# EMERGREEN Monitoring Methodology

## 1. Project Background

EMERGREEN or "Emerging Technologies for Greener Communities" is an interregional project funded under the Northern Periphery and Arctic Programme 2014-2020.

Commencing in October 2018, six partners from Ireland, Northern Ireland, Sweden, Finland, and Faroe Islands are working together to overcome the common territorial challenge of how to deliver quality and sustainable public services in remote areas, addressing factors such as long distances, high service delivery costs due to low demand aggregation, shortages in human and material resources, and lack of access to the latest innovations.

The new services being provided through this project are a Coastal Stories App, which uses crowdsourced images to monitor marine erosion, a Recycling App and Chatbot providing 24/7 information on recycling and waste management to citizens, a Chatbot providing customised solar energy information and advice, an online idea generation and citizen participation platform addressing food waste and recycling, and a website providing data and user experience stories on alternative heating solutions.

## 2. Purpose of Monitoring Methodology

A common methodology for the EMERGREEN services ensures that, while the services themselves are different, with varying goals, target groups and processes, not to mention technology, they all gather the relevant feedback that allows them to make an assessment of the services they have developed through this project, in terms of service effectiveness, usability, accessibility and ease of expansion.

Initially the process began with a Stakeholder analysis phase conducted with each partner. The main purpose of this phase was to establish requirements, target stakeholders and technical aspects for the piloting and testing phases. Areas covered included: the testing environment, stakeholders and their role in the ecosystem, stakeholder engagement strategy, stakeholder needs and expectations and data collection strategy.

Following this, each partner engaged with the relevant stakeholders in their organisations and with their general target groups , community groups, local small businesses, etc. and gathered data from them on the usefulness, usability and effectiveness of the services they created. The resulting data from that monitoring have been included in each service's assessment report.

The development approach adopted in the project reflects the dynamics of developing emerging technologies for public services. Given the strong emphasis on community-driven initiatives in this project, we followed user-centred design techniques to ensure that any technology developed in the project had citizens' needs as the top priority.

Since the technologies and methods used are different across the services, this monitoring methodology document presents the services in separate groupings, depending on the technology used, the functionality involved, and the types of data being captured.

# 3. Partner Services

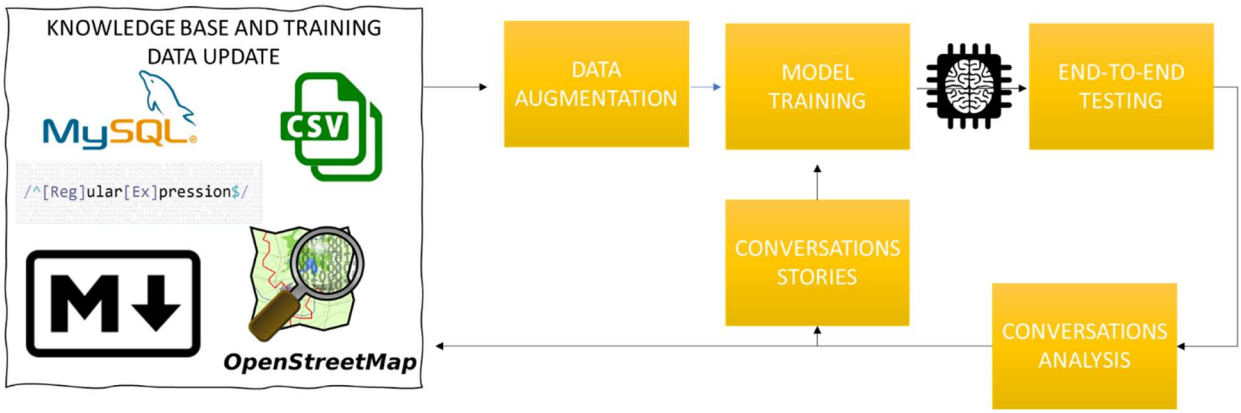
## 3.1 Chatbots – Recycling and Waste Management and Solar Information and Advice

The design approach for the provision of user support via Chatbot services followed he agile development process, involving more than 10 iterations of technology releases in close collaboration with our stakeholders Derry and Strabane City Council, Ireland and Vasternoorland, Sweden.

Every cycle was initiated with requirements' collection and analysis and finalised with relevant evaluation by our users in the form of dedicated focus groups. The feedback loops were supported on the technical side by leveraging an established GitLab solution for managing versions and tracking issues. For our non-functional requirements analysis, we adopted the framework by Mallios et al. [1] who list 14 features to evaluate different dialogue systems initiatives methodologies. We further extended the framework with four extra features covering technical (1-3) and user behaviour (4) issues: 1) Integration, 2) Open-source-based, 3) Modifiability and 4) Behavioural Change Promotion.

User groups were carefully chosen with the service partner to facilitate this process, in the case of Derry and Strabane, for example the groups included: 1) Older Age Groups; 2) Disability Groups; 3) Younger Age Groups; 4) Migrants, and 5) Students.

Each testing iteration followed the process flow in the diagram below: Step 1) Knowledge base development, Step 2) etc




To support the development process, two different instruments were developed: 1) *intents_template*, used for AI-training and implemented as a CSV file, this consists of a tabular

document in which the public service partners list the users' goals, or users' intents, that the chatbot solution should address, including various text examples that will trigger them. 2) C*hatbot_behaviour_template* a document exploring the chatbot functional requirements in-depth, and providing an overview of the organisation's maturity in terms of software infrastructure and API services availability.

Following the data gathered with the C*hatbot_behaviour_template* instrument, sample conversations between the chatbot and the users were written and these were then used as training data. Data augmentation techniques were utilised to artificially augment training datasets using programming languages and tools such as Chatito. This methodology allowed the creation of AI training samples by including predefined structures and sets of entity examples, and significantly increased the capacity of the chatbot to generalise well when presented with previously "unseen" queries.

