www.nautilos-h2020.eu

Organisation CNR
Department ISTI

Citizen Science tools and interface

Accompanying report

Date: 5/05/2022
Doc. Version: v.0.6

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101000825 (NAUTILOS). This output reflects only the author's view and the European Union cannot be held responsible for any use that may be made of the information contained therein.

Document Control Information

Settings	Value
Deliverable Title	Citizen Science tools and interface
Work Package Title	Data Management
Deliverable number	D8.8
Description	This deliverable will consist of the specific tools and interface for supporting the Citizen Science Campaigns, to integrate the data produced during these activities (see Task 12.2) within the web portal. The interface, realised in Task 8.4, will be based on geo-referenced maps indicating, for instance, plastic litter data. An accompanying report with the guidelines for the usage of these tools and interface will be produced.
Lead Beneficiary	CNR-ISTI
Lead Authors	Gabriele Pieri, Marco Tampucci
Contributors	ETT, NIVA
Submitted by	Gabriele Pieri
Doc. Version (Revision number)	0.6
Sensitivity (Security):	Public
Date:	05/05/2022

Document Approver(s) and Reviewer(s):

NOTE: All Approvers are required. Records of each approver must be maintained. All Reviewers in the list are considered required unless explicitly listed as Optional.

Name	Role	Action	Date
Flávio Martins	Reviewer	*Approve*	05/05/2022

Document history:

The Document Author is authorised to make the following types of changes to the document without requiring that the document be re-approved:

- Editorial, formatting, and spelling
- Clarification

To request a change to this document, contact the Document Author or Owner.
Changes to this document are summarised in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes
V0.1	24/01/22	M. Tampucci, G. Pieri	First structure of the deliverable
V0.3	11/02/22	G. Pieri	Update and finalised structure of the deliverable

V0.4	14/03/22	G. Pieri, A. Novellino	Integrated version with information from ETT
V0.5	30/03/22	M. Tampucci, G. Pieri	Updated version ready for review.
V0.6	5/05/22	G. Pieri	Final version after review

Configuration Management: Document Location

The latest version of this controlled document is stored in <location>.

Nature of the deliverable		
R	Report	
DEC	Websites, patents, filing, etc.	
DEM	Demonstrator	
O	Other	√

Dissemination level		
PU	Public	√
CO	Confidential, only for members of the consortium (including the Commission Services)	

ACKNOWLEDGEMENT

This report forms part of the deliverables from the NAUTILOS project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101000825. The Community is not responsible for any use that might be made of the content of this publication.

NAUTILOS - New Approach to Underwater Technologies for Innovative, Low-cost Ocean observation is an H2020 project funded under the Future of Seas and Oceans Flagship Initiative, coordinated by the National Research Council of Italy (Consiglio Nazionale delle Ricerche, CNR). It brings together a group of 21 entities from 11 European countries with multidisciplinary expertise ranging from ocean instrumentation development and integration, ocean sensing and sampling instrumentation, data processing, modelling and control, operational oceanography and biology and ecosystems and biogeochemistry such, water and climate change science, technological marine applications and research infrastructures.

NAUTILOS will fill-in marine observation and modelling gaps for chemical, biological and deep ocean physics variables through the development of a new generation of cost-effective sensors and samplers, the integration of the aforementioned technologies within observing platforms and their deployment in large-scale demonstrations in European seas. The fundamental aim of the project will be to complement and expand current European observation tools and services, to obtain a collection of data at a much higher spatial resolution, temporal regularity and length than currently available at the European scale, and to further enable and democratise the monitoring of the marine environment to both traditional and non-traditional data users.

NAUTILOS is one of two projects included in the EU's efforts to support of the European Strategy for Plastics in a Circular Economy by supporting the demonstration of new and innovative technologies to measure the Essential Ocean Variables (EOV).

More information on the project can be found at: https://www.nautilos-h2020.eu/

COPYRIGHT

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Among the tools and services foreseen explicitly in WP8, many are introduced and described relative to the activities of Citizen Science in this accompanying report. Different tools have been designed, implemented, and integrated into the NAUTILOS data infrastructure. The collection here described focuses on Citizen Science dedicated services and tools implemented for uploading and analysing data produced during the various Citizen Science campaigns. The related tools allow for data acquisition, management, and visualisation, but they mainly work as a common entry-point to operate as a bridge towards the NAUTILOS data infrastructure. The tools also include the integration of two physical devices: a smartphone microplastic NIR scanner for the identification of plastics allowing for widespread usage; and a customised device for detecting the presence of phytoplankton in a water sample.

The developed devices, tools and services have been developed to serve during the Citizen Science campaigns that will produce data in the context of Tasks 10.4 and 12.2. A primary goal has also been to distribute and make available the acquired information through a graphical interface, which integrates a map and indicates the locations and other information related to the campaigns performed (e.g. plastic litter collected, measurements from dedicated instruments, underwater fauna images, …). For this reason, an important aspect taken into consideration has been the integration of the developed tools within the NAUTILOS data management infrastructure developed in WP8.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

Abbreviation	Definition
CS	Citizen Science
ESPCE	European Strategy for Plastics in a Circular Economy
NIR	Near-InfraRed
CCB	Coalition Clean Baltic
GUI	Graphical User Interface
HAB	Harmful Algal Bloom
GPS	Global Positioning System

I. INTRODUCTION: TASK OBJECTIVES AND DELIVERABLE OUTLINE (SECTIONS DESCRIPTION)

This document represents an accompanying report to the interfaces and services developed and dedicated to supporting the Citizen Science (CS) campaigns. Furthermore, the various application and tools defined and implemented during Task 8.5 are presented and their use described.
The activities of Task 8.5 generally focused on providing support to CS campaigns. This support can be described in two domains: the developed software application covering various CS campaigns and other developed devices designed for specific CS activities. For the first domain, a broader and deeper analysis regarding end-users' requirements was needed because of the different fields that the application was intended to be used for. For the other devices, the focus was more on the design, development and testing of the specific devices.

The report is structured as follows: Section II focuses on the analysis and drafting of the requirements of the App, as they were collected with the help of end-users. The App requirements will consider the different types of usage and data to be collected and shared. The final requirements document is then reported.
Section III describes how the application has been designed and its structure following the different requirements emerging from the different typologies of use, focusing on the various use-case scenarios.
Section IV discusses the development of the App. In detail, it describes which framework and libraries have been used, followed by considerations on the development of App functionalities enlightening their peculiarities and the connection with the back-end (i.e. the data management infrastructure developed in Tasks 8.2) to store the produced data.
Finally, in section V, examples are presented concerning the App's use and the other developed devices for specific CS activities.

II. END-USER REQUIREMENTS

1. REQUIREMENTS FROM END-USERS

This section describes the work done in Task 8.5 to define the end-users' requirements. NAUTILOS Project has many different partners involved in CS campaigns. Each one does not necessarily perform the same type of activities or focus on the same kind of citizens or citizens organisations. The CS activities are distributed into two different Work Packages, WP10 and WP12. For WP10, Task 10.4 is devoted to the underwater campaigns, while for the activities in WP12, Task 12.2 will be devoted to the plastics-related campaigns in the context of the European Strategy for Plastics in a Circular Economy (ESPCE).

