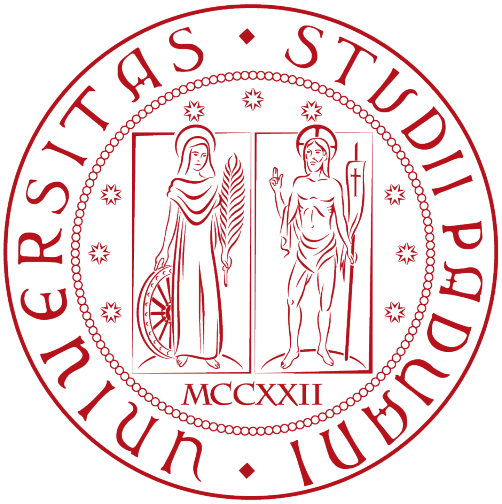

Lost and Found
Mobile Programming and Multimedia

Academic Year 2022 - 2023

Information about the document

Editing	Bacchin Francesco	ID 2089122
	Cavaliere Alessandro	ID 2076771

Period of writing	September 2023

Description

Technical report about the development of the mobile application *Lost and Found* as a project for the *Mobile Programming and Multimedia* course, Master Degree in Computer Science at University of Padua.

Index

1 Introduction

During our university journey, we realized that every time a student loses an item in the classrooms, they report the incident through the University of Padua's Telegram group.

The high frequency of these reports led us to wonder if there was a dedicated application specifically for reporting lost or found items. We noticed that the few available apps on the app stores that deal with this aim are not well-known, as evidenced by their low number of downloads. These applications offer an unappealing user interface and many bugs. For this reason, we set the goal of creating an application that provides an intuitive and user-friendly interface.

We then created *Lost and Found*, an application designed to collect all reports of lost or found items on a single platform. Each user can report a lost item, and thanks to the features available in *Lost and Found*, he will be notified if his lost item has been possibly found by another user. Then the user who lost the item and the user who found it will be able to chat and arrange the return of the item.

Lost and Found enables users to report items of two categories:

- Lost: indicating items that users have lost;
- Found: indicating items that users have found, presumably lost by someone else.

A user can:

- Report the loss of an item:
  1. The user submits a report of a lost item through the app;
  2. The user can search to see if the lost item has been found by another user using the search functionality;
  3. The user will receive a notification when another user submits a report for a found item that might match his lost item;
  4. When the user finds an item that matches one of his lost items, he can make a "claim". The claim process involves answering a question to prove the item's ownership. The user who reported the found item creates an appropriate question and evaluates the correctness of the claims' response.
- Report the finding of an item:
  1. The user enters a report for the found item in the application and creates a question to verify the legitimacy of possible future claims;
  2. The user can check if there is a missing report, for the found item he inserted, using the search functionality;
  3. The user will receive a notification when another user submits a report for a lost item that might match his found item;
  4. When the found item receives a claim from a user, the user who found the item will assess the accuracy of the response to the proposed question. If the response is accurate, the user will be contacted to arrange the return of the lost item. Once a claim has been approved by the user who found the item, then that item won't be visible anymore through the app.

2 Screens and Functionalities

2.1 Initial page, Registration and Login

2.1.1 Initial page

When the user launches *Lost and Found* for the first time, he will see the initial page (fig. 1). This page consists of a brief tutorial presented in a carousel format with three images. Users can swipe through the images by performing an horizontal scroll, a progress bar indicating the number of the current image in the tutorial. A small tutorial is needed, because the average user has never downloaded an application of this kind; therefore we considered it essential to introduce the basic operational principles of the application. A more in-depth tutorial is present inside the application.

On the Initial page, in addition to the tutorial, there are two buttons to navigate to the Login page (presented in section 2.1.3) or the Registration page (presented in section 2.1.2).

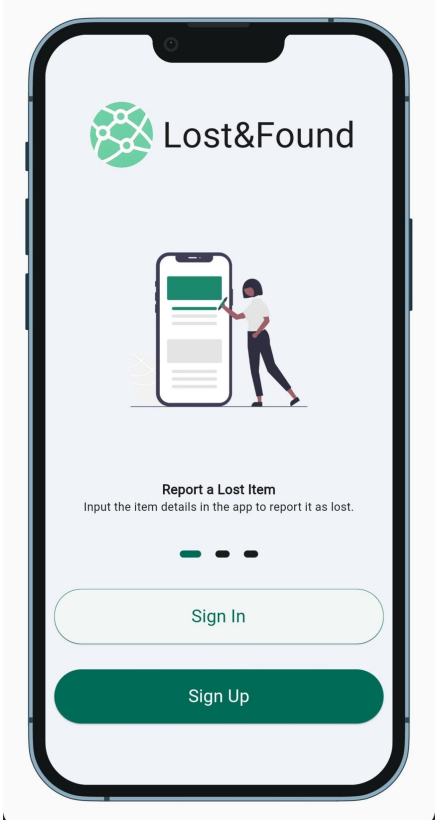

Figure 1: Initial page.

2.1.2 Registration

The registration page (fig. 2) allows the user to sign-up in *Lost and Found*.

For each entered field, checks are performed regarding:

- Username already in use, too short, contains non-alphanumeric characters;
- Invalid email format;
- Insufficiently secure password (minimum of 8 characters, with at least one uppercase letter and one number);
- Confirm password not matching the actual password.

If any of these checks are not met, the user will be notified of the error with a brief message displayed below the respective field where the error occurred (fig. 3). To navigate directly to the Login Page (presented in section 2.1.3), the user can click on a designated TextButton. The text is underlined to make it clear that it is clickable.

Figure 2: Registration page.

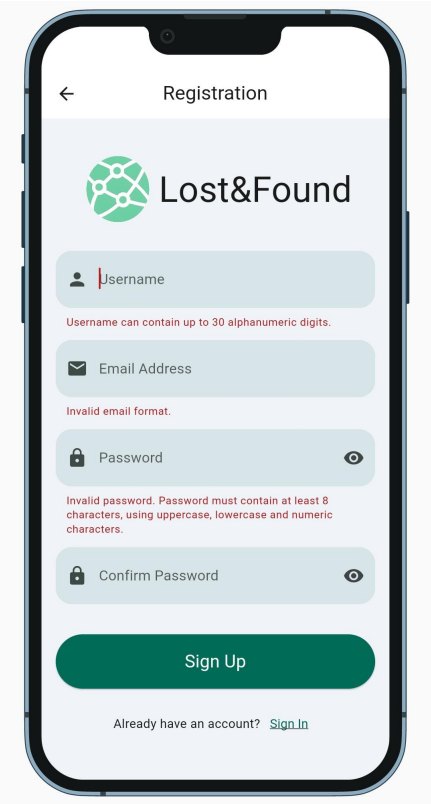

Figure 3: Registration page with errors.

2.1.3 Login

The login page (fig. 4) allows the user to sign-in in *Lost and Found*.

In order to sign-in the user has to enter a valid email or username and a password. If the email or the username are not in the correct format, the user will be notified of the error with a brief message displayed below the entered username (or email) field. If the entered credentials do not match to ones of an existing user or a server error occurs, a bottom snackbar will inform the user of this (fig. 5).

Once the user successfully authenticates, he will be automatically logged in on the next launch of the application.

On this page as well, there is a TextButton available to reach the Registration Page (presented in section 2.1.2) with a single click.

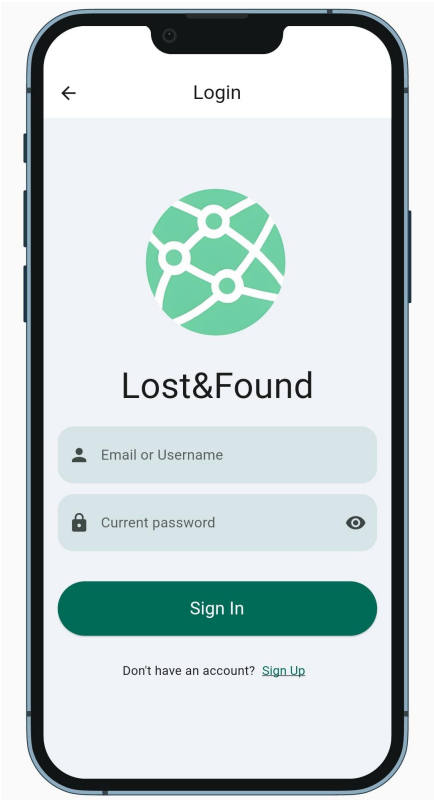

Figure 4: Login page.

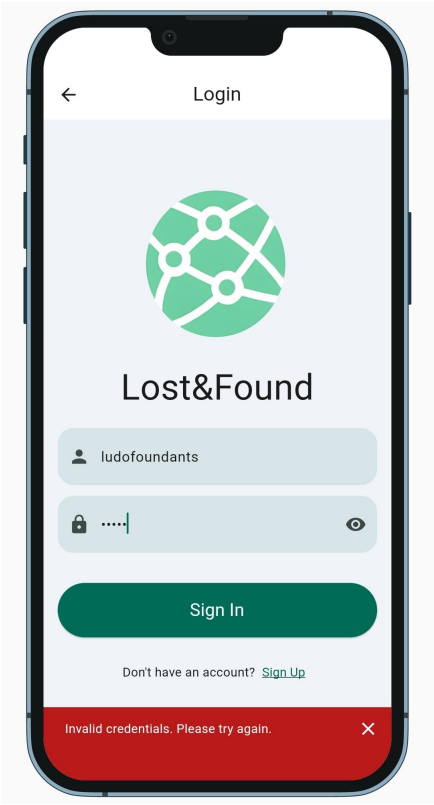

Figure 5: Login page with invalid credential.

2.2 Main content

2.2.1 Home

At the top right of the home page (fig. 6), there are buttons to:

- Go to the news page (presented in section section 2.2.4);
- Go to the claims page (presented in section section 2.2.5).

If the user performs a long press on one of these two buttons, a tooltip will appear with the name of the page to which the button redirects.

In the presence of unread news, there will be a red dot with a number to indicate to the user the number of unread notifications on the page associated with the button. The same applies for unread claims: there will be a red dot with a number to indicate to the user the number of unread claims on the page associated with that button.

Just below, there is a card with an informative purpose. When clicked, the card expands, giving the user the opportunity to select a tutorial to view (fig. 7). The tutorials are available on the tutorial page (presented in section section 2.2.20).

In the central part, there are two sections:

- Lost items: horizontally-scrollable collection of reports of items lost by the user;
- Found items: horizontally-scrollable collection of reports of items found by the user.

Each collections is sorted in a different way, to let the user reach an interested item in quickest possible way:

- Lost items are sorted in descending order of date insertion (first the newest, last the oldest);
- Found items are sorted with two different criteria:
  - First all the reports that have not been solved yet (no approved claims for them), then the resolved ones;
  - Secondly, the reports are sorted in descending order of insertion (first the newest, last the oldest).

Each report is represented by a card, which includes:

- A photo of the item;
- A title associated with the report;
- Only for found items, the number of claims received or if the report has been resolved (a claim for that report has been approved).

Each card is clickable. When clicked, the user will be redirected to the report page (presented in section 2.2.2) for the specific item (referred by the card). To see all the cards, the user can scroll horizontally in the section by doing an horizontal swipe.