As the functional requirements indicated a need for identifying street names, addresses, postcodes, recycling items, bin colours and synonyms, relevant lists of entities were gathered from different sources, such as OpenStreetMaps, CSV files, databases, or created from scratch using human annotators and regular expressions.

Each iteration in the development process followed the pipeline described in Figure 1 below.

After the chatbot model training phase, end-to-end testing was performed with different user groups, who interacted with the chatbots through a dedicated user interface. This testing incorporated both functional (relating to the information provided) and non-functional (usability, accessibility etc.) criteria. All the conversations were stored for further analysis, and users referred to their conversation IDs when providing feedback. This conversation monitoring provided a key instrument to analyse technical details, point out chatbot weaknesses, and to explore and inspire new user stories previously not considered. As the iterative process evolved, the need for a dynamic knowledge base system that would allow a simple method of updating chatbots' answers in different scenarios became imperative. This need motivated the creation of a REST API that would provide specific data upon request, and that would allow for ongoing content modification without impacting on the chatbots' deployment.

The chatbot monitoring strategy keeps track of the conversations the chatbot is having with the users on a continuous basis. This detailed information about the chatbot's behaviour supports the identification of the chatbot's knowledge base weak areas. Both service chatbots' architectures include a system to store this valuable information accessed via a, remotesecure and user friendly interface. This system is deployed on the Rasa X tool. The tool provides features to review conversations. ConversationsFigure 1 shows the main conversations panel view. In this example, in the conversation of December 2nd, we can observe a user has typed "hi there". The tool records the chatbot's predictions of this text, in this case the chatbot understood the user is trying to greet. We can observe as well that the tool informs the certainty of the detected user intent by providing a statistical value within 0 and 1. In this case, 0.96 indicates the chatbot was almost 100% sure the user is trying to greet. The same certainty value is provided for the action the chatbot performs after the user interaction, in this example the chatbot predicts and performs the action "action_greet" with 86% certainty, the "action_show_options" with 98% certainty, and the "action_listen" with 100% certainty.

The number of conversations available for review is visible in the top left corner. The list of conversations can be narrowed down using different filters, e.g. time. Figure 2 shows the different

time options available. This feature can be used, not only to find a specific conversation, but to monitor the chatbot usage along time.

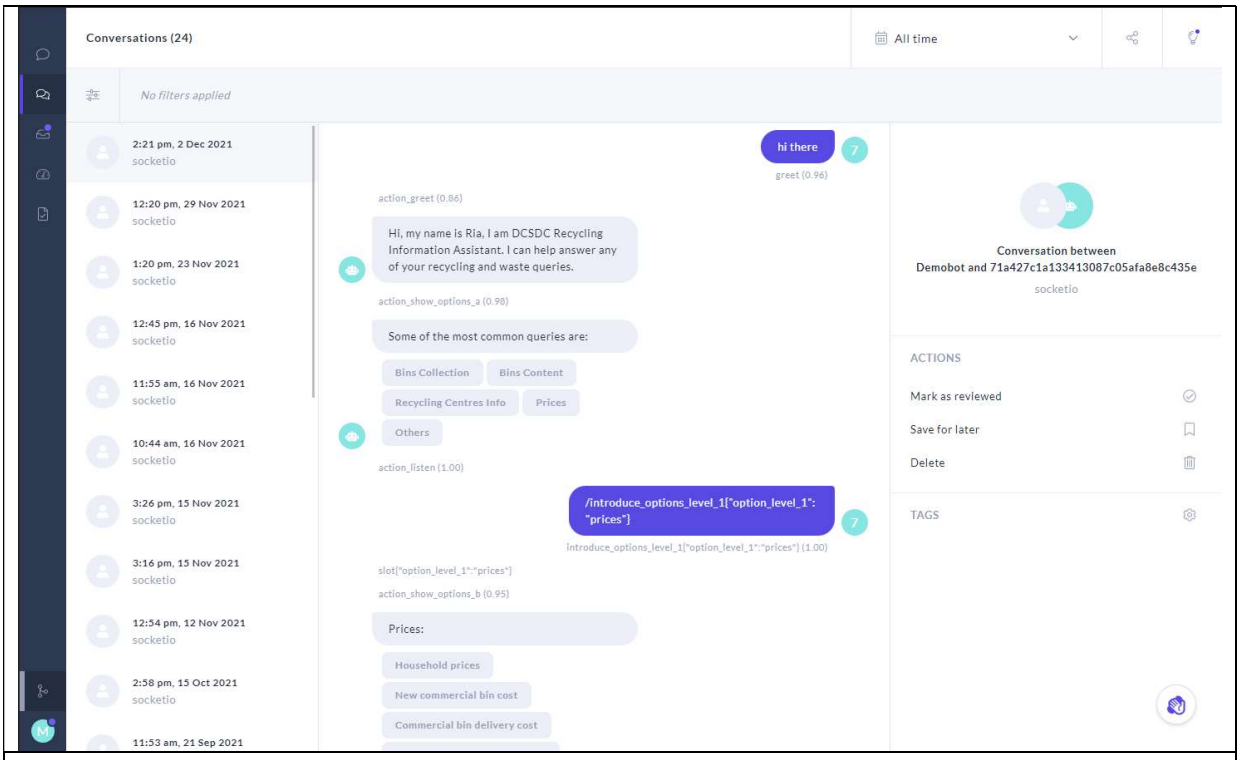



*Figure 1. Conversations*

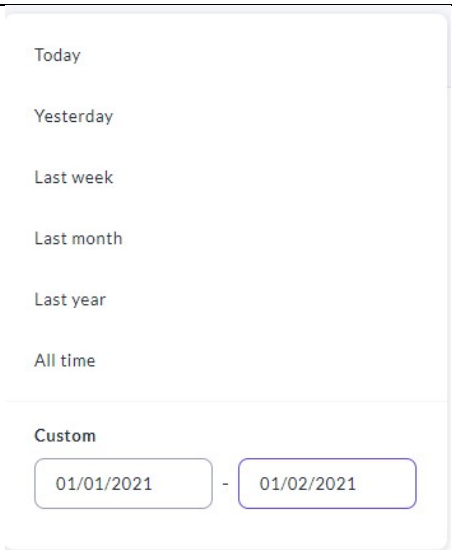



*Figure 2. Time Filters*

Other filters can help identify situations where the chatbot is having difficulties, where more training data might be needed, or potential new use cases developed. Figure 3 shows the different filters available and Figure 4 presents the filter based on confidence. The user can filter based on the confidence of intents (what the user is intending to do, and what the chatbot understands), or

