****

COMPUTER SCIENCE AND ENGINEERING

DESIGN AND IMPLEMENTATION OF MOBILE APPLICATIONS (DIMA)

Course 2021-2022



**Authors**

José Ignacio Daguerre Garrido

Francisco Javier Muñoz Ruiz

**Index**

[**1. Introduction**](#_heading=h.1fob9te) **3**

[1.1 Application proposal](#_heading=h.1zvwx4m2fhem) 3

[1.2 Document Organization](#_heading=h.1t3h5sf) 3

[**2. Workspace and Tools**](#_heading=h.pz74nilsni6p) **4**

[**3. Structure and implementation**](#_heading=h.3tbugp1) **5**

[3.1 Structure and organization of Java classes](#_heading=h.28h4qwu) 5

[3.2 Classes developed](#_heading=h.bqtc838db6z3) 6

[3.3 Database](#_heading=h.1b9ikwuz0ure) 8

[**4. ProCare Application**](#_heading=h.34g0dwd) **11**

[4.1 Pets Section](#_heading=h.ppbnqbex42z9) 12

[4.2 Upcoming Tasks Section](#_heading=h.xvir7l) 14

[4.3 Calendar Section](#_heading=h.4h042r0) 15

[4.4 Settings Section](#_heading=h.2nusc19) 21

[4.5 User Section](#_heading=h.319y80a) 23

[4.6 Login Section](#_heading=h.40ew0vw) 25

[4.7 Register Section](#_heading=h.184mhaj) 31

[**5. Usage and scenarios**](#_heading=h.3x8tuzt) **32**

[**6. Improvements & Future Work**](#_heading=h.1kc7wiv) **33**

[6.1 Improvements](#_heading=h.ymfzma) 33

[6.2 Future Work](#_heading=h.2wwbldi) 33

[**7. Conclusion**](#_heading=h.i20o549gw3rp) **33**

[**Bibliography**](#_heading=h.3abhhcj) **34**

# 

# **1. Introduction**

This chapter briefly explains the application and also synthesizes the structure of this document.

## 1.1 Application proposal

The continuous increase in the use of mobile devices gives a lot of food for thought about the creation of applications in order to achieve a stronger connection between users and mobile applications.

In this case, the idea was to create an application dedicated to the care and management of animals, whether it is an ordinary user, who wants to manage and organise the care of his pet, or a zoo employee, whose application helps him to manage the animals' schedules for their care.

With the goal of enhancing communication between application ProCare and clients, this desktop application allows clients to create an account to schedule appointments for their pets such as vaccinations or even set events on a social map.

## 1.2 Document Organization

This document is structured in 7 chapters. **Chapter 1** introduces the context of the project. **Chapter 2** names all the tools used in the development of this project. **Chapter 3** explains the structure and implementation we developed for the application and **Chapter 4** shows the user’s interaction with the application for each section and its layouts. **Chapter 5** covers the application usage for its proper performance. **Chapter 6** mentions future work and improvements that we didn’t have the time to accomplish and finally **Chapter 7** reviews the conclusions drawn from this project.

At the end, we attached all websites links and forums which have helped us to solve bugs and give us an idea of how to implement certain sections.

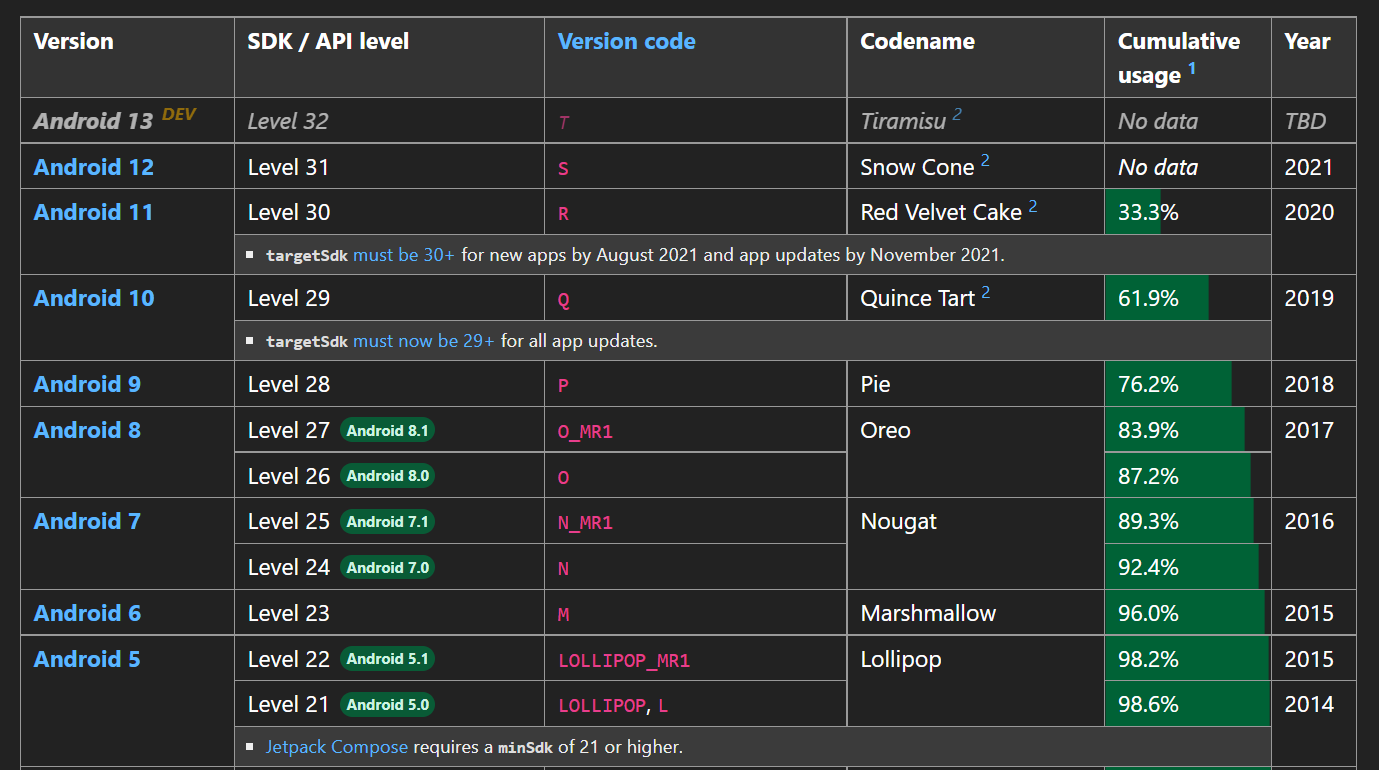
# **2. Workspace and Tools**

This chapter enumerates and describes the tools and environment used for developing the application.

**Android version**

The Android operating system is constantly being updated. In each release, new features can be developed, security measures can be changed, etc. At this moment, the newest release is Android 12.

We decided to use Android 12 to develop our app as it was the latest release at the time, which ensured that our app was up to date with Android standards. Our minimum Android version will be Android 5 as it is the largest and most ambitious release for Android yet.



*Figure 1: Existing Android Versions*

**Android Studio**

This tool was used in this project to develop the entire application. It gives the developer the choice to code in Java or in Kotlin. Because of the previous knowledge of Java that the members of the team possess, this app was written in Java.

**SQLite Database**

Android Studio, it comes with a local SQLite database implementation already built in. We decided to use a local database because querying the database in Cloud can take more time, as there can be *time-out* problems in the responses as well as unavailability of the application's service. We have decided to store our database locally because we are not storing large amounts of data, in case it is needed we will proceed migrating our database to Cloud.

**Github Desktop**

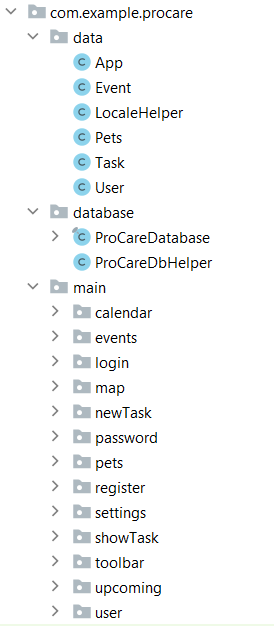
We used this tool to manage the versions of the application and to seamlessly merge each contribution of all the participants of the team, allowing a parallel development.

# **3. Structure and implementation**

This chapter gives a highly detailed explanation of how we structure our application, the functionalities of each java class and also the database created.