If no reports are present in one of the sections, a text will inform the user that there are no reports inserted for that section. Furthermore, there will be a button to allow the user to navigate to the "Insert a report" (presented in section 2.2.15), in order to quickly insert a new report.

Finally, a bottom menu is present, with which the user can navigate through the various pages of the application. A red dot may appear in the top right corner of the Home and Inbox icons, indicating that there are either new claims or unread news (for Home), or new messages (for Inbox) have arrived.

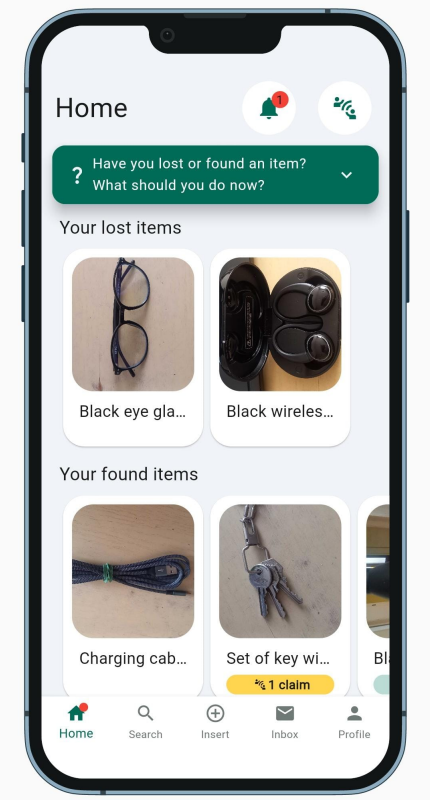

Figure 6: Home page.

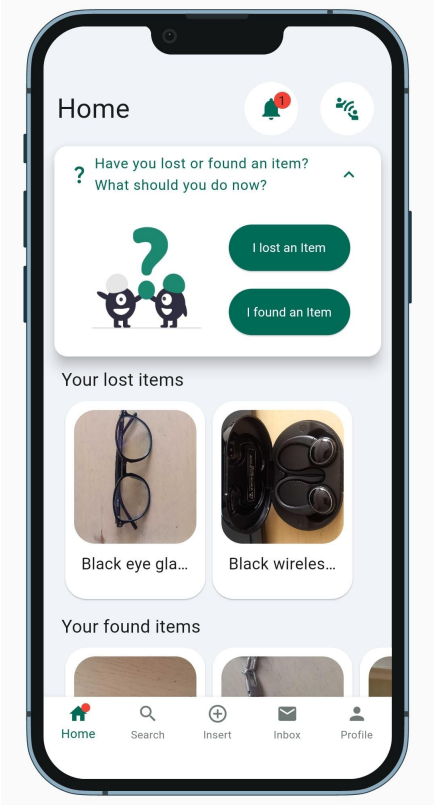

Figure 7: Home page with the info card expanded.

2.2.2 Report

When the user taps on a card in the homepage, he will be directed to the dedicated report page for the item shown on the card. There are two types of viewable pages:

- A page related to the report of an item found by the user (fig. 8 and fig. 9);
- A page related to the report of an item lost by the user (fig. 10 and fig. 11).

These two pages are equivalent in the first section, as both will display the main information of the report, including:

- The photo of the item;
- The title of the report;
- The location where the item was lost or found. By clicking on the underlined text “Open map”, the user has the possibility to see the location on Google Maps (Android) or Maps (iOS). This action can be performed after accepting to exit from the app in a dialog (this is done to prevent wrong clicks);
- The date corresponding to the report submission;
- The category of the item;

- A section that presents reports made by other user that could match with the report presented in the page. If a card is clicked, the user will be redirected to the Report of Generic user page (presented in section 2.2.14) relative to that report.

The page related to the report of an item found by the user also includes:

- The question decided by the user to verify the legitimacy of future claims;
- A section displaying all the claims made by users for that report. By clicking on a claim, the user will be redirected to the claim response page (presented in section 2.2.8). If there are no claims, the user will be informed by a text.

Both pages have a Pop Menu Button, represented by three vertical dots, in the top-right corner of the page. Clicking on it will open a menu that allows the user to:

- Edit the report. The user will be redirected to the "Edit Report" page (presented in section 2.2.3);
- Delete the report (this will delete the report from the app and all chats associated with it). This action can be performed after confirming to delete the report in a dialog, this is done to prevent wrong clicks;
- Mark the report as solved (this will hide the report in the app). This action can be performed after confirming to mark as solved the report in a dialog, this is done to prevent wrong clicks;

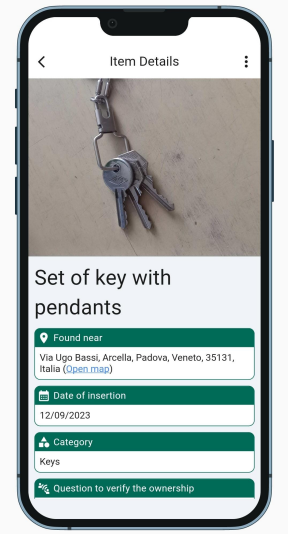

Figure 8: Report page of a found item (upper part).

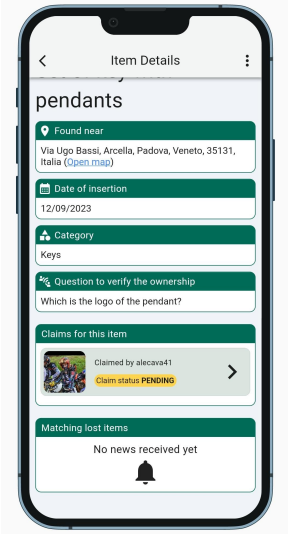

Figure 9: Report page of a found item (bottom part).

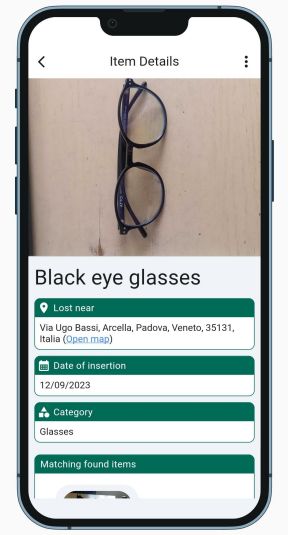

Figure 10: Report page of a lost item (upper part).

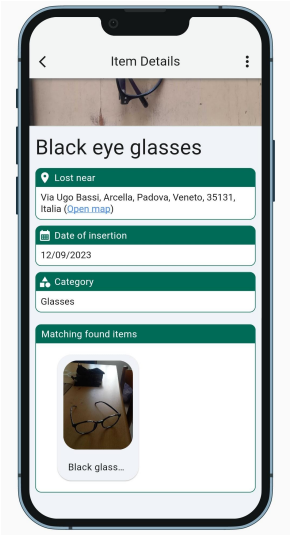

Figure 11: Report page of a lost item (bottom part).

2.2.3 Edit Report

On the Edit Report page (fig. 12), the user has the option to modify the information related to a previously-created report. The user can change all the information except for whether the report has been found or lost (the checkbox that changes this information is disabled). The information fields cannot be left empty. If any fields are left empty, the user will be notified of this with text in the corresponding field that was left empty or with incorrect values. If the user attempts to go back after making changes, they will need to confirm their choice because going back will result in the loss of the changes made (no screen is shown if no changes were made). As soon as the user presses the ”Update” button, the report's information will be updated, and the user will be informed of the success of this operation through a snackbar. The user will then be redirected back to the report page (presented in section 2.2.2).

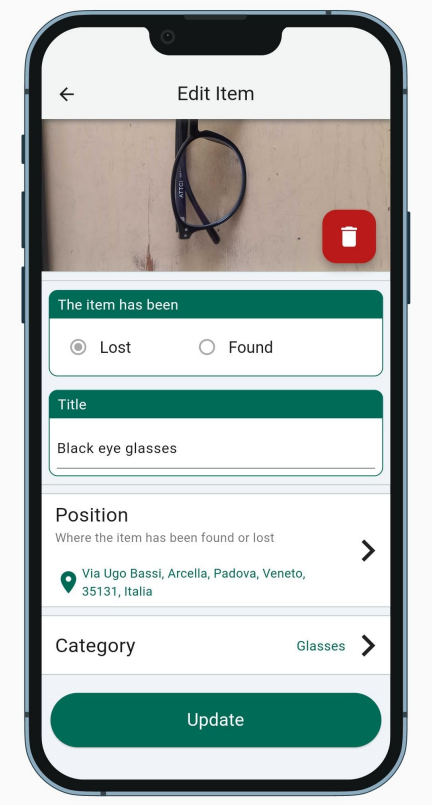

Figure 12: Edit Report page.

2.2.4 News

The news page (fig. 13) displays news that the user has received. This news may include instances where another user has found an item that the user previously reported as lost, or instances where another user has uploaded a report that possibly matches a found item report that the user previously uploaded.

Each news includes:

- The profile image of the user who might have found or lost the item;
- A brief text indicating the item that might have been found or lost;

If the news has not been opened yet (the user has not clicked on it) the background will be green. If the news has already been opened, it will have a white background. If the news page has no news an image and an informative text will be displayed informing the user about this (fig. 14).

By clicking on a news, the user can view the report uploaded by the other user.

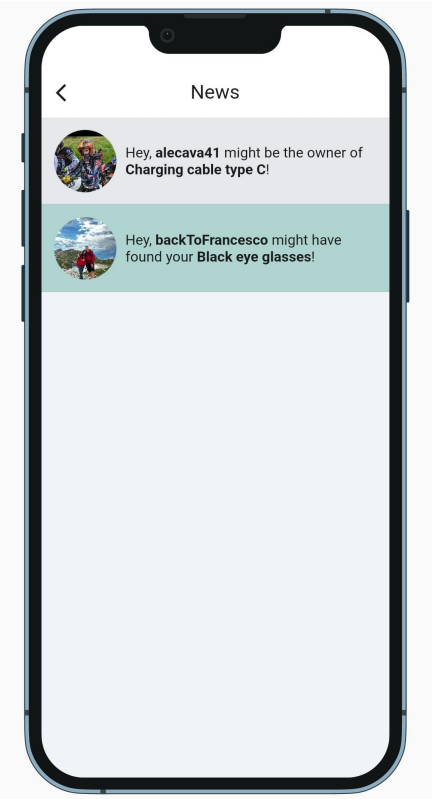

Figure 13: News page.

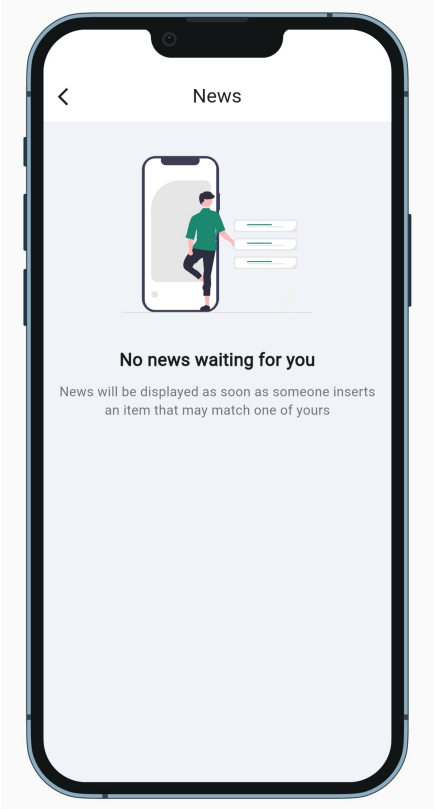

Figure 14: News page with no news received.