confidence of the actions performed (what is the next action the chatbot will perform). The first option allows to identify intents that may need more training data, or intents that are not being addressed by the chatbot solution. The later one, helps us identify situations where the chatbot is not sure about the next action to take. In this case, new conversation stories might be needed at the training phase.

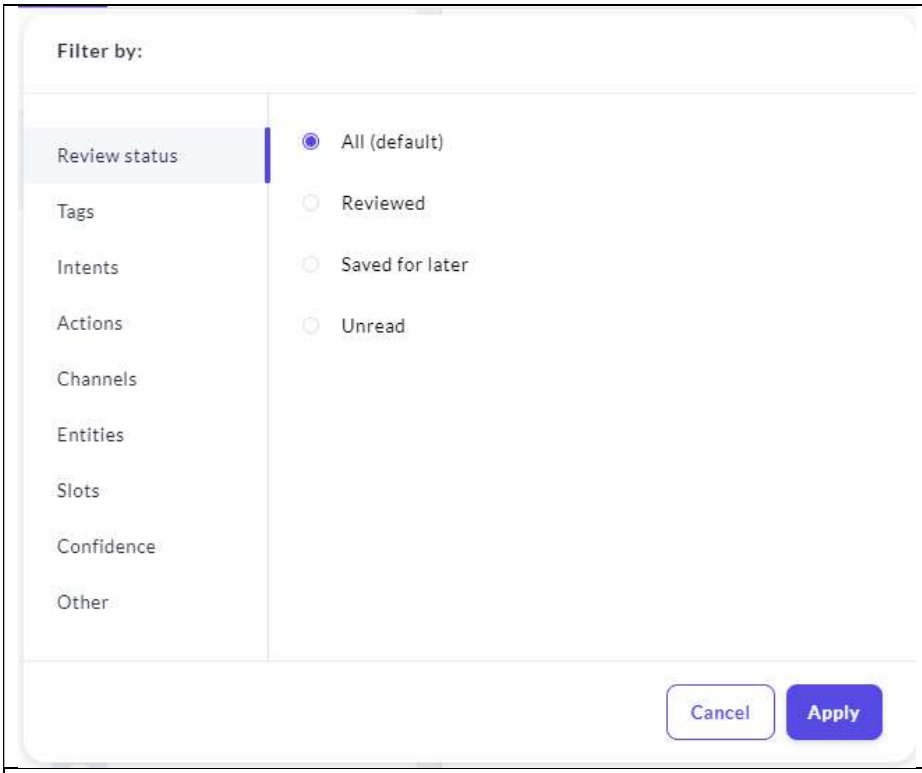



*Figure 3. Filters Window*

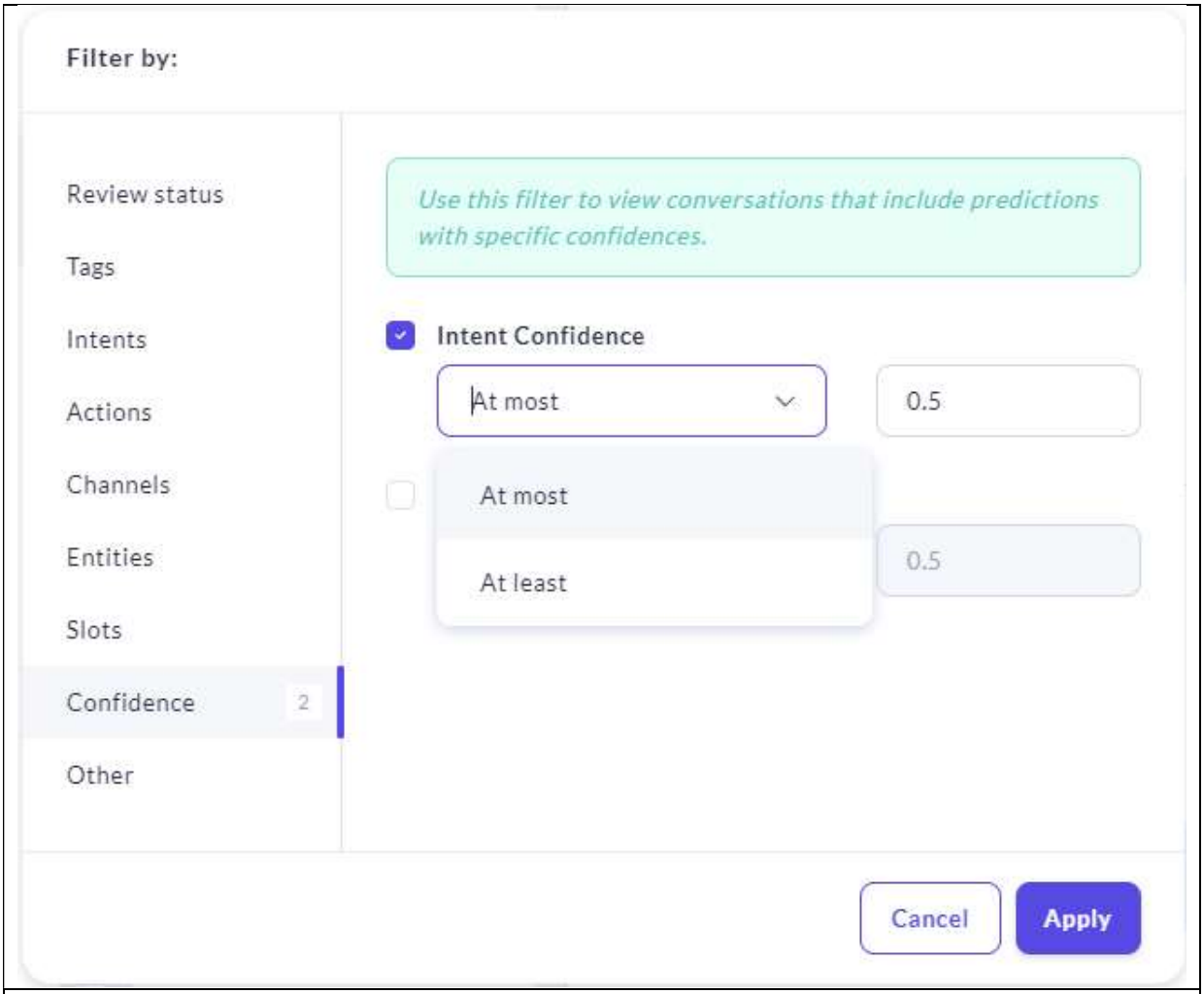



*Figure 4. Confidence Filter*

Finally, the filter by action feature allows for finding specific situations in the conversations. For instance, the "action_custom_fallback" presented in Figure 5 allows for filtering conversations where the chatbot presented a set of options for the user to select from in the event that the chatbot cannot decide what the user intent is, as depicted in the example in Figure 6.