Even considering the underwater activities, these can belong to different categories of end-users: for instance, there will be divers who will be encouraged to test some of the NAUTILOS project instrumentation, to test and provide both feedback and data; on the other hand, there will be an involvement of diving associations which will be more focused on the annotation of underwater photography to perform crowdsourcing of marine images. These two types of activities will require two different interfaces with different constraints for supplying data.

In addition to this, the campaigns relating to plastics arranged in WP12 add different types of information and data to be collected; finally, not all the campaigns will focus on the same kind of activities.
For instance, some will include microscopes data, like the *@lgawarning* activities; others will have information from the Near-InfraRed (NIR) scanner; others will foresee manual counting and description of plastics, and more initiatives could be arranged.

Considering this, we arranged several meetings and discussions to analyse and describe the end-user requirements to cover all possible scenarios and data retrieval. Ensuing incremental versions of the document have been built from a base discussed between the IT technical partners of NAUTILOS (namely CNR-ISTI, DFKI and ETT) and the other partners involved in the CS activities (namely HCMR, NIVA, CNR-ISMAR and EurOcean).

The following points were decided during the discussions and have been taken as the main basis during the drafting of the requirements document and subsequently the development of the application:

- Development of a mobile application for the CS web-based platform since CS will be mainly collecting their data on the beach;
- Give the option to the users to collect data (e.g. underwater or on the beach) and upload them later (when an internet connection will be available);
- Allow for either automatic GPS positioning or manual insertion of close-by position;
- Creation of a map where each entry will appear as a 'pin';
- Possibility of assigning colour-coded status to a location, also depending on the different types of CS campaigns;
- Tool for correcting mistaken entries;
- Support for the creation of personal accounts to identify registered users' entries (taking into consideration ethical issues and GDPR);
- Avoid duplication of other similar initiatives.

Concerning the last item, a brief but specific clarification must be made. Many initiatives exist or have already brought to the delivery of tools, applications, platforms devoted to fighting marine litter in general terms. The Coalition Clean Baltic (CCB), a network of 23 organisations for promoting the protection and improvement of the environment and natural resources of the Baltic Sea Area, has surveyed several of these applications (more than 15). But almost all of them are extremely specific and are dedicated to a single task, mainly covering plastic/litter collection. Among the main initiatives worth being cited are the Litterati database and App [1] and the Marine LitterWatch mobile app [2] developed by the European Environment Agency to strengthen Europe's knowledge base and support European policymaking.

Our primary goal in this development was to support various CS activities arranged by different partners in the NAUTILOS project and with different focuses. Still, the goal is not to provide general tools that anyone could use to increase and collect information. The heterogeneity of CS activities in NAUTILOS entails the need to deliver a broader instrument that allows to acquire data more homogeneously and export them to the web platform, which also provides for the sharing of data collections in a standardised way.

Aiming to guarantee more straightforward access to all scientific data collected inside the project, the data resulting from the CS activities will be compliant with the ERDDAP data format [3].

2. FINAL REQUIREMENTS DOCUMENT

The outcome of the mentioned discussions and analysis brought to the first version of the requirements, where four main areas/scenarios were identified:

1. **CS plastics-related campaigns** (WP12 – Task 12.2)
2. **Divers' campaigns** (WP10 – Task 10.4)
3. **Crowdsourcing for visual marine image annotations** (WP10 – Task 10.4 and WP8 Task 8.5)
4. **Algal bloom related campaign** (WP10 – Task 10.4 and WP8 Task 8.5)

The focus of the discussion, following this high-level distinction, was brought into two different directions: (i) defining basic standard features to be included in all the different scenarios and (ii) refining and detailing each specific scenario requirement.

For the standard features, **date**, **location** and **coordinates** have been identified, which can be added automatically online or offline (i.e. providing the manual addition of GPS coordinates).

Concerning the other specific features for each scenario, a minimal set of mandatory features for each area was decided, while the other features would be optional in terms of data insertion by end-users.

In the following Table 1, the finalised requirements for the CS application are reported, divided by different areas/scenarios, and highlighting the mandatory/required data (shown in a bold and underlined font) with respect to the optional data. Data in red in the table represent the above-mentioned standard features for all scenarios. Furthermore, the main partners deputed to the CS activities added a few other features that they considered optional additions for the developed App (in the table, these are reported in italics). The addition of these features to the App is discussed in the following section, where the design of the interface and its easiness and usability is presented and analysed.

Table 1: Final requirements document for CS activities in various identified scenarios.

Scenarios	Input data
Plastics-related campaigns (WP12 – Task 12.2)	**Type** of identified plastic litter through a list of options: bottles, bottle caps, cups, straws, fishing gear, plastic bags, containers and boxes, food wrapping, cigarette butts, toys, single-use cutlery, gloves, rope, shoes, tyres, etc. *Include microplastics as an option (but not necessary to identify origin).*
	Quantity (e.g. number of items, weight) for each type of plastic

	Date
	Coordinates
	Location
	Beach name
	Width of beach surveyed (from which data were collected)
	Length of beach surveyed (from which data were collected)
	Description of beach surroundings: e.g. sand dunes, trees, sea-side road, marina, anchoring boats, restaurants/bars, sun beds, etc.
	Description of beach sediment: sand, rocky, pebbles
	Weather: sunny, windy, rain, snow
	Upload photos of the beach or litter (optional)
Divers' campaigns (WP10 – Task 10.4)	**Type** of measured parameter (e.g. temperature, salinity, chlorophyll – to be identified with sensor providing partners)
	Numerical value (and units) of a measured parameter
	Date
	Coordinates
	Location
	Depth
	Description of sea bottom: mud, sand, rocky, meadows
	Weather: sunny, windy, rain
	Sea status: visibility/turbidity
	Upload photos of marine species (optional)
Crowdsourcing for visual marine image annotations	**Image acquired**
	Species/taxon name identified from images
	Numerical value (e.g. number of individuals, % surface cover)
	Date (when the image was taken /when data were submitted)

(Task 10.4 and 8.5)	Coordinates
	Location (where the image was taken /where data were submitted)
	Depth
	Intertidal/subtidal/ deep-sea
	Description of sea bottom: mud, sand, rocky, caves, meadows etc
	Weather: sunny, windy, rain
Algal bloom related campaign (Task 10.4 and 8.5)	**Image acquired at macro and microscopic level**
	Species/taxon name identified from images
	Numerical value (e.g. number of individuals, % surface cover)
	Date (when the image was taken /when data were submitted)
	Coordinates
	Location (where the image was taken /where data were submitted)
	Depth
	coastline or offshore
	Description of sea bottom: mud, sand, rocky, caves, meadows etc
	Weather: sunny, windy, rain

III. STRUCTURE/DESIGN OF THE APPS

The application has been designed and structured starting from the final requirements document. Due to the difference occurring in the fields proper of each identified scenario, the application manages each insertion form separately from the others.
Following the different scenarios, the app comprises five sections: the first four are relative to the earlier scenarios, while the last one is dedicated to viewing the inserted reports.
In the following subsections, the typology of use of the application and the arrangement of each scenario are described.

1. TYPOLOGIES OF USE

The application is designed to provide two main typologies of use: storing and visualisation.
The storing typology is composed of four independent main sections (i) Plastics, (ii) Diver campaigns, (iii) Image Annotation and (iv) Algal bloom; on the other hand, the visualisation is composed only of one section. Each section can be accessed from the main page by simply pressing the dedicated button and, depending on its type, is composed of one or more screens.