## 3.1 Structure and organization of Java classes

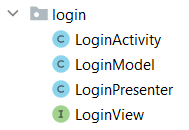
The application is composed of java classes together with XML components linked to its graphic design. Those java classes are grouped by packages to differentiate them by functionality from each other and to maintain a clear and solid structure. We have decided to implement land layouts for most of the java activities to give our application and horizontal and vertical orientations. Moreover, we created icons and personalized png files for visual content.



*Figure 2: Structure and organisation of Java classes*

We based on the MVP architecture for the application to be easily extensible and easy to maintain, we need to define well separated layers. Moreover, it provides a clear and simple structure so that future changes can be made more trivially and less complicated. There are many variations of MVP and everyone can adjust the pattern to their needs, which is why we have structured it as four components in this way [2]:

* **Activity Java class**: Responsible for rendering UI elements.
* **View Interface**: Responsible for loose coupling between view and model.
* **Presenter**: Responsible for view and model interaction.
* **Model**: Responsible for business behavior and state management such as the provider of the data we want to display in the view from the database.

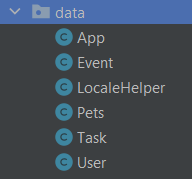


*Figure 3: Structure and organisation of Java classes*

## 3.2 Classes developed

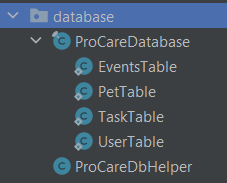
This section will explain each package and its contents:

* **Data**: This package (see *Figure 4*) contains java classes used to store and retrieve values from database tables (Event, Pets, Task, User). Also to obtain application information (App) and for language or translation purposes (LocaleHelper).



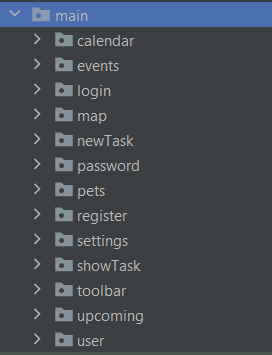
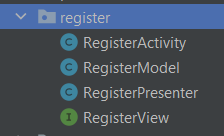
*Figure 4: Structure of data*

* **Database**: This package is based on a helper class to manage database creation and version management, and a java class which contains inner class that defines the table contents

**

*Figure 5: Structure of database*

* **Main**: This package contains all the basic functionalities packets. Within these packages are the classes forming an MVP architecture.

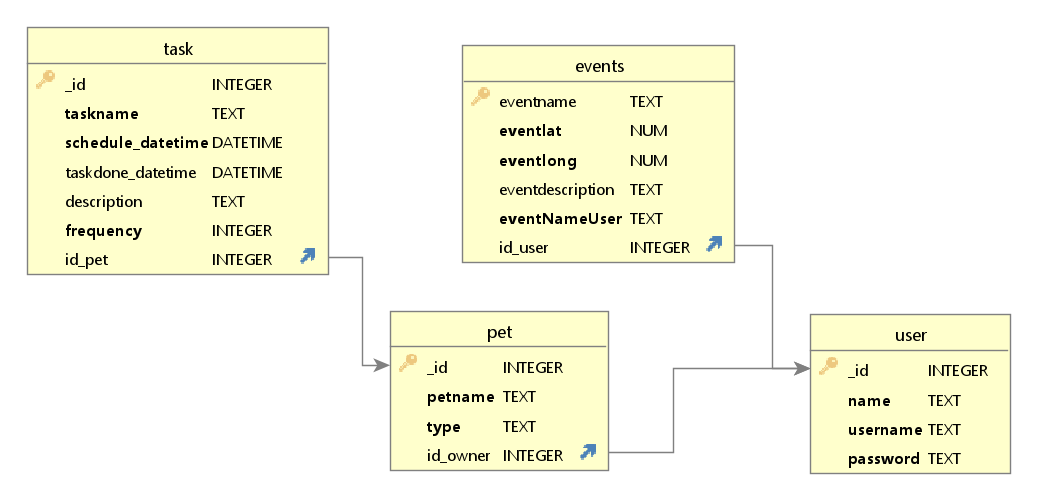


*Figure 6: Structure of main*

* + **Calendar**: This package manages the user's calendar and pending tasks.
  + **Events**: Manages all user event data and displays it on the map if necessary.
  + **Login**: It handles user login and manages the user's main data such as name and password.
  + **Map**: It shows all events in the form of a marker, the user can move freely around the map, and the user's current location is shown on the map.
  + **New Task**: This package creates new tasks and manages them in such a way that the user is notified when it is time for the task.
  + **Password**: Stores the user's password, this package is used to change the password.
  + **Pets**: This is where all information related to pets is stored, and also where all this information is managed for display and modification.
  + **Register**: This package stores all user data, which will later be used by many of the app's functions.
  + **Settings**: Saves user preferences, language, colour, notifications…
  + **Show Task**: Displays all tasks with their information
  + **Toolbar**: Here you manage all the functionalities of the bar below.
  + **Upcoming**: This package manages the main screen, which displays the user's pets.
  + **User**: In this package all user data is controlled and used for different functionalities.

## 3.3 Database

We first proceeded to implement a local database (ProCareDB) composed of four tables.



*Figure 7: Entity Relationship Diagram (ERD) of Database created*

Events Table groups all the events created on the map by each user and contains the following columns:

* **id\_user**: Assigns a unique numerical identifier to each user.
* **eventname**: Attribute used for setting the name of each event created.
* **eventlat**: Attribute used for setting each event latitude.
* **eventlong**: Attribute used for setting each event longitude.
* **eventdescription**: Contains a description of an event.
* **eventNameUser**: Attribute which contains the name of the user who created that specific event on the map.

| **Events** | | |
| --- | --- | --- |
| **Name** | **Type** | **Schema** |
| id\_user | INTEGER | FOREIGN KEY REFERENCES User(\_id) |
| eventname | TEXT | PRIMARY KEY |
| eventlat | NUMERIC | NOT NULL |
| eventlong | NUMERIC | NOT NULL |
| eventdescription | TEXT |  |
| eventNameUser | TEXT | NOT NULL |

*Table 1: Events Table structure*

Pet Table contains a list of the existing pets and it is composed of:

* **\_id**: Assigns an unique numerical identifier to each pet.
* **petname**: Attribute which sets the name of the pet.
* **type**: Differentiates the type of pet, beginning with the common ones (Dog = 0, Cat = 1, Bird = 2, Fish = 3, Turtle = 4, Horse = 5, Other = 6)
* **id\_owner**: Contains the id of the user who created or owns the pet.

| **Pet** | | |
| --- | --- | --- |
| **Name** | **Type** | **Schema** |
| \_id | INTEGER | PRIMARY KEY |
| petname | TEXT | NOT NULL |
| type | TEXT | NOT NULL |
| id\_owner | INTEGER | FOREIGN KEY REFERENCES User(\_id) |

*Table 2: Pet Table structure*

Task Table has the task created for an specific pet and it is made up for:

* **\_id**: Assigns an unique numerical identifier to each task.
* **taskname**: Attribute used for setting the name of each task.
* **schedule\_datetime**: Attribute for storing the date and time of the scheduled task.
* **taskdone\_datetime**: Attribute for storing the date and time when a task is done.
* **description**: Contains a description of a task.
* **frequency**: Sets a frequency for a task (Daily, Weekly, Monthly and Yearly or Today)
* **id\_pet**: Contains the id of the pet to which a task has been assigned.

| **Task** | | |
| --- | --- | --- |
| **Name** | **Type** | **Schema** |
| \_id | INTEGER | PRIMARY KEY |
| taskname | TEXT | NOT NULL |
| schedule\_datetime | DATETIME | NOT NULL |
| taskdone\_datetime | DATETIME |  |
| description | TEXT |  |
| frequency | INTEGER | NOT NULL |
| id\_pet | INTEGER | FOREIGN KEY REFERENCES Pet(\_id) |

*Table 3: Task Table structure*

User Table stores all the users registered and it is made up for:

* **\_id**: Assigns an unique numerical identifier to each user.
* **name**: Attribute which sets the name of the user.
* **username**: Attribute which sets the username of the user.
* **password**: Attribute which sets the password of the user.

| **User** | | |
| --- | --- | --- |
| **Name** | **Type** | **Schema** |
| \_id | INTEGER | PRIMARY KEY |
| name | TEXT | NOT NULL |
| username | TEXT | NOT NULL UNIQUE |
| password | TEXT | NOT NULL |