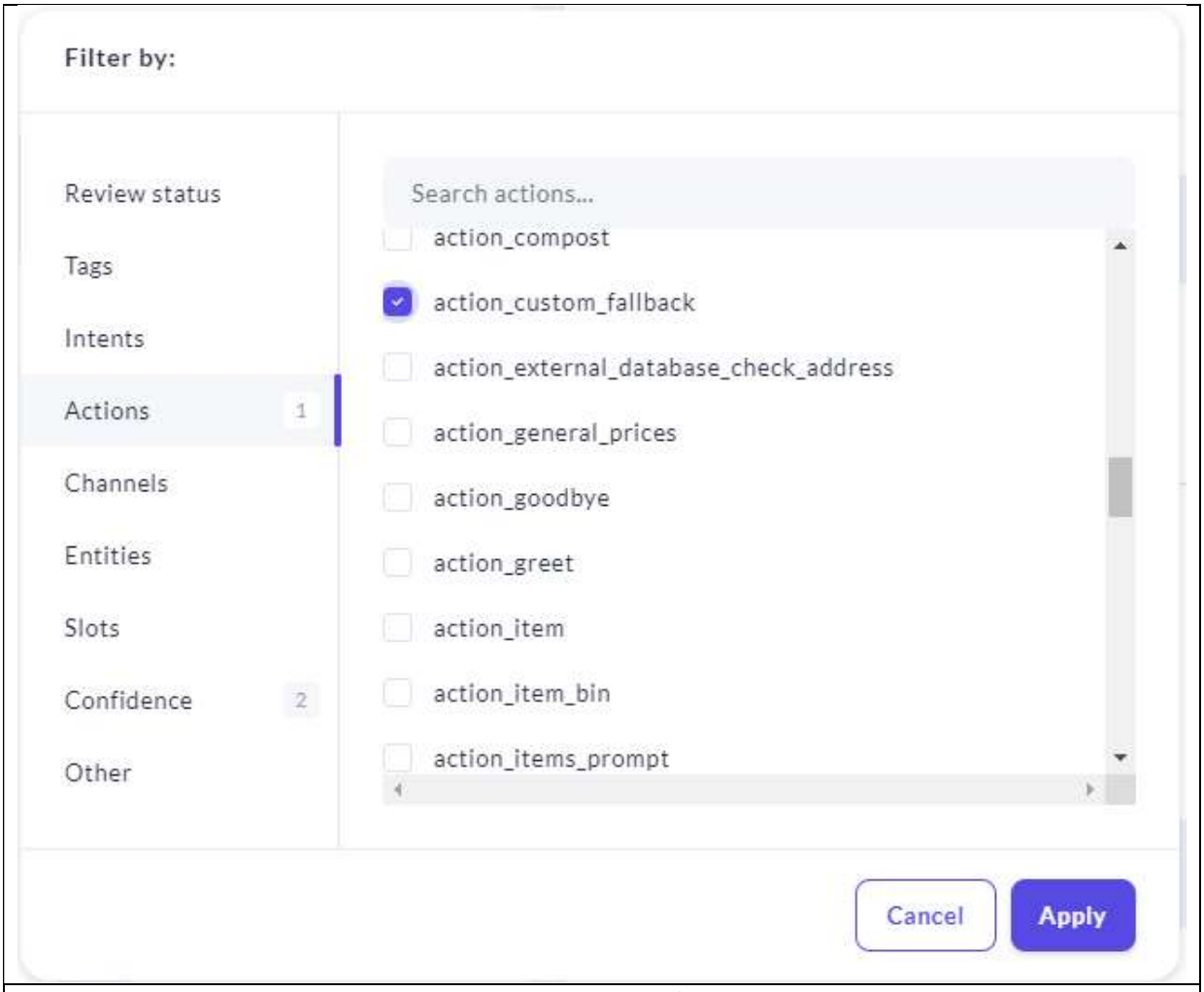



*Figure 5. Actions Filter*

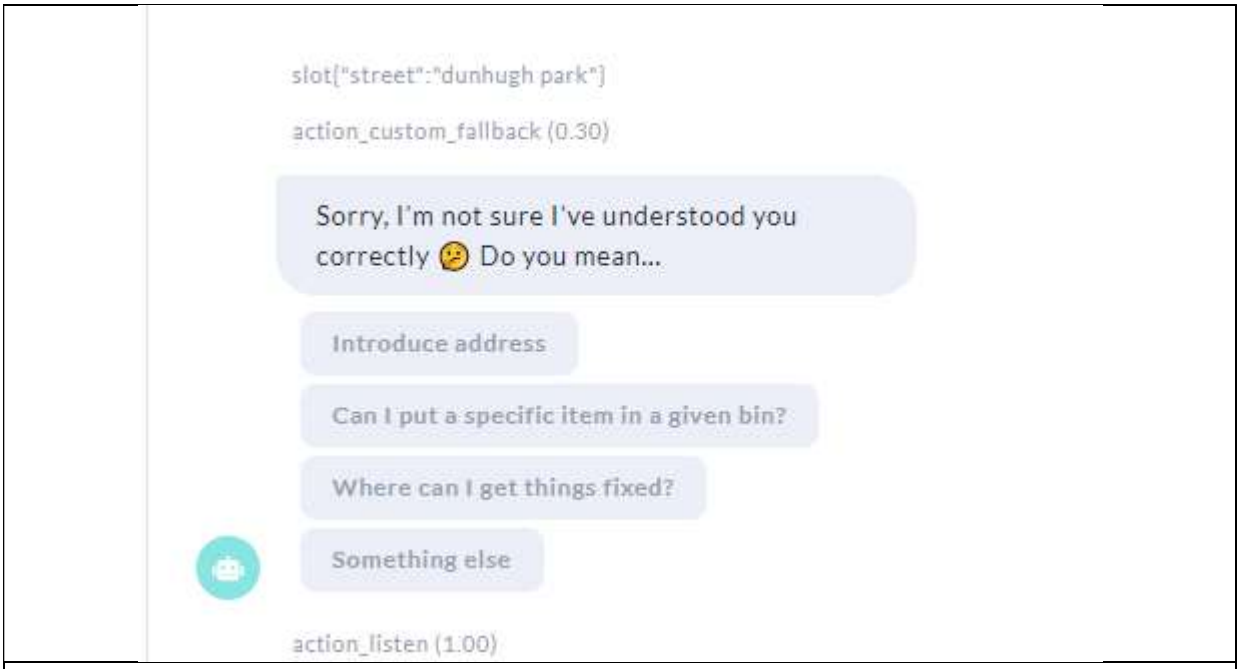



*Figure 6. Fallback Action*

In the case of RIA chatbot, a mechanism to log the user's explicit satisfaction has been Implemented. This feature enables the recording of users' sentiment about their chatbot interaction, recording the positive or negative answer in the database for further analysis. The data stored also encompasses the conversation id, the date and time when the satisfaction was logged.

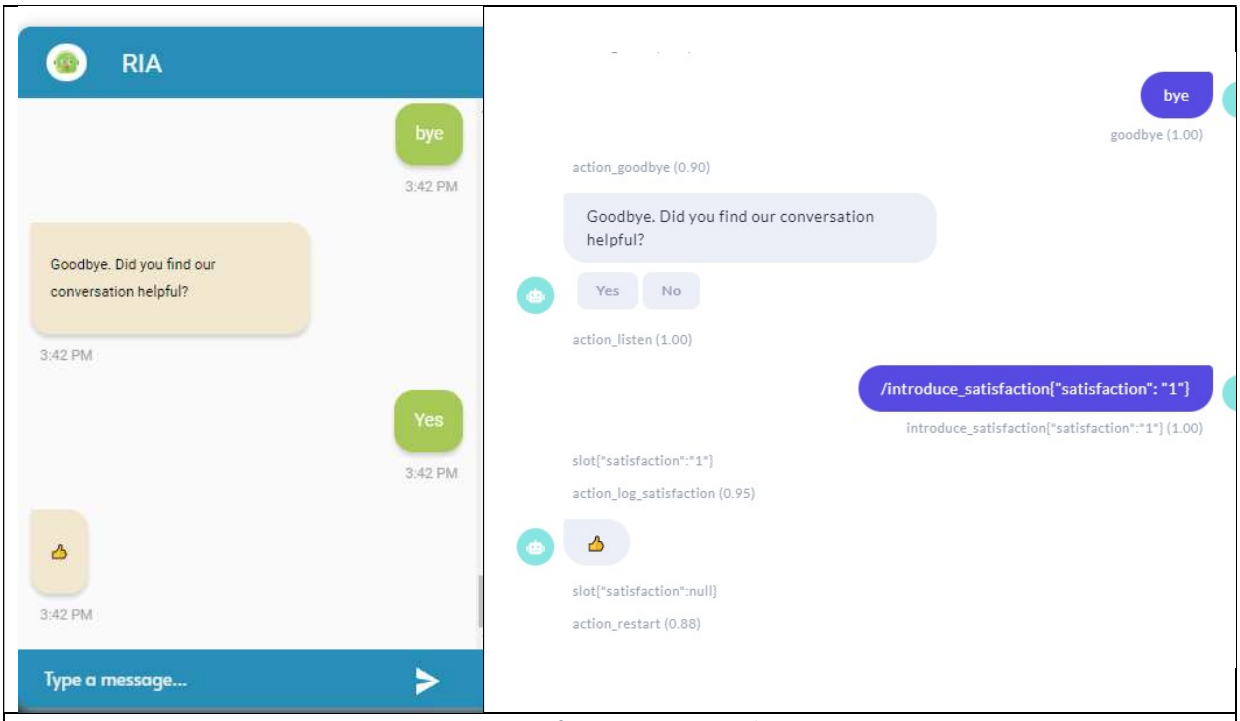



*Figure 7. Satisfaction Logging Mechanism*

## Interacting with Conversation's Platform via an API

An API (Application Programming Interface) is defined by the IEEE as a software interface that allows computers to request, retrieve, and exchange data and information in a standardized way. The conversations platform used for keeping track of the users' interactions with the conversational agents allow to be queried using a REST API. In this context, REST (short for REpresentational State Transfer) is an architectural style defined to help create and organize distributed systems, improving aspects such as performance, scalability, simplicity, modifiability, portability and reliability. REST architecture treats every content as a resource. Resources define what the services provided by the API will be about. Despite the fact that a resource can be represented in different formats, JSON has been widely used as the standard Data Transfer Format. JSON has many advantages over other data formats, such as XML, previously popular in the domain. It is lightweight since most of the information it transports belongs to the actual data. Another advantage is that it is easily readable by humans, and it can transport many data types other than strings.