The storing sections offer the user the possibility, through their screens, to compile a set of dedicated fields that will be used to generate a record to be stored in the data server. These sections and relative screens are similar and offer a set of dropdown menus and text fields that can be optional or mandatory. The mandatory fields vary depending on the selected scenario. Along with dropdown menus and text fields, the application manages special values: (i) latitude and longitude and (ii) images. Indeed, for each scenario, the user has to specify the coordinates of interest; in this release version, the application acquires and uses the GPS position of the device as a report referencing coordinates. Moreover, each scenario offers the possibility to enrich the report with a significant image (the image is mandatory in case of Image annotation and Algal bloom reports); thus, each storing section provides two more screens to allow the user to take a picture through the device camera or to pick an existing one from the device gallery. Once all the data are compiled, the application exploits a dedicated REST API to store them on the data server.

On the other hand, the visualisation section offers the user a quick look at the previously stored reports. The application has to query the data server to obtain all the reports sent by the user. These reports, characterised by different colours depending on the belonging scenario, are displayed on a map; by selecting one, it is possible to obtain further information such as the report type and the referenced date. The application will show a restricted set of the available data focusing only on the ones sent by the operator using the application; this strategy has been adopted to avoid an excessive burden on the application. From this visualisation section, only a few further information will be prompted to the user leaving a complete and detailed view of the reports to the Graphical User Interface (GUI) described in deliverable D8.7

2. ARRANGEMENT OF EACH USE-CASE SCENARIO

For each possible use case scenario (plastics, diver campaign, image annotation, algal bloom), application sections and subsections are described enlightening their peculiarities. Each scenario is described independently in a dedicated sub-section.

2.1. Plastics

Plastic campaigns are devoted to identifying and classifying the presence of plastics in coastal areas. Each report contains information about the presence of plastic, the location and the period where and when the campaign took place.
Regarding the presence of plastics, users have to choose their type and the approximate quantity found; it is also possible to enrich the report with a representative image of the discovered plastics. In the case of different kinds of plastic, the user has to insert as many reports as identified types of plastic.
The location where the campaign took place is, by default, the geographical coordinates (mandatory field) acquired by the GPS, but the users can modify it. Users can also add optional information such as the location and the beach name; it is also possible to specify the portion of the beach surveyed by the campaign. Other managed details are the description of the beach sediment and its surroundings and the weather condition during the campaign.
Users can enrich the report by adding a representative image that could be acquired directly from the camera or picked from the device's gallery.
Finally, the date when the campaign took place is a piece of mandatory information.

2.2. Diver Campaign

During these campaigns, divers receive one or more sensors, some of them developed among the project activities, that will be tested. These sensors can usually acquire information about the water (e.g. temperature, salinity and chlorophyll). Once the campaign is over, the diver will download the data acquired from the sensors and upload it through the application to the data server.
Thus, each report must specify the measured parameter and its value; in the case of multiple measured parameters, it is necessary to insert as many reports as measured parameters. Along with the measured parameter, it is mandatory to specify the geographical coordinates and the date when the campaign took place. Coordinates and date, by default, assume the actual position (retrieved by the GPS of the device) and date, but users can specify different coordinates or date, for instance, in the situation of data uploaded after the conclusion of the campaign.
Like the previous scenario, the user can enrich the report by adding a representative image that can be picked from the device's gallery.
Along with the mandatory data, the user can also specify the geographical area where the campaign took place and other details such as the depth where the parameter was measured, the seabed type, the level of water visibility and the weather conditions.

2.3. Image Annotation

Image annotation is based on crowdsourcing to collect many images related to information about the species identified in it. The images can come from divers, environmental operators or volunteers.
In this scenario, the image is a mandatory field. As in the previous scenarios, the user can capture a photo through the device camera or pick one from the device's gallery.
The user must specify the identified species (or its taxon name) for each image and a numerical value representing the number of occurrences or the percentage of the acquired area covered. Along with that, also the coordinates and date fields are mandatory and follow the same logic as the previous use case scenario.

The user can also specify the geographical area where the campaign took place and other details such as the depth of the picture, sea level, seabed type, and weather conditions.

2.4. @lwarning/Algal bloom

@lgawarning is designed for serving two main case scenarios: citizen-science activities and coastal monitoring.

The citizen-science use case implements the monitoring of the presence of phytoplankton in a water sample. The citizen-scientist is asked to collect the water sample and analyse it by the @lgawarning application combined with a portable microscope (DIPLE). The system takes a high-resolution photo of the sample that is enriched with key metadata (latitude, longitude, datum) for reports for further processing.

Routine coastal monitoring use case implements an expert use of the system: the operator collects the sample, verifies and documents the presence of algal species in the sample. Data is enriched with metadata and stored for further analysis and reporting purposes. This easy-to-use workflow (Figure 1) enables in situ bloom events and evolution monitoring. Notably, this workflow is particularly relevant in the case of Harmful Algal Bloom (HAB) monitoring and management.

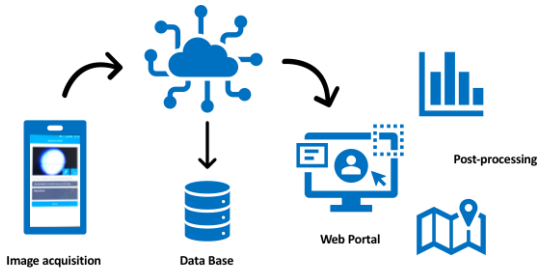

Figure 1: Scheme of @lgawarning data acquisition process

Collected data are then organised into the data server and discoverable via a web portal, where the users can find reports providing information on the composition and distribution of species.

IV. DEVELOPMENT OF THE CS SMARTPHONE APP

This section concerns the app's development and its connection with the back-end describing the adopted framework and enlisting the app functionalities related to the libraries exploited for the development.

1. "MATERIALS"

Mainly due to the image annotation scenario, where a crowdsourcing campaign is required to obtain as much data as possible, the citizen science application must be accessible to the majority of the people and widely used. Thus, due to the diversity of the available devices, it

has been required to develop the application exploiting a system-independent framework such as Flutter[1] and React Native[2].
Indeed, Flutter and React Native frameworks allow the development of applications that can run both on Android and iOS devices. Flutter is a framework developed by Google and based on DART language; meanwhile, React Native is designed by META and based on ReactJS. Both frameworks are almost equivalent, and both offer pros and cons that cannot allow making a clear choice on what is more convenient to be used. Thus, React Native has been chosen due to our significant expertise with this framework.
As mentioned before, React Native is based on React JS, which is a Javascript library used to implement user interfaces. It allows for the rapid development of complex applications that can interact with the device's sensors; the application can be enriched with peculiar functionalities and visual effects thanks to additional libraries developed by its active community that can be easily imported. Finally, developed applications can be built both for Android and iOS without any adjustments or dedicated code to be written.
The screens of the application, in general, are composed by 4 main sections: (i) imports, (ii) code executed during screen loading or in a particular case, (iii) visual content and (iv) style of the visual content. Figure 2 is an excerpt of the application's main screen code. In details:

- "*useEffect()*" is a function that is called at verifying of specific condition: the "[state.response]" in the example indicates that the function useEffect() will be called every time the screen is loaded and every time that the state.response value changes.
- the code inside the instruction *"return*" generates the visual content of the screen. "*<InsertButton>*" is a component devoted to the generation of the buttons used to explore the application section (Figure 3); *"name*", "*navigateTo*", "*navigation*" and "*id*" are parameters given to the component that are used for the button generation and correctly address the navigation inside the app.
- "*styles*" specifies the visual style and format of the screen element. It is a javascript object structured like a CSS documents

[1] https://flutter.dev/
[2] https://reactnative.dev/

```
import React, { useState, useEffect, useContext } from 'react'
import { View, Text, StyleSheet, Image } from 'react-native'
…
const HomeScreen = ({ navigation }) => {
    const [location, setLocation] = useState(null);
    const [errorMsg, setErrorMsg] = useState(null);
    const { state, addCameraPermission, addMediaLibraryPermission } = new useContext(AddDataContext);

    useEffect(() => {
        (async () => {
            let locationStatus = await Location.requestForegroundPermissionsAsync();
            if (locationStatus.status !== 'granted') {
                setErrorMsg('Permission to access location was denied');
                return;
            }
            …
        })();
    }, [state.response]);

    let text = 'Waiting for coordinates...';
    if (errorMsg) {
        text = errorMsg;
    } else if (location) {
        text = "";
    }

    return (
        <View>
            <Image source={require('../../assets/Nautilos.png')} style={styles.centeredLogo} />
            {location ? <Text style={styles.paragraph}>Coordinate: {location.coords.latitude}, {location.coords.longitude}</Text> :
<Text style={styles.paragraph}>{text}</Text>}
            <InsertButton
                name={"Plastics"}
                navigateTo={"Plastics"}
                navigation={navigation}
                id={1}
            />
            …
            <InsertButton
                name={"View Map"}
                navigateTo={"Map"}
                navigation={navigation}
                id={5}
            />
        </View>
    )
}

const styles = StyleSheet.create({
    centeredLogo: {
        width: 300,
        height: 50,
        alignSelf: 'center',
        marginTop: 100,
        marginBottom: 15
    },
    …
})
```

Figure 2 Excerpt of home screen code

```
const InsertButton = ({name, navigateTo, navigation, id}) => {

   return (
      <View>
         <TouchableOpacity style={styles.button} onPress={() => {
            (async () => {
               let { status } = await Location.requestForegroundPermissionsAsync();
               if (status !== 'granted') {
                  setErrorMsg('Permission to access location was denied');
                  return;
               }

               let location = await Location.getCurrentPositionAsync({});
               navigation.navigate(navigateTo, { id: id, latitude: location.coords.latitude, longitude: location.coords.longitude
})
               //console.log(`Segnalazione ${id} alle coordinate ${location.coords.latitude}, ${location.coords.longitude}`)
            })();
         }}>
            <Text style={styles.buttonText}>{name}</Text>
         </TouchableOpacity>
      </View>
   )
}
```

Figure 3 Code excerpt relative to the InsertButton component

2. "METHODS"

As described in section III, the application provides two main typologies of usage: storing and visualisation. Storing is based on the four identified scenarios and provides a dedicated subsection of the application for each of them. Each section is similar to the others due to managing similar content.

In addition to the textual fields and dropdown menus, storing sections provides three particular fields: coordinates, date and image.

Indeed, the user has to select the coordinates, and the date relative to the campaign and can (mandatory for the Image Annotation and Algal bloom scenarios) enrich the report with a significant picture.

As mentioned before, the user can select the coordinates relative to the report, but by default, the actual geographical position is provided; the application uses the expo-location library[3] to access the device's GPS and retrieve the current coordinates. Figure 4 contains the excerpt of code used to get the coordinates; in detail, the "*requestForegroundPermissionsAsync()*" function display a pop-up with the request of the permissions required from the application to access GPS information; through the function *"Location.getCurrentPositionAsync({})*", the coordinates are acquired from the GPS sensor.

```
let locationStatus = await Location.requestForegroundPermissionsAsync();
if (locationStatus.status !== 'granted') {
        setErrorMsg('Permission to access location was denied');
        return;
}
let location = await Location.getCurrentPositionAsync({});
setLocation(location);
```

Figure 4 Code excerpt used to get coordinates from the GPS sensor

[3] https://docs.expo.dev/versions/latest/sdk/location/

For each report sent to the data server, the user must specify the relative date through a calendar form. The calendar form is obtained through the library datetimepicker[4]; Figure 5 shows the function and code used to display the calendar form and pick the date selected by the user. In detail, the "*TouchableOpacity*" is composed of a calendar icon, and pressing it activates the "*DateTimePicker*" that generates the calendar view; the selection of a date raises a call to the "*onChange*" function that is in charge to write the date selected by the user in a dedicated variable.

```
const onChange = (event, selectedDate) => {
   const currentDate = selectedDate || date;
   setShow(Platform.OS === 'ios');
   setDate(currentDate)
   callback(currentDate);
};
…
<TouchableOpacity onPress={showDatepicker} style={styles.calendarIcon}>
   <FontAwesome name="calendar" size={24} color="black" />
</TouchableOpacity>
{show && (
   <DateTimePicker
      value={date}
      mode={mode}
      is24Hour={true}
      display="default"
      onChange={onChange}
   />
)}
```

Figure 5 Code excerpt used to compile the date form

Each report can (or has to) be enriched by a significant picture; thus, the application has to provide the functionality to take a new photo from the camera or to pick an existing image from the device gallery.

```
<Camera style={styles.camera} type={type} autoFocus={Camera.Constants.AutoFocus.on} ref={(r) => {
   camera = r
   }}>
      <View style={styles.buttonContainer}>
      <TouchableOpacity
         style={styles.flipButton}
         onPress={() => {
            setType(
               type === Camera.Constants.Type.back
                  ? Camera.Constants.Type.front
                  : Camera.Constants.Type.back
            );
         }}>
         <Text style={styles.text}> Flip </Text>
      </TouchableOpacity>
      <TouchableOpacity onPress={snap} style={styles.snapButton}>
         <Entypo name="camera" size={40} color="white" style={styles.cameraIcon} />
      </TouchableOpacity>
   </View>
</Camera>
```

Figure 6 Camera component.

Photo capture is obtained through two different libraries: expo-camera[5] and expo-media-library[6]. Expo-camera generates a component in the application containing the camera view,

[4] https://github.com/react-native-datetimepicker/datetimepicker
[5] https://docs.expo.dev/versions/latest/sdk/camera/
[6] https://docs.expo.dev/versions/latest/sdk/media-library/

which is in charge of taking the photo. On the other hand, media-library is in charge of storing the taken picture in the device gallery. Before the camera can be used and the gallery accessed, the user must provide the relative permissions to the application; this can be done similarly to the one shown in Figure 4 (relative to the access to the GPS sensor). Figure 6 contains the excerpt of code used to generate the component dedicated to show the camera view. Two buttons are also displayed; the first one ("*Flip*") is used to switch between the rear and front camera, the second one is used to take the photo.

```
let snap = async () => {
   if (camera) {
      const options = { quality: 1, base64: true };
      let photo = await camera.takePictureAsync(options);
      if (state.mediaLibraryPermission) {
         let asset = await MediaLibrary.createAssetAsync(photo.uri);
         let album = await MediaLibrary.getAlbumAsync('Nautilos')
         let albumAssets = null
         if (album === null) {
            MediaLibrary.createAlbumAsync('Nautilos', asset, false)
            .then(async () => {
               addImage({ uri: asset.uri, base64: photo.base64 })
               navigation.navigate(state.navigateBack, { image: asset.uri })
            })
            .catch(error => {
               console.log('err', error);
            });
         } else {
            MediaLibrary.addAssetsToAlbumAsync(asset, album, false)
               .then(async () => {
                  addImage({ uri: asset.uri, base64: photo.base64 })
                  navigation.navigate(state.navigateBack, { image: asset.uri })
               }).catch(error => {
                  console.log('err', error);
               });
         }
      } else {
         setErrorMsg("Permission to access media library was denied")
      }
   }
}
```

Figure 7 Code excerpt used to take a photo

A dedicated function, named "*snap*" (Figure 7), has been developed to take the photo and save it in the gallery. In detail, the image is taken without quality loss and converted in a *base64* format (see values declared in the "*option*" constant); once the picture is taken, the application generates (if it is not yet available) the "*Nautilos*" album and save in it the photo exploiting the library media-library. Then, the image URI and its base64 conversion are stored into a dedicated variable.