*Table 4: User Table structure*

# **4. ProCare Application**

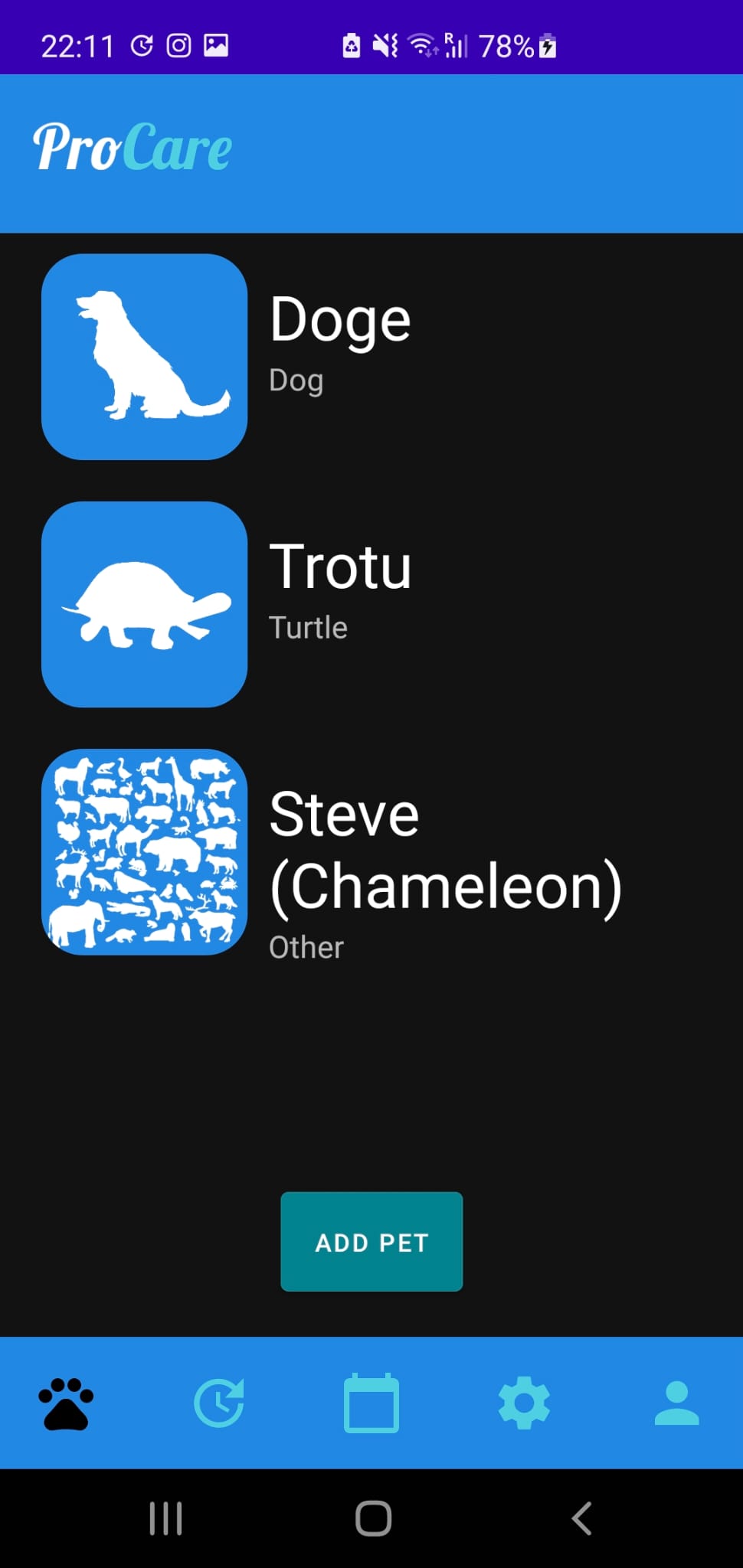
This chapter provides a detailed explanation of each layout of the application and the interaction of the user with it. We have also designed a logo as shown in *Figure 73*.

# 

*Figure 73:* *Application Logo*

The ProCare application has been structured through the use of Activities with its own layout attached. Its main composition consists of 5 sections managed and implemented by ToolBarManagement java class, which can be accessed by clicking on the bottom five bar icon:

* **Pets Section:** Entry point of the application. Shows some starting information of the analysis performed.
* **Upcoming Tasks Section:** Lists all the applications installed on the device and when one of them is selected displays information about it.
* **Calendar Section:** Fragment in charge of carrying out the signature analysis.
* **Settings Section:** Fragment in charge of carrying out the permission analysis.
* **User Section:** Fragment in charge of carrying out the log analysis.



*Figure 8: Section bar organisation*

## 4.1 Pets Section

This section is the visual entry point of the application, it displays all the pets a user can own as it is shown in *Figure 9*. Once the user clicks on a specific pet, it shows the pet´s information which is based on the task assigned to that pet. Also we implemented buttons Edit and Delete in case the user wants to modify a pet´s name or change the type of animal or directly remove it (*Figure 10*).

## 

*Figure 9: Pets Section Figure 10: Pet Information*

## 4.2 Upcoming Tasks Section

In the Upcoming Tasks section (*Figure 11*), the interface is divided into three parts: today's tasks, tomorrow's tasks and upcoming tasks. The tasks will be displayed with the scheduled date, the name of the task and the animal that will perform the task.

If you click on a task you can see more detailed information, in this new interface there are buttons to modify, delete and mark as done.(*Figure 12*)