In the context of this deliverable, the REST API service provided by RASA X tool can help develop tailored scripts that can automate the extraction of relevant data regarding the interaction of the users with the agent. Figure 8 and Figure 9 present two different API calls to retrieve, first, a list of conversations, and second, a list of messages within that conversation. More information about the API endpoints is available here: https://rasa.com/docs/rasa-x/0.28.6/api/rasa-x-http-api/#operation/getConversations

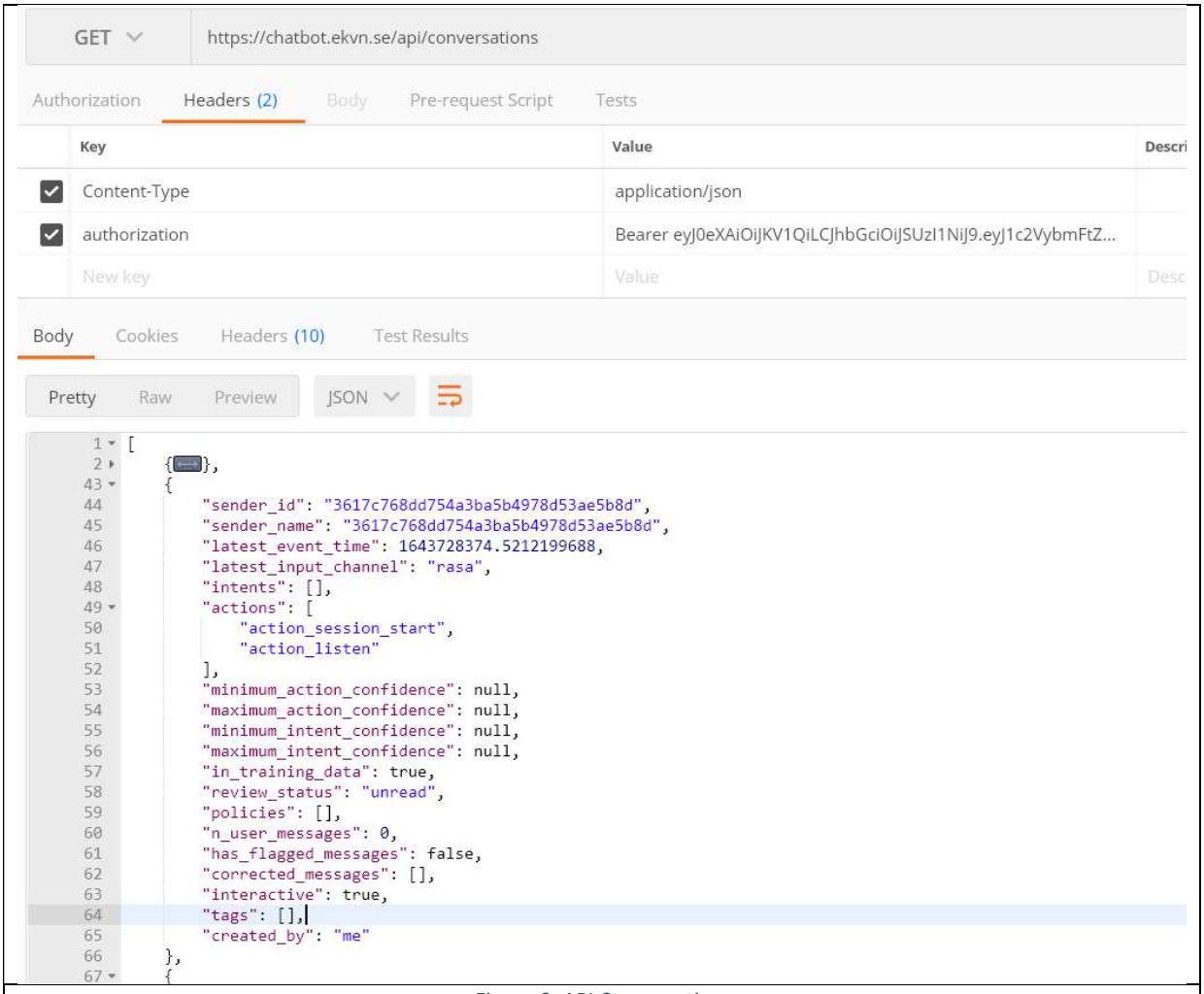



*Figure 8. API Conversations*

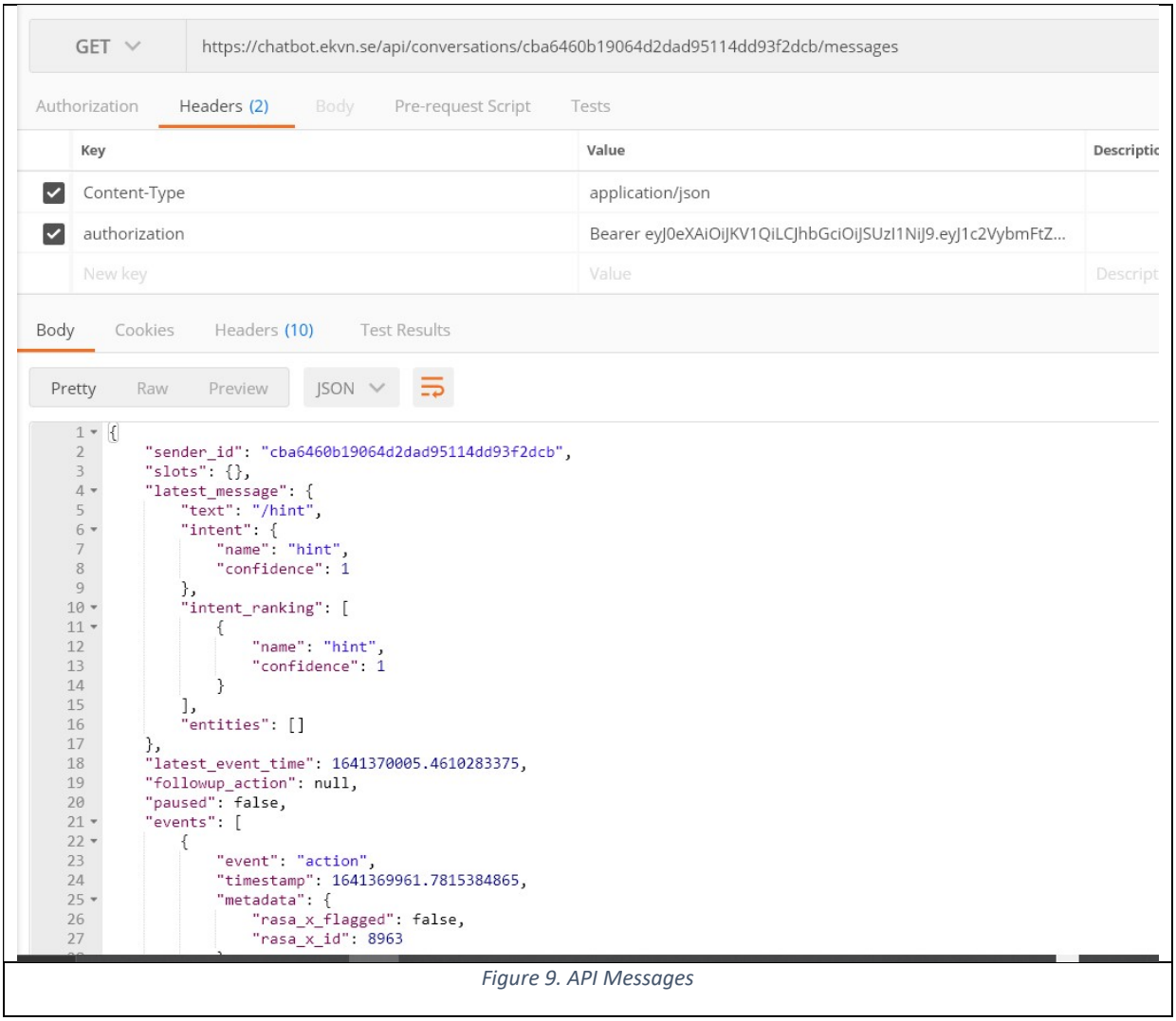



*Figure 9. API Messages*

## Coastal Erosion App Monitoring

The coastal erosion platform monitoring strategy is based on the use of the statistics dashboard, available through the