2.2.5 Claims

The claims page features two sections:

- The claims received for items reported by the user as found, which other users have claimed (fig. 15).
- The status of the claims made by the user (fig. 16).

These two sections are shown on the same page, and with the help of a TabBarView widget, the user can switch between sections by swiping from right to left to move from the received claims section to the claims status section, and from left to right to do the opposite. The same action can be achieved by tapping on the corresponding name on the tab. In the bottom part of each page, there is a small information section that informs the user about what claims he is viewing. Specifically, it indicates whether he is viewing claims made by himself or claims made by other users on his reports.

Each received claim includes:

- The image and name of the user who made the claim;
- The title of the report for which the user created the claim;

- The status of the claim (approved, pending, rejected).

If the user clicks on a received claim, he will be redirected to the “manage claim” (if the claim status is pending) or “manage claim review” pages presented in section 2.2.8 and section 2.2.9.

Each made claim includes:

- The image of the item for which the claim was made;
- The title of the report for which the user created the claim;
- The status of the claim (approved, pending, rejected).

If the user clicks on a claim he made, he will be directed to the “create claim review” page presented in section 2.2.7, where he can review the response he provided.

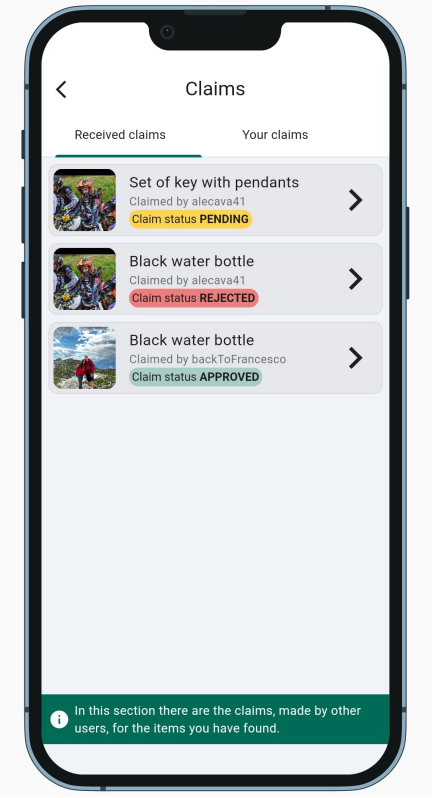

Figure 15: Claims page, received claims section.

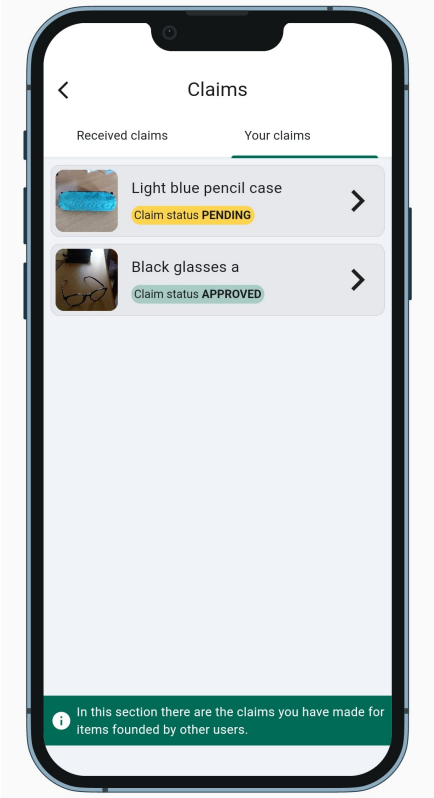

Figure 16: Claims page, claims made by the user section.

2.2.6 Create Claim

On Create Claim page (fig. 17), the user can make a claim for the report by answering a question to verify that he is the rightful and legitimate owner of the item. The page includes a brief summary of the item the user is claiming, which consists of:

- The image of the item;

- The title of the report;
- A ”See details” button that, when clicked, redirects the user to the ”Report of a generic user,” presented in the section section 2.2.14;
- The name and profile picture of the user who create the report (e.g. found the item), along with a ”Send Message” button to chat with him;
- The question that needs to be answered to make the claim;
- A field where the user must input his answer to the question.

Once the answer is inserted, if the user clicks the ”Send” button, he will make the claim, which will be sent to the user who found the item.

At the top of the page, there is an information section for the user. If clicked, it will expand to provide a more detailed description of the page and what the user needs to do.

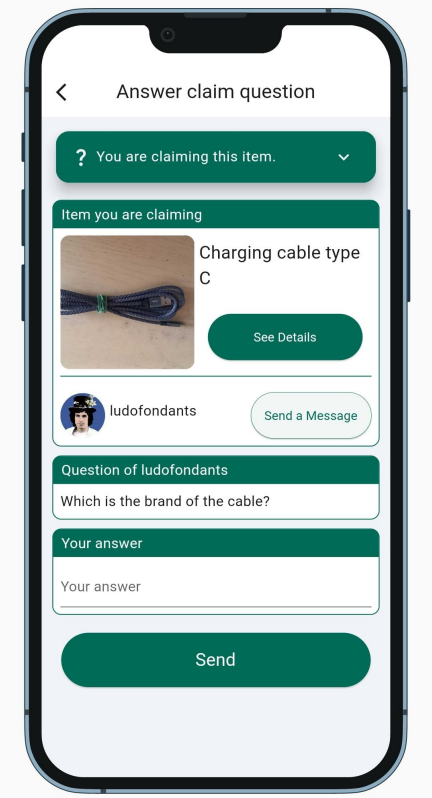

Figure 17: Create Claim page.

2.2.7 Create Claim Review

On Create Claim Review page (fig. 18), the user will be able to review the response given for a claim made on a report. The report’s information is the same as the ”Create Claim” page (presented in section section 2.2.6), with the addition of the user’s response. There is also a section indicating the status of the claim with a brief info text. The status of the claim can be:

- Pending: the user who uploaded the report has not yet confirmed or denied it;
- Approved: the claim has been approved;
- Rejected: the claim has been rejected.

At the top of the page, there is an information section for the user. If clicked, it will expand to provide a more detailed description of the page.

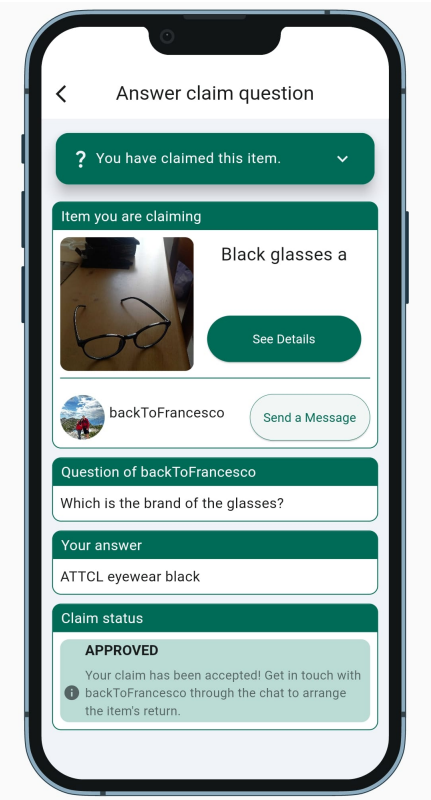

Figure 18: Create Claim review page.

2.2.8 Answer Claim

On Answer Claim page (fig. 19), the user can either accept or reject a claim that has been made by another user based on the response provided to verify the legitimacy of the claim. The page includes a brief summary of the item for which the claim has been made. It provides the following information:

- The image of the item;
- The title of the report;
- A ”See details” button that, when clicked, redirects the user to the report page presented in the section 2.2.14;
- The name and profile picture of the user who made the claim, along with a ”Send Message” button to chat with him;

- The question that needed to be answered to make the claim;
- The response of the user who made the claim;
- Two buttons for accepting or rejecting the claim.

Once the user has accepted the claim, all pending claims for that specific report will be automatically rejected.

At the top of the page, there is an information section for the user. If clicked, it will expand to provide a more detailed description of the page and what happened if the user accepts the claim.

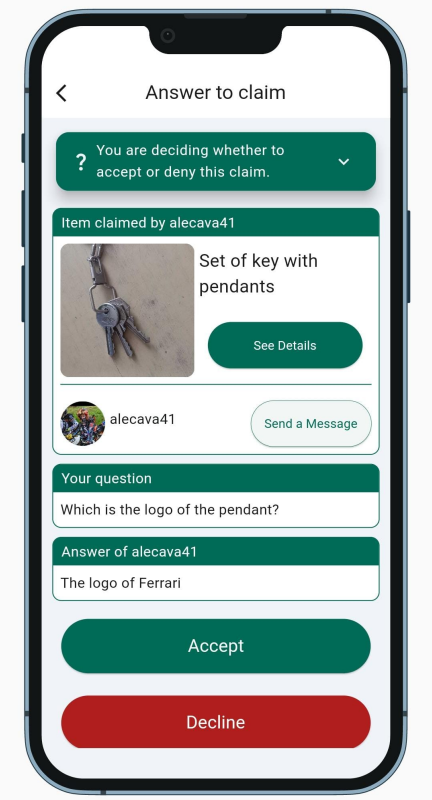

Figure 19: Answer Claim page.

2.2.9 Answer Claim Review

On Answer Claim Review page (fig. 20), the user can view the status of a claim he has received and to which he has responded (either accepting or rejecting it). The information about the item for which the claim was received is the same as that presented on the "Answer Claim" page (presented in section section 2.2.8). There is also a section indicating the status of the claim with a brief info text.

At the top of the page, there is an information section for the user. If clicked, it will expand to provide a more detailed description of the page.

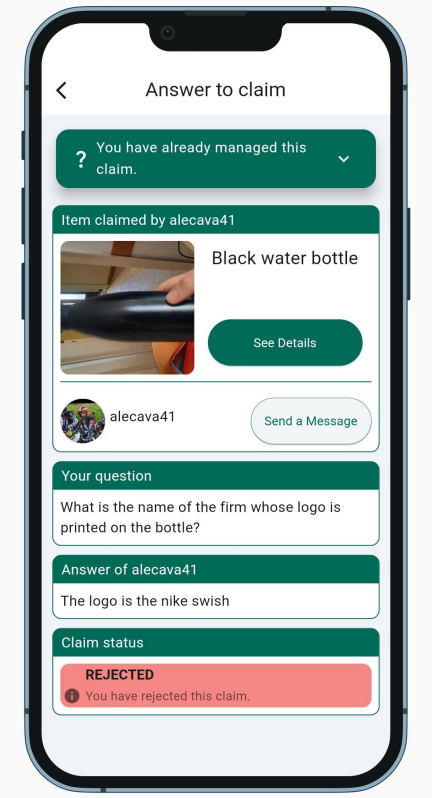

Figure 20: Answer Claim Review page.

2.2.10 Search

Search page (fig. 21) presents the criteria for conducting a search:

- A section that allows selecting the type of reports (e.g found items, lost items, or both). This can be done through checkbox buttons;
- A section that allows the user to select the location where the item was lost or found (presented in section 2.2.11);
- A section that enables the selection of the category to which the searched item belongs to (presented in section 2.2.12);
- A section that permits selecting the date from which the item was lost. This date refers to the report upload date.

