Project Report – ResoMap

Resource mapping for the collaborative economy

16th December 2015
12-780 Advanced Python and Web prototyping for infrastructure systems
Jose Maria Lopez Pestaña – chema@cmu.edu

1 INTRODUCTION

The purpose of this document is to explain the objective, requirements and features to be develop in order to introduce *ResoMap - Resource mapping for the collaborative economy*. This application will connect resources (normally people with different skills sets and interests) with their needs.

To begin with, the inspiration and background of this project will be explained, followed by potential uses and finally the detailed features and development roadmap will be exposed.

1.1 BACKGROUND

In the last decade some models have successfully benefited of the collaborative economy approach, making key resources available to use by everyone. While Uber, Lift, BlaBlaCar and others made shared transport affordable and convenient, CouchSurfing and Airbnb made short term accommodation affordable or even free in exchange for an intercultural experience.

This project propose a basic framework in order to link those needs with skills from volunteers and freelance professionals, in exchange for a wage, training or even just accommodation and meals in the case of volunteers.

The use is inspired by the work of a Carnegie Mellon University alumni and entrepreneur, Pablo Sánchez (E&TIM and Mech'15), founder and CEO of Bridge for Billions[1]. This platform aims to connect entrepreneurs from in development countries with their support resources. In any case, there will be no collaboration in terms of programming and development although this project could potentially be used as a prototype.

1.2 POTENTIAL USERS AND CUSTOMERS

This application allows different stakeholders to collaborate in different stages of a project, operations or a supply chain. Two main markets could be identified:

- Freelance professionals

  Professionals would publish both their capacity to contribute and their needs from other players (in terms of resources or skill sets). The application would allow a basic tracking and status, additionally to the skill and asset mapping.

  Resources could include knowledge, funding, raw materials, access to markets, etc.

[1] Bridge for Billions, https://bridgeforbillions.org/ [Accessed November 3rd 2015]

- Volunteers / International development

Similar to other collaborative frameworks (e.g. HelpX.net), the application would link needs from Non-governmental Organizations (NGOs) and Small and Medium Enterprises (SMEs) and specialists and volunteers from other parts of the world.

Because its simplicity, during this project I will focus on the second option, which doesn't include complex assets and only volunteers skills set.

2 Product Design

The product should allow for basic profile registration to add description of capabilities and needs, requests and approvals.

Using these capabilities, the platform should be able to search and connect those capabilities with their closer needs. Using machine learning algorithms it should also propose recommendations and analysis on resource availability and needs.

The app would integrate payment so all balances could be withdrawn or payed periodically from a credit card or bank account.

Finally, it would be very convenient if the tool integrates a basic project management schedule, connecting with a cloud based project management systems in order to track status and schedules for each request.

3 Platform Features

The following section explains the features developed and its current state.

3.1 Page design

Since the approach followed for this project is simplicity, I tried to put all relevant information (agent skills, map and results) in a single page.

Agent Name		**Search**
Location		**Map**
User skills		
Add skill	Remove skill	
Save		

3.2 Profile set up

The basic profile registration in order to set capabilities and needs is a must in order to enter the data. We will model the data in order to not distinguish between producers and requesters: the same agent may need raw materials and skilled employees and be able to provide manufactured resources to a following agent, as in several levels supply chains.

Following the simplified approach in which only employees with a certain skill set are linked to its needs (as in the volunteers for international development), we will introduce several general skills to be selected (see Apendix A).

3.3 MAP LOCATION AND SEARCH

Using a similar approach to helpx.net, I will develop a map localization (similar to the one found in Airbnb to find a convenient accommodation).