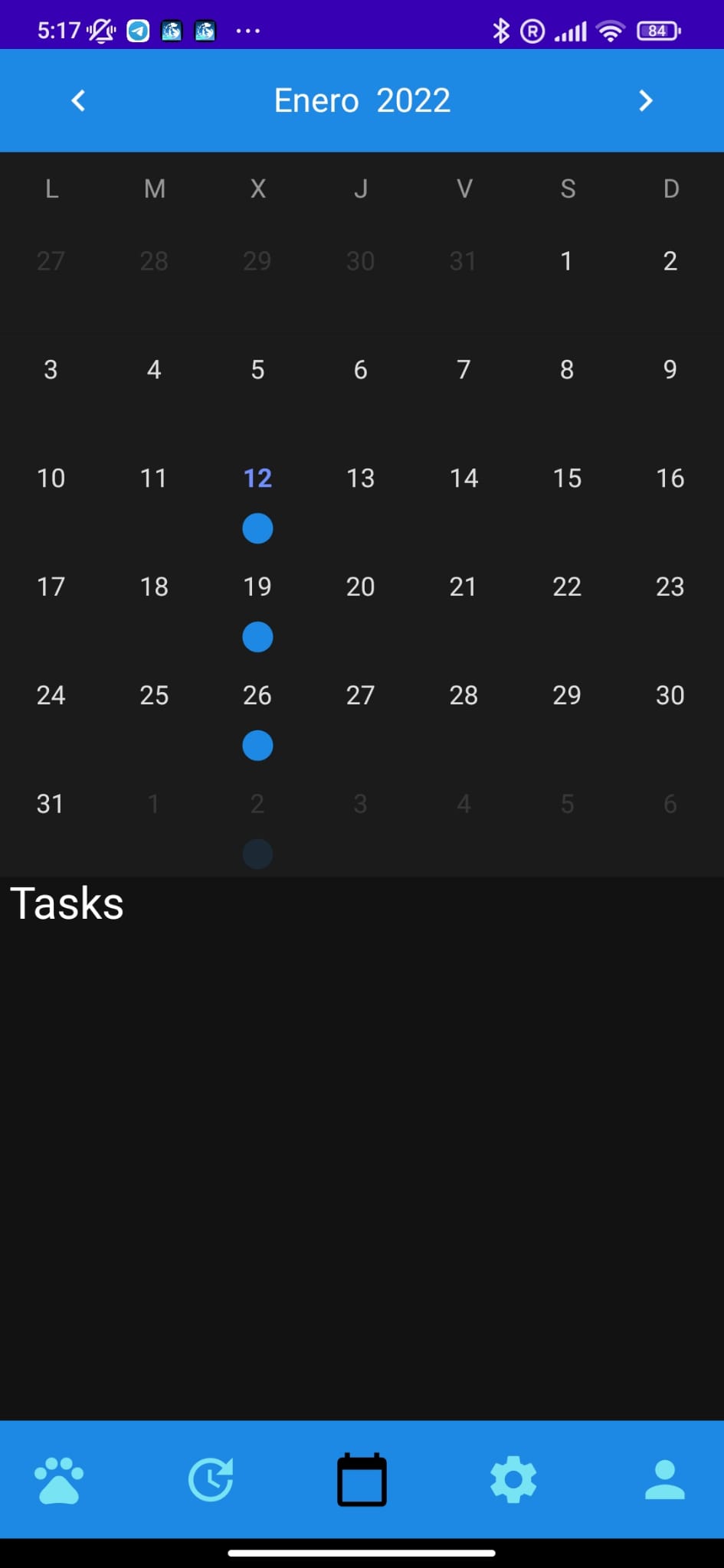
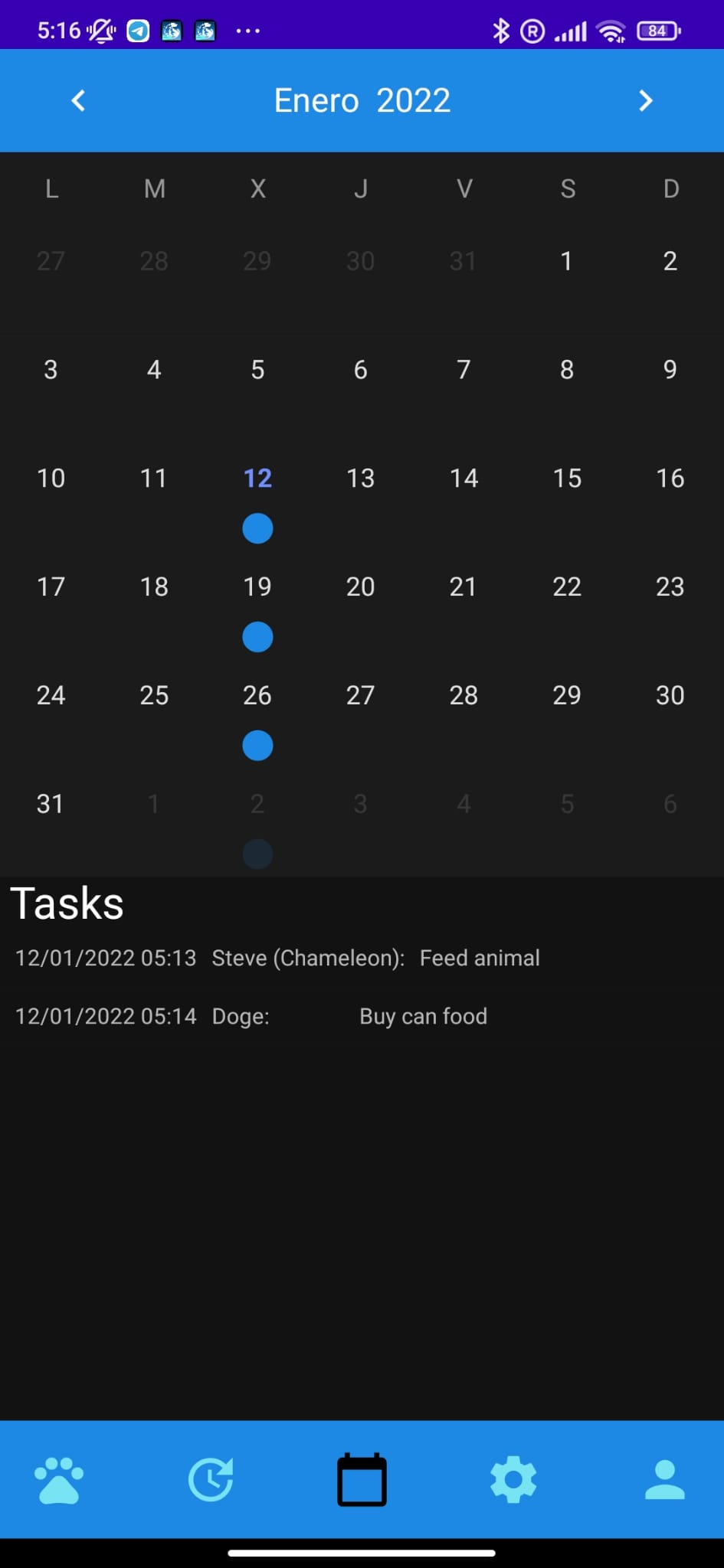
## 

*Figure 11:* *Upcoming Tasks Section Figure 12:* *Info Task*

## 4.3 Calendar Section

In this part of the app a large calendar appears, in which are marked with a colour the days that there is a pending task, if you click on that day, the scheduled tasks will appear just below the calendar. (*Figure 13)*

On that day, the scheduled tasks will appear just below the calendar. This screen is a general and different way to see the tasks that the user has programmed in the "Upcoming Tasks Section".



