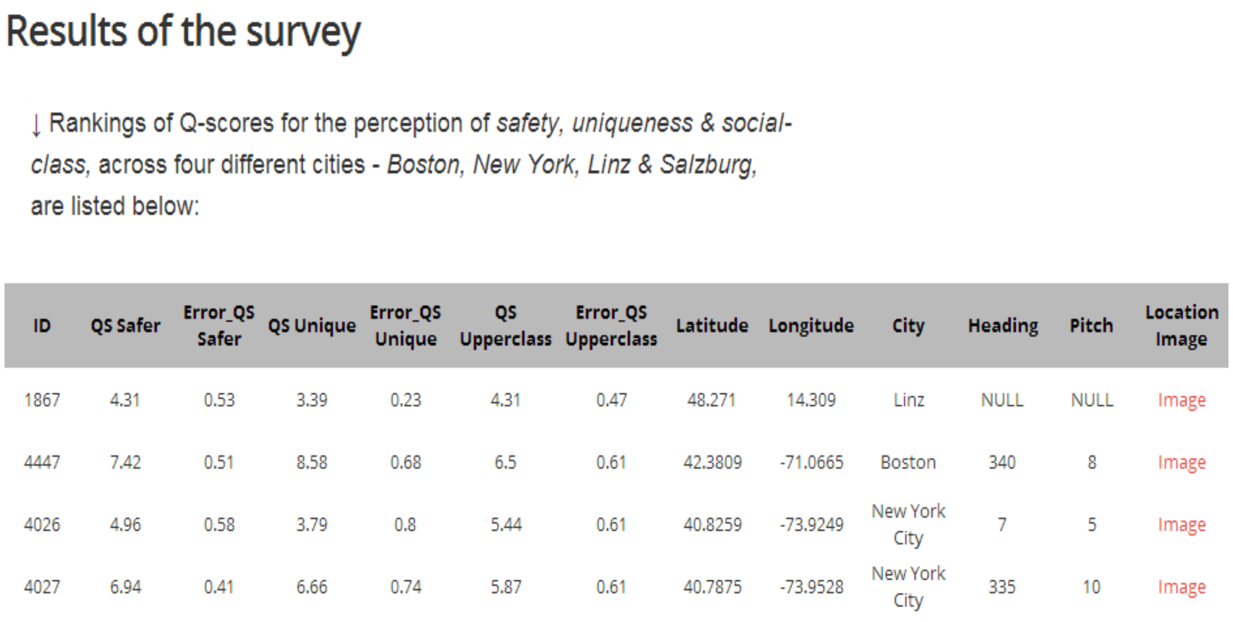
First of all, to provide credentials the users can use Facebook or Persona accounts. If he enters without a link of a poll he will be asked for a random questionnaire (in our case "Which place looks more lively?") and only two images from Google Street View will be displayed.



(Figure 18. Place Pulse screenshot)

It is not a game so the user can click images until he get tired. It have a lack of Gamification incentives.

Another important thing in the website is the access to results, that show the Q-scores of the perception of safety, uniqueness and social-class of four cities.



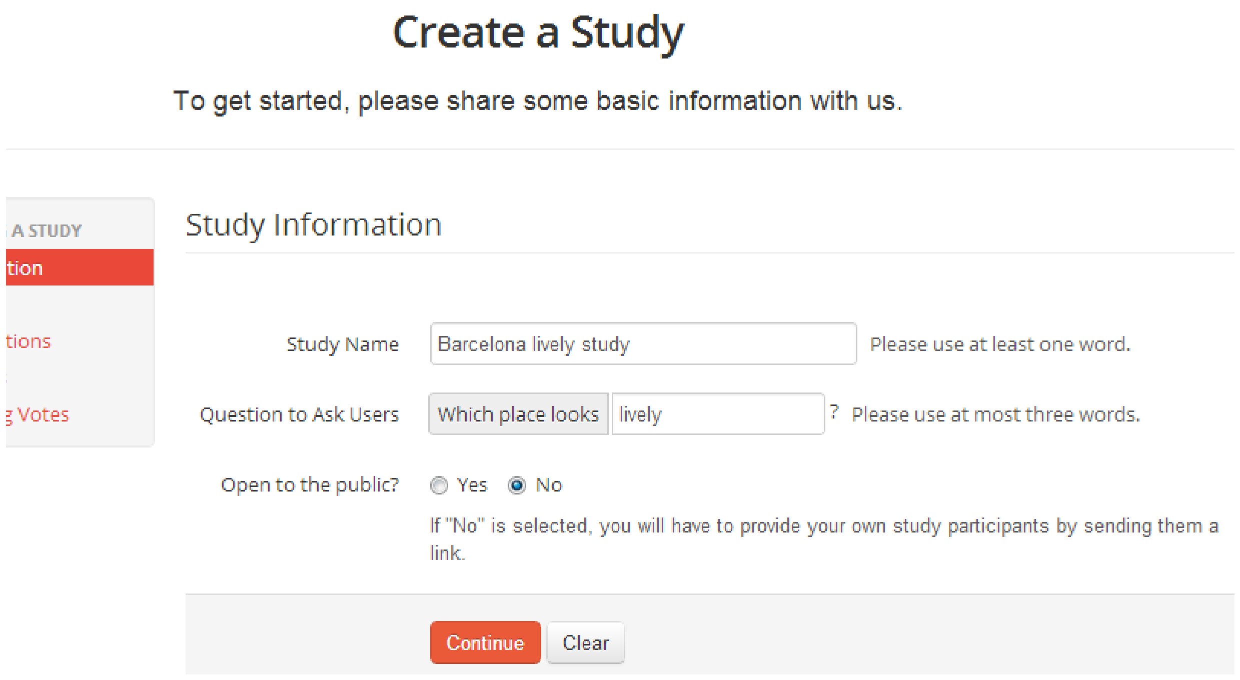
Results of the survey

↓ Rankings of Q-scores for the perception of *safety, uniqueness & social-class*, across four different cities - *Boston, New York, Linz & Salzburg*, are listed below:

ID	QS Safer	Error_QS Safer	QS Unique	Error_QS Unique	QS Upperclass	Error_QS Upperclass	Latitude	Longitude	City	Heading	Pitch	Location Image
1867	4.31	0.53	3.39	0.23	4.31	0.47	48.271	14.309	Linz	NULL	NULL	Image
4447	7.42	0.51	8.58	0.68	6.5	0.61	42.3809	-71.0665	Boston	340	8	Image
4026	4.96	0.58	3.79	0.8	5.44	0.61	40.8259	-73.9249	New York City	7	5	Image
4027	6.94	0.41	6.66	0.74	5.87	0.61	40.7875	-73.9528	New York City	335	10	Image

(Figure 19. Place Pulse screenshot 2)

The most interesting feature of the website is the possibility of create a study. Firstly, the user has to name the study and ask a question.



Create a Study

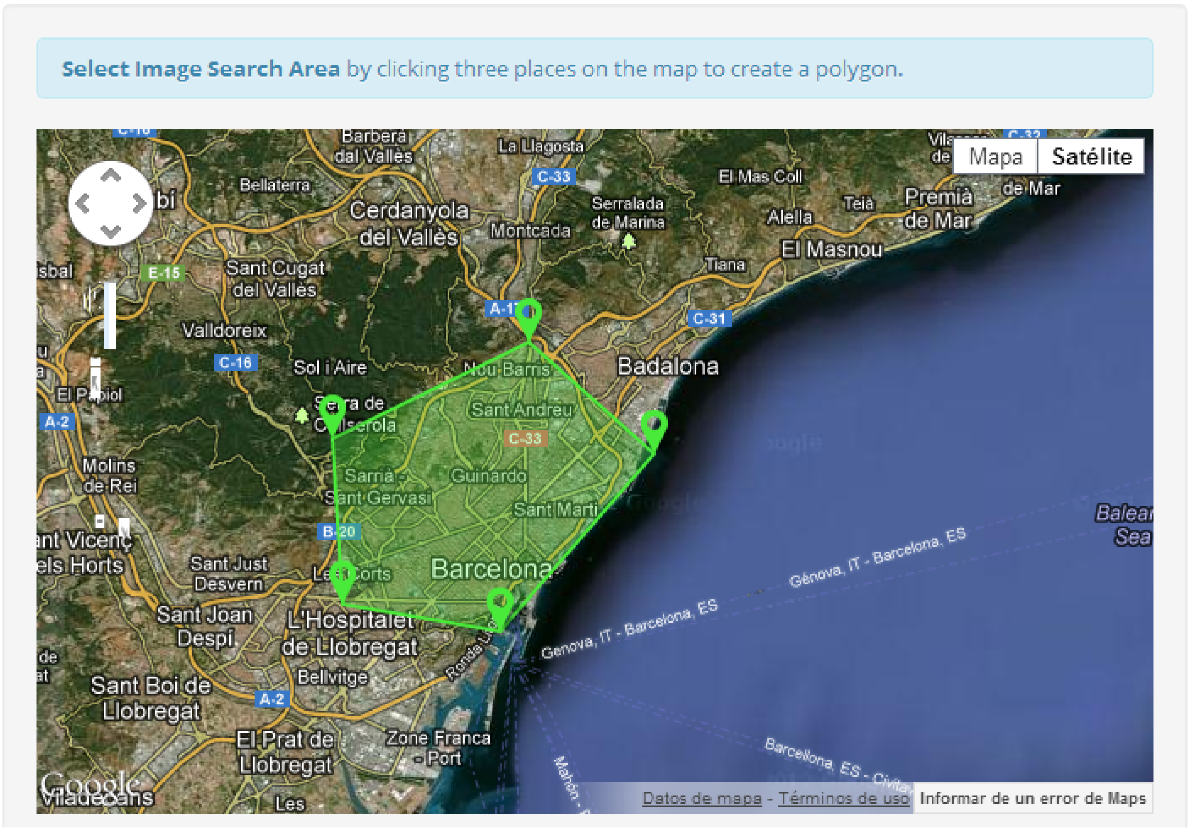
To get started, please share some basic information with us.

Study Information

Study Name	Barcelona lively study	Please use at least one word.
Question to Ask Users	Which place looks lively ?	Please use at most three words.
Open to the public?	Yes [✓ No]	If "No" is selected, you will have to provide your own study participants by sending them a link.

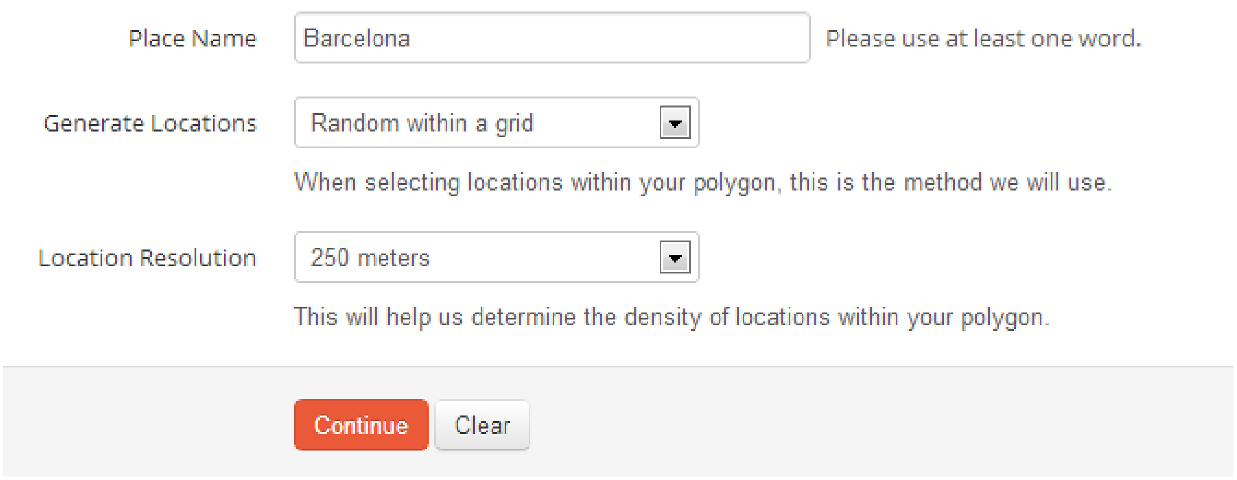
(Figure 20. Place Pulse screenshot 3)

Then, the user has to define an area with Google Maps.


Select Image Search Area by clicking three places on the map to create a polygon.

(Figure 21. Place Pulse screenshot 4)

Users can generate locations randomly, evenly or random with a grid and the density.


Place Name: Barcelona — Please use at least one word.

Generate Locations: Random within a grid

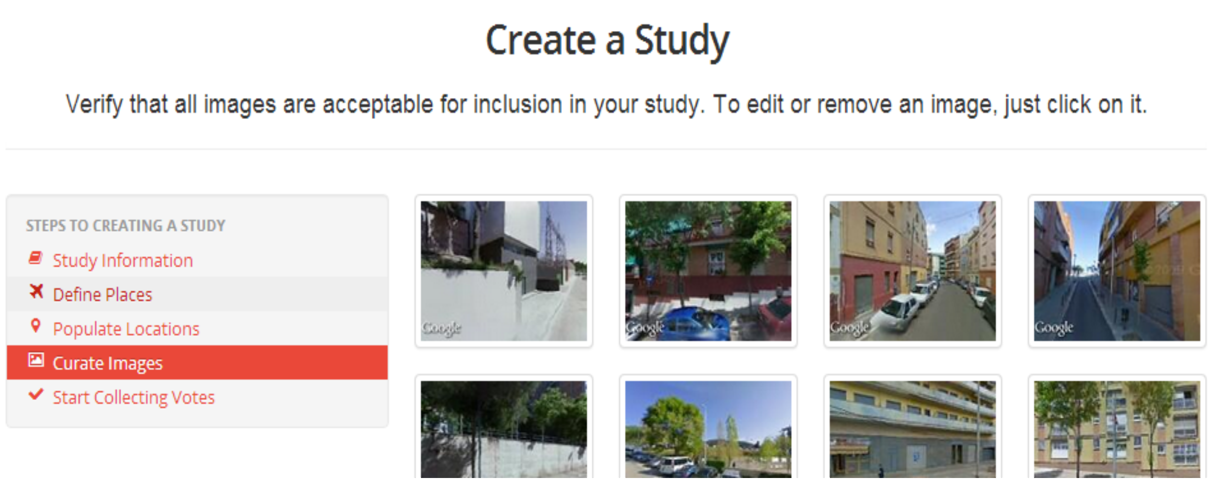
When selecting locations within your polygon, this is the method we will use.

