QROWD - Because Big Data Integration is Humanly Possible

Innovation action

D3.5 – Public endpoints and deployment

<table>
<thead>
<tr>
<th>Author/s</th>
<th>Eddy Maddalena, Luis-Daniel Ibáñez, and Richard Gomer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date</td>
<td>30.11.2019</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
<tr>
<td>Reviewed by</td>
<td>UniTN</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
</tbody>
</table>
Table of contents

ABSTRACT 3

EXECUTIVE SUMMARY 4

INTRODUCTION 5

DEPLOYMENTS AND ENDPOINTS 5
  Migration from Figure Eight to Amazon Mechanical Turk 5
  Virtual City Explorer 6
    VCE source code and licensing 6
    VCE prerequisites 6
    VCE configuration 7
  Qrowdsmith 10
    Qrowdsmith source code and licensing 10
    Qrowdsmith prerequisites 10
    Qrowdsmith configuration 10
  Trip Update Interface 10

FEEDBACK AND FUTURE DEVELOPMENT 12
  Feedback collected 12
  Future development 12

CONCLUSIONS 13
ABSTRACT

This deliverable discusses the endpoints and the deployment of the three crowdsourcing services developed in the WP3: (i) VCE (Virtual City Explorer), that is the tool we developed to create maps of city mobility items; (ii) Qrowdsmith, a platform that allows to carry out gamified crowdsourcing tasks, and (iii) the interface to correct the citizens trips automatically computed from the data gathered by the citizens mobile devices.
EXECUTIVE SUMMARY

This deliverable demonstrates the crowdsourcing services of QROWD and their public deployment. It is for anyone who requires to run crowdsourcing tasks to (i) create maps of the mobility infrastructures in cities, (ii) make use of the benefits of gamification to enhance the level of engagements with contributors, and (iii) correct citizens trips automatically computed from data gathered by mobile devices. Nevertheless, the deliverable might be of particular inspiration for anybody that needs to design and develop solutions similar to the ones we proposed.

By reading this deliverable, the reader can have a rough idea about the primary steps that need to be done to deploy the proposed tools, including which are the external services that each of our solutions relies on to work correctly. More detailed information is released alongside the source code.

The main output of the deliverable is an update on the final versions of the crowdsourcing services developed in WP3, whose design and first iteration was already presented in previous deliverables D3.1, D3.2, and D3.3.
1. INTRODUCTION

This deliverable details the state of advancement of the three crowdsourcing based solution we designed to support the use cases of the QROWD project: (1) the VCE (Virtual City Explorer), a standalone tool designed to European Municipalities or Mapping Agencies who need to map mobility infrastructure items inside specific areas; (2) Qrowdsmith, a crowdsourcing tool that allow to carry out crowdsourcing tasks in an innovative ways that include the use of gamification strategies to make tasks more enjoyable and fun for the contributors; (3) the user interface to correct the citizen trips computed by the QROWD infrastructure based on the data gathered from citizens devices. We discussed the three tools in the previews deliverables D3.2 and D3.3.

Section 2.1 discusses the migration from Figure Eight to Amazon Mechanical Turk as a platform to recruit crowdworkers. Such changes slowed down our advancements since it required us to redevelop part of our components.

In the rest of section 2, we present the deployment and endpoints of our three crowdsourcing solutions. The section includes instruction prerequisites, requirements and useful information to install our tools. Also, for each of them, we discuss licences and term of use.

Finally, the deliverable ends discussing the feedback that our crowdsourcing tools received when we presented them to external audiences, such as researchers, crowdworkers, citizens, and potential end-users. Such input, along with the ideas provided to us, have been particularly useful in helping since those allowed us to improve our tools, facilitating our ongoing and future design and developing decisions. Section 3 ends discussing the future development and direction.

2. DEPLOYMENTS AND ENDPOINTS

This section describes the deployments and the endpoints of the three crowdsourcing solution discussed in deliverable 3.x: the Virtual City Explorer (VCE), Qrowdsmith, and the interface to correct the misclassification of citizens trips. Before going into the details of such deployments, we discuss at important deviation we had to face in the development of our solutions, namely the migration of the crowdsourcing platform our services use to recruit crowdworkers. Such changes involved the VCE and Qrowdsmith.