|  |
| --- |

*Figure 13:* Calendar Section

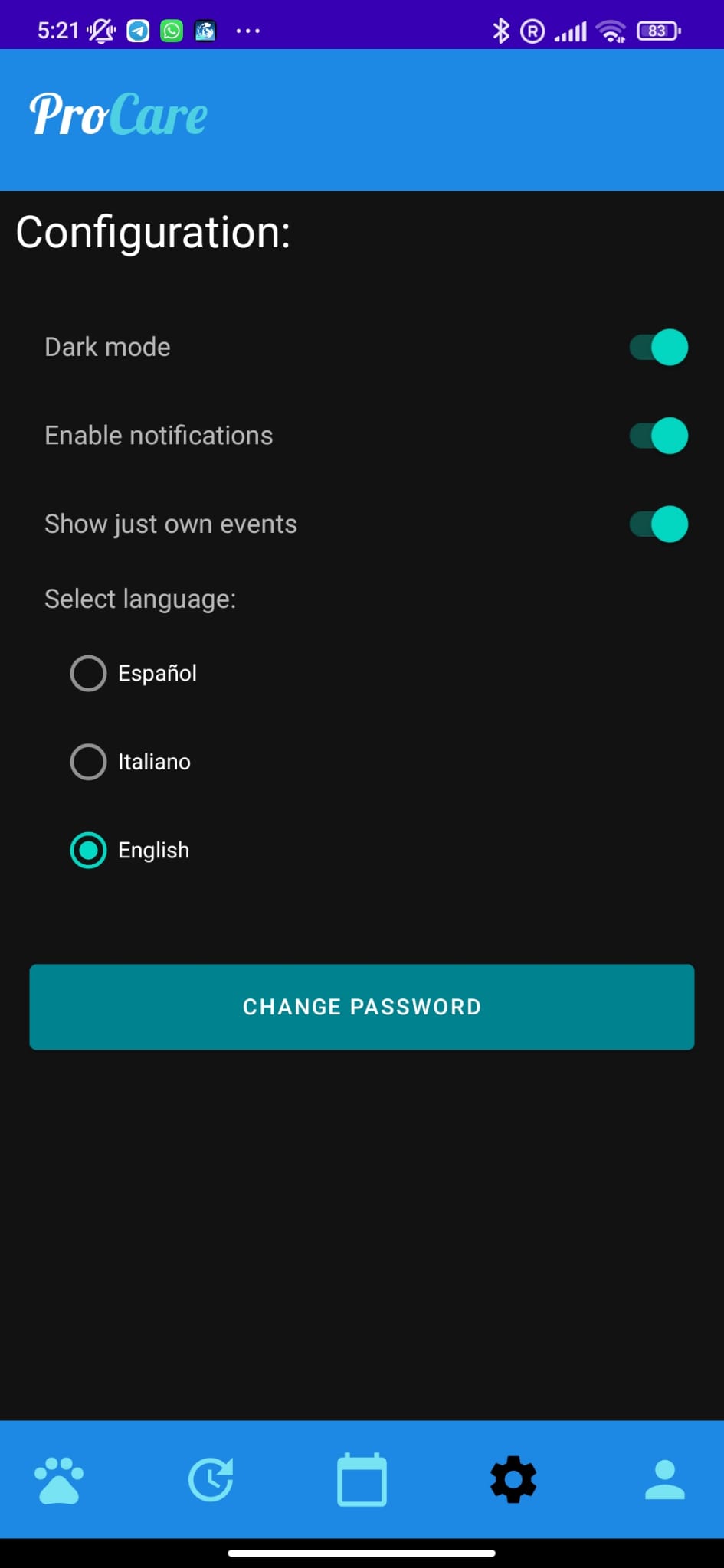
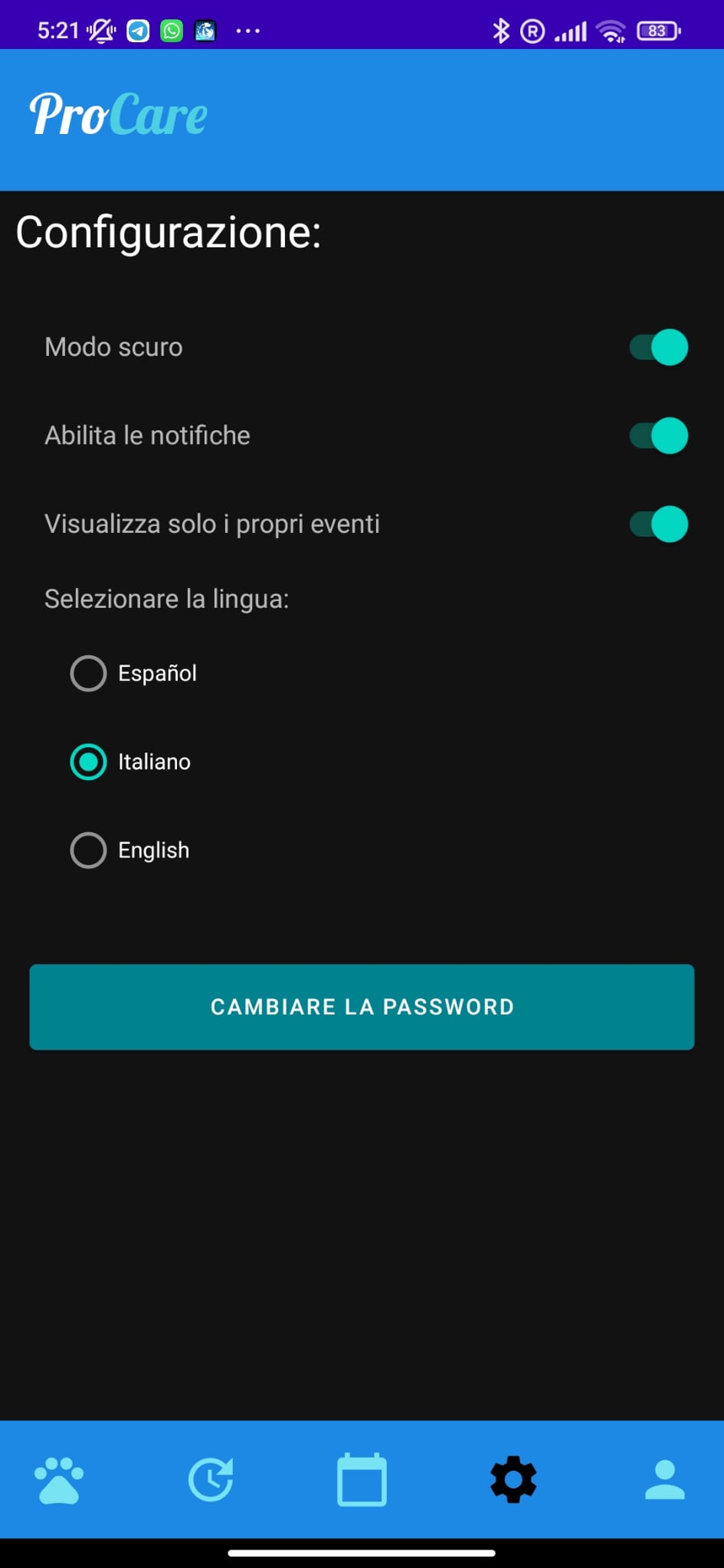
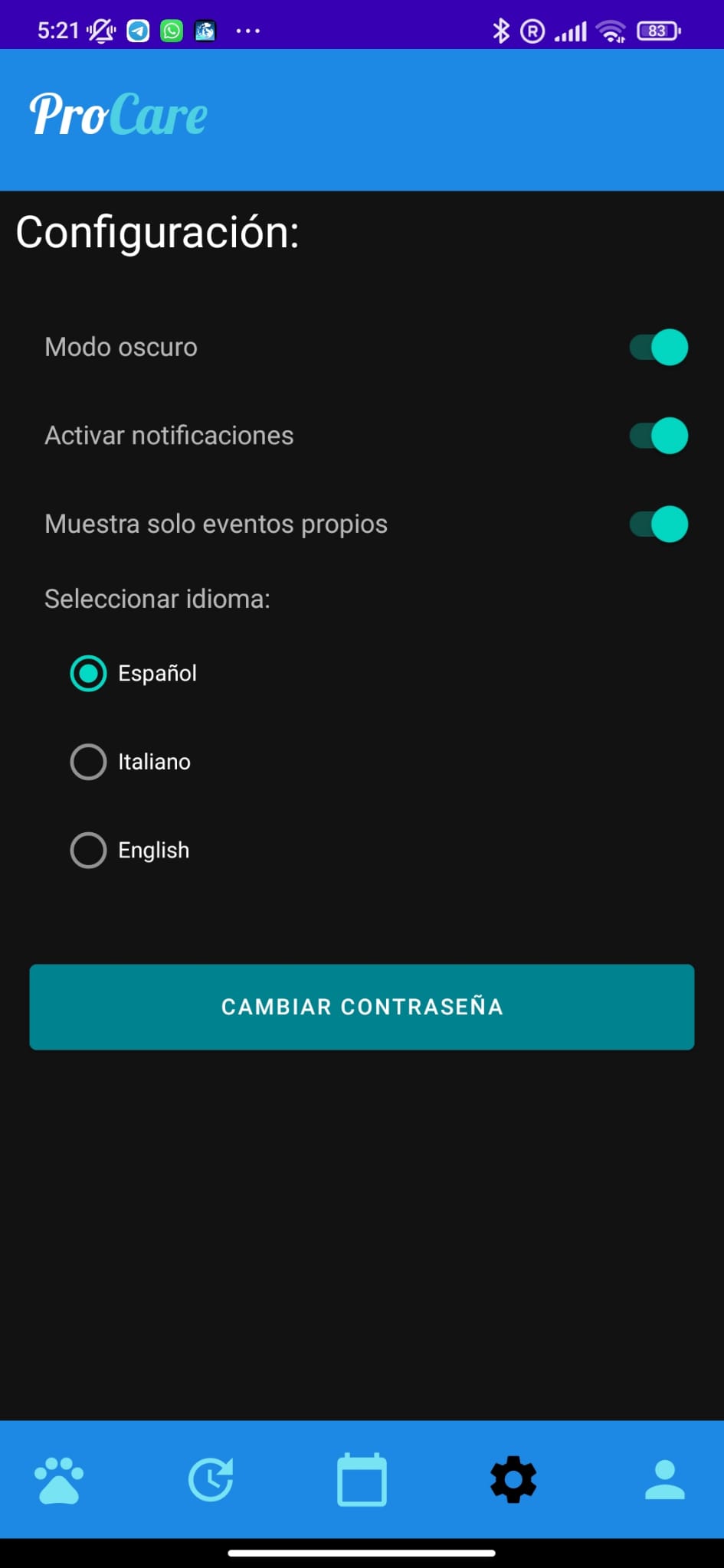
## 4.4 Settings Section

In this part we find the settings, first we can see three switches, the first one gives the user the option to enable or disable the dark mode; the second switch enables the notifications and the last one allows the user to see on the map all the events or only the ones created by him. These events are shown on the map that I will explain below. (*Figure 14)*

## 

*Figure 14:*Settings Section (Dark Mode)

In the second part of the interface you can see three language options (Spanish, Italian and English), the user can choose the one he wants and it will be changed automatically.(*Figure 15)*

*Figure 15:*Settings Section

Finally there is a button to change the password, when you press it you are directed to a window where you have to put your old password and the new one, once you press the "confirm" button the password will be changed. (*Figure 16)*