administration panel. Administrator users have the chance to accept or reject image submissions, and to create, edit, publish, and remove coastal stories, as shown in Figure 10.

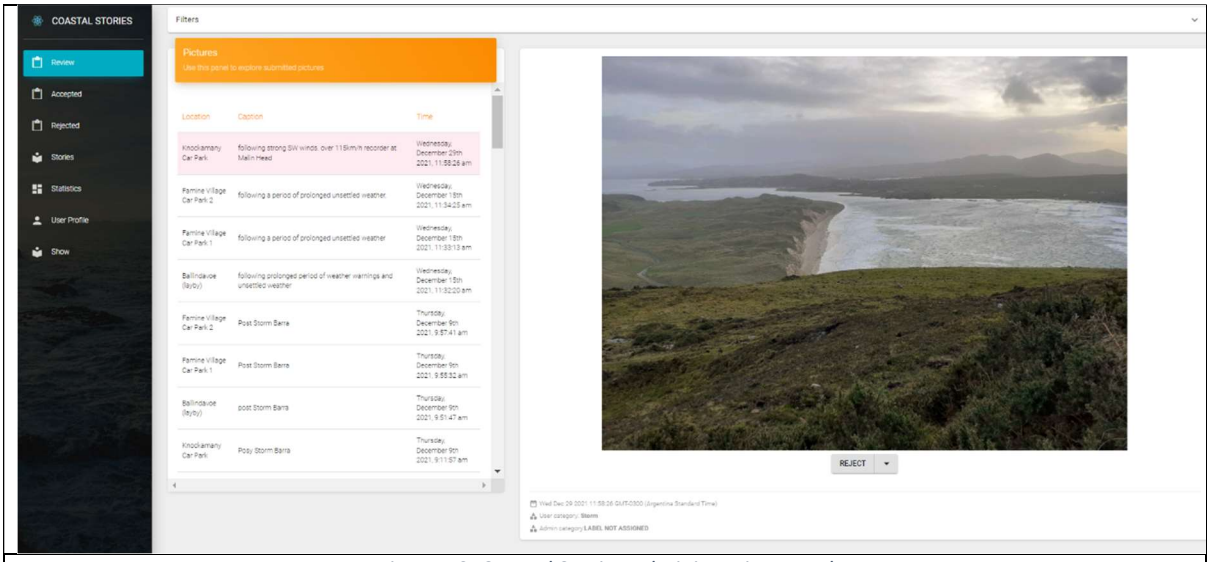



*Figure 10. Coastal Stories Administration Panel*

The statistics dashboard presents different charts that can help the administrators to quickly assess the overall image submissions rates, as shown in Figure 11. The top 4 first charts show the image submissions over the last 2 hours, last week, last month, and the total number of images submitted. These figures provide a quick overview of the system. The remaining four bar charts allow for the observation of four different aspects across time: monthly accepted images, monthly submitted images, monthly rejected images, and pending images.