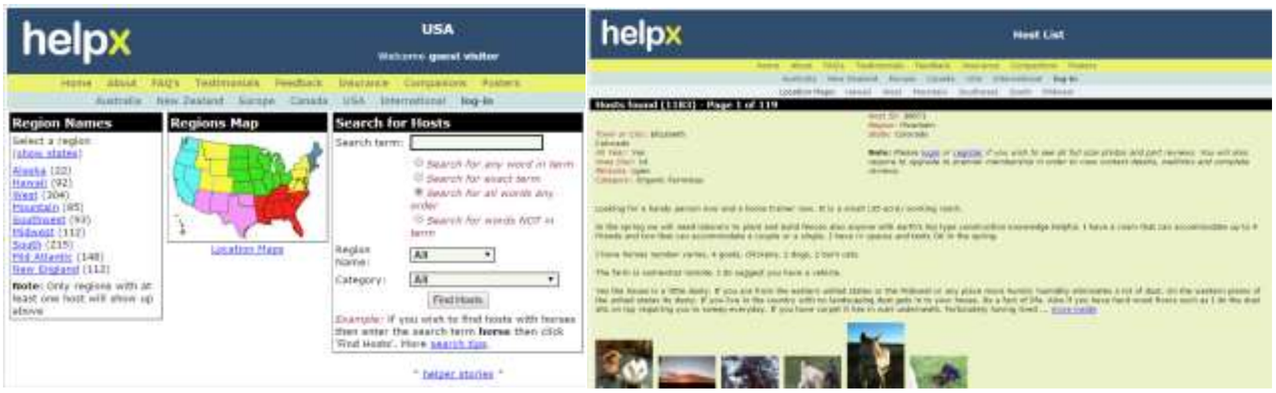


Figure 1. helpx map tool

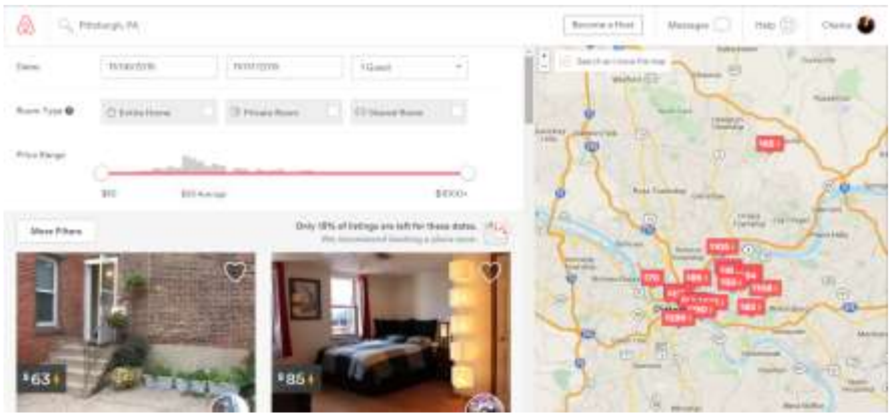


Figure 2. Airbnb listing

The map search will allow to search for the closest agent with a particular skill (either needed o provided).

4 IMPLEMENTATION

This application uses Django framework in order to connect with the SQLite database. In addition it calls JQuery and Google Maps API.

The following features have been analyzed and developed into the platform:

- Page design

Inspired on existing platforms and using the page designed previously, an HTML code was developed in order to interact with the user. It is relatively simple: it creates a table where fields and buttons are placed on the left side of the screen.

- Database model

Database should save all data related with Name, Location and Skills of each resource. In addition I added an ID field in order to each entry to be unique. The database used is SQLite, integrated by the Django framework.

- Google Maps API integration

Several features available on Google maps API Javascript where

- JQuery

JQuery library is used in order to simplify calls from HTML to JavaScript functions, and to use AJAX functions from JavaScript to Python components.

- Initialization of markers from database

Google Maps markers are loaded for each entry found in the database, placed in its corresponding location. It adds a label with the first letter of the name for quick identification.

- Infowindow content appears when clicking a marker

An event is added to the marker in order to show an infowindow each time a marker is clicked. It contains further information about the resource, such as Name and Skills.