## 

## 

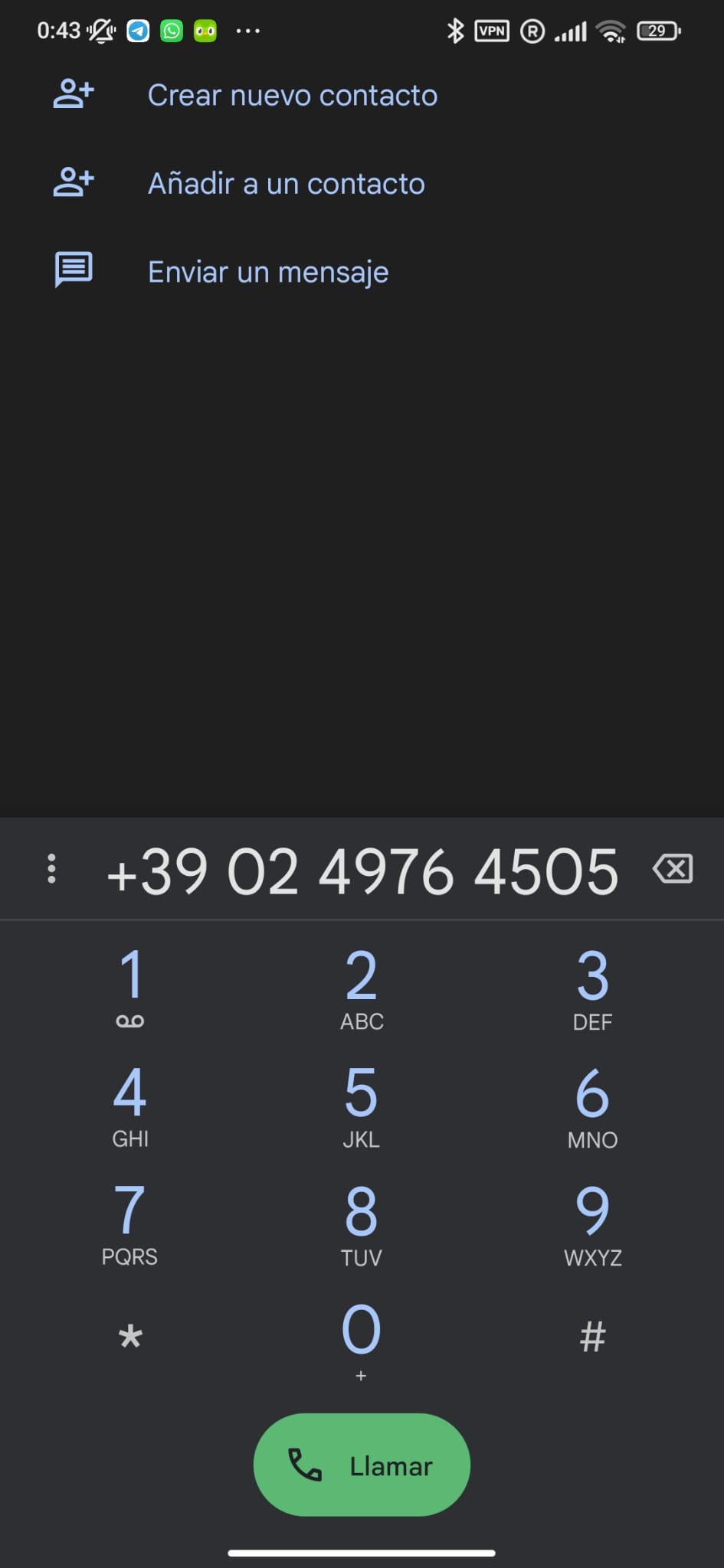
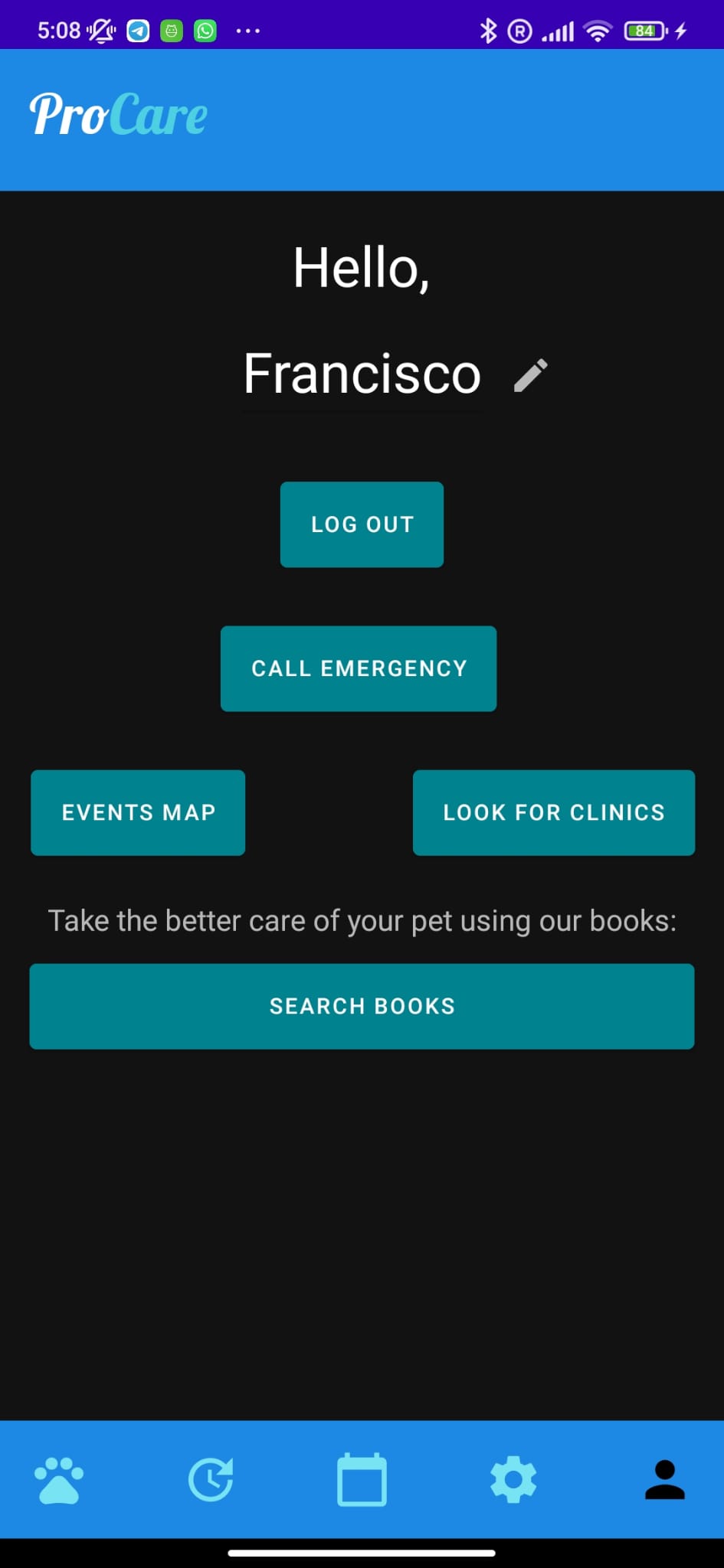
## 