Additionally, the user can select a picture from the ones stored on the device. Indeed, expo-image-picker library[7] is exploited to generate a screen where it is possible to navigate through the device galleries. The application requires the same permissions as the media-library for using this library.

Figure 8 shows the "*openImagePickerAsync()*" function to navigate the galleries and select an image. With the base64 option, the chosen image is also converted into base64 format. As a *snap* function, "*openImagePickerAsync*" stores the image URI and its base64 conversion into a dedicated variable.

[7] https://docs.expo.dev/versions/latest/sdk/imagepicker/

```
let openImagePickerAsync = async () => {
   let pickerResult = await ImagePicker.launchImageLibraryAsync({ base64: true });
   if (pickerResult.uri === undefined || pickerResult.cancelled === true) {
      return;
   }
   addImage({ uri: pickerResult.uri, base64: pickerResult.base64 })
   navigation.navigate(state.navigateBack, { image: pickerResult.uri })
};
```

Figure 8 Code excerpt to access the device gallery

On the other hand, the visualisation modality requires only access to the GPS coordinates and display a map where all the previously stored report retrieved by the data server are represented as markers. The map, and markers, are shown by exploiting react-native-maps library[8].

```
const mapRegion = {
   latitude: navigation.getParam('latitude'),
   longitude: navigation.getParam('longitude'),
   latitudeDelta: 0.5,
   longitudeDelta: 0.5
};
<MapView style={styles.map} region={mapRegion}>
   <Marker
      coordinate={{ latitude: navigation.getParam('latitude'), longitude: navigation.getParam('longitude') }}
      title="Your position" pinColor='red'
   />
   {state.plastics.map((item, index) => {
      return (<Marker
         coordinate={{
            latitude: item.latitude,
            longitude: item.longitude
         }}
         key={index}
         title={`${item.report_type} - Added at ${item.date}`}
         pinColor='blue'
      />);
   })}
…
</MapView>
```

Figure 9 Code excerpt dedicated to the map and markers generation

A "*MapView*" element is used to display a map on the screen through the library. The map is centred on the device coordinates ("*mapRegion*" constant). The map is then populated thanks to the iterative function "*.map*" that, for each element found, generates a marker enriched with report type and date information.

3. CONNECTION WITH THE DATA MANAGEMENT INFRASTRUCTURE

Both storage and visualisation usage typologies of the app require connecting with the NAUTILOS data server to send or retrieve reports. The data server is structured based on the ERDDAP formats. It exposes a set of dedicated HTTP services to acquire the reports and store them into the proper ERDDAP datasets (each identified scenario has a dedicated ERDDAP dataset on the data server) or to return a *GeoJSON* containing all the searched data. Thus, connection management with the data server is a key component of the application.

[8] https://github.com/react-native-maps/react-native-maps

The connection is achieved by exploiting the Axios library[9]. Axios is a Javascript library that uses HTTP protocol to connect with back-end APIs; based on promise objects allows the execution of async commands and non-blocking calls.
The user has to log in to use the application. It is a required passage to maintain information about who has sent the data (this will be crucial for crowdsourcing campaigns because it will allow to distinguish between reports sent by a volunteer or sent by professional operators). The authentication exploits Axios library to send an HTTP POST request to the data server API. The application also connects with the data server API each time a report is sent. Figure 10 shows the code excerpt used to connect to the data server and send the data relative to a plastic report. The "*method*" specifies the type of the requests (POST or GET), "*url*" contains the URL where the request has to be sent and "*data*" contains all the parameters to be sent to store the record. The image will be sent in base64 format inside this request if available.

```
await axios({
    method: 'post',
    url: 'https://62.94.85.14/erddap/tabledap/plastics_type.insert',
    data: {
        plastic_type: plastic_type,
        quantity: quantity,
        date: date,
        latitude: latitude,
        longitude: longitude,
        …
        weather_type: weather_type,
        user: user
    }
}).then((response2) => {
    addResponse({"response":response.data, "report_n":report_n})
}, (error2) => {
    setErrorMessage(error2);
    if (error2.response) {
        console.log(error2.response)
    } else if (error2.request) {
        console.log(error2.request)
    } else if (error2.message) {
        console.log(error2.message)
    }
})
```

Figure 10 Code excerpt to send report to the data server

In the case the data has been successfully sent, the data server will send a response to the application similar to the one shown in Figure 11

```
{
    "status": "success",
    "nRowsReceived": 1,
    "stringTimestamp": "2022-03-24T16:07:27.517Z",
    "numericTimestamp": 1648138047.517
}
```

Figure 11 Example of data server response

The application retrieves previously stored reports in the visualisation section by querying the data server. In this case, the application exploits the Axios library to perform an HTTP GET request. Based on these parameters the server will provide a *GeoJSON* containing all the reports sent by the user.

[9] https://www.npmjs.com/package/react-native-axios

V. USE OF THE APPS

This section describes the use of the developed apps. A section with a sort of user guide will be presented for each app. Regarding the smartphone application, all the usage typologies and identified scenarios will be discussed

1. SMARTPHONE APPLICATION

As discussed before, the application interface has been kept very simple and clear to promote its usage among the general public.
After the project logo splash screen, the user has to insert his username and password to authenticate inside the system. Then, the application connects with the data server, exploiting a dedicated API, to verify user credentials. Then, the user id is reported in each report sent by the user. In the forthcoming release, the application will keep track of user groups providing different insertion forms depending on user belonging groups.

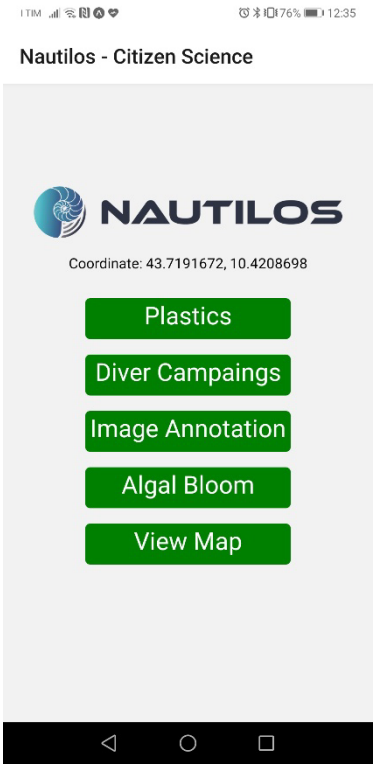

Figure 12 Main page of the application

Once the user is authenticated inside the system, the application's main screen will be displayed (Figure 12). From this screen, the user can access all the functionalities and options provided by the application. The main screen comprises five buttons: the first four is used to send a report to the system regarding one of the four identified scenarios; the last button is used to view all the previous report sent to the system by the user.