For the date entry, a simplified DatePicker widget (fig. 23) with only two wheels for selecting the month and year is utilized. This type of DatePicker was selected to make it quicker for users to choose a date, as compared to the basic DatePicker offered by Flutter, which includes day, month, and year, and has a different visual appearance with a calendar view.

To perform a search, the user must necessarily specify the following fields:

- Type of report (lost, found or both);

- Location;
- Category.

If the user fails to specify these fields, he will be notified with an error text displayed in the respective missing field (fig. 22). For a given field, when the user selects an option, it will be displayed on the right side of the field (fig. 24).

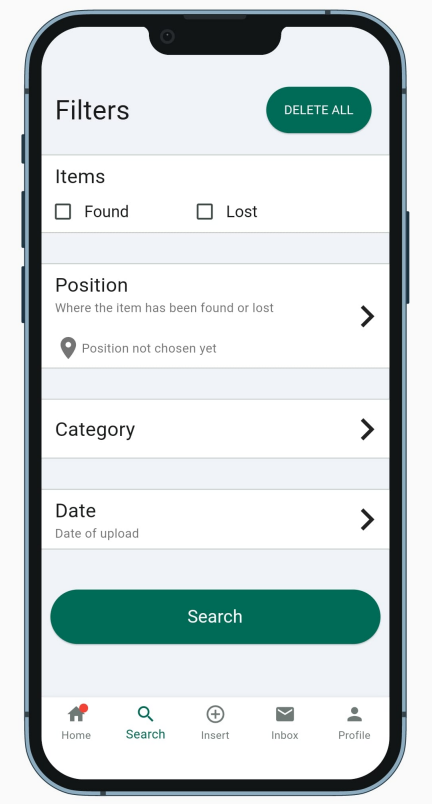

Figure 21: Search page.

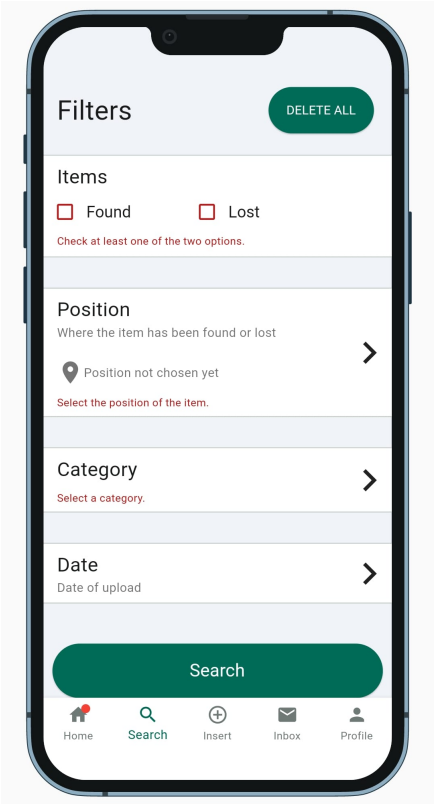

Figure 22: Search page with errors.

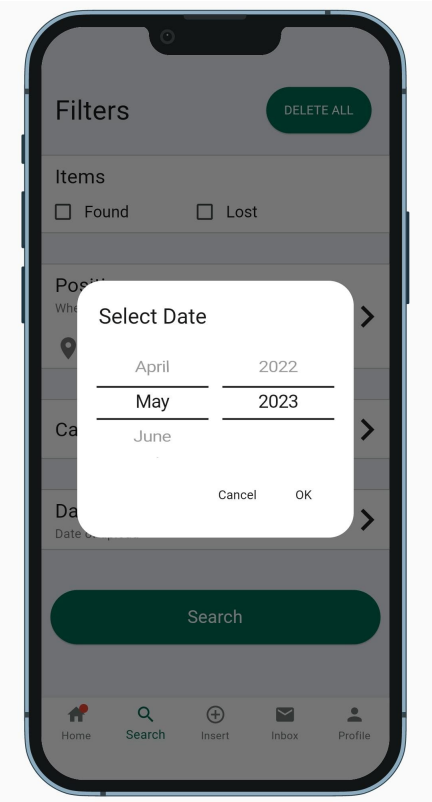

Figure 23: Search page data picker.

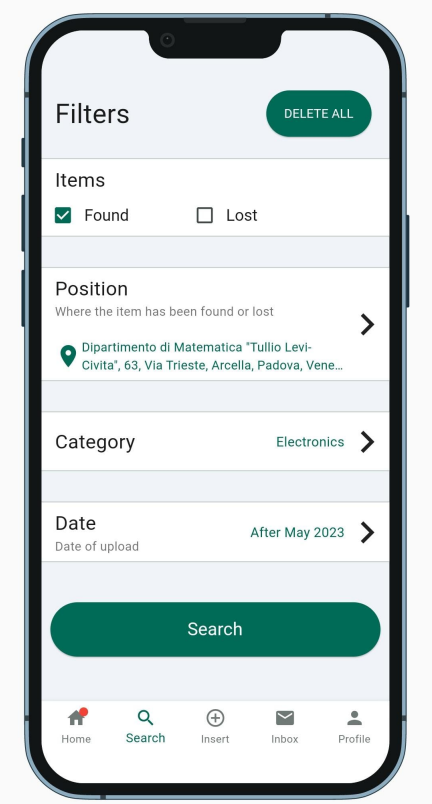

Figure 24: Search page with all fields filled.

2.2.11 Position Selection

The user will be redirected to Position Selection page (fig. 25) when he clicks the position selection button. The page, powered by the FlutterMap widget, displays a map of the world using the APIs of the open-source project OpenStreetMap.

In the center of the screen, there is a green marker that the user should use as a pointer to select a specific location. If the user clicks the "Use my Current Position" button, the pointer will be placed at the user's current location. The user will be localized using their smartphone's location, but to use this feature, he must first accept the application's request for location services access. This permission is not asked as soon as the screen is created, since the user could place the marker wherever he wants. In this way we make sure to ask for the user position only when really needed, avoiding to stress the user.

Dialogs are provided in the following cases:

- To inform the user that he needs to grant location access to use their current position if he has denied permission (permanently or just once);
- To inform the user that the location service is off and it needs to be activated;
- To inform the user that there's no internet connection.

The user can also search for a specific location by entering the name of a city or an address (fig. 26). If the address or city exists, the marker will be placed on the point of the map searched by the user. If the name of the place entered by the user is not found, a snack bar will inform the user of this. If the user presses the search button without entering any place, an error message will be displayed next to the empty input field. To prevent the user from entering a location and mistakenly thinking that by clicking the 'Select this position' button, he is choosing the location he typed, the 'Select this position' button is temporarily replaced with the 'Search' button while the user is still entering the location. To see the 'Select this position' button again, the user can either press the 'Search' button, click the 'Use my current location' button, or close the search screen by clicking on the button with a downward arrow that appears over the text field only when the user is typing the name of a position.

As soon as the user clicks the "Choose this Position" button, the position indicated by the marker in the center of the screen will be saved, and the user will be redirected back to the page where he clicked the position selection button. Once a position is selected, the address (if any), corresponding the selected coordinates, is displayed. That allows the user to assert the correct selection of the position.

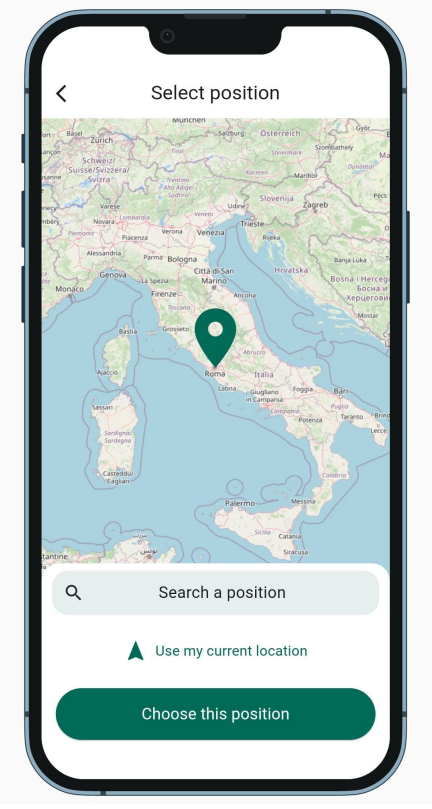

Figure 25: Position Selection page.

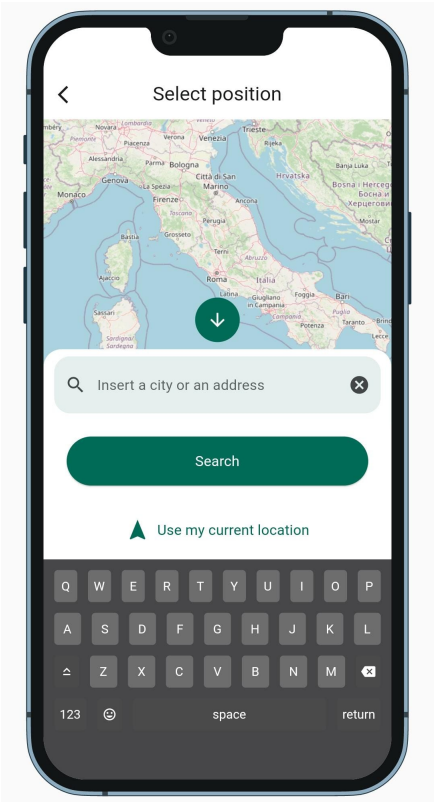

Figure 26: Position Selection page when the user search for a place.

2.2.12 Category selection

The user will be redirected to the Category selection page (fig. 27) when he clicks on the category selection button. This page uses a scrollable view to present a column list of custom buttons, each button has:

- The name of the category that represent;
- The description of the category, including some examples about items that can fit in that category;
- An icon representing the category.

The first button ("All") includes all categories. When a category is clicked, it will be saved, and the user will be redirected back to the page where he clicked the category selection button. To help the user to remember the category selected, its name will be displayed on the corresponding box.

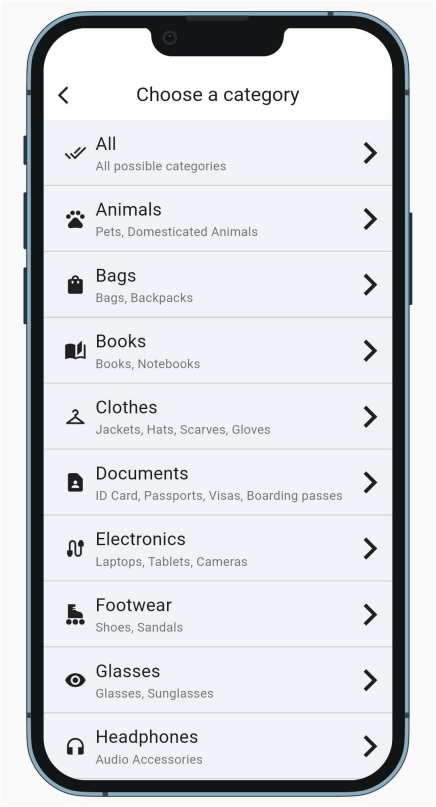

Figure 27: Category selection page.

2.2.13 Search Results

Once the user performs a search, he will see the results page (fig. 28). The results are presented in cards, similar to those shown on the home page, but these also include:

- The date of upload;

- The distance from the position specified in the search page;

Additionally, each card has a label indicating whether it's a report for a lost or found item. The cards are arranged in two columns and can be vertically scrolled.