*Figure 16:*  *Change password*

## 

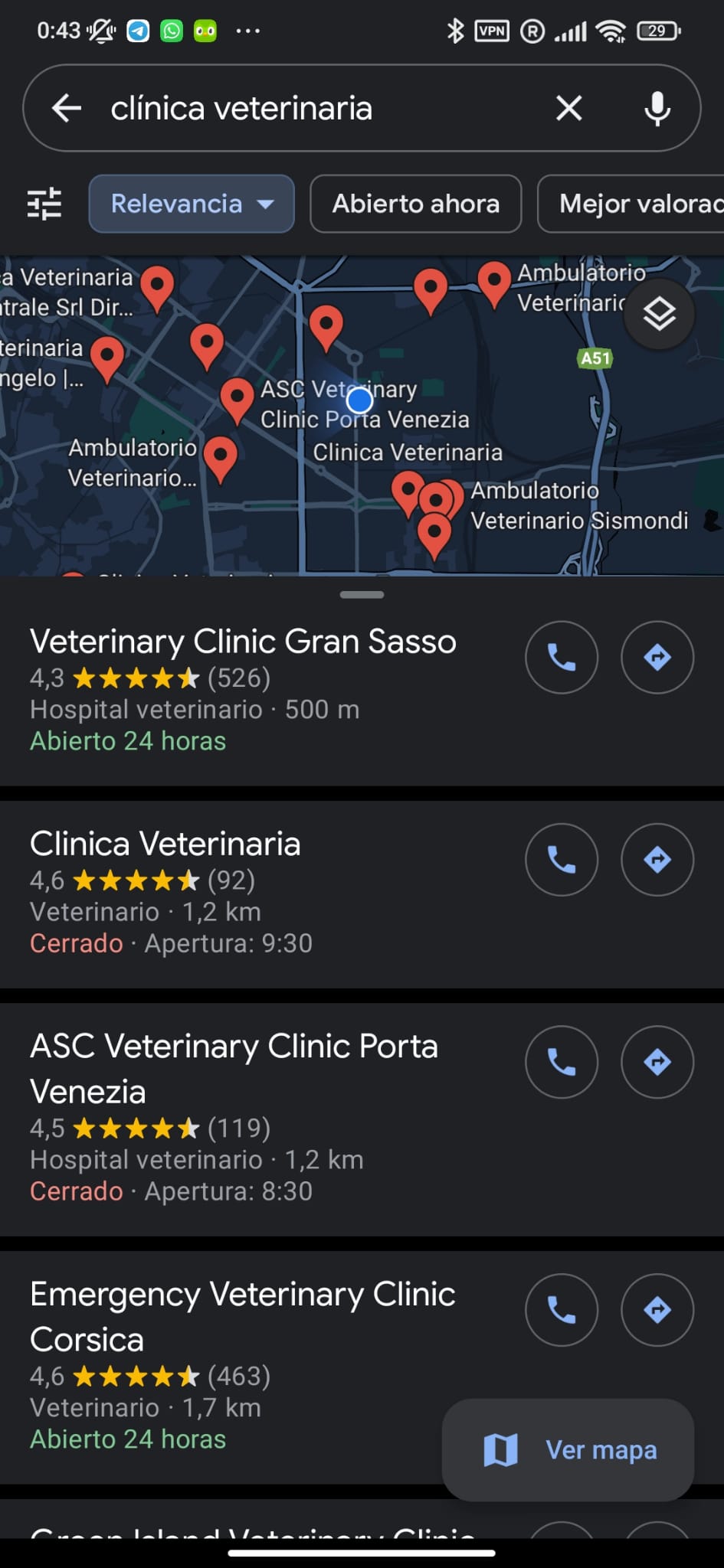
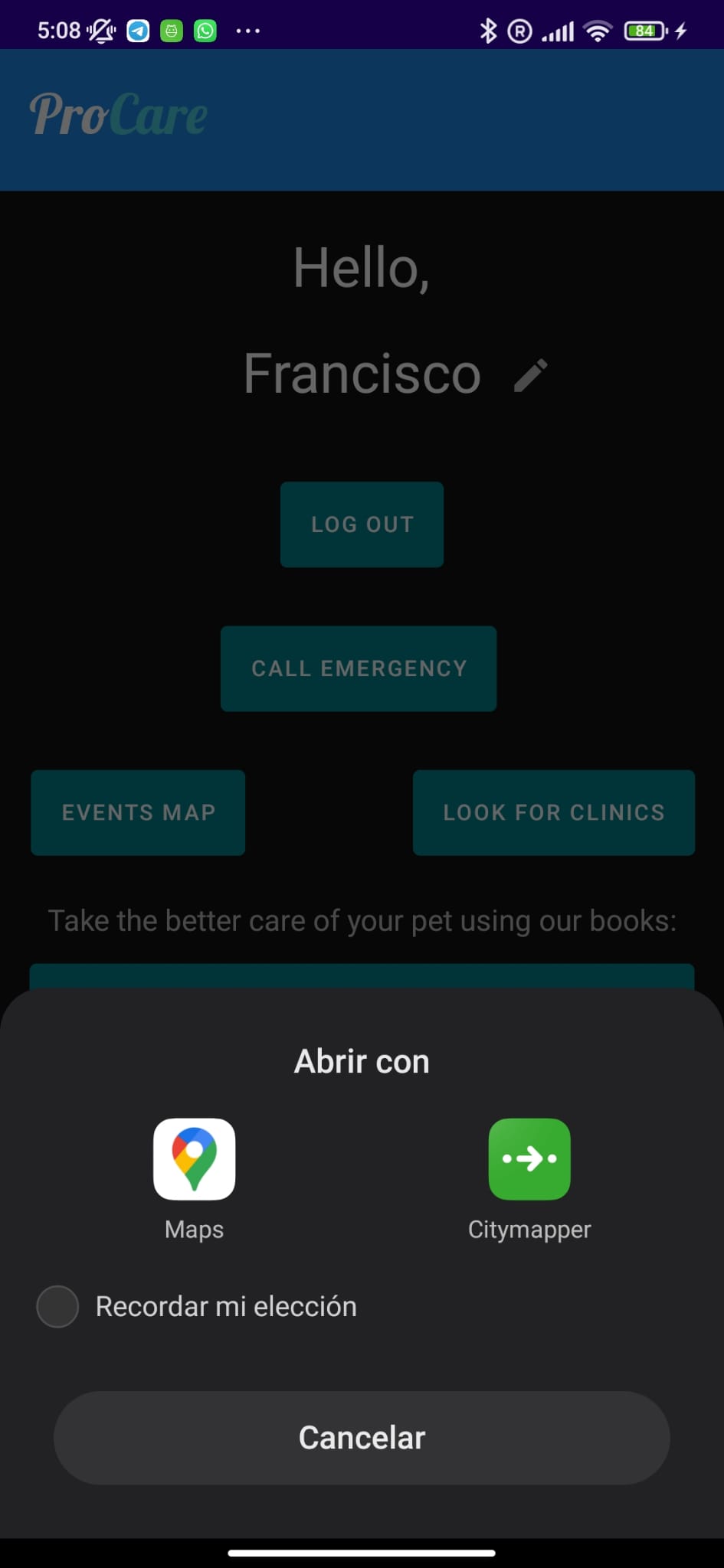
## 4.5 User Section

In this part we can find at the top, just below the app logo, how the app greets the user according to the language that is set in the settings. The user can change the name if desired. Just below the name we find the "Log out" button to change the user; the next button is Call Emergency", which when pressed dials the phone number of an emergency veterinary in Milan (*Figure 17).*