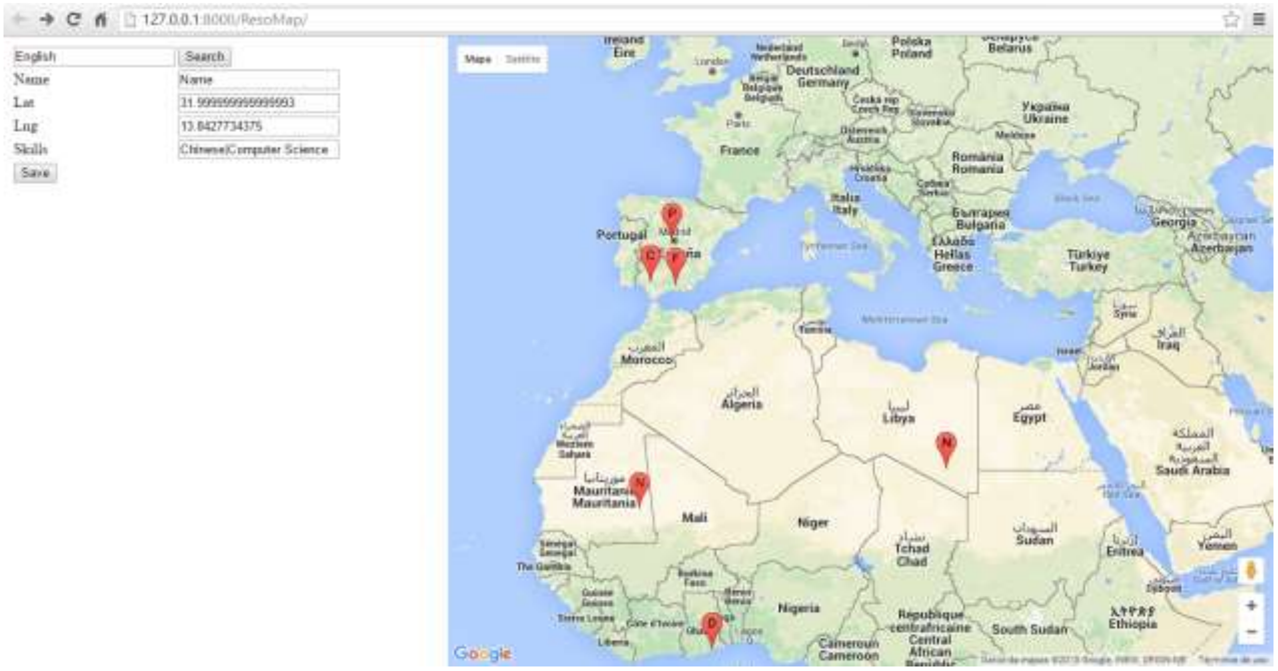
- Skills search engine

The application compares the string fulfilled in the search box with those stored in the database. For every positive result, it calculates the distance between the address field and the center of the map, and returns the one closer.

Distance between two resources is calculated in Python based on this StackFlow post, it uses Python standard math libraries.

- Save new profile

Profile defined in the left screen could be saved. The location is refreshed from the center of the map. When clicking 'Save' one entry is added to the database with the corresponding fields.


127.0.0.1:8000/ResoMap/

English	Search
Name	Name
Lat	31.999999999999993
Lng	13.0427734375
Skills	Chinese|Computer Science
Save	

5 Assessment

I think this project has accomplished its objective of visualize and search Resources in a map, using a very simple approach.

There are obviously several features missing but I think the use of Django and external libraries allowed to develop in a quick and simple way a prototype that could be showed to possible investors, or continue the design using it as a base.

I think course assignments have been very useful in order to implement most of the functions, and I was surprised of the potential of these tools for quick developing.

6 Video Location

https://youtu.be/3L43bwxbqGw

Video is 'unlisted', so you will need the link to access but it is not password protected.

7 APPENDIX - CODE

7.1 HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
      #map {
        position: absolute;
        bottom: 0px;
        right: 0px;
        width: 65%;
        height: 100%;
      }
    </style>
     <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></scrip
t>
     <script src="https://maps.googleapis.com/maps/api/js"></script>
     <script src="/static/ResoMap.js"></script>

    <title>ResoMap</title>
</head>
<body>
    <table>
        <tr>
            <td><input id="searchBar" value="English"></td>
            <td><button id="searchSkill">Search</button> </td>
        </tr>
        <tr>
            <td>Name</td><td><input id="Name" value="Name"></td>
        </tr>
        <tr>
            <td>Lat</td><td><input id="Lat" value="40"></td>
        </tr>
        <tr>
            <td>Lng</td><td><input id="Lng" value="0"></td>
        </tr>
        <tr>
            <td>Skills</td><td><input id="skills" value="Chinese|Computer
Science"></td>
        </tr>
        <tr>
           <td><button id="save">Save</button> </td>
        </tr>
    </table>