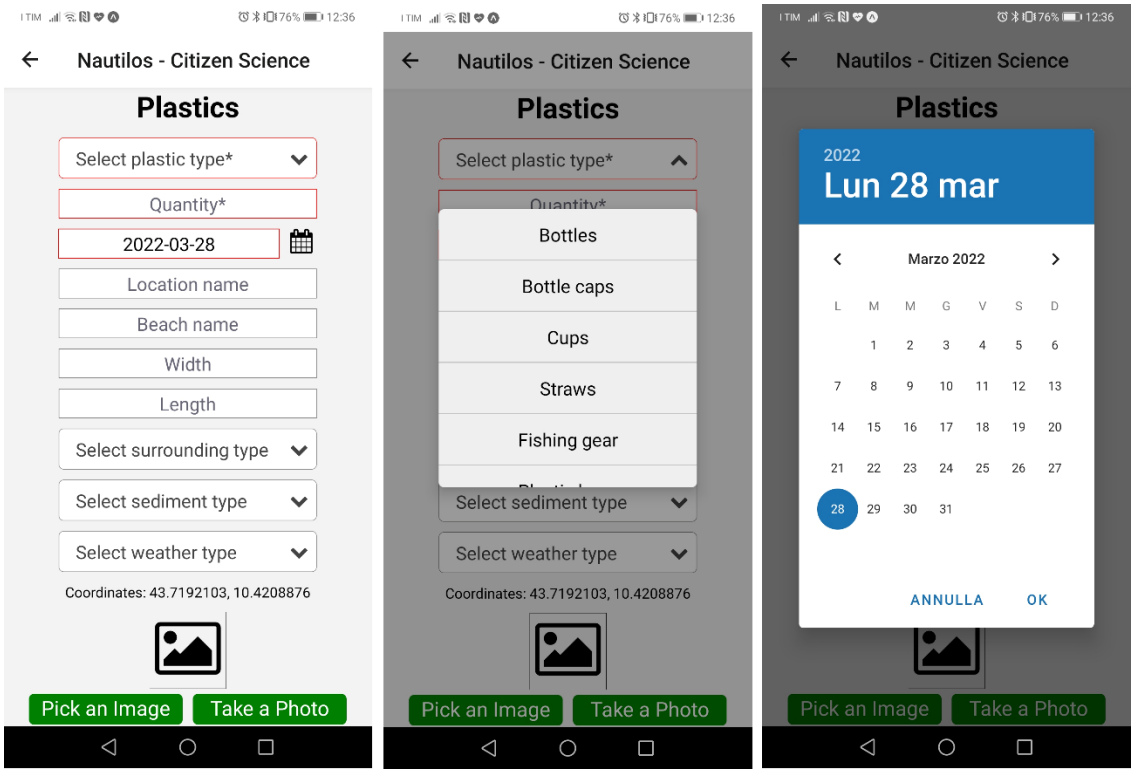

Figure 13 The plastic report form: (a) complete visual on the left; (b) detail of a dropdown menu in the centre; (c) detail of calendar form on the right

By selecting the "*Plastics*" button, the form to send a plastic report will be loaded (Figure 13-a). The user can specify a set of values that will compose the report. The fields with a red border are the mandatory fields. Depending on the nature of the field, the user fills form fields by selecting values from dropdown menus (Figure 13-b) or by inserting free text; the only exceptions are represented by the date form, where a calendar view (Figure 13-c) is shown and by the image selection/capture. The coordinates, reported in the lower part of the screen, are automatically taken by the GPS sensor of the device.
Once the report is completed, the user sends it to the data server, pressing the "Send Data" button. Then, the main screen of the application will be loaded with a message confirming the operation's success.

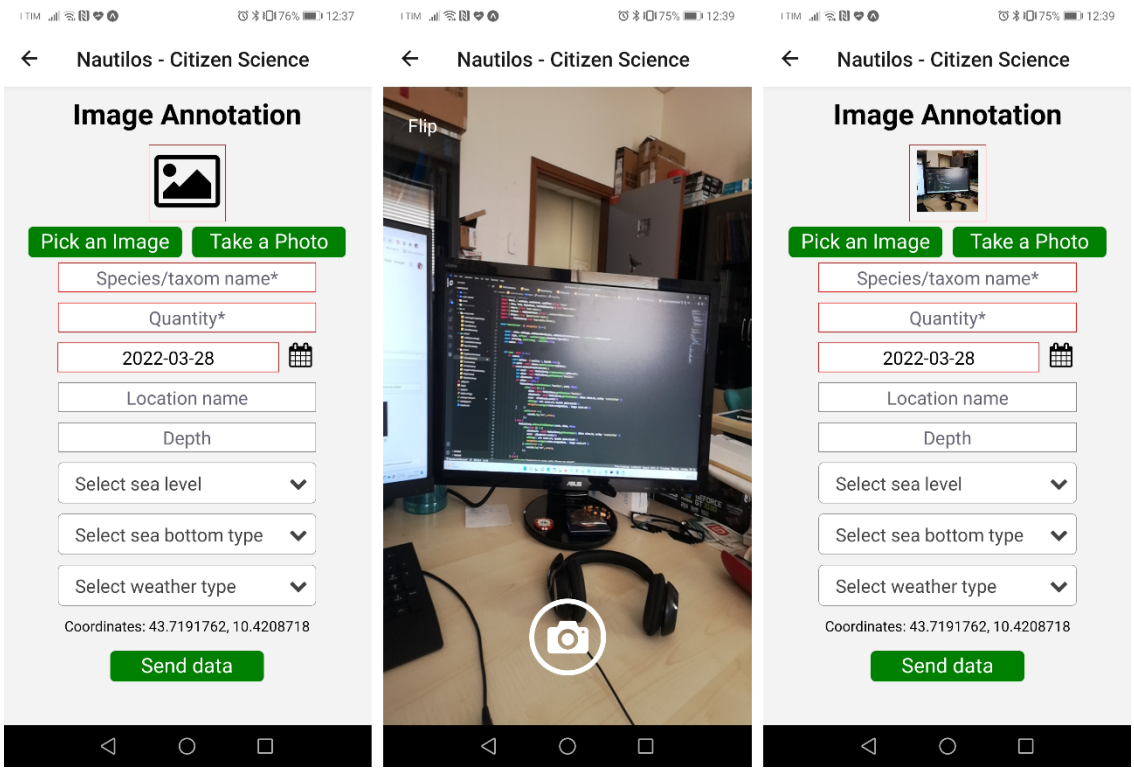

Figure 14 Example of Image Annotation form: (a) complete view on the left; (b) image capture in the centre; (c) complete view with a thumbnail of the captured image on the right

Figure 14 is an example of the Image Annotation scenario. The image selection/capture is placed as the first field in this form because this is a mandatory field. Pressing "*Take a Photo*" will load the camera screen (Figure 14-b). The camera screen is straightforward and shows the camera output with two sensible areas on it: (i) "*Flip*" to switch from the rear to the front camera and vice versa and (ii) the button to take a photo. After taking the picture, the previous screen will be loaded back, and, as can be seen in Figure 14-c, a thumbnail of the image selected/captured will be shown. As mentioned in the previous section, the selected/captured image is converted into base64 format that will be sent to the data server along with other information.