The administrator user can utilize the filters available and presented in Figure 12 to filter the data by dates, location, user categories and admin categories.

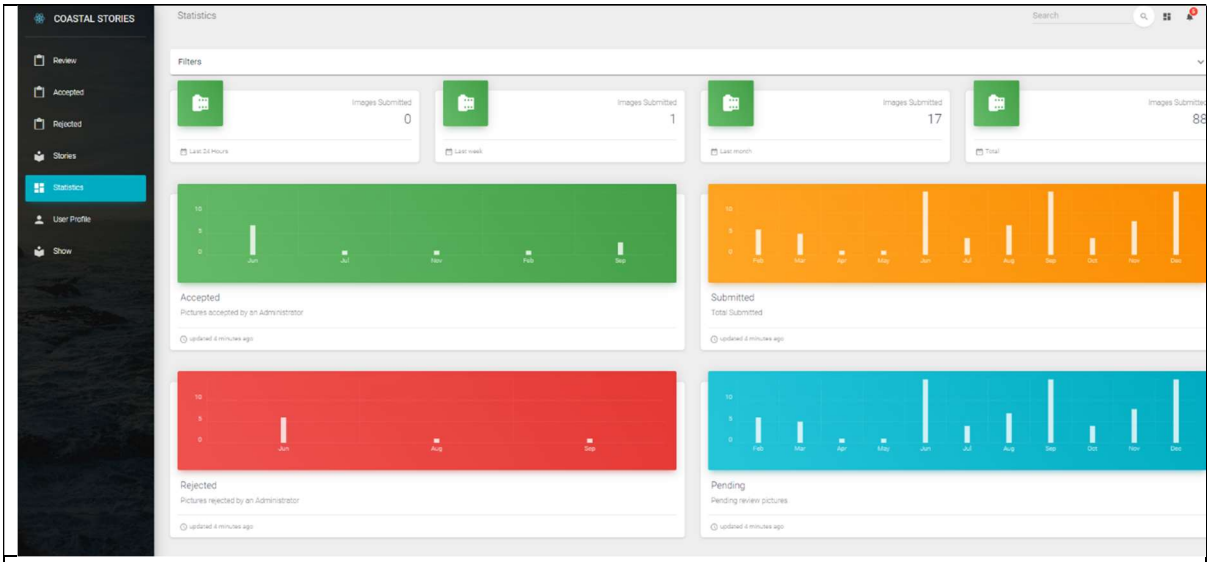



*Figure 11. Statistics Dashboard*

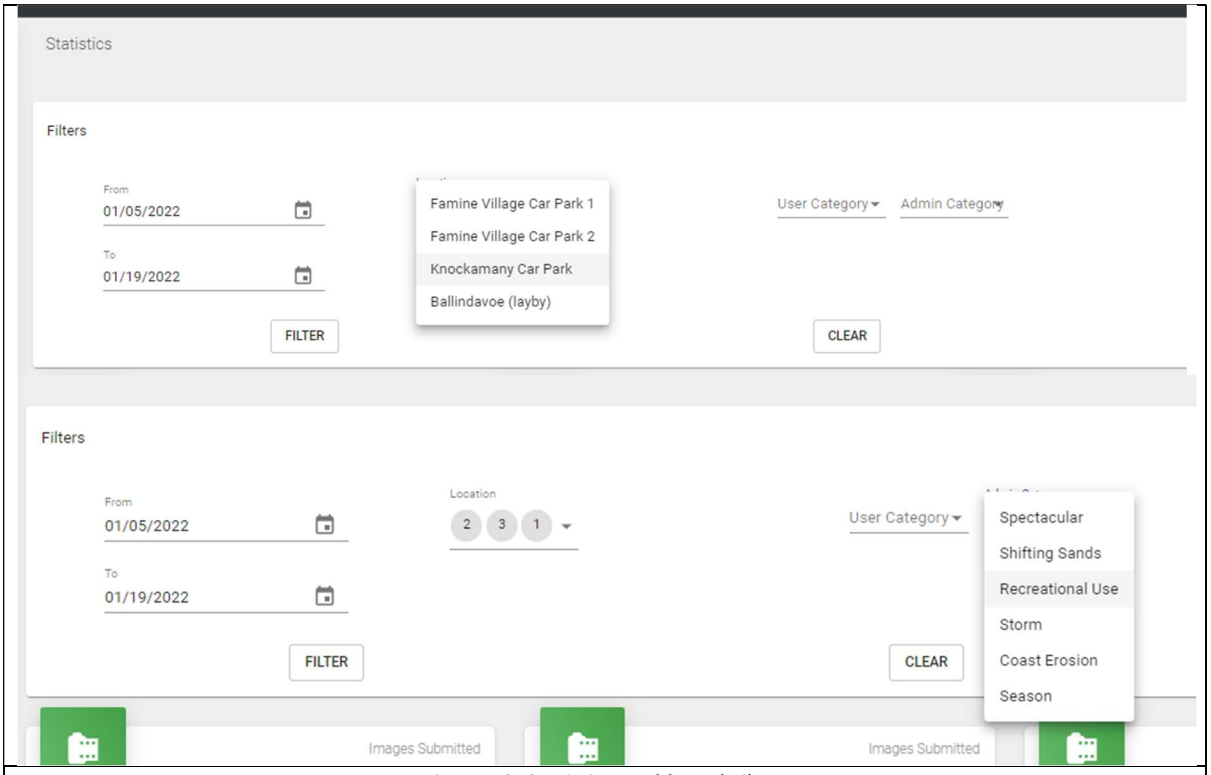



*Figure 12. Statistics Dashboard Filters*