Location Resolution: 250 meters

This will help us determine the density of locations within your polygon.

(Figure 22. Place Pulse screenshot 5)

The random images are displayed and the user can remove some of them if he wants.

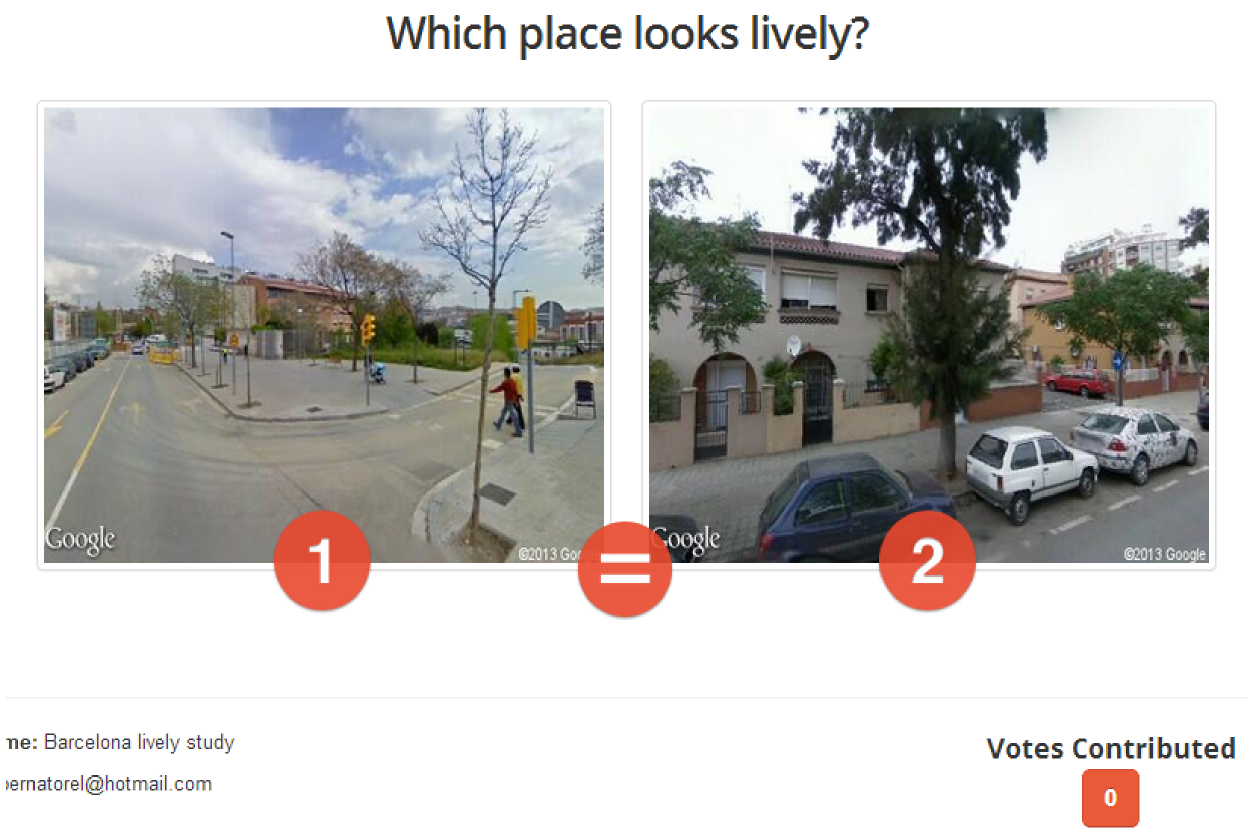


Create a Study

Verify that all images are acceptable for inclusion in your study. To edit or remove an image, just click on it.

(Figure 23. Place Pulse screenshot 6)

Finally he can confirm everything and publish the study.



Which place looks lively?

ne: Barcelona lively study

ernatorel@hotmail.com

Votes Contributed

0

(Figure 24. Place Pulse screenshot 6)

2.3.3.3.3. Criticism

Place Pulse has a big issue, the lack of gamification. The website project do not have any kind of reward to the user and that makes participating in very boring. The users feels like they are playing only for statics. It could improved doing the same that UrbanOpticon does, give a score depending on the similarity results with the other participants or make a ranking with the most happier, salfer... place in the city. Our project will have some gamification method that makes it funnier.

One thing that might be improved is a way to put points by a vector of coordinates and the direction of the camera. That improve will add the possibility of make user own points with other software that maybe can do a better selection of point with some other purpose like compare only the sights of the city or the streets that face the sea. We can do the same with the points of the map. Possibly this will be made for our project.

2.3.3.3.4. Technologies

They use a framework named Flask written in Python. Flask is called a microframework because it keeps the core simple but extensible. There is no database abstraction layer, form validation, or any other components where third-party libraries already exist to provide common functionality. However, Flask supports extensions, which can add such functionality into an application as if it was implemented in Flask itself. There are extensions for object-relational mappers, form validation, upload handling, various open authentication technologies, and more. Flask can not be used in our project because it has a short development time and Flask does not have all the features that it needs, like easy DB management.

PlacePulse uses a mongoDB and stores cities, places, studies and votes. Their old database was MySQL. To manage the maps and locations uses Javascript with Google Maps API and Google Street View to get the images. For front-end website tools they have used Bootstrap, Font Awesome (iconic font), jQuery and LESS. Some of these features will be used in our project.

2.3.3.4. Ushahidi

Ushahidi platform is built as a tool to easily crowdsource information using multiple channels, including SMS, email, Twitter and the web in emergency situations.

2.3.3.4.1. The goal

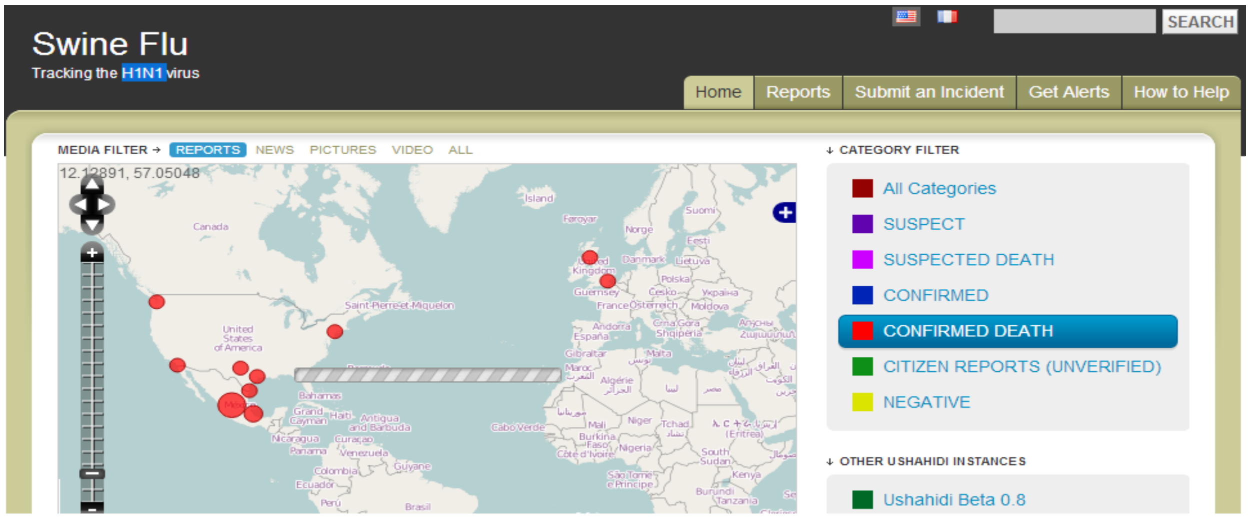
The goals of Ushahidi is be easy to use, accessible to anyone and deployable worldwide. Ushahidi is an open source platform, can be deployed by everyone and suit what the user community needs. It was used in emergency situations as post elections violence in kenya, swine flu, etc.

2.3.3.4.2. User experience

The users that have problems in an emergency can send messages looking for help. The messages can be submitted with a mobile phone or a personal computer.

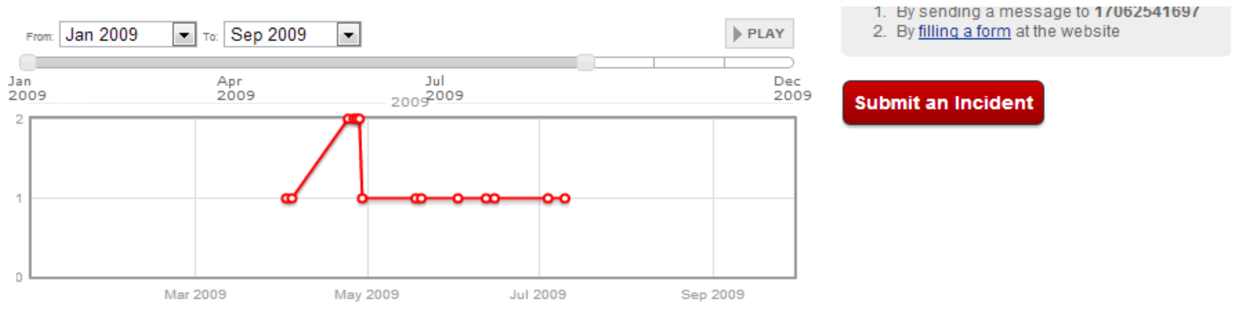
The developers can download the the web server in github and modify it, only little knowledge of PHP and MySql are need.

For example, a website that use of Ushahidi for the tracking of swine flu.



(Figure 25. Ushahidi screenshot)

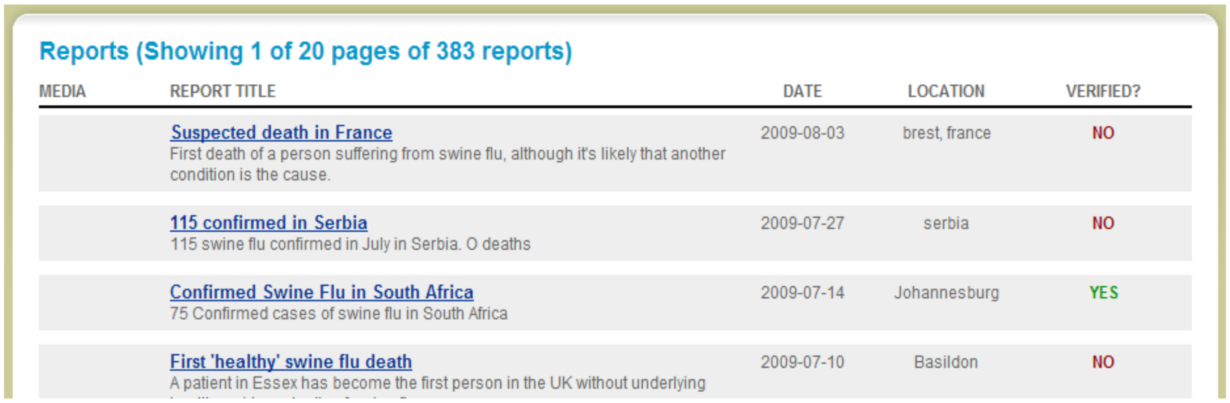
The user can put some filters in the map and see the chronology of the events. He can visualize dynamic statistics.



From: Jan 2009 To: Sep 2009
PLAY
Jan 2009
Apr 2009
Jul 2009
Dec 2009
2
1
0
Mar 2009
May 2009
Jul 2009
Sep 2009
1. By sending a message to 17062541697
2. By filling a form at the website
Submit an Incident

(Figure 26. Ushahidi screenshot 2)

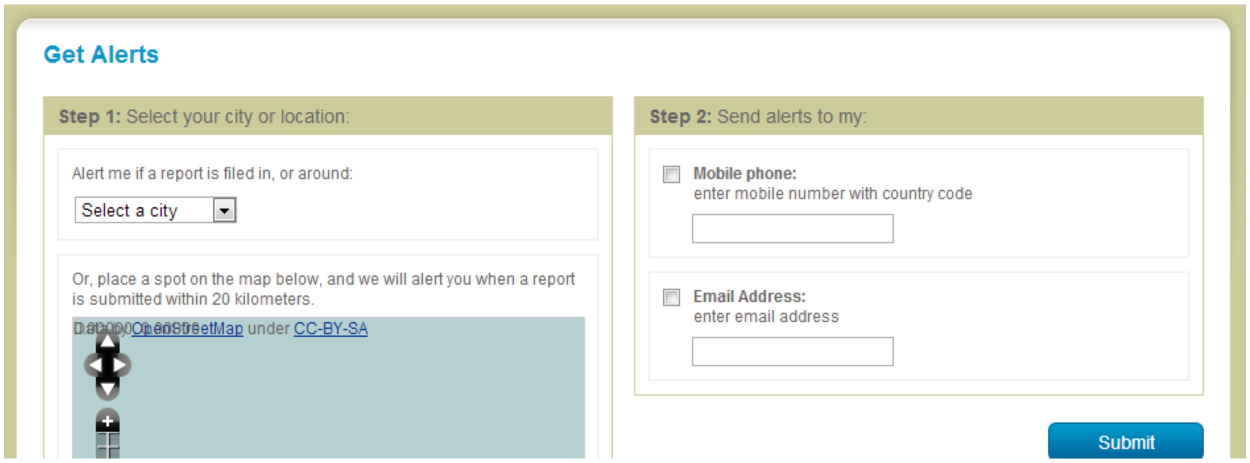
User can submit a new report in the website or send a SMS. He can see the reports of the other users and get alerts on his mobile phone or email.



Reports (Showing 1 of 20 pages of 383 reports)

MEDIA	REPORT TITLE	DATE	LOCATION	VERIFIED?
	Suspected death in France First death of a person suffering from swine flu, although it's likely that another condition is the cause.	2009-08-03	brest, france	NO
	115 confirmed in Serbia 115 swine flu confirmed in July in Serbia. O deaths	2009-07-27	serbia	NO
	Confirmed Swine Flu in South Africa 75 Confirmed cases of swine flu in South Africa	2009-07-14	Johannesburg	YES
	First 'healthy' swine flu death A patient in Essex has become the first person in the UK without underlying	2009-07-10	Basildon	NO

(Figure 27. Ushahidi screenshot 3)



Get Alerts

Step 1: Select your city or location:

Alert me if a report is filed in, or around:

Select a city

Or, place a spot on the map below, and we will alert you when a report is submitted within 20 kilometers.