2.1. Migration from Figure Eight to Amazon Mechanical Turk

The VCE and Qrowdsmith need human contributors to perform their tasks. Contributors can be employees of the municipality, citizen volunteers, and/or crowdworkers that are paid to perform small tasks (aka microtask crowdsourcing). In microtask crowdsourcing workers are recruited from online crowdsourcing platform which in exchange of a fee connects the crowdsourcing requester with a pool of crowdworkers. In the first half of the project, to recruit crowdworkers to run the VCE
and *Qrowdsmith* in the context of the business cases, we opted for *Figure Eight*\(^1\), a widely known crowdsourcing platform that could guarantee a wide range of crowdworkers all around the world. The experience with such a platform and relative APIs that we could claim in our team drove the decision of that choice. Nevertheless, about May and June 2019, the change of internal policies of *Figure Eight*, forced us to change our provider. We thus opted for *Amazon Mechanical Turk*\(^2\) (also called *mTurk*) one of the main competitors of *Figure Eight*. This change of plan required us a significant effort first in learning how the new systems worked and then in adapting our solutions to the new recruitment platform. Thus, currently, both *VCE* and *Qrowdsmith* support tasks from mTurk.

Despite the disruption caused by the service migration, we can claim that our original decision of developing our solutions as standalone tools and using third-party platforms only for the recruitment activity was the right one. In fact, if we would have embedded out tools in Figure Eight (rather than any other platform), the migration would have been certainly demanded more time and effort.

### 2.2. Virtual City Explorer

The VCE is a Web application, developed on top of Flask\(^3\), a micro web framework written in Python. It is deployed in *Heroku*\(^4\), a widely used cloud platform as a service supporting several programming languages. The deployment take place by means of the Heroku Client, that packages the VCE source codes and send them to Heroku cloud platform as instance to be run in so-called *Dynos*, isolated, virtualized Linux containers designed to execute code based on a user-specified command. Several types of Dynos exist at different costs. A free *Dyno* with limited resources is available that can be used to deploy the VCE for testing purposes. However, to support larger scale crowdsourcing tasks it might be convenient to opt for performing plan\(^5\) offering better performances. Alternatively, the VCE might be deployed in a Virtual Private Server having Python 3.3.6 installed.

#### 2.2.1. VCE source code and licensing

The latest release of the Virtual City Explorer is deposited in Zenodo\(^6\), which is a general-purpose open-access repository developed under the European OpenAIRE program and operated by CERN. The artifact is reachable from this link: [https://doi.org/10.5281/zenodo.3540843](https://doi.org/10.5281/zenodo.3540843). The project is published under the Apache License V2.0\(^7\).

#### 2.2.2. VCE prerequisites

Before focusing on the details of the VCE configuration for deployment, we discuss

---

1. *Figure Eight* website: [https://www.figure-eight.com/](https://www.figure-eight.com/)
2. *Amazon Mechanical Turk* website: [https://www.mturk.com/](https://www.mturk.com/)
5. *Heroku* dynos types and prices: [https://www.heroku.com/pricing](https://www.heroku.com/pricing)
6. Zenodo dynos types and prices: [https://zenodo.org/](https://zenodo.org/)
7. Apache License, version 2.0 website: [https://www.apache.org/licenses/LICENSE-2.0](https://www.apache.org/licenses/LICENSE-2.0)
its prerequisites:

- Amazon Mechanical Turk (MTurk) is a crowdsourcing platform that the VCE uses to recruit crowdworkers. In MTurk tasks are called Human Intelligence Tasks (HITs). The VCE has the template of a HIT that is first instantiated and then sent to MTurk for launching when the VCE end-user orders a new exploratory task. The MTurk account needs to be linked to an Amazon Web Services (AWS) account. This operation is required in the set-up phase of the account. AWS Identity and Access Management (IAM) enables managing access to AWS services and resources securely.

- A Heroku account with an active plan needs to be set up to host the VCE app. Additionally, the installation of the Heroku client is required to facilitate the installation and the managing of the VCE directly from the terminal. The instructions for that can be found in the Heroku website.