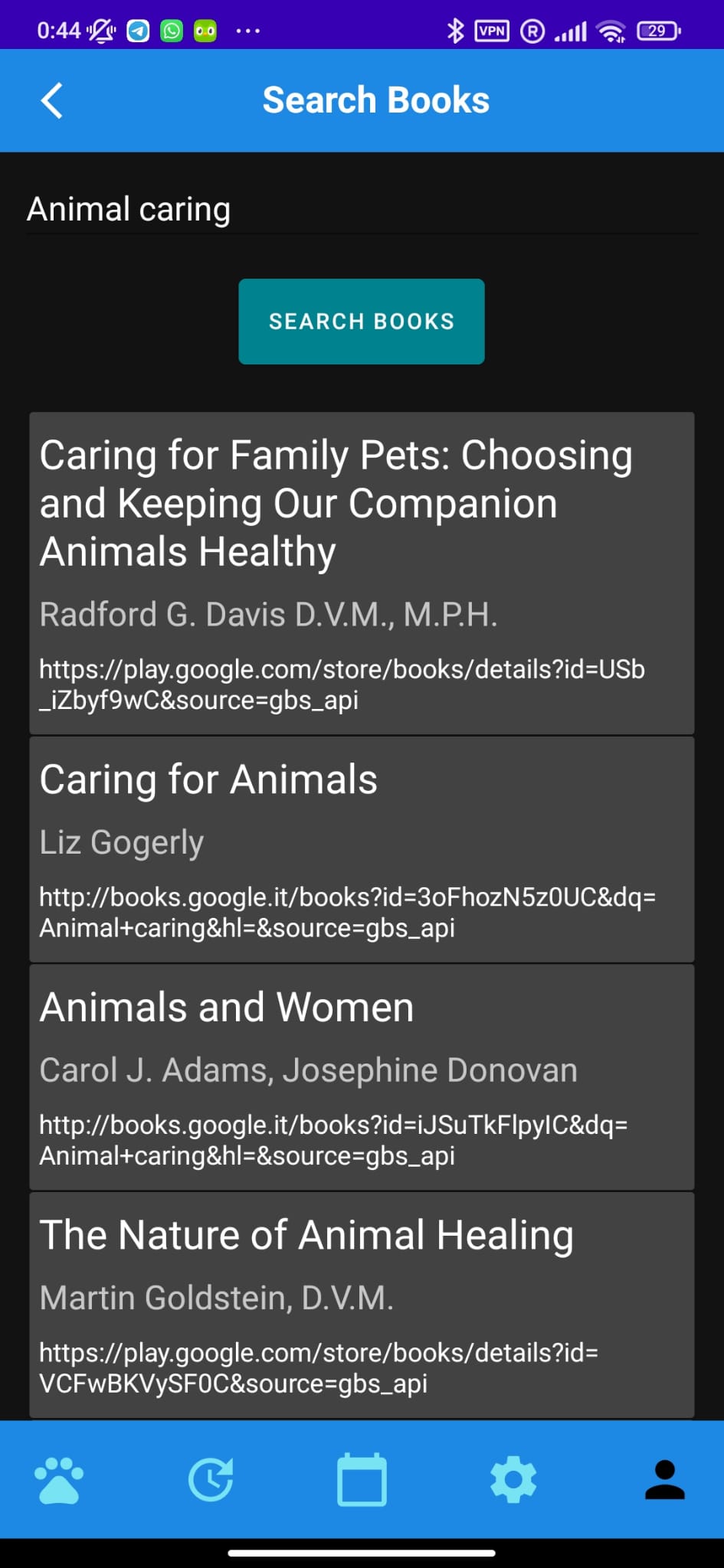


*Figure 17:*  *User Section -Call Emergency Option*

The "Look for Clinics" button searches google Maps for the nearest veterinary clinics (*Figure 18)*. Just to the left is the "See Map" button which when clicked shows a map inside the app, I will talk about this functionality later. Finally there is the "Search Books" button which searches for help books on the text you type in the search box, by default it searches for "animal caring". (*Figure 19)*



*Figure 18:*  *Look for Clinics Option*



*Figure 19:*  *Search Books Option*

The map that is executed when you click "Events Map" is a map inside the app, it is not necessary to go to google maps, this map shows the events created by all users (or only shows your own events, if you have configured it in the settings). In this interface we can see at the top right a button for the map to focus on our current location (this functionality is done by default when you open the map). Inside the map we can find different events, in which when clicking on it, the information about the event is shown, and if the event is the user's own, he can delete it, if he wants.

Just above the navigation bar of the app we see the button "Add Event", which enables a space to create an event, the user must put a name and a description to inform what this marker is about. The markers created by the user will be added in the location where they are at that moment.

## 

*Figure 20:* *Maps functionalities*

|  |  |
| --- | --- |
|  |  |

## 4.6 Login Section

This section displays a login form for the users in order to access the application functionalities. The use of SharedPreferences java interface predefined helped us to make the application save the user's session so that when he/she proceeds to register and accesses the application through the login and then closes the application and reopens it, he/she will be taken directly to the main page in this case PetsActivity. Also we decided to keep the username once the user reopens the application to facilitate the user's access.

## 

*Figure 21: Log Analyzer Fragment*

## 4.7 Register Section

This section displays a registration form for the new users in order to create a new account. It will store the user´s name, username or nickname and password. Once the user finishes with the registration form the application will redirect to the Login section.

## 

*Figure 22:* *Previous Results Fragment*

# **5. Usage and scenarios**

Our idea to make the app more complex was to implement a map without the need to use external apps. so we created the activity "add events", but when we implemented it we had several problems, most of them we managed to fix, but there is one that we were not able to solve. we managed to fix them, but there is one that we are not able to solve. The problem is when a person installs the app for the first time, when he/she gets into the map it will ask for permission to access the location (which is necessary for it to work). Once permission is granted, the map loads, but their location does not appear and they cannot see markers on the map, which means that they can't see the map. markers on the map, what the user has to do is change section and get back on the map, and it works normally. This error only appears the first time, but even then we have not been able to fix it.

If you want to test the operation of the app from android studio you must follow some steps so that the map implemented in the app works correctly. the app works correctly, the quickest and easiest way is the following:

1: Import project to android studio

2: Create a new project as follows: File>New>New Project>Google Maps Activity

3: Access to the file "google\_maps\_api.xml", then access to the first link

4: follow the 3 steps that will appear

5: once the API has been created, search for "credentials" in the search bar of Google Cloud Platform

6: when you have accessed the credentials click on the name of the api you have created

7: once logged in modify the package name to: com.example.procare

8: copy the API key from the top right hand corner

9: go to the ProCare project manifest and copy the following:

<application

<meta-data

android:name="com.google.android.geo.API\_KEY"

android:value="APIKEY" />

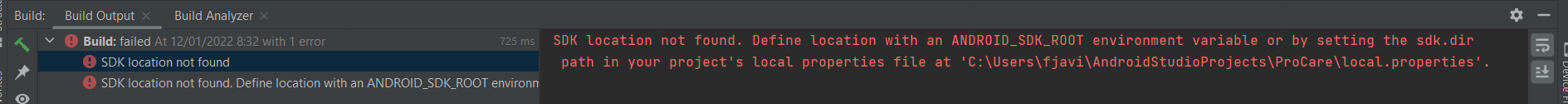
</application>

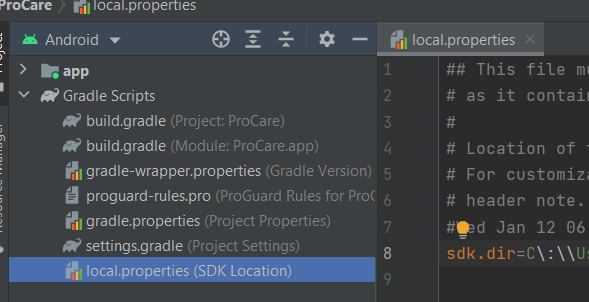
10: Where it says APIKEY paste if API key

11: Go to the google\_maps\_api.xml file of the ProCare project (it is located in the values folder) and change the content of tools:ignore="ExtraTranslation">AIzaSyBs8UMwHF6m98YORtBHE7l6NcIHWH0\_s\_o</string> to your API KEY

After you have done all these steps when you run the app the map should appear correctly.

<https://developers.google.com/maps/documentation/android-sdk/get-api-key?hl=es>





In case of this notification error, proceed to change the user colored in red in the file “local.properties” by your own user directory.

sdk.dir=C\:\\Users\\fjavi\\AppData\\Local\\Android\\Sdk

# **6. Improvements & Future Work**

## 6.1 Improvements

This section reflects various improvements that we have not had enough time to carry out despite having the knowledge to be able to implement them.

* Be able to enable/disable notifications such as the pending task schedule by the user in the settings section.
* Implement a method in the register section in order to force the user to create a more secure password such as a minimum number of characters, requiring a symbol and even capital letters.
* One of the biggest issues we faced during the development of the application consisted of setting the exact localization once the user opens, for the first time, the events map. We did not achieve it, so it just works correctly when the user opens for a second time the event map.

## 6.2 Future Work

This work has been carried out over approximately six months, which is the expected time for the development of this project. However, the work done here can be improved by extending its utilities as indicated below.

* When a user creates a task, and at the moment the "Pending Task" pop-up notification appears, knowing that the application is closed, the application can be accessed through the notification by simply clicking on the pop-up.
* In the event map we thought we could improve the user experience by creating an event without the need to be located there, as in this version the event is created at the user's current location. When adding a new event, let's add an address field so that the user can enter the location of the desired event.
* Furthermore, we could implement that a user can select an event on the map and there is a button, which indicates the route or directions on how to get there.
* As we experienced several issues with the implementation of the map we could create a personalised one in order to avoid asking every user to generate their own API Google key.

# **7. Conclusion**

We believe that our application can fill some of the crucial needs that every user of a mobile phone requires, such as reminders or notifications to care for your pet. There are certain illnesses that can affect the memory and thanks to ProCare, the user will be able to remember with ease

We also think that the application will encourage users to actively take care of their pets and that they will learn how to organise themselves.

# **Bibliography**

[1] Existing *Android Versions*. Retrieved from <https://apilevels.com/>

[2] The MVP (Model View Presenter) architecture. Retrieved from <https://antonioleiva.com/mvp-android/>

[3] We have taken as a reference and guide the official site specialised in the design of applications for the Android market. Retrieved from <https://developer.android.com/>

[4] We took the idea of the implementation of the calendar of this github user. Retrieved from <https://github.com/Applandeo/Material-Calendar-View>

[5] We followed the steps of this website to develop the translation of the app. Retrieved from <https://www.geeksforgeeks.org/how-to-change-the-whole-app-language-in-android-programmatically/#:~:text=Go%20to%20app%20%3E%20res%20%3E%20values,from%20the%20drop%2Ddown%20list.>

[6] It was really helpful consulting the blog for any kind of debug error. Retrieved from <https://es.stackoverflow.com/>

[7]. Usage API key for the Maps SDK for Android. Retrieved from <https://developers.google.com/maps/documentation/android-sdk/get-api-key?hl=es>