By clicking on a card, the user will be redirected to the page that presents the report featured on the Card, this page is presented in section 2.2.14.

The user can also modify the search parameters. By pressing the "Filters" button, he will be redirected to a page equal to the initial search page.

The user can arrange the display order of the results as he prefers. The available sorting criteria are as follows:

- Alphabetical order from A to Z or reverse;
- Order based on insertion date, from oldest to newest or vice versa;
- Order based on distance from the selected point, from nearest to farthest or vice versa.

Upon clicking the sorting button, a Modal Bottom Sheet widget will appear (fig. 29), presenting the various sorting methods. The currently selected method is highlighted in green. By clicking on the desired sorting method, the results will be arranged accordingly. Only one sorting criterion can be used at a time.

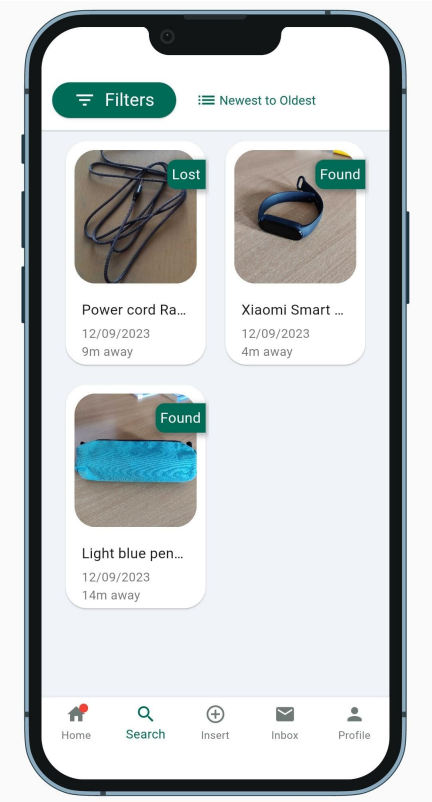

Figure 28: Search results page.

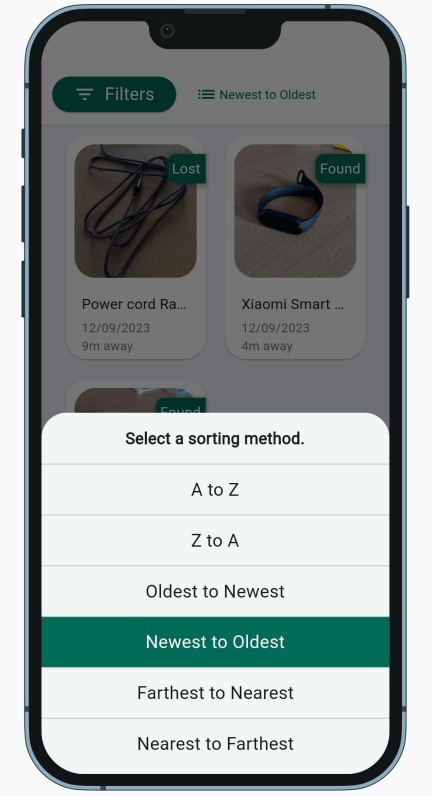

Figure 29: Modal Bottom Sheet where the user can choose the sorting method.

2.2.14 Report of Generic User

When the user taps on a card in the results page, he will be directed to the dedicated page for the item shown on the card. There are two types of viewable pages:

- A page related to the report of an item found by a user (fig. 30);
- A page related to the report of an item lost by a user (fig. 31).

These two pages are equivalent in the first section, as both will display the main information of the report, including:

- The photo of the item;
- The title of the report.
- The location where the item was lost or found, by clicking on the underlined text “Open map”, the user has the possibility to see the location on Google Maps (Android) or Maps (iOS). This action can be performed after accepting to exit from the app in a dialog, in order to prevent wrong clicks;
- The date the report was submitted.
- The category of the item.

The page related to the report of an item found by a user also includes:

- The question decided by the user who made the report to verify the legitimacy;
- A button that allows the user to claim the object, if clicked the user will be redirected to the “Create Claim” page presented in section 2.2.6. If the user has already claimed the item, he will see the status of the claim;

Lastly, both pages include a section displaying the user’s photo and name who created that report, along with a ”Send Message” button. Clicking this button will redirect the user to a chat with the user who created the report, the chat page is presented in section 2.2.17.

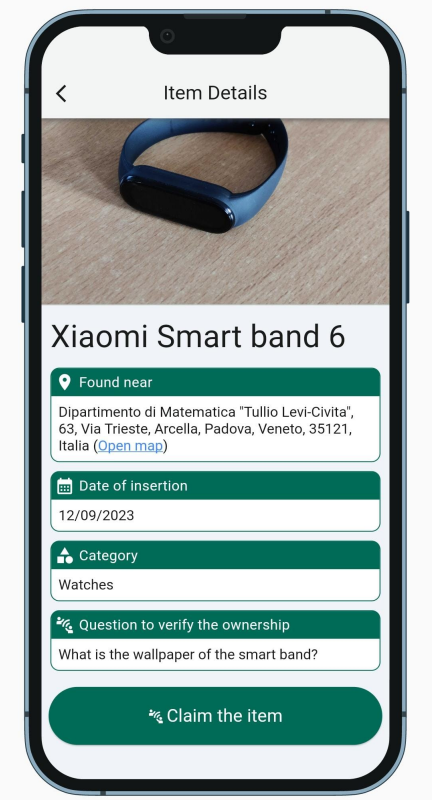

Figure 30: Report of Generic User page for a found item.

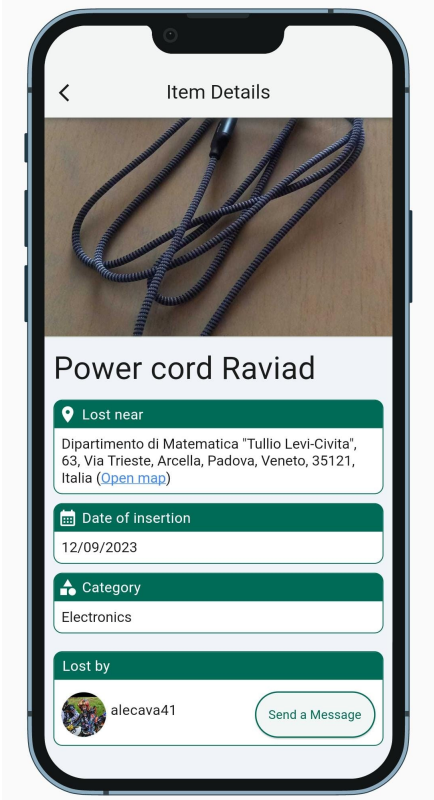

Figure 31: Report of Generic User page for a lost item.

2.2.15 Insert Report

On this page, the user has the possibility to create a report. The first section allows the user to upload a photo of the item. By clicking on the ”Upload a Photo” button, the user will see a dialog asking the media from which to upload the image. There are two options presented with buttons:

- Upload an existing photo from the gallery;
- Use the camera to take a photo and then upload it;

Using the camera requires permissions, and if these permissions are denied permanently, the user will be advised through a dialog to enable the camera permissions from their phone’s settings.

The user must specify whether the report refers to a lost (fig. 32) or found item (fig. 33) using two radio buttons. Radio buttons are used instead of checkboxes because a report for an item can only refer to one item, and that item can either be lost or found, not both at the same time. If the user selects ”Found” an additional field will be displayed where he must enter the question that will be used to verify the legitimacy of future claims. The user must enter:

- The title of the claim;
- The location, as presented in section 2.2.11;

- The category, as presented in section 2.2.12;

The title, location, category, and, if the item is found, the question to verify the legitimacy of future claims are mandatory fields. If the user tries to click the ”Create” button without completing all of them, he will be notified with an error text in the empty fields.

If the user has entered some information, such as the title, and tries to go back, he will be notified through a dialog that going back will result in the deletion of the entered information since he has not completed the report creation by entering all the required information and then clicking the ”Create” button. In the dialog, he can confirm whether he wants to go back or stay on the page.

Once all fields are filled correctly, by clicking the ”Create” button, the report will be uploaded to the server. The user will be notified of the success of the operation through a snackbar and he will be automatically redirected to the home page. If the item upload fails, the user will also be notified via a snackbar.

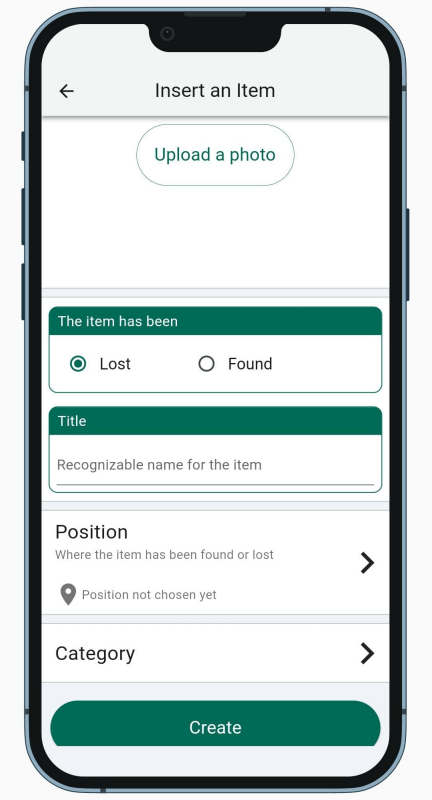

Figure 32: Insert Report page for a lost item.

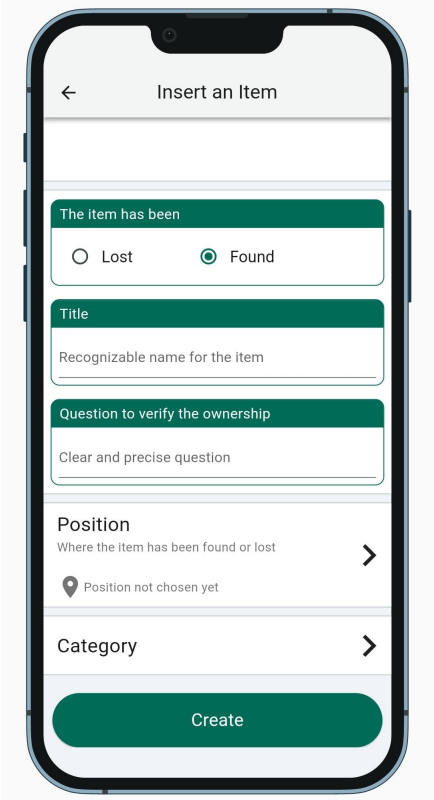

Figure 33: Insert Report page for a found item.

2.2.16 Inbox

On Inbox page (fig. 34), all current user chats are displayed vertically, one below the other, using the ListView widget. Each chat is characterized by the user to whom messages are sent and the

report for which the current user is chatting. Therefore, it's possible to have multiple chats with the same user if the current user is chatting about different reports. Each chat is presented with:

- The image and name of the user the current user is chatting with;
- The last message sent;
- The image and name of the report the current user is chatting about.

A chat that is colored green indicates that there are unread messages. By clicking on a chat, the current user is redirected to the chat page with a specific user for a specific report. This page is presented in section 2.2.17. If no active chats are present, an image and a text will explain this to the user (fig. 35).