    <!-- <td><textarea rows="6" cols="40"
id="listSkills">Skills</textarea></td> -->

    <div id="map"></div>
</body>
</html>
```

7.2 JavaScript

```
/**
 * Created by Chema on 04/12/2015.
 */
function refreshCenter(){
    //alert(map.getCenter());
    $("#Lat").val( map.getCenter().lat()) ;
    $("#Lng").val(map.getCenter().lng());
}


function Resource(id,name,lat,lng,skills)
{
    this.ResoID = id;
    this.Name = name;
    this.Lat = lat;
    this.Lng = lng;
    this.Skills = skills;
}

$(document).ready(function()
{
    // Initialize data (first entry)

    $("#searchSkill").click(searchSkill);
    $("#save").click(saveReso);

});

var resources = [];
var map;

function initialize() {
    var mapCanvas = document.getElementById('map');
    var mapOptions = {
        center: new google.maps.LatLng(32, 0),
        zoom: 4,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    map = new google.maps.Map(mapCanvas, mapOptions);
    map.addListener('center_changed', refreshCenter);
    //geocoder = new google.maps.Geocoder();
    //var xhttp = new XMLHttpRequest();
    // "GET", "/loadDrawings/",true
    $.ajax({
        url: "/loadResources/",
        success: function (result) {
            var data = result;
            var resoStrings = data.split(";");
            for (var i = 0; i < resoStrings.length; i++) {
                var resoString = resoStrings[i];
                var attributeStrings = resoString.split(",");
                var newResource = new Resource(attributeStrings[0],
attributeStrings[1], attributeStrings[2], attributeStrings[3],
attributeStrings[4], attributeStrings[5]);
                if (newResource.ResoID != "")
                    resources.push(newResource);
            }
            //alert("A total of " +resources.length + " resources are added to
the array.");

            //Create markers
            for (i=0;i<resources.length;i++){

                var contentString = '<div id="content">'+
                  '<div id="siteNotice">'+
                  '</div>'+
                  '<h1 id="firstHeading"
class="firstHeading">'+resources[i].Name+'</h1>'+
                  '<div id="bodyContent">'+
                  '<p>'+resources[i].Skills+'</p>'+
                  '</div>'+
                  '</div>';

                 var marker = new google.maps.Marker({
                    position: {lat: parseFloat(resources[i].Lat), lng:
parseFloat(resources[i].Lng)},
                    label: resources[i].Name,
                    content: contentString,
                    //icon: 'path to the image file',
                    //animation: google.maps.Animation.DROP,
                    map: map
                    });

                // Add infowindow
                /*var infowindow = new google.maps.InfoWindow({
                    //content: this.content;
                    });*/

                var infowindow = new google.maps.InfoWindow()
                marker.addListener('click', function() {
                    infowindow.setContent(this.content)
                    infowindow.setPosition(this.position)
                    //alert(this.content);
                    infowindow.open(map, this.marker);
                 });
        } // for
        }
    });
    refreshCenter();
}

google.maps.event.addDomListener(window, 'load', initialize);

function searchSkill() {
    //alert (mapCenter);
    $.ajax({url:"/searchResource/"+ "?skill=" +
document.getElementById("searchBar").value
        + "&lat1=" +  map.getCenter().lat() + "&lng1=" + map.getCenter().lng(),
//$('#searchBar').val(),
        success:function(result) {
            //alert(result);
            // Point and open the ID received
            var location = {lat: parseFloat(resources[result].Lat), lng:
```

```
parseFloat(resources[result].Lng)};
            map.setCenter(location);

            var i = result;
            var contentString = '<div id="content">'+
                  '<div id="siteNotice">'+
                  '</div>'+
                  '<h1 id="firstHeading"
class="firstHeading">'+resources[i].Name+'</h1>'+
                  '<div id="bodyContent">'+
                  '<p>'+resources[i].Skills+'</p>'+
                  '</div>'+
                  '</div>';

            var marker = new google.maps.Marker({
                    position: location,
                    label: resources[i].Name,
                    content: contentString,
                    //icon: 'path to the image file',
                    //animation: google.maps.Animation.DROP,
                    map: map
                    });