Step 2: Send alerts to my:

Mobile phone:
enter mobile number with country code

Email Address:
enter email address

(Figure 28. Ushahidi screenshot 4)

2.3.3.4.3. Criticism

Everyone can submit informa and some of them can lie with bad intentions like manipulate the public opinion. The user may have to contrast information with other sources. Our project does not need moderators because users do not give a public opinion.

2.3.3.4.4. Technologies

Ushahidi uses different languages in different modules like PHP for website, JAVA for android and other libraries and objective-C for the IPhone app. OUO can not use java and objective-C mostly because they do not have the libraries that it needs.

Ushahidi provides an API to the admins and users with some pre made stuff that allows them to go faster. As database they use MySql.

2.4. Technology table

The table with the technologies that the projects we have analyzed are using is found below:

	Server language	**Framework**	**Database**	**Web**
Kickstarter	Ruby	Ruby on Rails		HTML jQuery CSS
Amazon Mechanical Turk	Java .NET Perl, php Ruby			
99designs	PHP		MySql Memcached MongoDB Redis	HTML jQuery CSS
HoyRespiro	PHP	Symfony	MySql	
SUPERHUB	Java			
UrbanGems	Ruby		SQLite	HTML jQuery CSS
UrbanOpticom	PHP	Symfony	MySql	Bootstrap LESS
Place Pulse	Python	Flask	MySQL MongoDB	Font Awesome jQuery LESS
Ushahidi	PHP		MySql	

In conclusion, almost all the project uses some scripting language. They use some kind of framework with they project because it helps a lot when it comes to programming. The databases are not very different one from another, specially SQL databases. jQuery as well as CSS enhancement libraries are frequently used too for all this projects.

3. Technologies

In this section it is described the most interesting technologies and tools available and which can be more useful. The different technologies that we find are almost mandatory for our project because a web application needs a web server in a certain language, a database where the objects are saved and other technologies related with the client view through a browser.

3.1. Web server language

3.1.1. JavaEE

JavaEE is a programming platform for Java based on web server applications language that has several advantages: firstly and the most important, has a strong and wide community that have defended it for over 10 years. Secondly, it has a lot of documentation, examples and tutorials.

The big disadvantage of JavaEE is the poor scalability of the java applications and the developers of the other projects used libraries that are done for scripting languages.

3.1.2. PHP

PHP is a scripting language designed for web development to produce dynamic web servers. PHP on the web development widespread because its early release and there is a lot of documentation. The disadvantages of using PHP are: the learning curve is steep at the beginning and not very modular.

3.1.2.1. Symfony

Symfony is one of the best frameworks for PHP. It is designed to use the Model-View-Controller pattern. MVC is a software architecture pattern that separates the representation of information from the user's interaction with it. It separates the business logic, the server logic and the front-end web. It contains numerous tools and classes aimed at shortening the development time of a complex web application. In addition, it automates common tasks, allowing the developer to involve himself to the specifics of each application.

3.1.3. Python

Python is a general purpose programming language, which can be used to do anything. It is very easy to learn because of their clean syntax. It had extensive libraries and good frameworks.

Python is well-known for its code productivity. Python have problems with the memory intensive tasks.

3.1.3.1 Django

Django is a framework that use Python. It was designed to make common Web-development tasks fast and easy. Users can use Django without a database because it comes with an object-relational mapper in which users describe your database layout in Python code. It allows you to create elegant URLs and other fancy stuff that helps the web-developer.

3.1.4. Ruby

Ruby is an object oriented programming language. In Ruby everything is an object. It is said that Ruby follows the principle of least astonishment, meaning that the language does not confuse the experienced users. Some critics say that Ruby it is alive because of his famous framework and it is not useful by itself.

3.1.4.1. Ruby on Rails

RoR is a very complete framework that have everything that a web-developer needs, management of databases, create templates, gather information, etc.

3.2. Databases

3.2.1. SQL based

Structured Query Language is a specific programming language designed for managing data in relational database management systems.. SQL have a lot of documentation and it is widespread over the world of web-developing. The problems of SQL Databases come with the poor compatibility between the different interpretations of the language.

3.2.2. MongoDB

MongoDB is a document-oriented database, is the most popular NoSQL database management system. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as documents with dynamic schemas (BSON), making the integration of data in certain types of applications easier and faster.

3.3. Webpages

3.3.1. Bootstrap

Bootstrap is designed to help web developers. It includes HTML, CSS and Javascript. It has list of components that can be used easily like progress bars, buttons, labels, etc. and it has a few example layout. Finally, it can be used in a lot of web browsers, even in mobile web browsers.

3.3.2. Skeleton

Skeleton is a small collection of CSS basic files that gives you some help in the website development. It have different components that automate the tasks related with the front-end design. It works in different devices, even mobile phones.

3.3.3. 99Lime HTML KickStar

It is another package that includes everything, Javascript, CSS and HTML. Covers the necessities of a web designer who needs to create website layouts (slideshows, menus, buttons, etc) and make it in an easy, fast and beautiful way.

3.3.4 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.

3.3.5 CoffeeScript

CoffeeScript is a little language that compiles into JavaScript. It had shortcuts that helps you to write less code and go faster.

3.3.6. LESS

LESS extends CSS with dynamic behavior such as variables, mixins, operations and functions. With LESS you do not need to write all of these painful lines of CSS code.

3.3.7. HAML

Haml (HTML abstraction markup language) is based on one primary principle, markup should be beautiful. Haml accelerates and simplifies template creation down.

4. Problem description

Urban Opticon is a website that displays images of London and users have to answer what borough or tube station is closer. The crowdsourced results from the users generate a map of the most recognizable zones in London and can be useful for different studies.

Our goal is make an open source platform for the Urban Opticon that can be deployed by anyone for any city chosen by the user. This platform will allow the user to make their own divisions of the map of the city of study and choose the places that will be displayed to the final user depending on their preference or even make random points. The experimental results will be saved into the experimenter database and made accessible to everyone. Because of the magnitude of the project, and its deployment in production, the project is split in two main parts: (1) the back-end or administrator part, where the experiment is configured, and (2) the front-end or the players interface, where users play the game. This project is divided in two parts, one the administration part of the Open Urban Opticon that is described in this document and the other part, the user interface part, which will be made by another student from Brazil named João Paulo Pesce.

In our platform, besides of the option of create a study for a city, there is the possibility of create an study for the whole world. This feature will be very similar to the city study and it will be even easier to create.

The actors on this project are the following: (1) the experiment creators that deploy the website for their experiment, (2) the final users will be the people that will be asked if they can recognize the zone of an image, and (3) a last actor will be a passive actor that can check the results of the study.

Our project will be easily deployable by a non expert user that need a custom Urban Opticon for a certain city or for various cities. It will not require expert technology skills. All the customization will be made inside the web with an administration interface but if the experiment creator wants a high personalized website he will be able to change the code how he pleases. For the final user will be a classic website and everyone can access the results after the experiment creator approval.

5. Use cases

The overall functioning of the platform is explained for two actors: the administrators experiment (or administrators) creators and the final user.

5.1. Experiment creators

The experiment creator needs to do the following steps to set up an experiment:

- Provide a city division with a shapefile.
- Create and manage city division manually.
- Make easy-to-guess points and random points for an existing experiment.
- Create and manage points.
- World experiment points.

These steps are carefully explained below:

5.1.1. Create a city experiment with a shapefile (.shp)

The experiment creator goes to the import shapefile webpage. First he has to name the city (if already exist will override the experiment). He uploads the zip with the .shp (and .shx, .prj, .dbf files that a shapefile needs). Now, he has to select the column of the shapefile that he is interested in because this names are not standardized and each shapefile can have different number of columns with different names. When he selects “save” two maps will be displayed, one with all the areas and another one with the whole city calculated with the convex hull of all those areas, as a result we have a polygon with the union the areas which not allows blanks inside .

Title:	Provide a city division with a shapefile.
Description:	A experiment creator can create an experiment with a shapefile.
Primary Actor:	Experiment creator.
Preconditions:	The user has deployed the server.
Main Success Scenario:	1. The user names the city. 2. The user uploads the zip file with the shapefile. 3. User selects the right column of the shapefile.
Extensions:	1a. If the city already exists it is overridden. 3a. The result is displayed.

Frequency of Use:	Few times a year.

5.1.2. Create and manage cities and neighborhoods divisions manually

Some of the cities in the world does not have shapefile and we need a way to create studies for these cities. The areas and the cities can be created and managed in the similar way because both are polygons displayed over a map. When the experiment creator select Areas or Cities in the admin page a list of the objects in the database are displayed. The user can select an object to modify or click “add” to add a new object. A map of OpenStreetMap will be displayed. The experiment creator can make or manage zones. A zone consists in a bounding box. In the case of cities he just have to point the name of the city. For an area he has to point city name that it pertains. The experiment creator can delete the objects by selecting them and clicking delete in the dropdown box of the top.

Title:	Create and manage cities and neighborhoods divisions manually
Description:	A experiment creator can set up the city boundaries and its neighborhoods manually.
Primary Actor:	Experiment creator.
Preconditions:	The user has deployed the server.
Main Success Scenario:	1. User clicks “Areas” or “Cities” in the admin menu. 2. User clicks an object that already exists or add a new object. 3. User can make bounding boxes and specify the properties. 4. User clicks save.
Extensions:	2a. If the object already exists the map will zoom in to the bounding box. 2b. If the object does not exist a world map will be displayed. 3a. For cities he will specify the name. 3b. For areas, the city that the area pertains.
Frequency of Use:	Few times a year.

5.1.3. Make easy-to-guess points and random points for an existing experiment

The experiment creator will go to the Generate Points page and will select the city that he wants, he can select if he wants to delete the previous points. The experiment creator can select some of the 20 Points of Interest that he thinks that are interesting. We only display 20 because we have limitation in an external API. Now he has to select if the points have to be in the whole city bounding box or for each area of the city and the number of points. In the next web page an OpenStreetMap with the points will be displayed. He can remade the points and save them them.

Title:	Make easy-to-guess points and random points for an existing experiment
Description:	A experiment creator can create random points and the points of interes for the experiment.
Primary Actor:	Experiment creator.
Preconditions:	The user have deployed the server.
Main Success Scenario:	1. The user click on “Generate Points” in the admin page 2. The user select the city that he wants to add points 3. The user has to select the number of points and the type. 4. The users save points.
Extensions:	2a. Select if he wants to drop the previous areas. 3a. Points per area. 3b. Points per city.
Frequency of Use:	Few times a year.

5.1.3.1. Points of Interest

To motivate the users, well-known points will be displayed. Preliminary experimental results in the first version of Urban Opticon show that users get frustrated when they do not know any point of their city, especially when they have been living there for a long time. Therefore, to ensure right answers from the final user, some well-known reference points are introduced. These points will not count for the statistics because they are too obvious. In order to specify the reference points, the experiment creator can choose doing this manually or with our automatic process. This process will scan the areas in a certain radius of the center with Google Places API that gives the most relevant places given a point and a radius.

5.1.4. Create and manage Places

Users can add or change points clicking a new point in the OpenStreetMap, if the Google Street

View image is not found the image will not be saved. The Places can be related with a “City” or an “Area”. They have a boolean that indicates if it is a Point Of Interest.

5.1.5. World experiment points

The experiment creator can make a study for the whole world. The characteristics of the world points is that they are not related with a “City” or an “Area”. They can be created manually how we specified before or they can be auto generated.

When the experiment creator clicks on “Generate points” in the admin page he will be able to click on “World points”. There he has to specify the number of random world points. If he clicks “save” the points are saved in our database and in the images folder. Below that, the Points Of Interest for the world are displayed with an image for each one. The user can select the most interesting for him and save them.

5.2. Final user

The final user can only play the game, and his potential interactions with the system are:

- Look at the image displayed and select the zone that he thinks it is (ten every time that he plays).
- After each image get the answer and the score.
- Get his final score and publish his contribution in social networks.

These steps are listed below:

5.2.1. Point the correct zone

Every time that the user plays ten image will be displayed in turn. Users will point over Google Maps where they think it is the correct point.

5.2.2. Score

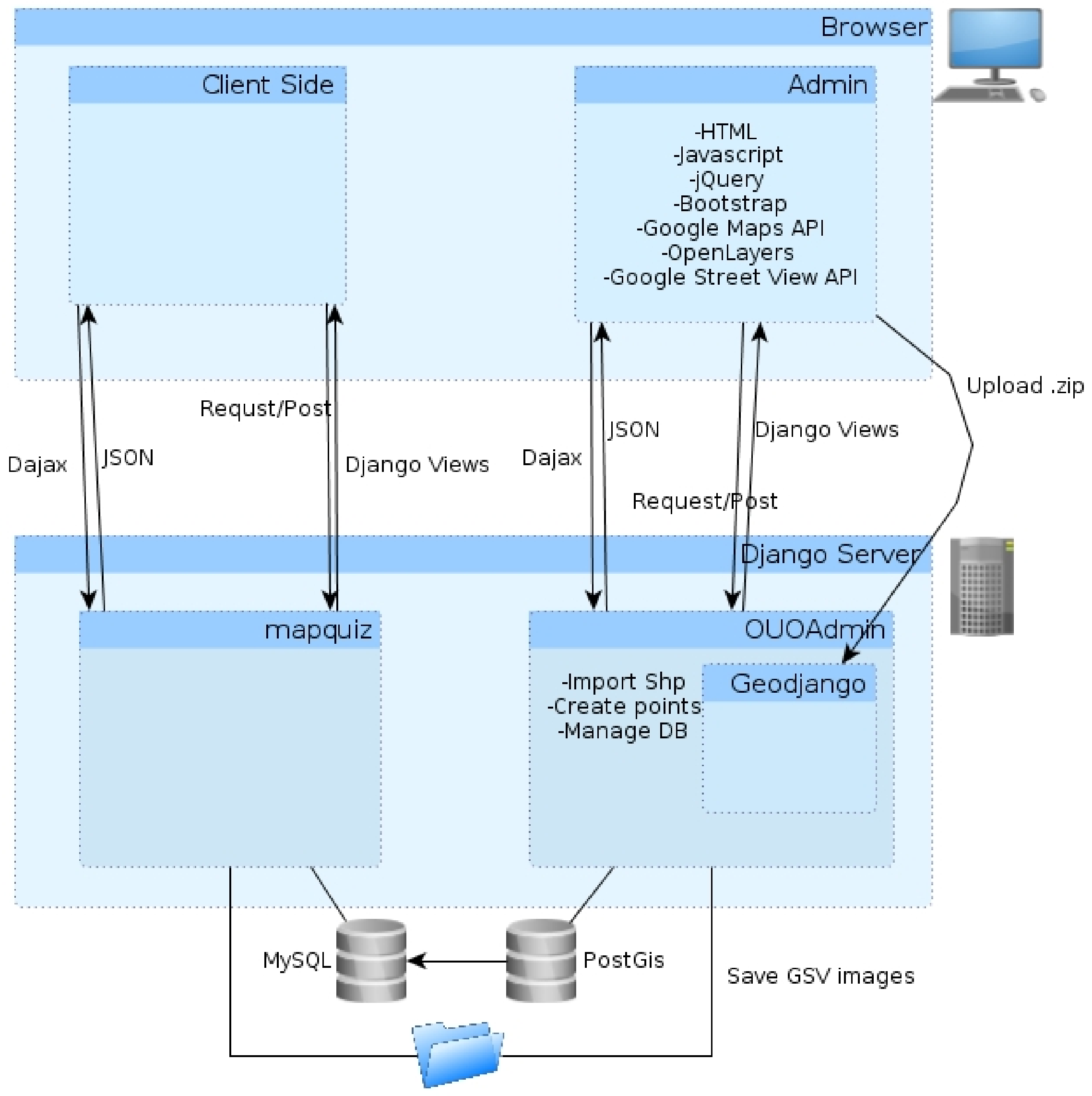
The score will be 100 point minus the relative distance from the correcting point. The score will be displayed in a pop up and if the user fails the name of the correct point will be displayed too. The final score will be displayed at the end.