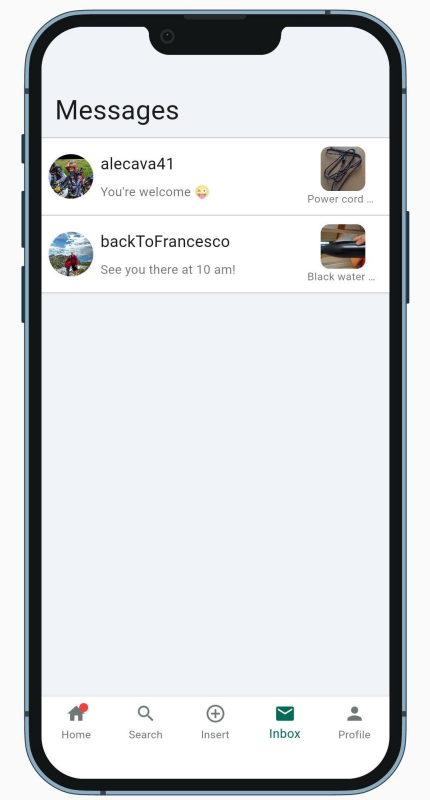

Figure 34: Inbox page.

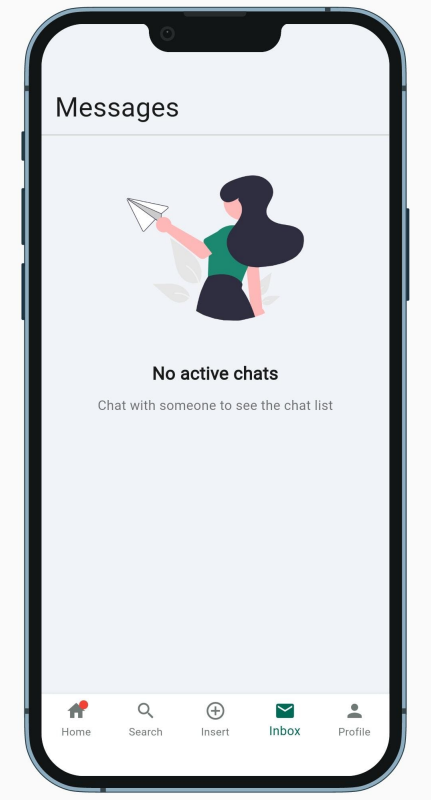

Figure 35: Inbox page with no active chats.

2.2.17 Chat

Chat page consists mostly of elements provided by the Flyer Chat library, which allows to create the user interface of a typical messaging chat, including messages and the message input box. Below the AppBar, which contains the name of the user with which the current user is chatting, the report of the chat is indicated.

Depending on the status of the claim, the current user will see three different types of cards:

- If the current user has claimed the report, he will see:

  - The image of the item;
  - The title of the report;
  - The status of the claim he has made;

  If the card is clicked, the user will be redirected to the "Create Claim Review" page, presented in section 2.2.7;

- If the current user has received a claim for the report (fig. 36), he will see:
  - The image of the item;
  - The title of the report;
  - The status of the claim, whether it has been rejected, accepted, or is awaiting a response.

  If the card is clicked, the user will be redirected to the "Answer Claim Review" page, presented in section 2.2.9.

- If one of the following conditions apply:
  - The current user has not claimed the item;
  - The current user created the report and the user he is chatting with has not make the claim yet;
  - The current user is chatting for a lost item reported by the user who is chatting with (fig. 37).

  The current user will see:
  - The image of the item;
  - The title of the report;

  If the report was created by the user, clicking the section will redirect the user to the "Report" page, presented in section 2.2.2. On the other case, clicking the section will redirect the user to the "Report Generic User" page, presented in section 2.2.14.

The card of the report the current user is chatting about is always visible to provide easy access to its related pages for the current user, eliminating the need to navigate from one page to another.

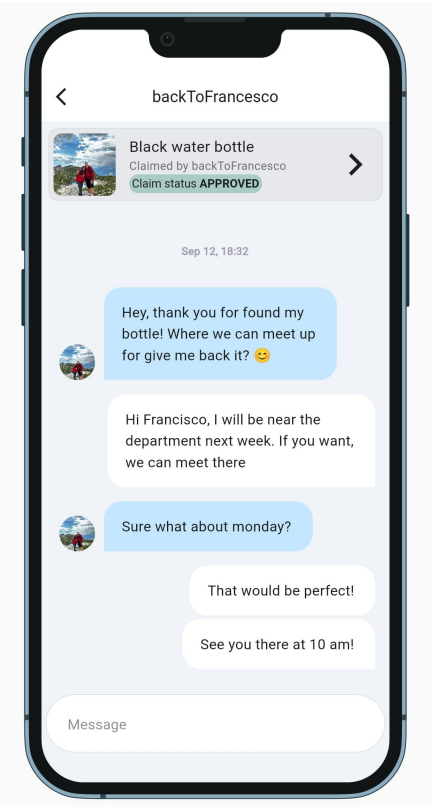

Figure 36: Chat page for a claim received.

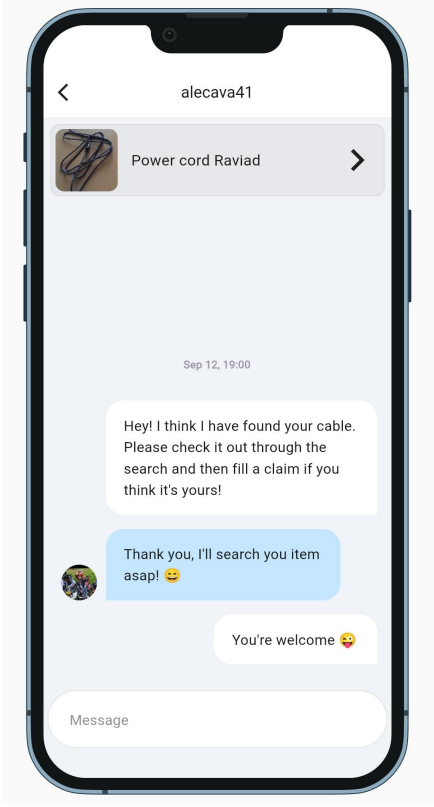

Figure 37: Chat page for a lost item reported by another user.

2.2.18 User Settings

User Settings page (fig. 38) displays the user's profile information, such as their profile picture, name and the email address.

By clicking on the profile picture, the user can edit it by uploading a new one. This option is presented to the user with an icon that represents a camera (commonly associated with editing profile pictures, as seen in apps like WhatsApp), located in the bottom right corner of the image.

Additionally, on this page, there are 5 buttons that allow the user to:

- Access "Tutorial" page, presented in section 2.2.20;
- access "Change Password" page, presented in section 2.2.19;
- Change the language of the app through a BottomModal widget, where the user can choose between Italian or English;
- Change the theme of the app through a BottomModal widget, the theme can be light or dark. The default theme is set using the system current theme;

- Log out from his account. To confirm this action, the user will be prompted to confirm his intent to log out through a Dialog. This confirmation step is in place to prevent unintended logouts caused by accidental touches (fig. 39).

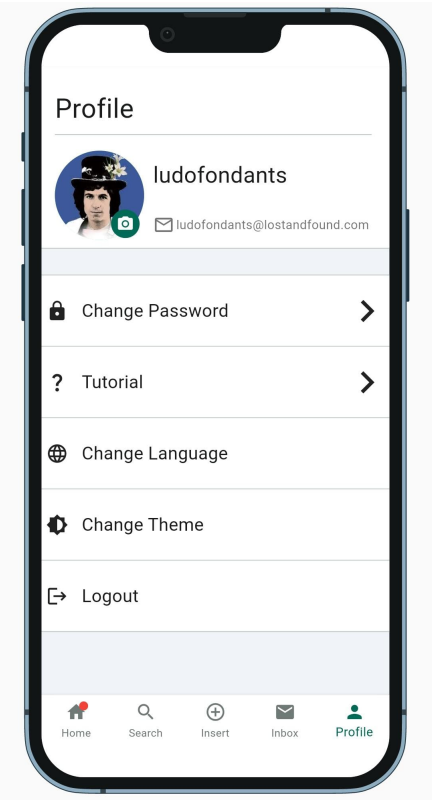

Figure 38: User Settings page.

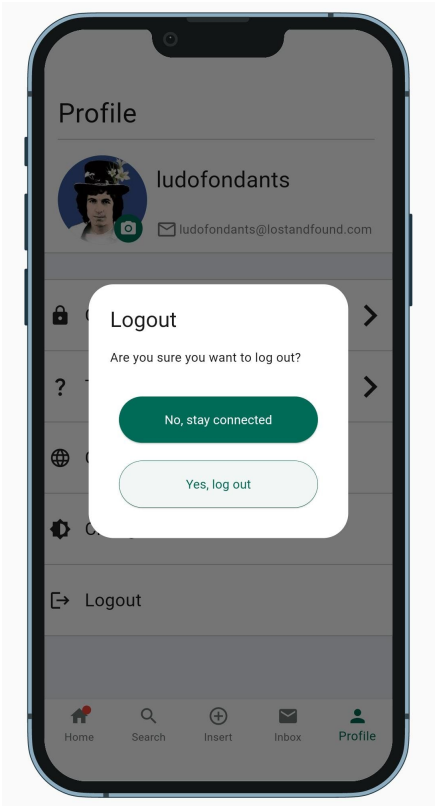

Figure 39: User Settings page logout dialog confirmation.

2.2.19 Change Password

On Change Password page (fig. 40), the user has the option to update his account password. To do so, he will need to input his current password, the new password, and confirm the new password.

If the current password is entered incorrectly, or the new password does not meet the minimum security criteria, or the passwords do not match, an error message will be displayed to the user in the corresponding field as soon as they click the ”Change Password” button for the first time.

If all criteria are met, when the user clicks the ”change password” button, a snackbar will notify him if the operation was successful. The user will then be redirected to the profile page.

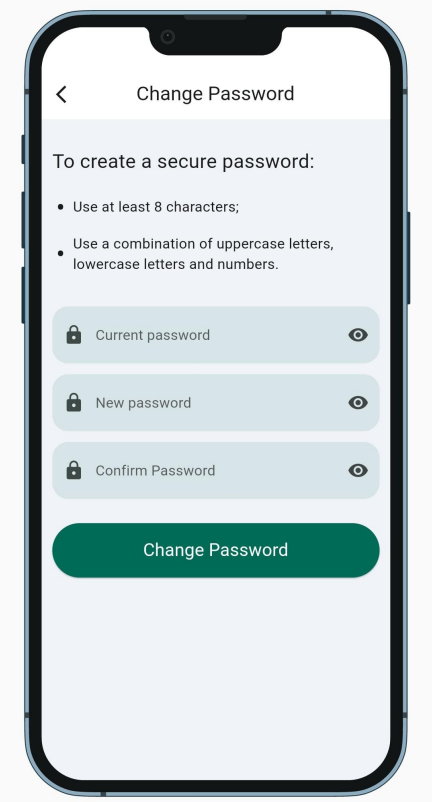

Figure 40: Change Password page.

2.2.20 Tutorial

In Tutorial page, there are two different sections presented via a PageView. The first section contains an in-depth tutorial that explains to the user the various steps he needs to do when he loses an item (fig. 41). The second tutorial, on the other hand, concerns the steps he needs to take after finding an object.