The Diver campaign and Algal bloom scenarios are similar to the Plastics and Image Annotation ones. The only difference is that the user can select the coordinates where the campaign has been conducted through the "*Pick coordinate*" button (Figure 15).

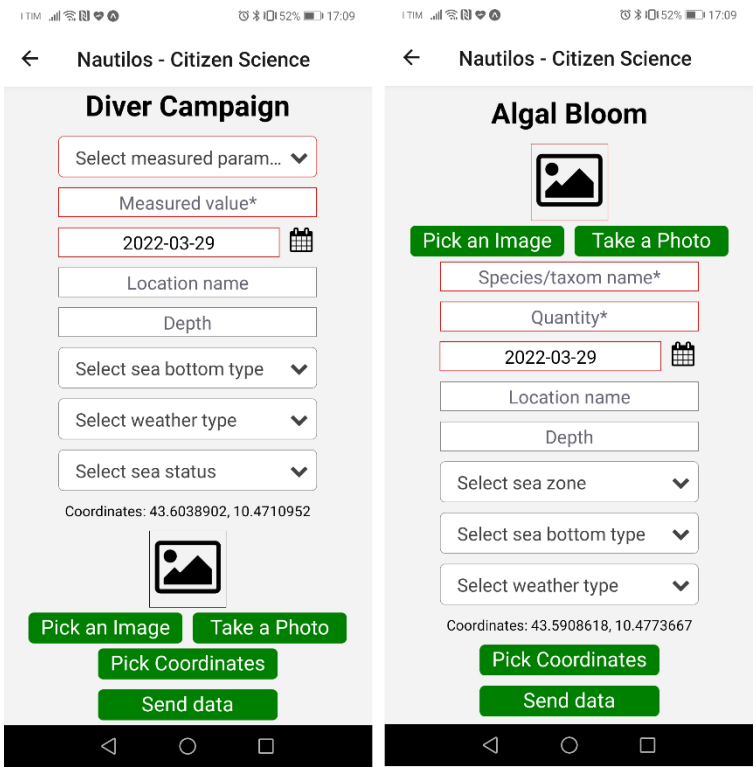

Figure 15 (a) Diver Campaign and (b) Algal bloom scenarios

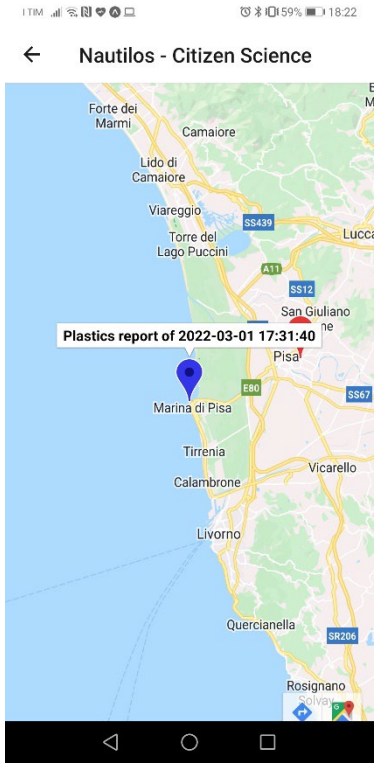

Figure 16 Example of View Map screen

Finally, by selecting the "*View Map*" button, the user can access the previous stored reports. This screen, shown in Figure 16, can be used by the user to have a brief list of his reports already sent to the data server. It is composed of a fullscreen map populated by sets of markers, where each one is characterised by a colour depending on the type of the report.

Selecting a marker, the user can access further information such as the report type and when the campaign relative to the report took place.

2. NIR PLASTIC SCANNING APPLICATION

The smartphone NIR scanner is used to establish the polymer type of plastic fragments, specifically where other identification is not possible. The scanner is especially suited for the further reporting of meso litter (2.5 cm – 5 mm) which are often are mistaken for food by marine species.

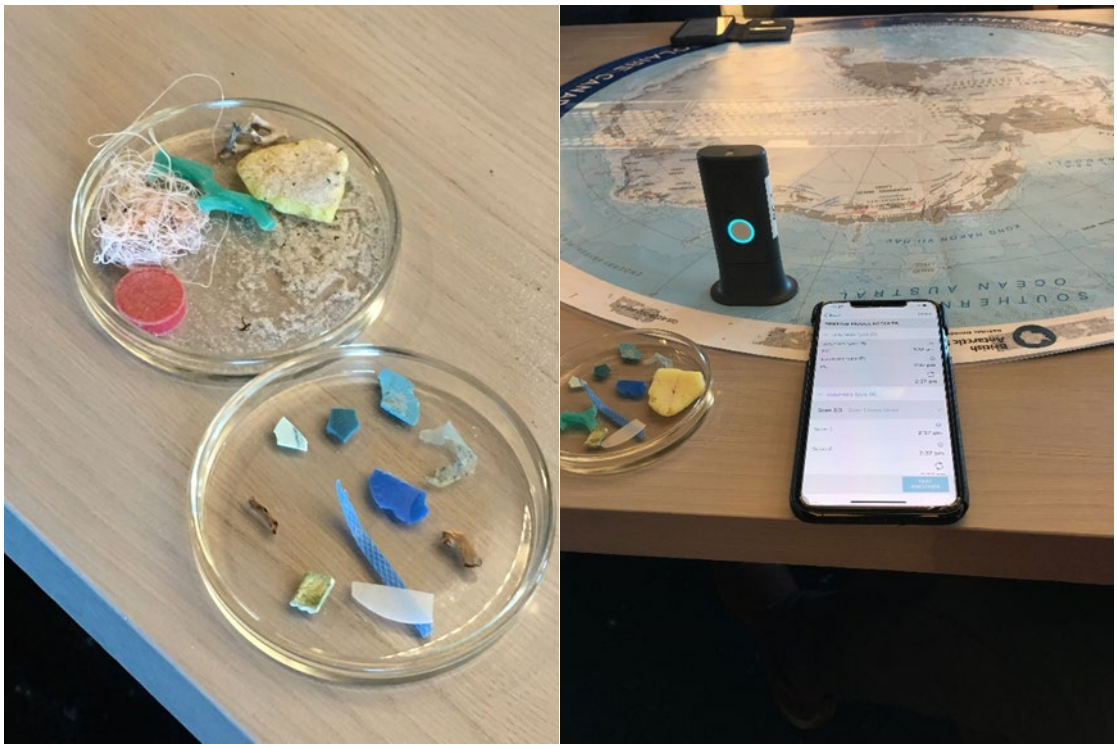

Figure 17. Example of the usage of the smartphone NIR scanner on polar marine samples.

The smartphone app is still under development and tests with citizen scientists and university students were performed under a license agreement to test the plastic litter data base of 407 items and the developed models. The procedure for analysing samples using the smartphone application and NIR scanner is outlined in Figure 18.