- Google Maps account to obtain the API key consisting of a unique identifier that is used to authenticate requests associated with the project for usage and billing purposes. The Google Maps API key is obtainable in the Google Cloud Platform Console.

- The results produced by the VCE, together with its configuration, are stored in a MongoDB instance.

### 2.2.3. VCE configuration

Once all the prerequisites are satisfied, come parameters of the VCE needs to be configured. Such parameters, that are expressed through the Config Vars of Heroku, are shown in Table 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMAPS_API_KEY</td>
<td>Google Maps API key</td>
</tr>
<tr>
<td>URI_MONGO</td>
<td>Connection String URI to connect to MongoDB</td>
</tr>
<tr>
<td>IAM_USER_ACCESS_KEY</td>
<td>AWS access key ID</td>
</tr>
<tr>
<td>IAM_USER_SECRET_KEY</td>
<td>AWS secret access key</td>
</tr>
<tr>
<td>MTURK_REGION_NAME</td>
<td>Default region when creating new connections</td>
</tr>
<tr>
<td>URL_MTURK_SANDBOX</td>
<td>Endpoint URL for the mTurk Sandbox environment</td>
</tr>
<tr>
<td>URL_MTURK_PRODUCTION</td>
<td>Endpoint URL for the mTurk Production environment</td>
</tr>
</tbody>
</table>

The Config Vars has to be set from the webpage of the Heroku app settings, as shown in Figure 1.

---

8 Instructions to install the Heroku Client: [https://devcenter.heroku.com/articles/heroku-cli](https://devcenter.heroku.com/articles/heroku-cli)

9 Instructions to obtain a Google Maps API key: [https://developers.google.com/maps/documentation/javascript/get-api-key](https://developers.google.com/maps/documentation/javascript/get-api-key)
We created and published a webpage to promote the Virtual City Explorer, accessible in this link:  [https://qrowd-poi.herokuapp.com/](https://qrowd-poi.herokuapp.com/). Such page offers a demonstrator that can be used to simulate a mapping activity that requires to map bike racks in the limited traffic zone of Trento, our use case city in Italy. Figure 2 shows a screenshot of such website.
The Virtual City Explorer (VCE) is a tool designed for European Municipalities which need to generate mobility infrastructure maps of static items located in towns, such as bike racks, parking spots for disabled people, or road signs. The VCE adopts a crowdsourced approach, involving voluntary mappers or crowworkers who are required to virtually capture interactive panoramas of real cities, making use of technologies like Google Street View, looking for items with specific target characteristics to be annotated and mapped. The combined effort of the independent mappers allows the creation of annotated maps to be consulted by municipalities or integrated as part of human-machine hybrid systems. (I need to know more about the VCE).

**Figure 2: Screenshot of the webpage used to promote the VCE**
2.3. **Qrowdsmith**

The development of Qrowdsmith allowed us to deliver a prototype that can be used to run gamified crowdsourcing experiments.

2.3.1. **Qrowdsmith source code and licensing**

As well as the VCE, the Qrowdsmith prototype is deposited in Zenodo, on this page: [https://doi.org/10.5281/zenodo.3558249](https://doi.org/10.5281/zenodo.3558249). The project is published under the Apache License V2.0\(^{10}\).

2.3.2. **Qrowdsmith prerequisites**

Qrowdsmith prerequisites are similar to the VCE ones and detailed Section 2.2.2, with two exceptions: (1) the Google Maps can be omitted if the end user does not require to run tasks that included the Google Street View service, and (2) to allow the real-time connections among workers, Qrowsmith relies on Redis\(^ {11}\), that is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. Redis is available as Heroku Add-on\(^ {12}\), thus it can be easily linked to an heroku app.