                // Add infowindow
                /*var infowindow = new google.maps.InfoWindow({
                    //content: this.content;
                    });*/

                var infowindow = new google.maps.InfoWindow({
                    content:contentString,
                    position: location
                });
            infowindow.open(map, marker);

        }
    });
}


function saveReso() {
     $.ajax({
         url: "/saveResource/" + "?Name=" +
document.getElementById("Name").value
         + "&Lat=" + map.getCenter().lat() + "&Lng=" + map.getCenter().lng()
         + "&Skills=" + document.getElementById("skills").value,
//$('#searchBar').val(),
         success: function (result){
             initialize();
             }
         });
```

7.3 Views

```
from django.shortcuts import render
from models import Resource
from django.http import HttpResponse
from math import sin, cos, sqrt, atan2, radians


def loadResourceRequest(request):
    resources = Resource.objects.all()
    result = ""
    for reso in resources:
        result = result + reso.ResoID+","
        result = result + reso.Name+","
        result = result + str(reso.Lat)+","
        result = result + str(reso.Lng)+","
        result = result + reso.Skills+";"
    return HttpResponse(result)

def searchResource(request):

    skill = request.GET["skill"]
    lat1 = request.GET["lat1"]
    lng1 = request.GET["lng1"]
    #return HttpResponse(skill)
    resources = Resource.objects.all()
    distance = 99999999999999
    result = 'Not found'
    for resource in resources:
        strings = resource.Skills.split('|')
        for string in strings:
            #print (string+' == ' + request)
            if (string == skill):
                #print ('Got ya! '+ string)
                newdistance = getDistance(lat1,lng1,resource.Lat, resource.Lng)
                #print (newdistance)
                if newdistance < distance:
                    distance = newdistance
                    result = str(resource.ResoID)
    #print (result)
    return HttpResponse(result)

def getDistance(lat1, lng1, lat2, lng2):
    R = 6373.0
    rlat1 = radians(float(lat1))
    rlon1 = radians(float(lng1))
    rlat2 = radians(float(lat2))
    rlon2 = radians(float(lng2))

    dlon = rlon2 - rlon1
    dlat = rlat2 - rlat1

    a = sin(dlat / 2)**2 + cos(rlat1) * cos(rlat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c

    return (distance)
```

```python
def saveResource(request):
    #print("Test")
    resources = Resource.objects.all()
    #print(resources.length)
    resoID = resources.__len__() + 1
    #print(ResoID)
    name = request.GET["Name"]
    lat = request.GET["Lat"]
    lng = request.GET["Lng"]
    skills = request.GET["Skills"]

    newResource =
Resource(ResoID=resoID,Name=name,Lat=lat,Lng=lng,Skills=skills)
    newResource.save()
    return HttpResponse("OK")
```

7.4 Model

```python
from django.db import models

# Create your models here.
class Resource(models.Model):
    ResoID = models.CharField(max_length=10)
    Name = models.CharField(max_length=20)
    Lat = models.DecimalField(max_digits=10, decimal_places=4)
    Lng = models.DecimalField(max_digits=10, decimal_places=4)
    Skills = models.CharField(max_length=1000)
```

7.5 URL

```python
"""ResoMap2 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.8/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Add an import:  from blog import urls as blog_urls
    2. Add a URL to urlpatterns:  url(r'^blog/', include(blog_urls))
"""
from django.conf.urls import include, url
from django.contrib import admin
from django.views.generic import TemplateView
from ResoMapApp2.views import loadResourceRequest, searchResource, saveResource

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^ResoMap/', TemplateView.as_view(template_name='index.html')),
    url(r'^loadResources/', loadResourceRequest, name="loadResourceURL"),
    url(r'^searchResource/', searchResource, name="searchResourceURL"),
    url(r'^saveResource/', saveResource, name="saveResourceURL"),
]
```