To facilitate navigation between sections, a lateral swipe is used. To progress from one step to the next in the tutorials the user has to do a scroll from bottom to top. To indicate this difference from the tutorial on the Initial page, the progress indicator is oriented vertically instead of horizontally. Additionally, there is a button that allows the user to advance through the steps without the need to swipe. When the user arrives in the last step of the tutorials, the button to that allows the user to advance will be disabled (fig. 42).

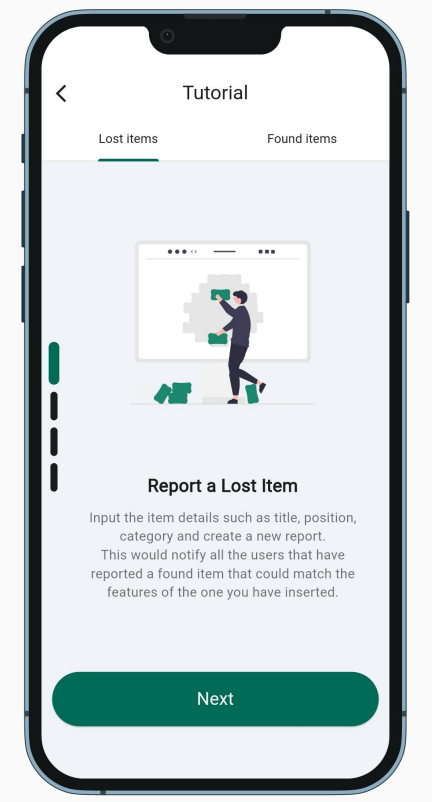

Figure 41: Tutorial page.

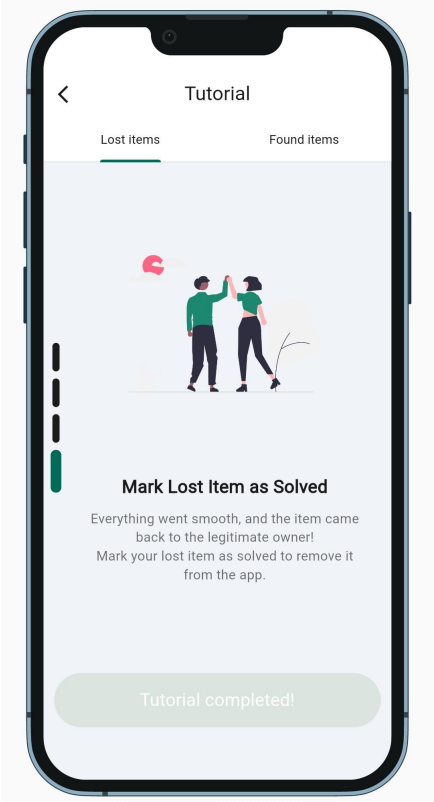

Figure 42: Tutorial page with tutorial completed.

3 General Design Choices

3.1 Test

The app's responsiveness has been tested using the Device Preview Library. The app has been tested on various virtual devices and preset screen sizes (the measure of each device is presented in logical pixels):

- Virtual Devices:
  - iPhone SE, screen size 4" (375.0x667.0 @2.0);
  - iPhone 12 Mini, screen size 5.42" (375.0x812.0 @2.0);
  - iPhone 13, screen size 6.06" (390.0x844.0 @3.0);
  - iPhone 12 Pro Max, screen size 6.07" (428.0x926.0 @3.0);
- Preset Screen Sizes (Android):
  - Android small (360.0x640.0 @2.0);
  - Android medium (412.0x732.0 @2.0);
  - Android big (480.0x853.0 @2.0).

Furthermore the app has been tested in the following physical devices:

- iPhone XR, screen size 6.1", iOS 16.6;
- OnePlus Nord, screen size 6.44", Android 12;
- Samsung Galaxy A41, screen size 6.1", Android 12.

3.1.1 Page Content Responsiveness

The widgets that were experiencing issues on various screen dimensions had dynamic sizes, adaptable to the screen size in which they are displayed. To make this possible, the Sizer Library was used.

In particular cases, static sizes were set that change based on the screen size. To determine the screen size, a query is performed that determines the size based on the screen's height. Depending on the screen dimensions, the screen is classified into 3 categories:

- Large: screen's height greater than 800 logical pixels;
- Medium: screen's height between 700 and 800 logical pixels;
- Small: screen's height smaller than 700 logical pixels.

To determine the right size, only the screen's height is used because the static dimensions we want to obtain are primarily used to determine the height of the widgets.

The way the widgets are positioned and how they respond to potential resizing allows the user to easily reach almost all clickable widgets with his thumb.

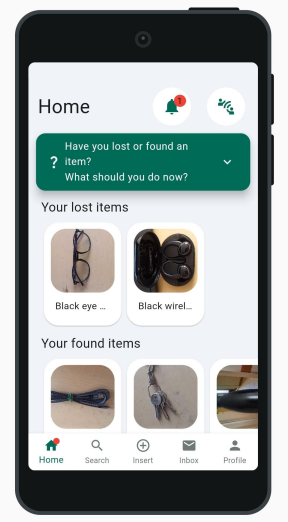

Figure 43: Home page on small Android device (360.0x640.0 @2.0).

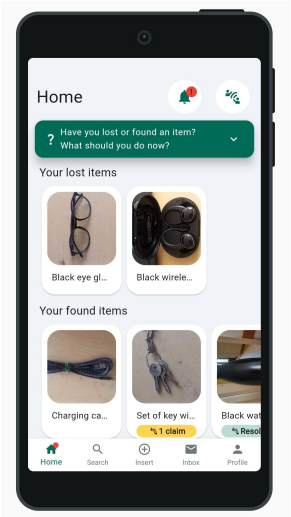

Figure 44: Home page on medium Android device (412.0x732.0 @2.0).

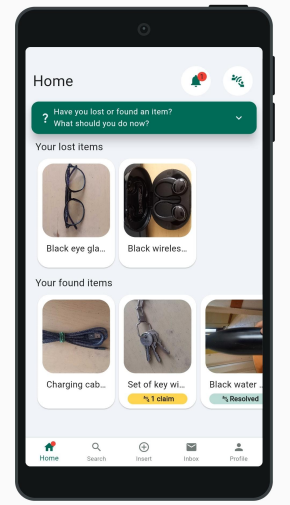

Figure 45: Home page on big Android device (480.0x853.0 @2.0)

3.1.2 Text and Button Sizes

Even though the widgets are responsive, all clickable widgets remain relatively large in size. This is done to ensure that users can click on them without any issues.

Using Flutter Inspector, it has been verified that all buttons, using screen size (360.0x640.0 @2), exceed the recommended minimum size of 44 pixels.

The text does not resize based on screen size, it is static. It has been verified that the standard sizes adapt to all screen sizes. However, there are no overflow issues when the text doesn't fit within the designated row; instead, it adds ellipsis (e.g., "text of pro..."). Additionally, if the

information presented in the text is essential (e.g., error messages), the text wraps to the next line. If the text size is modified through the phone settings, the text changes its size. It has been verified using Flutter Device Preview that the application remains usable even with a text scaling factor of 2.0 (doubling the text size), although it may have some overflow issues. The text on the cards in the home page has been made static because they had responsiveness problems that were fixed by setting the text size statically. Users can rely on the image to distinguish the item, and if he needs to view additional information, he can click on the card and read it on the Report page at the desired size.

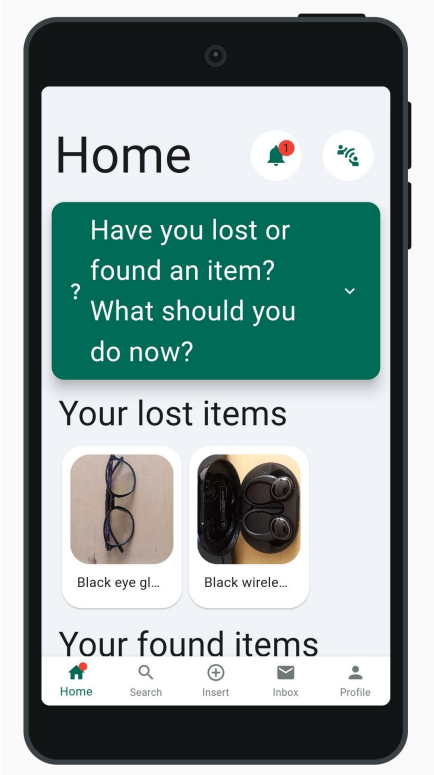

Figure 46: Home page with text scaling factor of 2.0 in a medium Android device.

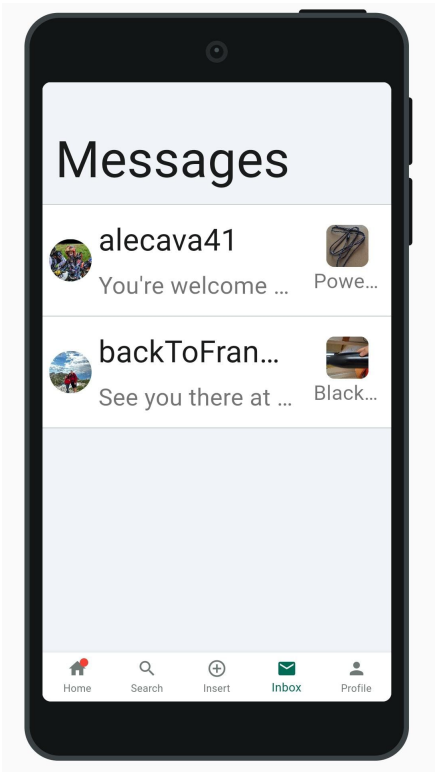

Figure 47: Inbox page with text scaling factor of 2.0 in a medium Android device.

3.1.3 Device Orientation

The application can only be viewed in portrait mode. If the user attempts to rotate their device, the content will not be displayed in landscape mode. This limitation is because the app has been designed exclusively for portrait mode.

3.2 Generic Error Page

An error during the loading of of a page is communicated to the user through an error page, featuring an image and a brief text that informs the user about the error (fig. 48). A button is provided to allow the user to retry the action.

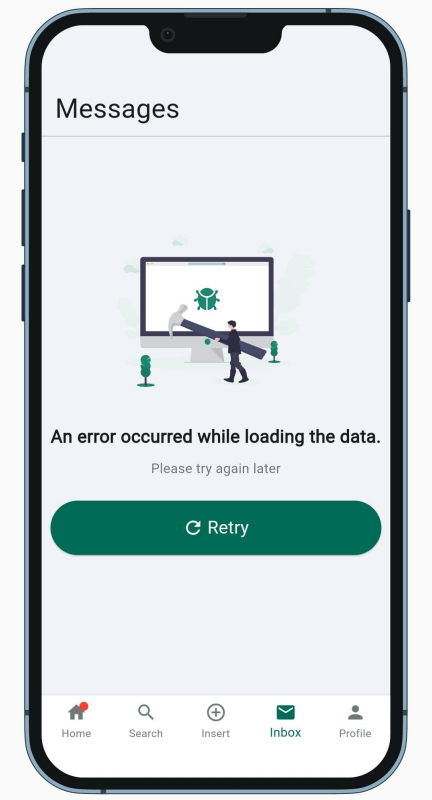

Figure 48: Generic Error page.