5.2.3. Contribution indicator

At the end of the game will display a progress bar with a % of answers that the study have depending on the wish of the study creator. Final users will be able to put on their facebook their contribution on the study, for example: “You helped to study the recognizability of the neighbourhoods of Barcelona in a 0.3%”.

6. System Structure

In the next sections the different modules that we are used in our project. Moreover, the implementation of technological solutions for each module is explained. Here we have a little scheme with the technologies and below they are explained one by one.



(Figure 29. Technology scheme)

6.1. The language: Python

Python is general purpose programming language, which can be used to do anything. It integrates well with our project because it needs different sorts of things like geographical databases and web applications. The frameworks that python have are very useful for our application and that why we choose python.

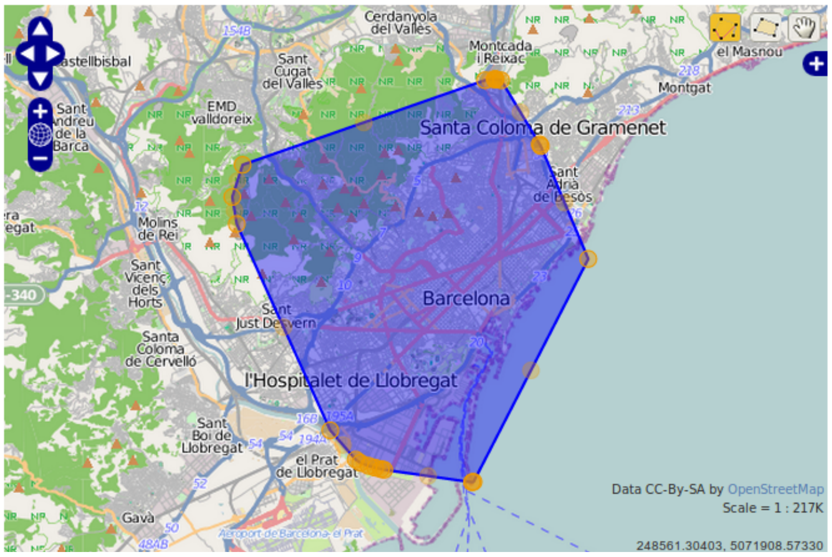
6.2. The web framework: Django

The previous work done for Urban Opticon was made on Python and with the Django framework we can improve the project. It helps in creating and managing the database with its object-relational mapper and the automatic admin interface. To create the database we only have to define the objects in a file with a very simple syntax and run a synchronization command, it is not need to use PostreSQL. Then, if we need to manage the database, the reference of the register can be managed as an object (you can call methods, get and set variables...). The administrator interface it is generated only pointing which objects of the model you want to manage.

6.3. The geographical framework: GeoDjango

GeoDjango it is another framework that uses Django as a base. We build our server with GeoDjango. It gives to Django a Geographycal Interface System (GIS) that have helped a lot because as django do, it adds to the automatic admin interface geographical features like maps and a geospatial database, for example, PostGIS.

The automatic interface allows the user to draw and modify polygons and points over a map.



(Figure 30. GeoDjango polygon edition)

The GeoDjango GIS have some useful implemented methods that we use, like contains(point) where you can check if a point is inside a polygon or convex_hull() that returns a convex hull of a bunch of polygons.

Geodjango have an utility named LayerMapping that allows to upload shapefiles and save them on the database.

6.4. Asynchronous communication: Dajax

Dajax is a ajax library for Django that permits to make calls to the server without having to reload the whole page. That is useful because it permits an easy asynchronous communication between the client and the server which is hard to do only with Django because it has to make a request to the server and refresh the page even if you do not need it.

Basically the client makes a call in javascript and this call give to the client a callback that you can process or not.

6.5. Creation of dynamical select list: Javascript

Javascript is a vastly used scripting language for the web and it have a lot of libraries which we are using and allows to manage the HTML elements easily, for example, create dynamic lists of select box.

6.6. Display geographical objects: OpenLayers

We are using OpenLayers for the maps. OpenLayers is an open platform that make it easy to display dynamical maps on a web. Geodjango uses OpenLayers too in the display of polygons and points.

OpenLayers works with layer where you can add different things. For example we can add a OpenStreetMap layer that displays a lot of information like boroughs, street names, places, etc. In our application we use different layers to display the polygons of the cities and the points that have been generated.

With OpenLayers you can configure the events related with the map, for example, the mouse over the features of a layer or control the click on a zone as well as enable or disable the movement.

6.7. Get relevant places: Google Maps API v3

We use this API because it is possible to make a request of the twenty most interesting places given a point and a radius. The API ever returns twenty places, there is no possibility to point the number that you want. The places object that it returns have the coordinates, the name, the classification, etc.

6.8. Get images from a coordinates: Street View Image API

This API give us the nearest Street View Image given a certain point. It gives a static image that we save in a folder. Sometimes, if there are not a close image it returns a ‘no image’ image, but we have managed to not save it.

6.9. Events, manage elements...: jQuery

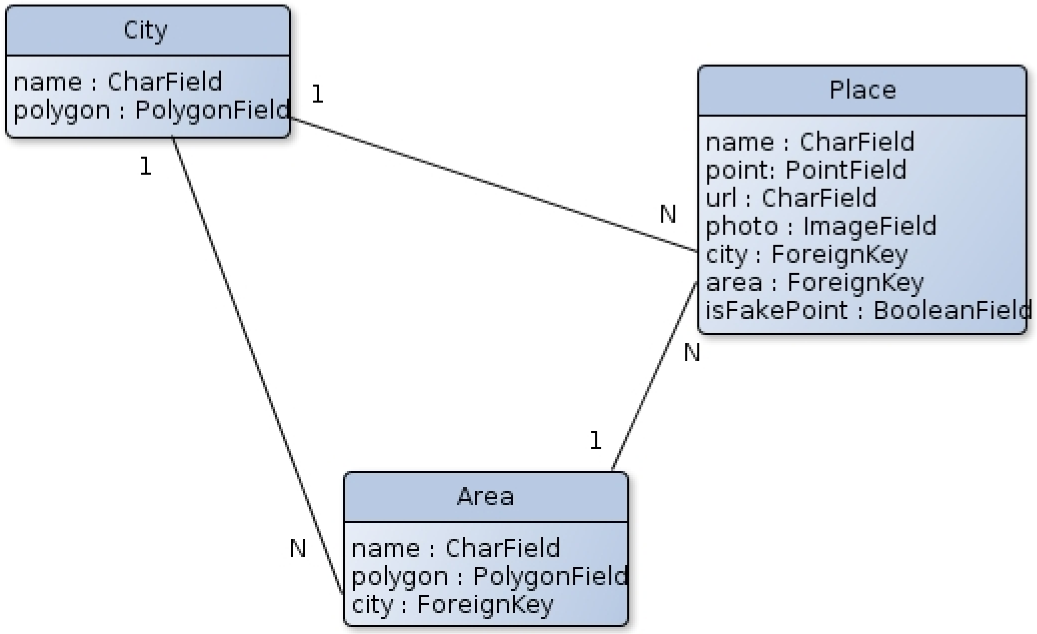
jQuery is a javascript library that simplifies javascript programming and allows to configure the HTML page easily and make events. We use it for hide and show elements of the pages, read values of inputs, etc.

6.10. Stylize the front-end: Bootstrap

Bootstrap is a front-end framework that helped us to personalize our application with nice buttons, a background and other stuff. You only have to define a premade class of Bootstrap your HTML tag (for example, class=”btn”) and the framework do all the work.

6.11. Geographical database: PostGIS

The database that we are using is PostGIS, that is a PostgreSQL database with geographical objects. This database allows to manage points in real geographical spaces like points or polygons.



(Figure 31. Entity-relationship diagram)

There are only three tables that we had to define in the database, the other tables that django needs (users and other settings) are made automatically.

- City table only contains name of the city and polygon fields.

- Area table have the name, the polygon and the relation with the city. The city can not be null because an area ever pertain to a city.

- Place table have the name and the point. Furthermore, the url which is the url of the Google Street View API, the photo which is a custom field from Djnago that allows to upload images in a certain folder, the city, the area and a boolean that points if it's a points is an easy-to-guess point or not. If a Place do not have city it is considered a world point.

7. Technological Challenges

In this section we will talk about the problems that we found during the programming process and the solutions that solves these problems. For example, the problems with the asynchronous communication between the client and the server, the different projections for the geographical objects, the images of the Google Street View Image API, the upload of the areas with a shapefile or the problems that we had to solve to generate world points.

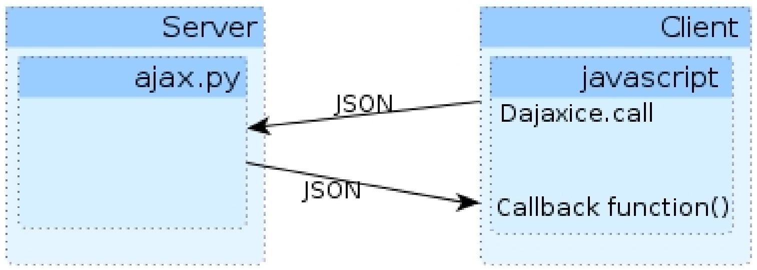
7.1. Django asynchronous communication

Django, for security reasons does not allow the asynchronous communication between the client and the server. Instead of asynchronous communication Django offers POST functions that allows to pass parameters with a HTML Form. That solution is good if you are thinking in change the web page but not very much if you just want to call an easy method that does not need to render the page one more time.

As solution we found is Dajaxice from http://www.dajaxproject.com/. Dajaxice is a project made to help the developers of Django to have an easy communication between the server and the client. It separates the server part and the client part.

The server part is just another application that you have to add to the project. Then, we have to add the url of Dajaxice and finally we have to make our *ajax.py* file. In this file we have to name the method and we can do whatever we want (change objects in the database, mathematical computations, etc.). Then we have the option of respond something, usually some data in JSON format. For example, a function that we made, *generateRandomPoints* calculate a given number random points in a Polygon, check if there are Google Street View Image and the save them. As response we have a JSON with all the points.

In the client-side everything is made with javascript. First we have to import the javascript libraries that we need, however, this task can be done easily since we just have to use the template language of Django. The complexity to call one of the methods that we have in the ajax.py is reduced, as we just have to put *Dajaxice.adminShapes.ourFunction(callback, {'data':data});*. *Callback* is a function that we have to create in order to receive the information that the functions returns. The other parameter is the data that we send to the function. Following the previous example, we call the function *generateRandomPoints* and it calls the javascript function *callbackRandomPoints* that draw the points to the OpenLayers map.



(Figure 32. Dajaxice functional scheme)

7.2. Projections

The map projections are the transformation between the sphere to a plain. In the case of the earth, normally the projections are represented with longitudes and latitudes (coordinates) but there are calculated in different way depending on standard used. The main differences amongst standards are the different usage of scale coordinates, the type of the map (sphere, ellipsoid...) the reference points, etc.

The main problem that I found for our application is that the PostGIS database uses one kind of projection named EPSG:900913 that manages the earth as an sphere and the Google Maps and the OpenLayers uses the EPSG:4326 that manages the earth as an ellipsoid.

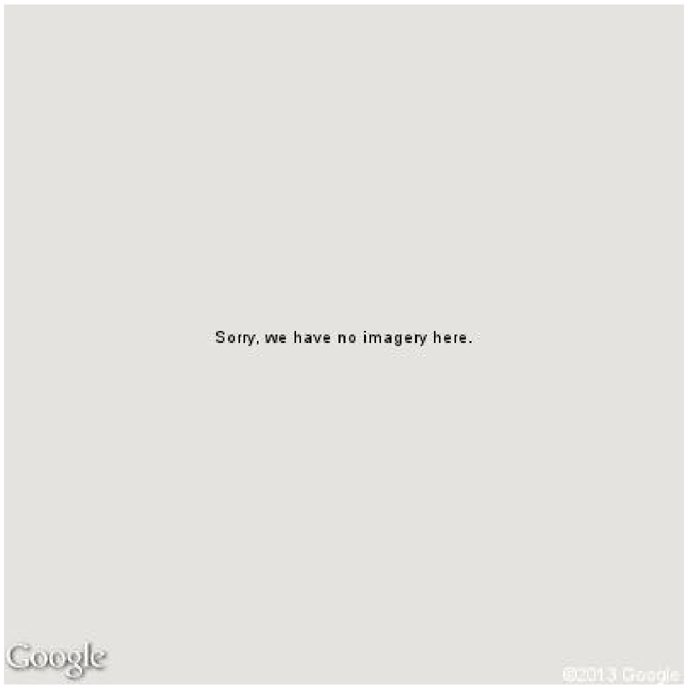
To solve this we found that all the geometries that we are using in the OpenLayers can be transformed with a self method giving two projections. Since all the methods that Google Maps are used simultaneously with OpenLayers we can use it too.