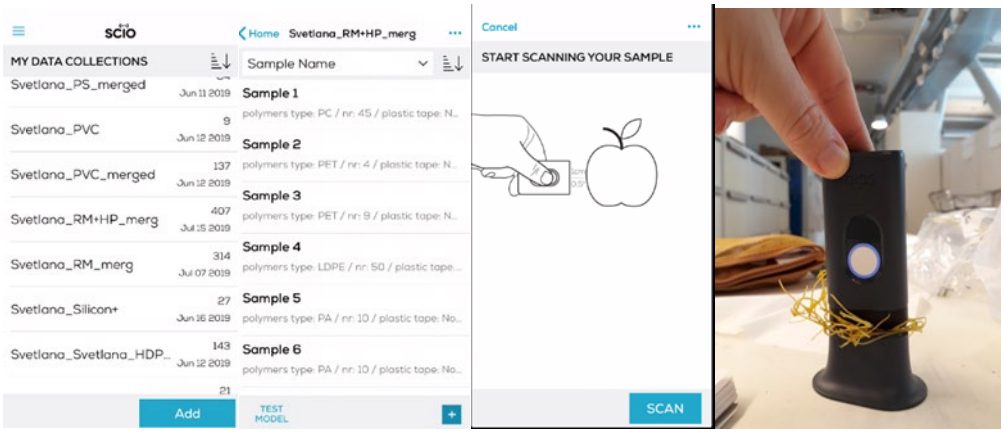

Figure 18. The sequence of scanning a marine litter sample for polymer identification

After analysis, the results are displayed, and the polymer type identification is provided (Fig. 19). In order to reduce the number of false identifications it is recommended to scan the sample three times. When the results from replicate analyses are not consistent the sample should be reported as unidentified.

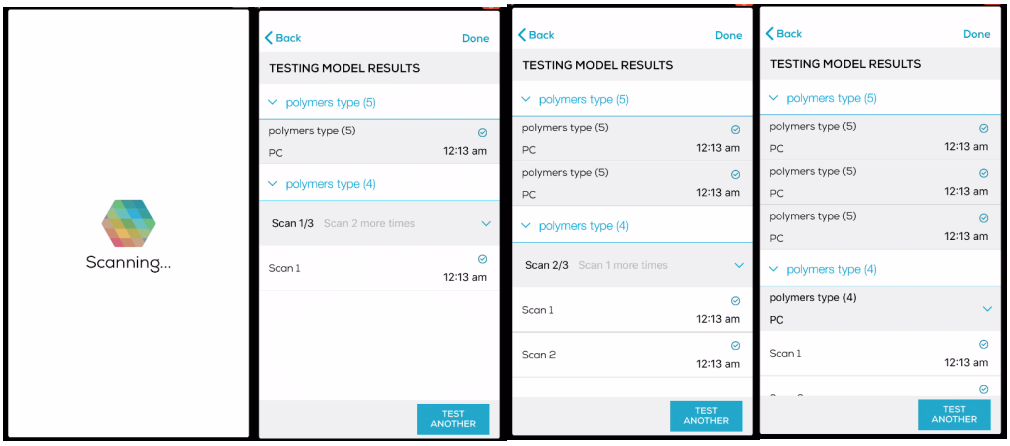

Figure 19. Example of the results of the scan and the polymer type identification in triplicate.

The smartphone NIR scanner was tested at two occasions for the NAUTILOS project. The scanner was used on board the MS Roald Amundsen after collecting (meso plastic) litter samples during a beach cleaning at a remote polar beach. Large items were registered using a beach survey protocol, the smaller items were taken on board for further analysis and discussion on potential sources.

Figure 20. Collecting samples with passengers of the MS Roald Amundsen in a remote polar beach.

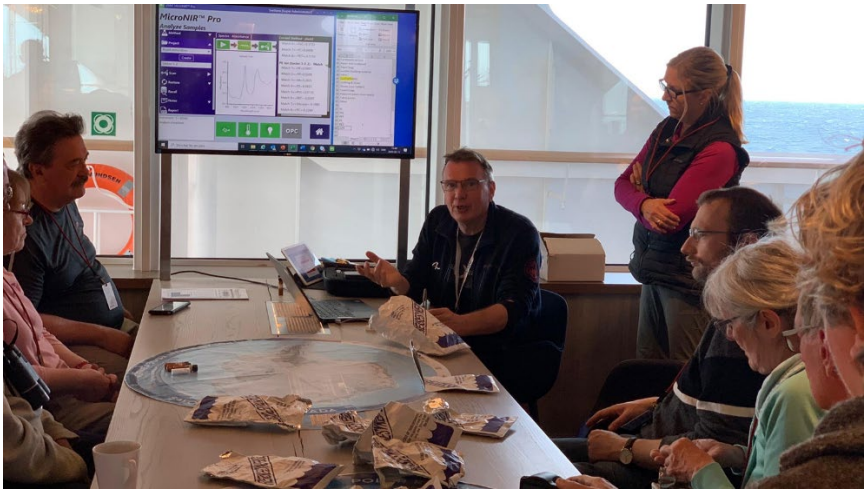

Figure 21. Identification of polymer type and discussion with the passengers on the MS Roald Amundsen

The user friendliness and usability were further tested on a group of university students at the University of Bergen (Fig. 22). Here samples were analysed on site for the polymer type and directly incorporated in a survey protocol. The app, database and uploading details are described in detail in D8.7, further evaluation and use of the scanner is part of D12.3.

Figure 22. Students from the University of Bergen testing the use of the smartphone scanner on site.

VI. CONCLUSIONS

This report represents the accompanying document for the CS tools and services developed during Task 8.4. In particular, the developed tools allow for data acquisition, management, and visualisation. Still, they mainly work as a common entry-point to operate as a bridge toward the NAUTILOS data infrastructure and back-end. Among the tools, the first one is the CS smartphone app, while another two devices are included: a smartphone microplastic NIR scanner for the identification of plastics allowing for widespread usage; and a customised device for detecting phytoplankton in a water sample.
The latter is described in more detail in the deliverable D8.7, but it is reported here as the CS smartphone app has achieved its data transfer integration.

End-users will then test the developed devices, tools and services during specific testing and demonstration campaigns. Feedback and results will bring further improved versions of the tools and services. To not replicate existing applications or services, the proposed CS app has been developed as a fully integrated service within the NAUTILOS data infrastructure and based on the well-known ERDDAP data format, which is designed specifically for environmental data collection. Moreover, the tools cover a wide range of environmental campaigns, from divers to plastic litter collection campaigns.

Despite aiming for wide distribution and greater availability of information to many users, analysis and discussions are being finalised to understand the possible publication of the application on mainstream commercial platforms (e.g. PlayStore, IOS). The use of these platforms would imply other constraints and would not allow having some specific controls over the usage of the developed app. At the same time, it can lead to widespread dissemination, but it could also bring undesired or uncontrolled information from external sources. Distributing an application with restricted access through official stores could be counterproductive. On the other hand, avoiding the distribution through the official store prevents the possibility to install it on iOS devices due to Apple Inc. policies. Distribution possibility and all related issues will be further analysed anyway, and a decision will be made during the NAUTILOS Project. The APK Android installer of the application will be realised and distributed through the official project Web portal.

VII. APPENDIX 1: REFERENCES AND RELATED DOCUMENTS

ID	Reference or Related Document	Source or Link/Location
1	D8.7 Fully developed Graphic User Interface	NAUTILOS OwnCloud (under submission)
2	D8.3 Data Management Workflow	NAUTILOS OwnCloud and SyGMa platform