2.3.3. **Qrowdsmith configuration**

This section describes the Heroku Config Vars required by Qrowdsmith.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMAPS_API_KEY</td>
<td>Google Maps API key</td>
</tr>
<tr>
<td>DATABASE_URL</td>
<td>Connection String URI to connect to</td>
</tr>
<tr>
<td>REDIS_URL</td>
<td>URL for Redis connection</td>
</tr>
<tr>
<td>ENCRYPT_KEY</td>
<td>Key for encryption</td>
</tr>
<tr>
<td>ENCRYPT_VI</td>
<td>Initialization Vector for encryption</td>
</tr>
<tr>
<td>SECRET_KEY</td>
<td>Flask APP secret key</td>
</tr>
</tbody>
</table>

2.4. **Trip Update Interface**

Based on the input from QROWDLab, and in connection with the work carried out for WP2 and WP6, we designed and developed a new version of the "Trip Update Interface", for citizens participating in Modal Split surveys to better amend their trips, and add trips that were undetected by the machine. Figure 3 shows a screenshot of

---

\(^{10}\)Apache License, version 2.0 website: [https://www.apache.org/licenses/LICENSE-2.0](https://www.apache.org/licenses/LICENSE-2.0)

\(^{11}\)Redis webpage: [https://redis.io/](https://redis.io/)

\(^{12}\)Redis add-on for Heroku: [https://elements.heroku.com/addons/heroku-redis](https://elements.heroku.com/addons/heroku-redis)
The interface is developed in JavaScript+React, and only needs an up-to-date web browser with java-script enabled, making it flexible enough to be used embedded in i-Log (the mode we chose for QROWDLab, described in D2.5), or other mobile or desktop app. In terms of input data, it can work either with a GeoJson description of a trip (as outputted by the Transport Mode Detector component), or with KML.

The source code of the final QROWD, used for the QROWSLab official experiment (D2.5) with Google's KML format is available in Zenodo\textsuperscript{13}.

\textsuperscript{13} 10.5281/zenodo.3532973
3. FEEDBACK AND FUTURE DEVELOPMENT

This section details the future exploitation and development of the crowdsourcing solutions we proposed. These plans, as well as the work done so far, are made also taking into consideration the feedback that our tools got in the multiple context they were adopted.

3.1. Feedback collected

During the last year we have been presenting the VCE and Qrowdsmith in several public events with variegate audience. For example, the VCE and Qrowdsmith were presented in several public events, and detailed in D9.4. During the Trento Smart City week 2019 we presented the VCE tools to citizens of Trento, we ran a user studies focusing on the improvement of the tool when used by end-users, e.g., the members of the municipality, to set up and run an explorative crowdsourcing task to map a certain area of the city. The study involved five members of the Municipality of Trento identified as potential VCE users, and allows us to identify several issues that affected the requester experience inside the VCE. We already addressed part of such issues, and destinate the remaining once to future works.

From the point of view of the contributor interface, we presented the VCE at the Southampton Science Festival 2019\textsuperscript{14} where members of the public were able to test the interface and provide face-to-face feedback that was integrated in subsequent development sprints, in combination with feedback provided by crowdsworkers through crowdsourcing platforms when we ran the approach in the context of the business cases.

3.2. Future development

The three crowdsourcing services discussed in the previous session were used in Qrowd to involve users, citizens and crowdsworkers in crowdsourcing activities to generate, curate, analyse data. The solutions, although being in a prototypal stage, are a valuable resource that can be adapted to more general contexts. Given the good results collected so far, we plan to include our solutions in other ongoing and future projects, that need to rely on human participation in hybrid workflows in crowdsourcing contexts. For example, both the Virtual City Explorer and Qrowdsmith are already been adopted in the ACTION\textsuperscript{15} (Participatory science toolkit against pollution) Project co-funded by the European Commission under the Horizon 2020 as solutions to support citizen science activities to combat and prevent major forms of pollution in the EU.

\textsuperscript{14} Southampton Science and Engineering Festival 2019 website: https://www.sotsef.co.uk/
\textsuperscript{15} H2020 Action project website: https://actionproject.eu/
4. CONCLUSIONS

This deliverable discussed the final iterations and deployments of the three crowdsourcing services designed in WP3 to support the Qrowd use-cases. In summary, the Work Package delivered solutions for (i) mapping mobility infrastructure items making use of efforts from crowdworkers or volunteers; (ii) gamify traditional crowdsourcing tasks, and (iii) correct data of citizens trips automatically computed from the data produced by the mobility sensors installed on their mobile devices.