7.3. Save and compare images of Google Street View

The Google Street View Image API works providing some parameters to its URL and returning a Google Street View static image. The parameters that we input are the size and the coordinates and it gives back an image corresponding to the GPS coordinate provided that we want to save in an image folder inside the project folder.

To save these images into the folder, we had to create a field in the Place object that was a image. That allows the administrator to upload an image but we had to find an automatic way (without downloading the image and then uploading to the server). To do this I found that it is possible to upload an image with a Python library named urllib only with the url. Then we can save this image in the image field.

Another big issue that we found was that the input coordinates provided to the GSVI API is too far away from a GSV spot it returns a 'no image' and we do not want points that can not be displayed to the final user. For example the coordinates (10.45557,79.30531) in India do not have GSV image.



(Figure 33. 'no image' response from Google Street View Image API)

To solve this we had to compare each image with the 'no image' and decide whether to save it or not. But the problem came when it is not possible to compare an image that is not saved. To solve this we had to save first the image, compare and then delete it if it is 'no image'.

Another problem that we found was the asynchronous communication of the Dajaxice. We had been using a method where we gave the point and it saved the image if it was not a 'no image'. We want this image displayed as a point in the map. When we wanted to read the responses we found that the method returns a callback before execute the whole method, it just responded the same very fast for each object without checking if there was image or not. We solved this passing the zone and the number of points to the server, it makes the points and it respond only with the points with GSV image.

To disable the possibility of saving and modifying the point of a Place without checking the 'no image' we had to override the saving method, allowing it to check if there are image and if there are not an error message is displayed and the Place not saved or deleted.

7.4. Import Shapefile

Geodjango give us the possibility of generate objects thought shapefiles. A shapefile is a *de facto* standard of spatial data that is saved in multiple files. A shapefile can contain different kinds of spatial forms such as points, lines and polygons in only one shapefile. Geodjango can upload shapefiles with an utility named LayerMapping.

Shapefile multiple files:
.shp — shape format; the feature geometry itself .shx — shape index format; a positional index of the feature geometry to allow seekin forwards and backwards quickly .dbf — attribute format; columnar attributes for each shape .prj — projection format; the coordinate system and projection information

The first problem with LayerMapping is that it is the user needs the basic files (.shp, .shx, .dbf), the user needs the .prj file. This file gives the projection in the geospatial space, without this, you only have the polygon with some properties.

In our case, we are only interested in upload polygons geometry (not points) and we have to filter the columns that we only have polygons. Another problem related with the shapefiles is that the columns names do not follow any standard in their naming, therefore the user have to know what it means the name of each column. For example, in the shapefile of Barcelona the name of borough column is "NBarri" which is not very easy to translate for people who do not know the language.

7.5. World Points

The first problem with the world points was an obvious consequence of getting random images of Google Street View: the vast majority of Google Street Images of the world are from roads roads are not very interesting in our study neither for the user experience. To fix this we have made an static list of the most relevant cities in the world and we take the random points from these cities given the center and a radius.

Since it is impossible to have the well-known points from the Google JS API because the radius that we have to give is very big (the radius of the earth), we search the center point from the list of relevant cities and we put the radius that covers the city. We have copied this points in a static list and deleted those points that have the ‘no image’.

8. Economic valuation

8.1. Technologic resources

Our project is just the administration part of a whole server that will be Open Urban Opticon. In fact, we do not need to deploy it to a dedicated server to run and configure it, just with the django debug server. Then, the task of deploying the server and the cost of the deploying is not evaluated here.

All the software that we have used is open source including the code repository. The restriction that we have is the Google Maps Javascript v3 API that have 25000 request/day but we had never reached the max since we have to call 25000 times the web-page of generate easy-to-guess points, which is not realistic or 3000 times the web-page world points. The Google Street View Image API have 25000 requests/day, implying that we have more or less a 70% of chances to find a the image given a point (that means 30% of 'no image'), it is possible to save 17500 points every day, large enough for the normal use of the application. In both API if you want to add request, every 1000 cost 0,50$.

8.2. Human resources

My work and the work of Daniel Villatoro do not need to be considered so we do not have to count it for the price. The estimated cost of a researcher as Daniele Quercia, my tutor out of the university, is 75€ per hour, he had worked for 26 weeks, two hours a week. That makes 3900€ in total.

26 weeks x 2 hours per week x 75 € per hour = 3900 €.

8.3. Total cost

The total cost of the project raises up to the budget of **3900 €**

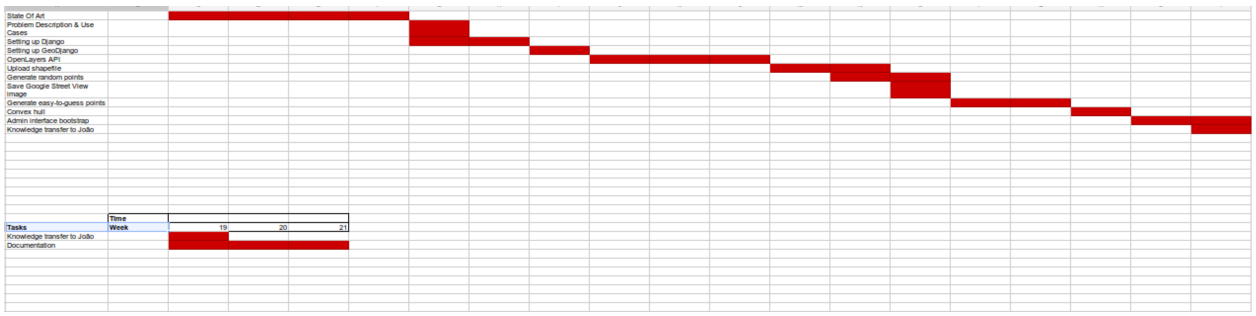
	Cost(€)
Technological resources	**0***
Human resources	**3900**
Total	**3900**

*With a normal usage of the application

9. Development Methodology

In the development of the Open Urban Opticon we have followed the Scrum methodology for software developing, which it’s flexible, iterative and incremental. Each iteration have a marked duration and it is known as Sprint. In our case, the sprint have a duration of a week. During the weekly meetings we have put short-term objectives with due dates. We have been in contact almost every day via email.

As Product Owners I have got my tutors Daniel and Daniele. They have had the role of Scrum Managers too because they have been managing the project tasks.



(Figure 34. Gantt chart screenshot)

In this link you can find the Gantt chart

https://docs.google.com/spreadsheet/ccc?key=0Ao4cJMiUGpbKdFJNb3VpTElRLUQ2R2U4V1JhY3QwR1E&usp=sharing:

10. Future work

In this section the potential improvements will be described. This section is divided in short-term work, which can be enhanced without making a lot of changes to the original project, and long-term work, that involves an exhaustive study of the viability and another point of view of the project.

10.1. Short-term work

10.1.1. Improve the city study creation

The first improvement that can be done it is to add a guided way to add a city in an orderly manner. Now this is already done but users have to go to different pages without an order. With orderly way I mean be able to draw a city, then draw the different areas and finally add the easy-to-guess and the random points. In fact, you can already do this but you have to go to the "Manage model" and first add a city, add each area one by one but then you can use the "Generate Points" to do it. We could do this more user friendly.

To make this we should think in different things:

- If we should allow users make their own areas.
- Divide areas automaticly with a grid.
- Let users draw areas outside the city polygon.

10.1.2. Make the world points dynamically random

Now, the world points are static. We have a list of the most relevant cities in the world in terms of population, growing and well-known. Then we have another static list of the easy-to-guess points of these cities, but some of them have high restrictions in Google Street View (specially chinese cities) and we had to reduce the number of points. The random points are also taken from these cities given the center and a radius. We had to make this static list because if we take random points around the world mostly of them was roads that do not have too much interest in our experiment.

To solve this we have different options, one could be to do a very large city list with the coordinates, for example, all the cities in the world with more than 50.000 inhabitants. To find this kind of list, we would need a web crawler and a web with accurate data. One of the website where we could obtain this data is dbpedia where the data is in a computer-friendly format.

The only way to that I can think to make the automatic comprovation of if an image is a road is look at every image and study the color histogram. The more grey an image have, the more possibilities have to be a city. Another analysis could be, if the image have more grey in the bottom and less in the top, it could be a road. There are a few problems, for example, if it is a park (the image will have no grey but is an interesting point) or if it is a street with vegetation (grey in the bottom and no grey in the top).

10.1.3. Filter the ‘no image’ in the easy-to-guess points

Now, in the easy-to-guess points ‘no images’ are displayed. That is because Google Street View Image API do not check if there are images or not. I could find the way to give a good response from the server (where the images are compared) and do not display it because the asynchrony of the communication.

The most obvious way to correct this is catch all the points in a list and pass them to the server. The server can check if there are or not image and respond with the list with only the list of points that have image.

10.2. Long-term work

10.2.1. Phone and tablet app

I think that platform can be easily exported to be used in mobile phone or tablet since it does not require too much user interface. We would not need any kind of graphic engine because with Google Street View and Google Maps is enough. The server engine could be even easier than now. With a RESTful API should be nice because we just need to pass the coordinates and not too much information more. The app should be able to render everything since the code is executed in the client side. A good option to code the client side could be jQuery Mobile because it works in a lot of devices.

Another interesting feature that a portable device adds it is the possibility to get the position of the user with the GPS and display places around him. That could give us very interesting data and make complete different game.

10.2.2. Monetization

I have been thinking in the possibility of adding an economic profit to our project and I think that the best way to make this project profitable in economical matters could be charge small amount of money from business that interested in the visibility of their company. That could be very easy to deploy and the user should not know that the data will have a commercial benefit and he will continue enjoying the game, which is the basis of the implicit crowdsourcing.

11. Conclusions

The idea was to make a platform that could be used for every city in the world that displays images from the Google Street View and the user have to guess where is that point. We had different approaches of the project. First was a server per city but then we decided to do *n* number of cities per server. Finally, we added the possibility of world points.

I have worked with a Master Student of Brazilian university named João Paulo Pesce. He made the previous Urban Opticon that was made for London. In the middle of my project he decided that he wants to join to our project making the final front-end gamers It allows me to focus on finishing the administrator part ensuring a good result. I made the whole administrator part, almost everything that you can find in the OUOadmin/ folder and everything that I described before and all that is explained in the "Administrator manual".

João has made the final user interface part, where the user can see the Google Street View and click in the Google Maps. He have made the part of saving the results too over my database.

In conclusion, our project will help in the challenge of discovering the imaginary map of the points that are easy to recognize by the user. The population of the cities in the world are growing very fast and our application can help people who have to manage this. Our administration, combined with the interface will make a full server application capable to make a great user experience and impressive results that will be used in other studies related with this subject and will be easy to deploy to a non-expert user.

The administrator part cover the needs of an experiment creator that wants to do different experiments of different cities (or the whole world) in different ways. We have made an easy to understand interface that allows non expert computer user capable to create a study following the instructions and, since it is a open source project, highly scalable and easy to customize.

For me, in the engineering perspective the project has been a great experience. I have learned how to be able to manage a large project, structure the goals and study the different possibilities. I have learned how to work with requirements from another person and how important they are. I had to study which technologies are more important and how they can help the project. In this aspect, I have learned how different APIs, maps, web frameworks and Javascript works I now I can make a imaginary structure in my mind that can be useful for any other web project.

In the investigation part, I have learnt how important the crowdsourcing and smartcity projects are and how much they will be in the future. I have learnt how to inquire into a project and find its pros and cons and gather information that can be useful.

In general, it has been a great project and I am sure that I will use mostly all the things that I have learned very frequently.

12. References

- https://dl.dropbox.com/u/6314563/papers/milgram/psychologicalMapNYC.pdf
- http://urbangems.org/
- http://urbanopticon.org/
- http://pulse.media.mit.edu/
- https://dl.dropbox.com/u/6314563/papers/www13_map20.pdf
- http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- http://www.udemy.com/blog/modern-language-wars/
- http://www.techrepublic.com/article/python-in-the-enterprise-pros-and-cons/1045768
- http://symfony.com/
- https://www.djangoproject.com/
- http://www.careerride.com/python-disadvantages.aspx
- http://en.wikipedia.org/wiki/Django_(web_framework)
- http://en.wikipedia.org/wiki/Ruby_on_Rails
- http://rubyonrails.org/
- http://en.wikipedia.org/wiki/SQL
- http://twitter.github.com/bootstrap/index.html
- http://www.getskeleton.com/
- http://jquery.com/
- http://buhrmi.tumblr.com/p.../how-coffeescript-makes-jquery-more-fun-than-ever
- http://coffeescript.org/

- http://lesscss.org/

- http://haml.info/

- https://github.com/djagdish/Place-Pulse

- http://www.alumnifutures.com/2012/07/crowdsourced.html

- http://en.wikipedia.org/wiki/Crowdsourcing

- http://mashable.com/2012/09/07/crowdsourced-developers/

- http://www.kickstarter.com/

- http://www.newrepublic.com/article/politics/magazine/110225/the-false-promise-kickstarter#

- https://www.mturk.com/

- https://requester.mturk.com/mturk/beginsignin

- https://en.wikipedia.org/wiki/Amazon_Mechanical_Turk

- http://www.superhub-project.eu/

- http://hoyrespiro.people-project.eu/

- http://www.ushahidi.com/

-http://e360.yale.edu/digest/growth_of_urban_areas_poses_long-term_threats_study_says/3095/

-http://www.superhub-project.eu/index.php?option=com_jdownloads&Itemid=20&view=finish&cid=16&catid=6&m=0

-http://www.kickstarter.com/blog/hiring-rails-developer

- https://github.com/jpesce/urbanopticon

- https://github.com/A-Barwell/urbangems

- https://github.com/ushahidi

-http://www.screamingatmyscreen.com/2012/2/business-decision-why-i-use-django-and-not-rub y-on-rails/

http://mundogeek.net/archivos/2007/08/20/ruby-on-rails-vs-django/

- http://www.dajaxproject.com/

- https://docs.djangoproject.com/en/dev/ref/contrib/gis/

- http://dbpedia.org/

13. Annexes

13.1. Code Review

Open Urban**Opticon** Admin

Code Review

13.1.1. Python files

13.1.1.1. models.py

Here the entities used in the application are defined.

First it detects the path where the application is because it might be deployed on different operating systems or directories.

13.1.1.1.1. City

A **City** has a name and polygon, the plural had to be redefined because the default was displaying “Citys”.