3.3 Empty Content

In case the following pages: ”News”, ”Claims” (each section, fig. 50), ”Search Result” and ”Inbox” do not contain any content (for example, when a search produces no results fig. 49), a brief informative text and an image specific to the page will be displayed to alert the user of this fact.

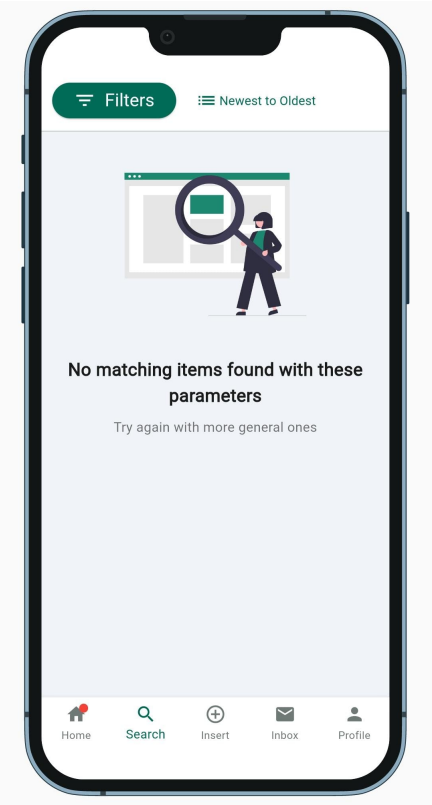

Figure 49: Search result page with no result.

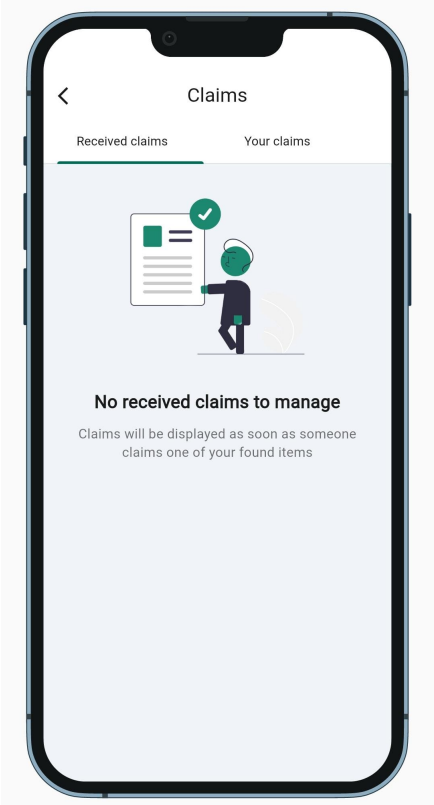

Figure 50: Received claims section with no claims received.

4 System Architecture

Here is a brief overview about the technical structure of *Lost and Found*. The system consists of a front-end mobile application (as described above) and a back-end server which acts as a bridge between the underlined database and the mobile app.

4.1 Front-end Mobile App

4.1.1 Framework

We chose Flutter as a framework for the cross-platform development of the mobile application.

Flutter is an open-source UI software development toolkit created by Google, designed to build natively compiled applications for mobile, web, and desktop using a single codebase. It focuses on high-quality user interfaces, rapid development, and native performance. Its rich set of features and strong community support make it a compelling choice for developers looking to create modern, engaging apps across multiple platforms.

Here is a list of the main advantages of Flutter:

- Single codebase: it allows to write a single codebase that can be used to create apps for multiple platforms such iOS and Android, but also web apps and desktop apps. Therefore developers can save a lot of time;
- Expressive UI: it has an expressive and highly customizable user interface. It provides a wide range of composable widgets and tools for creating visually appealing, complex and interactive designs. It also supports adaptive design, which means you can create apps that adapt to different screen sizes and orientations automatically;
- Fast Development: Flutter's "hot reload" feature enables real-time code changes and updates the app's interface instantly without the need for a full app restart. This greatly speeds up the development and testing process;
- Native Performance: Flutter compiles to native ARM code, providing near-native performance on both iOS and Android. It doesn't rely on web views or other intermediate layers, ensuring smooth animations and responsive interfaces;
- Community and Support: Flutter has a rapidly growing community of developers, which means there are plenty of resources, tutorials, and forums available for help and support.

4.1.2 App Architecture

In the development of our application, we tried to adhere to the Clean Architecture principles. *Clean Architecture* is a software design paradigm renowned for its emphasis on modularity, maintainability, and testability, all of which contribute to the creation of a resilient and adaptable application. Following those principles, led us to benefit from the following advantages:

- Maintainability: the partitioning of components simplifies the process of effecting changes or updates to the application without causing disruptions in unrelated sections;
- Testability: The Clean Architecture methodology strongly encourages the implementation of unit tests, facilitating the validation of the correctness of individual components in isolation;

- Flexibility: Clean Architecture promotes flexibility by enabling us to select the most suitable technologies and frameworks for each architectural layer. It permits the replacement of components without compromising the integrity of the entire system.
- Scalability: as the app evolves, Clean Architecture allows us to seamlessly scale it by introducing new features or modules without triggering cascading changes across the entire codebase.

4.1.2.1 Data Layer The data layer of our mobile Flutter app serves as the backbone for managing data interactions. Its primary role is to facilitate seamless communication between the app and external data sources, such as the back-end and other systems (like OpenStreetMap, IP-api, Firebase, etc.). To accomplish this, we have implemented Retrofit as our http client, a robust and efficient networking library for Flutter. With Retrofit in place, our data layer efficiently handles tasks like making network requests, handling responses, and converting data into usable formats within our app. This integration ensures that data retrieval and manipulation processes are both reliable and streamlined.

4.1.2.2 Domain Layer The domain layer of our mobile Flutter app serves as the heart of our application's business logic. Its primary function is to encapsulate the core functionalities and rules that govern our app's behavior. We implemented here app-specific logic, domain entities, and business rules. It acts as the bridge between the data layer and the presentation layer, orchestrating data manipulation and transformation while maintaining separation from platform-specific code.

4.1.2.3 Presentation Layer The presentation layer is responsible to display the data coming from the domain layer and to handle user interactions and communication. We used BLoC (Business Logic Component) for data processing and state management.

Presentation layer widgets are responsible for rendering UI components and subscribing to BLoC streams for updates. They capture user inputs (keyboard, tap, swipes, ...) and transmit them to the BLoC. BLoC, on the other hand, holds the application's business logic, processes data, and manages state. It emits state changes through streams, which presentation layer's widgets listen to, triggering UI updates accordingly.

By utilizing the BLoC pattern within our presentation layer, we've achieved a clean separation of concerns. This separation not only enhances maintainability but also simplifies the process of testing both the UI and business logic independently.

4.2 Back-end Services

4.2.1 Web Server

The web server has been written using the Go programming language, employing the Gin Gonic framework for its HTTP server capabilities. To uphold code maintainability and system adaptability, we have implemented the Clean Architecture pattern as the underlying architectural framework of our server. This approach significantly streamlines development, fostering agility and facilitating future expansions (as for the front-end app). Our web server predominantly functions as a RESTful API gateway, in order to enable simple communications between the front-end mobile app and back-end database.

In terms of containerized deployment, our architecture comprises two distinct containers:

- Web Server Container: this container hosts the core web server component, orchestrating the reception and routing of incoming HTTP requests. It proficiently maps these requests to the relevant endpoints. It serves as the nucleus of our server-side logic. The container engages communications with a separate container where the database system lies;
- Text Translation Container: the second container specializes in text localization and translation services, promoting adaptability for users from diverse linguistic backgrounds. It is useful to translate text coming from the database in different languages.

4.2.2 Database

The back-end system relies on the MySQL Database Management System for efficient data storage and management. MySQL is a widely-used open-source relational database that offers robust performance, scalability, and reliability, making it an ideal choice for our data storage needs.

Below is the database schema of the application. This schema outlines the structure and relationships of the tables that will store our data. The database contains the following tables:

- casbin_rule: information needed by casbin, a flexible and rule-based access control mechanism for enforcing permissions in applications;
- users: information about the users subscribed to Lost and Found, the main fields are: username, email, password, locale (language in use) and token (to send push notifications);
- items: information about the items inserted through the app, the main fields are: title, position, type (lost or found), safe question, creator and category;
- claims: information about the claims made by the users, the main fields are: answer, status of the claim, creator and item claimed;
- news: information about the possible matches (lost item corresponds to a found item and vice versa) between items, the main fields are two items that may match;
- categories: categories in which an item can fit into.

4.2.3 Firebase

Push notifications and real-time chat features are two pivotal components of user interaction and retention. Firebase services are used to seamlessly integrate these functionalities into our application.

4.2.3.1 Push Notifications Firebase Cloud Messaging (FCM) is a robust and scalable service that allows us to send push notifications to our users' devices. This functionality is essential for keeping users informed about updates, messages, and other relevant events in real-time. Users will be notified in the following cases:

- New matches: when a user inserts a new item through the app, the server will look for possible matches. It will send a push notification to all the users who have inserted an item that may match the one just created;
- New received claims: when a user fills a claim for a found item, the user who inserted that item will be notified. In this way there may be less time span between the claim's creation and the claim's management;

- Update on a filled claim: when a user evaluates a claim, the user who filled the claim will be notified;
- New chat messages: every time a user sends a message through the chat, the other user will receive a push notification.

Push notifications will be received only when the front-end app is in background or terminated state. If the app is in use and a push notification arrives, then the app will automatically manage it by adding badges (as described above) to the specific sections (based on the type of the notification).

4.2.3.2 Chat As mentioned above, Flyer is a real-time chat SDK that seamlessly integrates with Firebase to provide our users with a reliable and feature-rich chat experience. We used both the UI SDK and the back-end SDK to integrate this technology with our system.

4.2.4 Deployment

The deployment of the back-end containers has been accomplished on Amazon Web Services (AWS), making the most of the available 12-months free trial for EC2 (Elastic Compute Cloud) instances. This approach aligns with cost-effective infrastructure management, prioritizing efficiency and scalability in container hosting.

It's important to note that these instances are configured with limited resources, including CPU and RAM. Consequently, at the moment the system may not be equipped to handle significant user workloads that require substantial computing power and memory resources.

5 Pre-existing Configuration and Testing

The application has been pre-loaded with various data, including user profiles, reports, claims, and conversations. This approach has been adopted due to the potential cost implications of replicating an extensive dataset which can cover all the possible cases.

The already-existing users are:

Username	Password
alecava41	Alecava41
backToFrancesco	Back2francesco
ludofondants	Ludofondants1

Within the application, you can find pre-loaded items, conveniently located near Torre Archimede.

6 Flutter Final Considerations

By using Flutter, we had the opportunity to choose from a wide range of libraries created and made available by the Flutter community. This significantly expedited our work, as we didn't have to create custom widgets for every feature (e.g., chat). Instead, we could integrate pre-built widgets that work very well and, in most cases, offer a high level of customization.

It's worth mentioning Flutter's 'hot reload' feature, which saved us a lot of time during the graphic design phase. With hot reload, whenever we needed to update the application after making a change, only the modifications we made were loaded into the app. This eliminated the need to build and restart the application every time.

Unfortunately, Flutter doesn't provide an easy way to detect some system-level operations, such as when the back button on an Android phone is pressed with the keyboard open, resulting in the keyboard closing. Furthermore, certain widget functionalities behave differently depending on the platform. To address these issues, we had to implement specific workarounds in our code based on the device on which the application is running.