13.1.1.1.2. Area

An **Area** represents a region of a particular city, it has a link with the city it belongs to, a name and a polygon.

13.1.1.1.3. Place

A **Place** represents a point to be guessed by the user. It has a geographical point (with Latitude, Longitude), the URL to its photo in Google Street View (GSV), a link to the same photo (but stored locally in /image), links to the city and area it belongs to and a boolean that tells if it is an easy-to-guess (fake) location or not.

The *cache* method locally saves the image contained in the URL field. First it saves the image, then it compares it (using the *is_equal* method) with a previously downloaded GSV 'no image' file. If *is_noimage* returns TRUE, the saved image is deleted along with the object (using the deletePhoto method).

A **Place** will have a link to both a city and an area if it has been generated using the "X Points per Area" or only to a city of the "X Points per City" option was used. A Place without area and city is a "world point". There are two extra methods, one to reset the city of the Place checking in the cities that we have and the other one the same but for areas.

The **ShapefileZip** is an auxiliary class that allows the administrator to upload a zip archive with all the files necessary for the import (.shp, .prj, .shx and .dbf). It overrides the save method to have more utilities on it. First, it deletes all the shapefilesZip objects in the database if they already exist. It saves the zip in the /shapes folder and extracts it, then saves the name of the path and finally deletes the zip only leaving only the files.

13.1.1.2. admin.py

This file initializes and customizes the administration pages.

The filters (list_display) contains the attributes of the models that can be displayed and ordered in the admin page.

save_model overrides the saving method. Now, before saving the new place, it tries to delete the place if it already exists (in case you are changing the position of a place), then checks if there is an image of this place in Google Street View (obj.cache()) and if there isn't, doesn't save it and display an error message.

13.1.1.3 views.py

This is the logic of the application. It communicates with the models, perform various actions and prepares the data for the templates.

@staff_member_required only allows logged in administrators to use the view.

The view **generateetg** is a view that generates the easy-to-guess (fake) points of a city. First it deletes all the places that already exist for the city. It passes the city name and the polygons to the *generateetg.html* template. (From *createCity* it receives only the city name and from *generatePoints* it receives a list)

The view **generateRandom** passes to the *generateRandom.html* template either all the areas of a city (If the selected type was area) or the whole city polygon so it can generate random points inside them.

The view **importShapefileNewCity** is a view that, depending on how it's called (via GET or POST), does different things. If the method is a GET (means the user hasn't sent the shape file yet) it returns to the template a form for the zip file and a the city. If it's a POST (user has just sent the shape file) it saves the zip file and returns the columns of the shapefile and the administrator will be able to select the most interesting for him.

The view **importShapefile** does exactly the same as the previous view, but returns a different HTML. It is used if the city is already in the database. It adds the new areas to the ones already in the database and recalculates the city polygon.

13.1.1.4 ajax.py

This file contains the methods that are called dynamically via ajax. The server processes the request in real time and returns the desired information without having to reload the whole page. (For more info see dajaxproject.com)

The method *saveShapefile* defines the polygons of a city and its areas given: (1) the column that contains the polygons inside the shape file, (2) the path of the shape file and (3) name of the city. First, if there is already a city with that name, it deletes it along with its areas. Then it saves the new areas, put all the polygons in a list and make a convex hull out this list for the new city. Finally, it saves the new city and links all the new areas to it. It returns the areas and the city's polygons because when it returns, the template displays two maps: one with the areas and the other with the city polygon.

The *saveExistingShapefile* does the same as the previous method but considers the previous areas that the city has and just add the new ones without deleting them. It recalculates the polygon of the city with all the areas.

The *generateImage* method generates a new Place given the coordinates, it is used for the easy-to-guess points. If it doesn't have an image in GSV it isn't saved (cache() method). If it is a world point it is saved without a city, otherwise it uses the city passed as a parameter (data[2]).

The *generateRandomPoints* method generates random points given the number of points, the city and if the administrator wants to make points for the whole city or per area. It returns a string with the points that have been created to put them into a map.

The *get_random_point* is a helper for the above method that generates a random point inside a given box.

13.1.2. Templates

13.1.2.1. index.html

Overrides the index of the Django administration to add new buttons and hide the database manager. It uses Bootstrap.

13.1.2.2. base_site.html

It overriddes the base_site to change the name that it's on top, now it displays “Open Urban Opticon” instead of Django Administration.

13.1.2.3. createCity.html

It is the template where the administrator gives the name to a new study and select the way that he wants to create the city (now it has only shapefile).

13.1.2.4. importShapefileNewCity.html

In this template the user will select the zip file with the shapefiles and a column from the list of columns inside the shapefile. Two maps (areas and whole city) with the results will be displayed.

13.1.2.5. generateetg.html

This template displays a list with the well-known places name and images in the city that the administrator have selected. He can check the points that he wants to save and the images will be saved in the /images folder. After that, the user tells if he wants to generate points per area or per city and the number of points.

13.1.2.6. generateRandom.html

A map with the points that have been generated is displayed. The user have to wait for them to finish. If he wants he can change the number of points and regenerate them.

13.1.2.7. generatePoints.html

This template allows the user to change the points of a city. He can choose if he wants to delete the previous points or just add. The user has to select the city and then follow the steps like *generateetg.html* and *generateRandom.html*.

13.1.2.8. importShapefile.html

In this template the user can select a city and add areas with a shapefile like in the *importShapefileNewCity.html*.

13.1.2.9. worldPoints.html

Read from JSON different cities in the world and display their easy-to-guess points that the admin can select. It allows to generate random points inside this cities.

13.1.3. adminUtils.js

13.1.3.1. createMap

It displays a new map without controls. It has an OpenStreetMap layer that gives more information than a standard map. The vector layer is the vector that has all the polygons that are added as features but they have to be transformed to another projection. The zoom is set at the whole vector layer extension. An event is called whenever the mouse is over a feature in the vector layer.

13.1.3.2. initEtg

Create a map as it’s done in the previous method but with the center point (same as the previous map) and a radius (given by the distance between the center point and the top) make a request to the Google Places of Google Maps API to check the twenty most relevant places. The map is not displayed.

13.1.3.3. createMapRandomPoints

It follows the same way as *createMap* but it adds a new layer for the points that will be displayed after they have been created.

13.1.3.4. callbackGoogle

It is the function that the Google request calls, it fills a checkbox with the results that have been responded.

13.1.3.5 saveETG

This function saves the points that have been selected by the user in the checkbox that we made in callbackGoogle.

13.1.3.6. callbackRandomPoints

Reads the points that ajax responds and add them into the points layer of the map.

13.2. Administrator manual

Open Urban**Opticon** Admin

Administrator manual

13.2.1. Setting up the database

The database that we are using is PostGIS, that is a PostgreSQL database with geographical objects.

To set up the database first you need to install the Geospatial libraries that you can find here https://docs.djangoproject.com/en/dev/ref/contrib/gis/install/geolibs/. Please, follow the instructions depending on your OS.

Then you have to install the PostGIS database in your computer building it with these steps https://docs.djangoproject.com/en/dev/ref/contrib/gis/install/postgis/#building-from-source.

Once you have all of these, you can create the database running this commands:

```
createdb OUO
psql OUO
> CREATE EXTENSION postgis;
> CREATE EXTENSION postgis_topology;
```

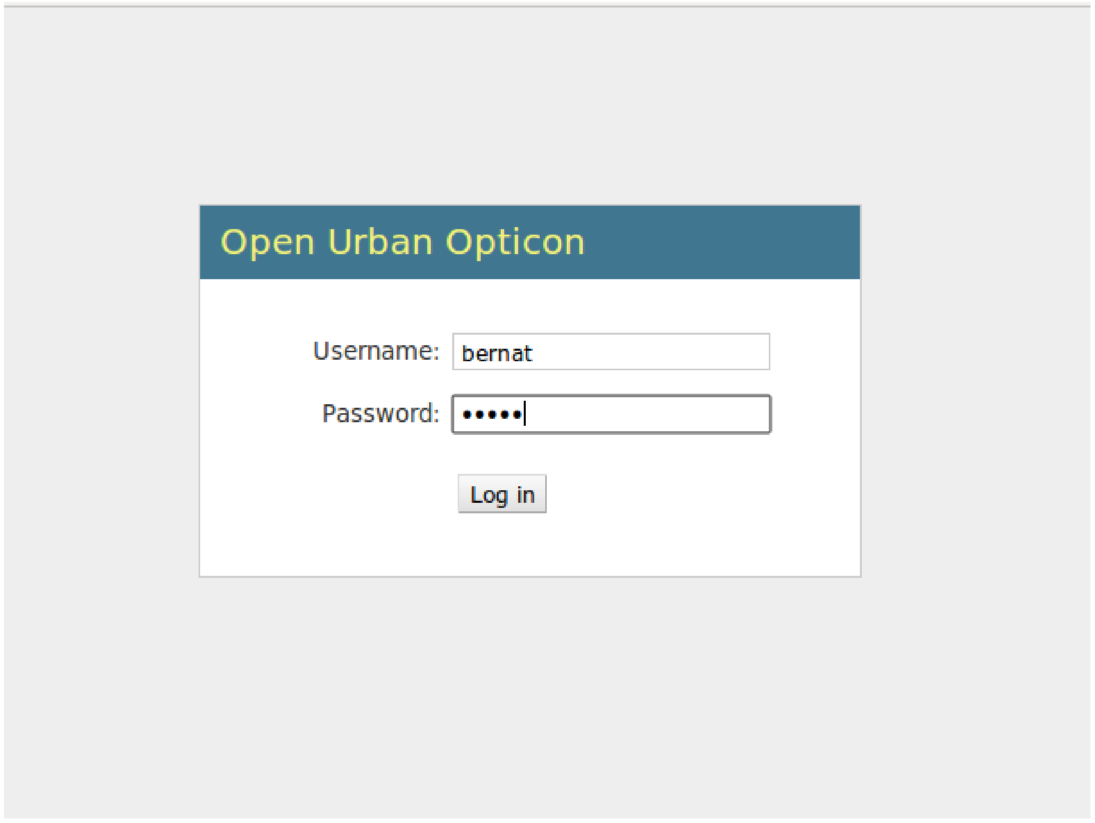
Your database has been created.

Now you have to synchronize the database. That will create all the tables that the application needs. In /urbanopticon/OUOadmin execute the command python manage.py syncdb . You will have to create a new administrator user with new credentials.

Now your database is ready to use.

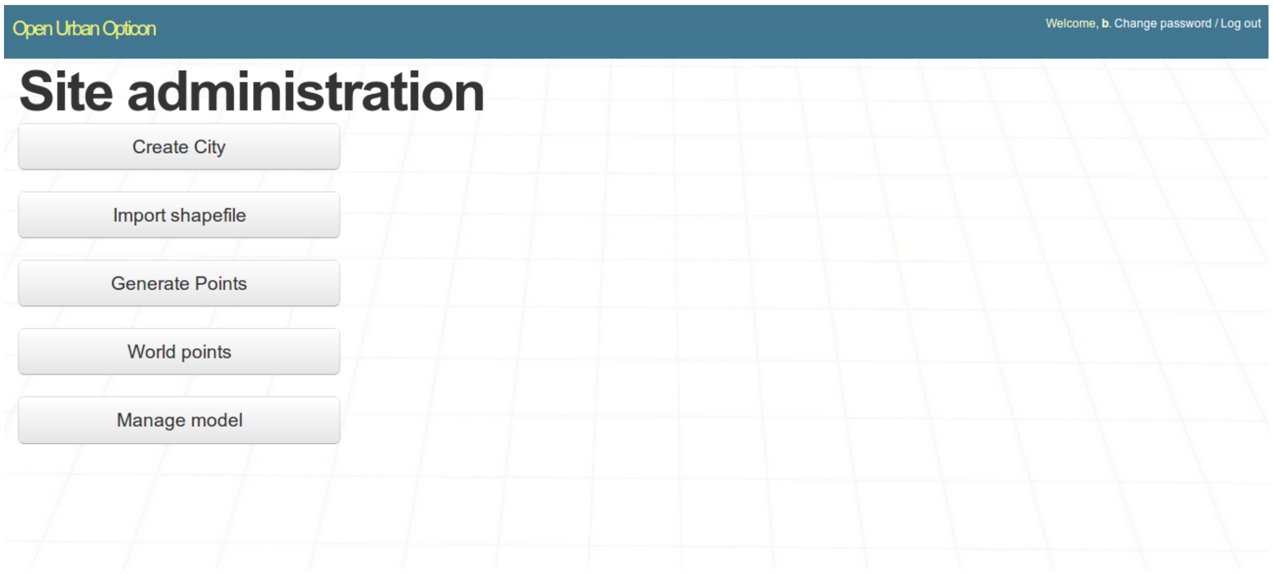
13.2.2. Login and index

Now you can log in with the credentials you set before.



(Figure 35. OpenUrbanOpticon login screenshot)

The administrator index will be displayed.

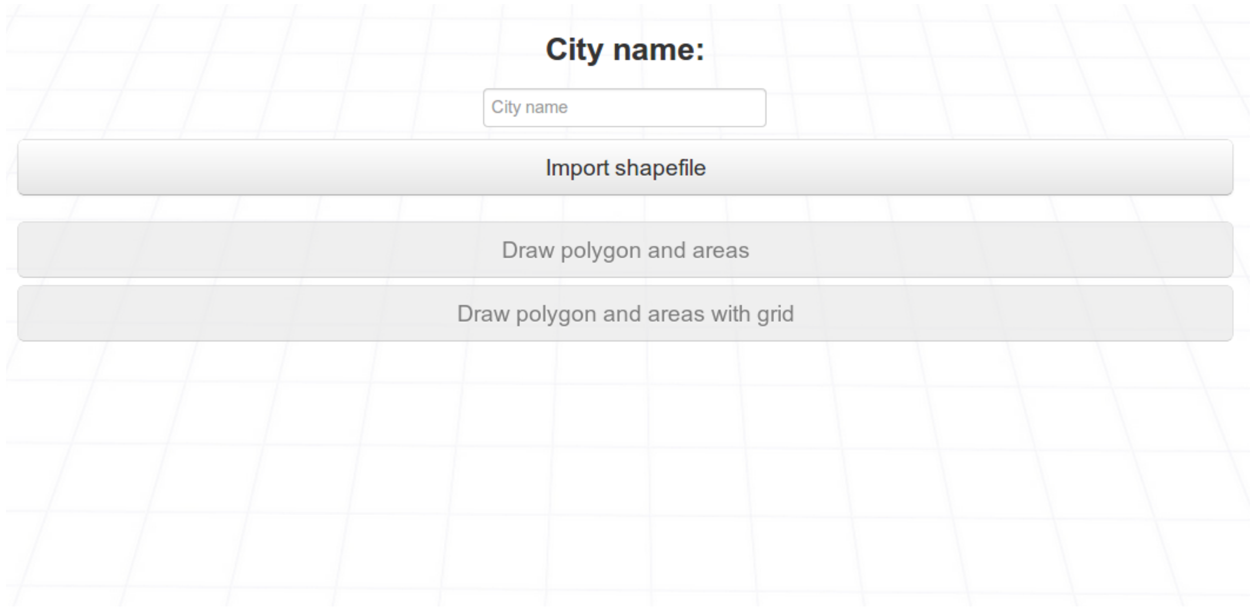


(Figure 36. Site administration screenshot)

At the top-right corner you can change your password and log out.

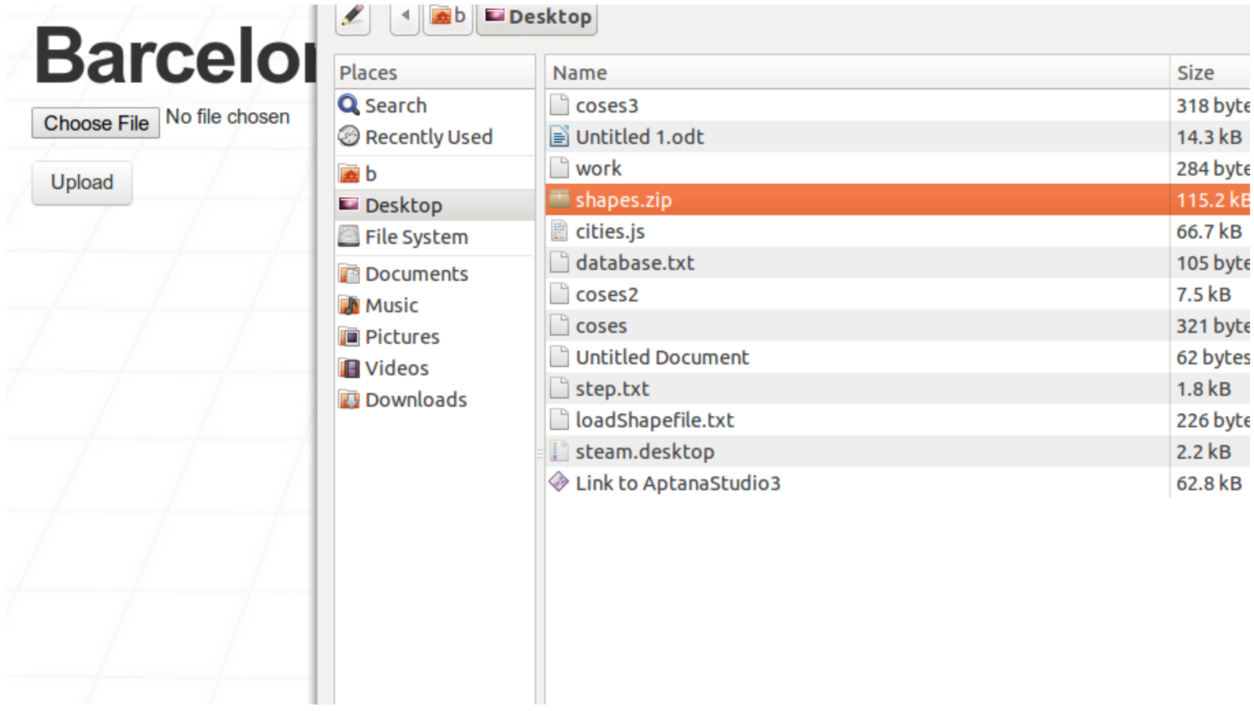
13.2.3. Create city

Now you can go to “Create city” to create a new city experiment.



(Figure 37. Create city screenshot)

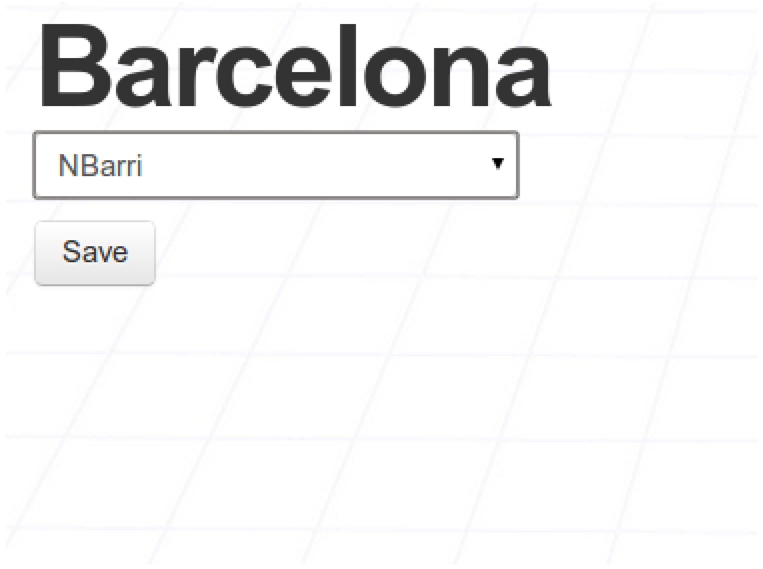
Here you have to name the city. If the city already exists you will be warned and if you want to override it you can continue. Then we click “Import shapefile”, the other options are disabled now.



Name	Size
coses3	318 byte
Untitled 1.odt	14.3 kB
work	284 byte
shapes.zip	115.2 kB
cities.js	66.7 kB
database.txt	105 byte
coses2	7.5 kB
coses	321 byte
Untitled Document	62 bytes
step.txt	1.8 kB
loadShapefile.txt	226 byte
steam.desktop	2.2 kB
Link to AptanaStudio3	62.8 kB

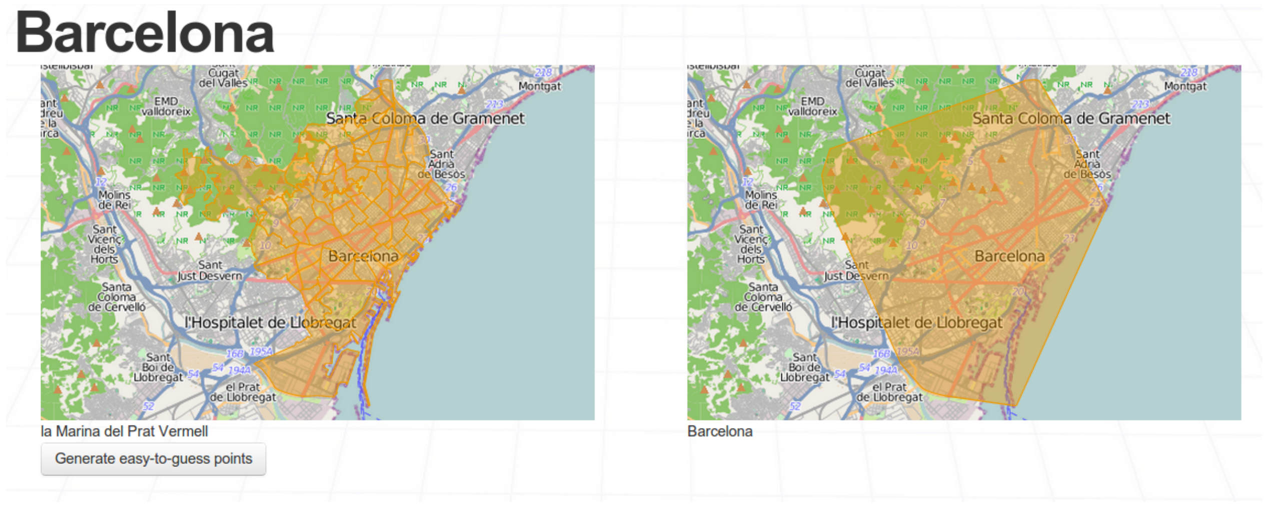
(Figure 38. Upload shapefile screenshot)

Now you have to select a zip with all the types of file that the import needs (.shp, .prj, .shx, .dbf) and click Upload.



(Figure 39. Choose column and save shapefile screenshot)

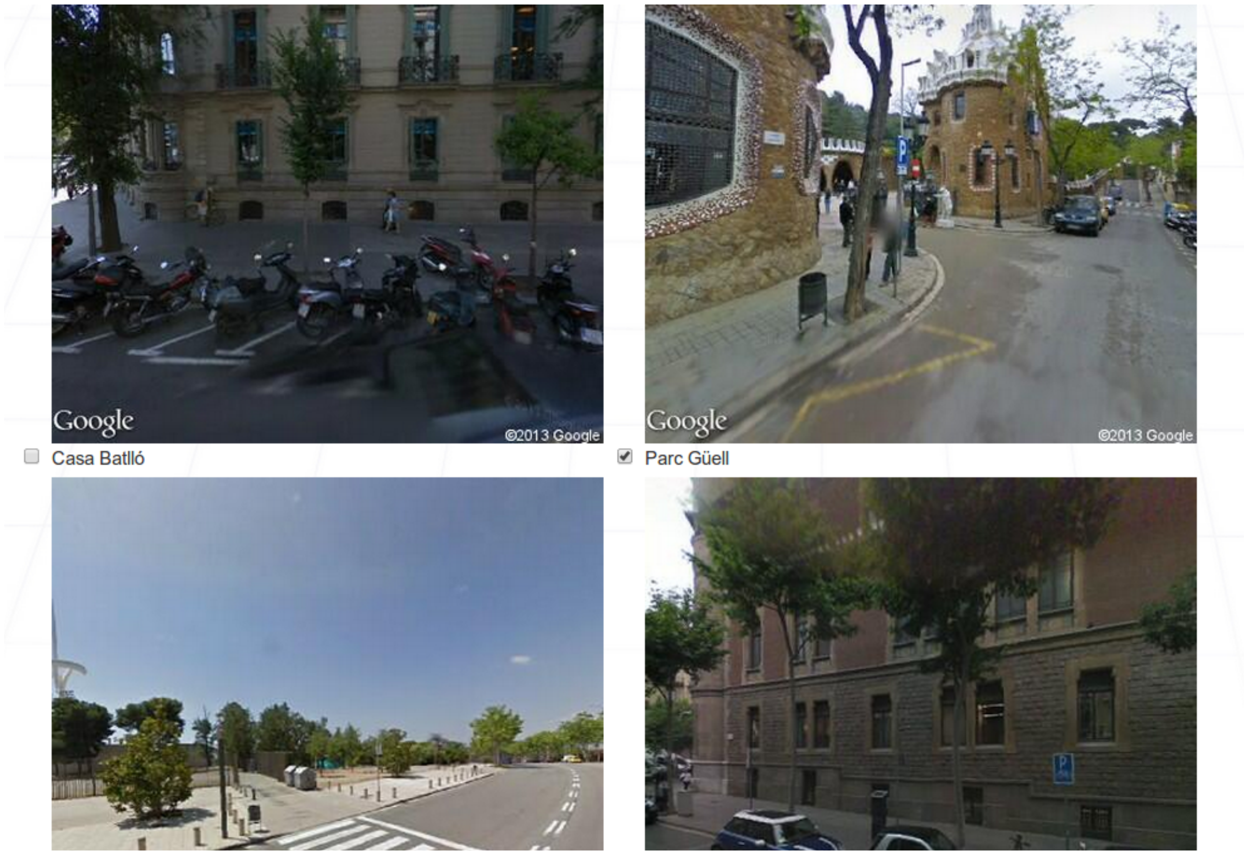
A list with the columns of the shapefile will be displayed. You will have to select that fits on your study (in this case “Nbarri” means borough in Catalan).



(Figure 40. Gantt chart screenshot)

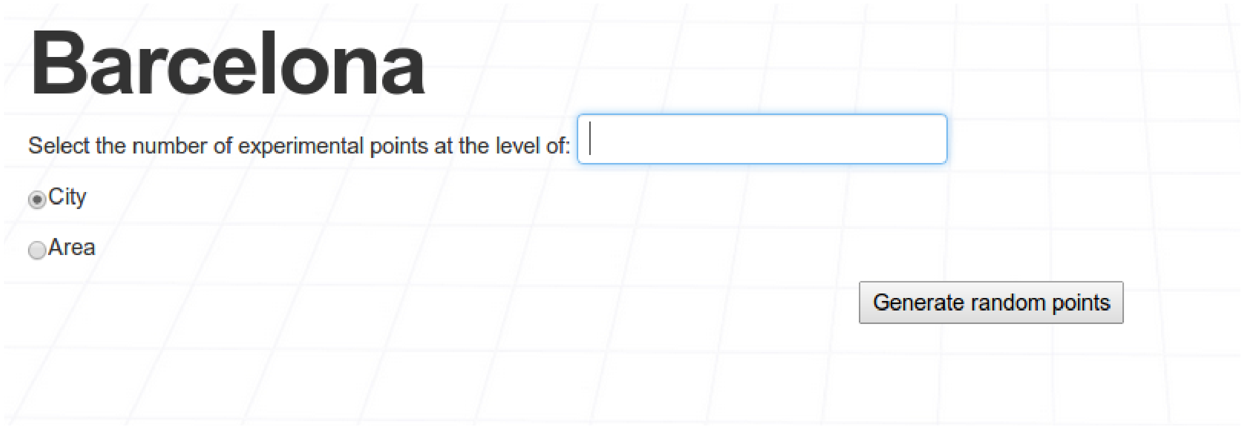
In the next web page you will be able to see two maps, one with the areas and one for the whole city, you can mouse over the zones and their name will be displayed.

In the next step you can choose the easy-to-guess points for the experiment. This points will encourage the user to keep playing because he will recognize the points and he will not be frustrated. These points will not count for the experiment results. You can choose between 0 and 20.



(Figure 41. Choose easy-to-guess screenshot)

Once you click the “Save easy-to-guess points” the next page will display an input and two radio buttons.



Barcelona

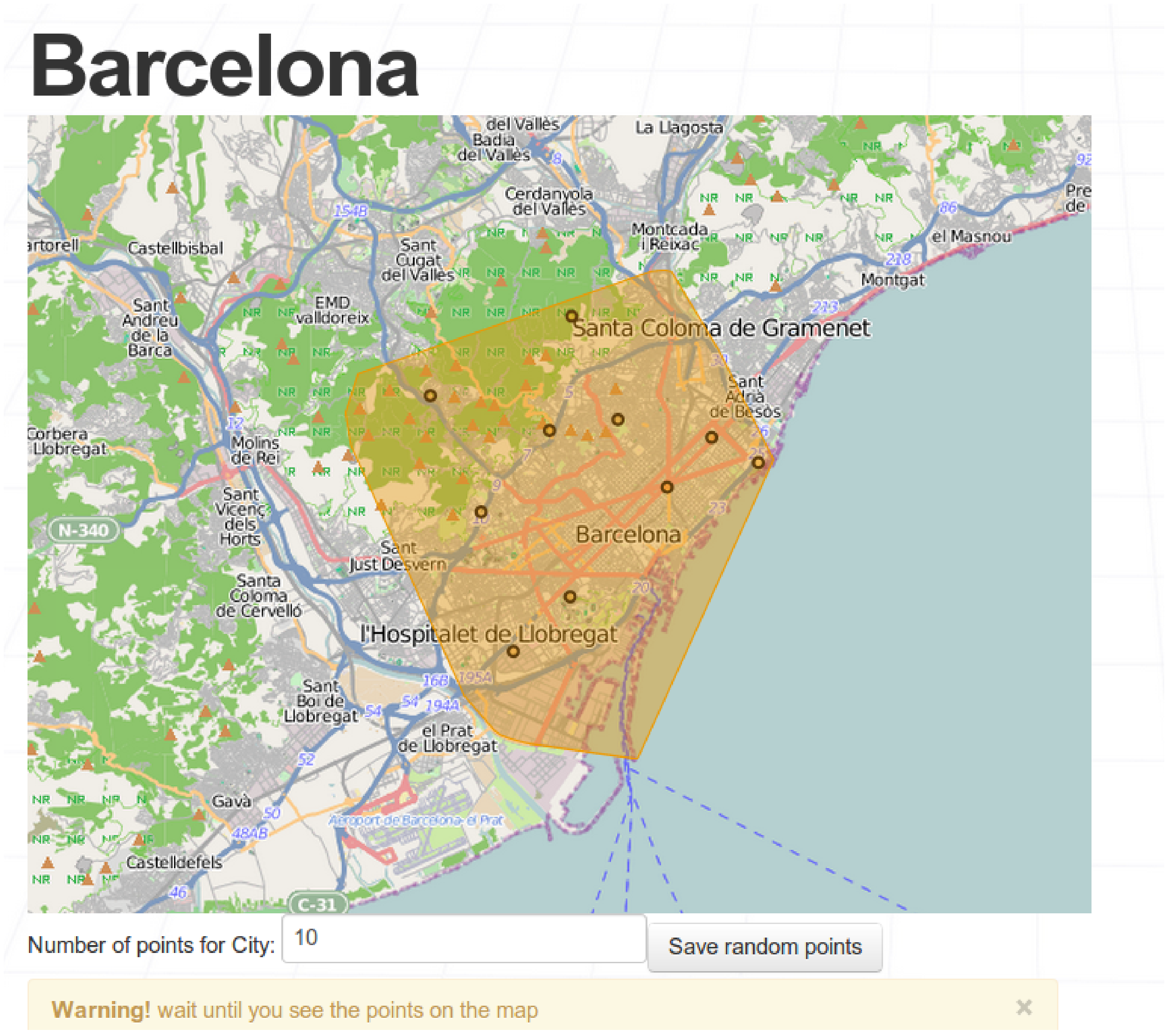
Select the number of experimental points at the level of:

[✓ City]

Area

(Figure 42. Number and type of random points screenshot)

If you select “Area” you will generate x points per area. If you select “City”, x points for the whole city. In our case we generated 10 points for the whole city and that's the result.


Barcelona

Number of points for City: 10

Save random points

Warning! wait until you see the points on the map

(Figure 43. Random point display screenshot)

You can regenerate the points changing the number and clicking “Save random points”.

Finally you just have to wait for the points and click “Finish”.

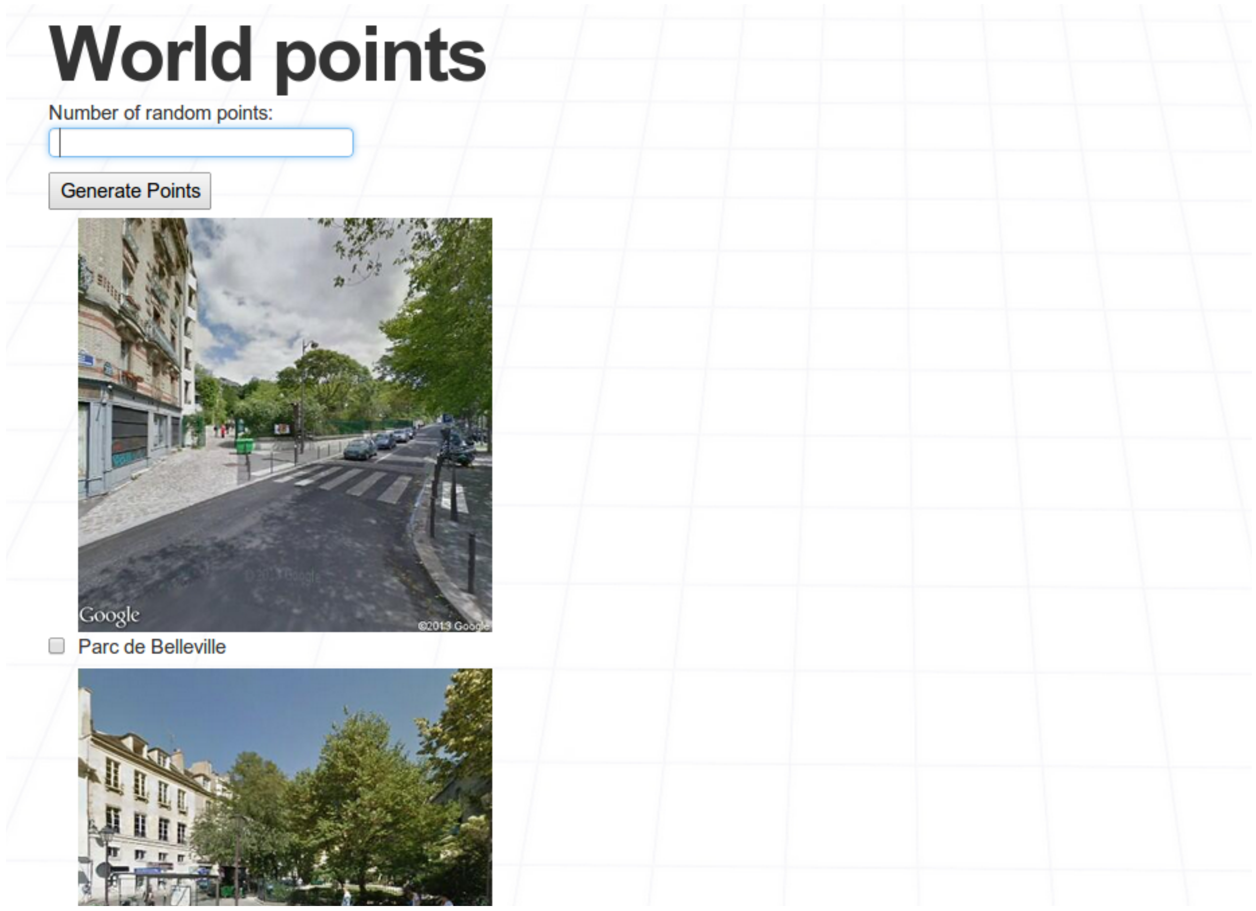
13.2.4. Import shapefile

You can go to "Import shapefile" to import new areas for a city. First you will have to select the city that you want to modify, select the zip file and finally select the column. Remember that the areas will be added to the previous areas, no one will be deleted. The city polygon will be remade in order to contain all the areas.

13.2.5. Generate Points

If you want to generate new points for a city you can do it easily here. You will select the city and you can create easy-to-guess points and random points just like we done in "Create city". You can select drop to delete the previous points. If you don't select it you will be able to create more points every time you click on "Save random points" in the last screen.

13.2.6. World Points

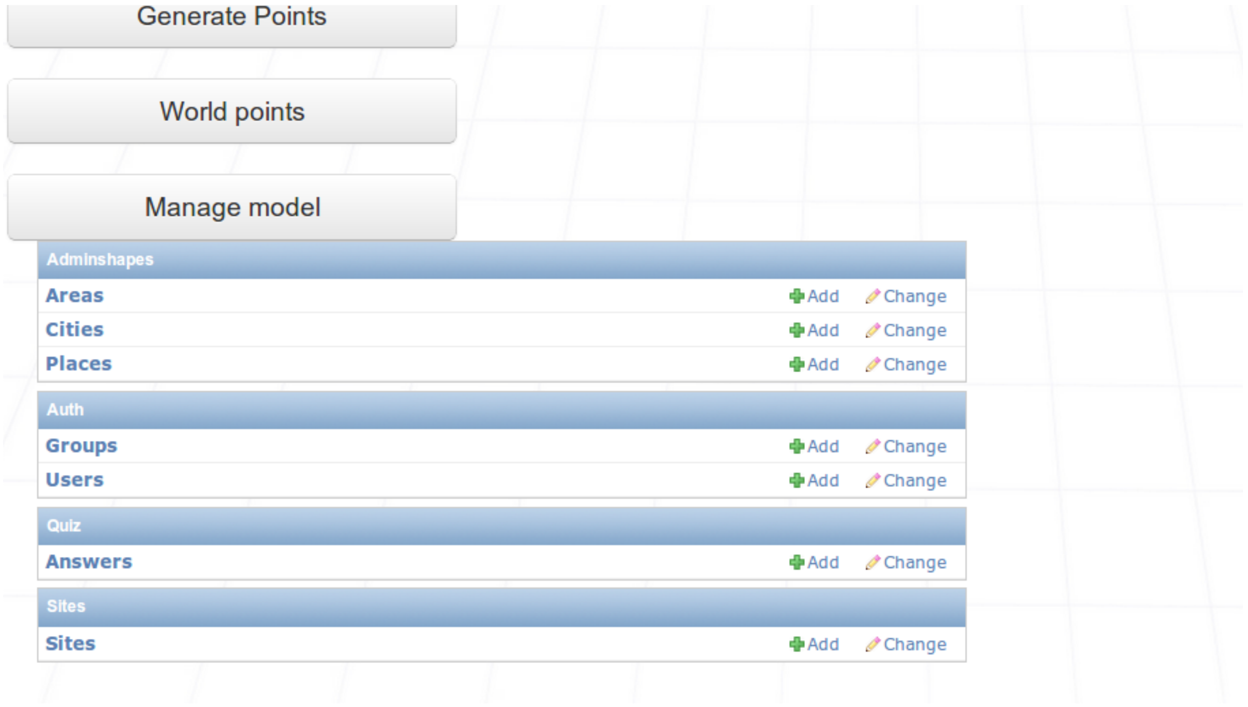


(Figure 44. World points screenshot)

In this option you can make random and easy-to-guess points for the world. The random points are from the 20 most relevant cities in the world. The images that are displayed are easy-to-guess points of these cities. You can save the number of random points that you want and save them. You can check the easy-to-guess points and save them too.

13.2.7. Manage model

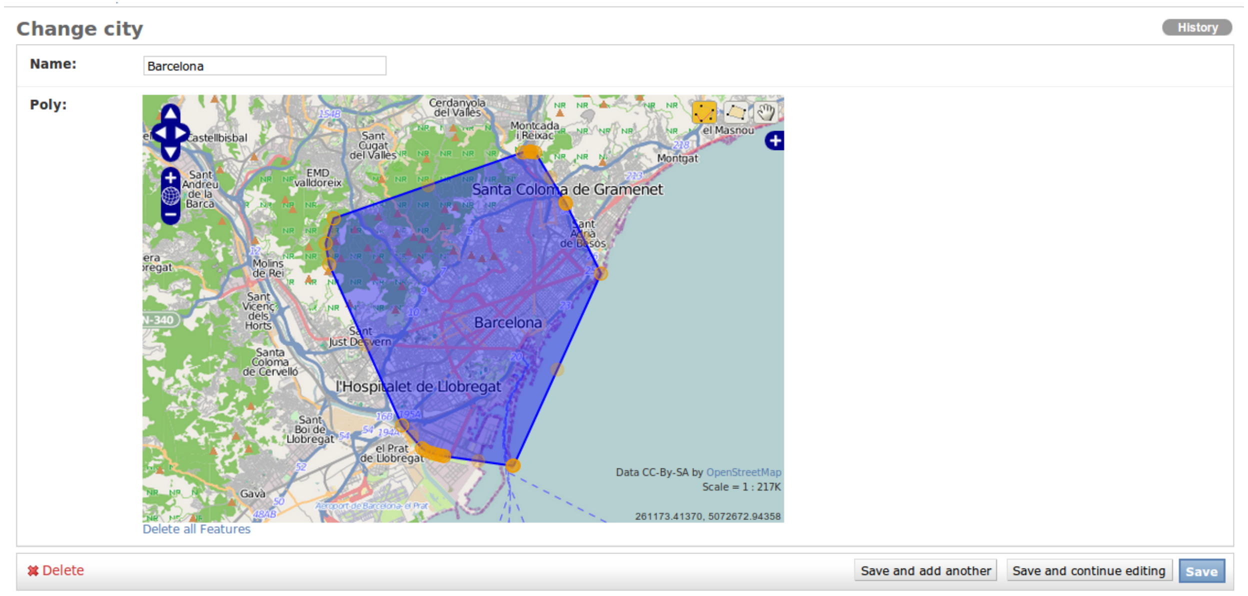
If we want, we can manage the database manually.



Adminshapes		
Areas	Add	Change
Cities	Add	Change
Places	Add	Change
Auth		
Groups	Add	Change
Users	Add	Change
Quiz		
Answers	Add	Change
Sites		
Sites	Add	Change

(Figure 45. Manage model screenshot)

We can add, modify and delete areas, cities or places. In the areas and cities we can modify the polygons and rename the objects.


Change city

History

Name: Barcelona

Poly:

Delete all Features

Delete

Save and add another | Save and continue editing | Save

(Figure 46. Manage polygon screenshot)

In the places we can create and modify the points, but if there are not image in Google Street View will not be saved.